5-2023

An Examination of the Effects of the Cost Volume on Vision Transformer-Based Optical Flow Models

Andrew Thompson
art1933@rit.edu

An Examination of the Effects of the Cost Volume on Vision Transformer-Based Optical Flow Models

Andrew Thompson

An Examination of the Effects of the Cost Volume on Vision Transformer-Based Optical Flow Models

Andrew Thompson

May 2023

A Thesis Submitted
in Partial Fulfillment
of the Requirements for the Degree of
Master of Science
in
Computer Engineering

RIT | Kate Gleason College of Engineering

Department of Computer Engineering

An Examination of the Effects of the Cost Volume on Vision Transformer-Based Optical Flow Models

Andrew Thompson

Committee Approval:

Dr. Dongfang Liu *Advisor* Date
Department of Computer Engineering

Dr. Andreas Savakis Date
Department of Computer Engineering

Dr. Cory Merkel Date
Department of Computer Engineering

Acknowledgments

I would first like to extend my gratitude to Dr. Dongfang Liu, my primary advisor, who has guided and supported my work along the way. I am additionally thankful for the support of Dr. Andreas Savakis and Dr. Cory Merkel in participating in my thesis committee. I would also like to acknowledge Research Computing at the Rochester Institute of Technology for providing computational resources and support that have contributed to the research results reported in this publication. And finally, I would like to thank my family, friends, and loved ones for their everlasting support, which has been instrumental in all I have accomplished.

To all those who have supported, pushed for, and believed in my success along the way.

Abstract

The optical flow task is an active research domain in the machine learning community with numerous downstream applications such as autonomous driving and action recognition. Convolutional neural networks have often been the basis of design for these models. However, given the recent popularity of self-attention architectures, many optical flow models now utilize a vision transformer backbone. Despite the differing backbone architectures used, consistencies in model design, especially regarding auxiliary operations, have been observed. Perhaps most apparent is the calculation and use of the cost volume. While prior works have well documented the effects of the cost volume on models with a convolutional neural network backbone, similar research does not exist for optical flow models based on other architectures, such as the vision transformer. Naturally, a research question arises: what are the effects of utilizing a cost volume in vision transformer-based optical flow models? In this thesis, a series of experiments examine the impact of the cost volume on training time, model accuracy, average inference time, and model size. The observed results show that cost volume use increases the model size and training time while improving model accuracy and reducing average inference time. These results differ from those regarding the effects of the cost volume on convolutional neural network-based models. With the results presented, researchers can now adequately consider the potential benefits and drawbacks of cost volume use in vision transformer-based optical flow models.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

The optical flow problem is a computer vision task to characterize the apparent motion within a scene from the point of view of a particular observer. In this sense, optical flow differs from the well-known concept of motion fields which represent the true motion within a scene. Given sequential, two-dimensional (2D) input images, the inferences provided by an optical flow model provide insights regarding the three-dimensional (3D) motion of the scene as depicted by the images provided. The representation of the inferred apparent motion is a series of 2D displacement vectors resulting from object movement within the scene, camera movement, or a combination of both.

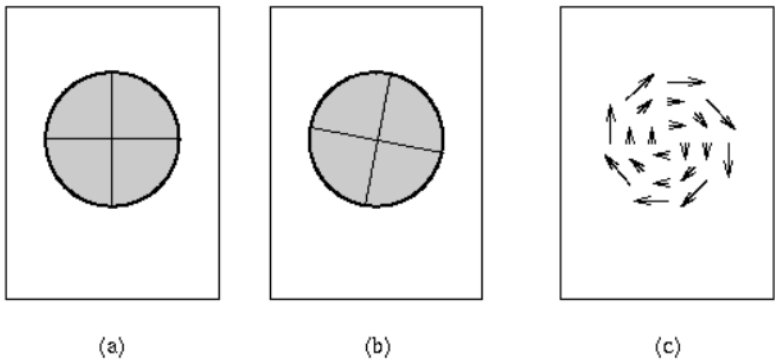

Figure 1.1: An example of the optical flow for a particular scene: (a) scene at time $t1$, (b) scene at time $t2$, (c) associated optical flow. Image source: [1].

The optical flow problem takes two forms in computer vision: sparse or dense. In sparse optical flow, the calculation of displacement vectors is only for a subset of the entire scene as defined by the input images. This subset may be detected objects within the frame, detected corners, or the result of any other pre-processing technique to reduce the number of pixels for which an optical flow model generates a hypothesis. Dense optical flow, however, calculates the apparent displacement of every pixel within a scene. While each variation has its uses, benefits, and drawbacks, only dense optical flow is examined in this document. All references within this document to "optical flow" are analogous to "dense optical flow" and operate with the definition as the apparent motion of all pixels within a scene established by two sequential, 2D images.

The rise of deep learning in the computer vision domain has rapidly accelerated the development of new optical flow models. As with many other computer vision tasks, there is a strong correlation between advancements in addressing the optical flow problem and overarching trends within the machine learning community. For example, many held convolutional neural networks (CNNs) as the premier deep learning architecture for many years. During this time, optical flow models with a CNN backbone predominately held the title of state-of-the-art [7, 12, 13, 14, 15]. Over time, recurrent architectures began to gain notoriety in the machine learning community for their ability to characterize sequential data. Given the temporal nature of the optical flow task, recurrent-based optical flow models garnered significant interest from researchers. It was not long until these models with recurrent-based backbones held state-of-the-art status in the optical flow domain [8, 16]. Most recently, vision transformer architectures have consumed much of the research focus across the computer vision domain due to their ability to effectively characterize spatiotemporal data. At present time, optical flow models with a vision transformer backbone are largely recognized as state-of-the-art [9, 17, 18].

Regardless of the variations in the underlying backbones of models used to address the optical flow problem, some consistency between models exists regarding auxiliary operations. One example is the computation and use of a cost volume as a pre-processing operation for model backbones [7, 8, 9]. While a cost volume may take various forms in the machine learning and computer vision domain, this document operates with the term "cost volume" referring to the similarity matching costs from the explicit correlation of raw pixels, or the extracted features thereof, from two subsequent frames of a scene for which the calculation of optical flow is performed.

Despite the extensive use of the cost volume in various optical flow models, limited research exists in examining the direct effects of the cost volume on overall model performance. To the author's knowledge, the few studies that do exist have only examined the effects of the cost volume on optical flow models with CNN-based backbones [7]. Thus, a research problem naturally arises. How does utilizing a cost volume impact vision transformer-based models in addressing the optical flow problem? Only vision transformer-based models are examined throughout this document as these are of the most significant research interest in the optical flow community.

1.2 Motivation

Researchers have long realized the benefits of optical flow in numerous applications, such as mobile robotics, biomedical, and virtual reality [19, 20, 21, 22, 23, 24]. Optical flow has also assisted various downstream computer vision tasks. For example, Huang et al. [25] use optical flow in video frame interpolation, while Zhu et al. [26] detail video recognition applications. For these reasons, the domain has attracted significant research interest. However, optical flow and other spatiotemporal tasks present inherent challenges that have hindered the effectiveness of addressing the applications mentioned. Challenges include large object displacements, occlusions, and variable lighting [27]. Numerous techniques have been proposed to address these challenges.

Utilization of the cost volume in deep learning-based models is one technique that has found prior success and continued use today. As shown by Dosovitskiy et al. [7], a cost volume can assist CNN-based optical flow models when it is used as a pre-processing operation for the CNN backbone. Rather then use uncorrelated data, the cost volume allows model backbones to operate with pre-processed information regarding the per-pixel similarity between the input images. The pre-processing provides preliminary insights regarding the spatiotemporal relationships of the input data. In a sense, the cost volume provides a head start to the model backbone in forming an optical flow inference. While Dosovitskiy et al. [7] detail some drawbacks of cost volume use, the community has largely regarded its use as net beneficial.

However, the success of the cost volume in CNN-based optical flow models should not be assumed for models with vision transformer backbones as these architectures differ drastically. Transformer architectures, including vision transformers, are celebrated for their inherent effectiveness in learning the spatiotemporal relationships between input data [5]. For this reason, questions naturally arise regarding the benefits provided and the trade-offs required for incorporating the cost volume as a pre-processing operation in vision transformer-based optical flow models. The effects of the cost volume on the time required to train a model, the accuracy of model inferences, model inference time, and model size are all unknown.

Given the downstream applicability of optical flow inferences, understanding the impacts on the metrics mentioned is critical for developing models to suit specific application needs as various applications may prioritize different metrics. For example, in the case of biomedical applications, the accuracy of inferences may be the most critical metric of those mentioned. In real-time applications, such as virtual reality, model inference time may be the most significant metric. Given resource-constrained applications, including for both the training and execution of models, metrics such as the time required to train a model and model size may take precedence.

1.3 Contributions

With the research problem and motivation presented, the following research questions arise regarding the utilization of a cost volume as a pre-processing operation in vision transformer-based optical flow models. The thorough analysis of these questions is the focus of this document and the primary contributions to the optical flow community.

- How does the cost volume affect model training time and convergence?
- How does the cost volume affect model accuracy?
- How does the cost volume affect model inference time?
- How does the cost volume affect model size in terms of the number of learnable parameters?

By answering these specific questions presented, full consideration of the multiple aspects of the effects of the cost volume can be given. With answers to these questions, those designing optical flow models with vision transformer-based backbones can make more informed decisions regarding the use of the cost volume in consideration to each unique application.

1.4 Document Structure

This chapter concludes with a brief outline of the remaining structure of this document. Chapter 2 provides a review of related materials and background information. Chapter 3 outlines the exact method used to answer the research questions specified. Chapter 4 discusses specific implementation details of the method outlined. Additionally, Chapter 4 reports the observed results of the method described. Chapter 5 provides a discussion of the results. This document finishes with concluding remarks in Chapter 6.

Chapter 2

Background

2.1 Convolutional Neural Network

CNNs are among the most recognizable architectures in the machine learning and computer vision domains. These architectures are a derivative of the more general artificial neural network (ANN) and often operate on visual data. CNNs are often identifiable by their stacks of convolution, pooling, and fully connected layers. These architectures have found success in various computer vision and machine learning tasks such as object detection and classification, segmentation, optical flow, and other pattern recognition tasks.

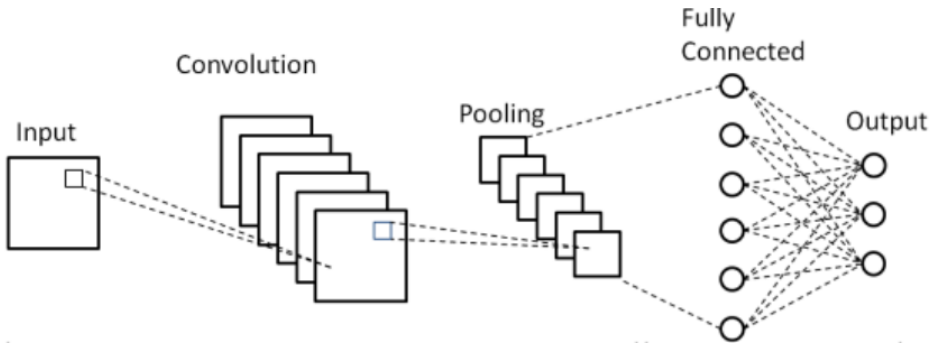

Figure 2.1: An example of a typical CNN architecture. Image source: [2].

The primary function of convolution layers within the CNN architecture is to refine data through the extraction of significant features. To do such, several filters known as kernels are utilized. The representation of each kernel is a series of learnable

weights refined through a traditional ANN training sequence with backpropagation. Following a sliding window technique, the dot products between a kernel and receptive field of the input data are calculated and provided to an activation function. The result is a new feature map utilized by subsequent layers of the CNN architecture. The spatial dimensionality of the resulting feature map is controlled by the kernel size, padding of the layer input data, and sliding window stride. The number of kernels within the convolution layer affects the depth of the resulting feature map. Each kernel is responsible for learning and extracting a particular feature of the layer input data.

