DISSERTATION

IMPROVING GESTURE RECOGNITION THROUGH SPATIAL FOCUS OF ATTENTION

Submitted by

Pradyumna Narayana

Department of Computer Science

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Fall 2018

Doctoral Committee:

Advisor: Bruce A. Draper

Ross J. Beveridge
Charles W. Anderson
Christopher Peterson

ABSTRACT

IMPROVING GESTURE RECOGNITION THROUGH SPATIAL FOCUS OF ATTENTION

Gestures are a common form of human communication and important for human computer interfaces (HCI). Most recent approaches to gesture recognition use deep learning within multi-channel architectures. We show that when spatial attention is focused on the hands, gesture recognition improves significantly, particularly when the channels are fused using a sparse network. We propose an architecture (FOANet) that divides processing among four modalities (RGB, depth, RGB flow, and depth flow), and three spatial focus of attention regions (global, left hand, and right hand). The resulting 12 channels are fused using sparse networks. This architecture improves performance on the ChaLearn IsoGD dataset from a previous best of 67.71% to 82.07%, and on the NVIDIA dynamic hand gesture dataset from 83.8% to 91.28%.

We extend FOANet to perform gesture recognition on continuous streams of data. We show that the best temporal fusion strategies for multi-channel networks depends on the modality (RGB vs depth vs flow field) and target (global vs left hand vs right hand) of the channel. The extended architecture achieves optimum performance using Gaussian Pooling for global channels, LSTMs for focused (left hand or right hand) flow field channels, and late Pooling for focused RGB and depth channels. The resulting system achieves a mean Jaccard Index of 0.7740 compared to the previous best result of 0.6103 on the ChaLearn ConGD dataset without first pre-segmenting the videos into single gesture clips.

Human vision has α and β channels for processing different modalities in addition to spatial attention similar to FOANet. However, unlike FOANet, attention is not implemented through separate neural channels. Instead, attention is implemented through top-down excitation of neurons corresponding to specific spatial locations within the α and β channels. Motivated by the covert attention in human vision, we propose a new architecture called CANet (Covert Attention Net),

that merges spatial attention channels while preserving the concept of attention. The focus layers of CANet allows it to focus attention on hands without having dedicated attention channels. CANet outperforms FOANet by achieving an accuracy of 84.79% on ChaLearn IsoGD dataset while being efficient ($\approx$35% of FOANet parameters and $\approx$70% of FOANet operations).

In addition to producing state-of-the-art results on multiple gesture recognition datasets, this thesis also tries to understand the behavior of multi-channel networks (*a la* FOANet). Multi-channel architectures are becoming increasingly common, setting the state of the art for performance in gesture recognition and other domains. Unfortunately, we lack a clear explanation of why multi-channel architectures outperform single channel ones. This thesis considers two hypotheses. The *Bagging* hypothesis says that multi-channel architectures succeed because they average the result of multiple unbiased weak estimators in the form of different channels. The *Society of Experts* (SoE) hypothesis suggests that multi-channel architectures succeed because the channels differentiate themselves, developing expertise with regard to different aspects of the data. Fusion layers then get to combine complementary information. This thesis presents two sets of experiments to distinguish between these hypotheses and both sets of experiments support the SoE hypothesis, suggesting multi-channel architectures succeed because their channels become specialized.

Finally we demonstrate the practical impact of the gesture recognition techniques discussed in this thesis in the context of a sophisticated human computer interaction system. We developed a prototype system with a limited form of peer-to-peer communication in the context of blocks world. The prototype allows the users to communicate with the avatar using gestures and speech and make the avatar build virtual block structures.

ACKNOWLEDGEMENTS

The successful completion of my dissertation was only possible because of the support I received from multiple people who I owe my gratitude to. Thanks to my advisor, Dr. Bruce A. Draper, for giving me an opportunity to be a part of Computer Vision lab. You played a major role throughout my graduate studies. Your guidance, encouragement, feedback and support made this dissertation possible. I am immensely grateful to you for giving me the freedom to work on problems that I was interested in. The discussions I had with you broadened my perspective about the field of computer vision. I would always cherish the past seven years I worked with you.

