FACE FLOW: CONSTRAINED OPTICAL FLOW FRAMEWORK FOR FACES

by

Muhannad Alkaddour

A Thesis Presented to the Faculty of the
American University of Sharjah
College of Engineering
in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science in
Mechatronics Engineering

Sharjah, United Arab Emirates

May 2020

Declaration of Authorship

I declare that this thesis is my own work and, to the best of my knowledge and belief, it does not contain material published or written by a third party, except where permission has been obtained and/or appropriately cited through full and accurate referencing.

Signature: Muhannad Alkaddour

Date: May 15th, 2020

Approval Signatures

We, the undersigned, approve the Master's Thesis of Muhannad Alkaddour

Thesis Title: Face Flow: Constrained Optical Flow Framework for Faces

Date of Defense: 14-May-2020

Name, Title and Affiliation	**Signature**
Dr. Usman Tariq Assistant Professor, Department of Electrical Engineering Thesis Advisor	
Dr. Abhinav Dhall Lecturer, Monash University Assistant Professor, Indian Institute of Technology Ropar Thesis Co-Advisor	
Dr. Mohammad Jaradat Professor, Department of Mechanical Engineering Thesis Committee Member	
Dr. Hasan Mir Associate Professor, Department of Electrical Engineering Thesis Committee Member	
Dr. Mohammad Jaradat Program Director Mechatronics Engineering Graduate Program	
Dr. Lotfi Romdhane Associate Dean for Graduate Studies and Research College of Engineering	
Dr. Sirin Tekinay Dean College of Engineering	
Dr. Mohamed El-Tarhuni Vice Provost for Graduate Studies Office of Graduate Studies	

Acknowledgements

First and foremost, I would like to thank my parents for their incomparable support for the duration of my studies and my life leading up to it, without whom this wouldn't have been possible, and my siblings for their love and all they have taught me. I also extend my appreciation to my advisor, Dr. Usman Tariq, who provided his support manifested in research guidance, continual patience, provision of necessary resources, including part of the FRG17-R44 research grant, and motivation. My thanks also go to Dr. Abhinav Dhall, whose continued involvement, research insight, and mentorship were fundamental to the completion of this dissertation.

I would also like to thank the directors and faculty of the Mechatronics Graduate Program Dr. Lotfi Romdhane, Dr. Mohammad Jaradat, Dr. Shayok Mukhopadhyay, and Dr. Rached Dhaouadi, who greatly supported me in my time at the mechatronics program and provided me with invaluable learning and research experiences, and to the mechatronics program as well for the sponsorship of my degree. Also critical to my academic proficiency were the faculty at the departments of Mathematics and Mechanical Engineering, who helped foster the academic passions that helped drive my work, including Dr. Mehdi Ghommem, Dr. Samir Emam, and Dr. Amjad Tuffaha, my undergraduate senior advisors.

Parallel to academic support are my friends in the department and my undergraduate degree, each to whom I owe for their wonderful friendship and support. Finally, I thank dearly my friends outside the university for their irreplaceable presence in my life despite the distance that forsakes us.

Dedication

To my loving parents, brothers, and sisters . . .

Abstract

In computer vision, models and methods for facial expression recognition are continually in development. Several models aim to describe the highly complex structure of different faces, which in turn allows researchers to digitally process the faces based on these models for various tasks. With the rise of deep learning in 2012, many works have since used deep networks to learn facial expressions through both static and dynamic images. One main source of information for dynamic features are optical flow algorithms. These algorithms predict, from a sequence of frames, where each pixel moves from one frame to the next. The recent optical flow algorithms based on deep learning employ frameworks that are similar to those of deep convolutional autoencoders. Faces have a peculiar structure. Hence, it makes sense to think that this optical flow should be constrained based on the physically allowable movements of facial features. Combined with robust facial alignment algorithms, good optical flow estimation for faces can be used as features for emotion recognition in robots. These vision-based techniques can aid the robot to better interact with humans by incorporating affect recognition in the interaction, adding a psychological element to it. To carry out this investigation, we propose to construct a dataset with ground truth optical flow generated by observing the deformation of face keypoints and their neighborhoods between any two consecutive face images. The dataset is then used to train the FlowNetS deep network specialized in learning optical flow, aiming to infer the constrained optical flow from a given pair of face images. The network trained with the dataset is then compared to other setups in the testing phase, and the overall results show that using the generated data during training helps the network predict better optical flow representations on face sequences. The results of this thesis can be used as a precursor to obtain and make use of the dynamic features for unsupervised learning of facial expressions, which are important in applications such as human robot interaction, online learning, and electronic consumer relationship management.

Keywords: ***Facial expression recognition, human-robot interaction, deep learning, optical flow, image warping***

Table of Contents

Abstract 6

List of Figures 9

List of Tables 11

1. Introduction 12

 1.1 Thesis Objectives 13

 1.2 Problem Formulation and Proposed Solution 14

 1.3 Thesis Organization 14

2. Background and Literature Review 16

 2.1 Topics in Facial Detection and Expression Analysis 16

  2.1.1 Face detection, modelling, and tracking 16

  2.1.2 Facial expression analysis 18

 2.2 Optical Flow and Motion Tracking 18

  2.2.1 Motion tracking algorithms 19

  2.2.2 Face optical flow 19

 2.3 Deep neural networks 20

  2.3.1 Artificial neural networks 20

  2.3.2 Convolutional neural networks 23

  2.3.3 Deep learning for videos and optical flow 27

  2.3.4 Autoencoders 29

 2.4 Performance Evaluation 31

3. Face Dataset Generation Using Image Warping 34

 3.1 Image Warping and Morphing 34

  3.1.1 Affine transformation and barycentric coordinates 34

  3.1.2 Resampling 36

 3.2 Dataset Generation Algorithm 38

4. Methodology 43

4.1 CNN Architecture and Training Details 43

4.1.1 CNN architecture: *FlowNetS* 43

4.1.2 Training details 47

4.1.2.1 Experimental setup 1: no cyclic loss 48

4.1.2.2 Experimental setup 2: with cyclic loss 49

4.1.2.3 Experimental setup 3: with cyclic loss, smoothness constraint, and average angular error 50

4.1.3 Evaluation of performance 51

5. Experimental Results and Discussion 52

5.1 Results for Setup 1: No Cyclic Loss 52

5.2 Results for Setups 2 and 3: With Cyclic Loss 54

6. Concluding Remarks 60

References 62

Vita 71

List of Figures

Figure 1: The perceptron, an example of the operations described in equation (3). 21

Figure 2: An example of a deep neural network with two hidden layers. 22

Figure 3: The top row (left to right) shows a single part of a convolution and max pooling operation ($f \in \mathbb{R}^{4\times4}$, $g \in \mathbb{R}^{3\times3}$). The bottom right image shows a common sequence of operations. 24

Figure 4: Top image shows a typical convolution in a CNN, and bottom shows a visualization of an entire CNN for digit recognition using the tool by Harley. 26

Figure 5: A convolutional autoencoder network architecture, showing the encoder and decoder layers 30

Figure 6: The angular error $AAE(V_1,V_2)$ for two vectors in $\mathbb{R}^2$ compared to their angle difference in the plane. 32

Figure 7: Sample images from the BP4D-Spontaneous showing different subjects from varying sequences. 39

Figure 8: Examples of automatically generated facial (noisy) ground-truth flow visualizations for pairs of images, along with their triangulations. Top row shows a flow field representation using streamlines, middle row shows the visualization using the color code shown. 42

Figure 9: FlowNetS architecture encoder layers, showing the separate input region, redirected layers, and the convolution sizes. 44

Figure 10: FlowNetS decoder layer, showing the upsampling operations and the intermediate predictions. 45

Figure 11: The top row of each example consists of two inputs X_1 and X_2. The second row shows the warped output (left) and the flow field visualization (right). 46

Figure 12: A pair of input images from BP4D-Spontaneous (top row); flow ground-truth and flow field generated from face-trained network, left to right (middle row); and flow field predictions from networks trained on chairs and Sintel, left to right (bottom row). The EPE for networks trained on faces, chairs, and Sintel are **1.16**, 2.78, and 1.35 respectively. This shows the effectiveness of the learnt representation from the proposed method. 53

Figure 13: Color coded optical flow prediction comparisons for the networks trained in each of the experimental setups, with examples to highlight how each network learns local motions. 56

Figure 14: Color coded optical flow prediction comparisons for the networks trained in each of the experimental setups, with examples to highlight how each network learns global motions. 58

Figure 15: More optical flow comparisons for all the trained networks, with mixed examples of both local and global motion. 59

List of Tables

Table 1: Summary of the loss weights, training/validation/testing splits, and learning rates used for each of the experiments described in sections 4.1.2.1-4.1.2.3. 51

Table 2: The average endpoint errors for each network described in subsection 4.1.2.1, trained and tested on all three datasets. 52

Table 3: Values of the different performance measures for each experimental setup. 54

Chapter 1. Introduction

Computer vision researchers have been interested in face biometrics for a variety of reasons. The tasks of describing, tracking, face recognition, and facial expression analysis are quite complex for computers compared to humans [1]. Research has been ongoing in the field for decades in attempts to narrow the gap between human level and machine competence. Over the last decade, recent advances in deep networks have unlocked a new era for potential advancements in this field. The use of deep convolutional neural networks (CNNs) to abstract high level features in images has been very promising. Many networks have outperformed previous algorithms in the same task, and amongst those algorithms is motion learning in video sequences.

Facial expressions are generated due to non-rigid movement in faces. From the perspective of automatic facial expression recognition (FER), the motion information has been well explored for the task of both micro and macro expression analysis. Optical flow information on faces can help characterize both micro and macro expressions, which are useful in expression recognition. A major motivation for using the motion information for FER is based on what is known as the facial feedback hypothesis [2], which, in summary, suggests that facial actions can both encode current emotions as well as *induce* or amplify emotions. An example of this would be that the furrowing of the brow could increase anger. It has also been demonstrated that some facial muscle movements are linked to the compound facial expression of negation [3]. Also, the relation between motion information extracted from the eyes and mouth has been studied in its association with the facial expressions of psychopaths [4]. Facial and head movements are also important in social contexts, such as head motion used to indicate particular social cues, or the famous twitching of the lip corners that suggest lying [5].

One primary motivation for this work is the application of facial optical flow in human-robot interactions. Facial motion information can be combined with facial detection and alignment in the preprocessing stage to help the robot separate multiple faces and infer the optical flow motion using the proposed approach. This is necessary in contexts where faces are not frontal or occluded, as is typically the case in human-robot interaction. The robust motion information can be used as input features for emotion classification, which aids the quality of the robot's interaction with humans.

Faces have a peculiar structure. Hence, in this work, we focus on learning optical flow specialized for faces which will attempt to constrain the algorithm to learn only lifelike expressions on faces, and in doing so we explore how well a CNN can perform in this task. We demonstrate that the proposed architecture will work well for faces compared to traditional optical flow algorithms. The results can serve as a precursor to designing motion-based features for supervised and unsupervised learning of facial expressions by drawing on existing research linking facial motion information to facial expression and emotion recognition.

1.1. Thesis Objectives

While there has been improvement with the recent advent of deep learning for computer vision, Chollet [6] explains that the field is still behind human level intelligence in many tasks. Although we have come a long way in face analysis, there still remain many open problems. Problems related to face dynamics are particularly interesting. In this work, we build on the existing deep optical flow algorithms and attempt to learn structures that are specialized to faces, as faces have a peculiar structure of motion. To achieve our objectives, we automatically generate optical flow data from existing facial sequences and to use the generated dataset as ground-truth to learn accurate constrained optical flow on faces using a convolutional autoencoder architecture.

We use the BP4D-Spontaneous dataset [7] consisting of videos of 41 participants with different facial expressions to generate the ground-truth optical flow between every pair of consecutive frames in the dataset. We then use this facial optical flow ground truth to train an optical flow-based convolutional autoencoder, FlowNetS [8], to learn optical flow specialized for facial motions, meaning that the motion learned should exhibit local coherency as would be expected on faces. To our knowledge, this is the first dataset for facial expression optical flow of this kind. We also modify the neural network by adding a cyclic loss to help the network reconstruct the latter image in the image pairs using the optical flow predicted by the network. We argue that adding this reconstruction in the learning framework improves the predicted optical flow.

1.2. Problem Formulation and Proposed Solution

The temporal information obtained from faces is important, since it provides essential information for facial expression recognition that is lost in static images [5]. Thus, to serve as a component for a comprehensive facial expression recognition system that can be used in computer vision applications such as real-time computer-human interaction, the motion that represents the temporal information must be accurately recognized. We emphasize that the motion that is predicted or recognized must be legal in the sense that it constitutes a realistic deformation of local facial features. For example, the region at both ends of the lips should deform along the same vertical direction in most cases. In light of the above, the problem can be formulated as follows: how can optical flow information, constrained to match realistic motion of facial features, be accurately inferred from consecutive face images to provide temporal information that is useful for facial expression and emotion recognition? The objective of this thesis, then, is to provide a solution to this problem that meets the criteria stated above and is robust and accurate enough to achieve state of the art results that can aid in facial expression recognition. The proposed solution can be described by the following tasks, defining the research objectives:

1. Generation of a ground-truth dataset for constrained optical flow across faces to allow training of the chosen network structure.

2. Construction of a convolutional neural network that can achieve reasonably high accuracy with the task of learning constrained (in the sense discussed above) optical flow across faces.

3. Quantitative evaluation of the predictions of the neural network based on optical flow performance evaluation techniques as well as qualitative evaluation, which can be done by studying how realistic the predictions are.

1.3. Thesis Organization

For the remainder of the thesis, the chapters are organized as follows. Chapter 2 provides a literature review on important topics related to FER, deep learning, optical flow, and their applications on analysis of faces. It also contains some preliminary theory of deep networks relevant to this thesis. Chapter 3 introduces some related concepts from image warping and feature extraction and describes the algorithm used for automatic generation of the face optical flow dataset. Chapter 4 includes the details of the convolutional autoencoder architecture and the different experiments conducted to learn the optical flow. Chapter 5 contains important results from the trained networks and a discussion of their implications, and chapter 6 concludes the research and provides suggestions for extensions to this work.

Chapter 2. Background and Literature Review

To gain a better understanding of how motion tracking for facial expression analysis can be used in convolutional neural networks, it is necessary to formulate a basis for each of the topics that go into the development of such algorithms. This is laying out the research that forms the theoretical and practical groundwork relevant to this project. First, work related to facial expression analysis and motion tracking are presented, followed by a survey of convolutional neural networks and different deep learning architectures, including those used to learn motion. We then present some of the general works related to optical flow and the challenges faced. In addition, we review current methods used for facial optical flow and some of their advantages and drawbacks.

2.1. Topics in Facial Detection and Expression Analysis

Before presenting the literature on convolutional neural networks, the face detection, modelling, and tracking methods are important fundamentals in the study of facial expression analysis. To understand the different classes of problems faced in facial expression analysis, face detection, modelling, and tracking are briefly introduced, in addition to a survey of the classical and more contemporary methods for facial analysis. Some of the difficulties and methods in facial recognition topics are also common to facial expression analysis, which is why we present the former prior to discussing the latter.

2.1.1. Face detection, modelling, and tracking. Face detection algorithms are very important in facial recognition, since they are the backbone in many of the more specialized tasks. There are several methods to use computer vision to detect faces and provide a suitable description of them, which are sometimes divided into two categories: feature-based or CNN-based recognition. Feature-based methods are usually based on extracting salient features from an image. One of the classical feature-based methods is the Viola-Jones algorithm [9], which is based on Haar-like features and a boosting technique to classify the salient points. Viola-Jones has been implemented in many tasks requiring face recognition as a prior to another task, or been modified for improvement in other instances, such as in [10]. Klemm *et al.* [11] demonstrate different methods for detection and description, such as using scale-invariant feature transform (SIFT) feature extraction and speeded up robust features (SURF) which uses Haar wavelets, and then a comparison of these local feature descriptors with global ones. Vinay *et al.* [12] describe an improved approach for SIFT and SURF feature detection on faces, and Kim *et al.* [13] also use SURF and support vector machines (SVM) to detect facial features. CNN-based face recognition is also more recently very popular, and there have been many successful CNNs for face recognition. One example by Bai *et. al.* [14] is based on what is known as ”generative adversarial networks” (GANs), which have proven very successful in surpassing the benchmarks for two datasets of faces in the wild, which usually contain images of a large number of low-resolution faces that makes recognition more difficult to achieve [14]. Other successful CNN-based face recognition implementations can be found in [15], [16], and more.

Popular face models are given in [5], such as the statistical shape model, active shape models, and active appearance models. Face models can also be constructed by dimensionality reduction using *principal component analysis* (PCA) to decompose the faces into the set of eigenfaces with highest variance [5], [17]. Li and Jain [5] also present nonlinear subspaces and manifold learning for facial recognition instead of linear subspaces, due to the complexity of face images. Zadeh *et al.* [18] use a *convolutional experts network* along with Tadas *et al.*'s [19] *constrained local neural fields* to track facial keypoints, with their implementation made widely available in the open source project *OpenFace* [20]. We use their work to track the keypoints during automatic dataset generation.

Other techniques are more involved with the alignment and orientation of faces prior to feature extraction. Jourabloo *et al.* [21] and Kazemi and Sullivan [22] deal with the problem of aligning faces when the head is in different orientations or poses. The problems of occlusion and pose are also addressed in [5]. Face motion tracking, which is essential to our research, is also a widely researched topic. Yu *et al.* [23] describe a novel approach using a particle filter and face/pose models for the task of facial motion tracking. Particle filters and Markov chains are used by Dornaika and Davoine in [24] for the same task.

2.1.2. Facial expression analysis. A survey of facial expression analysis can be found in [5], in which the authors describe common problems in facial expression analysis as well as classification of different descriptions for expressions, such as intensity, deliberate vs. spontaneity, and scene complexity. These factors usually influence the performance of facial expression recognition techniques. From a psychological viewpoint, Zheng *et al.* [25] describe how care should be taken in understanding human emotions, or affects, from data. They observe that many of the techniques that work well for deliberate expressions do not generalize as well in recognizing more spontaneous and subtle emotions. Combining static visual cues with audio and motion information, i.e. multimodal fusion, yields more information about the person's emotion than purely visual cues [26]. The authors provide detailed lists for emotion recognition systems and algorithms based on purely visual, purely auditory, and audiovisual-based techniques, as well as relevant databases. Corneanu *et al.* [27] provide an exhaustive survey of different facial expression analysis techniques, systematically classified as either RBG, 3D, thermal, or multimodal under several different tasks. For example, one of the lists is based on static 3D local, geometric, predesigned feature-extraction techniques. Dynamic techniques, particularly for feature extraction and expression analysis, are important for our research. They also offer valuable insight on emotion analysis based on facial expression from an evolutionary vantage point.

2.2. Optical Flow and Motion Tracking

Optical flow in images is used to estimate the motion of sets of pixels across images. To do this, the assumption of brightness constancy is used, meaning that a pixel with a certain intensity $I(x,y,t)$ is expected to have the same intensity after some time dt [25]. This can also be expressed as

$$\frac{dI(x,y,t)}{dt} = 0. \tag{1}$$

Denoting the velocity of the pixels by $V = [v_x, v_y]^T$ and taking the total derivative, we get

$$\begin{aligned} \frac{dI(x,y,t)}{dt} = \frac{\partial I}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial I}{\partial y}\frac{\partial y}{\partial t} + \frac{\partial I}{\partial t} = 0 \\ \frac{\partial I}{\partial x}v_x + \frac{\partial I}{\partial y}v_y + \frac{\partial I}{\partial t} = 0, \end{aligned} \tag{2}$$

which can be solved with another constraint equation, typically some energy minimisation functional as in [28], [29] for the velocities to obtain the optical flow. Gradient image operators such as those defined in [30] are commonly used to get the intensity derivatives.