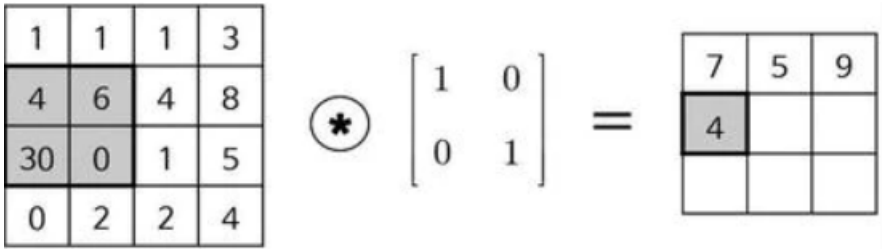

Figure 2.2: An example of the convolution layer within a CNN. The left of the figure represents the input data to the convolution layer. The highlighted section is the receptive field that a kernel, shown in the middle of the figure, operates on. The right of the figure details the resulting feature map with the highlighted field corresponding to the extracted feature associated with the highlighted receptive field. Image source: [3].

The pooling layers within a CNN are non-learnable layers used to reduce the spatial dimensionality of data. A pooling region, in sliding window format, compresses a feature map to generate a downsampled version. Various techniques are used to compress the data. In the often-used max pooling technique, only the maximum value within the pooling region is kept for the reduced resolution feature map. For the mean pooling technique, the resolution is reduced by calculating an average of the values within the pooling region. A combination of the pooling region size, padding of the layer input data, and sliding window stride affect the degree to which the spatial dimensionality of data is reduced. The depth of the data remains unchanged from the pooling operations.

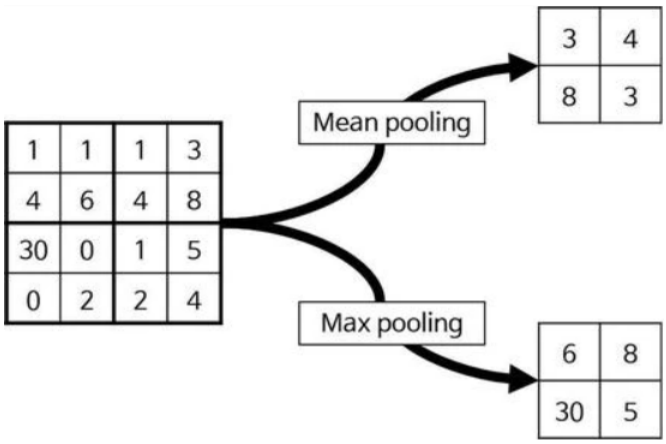

Figure 2.3: An example of the pooling layer within a CNN. Mean pooling and maximum pooling are shown. The left of the image details the layer input. The darker lines of the input distinguish the pooling fields used in generating the downsampled feature maps shown to the right of the figure. Image source: [3].

A fully connected layer within a CNN resembles the arrangement of neurons found in traditional ANNs. Every neuron is directly connected to each previous layer output and subsequent layer input. The purpose of the fully connected layers within a CNN is to correlate the features extracted from the layers prior. Utilizing these correlations, a CNN-based model can form a hypothesis for downstream tasks such as classification, segmentation, and optical flow. Often, a fully connected layer is the last layer of a CNN architecture.

While CNNs can address numerous computer vision tasks, various limitations exist in comparison to traditional ANNs. Some of these limitations include the requirement of large amounts of training data, a tendency to overfit, and a large model size [28]. Various CNN-based architectures have been proposed to address these issues. Some of the most popular include AlexNet [29], VGGNet [30], and ResNet [31], which have been used as the basis of numerous models for various downstream tasks. However, as Albawi et al. [32] discuss, CNNs as a whole struggle with applications in which the spatial positioning of input data is necessary to form inferences. Such a limitation is significant in addressing the optical flow problem as the spatial information of the input is key to forming optical flow inferences.

2.2 Recurrent Neural Network

Recurrent neural networks (RNNs) are another variant of ANNs. These networks minimally contain an input layer, a number of hidden layers, and an output layer. The differentiating factor of an RNN from a generic ANN is the recurrent connections, or feedback loops, within the hidden layers of the network. Because of their recurrent connections, RNNs can effectively characterize the temporal dependencies within an input sequence. Additionally, each generated inference is directly influenced by all prior inferences.

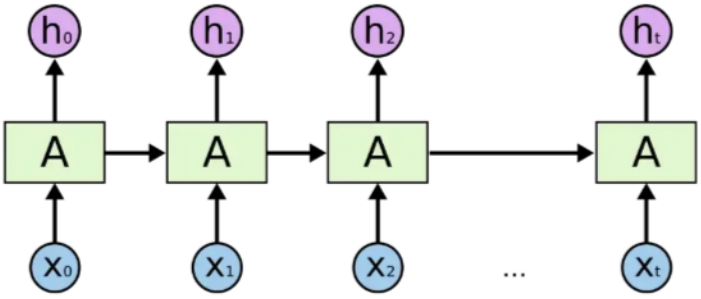

Figure 2.4: An example of a RNN where X represents the input layer at a given time t, A represents the hidden layer, and h represents the output layer at a given time t. Image source: [4].

While having found success in temporal applications such as time-series predictions, limitations of RNNs exist. As discussed by Salehinejad et al. [33], RNNs struggle to model data with long-range dependencies due to the vanishing gradient problem. To address this issue and others, various RNN-based architectures have been introduced. Some of the most well-known include Long Short-Term Memory (LSTM) [34] and Gated Recurrent Unit (GRU) [35] which are frequently used as backbone architectures to address various downstream tasks. Perhaps a more significant limitation, in terms of addressing the optical flow problem, is that RNNs tend to struggle with spatiotemporal data [33]. Such a limitation is consequential as both spatial and temporal information is required in forming an optical flow hypothesis.

2.3 Transformers

Introduced by Vaswani et al. [5], transformer architectures have been of great interest to researchers in the machine learning domain due to their successes in sequence-to-sequence tasks. These architectures are identified by the self-attention-based encoder and decoder modules. Rather than operate on an input sequence sequentially, as would be the case in a traditional recurrent architecture, a transformer works with the entire input sequence at once. The encoder transforms a sequence of input data into a latent representation by correlating the most relevant details of the entire input sequence. The decoder then decodes this latent representation into an associated target sequence.

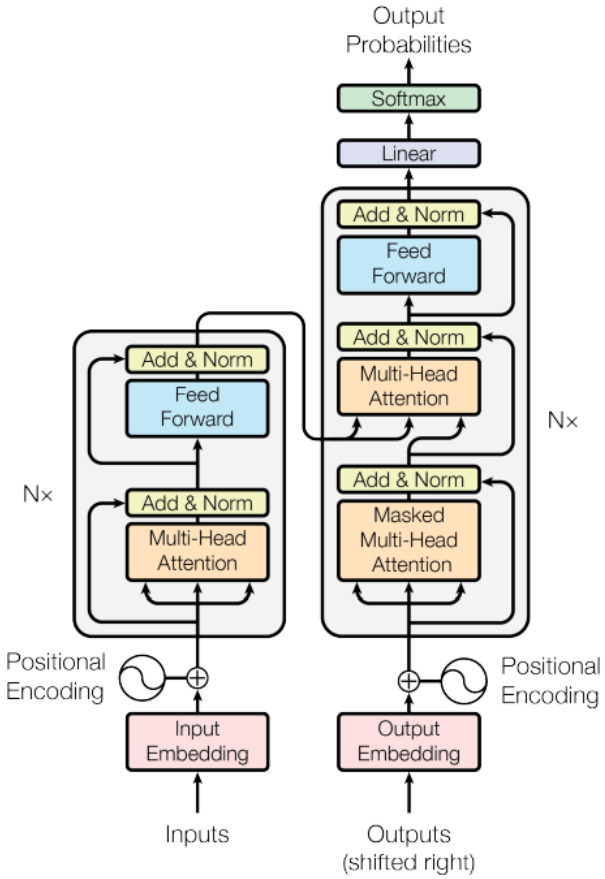

Figure 2.5: The original transformer architecture. The left of the figure details the transformer encoder. The right of the figure details the transformer decoder. Image source: [5].

The transformer must first prepare the input data for processing by the encoder. Preperation includes the tokenization of the input sequence by parsing the discrete values into tokens. Then, each token undergoes an embedding procedure to convert the token to a continuous-valued vector. A positional encoding is also added to the tokenized data to indicate the position of each token within the original input sequence.

The first operation of the transformer encoder is multi-head attention. The basis of multi-head attention is the more general concept of self-attention which was first introduced by Lin et al. [36]. While the specific implementations of self-attention mechanisms vary, the base functionality is similar to that of a recurrent architecture in that both are used to characterize the dependencies between the individual tokens of a sequential sequence. However, a self-attention mechanism operates on the entire input sequence at once rather than on the individual tokens in a sequential manner. Figure 2.6 details the specific self-attention mechanism, scaled dot-product attention, used in the original transformer architecture. In the context of multi-head attention in the encoder, the Q, K, and V values shown are identical matrices representing the tokens of a sequence.

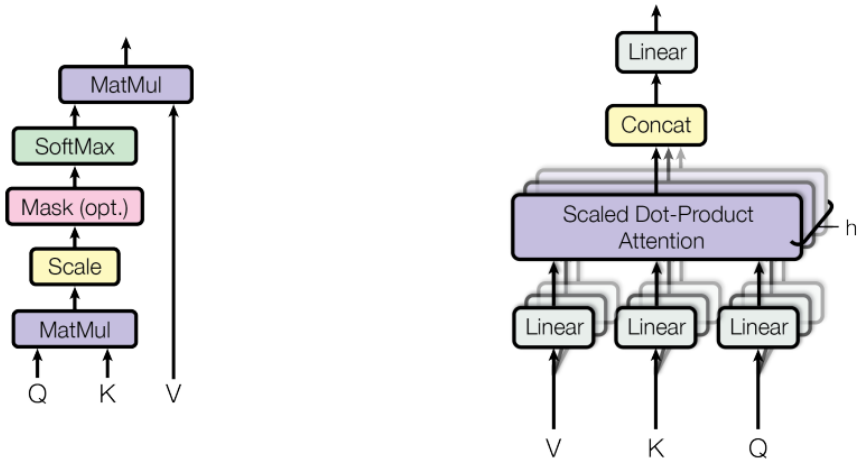

Figure 2.6: The self-attention mechanism utilized in the original transformer architecture. Left: The scaled dot-product attention mechanism. Right: The multi-head attention mechanism composed of multiple scaled dot-product attention units. Image source: [5].

Scaled dot-product attention first matrix multiplies the Q and K vectors. The result is scaled and provided to a softmax layer whose output is then matrix multiplied with the V vector. The resulting attention scores characterize the dependencies of each token in relation to the rest of the input sequence. Multi-head attention employs several scaled dot-product attention mechanisms in parallel as shown by Figure 2.6. Each mechanism operates with a unique representation of the Q, K, and V matrices obtained through learnable linear projections. This design allows for the examination of various representations of the input sequence when calculating attention. Before using the self-attention calculations, the various representations are concatenated and provided to a linear layer.

The feed-forward network, similar to that found in a traditional ANN, is applied to the resulting attention calculations of the input sequence. In doing such, the transformer encoder further learns the dependencies between input tokens by characterizing the relationships of the attention scores for each token.

Figure 2.5 also details layer normalization and residual connections in the transformer encoder. Both of these aspects, which are also present in the decoder module, help to stabilize model training.

For the decoder, the transformer must also prepare the input data. During training, the complete ground truth target sequence is provided as input to the decoder and is shifted forward by one unit, tokenized, and embedded. A positional embedding is also applied to each token. As the transformer is an autoregressive model, shifting the target sequence is necessary to ensure proper training. Without the shift, the decoder would learn to generate target tokens utilizing the ground truth value of the associated target token. In actuality, the transformer decoder must form a hypothesis with only the encoded input sequence and the previously generated output tokens. During testing, the predicted target sequence up to and including the last generated token is utilized as decoder input instead of the ground truth sequence.