My thanks as well to Dr. Ross Beveridge, for providing a different perspective to my research. You thoroughly critiqued my research, writing and presentation skills and always encouraged me to be better at presenting my research. Thank you to my committee members: Dr. Charles Anderson for introducing me to the field of Machine Learning, Dr. Chris Peterson for explaining the mathematical notions behind deep neural networks. I also am grateful to all of my fellow graduate students in the computer vision lab. I deeply appreciate your collaboration, support, and advice. I am truly thankful to have had the opportunity to work and learn from you all. Thanks to the faculty and staff of Computer Science department for all your help.

Thank you to my family for your love, support and encouragement. Thanks to you mom for teaching me everything in life. You are my first teacher and the best teacher. Thank you dad for being the most amazing human being I ever knew. You are the best father and always gave me whatever I asked for. Both of you encouraged me to dream big and aim higher. You are my biggest inspiration and I accredit all my achievements to both of you. Thank you brother for believing in me and being very supportive of my dreams. You are one of my major strengths. Finally, I need to thank the love of my life, Harini, for being by my side all the time. You sacrificed your career to stay with me and support me. I talked more about my research with you than anyone else. You proof read more of my documents and listened to more practice talks than anyone else. You

thought me a lot about work life balance and made sure I was having fun in my life. You made my life beautiful and I feel extremely lucky to have you in my life.

DEDICATION

I would like to dedicate this thesis to my mom, dad and Harini.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Motivation

Gestures are a natural form of human communication. When accompanying speech, gestures convey information about the intentions, interests, feelings and ideas of the speaker [53]. Gestures are even more important in noisy environments, at a distance, and for people with hearing impairments. In these scenarios, gestures replace speech as the primary means of communication, becoming both more common and more structured [75].

In collaborative tasks, gestures are as important as speech. Wang *et al*. explored the use of gestures in natural interactions between two people when working together in a constrained, simplified physical construction task [126]. Their experiments revealed that humans working on a collaborative task can finish the task more quickly when they can communicate using both gesture and speech. If only one mode is available, the task time increases significantly. Moreover, there was no significant difference between task finishing times in the speech only and gesture only modes.

Gestures are therefore an important part of communication among humans, and we intuit that gestures will aid communication between humans and computers as well. Unfortunately, current technology only allows people to communicate with machines using speech (e.g. Siri, Google Now, Amazon Echo). Perhaps motivated by a similar analysis, in 2015 the Defense Advanced Research Projects Agency (DARPA) funded the “Communication With Computers (CWC)" program [15]. The goal of CWC is to push the limits of human-computer interaction with the eventual goal of making human-computer communication as natural as human-human communication. If humans can communicate and work collaboratively with machines, the way they do with other humans, there are applications in areas ranging from biocuration to music composition to joint physical activities between humans and computers.

To make human-computer interaction with gestures a reality, computers must be able to understand human gestures. Automatic gesture recognition is an important domain of computer vision research with a large literature; see [10, 42, 101] for surveys. However, gesture recognition is still not a solved problem. Gesture recognition is challenging because: 1) there is a large diversity in how people perform gestures (intra class variance); 2) many gestures look similar (inter class similarity); 3) some gestures involve the whole body, while others involve only specific body parts (at which point motions of the uninvolved parts become noise); 4) gestures occur within complex environments including cluttered backgrounds, varying illumination, clothing and skin color. This thesis advances the state-of-the-art for gesture recognition.

1.2 Current Techniques and Shortcomings

Recent advances in deep learning have revolutionized the field of Computer Vision. Automatic feature learning makes deep neural network based methods an excellent choice for various recognition tasks. Convolutional neural networks (CNNs) are the current state-of-the-art technique for object recognition from a single frame [43, 44, 110]. CNNs can be extended to videos either by using CNNs to extract features from frames and encoding the temporal information using a recurrent neural network (RNN) [17, 134] or by using 3D CNNs [114]. All current state-of-the-art gesture recognition techniques use one of these these two strategies [77, 82, 123, 136].

Current state-of-the-art gesture recognition techniques based on CNN+RNN or 3D CNNs process the entire field of view (FOV). However, this is not how human visual perception works. Human visual perception retrieves information from the visual field by the process of selective attention [50]. Processing the entire field of view without selective attention makes it harder for learning to focus on the important information and discard the rest. Although a neural network may have access to the whole FOV, automatically learning to process the important parts of the FOV is analogous to finding a needle in a haystack. Moreover, nets that process full images are prone to overfit to background, illumination, clothing, etc. Instead, we reintroduce an old idea: focus of attention. Some gestures involve only hands, while others involve whole bodies. We use global

channels to process the whole video and look for gross motions, and focus channels to detect and process each hand.