2.2.1. Motion tracking algorithms. Many algorithms have been developed for motion tracking besides optical flow. Cremers *et al.* [31] estimate the motion by discretizing an image into segments and applying a Bayesian probabilistic approach to the motion along with geometric constraints. On the other hand, Goh and Vidal [32] exploit the fact that distinct motions are contained in different manifolds to precisely detect these distinct motions. For optical flow, other methods try to compensate for the different problems that might come up in the general optical flow framework. Chen *et al.* [33] develop a method using quaternions to deal with the possible inconsistency among the RGB channel intensities, Portz *et al.* [34] propose an algorithm to compute optical flow in blurred environments, and Porikli *et al.* [35] modify the optical flow algorithm to deal particularly with low frame-rate applications. Finally, Zappella *et al.* [36] present a comprehensive literature review and evaluation of motion tracking algorithms, including their advantages and applications. Some of those optical flow algorithms are specifically used to learn facial actions, such as in [37] and [38].

2.2.2. Face optical flow. More relevant to our topic is motion tracking methods used for faces, usually for facial expression analysis. One important work in learning optical flow for facial expressions by Snape *et al.* is *Face Flow* [39], which minimizes a proposed energy to learn the flow field for a sequence of frames consisting of facial expressions. Another highly relevant work is optical flow dataset generation done by Le *et al.* [40] who are also concerned with producing optical ground-truth data for general video sequences. According to them, there is very little prior work in the literature on how the performance of CNNs is influenced by optical flow datasets, and their main focus is that of non-rigid motion. Our work can be considered to be a contribution to the study of optical flow's effects on CNNs, with the difference being that we focus on facial datasets instead.

Koelstra *et al.* [37] use a ”dynamic texture-based approach” to estimate the facial rigid and nonrigid motion to learn facial action units. In [41], Allaert *et al.* propose a post-processing methodology for filtering noisy facial optical flow, which is useful when optical flow is used for facial expression recognition. They also suggest in [42] that motion-based approaches are useful for recognition of both macro and micro expressions. Hsieh *et al.* [43] use a probabilistic framework to estimate constrained optical flow, which accounts for illumination variation, to allow them to perform face recognition with just one training sample. Duthoit *et al.* [44] use the classical Lucas-Kanade method to find flow fields on faces and then, subsequently, for emotion recognition. Other ways of finding the optical flow on faces are model-based methods, such as in [45] which use a deformable face model along with uncertainty estimation using a Kalman filter to find the facial flow. Finally, a review and evaluation of different optical flow techniques specialized for facial expression recognition can be found in [46].

2.3. Deep neural networks

In machine learning, artificial neural network algorithms have been developed starting in the 40's by modelling neurons and then by successively developing single and multilayer learning [47]. This section will introduce the formulation of deep neural networks, specifically artificial neural networks, convolutional neural networks, and convolutional autoencoders, along with related works.

2.3.1. Artificial neural networks. Artificial neural networks are mathematical systems that define a function $\phi : U \mapsto V$, where U, V are the spaces in which the input and output data, respectively, are elements of. In general, this function ϕ depends on a collection of intermediate operations $g(a)$ and is a composition of these operations. Typically, for a multi-input neural network, the input vector $\boldsymbol{x} \in \mathbb{R}^m$ is defined at the ”input layer”. The second layer contains nodes which are functions of the input [48].

The weighted sum $z_j^{(1)} \in \mathbb{R}$ of $\boldsymbol{x}$ at node j in the second layer can be expressed as

$$z_j^{(1)} = (W_j^{(1)})^T \boldsymbol{x} + b_j^{(1)} \tag{3}$$

where $W_j^{(1)} = \begin{bmatrix} w_{1j}^{(1)} & w_{2j}^{(1)} \cdots & w_{mj}^{(1)} \end{bmatrix}^T \in \mathbb{R}^m$ denote the weights of node j, denoted by the subscripts, in hidden layer 1, denoted by the superscripts. The b_j^1 term represents a bias term added to the weighted sum. To find the value $a_j^{(1)}$ of this node, a nonlinear activation function $g : \mathbb{R} \mapsto \mathbb{R}$ is specified, and then $a_j^{(1)} = g(z_j^{(1)})$. This process for one node can be represented by Figure 1 [49], which shows a network, the perceptron, consisting of only one such sequence of operations.

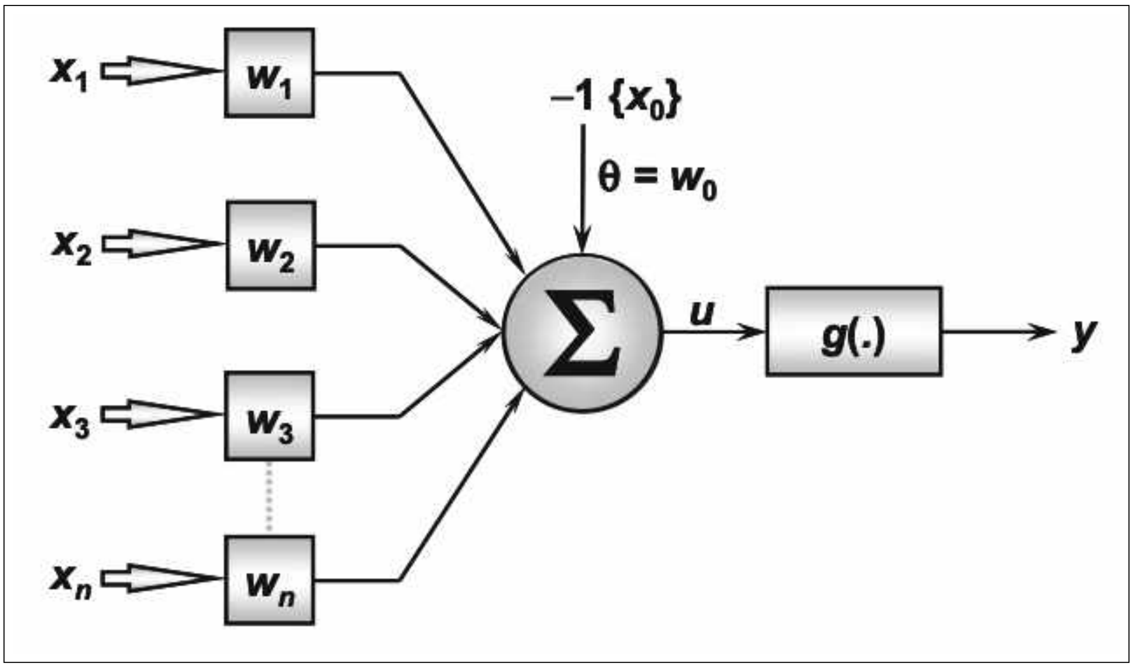

Figure 1. The perceptron, showing an example of the operations described in equation (3) (image source - [49]).

If there are a total of n_i nodes at this layer, then the input to the next layer is $\boldsymbol{a}^{(1)} = \begin{bmatrix} a_0^{(1)} & a_1^{(1)} \cdots & a_{n_i}^{(1)} \end{bmatrix}^T$. This combination of a nonlinear activation and bias allows the network to learn representations that are more complex than just a linear transformation [47]. This is continued for the subsequent layers i, possibly with a different number of nodes n_i for each layer, and the output vector for some layer i can then be expressed as

$$\boldsymbol{a}^{(i)} = g\left(\boldsymbol{W}^{T_{(i-1)}}\boldsymbol{a}^{(i-1)} + \boldsymbol{b}^{(i-1)}\right) \quad (4)$$

where $\boldsymbol{W}^{(i-1)} \in \mathbb{R}^{n_{i-1} \times n_i}$ contains the weight vector for every node j from the previous layer. An example ANN architecture with two hidden layers is shown in Figure 2 [49].

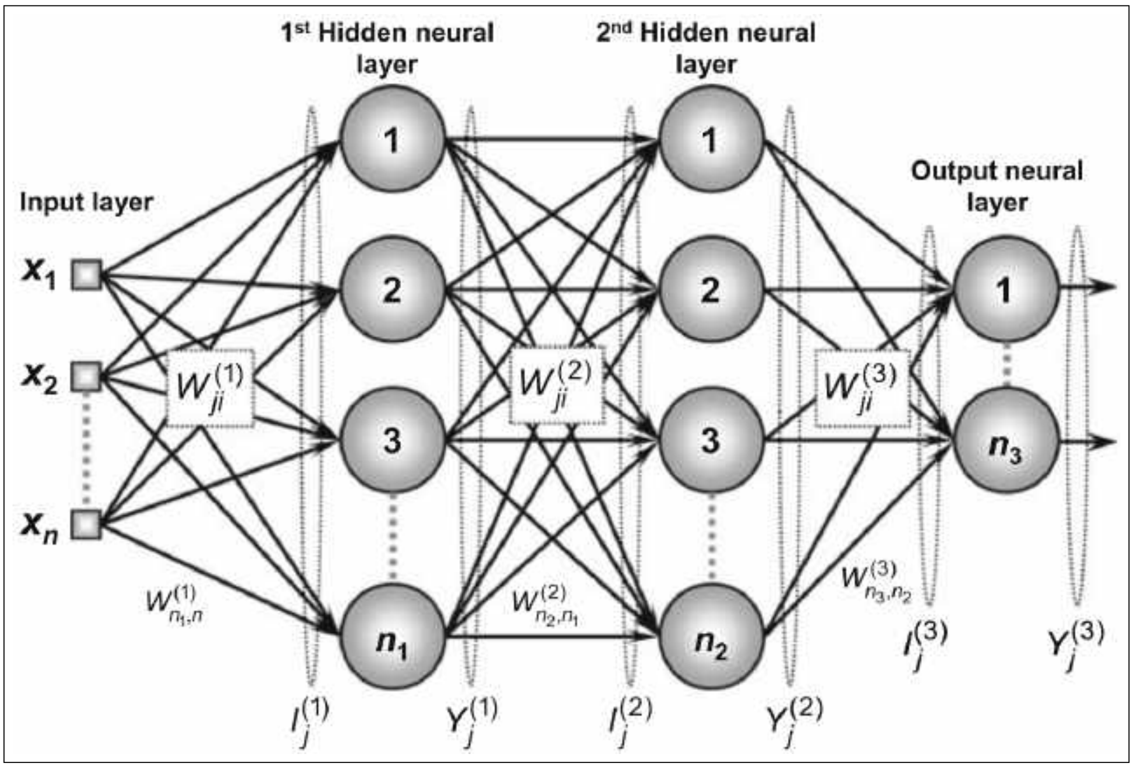

Figure 2. An example of a deep neural network with two hidden layers (image source - [49]).

The network output would be the output $\boldsymbol{a}^{(N)}$ of the activation function(s) of the last layer N, which could be a scalar or multidimensional. In a classification problem with k classes, for example, the output space V would be the set $\{0, 1, \cdots, k-1\}$.

To train a neural network to learn a function for a specific task, a suitable choice of architecture and parameters of the architecture, which are the weights and biases of the connections stored in matrices, should be chosen to accurately map the inputs to the outputs. This accuracy is determined by how the user trains the network. In a supervised problem, the inputs $\{\boldsymbol{x}^{(t)}\}_{t=0}^{M}$ to the network in a training set with M elements are paired with ground-truth quantities $\{\boldsymbol{y}^{(t)}\}_{t=0}^{M}$. The loss is defined to be a suitable measure of the network outputs compared to the ground-truth $\boldsymbol{y}$. An objective function which quantifies this loss is chosen such that it is large when the output of the network does not match the true output and small otherwise. This is similar to a least-squares regression problem in which parameters θ_i are selected to minimize the least-squares error (the objective function)

$$J(\boldsymbol{\theta}) = \frac{1}{2M} \sum_{m=0}^{M} \|\phi(\boldsymbol{x}^{(m)};\boldsymbol{\theta}) - \boldsymbol{y}^{(m)}\|_2^2 \tag{5}$$

where $\boldsymbol{\theta}$ contains all the weights and biases of the network, M is the number of training data, and $\phi(\boldsymbol{x}^{(m)};\boldsymbol{\theta})$ is the network output for the current $\boldsymbol{\theta}$. This is an optimization problem, and gradient-based methods such as gradient descent, stochastic gradient descent, and other variants are used to find the set of parameters $\boldsymbol{\theta}$ that minimize $J(\boldsymbol{\theta})$ [6]. Books such as [50] and [51] contain more details on different types of optimization techniques. The operations in the neural networks should be differentiable to allow optimization, and then an algorithm known as *backpropagation* is used to compute the gradients of the loss function for optimization [48].

2.3.2. Convolutional neural networks. The problem in training these kinds of classical neural networks on datasets comprised of images, such as our optical flow dataset, lies in the fact that the number of parameters is simply too large to learn. For example, a single greyscale image $I \in \mathbb{R}^{h\times w}$ will require an operator with $h \times w$ distinct parameters to connect to a single node in one hidden layer [6], rendering the computation infeasible. Instead, the weighted-sum operations described earlier are replaced by other operations which reduce the number of parameters required. 2D convolutions $(*)$ are very commonly used, which are defined for functions $f : \mathbb{R}^2 \mapsto \mathbb{R}^{h\times w}$ and $g : \mathbb{R}^2 \mapsto \mathbb{R}^{p\times q}$ to be

$$f * g = \sum_{a=-h}^{h} \sum_{b=-w}^{w} f(a,b)g(x-a, y-b). \tag{6}$$

If $h = w$ and $p = q$ in the equation (6), then the convolution $f * g \in \mathbb{R}^{(h-q+1)\times(h-q+1)}$ [52]. It is clear that in the special case where $dim(f) = dim(g)$, then $f * g \in \mathbb{R}$. It is more standard to use the notations n_a, n_b, and n_c for the dimensions of f, g, and $f * g$, and we will adopt them from here onwards. A visual representation of this operation is shown at the top left in Figure 3 [52]. f and g are square here of sizes 4×4 and 3×3, respectively, and the operation is visualized by starting from the upper left corner, taking a window of size $min(n_a, n_b)$, and taking the sum of the elementwise product of this window. This process is repeated for all such possible windows, which in this case is 4, resulting in a 2×2 image, since $n_a - n_b + 1 = 4 - 3 + 1 = 2$. When $dim(g) < dim(f)$, it is called a *kernel* by many texts, and serves as a basis for different image processing techniques. For example, horizontal edge-detectors can be 3×3 kernels with known entries and can be convolved with any image to produce a new image with pixels at the horizontal edges brighter than other pixels [1].

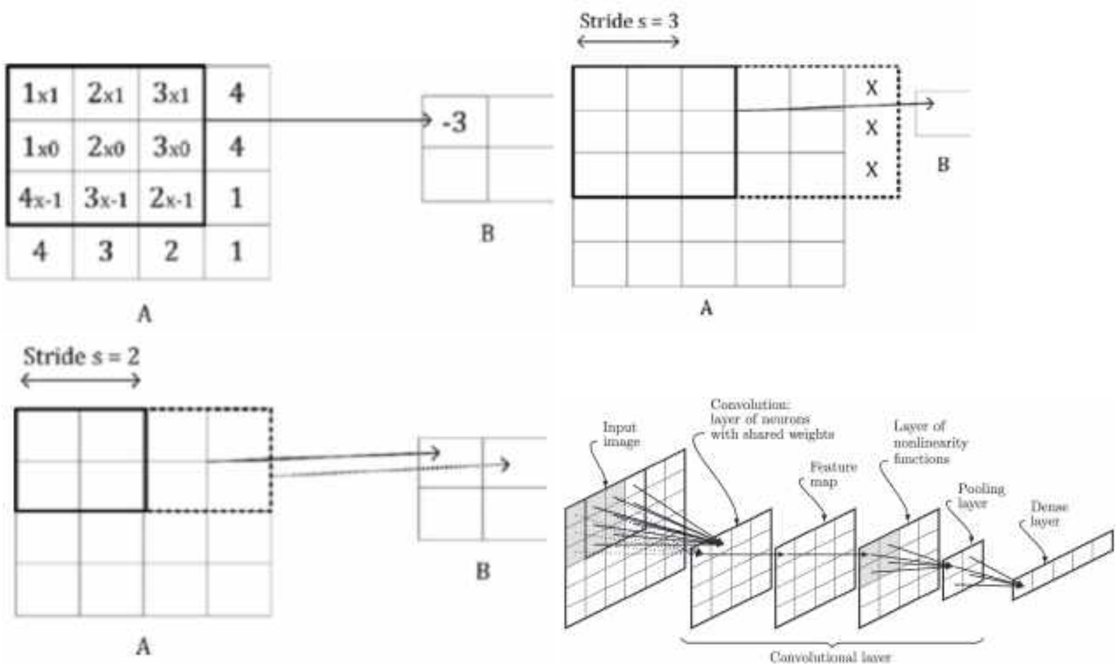

Figure 3. The top row (left to right) shows a single part of a convolution and max pooling operation ($f \in \mathbb{R}^{4\times4}$, $g \in \mathbb{R}^{3\times3}$) (image source - [52]). The bottom right image shows a common sequence of operations (image source - [48]).

In this first example, the stride s of the moving window, which is the number of cells it shifts after every computation, is 1. Increasing the stride to some s_0 in a convolution would result in the window sliding to the right or down by s_0 units after each computation, visualized in the top right of Figure 3 for $s = 3$. Higher strides decrease the output size to $\lfloor \frac{n_a - n_b}{s} + 1 \rfloor$ [52]. Since the dimensions of f do not accommodate the window with stride $s = 3$, only the first window can be computed, and the output is a scalar. In fact, $\lfloor \frac{5-3}{3} + 1 \rfloor = \lfloor \frac{5}{3} \rfloor = 1$. It is not too difficult to see that the addition of a stride decreases n_c, since

$$\begin{aligned}\left\lfloor \frac{n_a - n_b}{s} + 1 \right\rfloor &= \left\lfloor \frac{n_a - n_b}{s} \right\rfloor + 1 \\ &\leq \frac{n_a - n_b}{s} + 1 \\ &\leq n_a - n_b + 1,\end{aligned} \tag{7}$$

since $\left\lfloor \frac{p}{q} + k \right\rfloor \leq \left\lfloor \frac{p}{q} \right\rfloor + k$ when $k \in \mathbb{Z}$ and $s > 1$. To circumvent this, a *padding* of size p may be added to the image, which expands the input dimensions from $n_a \times n_a$ to $(n_a + 2p) \times (n_a + 2p)$ by adding zeros next to and above every element in the boundary [52]. The output dimension is then $n_c = \left\lfloor \frac{n_a + 2p - n_b}{s} + 1 \right\rfloor$, and one can choose p depending on the desired n_c. For example, to preserve the input dimensions, we can find p by setting the actual dimension and desired dimension equal and solving for p

$$\begin{aligned}\left\lfloor \frac{n_a + 2p - n_b}{s} + 1 \right\rfloor &= n_a - n_b + 1 \\ \left\lfloor \frac{n_a + 2p - n_b}{s} \right\rfloor &= n_a - n_b.\end{aligned} \tag{8}$$

which is satisfied by setting $p = \left\lceil \frac{1}{2}(s-1)(n_a - n_b) \right\rceil$. If $p \in \mathbb{Z}$, it is easy to check by substitution in equation (8). If $p \notin \mathbb{Z}$, i.e. both $s - 1$ and $n_a - n_b$ are odd, then it still works. In fact, let $s - 1 = 2k_1 + 1$ and $n_a - n_b = 2k_2 + 1$ for some $k_1, k_2 \in \mathbb{Z}$, both odd. Then

$$\begin{aligned}p = \left\lceil \frac{1}{2}(2k_1 + 1)(2k_2 + 1) \right\rceil &= \left\lceil \frac{1}{2}\left[4k_1k_2 + 2(k_1 + k_2) + 1\right] \right\rceil \\ &= \left\lceil 2k_1k_2 + k_1 + k_2 + \frac{1}{2} \right\rceil \\ &= 2k_1k_2 + k_1 + k_2 + 1.\end{aligned} \tag{9}$$

Substituting equation (9) into equation (8) and with $n_a - n_b$, s as defined, we get

$$\begin{aligned}\left\lfloor \frac{n_a + 2p - n_b}{s} \right\rfloor &= \left\lfloor \frac{2(2k_1k_2 + k_1 + k_2 + 1) + 2k_2 + 1)}{2(k_1 + 1)} \right\rfloor \\ &= \left\lfloor \frac{2k_2(k_1 + 1) + k_1 + 1}{k_1 + 1} + \frac{1}{2(k_1 + 1)} \right\rfloor \\ &= 2k_2 + 1 = n_a - n_b,\end{aligned} \tag{10}$$

as desired.