Within the decoder, the masked multi-head attention layer is used to calculate the attention scores between all previously generated target tokens. During training, the self-attention scores are calculated between the ground truth target tokens instead. This layer is functionally similar to the multi-head attention layer found in the encoder but contains one notable difference. During the training process, future tokens of the ground truth target sequence are masked. Doing such ensures that the attention calculations do not include tokens that would otherwise not be available to the decoder during testing. In both cases, Q, K, and V are identical matrices.

The output of the masked multi-head attention layer is then provided to a multi-head attention layer that is much the same as that in the encoder. However, in this instance, the multi-head attention layer utilizes the encoded output sequence as the K and V matrices and the output of the masked multi-head attention layer as the Q matrix. This is known as cross-attention. By utilizing cross-attention the decoder can calculate attention scores relating the encoded input sequence and the target tokens to assist in forming the output sequence. As such, the generation of each token of the output sequence is dependent on all prior target tokens.

The feed-forward network of the decoder is then applied to the output of the multi-head attention layer. Through using the feed forward network, the decoder is able to further characterize the attention calculations that relate the encoded and target sequences.

Lastly, a linear layer converts the final decoder output to a probability of an output token's particular value given the specific application. Probabilities are based on potential target sequence values.

In expanding the transformer, Dosovitskiy et al. [6] introduce the vision transformer. The vision transformer strongly resembles the original transformer but with some slight modifications to adapt the architecture for use with visual and grid-like data.

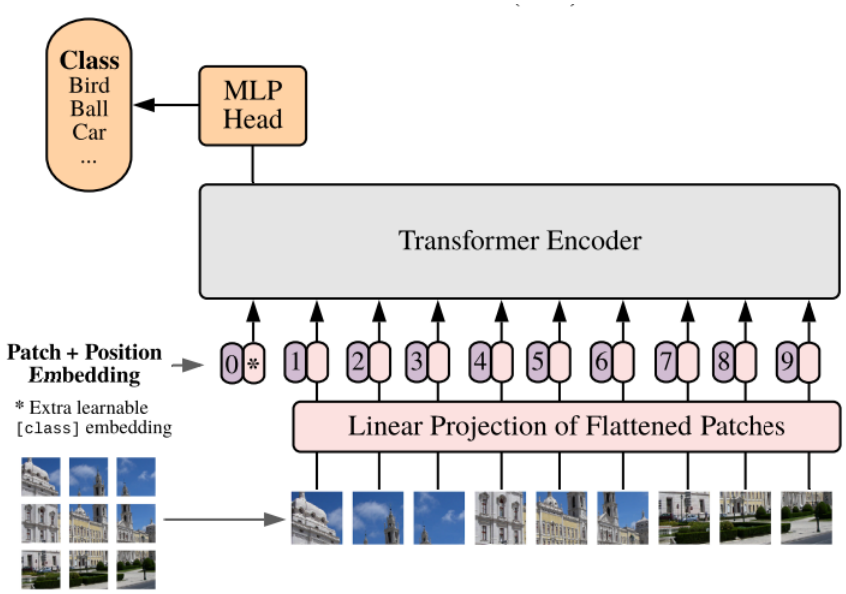

Figure 2.7: The original vision transformer architecture. This particular model is used for the image classification task. Image source: [6].

The first modification is in the preparation of data provided to the transformer encoder. Data tokenization occurs by splitting the input image into flattened, fixed-sized patches. Embedding and positional encoding is then performed as is done in the original transformer design. Figure 2.7 also details the use of an additional token to assist the model in performing the classification task. This extra token is optional with its usage dependent on the specific downstream task. The prepared data is then provided to a transformer encoder identical to that of the original transformer proposed by Vaswani et al. [5].

The second modification introduced by the vision transformer is that a self-attention-based decoder is not expressly required. In the case of the model shown in Figure 2.7, a multi-layer perceptron (MLP) head serves to generate class predictions by decoding the encoded representation of the input sequence. The specific method to generate inferences from the latent representation should be chosen based on the downstream task. The design may utilize a transformer decoder, an MLP head as shown, any other method, or nothing at all.

Vision transformers have largely been an effective means to close the gap between spatial and temporal machine learning applications. Vaswani et al. [5] was the first to detail the successes of the transformer architecture in natural language processing tasks. Dosovitskiy et al. [6] further detailed the successes of the transformer in vision tasks. However, the work by Jaegle et al. [17] demonstrated that the transformer can be successfully used in a variety of machine learning tasks including spatiotemporal tasks like optical flow.

2.4 Optical Flow

Before it was considered a computer vision and machine learning task, the optical flow problem was first introduced in the psychology domain. In the mid-20th century, J.J. Gibson theorized that observers utilize the apparent motion of objects within a particular frame of reference to draw conclusions about the environment they reside in [37]. It was not until the work by Marr and Poggio [38] in the 1970s that researchers began to consider the implications of optical flow in the computer vision domain due to the significant potential to assist downstream tasks. Since then, optical flow has remained an active research field in the computer vision domain with numerous advances over the years.

2.4.1 Variational Approaches

Many of the first models to address the optical flow problem as a computer vision task are based on variational approaches. Perhaps the most well-known is the Horn-Schunck method [39] which many consider one of the first successful methods to address the dense optical flow problem using differential equations. The basis of the Horn-Schunck method is the minimization of the energy function given by Equation 2.1 where $\mathcal{E}_c^2$ is represented by Equation 2.2, $\mathcal{E}_b$ is represented by Equation 2.3, and α is a weighting factor.

$$\mathcal{E}^2 = \int\int (\alpha^2 \mathcal{E}_c^2 + \mathcal{E}_b^2) dx dy \tag{2.1}$$

$$\mathcal{E}_c^2 = (\frac{\partial u}{\partial x})^2 + (\frac{\partial u}{\partial y})^2 + (\frac{\partial v}{\partial x})^2 + (\frac{\partial v}{\partial y})^2 \tag{2.2}$$

$$\mathcal{E}_b = E_x u + E_y v + E_t \tag{2.3}$$

Equation 2.2 characterizes the smoothness between two images for which optical flow is calculated. Equation 2.3 assists in characterizing the changes in brightness between the images. Within these equations, u and v represent the individual dimensions of a 2D displacement vector that characterizes the calculated optical flow for a particular pixel. x and y represent the spatial positioning of a particular pixel and t represents the temporal positioning between the two images. Lastly, E_x, E_y, and E_t represent the partial derivatives of image brightness with respect to the specific subscript.

While numerous other variational approaches have been built upon the work by Horn and Schunck, several limitations to the fundamentals of the approach exist. Often, assumptions are required that limit the generality and applicability of models to downstream tasks. In the case of the Horn-Schunck method, smoothness and brightness assumptions are both made. In addition to limitations regarding preliminary assumptions of the data utilized by the models, variational methods have also struggled with modeling occlusions as well as large displacements [40]. While significant progress had been made in addressing these limitations, the improved models are often significantly more complex with increased inference times.

2.4.2 Convolutional Neural Network Backbone

The rise of deep learning has had tremendous impacts on the computer vision domain. Given the profound success of CNNs in pattern-based computer vision tasks, researchers began to explore whether optical flow models would also benefit from utilizing a CNN backbone architecture. The pioneering work by Dosovitskiy et al. [7] in proposing the CNN-based FlowNet model showed promising results, which paved the way for future research. Since then, optical flow models have trended away from traditional approaches to addressing the optical flow task and instead towards deep learning methods that continually achieve state-of-the-art results in numerous benchmarks.

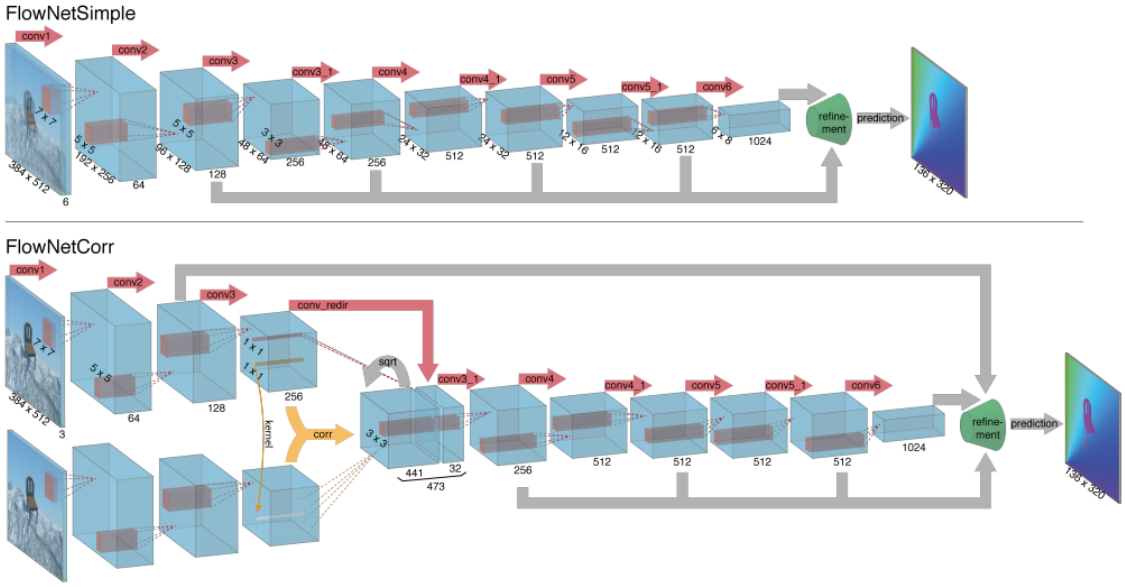

Figure 2.8: The FlowNet models. Top: FlowNetSimple. Bottom: FlowNetCorr. Image source: [7].

While two variations of FlowNet [7] exist, the underlying designs both utilize a CNN backbone to directly form an optical flow hypothesis. In FlowNetSimple, the two input images are stacked and then provided to a CNN feature extractor. On the contrary, in FlowNetCorr, each image is first independently provided to identical CNN-based feature extractors. A correlation between the extracted features is then

obtained via a cost volume that is then provided to another CNN-based feature extractor. In both designs, the latent representations from the CNN backbone are utilized by a refinement network composed of convolution and reverse pooling layers. The purpose of this refinement network is to up-sample the extracted features and form the final optical flow hypothesis for the provided images. The authors find that both methods were comparable to existing methods in terms of accuracy but are much more efficient in terms of run time.

Since the seminal work of Dosovitskiy et al. [7], numerous CNN-based optical flow models have built upon the proven success of FlowNet with various modifications. For example, SpyNet [41] applies a coarse-to-fine approach with a spatial-pyramid network. In this design, low-resolution optical flow predictions are updated to form accurate final predictions. FlowNet2.0 [12] utilizes a stacked architecture of FlowNetSimple and FlowNetCorr sub-modules. PWC-Net [14] employs a combinational approach using various aspects of optical flow domain knowledge including pyramidal processing, warping, and cost volumes. MR-Flow [13] employs the semantic segmentation task to assist in generating optical flow predictions. While numerous other variants of CNN-based optical flow models exist, these are some of the most popular that have achieved once state-of-the-art status in various metrics.

2.4.3 Recurrent Backbone

The next major shift in addressing the optical flow task came with the introduction of recurrent architectures within model designs. Recurrent architectures, including LSTM [34] and GRU [35], were found to achieve favorable results in temporally dependent machine learning tasks. Researchers were quick to explore the applicability of recurrent designs in the optical flow domain due to the temporal nature of the task. One of the most well-known optical flow models to use a recurrent design is Recurrent All-Pairs Field Transforms (RAFT), proposed by Teed and Deng [8].

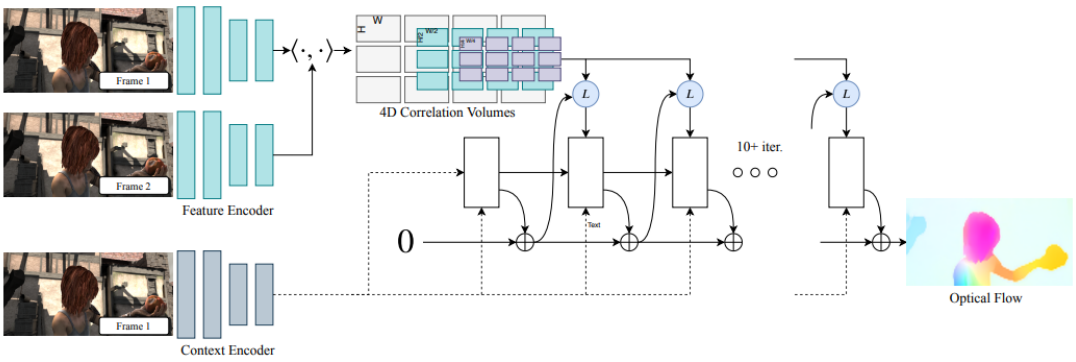

Figure 2.9: The RAFT model. Image source: [8].