In CNNs, it is typical to convolve a color image $I \in \mathbb{R}^{h \times w \times 3}$ with a number n_k of filters for every layer. One of the motivations in using convolutions is that the choice of kernel elements translate to outputs with different features, and are referred to as *feature maps*. The convolution so far described can be extended by applying each of the n_k filters onto the color image, and the output feature map is the concatenation of the results of each convolution [52]. A nonlinear activation function is also used in CNNs on the elements of the feature maps, such the ReLU activation $g(x) = max(0,x)$ for every element x. An example of a layer and visualization of a CNN for digit recognition is shown in Figure 4.

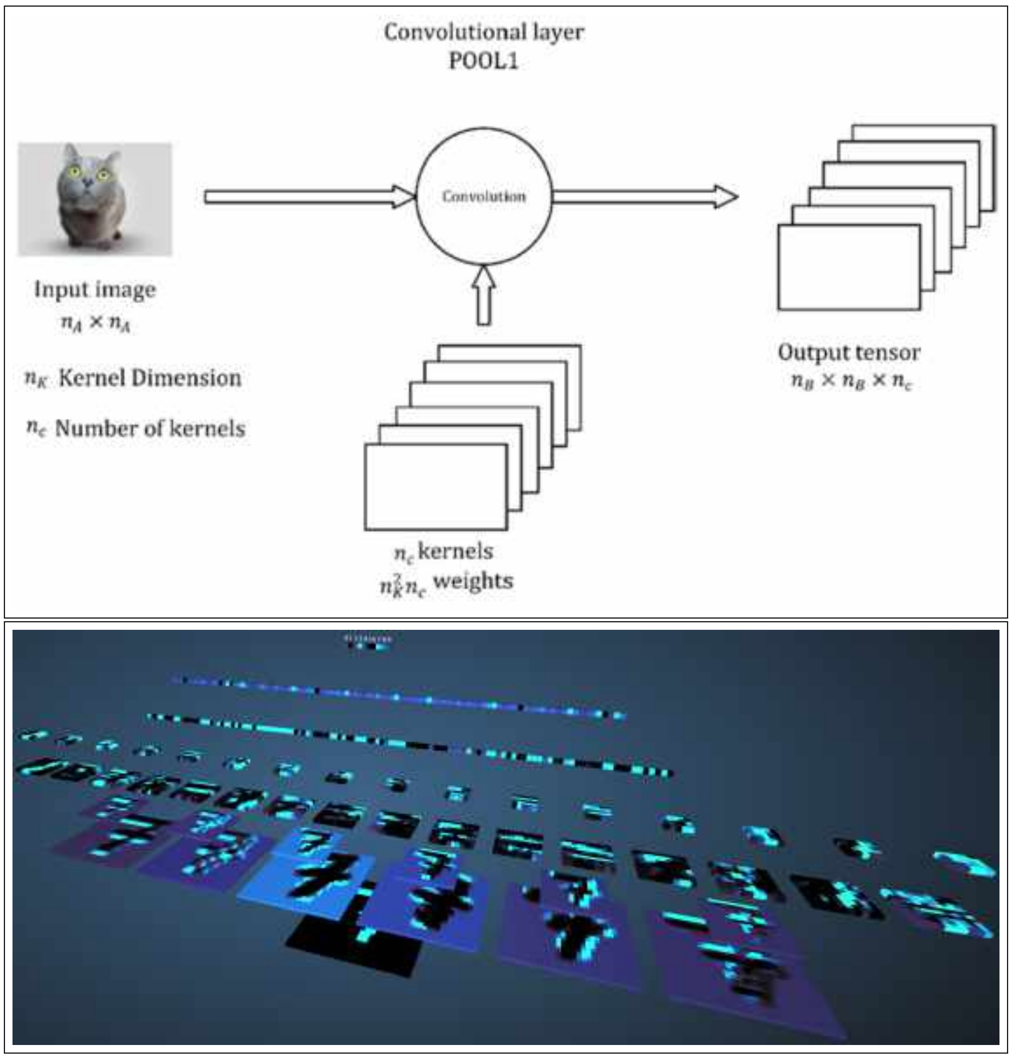

Figure 4. Top image shows a typical convolution in a CNN (image source - [52]), and bottom shows a visualization of an entire CNN for digit recognition using the tool by Harley (tool source - [53]).

Analogous to the ANN, the CNN's weights to be optimized are the elements of the filters, and it can be shown that they are significantly less than a fully connected network, as in a typical ANN case. The weights are chosen such that the feature maps learn useful abstractions depending on the learning task, which helps provide as close outputs as possible. The loss functions should be suitably chosen depending on the learning problem.

CNNs gained popularity with the ground-breaking success of AlexNet in 2012 [54], which inspired a large number of CNN architectures such as in [55], [56], [57], and many more. The Universal Approximation Theorem states that any continuous function over a compact set in $\mathbb{R}^m$ can be approximated, up to any desired accuracy, by any arbitrary activation function g and a finite number N of nodes in a single-layer neural network [58]. Analogous results for convolutional neural networks have been proven as well, and the reader is referred to [59] and [60] for the exact statements. For our purposes, these results imply the existence of a suitable convolutional neural network for the task at hand, although they are inconclusive in terms of computational feasibility.

2.3.3. Deep learning for videos and optical flow. Although convolutional neural networks are known to allow transfer learning, which means that the same network architecture can be used to learn tasks different than the one it was originally trained on, there are several network architectures proposed specifically for learning of motion across frames in videos. In [61], Janai *et al.* use a CNN architecture known as a Siamese Triplet Network is used to predict the motion of objects by training on three different patches X_1, X_2, and X_3, where X_1 and X_2 are two subsequent frames of an object in motion and X_3 is a random sample of something different than the object. A distance metric $d(x,y)$ is learned by minimizing $d(X_1,X_3)$ and maximizing $d(X_1,X_2)$ during training. Alternatively, in [62], Datta *et al.* use the Siamese Triplet Network to learn facial representations to identify whether or not two faces are similar by comparing bounding boxes of faces in every 10th frame to track faces. Although both these works are not strictly in the same domain as ours, their data mining procedure and ground truth labelling in an unsupervised manner is a shared attribute, as we also aim to generate ground truth datasets algorithmically. The latter [62] is also a precedent in using a CNN to detect motion, which contain possible methods that we could draw inspiration from in our work for facial motion tracking. Other works such as [63], [64], and [65] also use novel CNN architectures to specifically learn facial expressions.

Other networks learn to track motion in dynamic environments. Zhang *et al.* [66] demonstrate a CNN which learns micro facial expressions in long videos, and Silva *et al.* [67] use static and dynamic inputs of video frames, with their respective results averaged at the end for action recognition. Sun *et al.* [68] use the pyramid-structure CNN architecture *PWC-Net* for optical flow prediction, which we use in this work to test on the face optical flow dataset as a benchmark implementation and compare with our performance.

With the surge and success of deep learning applications this decade, there has also been a rise in using convolutional neural networks to learn optical flow, beginning with the seminal work of Fischer *et al.* [8] with their *FlowNet* CNN architecture. Building on the success of FlowNet, *FlowNet2.0* [69] was introduced a few years later to improve performance by stacking networks, scheduling the training data, and learning small-motion datasets. For our experiments, we use the *FlowNetS* architecture adapted from [8]. By demonstrating how we can adapt FlowNetS to perform well on datasets consisting of only faces, we can later improve even further by incorporating refinements in the same way the FlowNet developers have, as well as solutions proposed to handle difficulties encountered in optical flow estimation.

While FlowNet is one of the more popular optical flow deep learning architectures, several other architectures have since been proposed to deal with certain challenges. Janai *et al.* [61] deal with the problem of unsupervised learning of optical flow in occluded settings by considering a triplet instead of a pair of frames and a 'photometric' loss to handle the occlusions. Ren *et al.* [70] also use the photometric loss to learn optical flow, with their focus being how well image warping techniques can be used for unsupervised training on optical flow datasets. Meister *et al.* [71] build on these concepts by applying their own loss function to improve results of unsupervised learning of optical flow, as well as learning the flow in the forward and reverse directions as in [61].

In their seminal work , Zhu *et al.* [72] develop the *cycleGAN*, which is a type of generative adversarial network, that implements a cyclic loss function which is used as a metric to evaluate the network's prediction as compared with one of the inputs. This loss function is also used in the context of optical flow learning. Yu *et al.* [73] use this cyclic loss, which they dub "warp loss", to train a Flownet architecture for optical flow learning. It is also adapted by Lai *et al.* [74] also in the context of optical flow but using instead a generative adversarial network. Both latter architectures use a differentiable spatial transformer layer with learnable parameters, adapted from Jaderberg *et al.* [75].

Finally, we mention a few implementations of deep learning to the learning of *microexpressions* using optical flow. Liong *et al.* [76] exploit the optical flow in a video sequence between the frame with the highest intensity, called the apex, and each of the rest of the frames, using the optical flow as input to learn microexpressions by a deep network. Just as we do, they assume small motion across short times between frames as well as brightness constancy. It also serves as a precedent for the effectiveness of facial optical flow as features for expression recognition, which is one of the motivations of our work. Taking a different approach, Li *et al.* [77] also use facial optical flow to learn microexpressions, but they obtain their flow features by first passing the facial images through the FlowNet2.0 architecture which we referred to earlier and then using a support vector machine for microexpression detection. Peng *et al.* [78] and Liu *et al.* [79] also use a deep CNN to learn microexpressions with optical flow ground-truth computed based on a modified version of the traditional Lucas-Kanade algorithm from [80].

2.3.4. Autoencoders. One interesting class of neural networks are autoencoders. Autoencoders are used to accept an input image and reconstruct it at the output. To avoid trivial reconstruction (identity mapping), the dimensions in the hidden layers are generally less than the dimension of the inputs, and so with the goal of reconstructing the input image, the autoencoder attempts to learn the most significant features of the image [81]. Autoencoders can also employ weight sharing and sparse connectivity in CNNs, and are also referred to as convolutional autoencoders [82]. Figure 5 shows a typical structure of a convolutional autoencoder [82] used to reconstruct the input images.

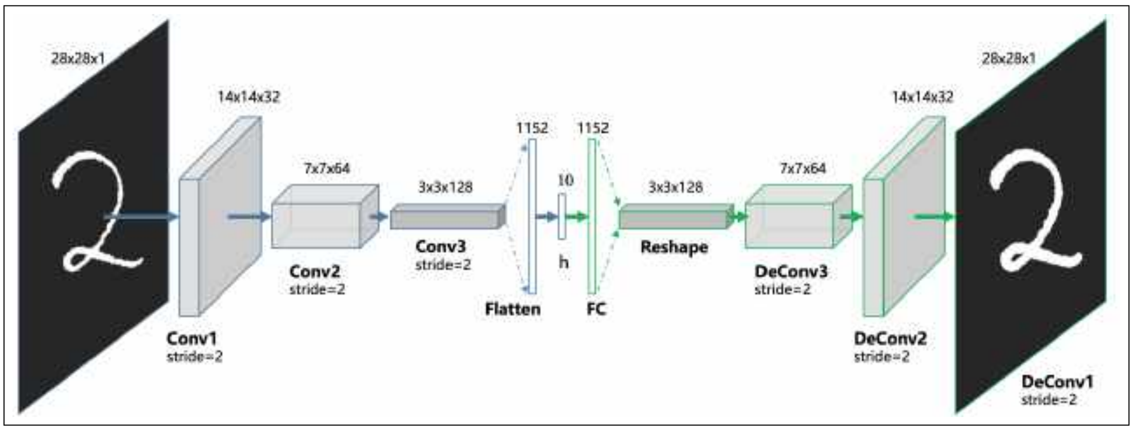

Figure 5. A convolutional autoencoder network architecture, showing the encoder and decoder layers (image source - [82]).

The encoder downsamples the input image into the latent space representation, which is at the intermediate fully-connected layer, before being passed as a latent input to the decoder, whose output is an image. The objective function would measure the error between the reconstructed image and the input image, in this case. The FlowNetS architecture that we use in this work is a type of convolutional autoencoder, with one difference being that the optical flow field is reconstructed instead of the input image at the output and compared with the ground-truth flow field of the two input images.

There are several variants of convolutional autoencoders in the literature. For example, Alejandro *et al.* [57] describe a “stacked hourglass” architecture used for the task of pose estimation, achieved by detecting the motion of the limbs in addition to the posture. This architecture is composed of downsampling and pooling layers (reducing the image size) followed by upsampling layers to make a single hourglass. The idea behind this type of architecture is that the successive downsampling and upsampling allows the network to learn features at different resolutions [57]. Other networks have also used hourglass-type architectures, such as in [83], [84], and [85]. This practice of combining high and low level information during training has been shown to be effective in such architectures.

2.4. Performance Evaluation

To evaluate the performance of our network in learning optical flow representations for faces, we look to quantitative methods to measure the performance. Baker *et al.* [86] define two such measures, which have become standard in later works on optical flow, termed the *average endpoint error* (EPE) and *average angular error* (AAE). In addition to these standard flow errors, the *structural similarity index* (SSIM) [87] is a widely-used metric to quantify thow similar two images are, which can be used to quantitatively compare the image warped by the predicted flow with the actual image. A variant of SSIM for video sequences, the V-SSIM, can be found in [88].

Consider two optical flow fields $\boldsymbol{U}_1$, $\boldsymbol{U}_2 \in \mathbb{R}^{h\times w\times 2}$, with the ijth element given by $[\boldsymbol{U}_1]_{ij} = (V_{ij})_1 = (u_{ij}, v_{ij})_1$. The average endpoint error between $EPE(\boldsymbol{u}_1, \boldsymbol{u}_2)$ is

$$EPE(\boldsymbol{U}_1, \boldsymbol{U}_2) = \frac{1}{h\times w}\sum_{i=1}^{h}\sum_{j=1}^{w}\left\|(u_{ij}, v_{ij})_1 - (u_{ij}, v_{ij})_2\right\|_2, \tag{11}$$

which is the pixelwise Euclidean norm for each optical flow vector [86]. This is one of the most commonly used measures in flow field comparisons, and is used in FlowNetS to construct the loss function for the neural network.

The other flow field measure, the average angular error, is similar to the EPE but it instead measures pixelwise angular difference rather than flow magnitude. Specifically, for one pair of flow vectors, it is defined to be the average of the angle between the two flow vectors in the ijth pixel on the $z = 1$ plane, denoted by $(V_{ij})_1 = (u_{ij}, v_{ij}, 1)_1$ and $(V_{ij})_2 = (u_{ij}, v_{ij}, 1)_2$. Dropping the ij subscript, the angle θ between V_1 and V_2 can found using

$$\begin{aligned} \|V_1 \times V_2\|_2 &= \|V_1\|_2\|V_2\|_2 \sin\theta \\ V_1 \cdot V_2 &= \|V_1\|_2\|V_2\|_2 \cos\theta. \end{aligned} \tag{12}$$

By averaging equation (12) over all pixels, we get

$$AAE(\boldsymbol{u}_1, \boldsymbol{u}_2) = \frac{1}{h\times w}\sum_{i=1}^{h}\sum_{j=1}^{w}\arctan\left(\frac{\left\|(u_{ij}, v_{ij}, 1)_1 \times (u_{ij}, v_{ij}, 1)_2\right\|_2}{(u_{ij}, v_{ij}, 1)_1 \cdot (u_{ij}, v_{ij}, 1)_2}\right), \tag{13}$$

where the operators $(\cdot)$ and $(\times)$ denote the usual vector dot and cross products, respectively. The $AAE(V_1,V_2)$ in homogeneous coordinates for two vectors $V_1,V_2 \in \mathbb{R}^2$ and in the plane, denoted by α, is shown in Figure 6.

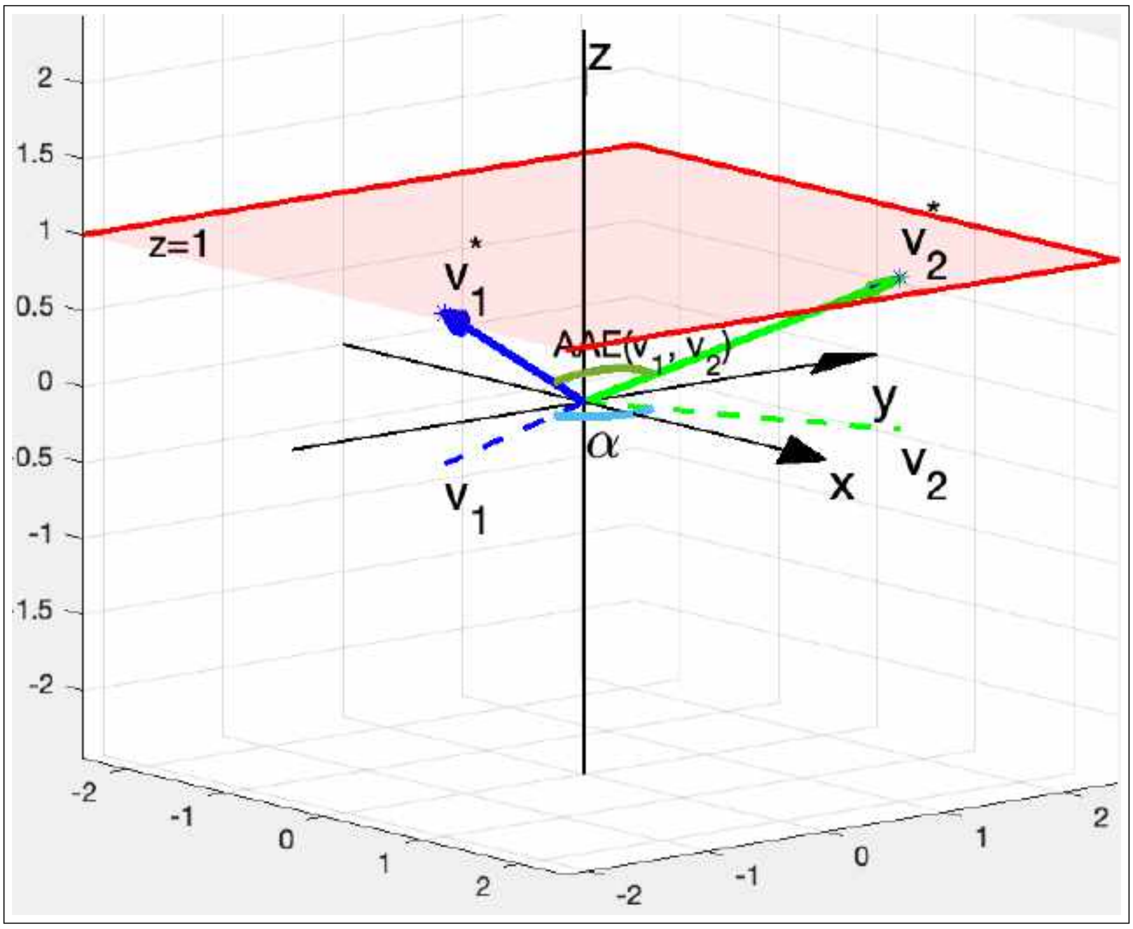

Figure 6. The angular error $AAE(V_1,V_2)$ for two vectors in $\mathbb{R}^2$ compared to their angle difference in the plane.