RAFT [8] first uses a CNN-based feature encoder to perform per-pixel feature extraction of both input images. Additionally, the first image of the sequence is provided to a separate CNN-based context encoder. The extracted information from the feature encoder is then utilized by a correlation layer to construct a cost volume via the inner product of all feature vector pairs. The recurrent-based update operator is then used to form an optical flow inference. This step calculates and recurrently updates optical flow inferences until the final prediction is produced. The basis of the update operator is a derivative of the GRU model [35] that utilizes the cost volume, context encoder output, and the prior iteration of the optical flow inference. At the time of publication, RAFT [8] achieved state-of-the-art status for accuracy across several optical flow datasets. The model is also efficient regarding inference time, the number of learnable parameters, and the time required to train the model.

With the success of RAFT [8], there was a rise in optical flow models containing recurrent elements. These models often outperformed pure CNN-based approaches that did not incorporate recurrent elements. Global Motion Aggregation (GMA) [16] is one model that utilizes CNNs, self-attention mechanisms, and the recurrent GRU [35] network for refinement. GMA [16] has performed notably well, especially in the case of occlusions. It has also been once considered state-of-the-art and outperforms CNN-based models.

2.4.4 Vision Transformer Backbone

Many celebrate transformer architectures for their effectiveness in learning spatiotemporal dependencies while overcoming the limitations of recurrent architectures. While initially popular with natural language processing tasks, the success of Dosovitskiy et al. [6] in applying transformer architectures to the computer vision domain prompted researchers to explore the potential applications of use in the optical flow problem. While transformer architectures within optical flow models are a recent trend, one of the most promising models is FlowFormer, proposed by Huang et al. [9].

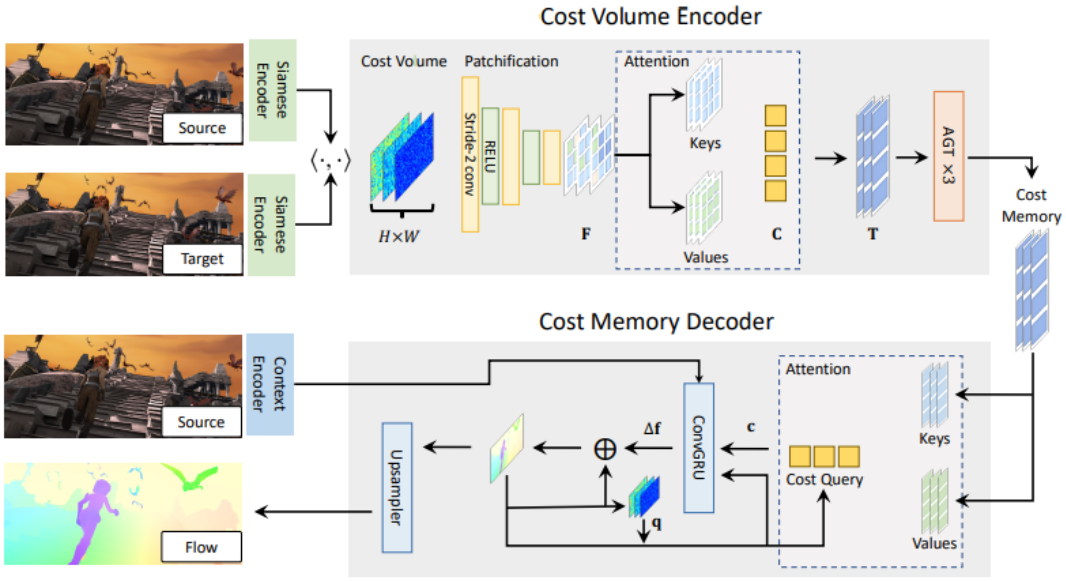

Figure 2.10: The FlowFormer model. Image source: [9].

FlowFormer [9] follows an encoder-decoder design in which an encoder produces a latent representation of the input data that is subsequently decoded to form an optical flow hypothesis. However, before calculating the latent form, Siamese encoders extract features from each of the two input images. A cost volume is then obtained from the extracted features on which the FlowFormer [9] encoder operates. While also containing some novel aspects, the encoder is largely based on a vision transformer architecture. The latent cost memory obtained from the encoder, as well as the result of separate context encoding from the first image, is provided to the decoder to produce

an optical flow hypothesis. The decoder also uses some additional operations but is primarily modeled from a recurrent version of the vision transformer. FlowFormer [9] has achieved state-of-the-art status regarding accuracy in terms of average endpoint error and generalizes well across various optical flow datasets.

While transformer-based optical flow models are still a new design trend, other models in addition to FlowFormer [9] have been proposed. Perceiver IO [17] is largely regarded as the first model to address optical flow with a transformer backbone. Perceiver IO [17] outperforms non-transformer-based optical flow models while performing well on large displacements and occlusions. GMFlow [18] is another transformer-based optical flow model that sought to address the problem of large displacements. In addition to performing well on large displacements, GMFlow [18] is found to be more computationally efficient than many non-transformer-based approaches. Despite the vast differences in their designs, a common trend is observed in that transformer-based optical flow models frequently outperform non-transformer-based designs across various metrics. In general, optical flow models with transformer architectures are the most accurate and efficient models to date.

2.4.5 Cost Volume Utilization

As Hur and Roth [42] discuss, many of the first optical flow models to utilize the cost volume chose to use a 3D cost volume. Typically, the costs associated with the direct matching of all features between two images would result in a four-dimensional (4D) vector. However, due to concerns regarding computational feasibility, many models sacrificed the accuracy of a 4D cost volume for the computational efficiency of a 3D cost volume. It was not until the work of Xu et al. [43] in the DCFlow model that 4D cost volumes became common. The DCFlow model [43] demonstrated a method to efficiently construct a 4D cost volume using the learned features of the input images while achieving once state-of-the-art status in terms of accuracy.

The seminal work by Dosovitskiy et al. [7] in FlowNet provides a comprehensive and direct analysis of the effects of utilizing the cost volume as a pre-processing operation for a CNN-based optical flow model. With two models of similar design, the authors find that the model employing a cost volume achieves better accuracy than an alternate model that does not utilize the cost volume. The authors also find that cost volume use increases the time to generate optical flow inferences. However, the number of learnable parameters in both models is similar. Numerous CNN-based models have since utilized the cost volume as a pre-processing operation to assist the backbone architecture. To the author's knowledge, the work by Dosovitskiy et al. [7] is the only work in the optical flow domain to provide a direct and comprehensive comparison of the effects of cost volume use in optical flow models.

As both recurrent and transformer-based optical flow models rose in popularity, cost volume use as a pre-processing operation continued. However, to the author's knowledge, no work exists in directly examining the associated effects in these models. Despite the lack of research, some have offered critiques. Xu et al. [18] discuss that cost volume use limits the ability to handle large displacements. Lu et al. [44] discuss similar concerns in that the cost volume results in a limited receptive field, heightened susceptibility to outliers, and increased computational complexity. To address their concerns, Xu et al. [18] introduce GMFlow, and Lu et al. [44] introduce the current state-of-the-art optical flow model TransFlow. Both are vision transformer-based and do not use the cost volume as a pre-processing operation. While the authors of both papers provide an extensive analysis of the benefits of their designs, neither provides a direct analysis of how the cost volume specifically affects vision transformer-based optical flow models. An analysis of this type requires the comparison of two models that only differ in the use of the cost volume, similar to how Dosovitskiy et al. [7] examined the effects of the cost volume in the CNN-based FlowNet design.

Chapter 3

Methodology

3.1 Model Design

The typical optical flow model utilizes a backbone architecture to produce a latent representation of the input data. Then, using the latent form, a model head generates the optical flow hypothesis. As such, the effects of using the cost volume as a pre-processing operation for the model backbone would be most apparent in the process of creating the latent data. Regarding latent forms, the work by He et al. [45] has shown that the latent representation, and the creation thereof, significantly influences the overall performance of machine learning models. Tong et al. [46] verified these findings regarding video data. To examine how the utilization of the cost volume as a pre-processing operation affects vision transformer-based optical flow models, the process of creating the latent representation of input data is the primary focus for analysis. Two autoencoder models are developed for this analysis.

3.1.1 Autoencoder Network

An autoencoder network consists of an encoder and decoder module and follows a self-supervised training strategy [47]. The encoder transforms input data into a latent representation while the decoder converts the latent data back to the original form of the encoder input. While not required, the decoder often resembles the inverse

of the encoder. The overall goal of an autoencoder model is that the decoded data resembles the original input data as closely as possible. From the training process, the decoder learns to decode the latent form, and the encoder learns how to create a feature-rich latent space to represent input data. Data collected from this training process can be collected to characterize the encoder's ability to form meaningful latent representations of the input data. In consideration of the work by He et al. [45], if the encoder were to produce the latent form of input data in a manner similar to an optical flow model, the data collected would also provide subsequent insights into that associated optical flow model.

Two autoencoders are used to examine and compare the effects of the cost volume on the creation of the latent form. One autoencoder utilizes the cost volume as a pre-processing operation for the encoder, while another does not. In comparing the data collected from both models, the effects of the cost volume on the latent representation can be understood. By extension, the overall impact of the cost volume on a complete optical flow model is also understood.

3.1.2 Direct Model

The first autoencoder model, known as the direct model, does not utilize a cost volume. In following the FlowNetSimple [7] optical flow model, two input images are first stacked along the channel dimension resulting in a vector of shape $[1 \times 2D \times H \times W]$, where D represents the number of channels of the input images, and H and W represent the spatial dimensionality of the images. This input data vector is provided directly to a vision transformer-based encoder to generate a latent representation of the input images. Another vision transformer, used as the decoder, returns the latent representation to the original form of the input data provided to the autoencoder model. An additional linear layer up-samples the decoder output to ensure the correct output resolution.

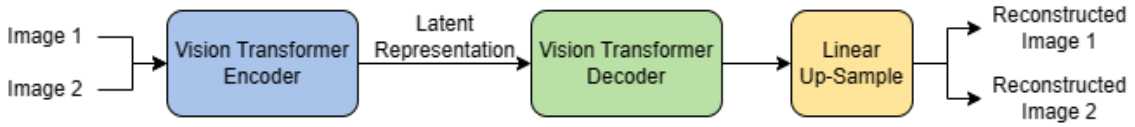

Figure 3.1: The direct autoencoder model.

While the encoder design is relatively simplistic, the primary functionality of the encoder produces a latent representation of the input data in a manner similar to popular vision transformer-based optical flow models. FlowFormer [9], Perceiver IO [17], and GMFlow [18] each operate with various auxiliary operations but ultimately rely on a vision transformer backbone to produce a latent form of the input data.

In following the typical autoencoder scheme, the decoder implementation strongly resembles the inverse of the encoder. A vision transformer similar to that of the encoder is utilized, but with slight modifications to ensure the proper handling of the latent data and the correct output format. All other aspects of the vision transformer decoder are identical to that of the encoder.

Due to concerns regarding the overall size of the autoencoder and resource limits, the implementation of the vision transformer decoder may not be able to produce an output that matches the original resolution of the input images. Therefore, an additional linear layer is employed to obtain the correct resolution of the decoded output.

3.1.3 Cost Volume Model

The second autoencoder developed is the cost volume model. In this design, the input image data is pre-processed rather than provided directly to the encoder. The basis of the pre-processing operations is the construction of a cost volume from the extracted features of the input images. Utilizing the constructed cost volume, the encoder produces the latent representation of the input images. With the encoded data, the decoder forms a reconstruction of the original input images.

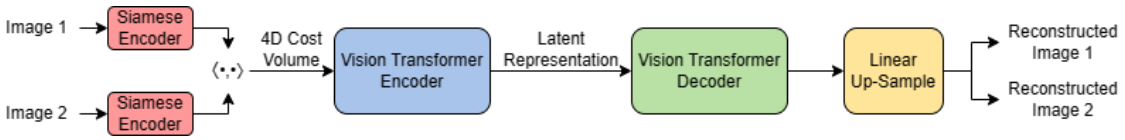

Figure 3.2: The cost volume autoencoder model.

Since DCFlow [43], many of the optical flow models that use a cost volume as a pre-processing operation do so by constructing a 4D cost volume from a learned feature map of each image. In following this trend, and explicitly following the design of FlowFormer [9], a learnable Siamese encoder is used to extract a feature map for each of the two input images provided to the autoencoder model. A cost volume is then formed by calculating the dot-product similarity between each of the feature vector pairs across the two feature maps. Equation 3.1 [8] characterizes this calculation in which $g_\theta(I_1)_{ijh}$ represents the extracted features of the first image I_1, and $g_\theta(I_2)_{klh}$ represents the extracted features of the second image I_2.

$$C_{ijkl} = \sum_h g_\theta(I_1)_{ijh} \cdot g_\theta(I_2)_{klh} \tag{3.1}$$

In the described calculation, the resulting cost volume of the extracted features is of shape $[I \times J \times I \times J]$ where I and J represent the spatial dimensions of the extracted feature vectors. To further illustrate this calculation, consider Figure 3.3 which details a cost volume calculated from feature maps with a spatial dimensionality of 2×2.

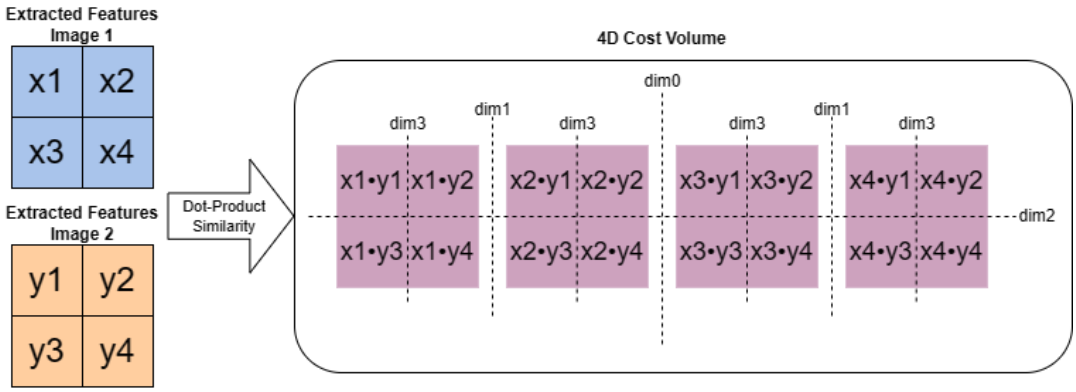

Figure 3.3: An example of the calculation and representation of the 4D cost volume. xi and yi represent the extracted features of the input images from the Siamese encoder.

The cost volume is then provided to the encoder to produce the latent representation of the input data. As with the direct model, the basis of the encoder design is a vision transformer architecture. Aside from the input data size parameter of the vision transformer, all aspects of the encoder design are identical to that of the direct model. The latent representation of the input data is then provided to the decoder to reconstruct the original input data. The decoder implementation is identical to that of the direct model as the latent representation of input data is the same between model designs. An additional linear layer not part of the decoder is also employed to obtain the correct resolution of the output data due to concerns regarding the model size and resource limitations.

3.2 Model Evaluation

To examine the impacts of the cost volume between model designs, the training and evaluation process of each autoencoder is identical. Data regarding the time required to train each model, the accuracy of model validation inferences, model inference time, and model size are collected. As each autoencoder is identical apart from the utilization of the cost volume, any differences observed are directly attributable to the cost volume and associated pre-processing operations. By having ensured that each autoencoder constructs a latent representation of input images in a manner similar to state-of-the-art vision transformer-based optical flow models, the data collected provides valuable insights that also apply to the optical flow domain. Unlike the work of Xu et al. [18] and Lu et al. [44] this comparison of two models differing only in cost volume use allows for a direct examination of how the cost volume affects vision transformer-based optical flow models.

3.2.1 Training Sequence

The training and evaluation processes of each autoencoder is with identical sequences of data to ensure an accurate comparison of the data collected from each model. In a random but reproducible manner, a dataset is shuffled and split to form separate training and validation sequences consistent between the training process of each autoencoder.

The mean squared error (MSE) loss function is the basis of evaluation for the training sequence of each autoencoder. In following the MSE loss function given by Equation 3.2, an average of the squared difference between each pixel of the reconstructed images and the associated input images is calculated.

$$MSELoss = \frac{1}{1/(2 * D * H * W)} \sum (y - x)^2 \tag{3.2}$$

Within Equation 3.2, y represents the reconstructed input images produced by the autoencoder, and the variable x represents the original input images.

3.2.2 Evaluation of the Effects on Training Time

The number of epochs required for each autoencoder to converge on a training set is used to characterize the effects of the cost volume on model training time. To assist in the determination of model convergence, the average MSE loss associated with a validation sequence is of primary focus. When plotting the average validation MSE loss per training epoch the observation of a consistent and non-random increase or plateau indicates that the model has converged since additional training provides no benefit to the model. After identifying model convergence, the epoch associated with the global minimum of the plot is used to quantify the number of epochs required for model convergence.

3.2.3 Evaluation of the Effects on Accuracy

The average MSE loss of a validation sequence is also used to characterize the effects of the cost volume on model accuracy as both metrics are related. A low MSE loss indicates a high reconstruction accuracy of a validation sequence and thus a high-quality latent representation of the input data. In plotting the average validation MSE loss for each epoch of each autoencoder's training sequence, the effects of the cost volume on overall model accuracy is easily compared between models.

3.2.4 Evaluation of the Effects on Inference Time

By examining the average elapsed wall clock time for inference generation across a validation set, a thorough and accurate comparison of the effects of the cost volume on model inference time is provided. Software timers are employed to perform these measurements. Auxiliary operations, like the calculation of the MSE loss, are not included in timing measurements as they are irrelevant outside the training process. In further keeping with potential real-world use cases, the batch size for timing measurements is one image pair.

3.2.5 Evaluation of the Effects on Model Size

The number of learnable parameters within each autoencoder are counted to examine and compare the effects of the cost volume on model size. A larger model will contain more learnable parameters, while a smaller model will utilize less.

Chapter 4

Findings and Results

4.1 Implementation Details

The design of each autoencoder is strongly influenced by the FlowFormer optical flow model introduced by Huang et al. [9]. The specific implementation of the vision transformer architectures within each autoencoder draws inspiration from the work of He et al. [45], who have prior utilized vision transformers in an autoencoder format.

4.1.1 Direct Model

The encoder of the direct model follows the original vision transformer architecture proposed by Dosovitskiy et al. [6]. The encoder utilizes a patch size of 8 to operate on the input data vector. The depth of the encoder of the vision transformer is 12. The number of attention heads is also 12. The raw latent data shape is $[1 \times (1/8H * 1/8W * 2)]$, which is reshaped to $[1 \times 2 \times (1/8H * 1/8W)]$. A decoder is not employed within the transformer architecture utilized.

The design of the decoder of the direct model is also based on the Dosovitskiy et al. [6] vision transformer architecture. The decoder operates on the latent data with a patch size of 8. The depth of the encoder of the transformer architecture is 8, and the number of attention heads is 16. The decoded output vector is of shape $[1 \times (1/8H * 1/8W * D * 2)]$. No decoder is utilized within the transformer architecture.

4.1.2 Cost Volume Model

The design of the Siamese encoder utilized in the pre-processing operations of the cost volume model is a variation of the Twins-SVT-L architecture [48] pre-trained on the ImageNet-1k dataset [49]. The Siamese encoder contains the first two stages of the Twins-SVT architecture [48] followed by a convolution layer. The resulting feature map for each image is of shape $[1 \times 256 \times 1/8H \times 1/8W]$. The resulting cost volume is of shape $[1/8H \times 1/8W \times 1/8H \times 1/8W]$. To assist in the processing of the 4D cost volume by the vision transformer-based encoder, the cost volume is reshaped to $[1 \times (1/8H * 1/8W) \times 1/8H \times 1/8W]$.

Aside from adjusting the expected size of the input sequence, all aspects of the cost volume model encoder implementation are identical to that of the direct model. The size of the latent space remains $[1 \times (1/8H * 1/8W * 2)]$, and is reshaped to $[1 \times 2 \times (1/8H * 1/8W)]$. In regard to the decoder module, the implementation is identical to that of the direct model. As such, the resulting decoded output vector is also of shape $[1 \times (1/8H * 1/8W * D * 2)]$.

4.1.3 Environment and Training Sequence

The implementation and evaluation of both models are in Python with the PyTorch framework. For the training and evaluation sequences of each model, a batch size of 1 is utilized. The optimizer used during the training process is stochastic gradient descent (SGD), with a learning rate of 0.1. A seed value of 1933 is employed to ensure reproducible results and an accurate comparison between each model.

Regarding the training environment, 3 NVIDIA A100 GPUs are utilized for the direct model, and 4 NVIDIA A100 GPUs for the cost volume model. The multiple GPUs are employed to support model parallelism. In both models, the encoder, the decoder, and the linear up-sample mechanisms exist on dedicated GPUs. For the cost volume model, the Siamese encoder also utilizes a dedicated GPU.

4.2 Flying Chairs Dataset

Both models are trained and evaluated on the Flying Chairs dataset, a synthetic dataset proposed by Dosovitskiy et al. [7] as part of their paper introducing the FlowNet model. It has since become one of the most widely used datasets in the training and evaluation of optical flow models.

The Flying Chairs dataset [7] contains 22,872 RGB image pairs and associated ground truth optical flow fields. Given the self-supervised training process of autoencoders, the optical flow fields are not required. The dimensions of each image are (512×384) pixels. For training and evaluation, all images are center cropped to a size of (256×192) pixels.

In following the widely accepted practice in the optical flow domain, the complete Flying Chairs dataset [7] is split into pre-defined training and validation sets of 22,232 samples and 640 samples, respectively. Both models are trained and evaluated on the same sequences for 100 epochs, the maximum amount given the run-time constraints of the training environment.

Figure 4.1: An example of two training samples from the Flying Chairs dataset [7]. Black borders are added to help distinguish the individual images. Left: scene at time $t1$, Center: scene at time $t2$, Right: associated optical flow represented in the color coding format introduced Baker et al. [10].

4.2.1 Training Time

The average MSE loss of the validation set is recorded after each training epoch to assist in determining model convergence and the required training time for each model. The per-epoch average MSE loss of every one-thousandth training sample is also shown. Figure 4.2 and Figure 4.3 detail the observed results.

Figure 4.2: A plot of the average MSE losses per training epoch of the Flying Chairs dataset [7] for the direct model.

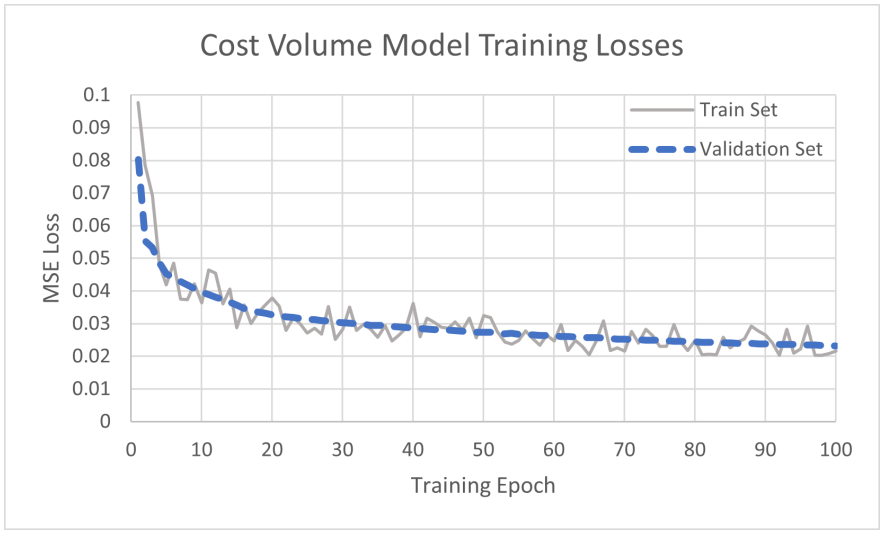

Figure 4.3: A plot of the average MSE losses per training epoch of the Flying Chairs dataset [7] for the cost volume model.