It is defined this way to more accurately represent the actual difference between two flow vectors in computer vision due to the projective nature of images. The EPE and AAE are later used to construct the loss functions that we use in our experimental setups to train our deep network. They are also used to measure the performance on our test sets by comparing predicted flow fields with the generated ground-truth flow fields.

In addition to the flow performance measures, we will also compare the result of the image deformed by the predicted flow field to the actual image. For two input images X_1,X_2, we use the predicted flow field to deform X_1 to $\hat{X}_2$ and compare $X_2,\hat{X}_2$

using the structural similarity index. Two images X and Y are compared by luminance, contrast, and structure, which are given by [87]

$$\begin{aligned} l(X,Y) &= \frac{2\mu_X\mu_Y + C_1}{\mu_X^2 + \mu_Y^2 + C_1} \\ c(X,Y) &= \frac{2\sigma_X\sigma_Y + C_2}{\sigma_X^2 + \sigma_Y^2 + C_2} \\ s(X,Y) &= \frac{\sigma_{XY} + C_3}{\sigma_x\sigma_Y + C_3}, \end{aligned} \tag{14}$$

where $\mu_X, \mu_Y, \sigma_X, \sigma_Y$ are the discrete mean and standard deviations of the images X, Y and σ_{XY} is the correlation coefficient between X and Y. $C_i = (K_iL_i)^2$ for small K_i, and L_i is equal to the range of possible pixel values in the image, which are used to prevent division by zero [87]. By defining $C_3 = C_2/2$ and taking the product of the three quantities in equation (14), the SSIM between X and Y is

$$SSIM(X,Y) = \frac{(2\mu_X\mu_Y + C_1)(2\sigma_{XY} + C_2)}{\left(\mu_X^2 + \mu_Y^2 + C_1\right)\left(\sigma_X^2 + \sigma_Y^2 + C_2\right)}. \tag{15}$$

We also use the standard L_1 and L_2 norms for further quantification of the image results.

Chapter 3. Face Dataset Generation Using Image Warping

In this chapter, the preliminaries for the methods we use in data generation are first introduced, followed by a detailed account of the procedure used to generate the ground-truth optical flow based on the BP4D-Spontaneous face dataset from [7].

3.1. Image Warping and Morphing

In addition to convolutional neural networks, basics of image warping and morphing are a prerequisite to our plan to generate a ground-truth constrained optical flow dataset for faces. Only the relevant essentials that we use in our dataset generation algorithm are discussed in this subsection, and the reader is referred to [89], [90], and [91] for a more complete treatment.

Image warping is the process of generating a target image from a source image by applying some sort of transformation on the source image, which is usually a bijective mapping with a continuous inverse (or, to use the topological term, a homeomorphism) [90]. Several transformations exist, such as contractions, expansions, and isometries, which respectively shorten, extend, or preserve the distance norm between any two points in the target image. An operation $T : U \subset \mathbb{R}^n \mapsto W \subset \mathbb{R}^n$ is a contraction, expansion, or isometry if $\forall x, y \in \mathbb{R}^n$,

$$\|T(y) - T(x)\| \leq \alpha \|y - x\| \tag{16}$$

where $|\alpha| < 1$ for a contraction, $|\alpha| > 1$ for an expansion, and $\alpha = 1$ for an isometry [92]. One class of such functions of interest in this work is affine transformations, which map lines to lines and, by extension, convex sets to convex sets.

3.1.1. Affine transformation and barycentric coordinates.

A *barycenter* of a collection of K points $\{v_i\}_{i=0}^{K}$ in $\mathbb{R}^2$ is the point $y \in \mathbb{R}^2$ associated with $\{\lambda_i\}_{i=0}^{K}$ such that

$$\begin{gathered} y = v_0 + \sum_{i=0}^{K} \lambda_i (v_i - v_0) \\ \sum_i \lambda_i = 1, \end{gathered} \tag{17}$$

for any choice of $v_0 \in \mathbb{R}^2$ [92]. This is closely related to the *convex combination* of the set of points $\{v_i\}_{i=0}^{K}$, which is defined as

$$\sum_{i=0}^{K} \lambda_i v_i \tag{18}$$

with the λ_i as in equation (17). When $K = 1$, all the possible convex combinations yield the points on the line segment between v_0 and v_1, and for $K = 2$, all possible convex combinations yield all the points in the closure of the triangle with vertices $\{v_0, v_1, v_2\}$ [92]. In other words, by taking all convex combinations of $\{v_0, v_1, v_2\}$ on the plane satisfying $\lambda_0 + \lambda_1 + \lambda_2 = 1$, we can obtain all points in the interior and boundary of a triangle. This motivates the idea of barycentric coordinates, which is treated in more detail in [92] and [90]. For any given point $v \in \mathbb{R}^2$, if it can be written as a convex combination of the vertices of a triangle, the λ_is in the combination are the *barycentric coordinates* of v relative to the set $\{v_0, v_1, v_2\}$ and are unique [92]. This allows the determination of all points in the interior of a triangle.

Affine maps are a special class of warping maps that preserve barycentric coordinates for all points in a given triangle. In some works, such as [92], they are *defined* as such. Affine maps can in general be represented by

$$\begin{aligned} T &: U \mapsto W \\ T(x) &= Ax + b \end{aligned} \tag{19}$$

where $U, W \subset \mathbb{R}^2$, $A \in \mathbb{R}^{2\times 2}$, and $b \in \mathbb{R}^2$. They preserve several geometric quantities, one which is of interest in this work is the mapping of n-gons to n-gons, and in particular, triangles to triangles [92]. A sufficient condition for uniquely defining an affine map is a priori knowledge of how three points are mapped. Thus, if we know where a triangle's vertices $V = \{v_0, v_1, v_2\}$ are mapped to, we can define the affine map that sends these vertices to their values. Moreover, since the map preserves barycenters, we can also determine where every point in the interior of the triangle formed by the $v_i \in V$ is sent to as well by taking the *same* convex combinations of the mapped vertices. Given a triangle with these vertices, we can infer an affine map by defining the homogeneous coordinates $v^* \in \mathbb{R}^3$, $v^* = [v^T\,1]^T$, which represents the set of all points in R^3 on the plane $P = \{(x_1, x_2, x_3) | x_3 = 1\}$. By applying a linear transformation A^* that maps points in P to other points in P, we can accomplish both translation and the other warping performed by A and b from equation (19). The affine map can be written, then, as

$$\begin{gathered} T^* : P \mapsto P \\ T^*(x^*) = \begin{pmatrix} A & b \\ 0_{2\times 2} & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix}. \end{gathered} \tag{20}$$

By defining A^* to be the matrix in equation (20) and known that each $v_i^* \in V^*$ is mapped to some known $w_i^* \in W^*$, we can write the equations mapping the v_i^* to w_i^* to get

$$\begin{aligned} A^* \begin{pmatrix} v_1^* & v_2^* & v_3^* \end{pmatrix} &= \begin{pmatrix} w_1^* & w_2^* & w_3^* \end{pmatrix} \\ A^* &= \begin{pmatrix} v_1^* & v_2^* & v_3^* \end{pmatrix}^{-1} \begin{pmatrix} w_1^* & w_2^* & w_3^* \end{pmatrix}, \end{aligned} \tag{21}$$

which exists since the $\{v_i\}$ are linearly independent [90]. A and b can then be obtained using equation (20). In general, A is a composition of several fundamental geometric transformations, including rotations, shears, and scaling [93].

3.1.2. Resampling. Affine maps (and in general, image warping maps) are nicely behaved when the range of the mapping, the target set, is continuous. However, problems of reconstruction arise when the target set is discrete, since some of the pixels can be warped to points that are not on the target image grid (i.e. the range of the affine map $W \not\subset \mathbb{Z}^+ \times \mathbb{Z}^+$) [90]. To handle this, resampling is performed on the target image on the discrete image grid $W_d = [0, h] \times [0, w] \subset \mathbb{Z}^+ \times \mathbb{Z}^+$ for an $h \times w$ image to recover all pixel values on the digital grid.

Recovering the values can be done by a choice of different interpolation schemes, as in [94], over the mapped values. In this work, bicubic spline interpolation is used for this task to construct functions $f(x, y)$ for any pair (x, y) based on known values of (x_i, y_i). Cubic interpolation is the construction of some cubic piecewise function $f(x)$ which passes through a set of N points $\{x_i\}_{i=1}^N$. Specifically, for $x \in [x_i, x_{i+1}]$,

$1 \leq i \leq N-1$, we have $f_i(x) = a_i x^3 + b_i x^2 + c_i x + d_i$. $f(x)$ is well-defined when some conditions are imposed to solve for the coefficients a_i, b_i, c_i, d_i, which are continuity of the function and its first and second derivatives at the endpoints of every interval $[x_i, x_{i+1}]$ [95]. Extending this to the 2D case for images, the problem becomes that of constructing a function $f(x,y)$ which passes through a known set of $h \times w$ points $\{x_i, y_j\}_{i,j=0}^{i=w,j=h}$. Analogous to the 1D case, the function is piecewise defined for every region $M_{i,j}$ with vertices $\{(x_i, y_j), (x_{i+1}, y_j), (x_i, y_{j+1}, (x_{i+1}, y_{j+1}))\}$ for $0 \leq i \leq w-1, 0 \leq j \leq h-1$ [96]. The continuity conditions imposed at all vertices (x_i, y_j) of each such region are continuity of $f(x_i, y_j), \frac{\partial f}{\partial x}(x_i, y_j), \frac{\partial f}{\partial y}(x_i, y_j)$, and $\frac{\partial f}{\partial xy}(x_i, y_j)$, and for each region $M_{i,j}$ the function $f_{i,j}(x,y)$ is defined as

$$f_{i,j}(x,y) = \sum_{m,n=0}^{3} c_{mn}(x - x_i)^m (y - y_j)^n \tag{22}$$

which reduces to a series of cubic spline problems after imposing the continuity conditions for every vertex on the mesh [96]. In images, the gradients $\frac{\partial f}{\partial x}(x_i, y_j), \frac{\partial f}{\partial y}(x_i, y_j)$ can be approximated using finite difference or image operators, methods which can be found in any standard textbook on image processing such as [1].

To apply bicubic interpolation to image warping and reconstruction, consider the source image mesh I_S and a target, *discrete* image mesh I_T (normally, $I_T = I_S$ when the source and target images have equal sizes). We seek a warping function $T : I_S \mapsto I_T$ which maps quantities $q_{i,j}$ defined on I_S to I_T, with possibly different locations. In the case of image warping, $q_{i,j} = (r,g,b)_{i,j}$ represents the color values at the pixel $p_{i,j} = (i,j)$. When a warping function T is applied on $p_{i,j}$, the quantity $q_{i,j}$ is mapped to some pixel $p_{i+d_i, j+d_j}$ given by equation (19), but it is not necessary that $p_{i+d_i, j+d_j} \in I_T$ if the mapped pixel locations lie on an irregular mesh [90]. The irregular mesh is then used to define all the vertices to form the approximating function $f(x,y)$ using bicubic interpolation, defined piecewise as in equation (22) over the mapped pixels $p_{i+d_i, j+d_j}$. If the quantity $q_{i,j} \in \mathbb{R}^n$, n such functions are necessary to interpolate each component of $q_{i,j}$ at the discrete pixel locations in I_T. In the cases of color values or 2D flow fields, we require three or two such functions each, respectively.

In this thesis, image warping and reconstruction is used on face images to generate the ground-truth optical flow based on learning affine maps that warp one face frame to the next, from which the optical flow can be inferred by taking the position difference for all mapped pixels (section 3.2). Inferring those affine maps will be based on facial keypoints that are detected for each frame.

3.2. Dataset Generation Algorithm

Our method is inspired by the progress in self supervised learning techniques for action recognition [97] and eye gaze prediction [98]. We introduce the notation that we'll use throughout this section to generate the optical flow ground truth from the BP4D-Spontaneous dataset [7]. We define S_{ij} to be sequence j for the subject indexed by i, where $0 \leq j \leq 7$ and $0 \leq i \leq 40$. For each S_{ij}, we denote the frames contained in that sequence by $F_{ij} = \{f_0, ..., f_{N_f}\}_{ij}$, where $f_k \in \mathbb{R}^{H \times W \times 3}$ are the ordered frames, $0 \leq k \leq N_{f_{ij}}$. Our aim in this section is to compute the optical flow field $\boldsymbol{U}_{ij}$ separately for each video, represented by the sequence of frames F_{ij}, where $\boldsymbol{U}_{ij} = \{\boldsymbol{u}_0, ..., \boldsymbol{u}_{N_f}\}_{ij}$ contains the optical flow fields $\boldsymbol{u}_k = (u_k, v_k) : \mathbb{R}^{H \times W} \mapsto \mathbb{R}^{H \times W \times 2}$ for every frame f_k. The $\boldsymbol{u}_k$ are vector-valued functions defined on the image grid. Lastly, given a sequence S_{ij}, we denote the facial landmarks tracked on the face in each frame f_k by $\boldsymbol{P}_k = (\boldsymbol{p}_{\mathbf{0}} \ldots \boldsymbol{p}_{68})_k^T \in \mathbb{R}^{68 \times 2}$. Concisely, the tuple $\{S_{ij}, F_{ij}, \boldsymbol{P}_{ij}, \boldsymbol{U}_{ij}\}$ encodes the frames, landmarks, and flow fields for S_{ij}.

We use the BP4D-Spontaneous dataset [7], which consists of 41 subjects with 8 video sequences, each containing videos of elicited emotions. The motivation for using BP4D-Spontaneous stems from its environment as there is little head movement across frames. Our aim is to let the network learn the facial movements. It should be noted that for FER, head pose in-variance is useful in real-world conditions, though this is not the focus of this paper. Figure 7 shows example images taken from the dataset from six different subjects [7].

Figure 7. Sample images from the BP4D-Spontaneous showing different subjects from varying sequences (image source - [7]).

We describe the procedure for some frame S_{ij} and we drop the i, j for brevity. Landmarks $\boldsymbol{P}$ on the face in S are tracked for each frame using the open source OpenFace pipeline [19], which uses the *Convolutional Experts Constrained Local Model* [18] to obtain 68 landmarks per face. Next, we completely partition the first face f_0 into a triangular mesh using Delaunay triangulation on $\boldsymbol{P}_0$. This was done using Scipy's Delaunay triangulation package. Theoretical background related to Delaunay triangulation can be found in [93], and descriptions of different algorithms can be found in [99] and [100]. This mesh divides the face into N_t disjoint triangles $\boldsymbol{T}_0 = \{\boldsymbol{t}_0, ..., \boldsymbol{t}_{N_t}\}$, where each $\boldsymbol{t}_l = (\boldsymbol{v}_0, \boldsymbol{v}_1, \boldsymbol{v}_2)_l^T \in \mathbb{R}^{3\times 2}$ is the set of vertices of triangle l. After triangulating f_0, we use similar triangulation on the remaining frames in the sequence, yielding the set of triangulations $\{\boldsymbol{T}_k\}_{k=0}^{N_f}$ for subject i, sequence j.

We use the triangulation to capture the local motion on every triangle in the face partition from frame f_{k-1} to frame f_k. Given a triangle $\boldsymbol{t}_{k-1}^l$, we infer an affine map $\boldsymbol{A}_{k-1,k}^l \in \mathbb{R}^{3\times 3}$ that sends its vertices to the vertices in $\boldsymbol{t}_k^l$. Recalling that three known mappings are sufficient to uniquely specify an affine map from equation (21), we can define $\boldsymbol{t}^* \in \mathbb{R}^{3\times 3}$ to be $\boldsymbol{t}$ in homogeneous coordinates by appending the scalar 1 to the end of each vertex $\boldsymbol{v} \in \boldsymbol{t}$. Then, for triangles k and $k-1$, $\boldsymbol{A}$ is uniquely determined:

$$\boldsymbol{A} = \left(\boldsymbol{t}^*_{k-1}\right)^{-1} \boldsymbol{t}^*_k \tag{23}$$

This gives the required matrix for the affine map. Note that if $\boldsymbol{t}^*_{k-1}$ is singular, this means the triangle is degenerate. Once the correspondence between the two triangles in the frames is known, $\boldsymbol{A}$ also maps the interior of $\boldsymbol{t}^l_{k-1}$ to the interior of $\boldsymbol{t}^l_k$, since barycentric coordinates are invariant under affine maps.

We use the barycentric coordinates to compute the interiors of all the triangles in f_0, and then learn each affine map $\boldsymbol{A}^l_{k-1,k}$ as described above to map all the triangle interiors from f_0 to f_1. To compute the interior of the triangle using barycentric coordinates, an efficient algorithm from [101] can be used to test if an arbitrary point $\boldsymbol{v}$ is contained in a given triangle by taking the convex combination with the triangle vertices

$$\begin{aligned} \boldsymbol{v} &= (1-\lambda_1-\lambda_2)\boldsymbol{v}_0 + \lambda_1\boldsymbol{v}_1 + \lambda_2\boldsymbol{v}_2 \\ \boldsymbol{v}-\boldsymbol{v}_0 &= \lambda_1(\boldsymbol{v}_1-\boldsymbol{v}_0) + \lambda_2(\boldsymbol{v}_2-\boldsymbol{v}_0). \end{aligned} \tag{24}$$

By taking the dot product of equation (24) with $\boldsymbol{v}_1-\boldsymbol{v}_0$ and $\boldsymbol{v}_2-\boldsymbol{v}_0$, a 2×2 system of equations can be solved for λ_1, λ_2, and $\lambda_3 = 1-\lambda_1-\lambda_2$ [101],

$$\begin{pmatrix} \|\boldsymbol{v}_1-\boldsymbol{v}_0\|_2^2 & (\boldsymbol{v}_2-\boldsymbol{v}_0)\cdot(\boldsymbol{v}_1-\boldsymbol{v}_0) \\ (\boldsymbol{v}_2-\boldsymbol{v}_0)\cdot(\boldsymbol{v}_1-\boldsymbol{v}_0) & \|\boldsymbol{v}_2-\boldsymbol{v}_0\|_2^2 \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \end{pmatrix} = \begin{pmatrix} (\boldsymbol{v}-\boldsymbol{v}_0)\cdot(\boldsymbol{v}_1-\boldsymbol{v}_0) \\ (\boldsymbol{v}-\boldsymbol{v}_0)\cdot(\boldsymbol{v}_2-\boldsymbol{v}_0) \end{pmatrix} \tag{25}$$

and if $0 \le \lambda_i \le 1\ \forall i$, then $\boldsymbol{v}$ is in the triangle of interest. We test all points in this way using a digital grid surrounding the triangle. Repeating this for every frame in a given video is overall computationally expensive, so we only do it for triangles in the *first* frame of that video. By invariance of barycentric coordinates under the affine maps $\boldsymbol{A}^l_{k-1,k}$, we know the barycentric coordinates for all subsequent frames f_k, $k > 0$.