In both figures, a consistent increase, or plateau, of the average validation losses is not observed. Further, the training set losses do not diverge from the validation set losses, nor have they reached near-zero. As a result, neither model has converged. No determination can be made as to the training time required for either model.

4.2.2 Accuracy

The average validation MSE loss of each epoch for each model is displayed on the same plot to examine the accuracy's of the direct model and the cost volume model. Figure 4.4 details the plot comparing the results.

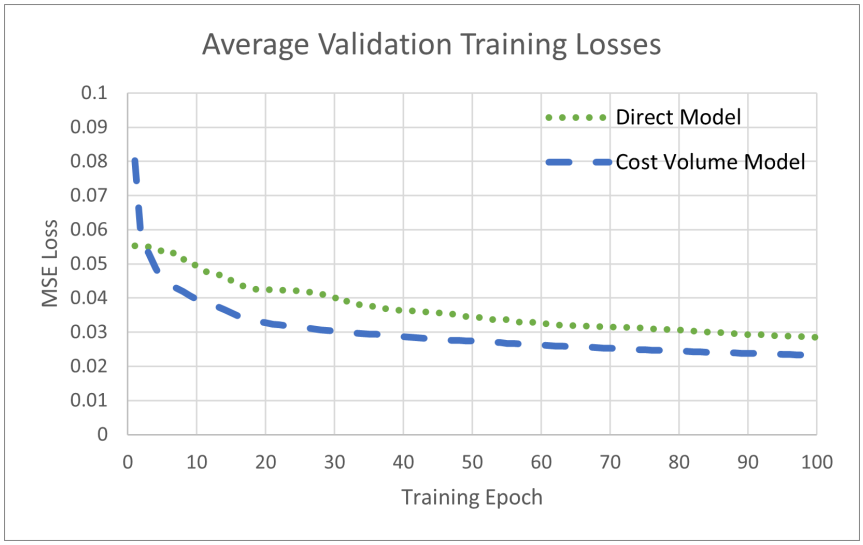

Figure 4.4: A plot of the average validation MSE loss per training epoch of the Flying Chairs dataset [7] for each model.

Although neither model has converged, the trend of the average validation losses provides qualitative insights into the overall accuracy of each model. The direct model is initially more accurate than the cost volume model as it has a lesser average validation MSE loss. However, by the third epoch, the cost volume model becomes more accurate than the direct model. This observation holds for the remainder of the 100-epoch training sequence. In following the trend line of the losses for each model, it is likely that the cost volume model is overall more accurate than the direct model.

4.2.3 Inference Time

Following the completion of the 100-epoch training sequence of each model, the time to generate inferences for the complete validation set is measured. The average inference time is then calculated to characterize the inference time requirements of each model. Table 4.1 details the total validation inference time and the resulting calculated average.

Table 4.1: Model Inference Time Results - Flying Chairs Dataset [7]

	Direct Model	**Cost Volume Model**
Total Validation Inference Time (Seconds)	15.78	15.34
Validation Samples	640	640
Average Inference Time (Seconds)	**0.02466**	**0.02397**

For the direct model, 15.78 seconds are required to produce the 640 inferences of the validation sequence for the Flying Chairs dataset [7]. Therefore, the average inference generation time for the direct model is 0.02466 seconds.

For the cost volume model, 15.34 seconds are required to produce those same inferences. Therefore, the average inference generation time for the cost volume model is 0.02397 seconds.

4.2.4 Model Size

PyTorch is utilized to count and report the number of learnable parameters for each model. In addition to the overall parameter counts, the number of parameters for the individual aspects of each model is recorded as well. The size of the encoder, decoder, and linear up-sample modules are recorded for the direct model. For the cost volume model, the size of the Siamese encoder is also recorded. Table 4.2 details the observed results.

Table 4.2: Model Size Results - Flying Chairs Dataset [7]

	Direct Model	**Cost Volume Model**
Siamese Encoder (parameters)	X	4,284,160
Encoder (parameters)	341,752,320	415,498,752
Decoder (parameters)	2,039,569,920	2,039,569,920
Linear Up-Sample (parameters)	339,886,080	339,886,080
Total Parameter Count	**2,721,208,320**	**2,799,238,912**

The encoder of the direct model contains 341,752,320 learnable parameters, while the decoder contains 2,039,569,920. The linear up-sample module contains 339,886,080 learnable parameters. The total size of the direct model is 2,721,208,320 learnable parameters.

The cost volume model includes a total of 2,799,238,912 learnable parameters. The makeup is 415,498,752 learnable parameters for the encoder module and 2,039,569,920 for the decoder. The linear up-sample module contains 339,886,080 learnable parameters. The Siamese encoder, unique to the cost volume model, contains 4,284,160 learnable parameters.

4.3 Sintel Dataset

To further verify findings, each model is also trained and evaluated on the Sintel dataset introduced by Butler et al. [11]. Like Flying Chairs [7], this is a synthetic dataset frequently used to train and evaluate optical flow models. The "clean" version is utilized for all experiments.

The training set of Sintel [11] contains 1,041 RGB image pairs and associated ground truth optical flow fields. The optical flow fields are not required for the training sequence of either autoencoder. The dimensions of each image are (1024 $\times$ 436) pixels. For training, all images are center cropped to a size of (256 $\times$ 192) pixels.

The Sintel dataset [11] also contains a dedicated test sequence that is used in the validation of each autoencoder model. The test set contains 552 RGB image pairs. No ground truth optical flow fields are provided as part of the test set, however, they are not needed for the validation of either autoencoder. As with the training set, each test set image used for validation is (1024 $\times$ 436) pixels center cropped to a size of (256 $\times$ 192) pixels.

Each model is trained for 1,500 epochs, the maximum amount given the run-time constraints of the training environment.

Figure 4.5: An example of two training samples from the Sintel dataset [11]. Black borders are added to each image to distinguish the individual images. Left: scene at time $t1$, Center: scene at time $t2$, Right: associated optical flow represented in the color coding format introduced Baker et al. [10].

4.3.1 Training Time

Figure 4.6 and Figure 4.7 detail plots of the average validation MSE losses of the direct model and the cost volume model, respectively. The average validation loss is calculated and plotted after each training epoch to assist in determining model convergence and model training time. The per-epoch average MSE loss of every fiftieth training sample is also shown to assist in the determination of model convergence on the training set.

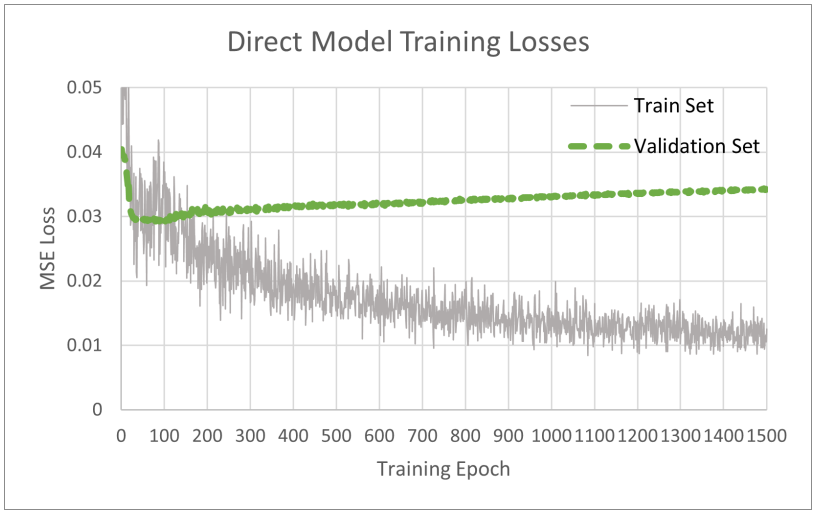

Figure 4.6: A plot of the average MSE losses per training epoch of the Sintel dataset [11] for the direct model.

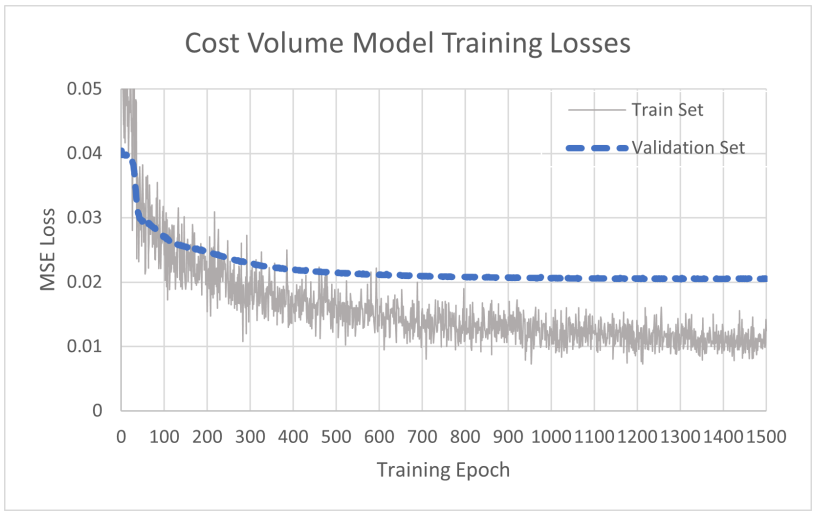

Figure 4.7: A plot of the average MSE losses per training epoch of the Sintel dataset [11] for the cost volume model.

Figure 4.6 details a clear and non-random increase in the average MSE loss of the validation sequence for the direct model. The training set losses further indicate convergence as they diverge from the average validation losses. The minimum MSE loss occurs at epoch 89.

Figure 4.7 details the convergence of the cost volume model via a plateau of the average validation MSE losses, and near-zero training losses. The minimum observed MSE loss of the cost volume model occurs at epoch 1,282.

4.3.2 Accuracy

The average validation MSE losses for both models are shown in the same plot to support analysis of the accuracy's of the direct and the cost volume models. Figure 4.8 details the associated plot.

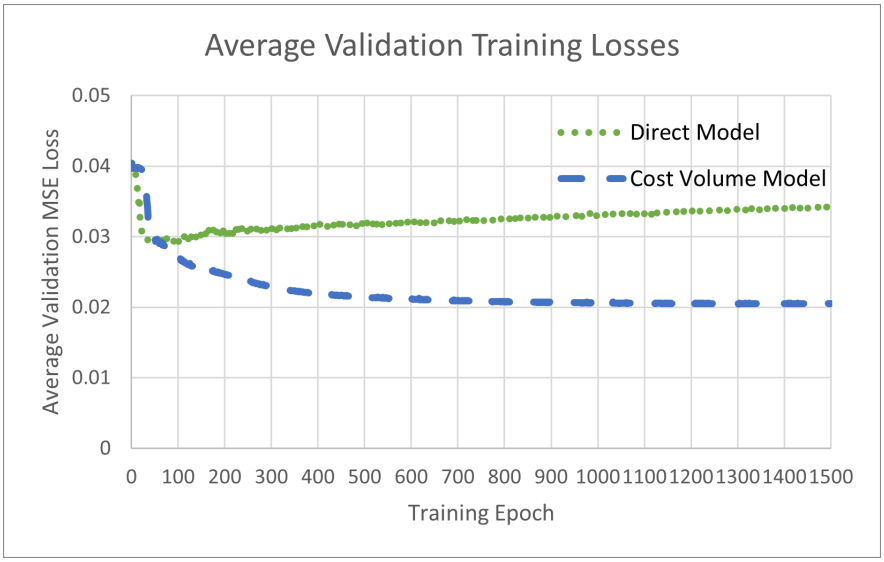

Figure 4.8: A plot of the average validation MSE loss per training epoch of the Sintel dataset [11] for each model.