After determining the affine maps and mapping the triangles and their interior pixels $\boldsymbol{p}_{k-1}$ to $\boldsymbol{p}_k$, we compute the per-pixel optical flow vector $\boldsymbol{u}_{k-1}$ by

$$\tilde{\boldsymbol{u}}_{k-1} = \boldsymbol{p}_k - \boldsymbol{p}_{k-1}. \tag{26}$$

However, it is not guaranteed that the $dom(\boldsymbol{A}) \subset \mathbb{Z}^{+} \times \mathbb{Z}^{+}$ for each affine map $\boldsymbol{A}$, as discussed in 3.1.2, and so the optical flow fields $\tilde{\boldsymbol{u}}_k$ are defined on points that are not necessarily pixel coordinates. Note that this affects the frames *after* the first frame f_0; the optical flow field u_{k-1} from equation (26) is defined on a discrete grid, but the pixels that are mapped from f_0 to f_1 will subsequently be mapped from f_1 to f_2, in which case it is not guaranteed that they lie on a regular mesh. However, if the pixel locations were resampled to learn the next affine map, the barycentric coordinates will have to be recomputed, and so we learn the mappings on the irregular mesh for all frames. We use resampling on the flow field quantities $\tilde{\boldsymbol{u}}_{k-1}$ to determine the actual flow field on the discrete mesh using bicubic spline interpolation. The flow fields are stored in .flo formats for later use in the experiments.

Together with the resampling stage, this procedure gives us the ground-truth vector field for all pixels of frame f_{k-1}. The details can be summarized as follows:

1. Starting from frame f_0, determine the interiors of all triangles $\boldsymbol{t}^l$ using barycentric coordinates.
2. Learn the affine maps sending all $\boldsymbol{t}_0^l$ to $\boldsymbol{t}_1^l$ and transform the entire face to obtain the first optical flow field $\boldsymbol{u}_0$.
3. For all frames starting from f_1, again infer the affine maps sending all $\boldsymbol{t}_1^l$ to $\boldsymbol{t}_2^l$ and apply the transformation on all the pixels which have already been mapped from frame f_0. This removes the need to compute the triangle interiors for frame f_1 while still finding the optical flow field $\tilde{\boldsymbol{u}}_1$.
4. From $\tilde{\boldsymbol{u}}_1$, resample the flow field over a discrete grid to yield the ground-truth flow $\boldsymbol{u}_1$.
5. Repeat step 3 for the remaining frames in the sequence $\{f_k\}_{k=2}^{N_f}$, for all sequences and subjects.

Figure 8 shows examples of ground-truth flow field visualizations with their corresponding images and triangulations. The color coding which determines direction by color and magnitude by intensity is commonly used to visualize optical flow [86]. In the top left image, the flow directions near the mouth region capture the local motion representing the opening of the mouth with arrows surrounding the mouth pointing almost radially outwards. The top right example shows a more global head motion in the directions of the streamlines (counterclockwise head motion). In the color-coded flow visualizations, it is easier to see the intensity of motion in certain regions relative to other regions.

In both images, the local motion near the eye region is more intense compared to other facial motions. In the first of the two color examples, the local motion of the opening eyelids is clearly demarcated by the color and intensity opposite to the rest of the motion, which is global head motion moving slightly downwards. From this, we get a qualitative validation that the algorithm is successful based on the choice of face mesh. A keypoint tracker with more feature points would allow for more accurate and dense motion capture.

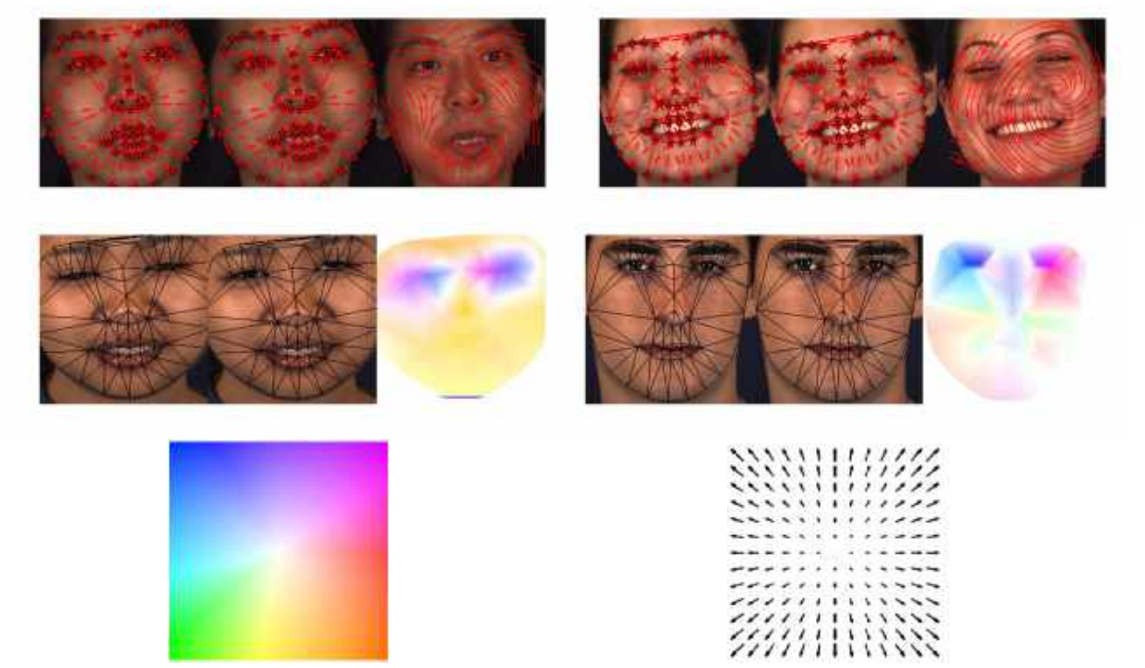
Figure 8. Examples of automatically generated facial (noisy) ground-truth flow visualizations for pairs of images, along with their triangulations. Top row shows a flow field representation using streamlines, middle row shows the visualization using the color code (image source - [86]) shown in the bottom row.

The total number of images in the generated dataset is 325720, and these were partitioned into 228171, 65130, and 32419 for training, validation, and test data respectively. The dataset generation was completed in a total of about five days using multicore CPU prallel processing with four parallel processes running at a time.

Chapter 4. Methodology

In this chapter, we describe the methodology by which we adapt the FlowNetS [8] architecture to train a CNN to learn optical flow from the automatically generated dataset. The network architecture is introduced, followed by the details of the three conducted experiments.

4.1. CNN Architecture and Training Details

The CNN architecture used is described in this section, followed by the training details of the different experimental setups implemented to test the network on the face datasets. These details include the different hyperparameters used in the different experimental setups, such as the choices of loss functions, the loss weights, and training/test data split.

4.1.1. CNN architecture: *FlowNetS*. To test the effects of having a large, ”noisy” ground-truth optical flow dataset specialized for faces on CNNs, the FlowNetS [8] architecture was used. FlowNetS is one of the pioneering CNNs on optical flow learning, and while more sophisticated optical flow architectures were developed, our purpose is to demonstrate the effect of training a CNN with face data compared with other datasets, namely the FlyingChairs dataset, as a proof of concept. Should we discover an improvement, it will be left for future work to further tackle other problems (e.g. robustness to occlusion) which were discussed in Section 2.2.2.

FlowNetS is a convolutional autoencoder architecture which consists of a sequence of downsampling layers in the encoder followed by upsampling layers in the decoder. Figure 9 shows the encoder architecture from [8]. We use the same architecture as shown, except we replace the cross-correlation layer, indicated by the yellow arrow fusing the two inputs, by a simple concatenation. This is the main difference between FlowNetS (with concatenation) and FlowNetCorr (with cross-correlation). The difference in performance reported in [8] is not too significant, and including the cross-correlation layer during training resulted in the inconvenience of much longer training times.

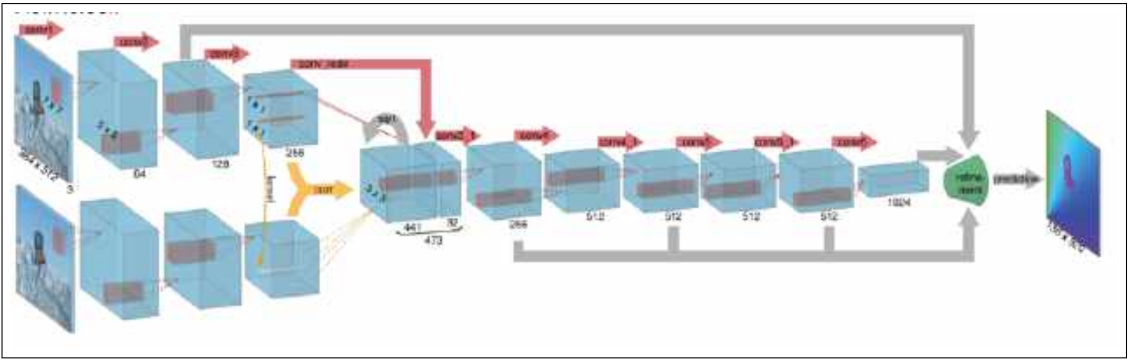

Figure 9. FlowNetS architecture encoder layers, showing the separate input region, redirected layers, and the convolution sizes (image source - [8]).

The network accepts two images and outputs the per-pixel optical flow. In the encoder layer, identical but separate convolutions operate on the first few layers where each layer uses same padding, has strides of 2, and has a ReLU activation. This means that the resolution is divided by 2 at every layer. Filters are fewer at the beginning with larger size, followed by the converse as the network gets deeper. Note that the layer weights are shared during training in the separate streams. At layer three, the two separate streams are concatenated together. This feature map is concatenated with the left feature map in the earlier layer (labelled conv_redir in the figure). More convolutional layers are applied until it reaches a low resolution in the latent space before entering the decoder part of the network.

In the decoder, the convolutions are identical to those in the encoder except they are transposed, meaning that the strides of 2 will result in upsampling instead of downsampling, also known as deconvolution. Additionally, each feature map in the decoder that shares its resolution with a layer in the encoder (for example, layer 2 of the decoder with layer 5 of the encoder) are concatenated together after its encoder counterpart undergoes a 1D convolution. Intermediate predictions are made in the decoder as well, and each of these predictions are concatenated with the layers. The decoder is shown in Figure 10. The output resolutions of each of the flow predictions in our network are also slightly different than the ones shown. Specifically, the ratio of our flow heights to theirs is $\frac{24}{17}$ and our widths to theirs is $\frac{4}{5}$. Bilinear upsampling is used to allow the concatenation of the intermediate flow predictions with the main pipeline by increasing their resolution to those of the main feature maps.

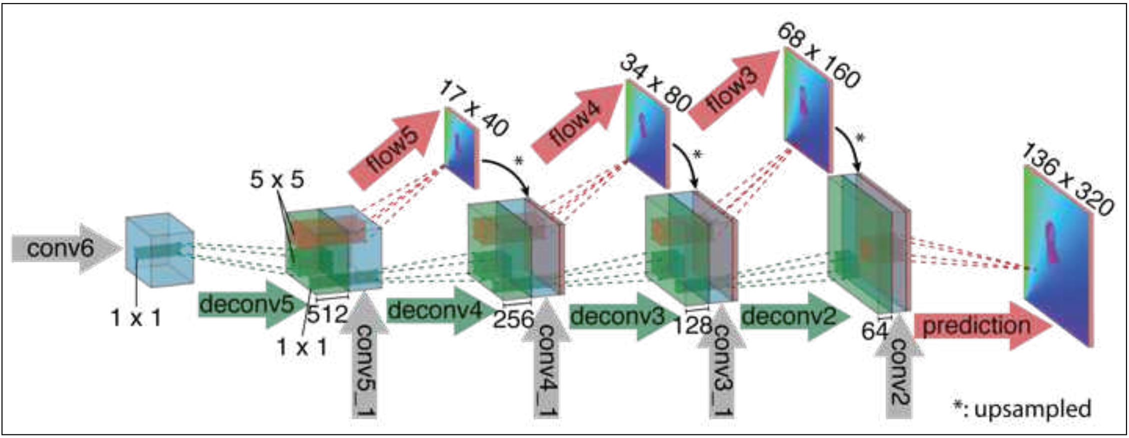

Figure 10. FlowNetS decoder layer, showing the upsampling operations and the intermediate predictions (image source - [8]).

For some of the experiments described in the next section, a cyclic loss is implemented to minimize the difference between the output predicted using the flow prediction and the second input image. This resulted in an additional warping layer to the network that acts on the largest flow prediction. The warping layer uses the predicted per-pixel flow field vectors to warp the first input image, and the result is recovered using bilinear interpolation. We note that structures inherent to the second input cannot be reproduced in the warped output, since the warping function only changes pixel locations from the first input and does not contain any learnable parameters. Figure 11 shows two examples of this phenomena from FlyingChairs and our face dataset, showing the original input image pair (X_1, X_2), the image X_2' deformed using the flow field, and visualization of the flow field Y.

For the FlyingChairs image pair, the predominant motion from the flow field is rightward motion of the left armchair to the right. The location of the armchair in the warped image is correct, but the reconstruction of the warped portion is missing. This is also present in the smaller desk chair, making a copy of itself at the warped location during reconstruction. Due to these large differences in the images, adding a warping layer while training on the chairs dataset is likely to worsen the network's performance. However, this effect is much more subtle in our face dataset due to the higher frame rate of the sequences, which causes lower magnitude motion between every two frames.

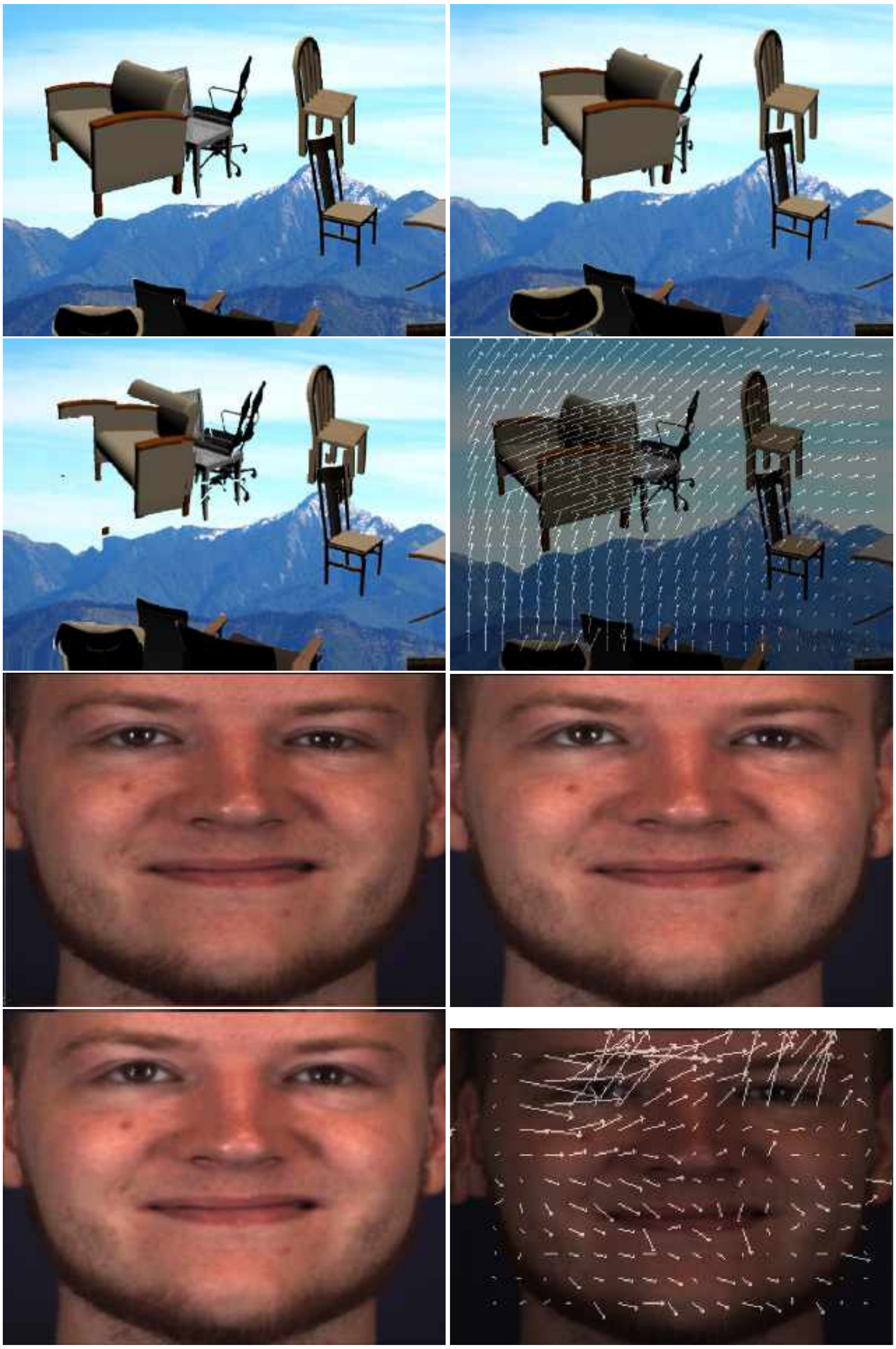

Figure 11. The top row of each example consists of two inputs X_1 and X_2. The second row shows the warped output (left) and the flow field visualization (right).

For the face example in Figure 11, the deformed image X_2' is perceptually similar to the actual X_2, particularly in the upwards motion of the eyes and the slight rightward motion caused by the furrowing of the brow. Since the time difference between two frames is very small in the face dataset, it is very unlikely for new structures to be introduced in X_2. A notable exception to this is the opening (closing) of the mouth due to revealing (hiding) teeth, which cannot be reproduced by pixel rearrangement alone. Another exception would be the squinting or widening of the eyes for the same reason, since the eyelid or eyeball would not be present in the first image. Although the artifacts caused by the warping produced a flawed image in the chairs dataset, we hypothesize that it will still help guide the directions of the predicted flow when training on faces since the undesirable effects are considerably less due to the considerably lower amount of new structure.

4.1.2. Training details. The training details of the aforementioned architecture are described in this section. We denote by (X_i, X_{i+1}) the pair of successive input frames, where $X_i, X_{i+1} \in \mathbb{R}^{384\times512\times3}$, $Y_i \in \mathbb{R}^{384\times512\times2}$ is the ground-truth flow field, and $\hat{\boldsymbol{Y}}_i = \{(\hat{Y}_i)_k\}_{k=1}^{5}$ contains the intermediate flow field predictions, where each element $(\hat{Y}_i)_k \in \mathbb{R}^{H_k\times W_k\times2}$. The i ennumerates the entire training set, and successive image frames are input to the network at every iteration. The resolutions of the flow predictions are $(H_k, W_k) = (384\times2^{-k}, 512\times2^{-k})$ for $k \in \{1,2,3,4,5\}$, as defined by the decoder in the network architecture. $(\hat{Y}_i)_1$ is the largest flow prediction, and we drop the added subscript and call it $\hat{Y}_i$ when referring to it later.

Since we assume that the background is stationary, much of the ground-truth flow field outside of the boundaries defined by the keypoints are zero vectors. To make the training more practical, we zoom on the box with vertices defined by the keypoints with maximal and minimal coordinates plus some offset in the x and y directions, and the images and flow fields are resized using bilinear interpolation. To preserve the units of the flow vectors as pixels, they are scaled accordingly in the horizontal and vertical directions. If a flow field $\boldsymbol{U}_1$ of size H_1, W_1 is to be rescaled to a flow field $\boldsymbol{U}_2$ of size

H_2, W_2, the flow vector in $\boldsymbol{U}_2$ at pixel j is $(\boldsymbol{u}_2)_j = (u_j, v_j)_2^T$, and is rescaled by setting

$$(u_j, v_j)_2^T = \left(\frac{W_2}{W_1} u_j, \frac{H_2}{H_1} v_j\right)_1^T. \tag{27}$$

The different experimental setups used to train the networks are next described.

4.1.2.1. Experimental setup 1: no cyclic loss. In this experiment, the architecture is used without the additional warping layer. The network was trained for 30, 40, and 400 epochs on the face, FlyingChairs, and Sintel datasets respectively, with 15000, 21592, and 870 training and 1000, 640, and 271 validation input image pairs each. The batch size used for training is 16 input pairs. The loss function is the average endpoint error $\mathscr{L}_1(Y_i, \hat{\boldsymbol{Y}}_i)$, defined for one output by using equation (11)

$$\mathscr{L}_1(Y_i, \hat{\boldsymbol{Y}}_i) = \sum_{k=1}^{5} \frac{w_k}{H_k \times W_k} \sum_{j=1}^{H_k \times W_k} \left\| \boldsymbol{y}_{ij} - (\hat{\boldsymbol{y}}_{ij})_k \right\|_2. \tag{28}$$

Here, the w_k are loss weights for each intermediate flow prediction loss, given by $w_k = 2^{-k}$. H_k, W_k are the sizes of the intermediate predictions and the $\boldsymbol{y}_{ij}, (\hat{\boldsymbol{y}}_{ij})_k$ are the flow vectors for the jth pixel of ground-truth and kth predicted flow fields Y_i and $(\hat{Y}_i)_k$. The flow fields Y_i are resized to compute the error for each intermediate prediction. The optimizer used is Adam, with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ as in [8], since it performed better than alternatives such as RMSprop and Stochastic Gradient Descent (SGD). The learning rate α was initialized at $1e{-}4$ for faces and $5e{-}5$ for FlyingChairs and scheduled similar to [8]. The schedule initializes the learning rate at the initial value for 10k iterations, is then multiplied by 100 and kept constant for 100k iterations, and halved afterwards for every 100k iterations. Given our training data and number of epochs, the schedule only halves the learning rate once. No data augmentation is performed. The network was trained once on the face data and once using the FlyingChairs dataset. Each trained network was then tested on both datasets each, as well as the Sintel dataset [102]. After this initial test, the same network was trained on the same data partition as the next two experiments, making them comparable.

4.1.2.2. Experimental setup 2: with cyclic loss. When the warping layer at the end of the network is included, it is necessary to define a cyclic loss function for the warped output $\hat{X}_{i+1}$ and the second input X_{i+1}. From this we expect to see an improvement in the flow prediction. For this experiment, an additional cyclic loss function $\mathscr{L}_2(X_{i+1}, \hat{X}_{i+1})$ is defined for one output as

$$\begin{aligned} \mathscr{L}_2(X_{i+1}, \hat{X}_{i+1}) &= \frac{1}{H \times W} \sum_{j=1}^{H \times W} \|X_{i+1} - \hat{X}_{i+1}\|_{H_1} \\ \|x\|_{H_1} &= \begin{cases} \frac{1}{2}x^2 & |x| \leq d \\ \frac{1}{2}d^2 + d(|x| - d) & |x| > d \end{cases} \end{aligned} \tag{29}$$

which uses the Huber loss function $\|x\|_{H_1}$ [103], a variant of the L_1 loss that is everywhere differentiable, since it is quadratic for small values of x. This makes the loss function differentiable for $x = X_{i+1} - \hat{X}_{i+1}$. We also note that $\hat{X}_{i+1}$ is a function of X_i and $\hat{Y}_i$ (the largest flow prediction). The total loss function $J(Y_i, \hat{\mathbf{Y}}_i, \hat{X}_{i+1})$ is then

$$J(Y_i, \hat{\mathbf{Y}}_i, \hat{X}_{i+1}) = \frac{1}{M} \sum_{m=0}^{M} \lambda_1(\mathscr{L}_1)_m + \lambda_2(\mathscr{L}_2)_m \tag{30}$$

with $\mathscr{L}_1, \mathscr{L}_2$ defined as in equations (28) and (30) and λ_1, λ_2 to be specified, averaged over all M training examples. In this experiment, the network was trained on both faces and chairs datasets using two different sets of loss weights λ_1, λ_2. One network with more emphasis on reconstruction and hence a higher weight for the cyclic loss, and the other with higher weight assigned to the endpoint error. Note that the w_i in equation (28) should sum to λ_1. In the following, $\sum w_i$ represents the weights of only the intermediate low-resolution flow predictions, which are all the flow predictions except for the final output. They are a designated fraction of the final flow prediction, and their combined sum is lower than the one assigned to the largest flow prediction, since the latter is the primary output of the network. The values for each case are

1. More reconstruction weights (Case I): $\lambda_1 = 0.4$, $\lambda_2 = 0.6$, $\sum w_i = 0.1$

2. Less reconstruction weights (Case II): $\lambda_1 = 0.75$, $\lambda_2 = 0.25$, $\sum w_i = 0.25$

For both cases, the network was trained on faces for 15 epochs and 228160 training pairs, and on chairs for 100 epochs and 21592 pairs. Learning rates were kept constant for these experiments throughout training, since scheduling them as previously done lead to very large gradients halfway through training. In Case I, the learning rates were $2.5e{-}6$ and $1.25e{-}6$ for faces and chairs respectively, and in Case II, they were both set to $2.5e{-}6$. The results were then tested on the test sets of nearly 31k image pairs.

4.1.2.3. Experimental setup 3: with cyclic loss, smoothness constraint, and average angular error. In this experiment, an additional loss function $\mathscr{L}_3(\hat{Y}_i)$ was added. In Case I, a smoothness constraint was imposed on the flow prediction by minimizing the flow gradients, defined as

$$\mathscr{L}_3(\hat{Y}_i) = \frac{1}{H \times W} \sum_{j=1}^{H \times W} \left\| \frac{\partial \hat{u}_{ij}}{\partial x} \right\|_{H_1} + \left\| \frac{\partial \hat{u}_{ij}}{\partial y} \right\|_{H_1} + \left\| \frac{\partial \hat{v}_{ij}}{\partial x} \right\|_{H_1} + \left\| \frac{\partial \hat{v}_{ij}}{\partial y} \right\|_{H_1} \tag{31}$$

where $(\hat{u}_{ij}, \hat{v}_{ij})$ are the components of the predicted flow vector $\hat{\boldsymbol{y}}_{ij}$ at every pixel j.

Another common metric to quantify performance of optical flow algorithms [86] is the average angular error (AAE). The average angular error between two flow vectors is the average of the angle difference between every ground-truth and predicted flow vectors in the homogeneous coordinates, which are $\boldsymbol{y}_j^* = (u_j, v_j, 1)^T$ and $\hat{\boldsymbol{y}}_j^* = (\hat{u}_j, \hat{v}_j, 1)^T$ respectively. In Case II of this experiment, the loss function $\mathscr{L}_3(Y_i, \hat{Y}_i)$ is defined using equation (13) as

$$\mathscr{L}_3(Y_i, \hat{Y}_i) = \frac{1}{H \times W} \sum_{j=1}^{H \times W} \arctan \left(\frac{\|\boldsymbol{y}_{ij}^* \times \hat{\boldsymbol{y}}_{ij}^*\|}{\boldsymbol{y}_{ij}^* \cdot \hat{\boldsymbol{y}}_{ij}^*} \right). \tag{32}$$

The total loss function is then a weighted sum of the loss functions,

$$J(Y_i, \hat{\mathbf{Y}}_i, \hat{X}_{i+1}) = \frac{1}{M} \sum_{m=0}^{M} \lambda_1 (\mathscr{L}_1)_m + \lambda_2 (\mathscr{L}_2)_m + \lambda_3 (\mathscr{L}_3)_m. \tag{33}$$

The network was trained on only the faces dataset for 14 epochs and 228160 training pairs, with $\lambda_1 = 0.3$, $\lambda_2 = 0.5$, $\lambda_3 = 0.2$, and learning rate $2.5e{-}6$. The weights were initialized from the results of Experiment 2, Case I, to see if there is any improvement in flow prediction after adding $\mathscr{L}_3$. Table 1 summarizes the experimental setups described with λ_i for $i = 1,2,3$ as defined in equation (33) and $\sum w_i$ again denoting the weights of the intermediate flow predictions, along with other training details.

Table 1. Summary of the loss weights, training/validation/testing splits, and learning rates used for each of the experiments described in sections 4.1.2.1-4.1.2.3.

	Experiment 1	Experiment 2		Experiment 3	
	—	Case I	Case II	Case I	Case II
λ_1 (EPE)	1	0.4	0.75	0.375	0.3
λ_2 (Cyclic)	0	0.6	0.25	0.625	0.5
λ_3 (AAE)	0	0	0	0	0.2
$\sum w_i$	0.46875	0.1	0.25	0.075	0.06
Training data	15k	228k	228k	228k	228k
Validation data	1k	65k	65k	65k	65k
Testing data	1k	31k	31k	31k	31k
No. of epochs	30	15	15	15	15
Learning rate	$1e{-}4$	$1.25e{-}6$	$1.25e{-}6$	$1.25e{-}6$	$1.25e{-}6$

4.1.3. Evaluation of performance. The evaluation of both of the constrained optical flow data generation and the neural network performance in predicting the optical flow should be based on objective and informative criteria. The optical flow performance can be quantitatively analyzed by the loss reported by the neural network on test datasets as well as the flow metrics, the EPE and AAE. In addition to the flow metrics, the structural similarity index (SSIM), L_1, and L_2 errors between the warped output image $\hat{X}_{i+1}$ and the actual input image X_{i+1} is also computed during testing but separately and for 500 image pairs. The SSIM [87] $S(X_{i+1}, \hat{X}_{i+1})$ is computed using equation (15)

$$S(X_{i+1}, \hat{X_{i+1}}) = \frac{(2\mu_{X_{i+1}}\mu_{\hat{X}_{i+1}} + C_1)(2\sigma_{X_{i+1}\hat{X}_{i+1}} + C_2)}{(\mu^2_{X_{i+1}} + \mu^2_{\hat{X}_{i+1}} + C_1)(\sigma^2_{X_{i+1}} + \sigma^2_{\hat{X}_{i+1}} + C_2)} \tag{34}$$

where $\mu_{X_{i+1}}, \mu_{\hat{X}_{i+1}}, \sigma_{X_{i+1}}, \sigma_{\hat{X}_{i+1}}$ are the discrete mean and standard deviations of the images $X_{i+1}, \hat{X}_{i+1}$ and $\sigma_{X_{i+1}\hat{X}_{i+1}}$ is the correlation coefficient between X_{i+1} and $\hat{X}_{i+1}$. In addition to these metrics, the test dataset of 31k images was also passed through another optical flow CNN implementation [68], and we compute the EPE and AAE for the predicted flows to compare with our results.

Chapter 5. Experimental Results and Discussion

5.1. Results for Setup 1: No Cyclic Loss

The results of Experiment 1, which comprises of the network trained and tested on faces, chairs, and Sintel dataset, are first described. We report the performance in terms of average endpoint error. In addition, we also show a pair of face images for visualization. Table 2 shows the first experiment's overall statistics for each network when tested on 3000 samples from our face dataset. In addition, we also show the performance comparison when trained and tested on other datasets. It is worth noting that the subjects that appear in the training set do not appear in the validation or test set of our face data.

Table 2. The average endpoint errors for each network described in subsection 4.1.2.1, trained and tested on all three datasets.

		Tested on		
		Faces	Chairs	Sintel
Trained on	Faces	0.4054	5.8495	5.1731
	Chairs	1.4040	1.4413	3.0300
	Sintel	0.8282	7.7613	6.2358

The error values in Table 2 are in pixels, averaged over each of the test sets. Row 1 shows the results when the network was trained on faces and tested on the data from the other three datasets. Similarly, rows 2 and 3 are trained on chairs and Sintel and tested on all three. From Table 2, we observe that the network trained on our BP4D-derived face dataset performs best when tested on faces. This is likely due to the nature of the dataset the network was trained on. The flow fields on our face dataset consist of small, non-rigid motions, especially when the head motion is lacking, whilst the motion fields in the chair dataset has larger magnitude and is more rigid. The Sintel dataset is also different in nature than the face dataset, but has smaller overall motion, and thus it is likely that the network trained on chairs is overestimating the motion on the face dataset. Note that the results showed in Table 2 are comparable to state of the art methods on the Sintel dataset, as can be seen in [102].

We show an example in Figure 12 of predicted flow from each dataset on an image pair from the face dataset to better understand the network performances. We

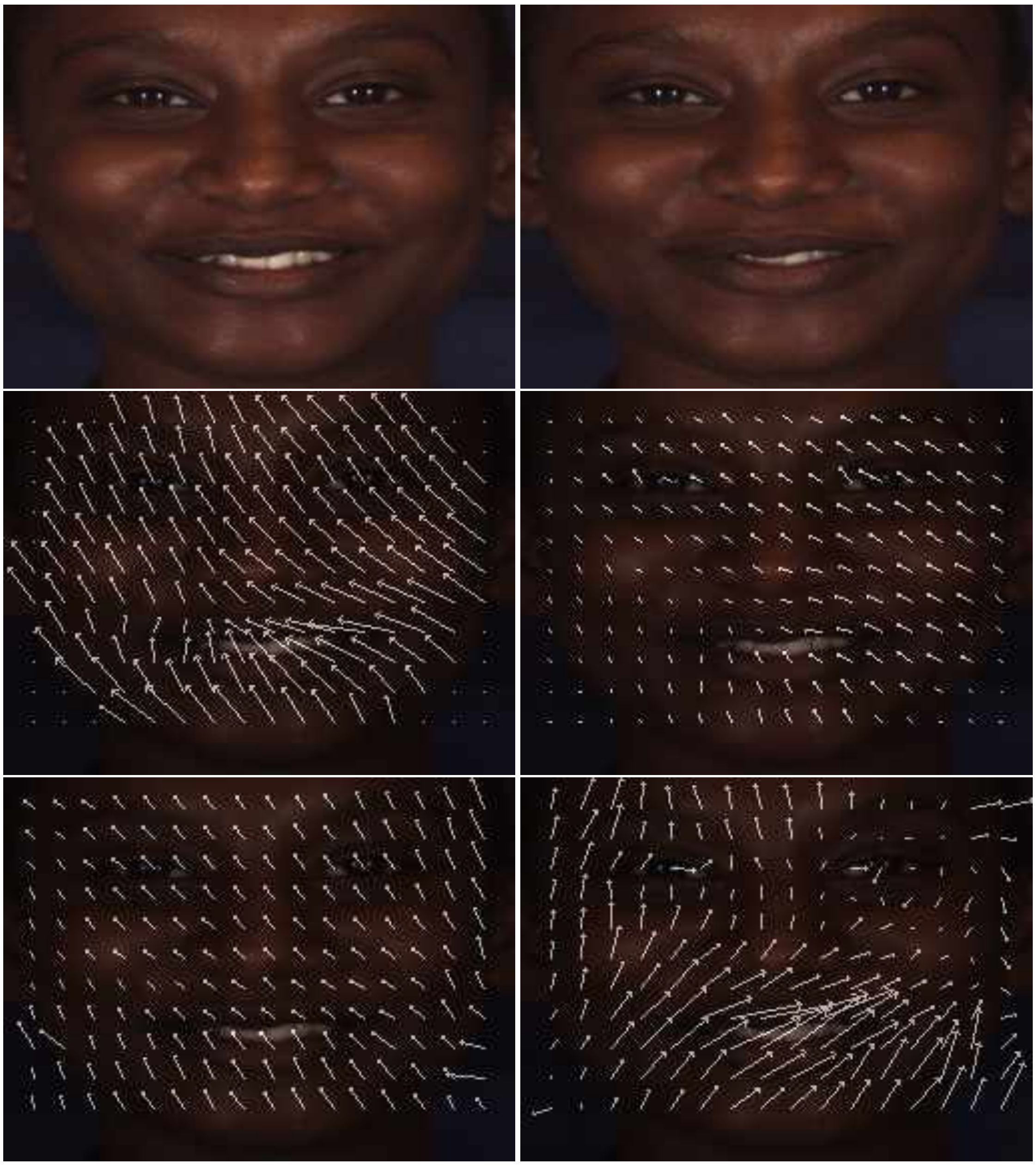
Figure 12. A pair of input images from BP4D-Spontaneous (top row); flow ground-truth and flow field generated from face-trained network, left to right (middle row); and flow field predictions from networks trained on chairs and Sintel, left to right (bottom row). The EPE for networks trained on faces, chairs, and Sintel are **1.16**, 2.78, and 1.35 respectively. This shows the effectiveness of the learnt representation from the proposed method.

note that the visualizations are *not* to scale in pixel units; that is, only the directions and relative magnitudes are of importance. In the prediction from the face network, we see that the flow arrows are well-aligned with the ground-truth arrows in most locations of the face, except for the left cheek, which is also where the triangles were largest during dataset generation. The arrows are steeper around the mouth region, also showing how the network was able to capture the local motion in addition to the global upwards head movement. In contrast, the chairs prediction also captures the general upwards global motion, but does not discriminate between the global and local motion in the mouth regions. Finally, the Sintel prediction did not yield a very good optical flow representation for the face image at all locations. However, although the directions are different, the local steepness around the mouth region is also captured. From this pair amongst others and the data in Table 2, we can conclude that the network trained on faces does a better job at capturing both global and local motions than the other two networks.

5.2. Results for Setups 2 and 3: With Cyclic Loss

After adding the cyclic loss and training for more data and epochs, we expect to observe a difference in performance compared with Experiment 1. In this section, we show the results of the networks with cyclic loss trained as described in sections 4.1.2.2 and 4.1.2.3.

Table 3 summarizes the statistics computed based on the results of Experiments 2 and 3. The statistics related to the flow fields (AAE and EPE) are computed for all 31k image pairs in the test set, and the other quantities, namely SSIM, L_1 loss, and L_2 loss are based on a test set of 500 images.

Table 3. Values of the different performance measures for each experimental setup.