In a qualitative analysis of the average validation losses of each model, the direct model is initially more accurate than the cost volume model, as indicated by lower average validation MSE losses. However, by epoch 51, the cost volume model becomes more accurate than the direct model, where it remains for the rest of the 1,500 epoch training process.

A quantitative comparison can also be made regarding the overall accuracy of each model as both have converged on the training set. The minimum observed average MSE loss of the validation sequence for the direct model is 0.02924 at epoch 89. For the cost volume model, a 0.02022 average validation MSE loss is observed at epoch 1,282.

4.3.3 Inference Time

The average inference time for each model is calculated from the total observed time to generate all inferences of the Sintel dataset [11] test sequence. Table 4.3 details the observed and calculated results to characterize the required inference time of each model.

Table 4.3: Model Inference Time Results - Sintel Dataset [11]

	Direct Model	**Cost Volume Model**
Total Validation Inference Time (seconds)	15.65	15.27
Validation Samples	552	552
Average Inference Time (seconds)	**0.02835**	**0.02766**

For the direct model, 15.65 seconds are required to produce the 552 inferences of the Sintel dataset [11] test sequence. From such, the average inference generation time is 0.02835 seconds.

The cost volume model requires 15.27 seconds to produce those same inferences. Therefore, the average inference generation time for the cost volume model is 0.02766 seconds.

4.3.4 Model Size

The number of learnable parameters for each model is reported according to PyTorch. Table 4.4 details the observed results regarding the overall size of each model as well as the size of the individual components that contribute to the total model sizes.

Table 4.4: Model Size Results - Sintel Dataset [11]

	Direct Model	**Cost Volume Model**
Siamese Encoder (parameters)	X	4,284,160
Encoder(parameters)	341,752,320	415,498,752
Decoder (parameters)	2,039,569,920	2,039,569,920
Linear Up-Sample (parameters)	339,886,080	339,886,080
Total Parameter Count	**2,721,208,320**	**2,799,238,912**

All images from the Sintel dataset [11] are cropped to (256 $\times$ 192) pixels, the same size used for the Flying Chairs dataset [7]. As a result, the direct model and the cost volume model are structurally identical across both datasets. Therefore, the overall model size and composition results reported in Table 4.4 are identical to those reported in Table 4.2.

Chapter 5

Discussion

5.1 Analysis of Results

The reported results are summarized in Table 5.1. The percentage change of the training time, model accuracy, average inference time, and model size from the direct model to the cost volume model is also reported to assist in analyzing the effects of cost volume use. For all reported values in Table 5.1, those on the left of the cell correspond to the Flying Chairs dataset [7], and the values on the right correspond to the results for the Sintel dataset [11].

Table 5.1: Summary of Results (Flying Chairs [7]/Sintel [11])

	Direct Model	**Cost Volume Model**	**Difference**
Training Time (epochs)	X/89	X/1282	**X/1,340 %**
Accuracy (MSE Loss)	X/0.02924	X/0.02022	**X/-30.8 %**
Average Inference Time (seconds)	0.02466/0.02835	0.02397/0.02766	**-2.80/-2.43 %**
Model Size (parameters)	2,721,208,320	2,799,238,912	**2.87 %**

5.1.1 Training Time

In regards to training time, an analysis cannot be provided for the models trained on the Flying Chairs dataset [7], as neither the direct model nor the cost volume model has converged. However, an analysis is provided for the Sintel dataset [11] as convergence is achieved for both models. Given the significant difference in the size of the training sets, 22,872 samples versus 1,041 samples, it is not unexpected that convergence is only observed on the Sintel dataset [11] due to run-time limits of the training environment. Since the Sintel dataset [11] is smaller than the Flying Chairs dataset [7], the per-epoch iteration time is less. As a result, the models can be trained for more epochs within the run-time limit of the training environment. While increasing the training set batch size can reduce the per-epoch iteration time, this is not possible due to memory constraints.

On the Sintel dataset [11], the direct model converges in 89 epochs, while the cost volume model converges in 1,282 epochs. The result is a 1,340% increase in training time for the cost volume model over the direct model. While the training time will vary between specific model implementations and training datasets, the substantially differing results show that utilizing the cost volume as a pre-processing operation increases model training time. As the production of the latent form in each autoencoder is consistent with complete optical flow models, the reported results signify that cost volume utilization increases the training time for vision transformer-based optical flow models.

These results are consistent with other works, such as that by Cordonnier et al. [50], who show that as the number of learnable parameters increases for self-attention-based models, the number of epochs required for model convergence also increases. As increasing the number of parameters within a model effectively increases model complexity, additional training is required for models to learn the relationships between input data and generate accurate inferences.

5.1.2 Accuracy

As discussed prior, neither the direct model nor the cost volume model has converged on the Flying Chairs dataset [7], meaning a quantitative analysis is not possible to examine how the cost volume affects model accuracy. However, an analysis is possible for the Sintel dataset [11] given that convergence is observed for both models.

The best observed average validation MSE loss for the direct model is 0.02924. For the cost volume model, the best observed average MSE loss is 0.02022, a 30.8% improvement. These results indicate that cost volume use to assist in producing the latent form of input data improves model accuracy. One reason for this improvement is an increase in the quality of the latent representation of input data. As the production of the latent form is similar to that of complete optical flow models, the reported results also indicate that cost volume use in vision transformer-based optical flow models can improve the latent space quality, and thus accuracy of optical flow inferences.

These findings coincide with those of Dosovitskiy et al. [7], who examined the use of the cost volume in CNN-based optical flow models. They find that on the Sintel clean test set [11], the FlowNetSimple model achieves an average endpoint error of 7.42. The model that utilizes a cost volume, FlowNetCorr, has an average endpoint error of 7.28, a 1.89% increase in model accuracy. While one might conclude that utilizing the cost volume in vision transformer-based optical flow models is more beneficial than in CNN-based models, this assertion is not necessarily accurate. The results by Dosovitskiy et al. [7] detail the accuracy of complete optical flow inferences. The results reported in this document detail the quality of the latent form used in producing optical flow inferences. While the quality of the latent form is directly related to overall model accuracy, a direct comparison can not be made between the differing backbone architectures. However, the holistic observation holds that the cost volume benefits both styles of optical flow models.

5.1.3 Inference Time

Regarding the average model inference generation time of the Flying Chairs dataset [7], the direct model requires 0.02466 seconds while the cost volume model requires 0.02397 seconds, a 2.80% improvement. Similar results are found for the Sintel dataset [11]. The direct model inference generation requires 0.02835 seconds, while the cost volume model produces inferences in an average of 0.02766 seconds. In this instance, the cost volume model is 2.43% faster than the direct model. As the latent data production of the autoencoders is in a manner similar to complete optical flow models, the results indicate that cost volume use provides a slight benefit to vision transformer-based optical flow models regarding average inference time.

These results differ from those observed for CNN-based optical flow models as reported by Dosovitskiy et al. [7]. In the case of FlowNet, cost volume use results in an average inference time of 0.15 seconds, an 87.5% increase over the 0.08 seconds to generate inferences without the cost volume. One explanation for this discrepancy is differences in how reducing the resolution of input data affects the different backbone architectures. As indicated in the original vision transformer paper, reducing input resolution by a factor of 2 reduces inference generation time by about 70% [6]. For a CNN, the same reduction of input resolution only results in a 27.4% improvement in inference time [51]. For all experiments performed within this document, the cost volume reduces input resolution by a factor of 8. While Touvron et al. [52] indicate that the increased size of the vision transformer-based cost volume model would typically increase inference time, it is likely that the magnitude of the benefits provided from reduced input resolution outweighs the negative impacts of increased model size. The result is that cost volume utilization provides an overall net decrease in the inference time for vision transformer-based models. In the case of CNN-based optical flow models, the lesser magnitude of improvement from reduced input resolution is likely insufficient, resulting in a net increase in inference time from cost volume use.

5.1.4 Model Size

The input data from the Flying Chairs dataset [7], and the Sintel dataset [11], is cropped to a standard size of (256×192) pixels resulting in identical model sizes between datasets. The overall model size of the direct model is 2,721,208,320 learnable parameters. The cost volume model size is 2,799,238,912 learnable parameters, a 2.87% increase over the direct model. Table 4.2 and Table 4.4 indicate that the source of the increase in overall model size is the addition of the Siamese encoder for feature extraction as well as an increase in the size of the encoder to generate the latent representations of the input data. As both of these differences would also be present for a complete optical flow model, these results indicate that cost volume utilization results in a slight increase in model size for vision transformer-based optical flow models.

The reported results are consistent with those regarding the effects of the cost volume on model size for CNN-based optical flow models. While Dosovitskiy et al. [7] do not provide official FlowNet source code or report the exact model sizes, an unofficial implementation by Clément Pinard [53] reveals that the cost volume-based model requires 39,179,152 learnable parameters. The model that does not utilize a cost volume requires 38,680,336 parameters. In the case of the CNN-based FlowNet model, cost volume utilization increases model size by 1.29%. While the cost volume-based FlowNet model does not utilize an additional Siamese encoder for feature extraction, overall model size still increases from cost volume use. Despite the fact that the cost volume has a smaller resolution than the raw input images, the overall size of the cost volume is greater due to increased depth. For a CNN to accommodate the increased depth, the convolution layers must increase their number of learnable parameters, thus increasing overall model size. Vision transformers must also increase in size to accommodate the increased depth, which in addition to the use of Siamese encoders, increases overall model size from cost volume utilization.

5.2 Limitations

The results reported regarding the effects of the cost volume on vision transformer-based optical flow models only apply to models in which the cost volume is used as a pre-processing operation to assist the model backbone in forming latent representations of the input data. While this is the typical method employed by optical flow models, some models use the cost volume in an alternate manner, such as the vision transformer-based GMFlow model introduced by Xu et al. [18]. Rather than utilize the cost volume to assist in pre-processing the input images, GMFlow uses the cost volume as the basis of an optical flow hypothesis which is refined to a final prediction. In this instance, the cost volume is a primary feature of the model rather than an auxiliary pre-processing operation used to assist the model backbone. For models such as these, the reported effects of cost volume use do not apply.

Additionally, the generalization of the results on non-synthetic datasets is unknown since all experiments were performed on synthetic datasets. Given the requirements for large amounts of labeled training data, especially in the case of transformer architectures [5], many rely on synthetic datasets such as FlyingChairs [7] and Sintel [11] to assist in training optical flow models. While these synthetic datasets are of high quality, they do not accurately represent data encountered in "real world" scenarios. Unfortunately, very few realistic optical flow datasets exist. Of those that do, such as KITTI [54], they are often smaller than synthetic datasets and are typically for sparse optical flow predictions. This discrepancy is a direct result of the difficulty in labeling realistic data for the optical flow task. As the reported results examine the latent representation of input data rather than the construction of complete optical flow fields, it is unlikely that a non-synthetic dataset would produce significantly different results. However, this claim cannot be made with complete certainty.

Chapter 6

Conclusion

In the typical use of a cost volume in optical flow models, the most direct impact is observed in producing the latent form of input data. Therefore, to ensure the generalization of results across various implementations of complete optical flow models, only the effects of the cost volume on the latent form are examined. To do such, two autoencoders are utilized in performing all experiments. The first autoencoder provides input images directly to the encoder module. The second produces a cost volume of the input data which is then provided to the encoder. In both models, the encoder is implemented as a base vision transformer architecture to ensure that the production of latent data is in a generic manner applicable to various optical flow implementations.

As discussed in Section 3.1, numerous works have shown that the process of creating a latent representation of input data directly influences overall model behavior, regardless of the target application. Because the direct model (Section 3.1.2) and the cost volume model (Section 3.1.3) produce the latent form of input data similar to that of an optical flow model, the following conclusions can be drawn regarding the effects of the cost volume on vision transformer-based optical flow models: cost volume use improves model accuracy and reduces inference generation time, but increases model training time and overall model size. As the latent representation used to form an optical flow hypothesis was examined in this thesis, rather than the optical

flow hypothesis itself, the exact improvement or detriment provided by cost volume use is unknown. However, given the careful attention to the construction of the latent space, these overarching conclusions are made with confidence as to their applicability to a complete optical flow model.