	Experiment 1	Experiment 2		Experiment 3		PWC-Net
	—	Case I	Case II	Case I	Case II	—
AAE	0.1987	0.3141	0.1698	0.3471	0.3251	0.4749
EPE	0.2895	0.5331	0.2493	0.5291	0.6001	1.7722
SSIM	0.9504	0.9865	0.9805	0.9814	0.9858	0.9770
L1	3.9823	2.9162	3.6797	3.1087	2.8809	6.1260
L2	2.5008	1.8685	2.3245	1.9842	1.8496	3.7648

Case I and II under Experiment 2 represent, respectively, the higher and lower reconstruction weight experiments, and Case I and II under Experiment 3 represent the experiment with smoothness constraint and the experiment with average angular error.

There are several observations to be made from these results. Adding the cyclic loss but with *lower* reconstruction weights (Experiment 2, Case II) improves the flow prediction compared to using only the EPE loss (Experiment 1), since both EPE and AAE decrease significantly. The image statistics also improve in that case with an increase in SSIM and a decrease in L_1 and L_2 losses. When there is higher weight on reconstruction loss (Experiment 2, Case I), the image statistics also improve as the network alters the predicted flow to improve the structural similarity and the warped output's semblance to X_2. However, the higher focus on reconstruction worsens the performance of the AAE and EPE. One reason could be that the noisy ground truth does not necessarily reconstruct X_2 from X_1 very well, which makes the change in SSIM adversarial to the change in flow EPE and AAE. The flow representations learned after adding the cyclic loss are better at reconstruction, as verified by all of the image statistics from Table 3.

Experiment 3 with the smoothness and AAE losses yields worse outcomes than the other two in terms of predicted flow, particularly compared to Experiment 2, Case I, since Experiment 3 weights are initialized from there to test any change in performance. This could be due to the decreased weight in the EPE loss, which suggests that the EPE is a stronger indicator of flow performance than the AAE. The EPE encodes the direction in addition to the magnitude information. Another explanation would be that training data with angular error as a loss metric does not generalize well to test data, unlike the other metrics. By decreasing the reconstruction weight λ_2 from 0.6 to 0.5, the SSIM value decreases slightly as well. The decrease in SSIM is larger in Experiment 3, Case I, which is likely due to the imposed smoothness constraints, since some regions of the face do not have necessarily small flow gradients necessary to reconstruct the second image. For example, upwards global motion of the face coupled with downwards motion of the upper lip when closing the mouth means that the flow field will experience abrupt changes in direction and magnitude. These abrupt changes necessary for reconstruction are restricted by the smoothness constraints, which may explain the decrease in SSIM.

For all cases, the networks trained on our automatic face dataset perform better than PWC-Net [68], which is a successful optical flow implementation using a different architecture than FlowNetS that we use as a benchmark for our results. The high EPE is likely due to an overestimation of the flow prediction magnitudes. The reconstruction errors are also worse, as seen from the SSIM value as well as L1 and L2 losses.

To visualize this comparison, Figure 13 shows color-coded flow predictions for each of the experimental setups, as well as the output from PWC-Net for cases with both global and local motion. The first two columns are the input pair X_1 and X_2, third column is ground-truth, and rest are the experiments in the same order as in Table 3.

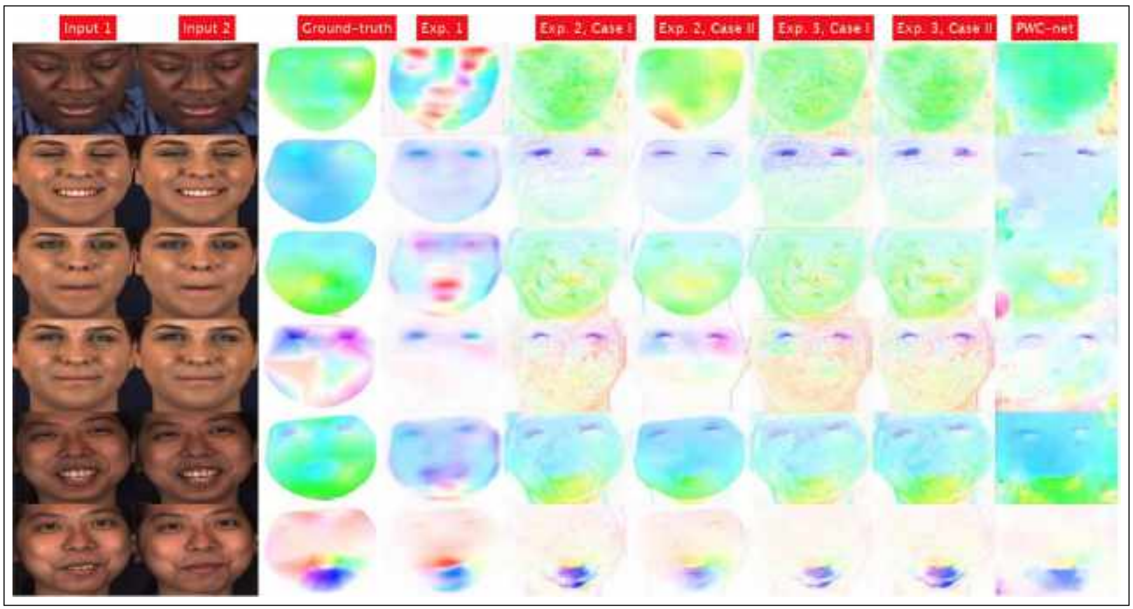

Figure 13. Color coded optical flow prediction comparisons for the networks trained in each of the experimental setups, with examples to highlight how each network learns local motions.

As in the flow visualizations in Figure 12, the flow fields are not to scale from one prediction to the other. The saturation intensity in a given image is only representative of the intensity of that region relative to the other pixels of the *same* image. The same intensity in two images may have substantially different optical flow vector values. This is common practice in optical flow visualization, since it places emphasis on which motion is more salient for a given image. In images with small motion, as is the case in many frames in the BP4D dataset, the visualization would not convey important local motion information.

The image pairs in rows 2 and rows (4-6) all contain local motion, mainly in the eye region except for the jaw region in the last row. The network trained using only the EPE (Experiment 1) also magnifies the eye motion in the images with little eye action. It could be that the shape of the eyes from the input images is a markedly distinct feature than other regions, and is thus more easily abstracted by the network.

Both predictions from the network with the cyclic loss function in Experiment 2 (columns 5 and 6) seem to exhibit different learned motion. When the reconstruction weights are higher than the EPE weights (Case I), the motion varies more frequently across the face than when compared to the reverse case (Case II). The results from Table 3 show that the flow predictions are better in Case II, and that could be because the increased focus on the cyclic loss in Case I leads to a more dense flow representation due to the network attempting to reconstruct the input X_2 at every pixel. However, the ground-truth is sparse and shows the general direction of motion per face region, which leads to higher error when compared with a denser flow field (Experiment 2, Case I) than with a similarly sparse flow field (Experiment 2, Case II). This is also supported by the data in Table 3.

The flow predictions from PWC-Net show perceptually similar flow fields compared to the ground-truth and other predictions. One notable difference is that the predicted flow field in the areas outside the face are nonzero, which likely contributed to the much higher EPE and AAE compared to the other cases. Otherwise, the motion directions in the other regions seem reasonable.

Figure 14 shows another set of examples of predicted optical flow on examples with more global motion compared to Figure 13. Due to the enormity of the test set, it is difficult to visualize all the different examples. However, we focus on comparing the networks' abilities to capture local and global motion in addition to the density of the flow to infer the qualitative differences of the learned representations that complement the statistics in Table 3 earlier.

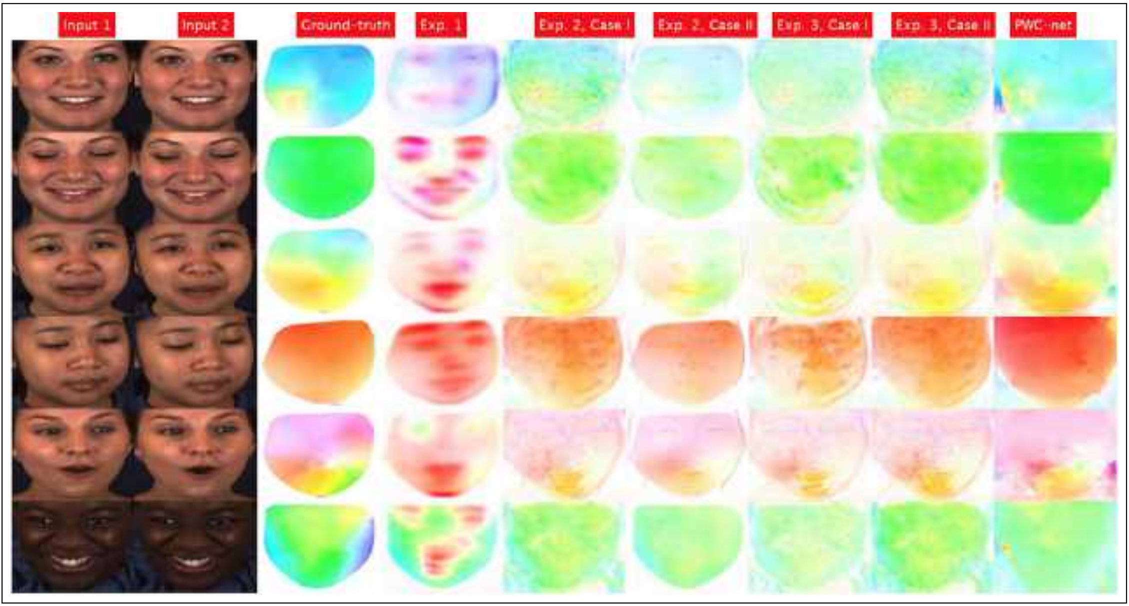

Figure 14. Color coded optical flow prediction comparisons for the networks trained in each of the experimental setups, with examples to highlight how each network learns global motions.

The three examples from rows (2, 4, 6) are characterized by predominantly global motion in *one* direction. Two examples from rows (1, 3) are characterized by global motion with varying directions, and row 5 contains both global and local motion. For the examples with uniform global motion, all of the network predictions correctly estimate the motion direction and magnitudes in different regions of the face except for the prediction from Experiment 1. This exemplifies the advantage of adding the cyclic loss to the network by showing how the emphasis on reconstruction helps guide the flow to learn the global head motion. This observation will vary across test images, but it helps explain how adding the cyclic loss improved the statistics in Table 3. In rows (1, 3), the flow visualizations indicate that the representations can learn optical flow fields of various types. The optical flow fields in the cases with higher reconstruction weights (Experiment 2, Case I, and both cases of Experiment 3), the network learns a *denser* flow field in its attempt to reconstruct the second input, while the higher focus on EPE makes the network learn flow field similar to the nature of the ground-truth flow. More examples are shown in Figures 16 and 17 showcasing a mix of both global and local motions.

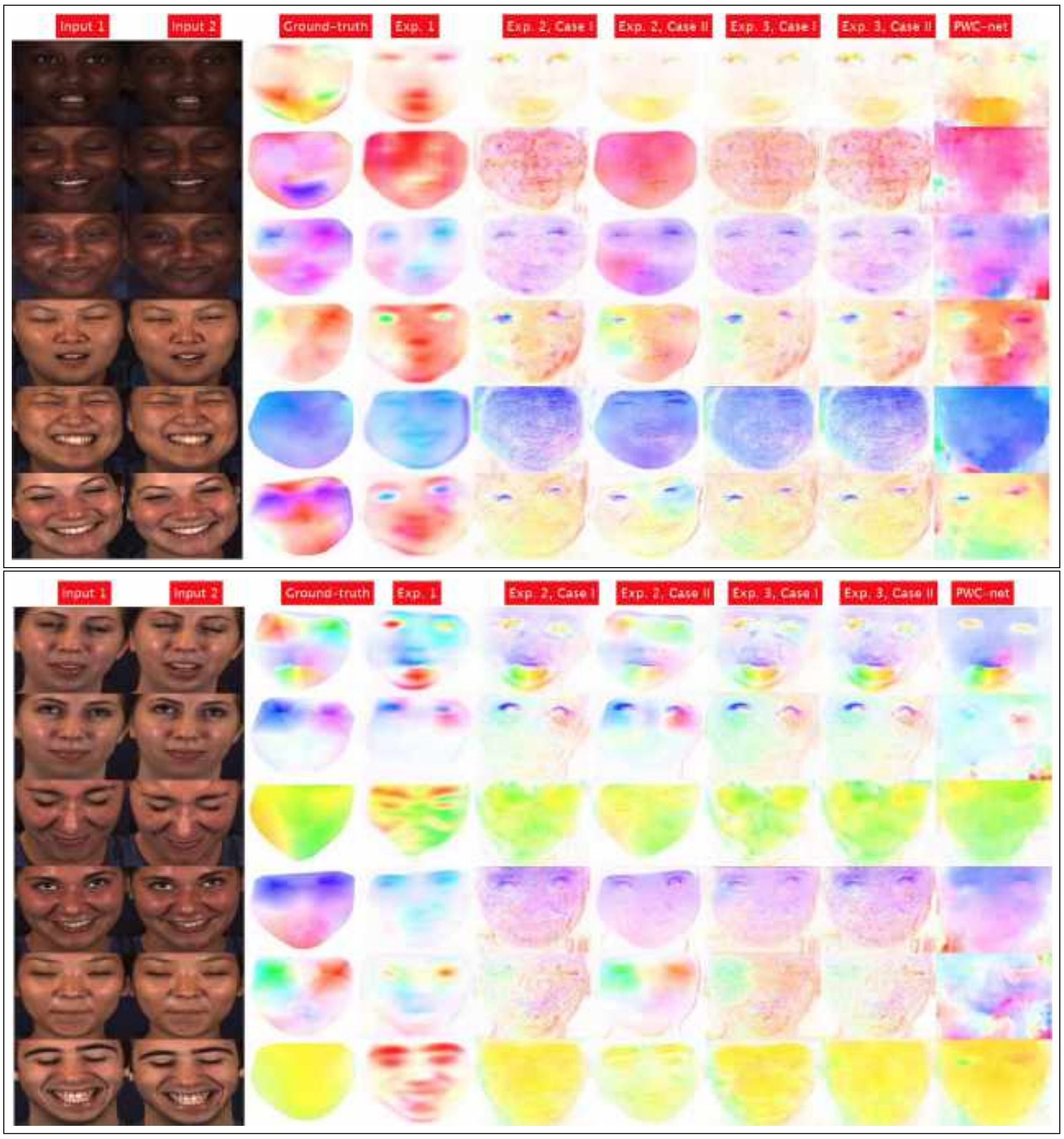

Figure 15. More optical flow comparisons for all the trained networks, with mixed examples of both local and global motion.

From the overall results of the experiments, our network trained on the automatically generated face dataset is better-suited at predicting the optical flow on faces compared to other networks. Even if the ground-truth is not the best representation of the actual optical flow, it is demonstrated that the network can adapt to the type of ground-truth, and will likely outperform other networks when trained on denser face optical flow ground-truth.

Chapter 6. Concluding Remarks

In this thesis, we explore the possibility of using a facial expression dataset to learn optical flow representations based on a self-supervised technique. Motion information on faces has been shown to be useful in facial expression analysis, which is particularly useful in human-robot interaction when combined with multimodal fusion techniques for better emotion recognition.

The dataset is generated by using the image sequences from the BP4D-Spontaneous dataset to compute the optical flow ground-truth. The OpenFace 2.0 toolbox, which uses a constrained local model, is used to locate the facial landmarks on every image. Delaunay triangulation is then used on the resulting set of points to form the face mesh and allow the computation of the optical flow for every pair of images using triangle-to-triangle affine maps to develop an automatic facial optical flow dataset. The generated dataset, with a total of nearly 324k image pairs, is used as a ”noisy ground-truth” for optical flow to train the FlowNetS convolutional autoencoder architecture with 228k pairs in the training partition.

It was observed that training the state of the art FlowNetS architecture for optical on this automatically generated noisy ground-truth data improved the network’s ability to predict optical flow on face data in particular. The learned representations also helped the network give good accuracy on the FlyingChairs and Sintel datasets. This demonstrates that the facial movements are nicely encoded in our data which en-ables the network to learn subtle movements that are useful on the challenging Sintel dataset as well. A cyclic loss was also added for optimization to help the network use the predicted flow to reconstruct the second image, and the flow results from different experimental setups are compared. It was seen that the flow predictions are best when there is less emphasis on reconstruction, due to denser representations learned with reconstruction that are not present in the ground-truth flow fields. Compared with an implementation by PWC-Net, which is another high performing optical flow CNN, it was shown that the networks trained on the generated dataset predict better flow representations, as quantified by the flow error metrics. This implies that a network trained on good face optical flow ground-truth have the propensity to outperform networks trained on other datasets.

For further investigation and improvement, future work related to this thesis can include the following:

1. Use a denser tracker such as Zface to track a higher number of keypoints for a finer triangulation and denser optical flow ground-truth after implementing our automatic data generation algorithm.
2. Compare the denser optical flow with several state of the art implementations for a more rigorous quantification of the network performances.
3. Train the optical flow network on faces with some head rotation, such as pan and tilt, to learn optical flow for non-frontal faces.
4. Tackle challenges in optical flow learning, such as in environments with occlusion and illumination, to increase the robustness of facial optical flow.

In addition to these improvements for optical flow learning, we can also use the optical flow generated from the network to train a system for emotion recognition on the BP4D-Spontaneous and other datasets. Obtaining finer optical flow by using the aforementioned improvements will be useful as well in other facial expression analysis tasks, like action-unit recognition and microexpression detection.

References

[1] M. Sonka, V. Hlavac, and R. Boyle, *Image Processing, Analysis, and Machine Vision*, 2nd ed. Cengage Learning, 2014.

[2] S. Söderkvist, K. Ohlén, and U. Dimberg, "How the experience of emotion is modulated by facial feedback," *Journal of Nonverbal Behavior*, vol. 42, no. 1, pp. 129–151, March 2018.

[3] C. F. Benitez-Quiroz, R. B. Wilbur, and A. M. Martinez, "The not face: A grammaticalization of facial expressions of emotion," *Cognition*, vol. 150, pp. 77 – 84, May 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0010027716300324

[4] S. M. Gillespie *et al.*, "Psychopathic traits are associated with reduced attention to the eyes of emotional faces among adult male non-offenders," *Frontiers in Human Neuroscience*, vol. 9, p. 552, Oct. 2015. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnhum.2015.00552

[5] A. J. O'Toole, *Psychological and Neural Perspectives on Human Face Recognition*. New York, NY: Springer New York, 2005, pp. 349–369.

[6] F. Chollet, *Deep Learning with Python*, 1st ed. USA: Manning Publications Co., 2017.

[7] X. Zhang *et al.*, "Bp4d-spontaneous: A high-resolution spontaneous 3d dynamic facial expression database," *Image and Vision Computing*, vol. 32, pp. 692–706, June 2014.

[8] A. Dosovitskiy *et al.*, "Flownet: Learning optical flow with convolutional networks," in *Proceedings of the IEEE International Conference on Computer Vision*, Santiago, Chile, Dec. 2015. [Online]. Available: http://lmb.informatik.uni-freiburg.de/Publications/2015/DFIB15

[9] P. Viola and M. Jones, "Robust real-time object detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, Jan. 2002.

[10] K. Vikram and S. Padmavathi, "Facial parts detection using viola jones algorithm," in *4th International Conference on Advanced Computing and Communication Systems (ICACCS)*, Coimbatore, India, Jan. 2017, pp. 1–4.

[11] S. Klemm *et al.*, "Robust face recognition using key-point descriptors," in *VISAPP 2015: 10th International Conference on Computer Vision Theory and Applications*, vol. 2, Berlin, Germany, Jan. 2015, pp. 447–454.

[12] V. A *et al.*, "Two novel detector-descriptor based approaches for face recognition using sift and surf," *Procedia Computer Science*, vol. 70, pp. 185 – 197, Dec. 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050915032342

[13] D. Kim and R. Dahyot, “Face components detection using surf descriptors and svms,” in *2008 International Machine Vision and Image Processing Conference*, Los Alamitos, CA, USA, Sept. 2008, pp. 51–56.

[14] Y. Bai *et al.*, “Finding tiny faces in the wild with generative adversarial network,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, June 2018, pp. 21–30.

[15] H. Jiang and E. Learned-Miller, “Face detection with the faster r-cnn,” in *12th IEEE International Conference on Automatic Face Gesture Recognition*, Washington, DC., USA, June 2017, pp. 650–657.

[16] Y. Bai and B. Ghanem, “Multi-branch fully convolutional network for face detection,” *CoRR*, vol. abs/1707.06330, July 2017. [Online]. Available: http://arxiv.org/abs/1707.06330

[17] S. Gong, S. J. McKenna, and A. Psarrou, *Dynamic Vision: From Images to Face Recognition*, 1st ed. GBR: Imperial College Press, 2000.

[18] A. Zadeh, T. Baltrusaitis, and L. Morency, “Deep constrained local models for facial landmark detection,” *CoRR*, vol. abs/1611.08657, Nov. 2016. [Online]. Available: http://arxiv.org/abs/1611.08657

[19] T. Baltrusaitis, P. Robinson, and L. Morency, “Constrained local neural fields for robust facial landmark detection in the wild,” in *Proceedings of the 2013 IEEE International Conference on Computer Vision Workshops*, Sydney, Australia, Dec. 2013, pp. 354–361.

[20] T. Baltrusaitis *et al.*, “Openface 2.0: Facial behavior analysis toolkit,” in *2018 13th IEEE International Conference on Automatic Face Gesture Recognition*, Xi’an, China, May 2018, pp. 59–66.

[21] A. Jourabloo and X. Liu, “Pose-invariant 3d face alignment,” in *2015 International Conference on Computer Vision*, Santiago, Chile, Dec. 2015, pp. 3694–3702. [Online]. Available: https://doi.org/10.1109/ICCV.2015.421

[22] V. Kazemi and J. Sullivan, “One millisecond face alignment with an ensemble of regression trees,” in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, OH, USA, June 2014, pp. 1867–1874.

[23] J. Yu and Z. Wang, “A video-based facial motion tracking and expression recognition system,” *Multimedia Tools and Applications*, vol. 76, no. 13, pp. 14 653–14 672, July 2017. [Online]. Available: https://doi.org/10.1007/s11042-016-3883-3

[24] F. Dornaika and F. Davoine, “Simultaneous facial action tracking and expression recognition in the presence of head motion,” *International Journal of Computer Vision*, vol. 76, no. 3, pp. 257–281, Mar. 2008. [Online]. Available: https://doi.org/10.1007/s11263-007-0059-7

[25] M. Zeng and G. Q. Huang, "Video summarization by motion analysis: Using optical flow technique," in *2011 International Conference on Information Management, Innovation Management and Industrial Engineering*, vol. 3, Shenzhen, China, Nov. 2011, pp. 205–208.

[26] Z. Zeng *et al.*, "A survey of affect recognition methods: Audio, visual, and spontaneous expressions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 1, pp. 39–58, Mar. 2009.

[27] C. A. Corneanu *et al.*, "Survey on rgb, 3d, thermal, and multimodal approaches for facial expression recognition: History, trends, and affect-related applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 8, pp. 1548–1568, Jan. 2016.

[28] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the 7th International Joint Conference on Artifical Intelligence*, vol. 2. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., Apr. 1981, pp. 674–679. [Online]. Available: http://dl.acm.org/citation.cfm?id=1623264.1623280

[29] M. R. Balazadeh Bahar and G. Karimian, "High performance implementation of the horn and schunck optical flow algorithm on fpga," in *20th Iranian Conference on Electrical Engineering*, Tehran, Iran, May 2012, pp. 736–741.

[30] E. P. Simoncelli, "Design of multi-dimensional derivative filters," in *Proceedings of 1st International Conference on Image Processing*, vol. 1, 1994, pp. 790–794 vol.1.

[31] D. Cremers and S. Soatto, "Motion competition: A variational approach to piecewise parametric motion segmentation," *International Journal of Computer Vision*, vol. 62, no. 3, pp. 249–265, May 2005. [Online]. Available: https://doi.org/10.1007/s11263-005-4882-4

[32] A. Goh and R. Vidal, "Segmenting motions of different types by unsupervised manifold clustering," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*, Minneapolis, MN, USA, June 2007, pp. 1–6.

[33] E. Chen *et al.*, "Quaternion based optical flow estimation for robust object tracking," *Digital Signal Processing*, vol. 23, no. 1, pp. 118 – 125, Jan. 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1051200412001807

[34] T. Portz, L. Zhang, and H. Jiang, "Optical flow in the presence of spatially-varying motion blur," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, Providence, RI, USA, 2012.

[35] F. Porikli and O. Tuzel, "Object tracking in low-frame-rate video," in *SPIE Conference on Image and Video Communications and Processing*, vol. 5685, Mar. 2005, pp. 72–79. [Online]. Available: https://www.merl.com/publications/TR2005-013

[36] L. Zappella, X. Lladó, and J. Salvi, "Motion segmentation: A review," in *Proceedings of the 2008 Conference on Artificial Intelligence Research and Development: Proceedings of the 11th International Conference of the Catalan Association for Artificial Intelligence*. Sant Martí d'Empúries, Spain: IOS Press, July 2008, pp. 398–407.

[37] S. Koelstra, M. Pantic, and I. Patras, "A dynamic texture-based approach to recognition of facial actions and their temporal models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 11, pp. 1940–1954, Nov. 2010.

[38] V. Le, H. Tang, and T. S. Huang, "Expression recognition from 3d dynamic faces using robust spatio-temporal shape features," in *IEEE Face and Gesture 2011*, Santa Barbara, CA, USA, March 2011, pp. 414–421.

[39] P. Snape *et al.*, "Face flow," in *Proceedings of the 2015 IEEE International Conference on Computer Vision*, Santiago, Chile, Dec. 2015, pp. 2993–3001.

[40] H. Le *et al.*, "Unsupervised generation of optical flow datasets from videos in the wild," *CoRR*, vol. abs/1812.01946, Dec. 2018. [Online]. Available: http://arxiv.org/abs/1812.01946

[41] B. Allaert, I. M. Bilasco, and C. Djeraba, "Consistent optical flow maps for full and micro facial expression recognition," in *12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, vol. 5, Porto, Portugal, Feb. 2017, pp. 235–242.

[42] B. Allaert, I. M. Bilasco, and C. Djeraba, "Advanced local motion patterns for macro and micro facial expression recognition," *CoRR*, vol. abs/1805.01951, May 2018. [Online]. Available: http://arxiv.org/abs/1805.01951

[43] C.-K. Hsieh, S.-H. Lai, and Y.-C. Chen, "An optical flow-based approach to robust face recognition under expression variations," *IEEE Transactions on Image Processing*, vol. 19, pp. 233–240, Jan. 2010.

[44] C. J. Duthoit *et al.*, "Optical flow image analysis of facial expressions of human emotion: Forensic applications," in *Proceedings of the 1st International Conference on Forensic Applications and Techniques in Telecommunications, Information, and Multimedia and Workshop*, Adelaide, Australia, Jan. 2008, pp. 1–6.

[45] D. DeCarlo and D. Metaxas, "Optical flow constraints on deformable models with applications to face tracking," *International Journal of Computer Vision*, vol. 38, no. 2, pp. 99–127, July 2000. [Online]. Available: https://doi.org/10.1023/A:1008122917811

[46] B. Allaert *et al.*, "Optical flow techniques for facial expression analysis: Performance evaluation and improvements," *CoRR*, vol. abs/1904.11592, Apr. 2019. [Online]. Available: http://arxiv.org/abs/1904.11592

[47] E. Alpaydin, *Introduction to Machine Learning*, 2nd ed. The MIT Press, 2010.

[48] C. C. Aggarwal, *Neural Networks and Deep Learning*. Springer, 2018.

[49] I. N. da Silva *et al.*, *Artificial Neural Networks: A Practical Course*, 1st ed. Springer Publishing Company, Incorporated, 2016.

[50] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[51] S. Sra, S. Nowozin, and S. J. Wright, *Optimization for Machine Learning*. The MIT Press, 2011.

[52] U. Michelucci, *Advanced Applied Deep Learning: Convolutional Neural Networks and Object Detection*. Apress, 2019.

[53] A. W. Harley, "An interactive node-link visualization of convolutional neural networks," in *Advances in Visual Computing: 11th International Symposium*, G. Bebis *et al.*, Eds. Las Vegas, NV, USA: Springer International Publishing, Dec. 2015, pp. 867–877.

[54] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, May 2017. [Online]. Available: https://doi.org/10.1145/3065386

[55] K. He *et al.*, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, June 2016, pp. 770–778.

[56] F. Gao *et al.*, "Deep residual inception encoder–decoder network for medical imaging synthesis," *IEEE Journal of Biomedical and Health Informatics*, vol. 24, no. 1, pp. 39–49, Apr. 2020.

[57] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *Proceedings of the 14th European Conference on Computer Vision*, B. Leibe *et al.*, Eds. Amsterdam, The Netherlands: Springer International Publishing, Oct. 2016, pp. 483–499.

[58] Z. Lu *et al.*, "The expressive power of neural networks: A view from the width," *CoRR*, vol. abs/1709.02540, Sept. 2017. [Online]. Available: http://arxiv.org/abs/1709.02540

[59] P. Petersen and F. Voigtländer, "Equivalence of approximation by convolutional neural networks and fully-connected networks," *CoRR*, vol. abs/1809.00973, Sept. 2019. [Online]. Available: https://arxiv.org/abs/1809.00973

[60] D. Yarotsky, "Universal approximations of invariant maps by neural networks," *CoRR*, vol. abs/1804.10306, 2018. [Online]. Available: http://arxiv.org/abs/1804.10306

[61] J. Janai *et al.*, "Unsupervised learning of multi-frame optical flow with occlusions," in *Proceedings of the 15th European Conference on Computer Vision*, V. Ferrari *et al.*, Eds. Munich, Germany: Springer International Publishing, Sept. 2018, pp. 713–731.

[62] S. Datta, G. Sharma, and C. V. Jawahar, "Unsupervised learning of face representations," *CoRR*, vol. abs/1803.01260, Mar. 2018. [Online]. Available: http://arxiv.org/abs/1803.01260

[63] K. Shan *et al.*, "Automatic facial expression recognition based on a deep convolutional-neural-network structure," in *2017 IEEE 15th International Conference on Software Engineering Research, Management and Applications*, London, UK, June 2017, pp. 123–128.

[64] N. Zeng *et al.*, "Facial expression recognition via learning deep sparse autoencoders," *Neurocomputing*, vol. 273, pp. 643 – 649, Jan. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231217314649

[65] J. Li *et al.*, "Facial expression recognition with faster r-cnn," *Procedia Computer Science*, vol. 107, no. C, pp. 135–140, Apr. 2017. [Online]. Available: https://doi.org/10.1016/j.procs.2017.03.069

[66] Z. Zhang *et al.*, "Smeconvnet: A convolutional neural network for spotting spontaneous facial micro-expression from long videos," *IEEE Access*, vol. 6, pp. 71 143–71 151, Nov. 2018.

[67] V. de Oliveira Silva, F. de Barros Vidal, and A. R. Soares Romariz, "Human action recognition based on a two-stream convolutional network classifier," in *16th IEEE International Conference on Machine Learning and Applications*, Cancun, Mexico, Dec. 2017, pp. 774–778.

[68] D. Sun *et al.*, "Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume," *CoRR*, vol. abs/1709.02371, Sept. 2017. [Online]. Available: http://arxiv.org/abs/1709.02371

[69] E. Ilg *et al.*, "Flownet 2.0: Evolution of optical flow estimation with deep networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, HI, USA, July 2017, pp. 1647–1655. [Online]. Available: http://lmb.informatik.uni-freiburg.de/Publications/2017/IMSKDB17

[70] Z. Ren *et al.*, "Unsupervised deep learning for optical flow estimation," in *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, San Francisco, CA, USA, Feb. 2017, pp. 1495–1501.

[71] S. Meister, J. Hur, and S. Roth, "Unflow: Unsupervised learning of optical flow with a bidirectional census loss," *CoRR*, vol. abs/1711.07837, Nov. 2017. [Online]. Available: http://arxiv.org/abs/1711.07837

[72] J. Zhu *et al.*, "Unpaired image-to-image translation using cycle-consistent adversarial networks," *CoRR*, vol. abs/1703.10593, Mar. 2017. [Online]. Available: http://arxiv.org/abs/1703.10593

[73] J. J. Yu, A. W. Harley, and K. G. Derpanis, "Back to basics: Unsupervised learning of optical flow via brightness constancy and motion smoothness," in *Computer Vision – ECCV 2016 Workshops*, G. Hua and H. Jégou, Eds. Amsterdam, The Netherlands: Springer International Publishing, Oct. 2016, pp. 3–10.

[74] W.-S. Lai, J.-B. Huang, and M.-H. Yang, “Semi-supervised learning for optical flow with generative adversarial networks,” in *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Long Beach, CA, USA: Curran Associates Inc., Dec. 2017, pp. 353–363.

[75] M. Jaderberg *et al.*, “Spatial transformer networks,” *CoRR*, vol. abs/1506.02025, June 2015. [Online]. Available: http://arxiv.org/abs/1506.02025

[76] Y. Gan *et al.*, “Off-apexnet on micro-expression recognition system,” *Signal Processing: Image Communication*, vol. 74, pp. 129 – 139, May 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0923596518310038

[77] Q. Li *et al.*, “Micro-expression analysis by fusing deep convolutional neural network and optical flow,” in *2018 5th International Conference on Control, Decision and Information Technologies*, Thessaloniki, Greece, Apr. 2018, pp. 265–270.

[78] M. Peng *et al.*, “Dual temporal scale convolutional neural network for micro-expression recognition,” *Frontiers in psychology*, vol. 8, no. 1745, pp. 1745–1745, Oct. 2017. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/29081753

[79] Y. Liu *et al.*, “A main directional mean optical flow feature for spontaneous micro-expression recognition,” *IEEE Transactions on Affective Computing*, vol. 7, no. 4, pp. 299–310, Oct. 2016.

[80] T. Senst, V. Eiselein, and T. Sikora, “Robust local optical flow for feature tracking,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 9, pp. 1377–1387, May 2012.

[81] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[82] X. Guo *et al.*, “Deep clustering with convolutional autoencoders,” in *2017 International Conference on Neural Information Processing*, D. Liu *et al.*, Eds. Guangzhou, China: Springer International Publishing, Nov. 2017, pp. 373–382.

[83] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition*, Boston, MA, USA, June 2015, pp. 3431–3440.

[84] Y. Liu *et al.*, “Hourglass-shapenetwork based semantic segmentation for high resolution aerial imagery,” *Remote Sensing*, vol. 9, May 2017.

[85] J. Zhao *et al.*, “Stacked what-where auto-encoders,” *CoRR*, June 2015. [Online]. Available: https://arxiv.org/abs/1506.02351

[86] S. Baker *et al.*, “A database and evaluation methodology for optical flow,” *International Journal of Computer Vision*, vol. 92, no. 1, pp. 1–31, Mar. 2011. [Online]. Available: https://doi.org/10.1007/s11263-010-0390-2

[87] T. Samajdar and M. I. Quraishi, "Analysis and evaluation of image quality metrics," in *Information Systems Design and Intelligent Applications: Advances in Intelligent Systems and Computing*, J. K. Mandal *et al.*, Eds., vol. 340. New Delhi: Springer India, Jan. 2015, pp. 369–378.

[88] K. Seshadrinathan and A. C. Bovik, "A structural similarity metric for video based on motion models," in *2007 IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 1, Honolulu, HI, USA, Apr. 2007, pp. 869–872.

[89] G. Wolberg, *Digital Image Warping*, 1st ed. Washington, DC, USA: IEEE Computer Society Press, 1994.

[90] L. Velho, A. C. Frery, and J. Gomes, *Image Processing for Computer Graphics and Vision*, 2nd ed. Springer Publishing Company, Incorporated, 2014.

[91] J. Gomes *et al.*, *Warping and Morphing of Graphical Objects*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998.

[92] J. Gallier, *Geometric Methods and Applications: For Computer Science and Engineering*, 2nd ed. Springer Publishing Company, Incorporated, 2013.

[93] A. Reventós, *Affine Maps, Euclidean Motions and Quadrics*. Springer-Verlag London, 2011.

[94] M. Patil, "Interpolation techniques in image resampling," *International Journal of Engineering and Technology*, vol. 7, pp. 567–570, Sept. 2018.

[95] E. Kaya, "Spline interpolation techniques," *Journal of Technical Science and Technologies*, vol. 2, p. 47, Jan. 2014.

[96] R. Bellman, B. Kashef, and R. Vasudevan, "Dynamic programming and bicubic spline interpolation," *Journal of Mathematical Analysis and Applications*, vol. 44, no. 1, pp. 160 – 174, Oct. 1973. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0022247X73900334

[97] X. Wang and A. Gupta, "Unsupervised learning of visual representations using videos," in *Proceedings of the 2015 IEEE International Conference on Computer Vision*. Santiago, Chile: IEEE Computer Society, Dec. 2015, pp. 2794–2802. [Online]. Available: https://doi.org/10.1109/ICCV.2015.320

[98] N. Dubey, S. Ghosh, and A. Dhall, "Unsupervised learning of eye gaze representation from the web," in *2019 IEEE International Joint Conference on Neural Networks*. Budapest, Hungary: IEEE, July 2019, pp. 1–7.

[99] P. George and H. Borouchaki, *Delaunay Triangulation and Meshing: Application to Finite Elements*. Hermès, 1998.

[100] F. Razafindrazaka, "Delaunay triangulation algorithm and application to terrain generation," Master's thesis, International Institute for Software Technology, Macau, China, May 2009.

[101] C. Ericson, *Real-Time Collision Detection*. USA: CRC Press, Inc., 2004.

[102] D. J. Butler *et al.*, “A naturalistic open source movie for optical flow evaluation,” in *Proceedings of the 12th European Conference on Computer Vision*, A. Fitzgibbon *et al.*, Eds. Florence, Italy: Springer Berlin Heidelberg, Oct. 2012, pp. 611–625.

[103] P. J. Huber, “Robust estimation of a location parameter,” *Annals of Mathematical Statistics*, vol. 35, no. 1, pp. 73–101, Mar. 1964. [Online]. Available: https://doi.org/10.1214/aoms/1177703732

Vita

Muhannad Alkaddour was born in Aleppo, Syria, and received his bachelor of science degrees with magna cum laude in Mechanical Engineering and Mathematics from the American University of Sharjah, UAE in 2018.

Muhannad joined the Mechatronics Graduate Program at AUS in 2018, where he both studied and worked as a graduate teaching and research assistant. His current research interests include deep learning, image processing and computer vision, mechanical vibrations, control systems, and mobile robots.