Supplemental work on this topic could provide additional insights regarding cost volume use to further benefit those researching vision transformer-based optical flow models. One potential area for further work is to train and evaluate each autoencoder model on non-synthetic datasets. In doing such, the results presented could be further verified for the non-synthetic domain. This additional work may benefit researchers developing models for "real world" applications such as autonomous driving. Another area for further work would be to repeat the described experiments with a complete vision transformer-based optical flow model, rather than autoencoders. While the overarching trends of the reported results are unlikely to change, this additional experiment would provide insights into how specific models are affected by cost volume use. With insights regarding complete optical flow models, and in consideration of specific applications, those designing complete vision transformer-based optical flow models can more precisely weigh the benefits and drawbacks of cost volume use.

In conclusion, the work presented in this thesis advances the optical flow domain by addressing a knowledge gap previously overlooked. Before this work, the effects of the cost volume were only examined for CNN-based optical flow models. This is despite the fact that the cost volume is widely used in vision transformer-based optical flow models as well. As this work has shown, backbone architectures are affected differently by cost volume use. Therefore, with the findings reported, researchers can now properly consider the various effects of the cost volume in designing vision transformer-based optical flow models.

Bibliography

[1] R. R. Beichel, "Chapter 14, motion analysis: Optical flow," 1997. [Online]. Available: https://user.engineering.uiowa.edu/~dip/Lecture/Motion2.html

[2] V. H. Phung and E. J. Rhee, "A high-accuracy model average ensemble of convolutional neural networks for classification of cloud image patches on small datasets," *Applied Sciences*, vol. 9, no. 21, 2019. [Online]. Available: https://www.mdpi.com/2076-3417/9/21/4500

[3] P. Kim, "Convolutional neural network," in *MATLAB Deep Learning*. New York, NY, USA: Apress, 2017, pp. 121–147. [Online]. Available: https://doi.org/10.1007/978-1-4842-2845-6_6

[4] Manu, "A simple overview of rnn, lstm and attention mechanism," Feb 2021. [Online]. Available: https://medium.com/swlh/a-simple-overview-of-rnn-lstm-and-attention-mechanism-9e844763d07b

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[6] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=YicbFdNTTy

[7] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015. [Online]. Available: https://openaccess.thecvf.com/content_iccv_2015/papers/Dosovitskiy_FlowNet_Learning_Optical_ICCV_2015_paper.pdf

[8] Z. Teed and J. Deng, "Raft: Recurrent all-pairs field transforms for optical flow," in *Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds. Cham, Switzerland: Springer International Publishing, 2020, pp. 402–419. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-58536-5_24

[9] Z. Huang, X. Shi, C. Zhang, Q. Wang, K. C. Cheung, H. Qin, J. Dai, and H. Li, "Flowformer: A transformer architecture for optical flow," in *Computer Vision – ECCV 2022*, S. Avidan, G. Brostow, M. Cissé, G. M. Farinella, and T. Hassner, Eds. Cham, Switzerland: Springer Nature Switzerland, 2022, pp. 668–685. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-031-19790-1_40

[10] S. Baker, S. Roth, D. Scharstein, M. J. Black, J. Lewis, and R. Szeliski, "A database and evaluation methodology for optical flow," in *2007 IEEE 11th International Conference on Computer Vision*, 2007, pp. 1–8. [Online]. Available: https://ieeexplore.ieee.org/document/4408903

[11] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A naturalistic open source movie for optical flow evaluation," in *Computer Vision – ECCV 2012*, A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 611–625. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-33783-3_44

[12] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2017/papers/Ilg_FlowNet_2.0_Evolution_CVPR_2017_paper.pdf

[13] J. Wulff, L. Sevilla-Lara, and M. J. Black, "Optical flow in mostly rigid scenes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2017/papers/Wulff_Optical_Flow_in_CVPR_2017_paper.pdf

[14] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2018/papers/Sun_PWC-Net_CNNs_for_CVPR_2018_paper.pdf

[15] S. Zhao, Y. Sheng, Y. Dong, E. I.-C. Chang, and Y. Xu, "Maskflownet: Asymmetric feature matching with learnable occlusion mask," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. [Online]. Available: https://openaccess.thecvf.com/content_CVPR_2020/papers/Zhao_MaskFlownet_Asymmetric_Feature_Matching_With_Learnable_Occlusion_Mask_CVPR_2020_paper.pdf

[16] S. Jiang, D. Campbell, Y. Lu, H. Li, and R. Hartley, "Learning to estimate hidden motions with global motion aggregation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021, pp. 9772–9781. [Online]. Available: https://openaccess.thecvf.com/content/ICCV2021/papers/Jiang_Learning_To_Estimate_Hidden_Motions_With_Global_Motion_Aggregation_ICCV_2021_paper.pdf

[17] A. Jaegle, S. Borgeaud, J.-B. Alayrac, C. Doersch, C. Ionescu, D. Ding, S. Koppula, D. Zoran, A. Brock, E. Shelhamer, O. J. Henaff, M. Botvinick, A. Zisserman, O. Vinyals, and J. Carreira, "Perceiver IO: A general architecture for structured inputs & outputs," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=fILj7WpI-g

[18] H. Xu, J. Zhang, J. Cai, H. Rezatofighi, and D. Tao, "Gmflow: Learning optical flow via global matching," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 8121–8130. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2022/papers/Xu_GMFlow_Learning_Optical_Flow_via_Global_Matching_CVPR_2022_paper.pdf

[19] A. Mitiche and P. Bouthemy, "Computation and analysis of image motion: A synopsis of current problems and methods," *International Journal of Computer Vision*, vol. 19, no. 1, pp. 29–55, Jul. 1996. [Online]. Available: https://doi.org/10.1007/bf00131147

[20] D. Liu, Y. Cui, W. Tan, and Y. Chen, "Sg-net: Spatial granularity network for one-stage video instance segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 9816–9825.

[21] Y. Cui, L. Yan, Z. Cao, and D. Liu, "Tf-blender: Temporal feature blender for video object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 8138–8147.

[22] D. Liu, Y. Cui, Y. Chen, J. Zhang, and B. Fan, "Video object detection for autonomous driving: Motion-aid feature calibration," *Neurocomputing*, vol. 409, pp. 1–11, 2020.

[23] Z. Cao, Z. Chu, D. Liu, and Y. Chen, "A vector-based representation to enhance head pose estimation," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021, pp. 1188–1197.

[24] L. Yan, Q. Wang, Y. Cui, F. Feng, X. Quan, X. Zhang, and D. Liu, "Gl-rg: Global-local representation granularity for video captioning," *IJCAI*, 2022.

[25] Z. Huang, T. Zhang, W. Heng, B. Shi, and S. Zhou, "Real-time intermediate flow estimation for video frame interpolation," in *Computer Vision – ECCV 2022*, S. Avidan, G. Brostow, M. Cissé, G. M. Farinella, and T. Hassner, Eds. Cham, Switzerland: Springer Nature Switzerland, 2022, pp. 624–642. [Online]. Available: https://link-springer-com.ezproxy.rit.edu/chapter/10.1007/978-3-031-19781-9_36

[26] X. Zhu, Y. Xiong, J. Dai, L. Yuan, and Y. Wei, "Deep feature flow for video recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[27] M. Zhai, X. Xiang, N. Lv, and X. Kong, "Optical flow and scene flow estimation: A survey," *Pattern Recognition*, vol. 114, p. 107861, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320321000480

[28] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," *Journal of Big Data*, vol. 8, no. 1, Mar. 2021. [Online]. Available: https://doi.org/10.1186/s40537-021-00444-8

[29] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[30] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015. [Online]. Available: https://arxiv.org/abs/1409.1556

[31] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf

[32] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 International Conference on Engineering and Technology (ICET)*, 2017, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/8308186

[33] H. Salehinejad, S. Sankar, J. Barfett, E. Colak, and S. Valaee, "Recent advances in recurrent neural networks," 2018. [Online]. Available: https://arxiv.org/abs/1801.01078

[34] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6795963

[35] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in *NIPS 2014 Workshop on Deep Learning, December 2014*, 2014. [Online]. Available: https://arxiv.org/pdf/1412.3555.pdf

[36] Z. Lin, M. Feng, C. N. dos Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, "A STRUCTURED SELF-ATTENTIVE SENTENCE EMBEDDING," in *International Conference on Learning Representations*, 2017. [Online]. Available: https://openreview.net/forum?id=BJC_jUqxe

[37] J. J. Gibson, *The Perception Of The Visual World*. Boston, MA, USA: Houghton Mifflin, 1950.

[38] D. Marr and T. Poggio, "A computational theory of human stereo vision," *Proceedings of the Royal Society of London. Series B, Biological sciences*, vol. 204, no. 1156, pp. 301–328, 1979. [Online]. Available: https://mcgovern.mit.edu/wp-content/uploads/2019/01/rspb.1979.0029.pdf

[39] B. K. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, no. 1, pp. 185–203, 1981. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0004370281900242

[40] Z. Tu, W. Xie, D. Zhang, R. Poppe, R. C. Veltkamp, B. Li, and J. Yuan, "A survey of variational and cnn-based optical flow techniques," *Signal Processing: Image Communication*, vol. 72, pp. 9–24, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0923596518302479

[41] A. Ranjan and M. J. Black, "Optical flow estimation using a spatial pyramid network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2017/papers/Ranjan_Optical_Flow_Estimation_CVPR_2017_paper.pdf

[42] J. Hur and S. Roth, "Optical flow estimation in the deep learning age," in *Modelling Human Motion: From Human Perception to Robot Design*. Cham, Switzerland: Springer International Publishing, 2020, pp. 119–140. [Online]. Available: https://doi.org/10.1007/978-3-030-46732-6_7

[43] J. Xu, R. Ranftl, and V. Koltun, "Accurate optical flow via direct cost volume processing," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2017/papers/Xu_Accurate_Optical_Flow_CVPR_2017_paper.pdf

[44] Y. Lu, Q. Wang, S. Ma, T. Geng, Y. Chen, H. Chen, and D. Liu, "Transflow: Transformer as flow learner," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2023.

[45] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 16 000–16 009. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2022/papers/He_Masked_Autoencoders_Are_Scalable_Vision_Learners_CVPR_2022_paper.pdf

[46] Z. Tong, Y. Song, J. Wang, and L. Wang, "VideoMAE: Masked autoencoders are data-efficient learners for self-supervised video pre-training," in *Advances in Neural Information Processing Systems*, A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, Eds., 2022. [Online]. Available: https://openreview.net/forum?id=AhccnBXSne

[47] D. Bank, N. Koenigstein, and R. Giryes, "Autoencoders," 2020. [Online]. Available: https://arxiv.org/abs/2003.05991

[48] X. Chu, Z. Tian, Y. Wang, B. Zhang, H. Ren, X. Wei, H. Xia, and C. Shen, "Twins: Revisiting the design of spatial attention in vision transformers," in *Advances in Neural Information Processing Systems*, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., 2021. [Online]. Available: https://openreview.net/forum?id=5kTlVBkzSRx

[49] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255. [Online]. Available: https://ieeexplore.ieee.org/document/5206848

[50] J.-B. Cordonnier, A. Loukas, and M. Jaggi, "On the relationship between self-attention and convolutional layers," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=HJlnC1rKPB

[51] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy, "Speed/accuracy trade-offs for modern convolutional object detectors," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2017/papers/Huang_SpeedAccuracy_Trade-Offs_for_CVPR_2017_paper.pdf

[52] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jegou, "Training data-efficient image transformers &; distillation through attention," in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 18–24 Jul 2021, pp. 10 347–10 357. [Online]. Available: https://proceedings.mlr.press/v139/touvron21a.html

[53] C. Pinard, "Flownetpytorch." [Online]. Available: https://github.com/ClementPinard/FlowNetPytorch

[54] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3354–3361. [Online]. Available: https://ieeexplore.ieee.org/document/6248074