There have been previous systems in the literature that divide processing between multiple channels. Simonyan and Zisserman [107] proposed a two stream convolutional network architecture for action recognition in videos. Their architecture consists of an RGB stream that captures information about scenes and objects in the video, and a optical flow stream that captures the motion of the camera and the objects. Their architecture draws inspiration from the two stream hypothesis [34] of the human visual cortex: the ventral stream performs object recognition and the dorsal stream recognizes motions. Many other systems, e.g. [77, 82, 123, 136], also process multiple data modalities using multiple streams. These systems divides processing by modality, not locality. Both are good ideas: we propose two architectures that divide processing both spatially and by modality.

A problem with dividing processing into channels is that the resulting information from each channel must be fused back together for classification. There is a spectrum of approaches in the literature for fusing information from parallel channels. On one extreme, channel outputs can be averaged together, as in [82, 123, 136]. This is a "weak" technique in the sense that it requires neither training nor *a priori* knowledge, but still produces better results than using a single channel. It is not clear that it makes the best use of the information available in the multiple channels, however. On the other extreme are systems that stack features from multiple channels and train a fully connected neural layer on top to do the classification [51]. This is a "strong" technique in that it relies on training and may produce better results by intelligently combining relevant information from multiple channels. However, the number of weights to be learned are proportional to the number of features times the number of channels, so this technique is susceptible to overfitting and becomes infeasible for large number (> 3) of channels.

Typical gesture datasets have tens of thousands of short videos containing hundreds of thousands of frames, e.g [82, 119, 125]. At frame level, it might seem like these datasets have enough data to prevent overfitting. However, the frames are highly correlated to each other. At the video

level, there is not enough data to prevent overfitting while fusing multiple channels. It is therefore important to have a fusion mechanism with relatively few parameters. We introduce exactly such a mechanism called sparse network fusion.

1.3 Summary of Research: Focus of Attention

By re-introducing the idea of spatial focus of attention and fusing channels through sparse network fusion, we raise the state-of-the-art for recognition accuracy from 67.71% to 84.79% for the ChaLearn IsoGD dataset, and from 83.8% to 91.28% for the NVIDIA dataset. We also surpass the human accuracy of 88.4% on the NVIDIA dataset. When processing continuous data, we achieve a mean Jaccard Index[1] of 0.7740 compared to the previous best result of 0.6103 on the ChaLearn ConGD dataset. The ChaLearn [119] datasets are the largest and most varied gesture datasets available, with 249 gestures from a variety of domains including mudras (Hindu/Buddhist hand gestures), Chinese numbers, and diving signals. On the other hand, the NVIDIA driving gesture dataset [82] is specific to a single domain: touch-less interfaces in cars.

We propose two architectures in this thesis that build on the idea of focus of attention, and on previous systems that use multiple channels to process different data modalities, e.g. [77, 82, 123, 136]. These architectures separate processing by both modality and spatial location, and finally fuse the information from the resulting channels using sparse fusion networks.

The first architecture, FOANet, divides processing into $M \times L$ channels, where M is the number of modalities and L is the number of spatial locations (body parts) focused on. FOANet uses spatial focus of attention (FOA) to restrict some channels to focus on specific body parts, namely hands. Gestures have both global and local components. Some involve sweeping motions of the arms and torso, while others are defined by detailed hand poses. FOANet trains multiple sub nets with specific purposes: global channels process the whole video and look for gross motions, while focused channels detect and process each hand. FOANet introduces a separate channel for every

[1]The Jaccard index measures the average relative overlap between true and predicted sequences of frames for a given gesture.

focus region L (including the global "whole image" region) and modality M. In this thesis, we process 4 modalities - RGB, depth, and two types of flow fields - and 3 locations - global, right hand, and left hand (if visible). The result is $M \times L$ channels processing different types of localized data, as shown in Figure 1.1. Figure 1.1 is the architectural instantiation used in Chapters 3 through 6 for the ChaLearn IsoGD and ChaLearn ConGD datasets.

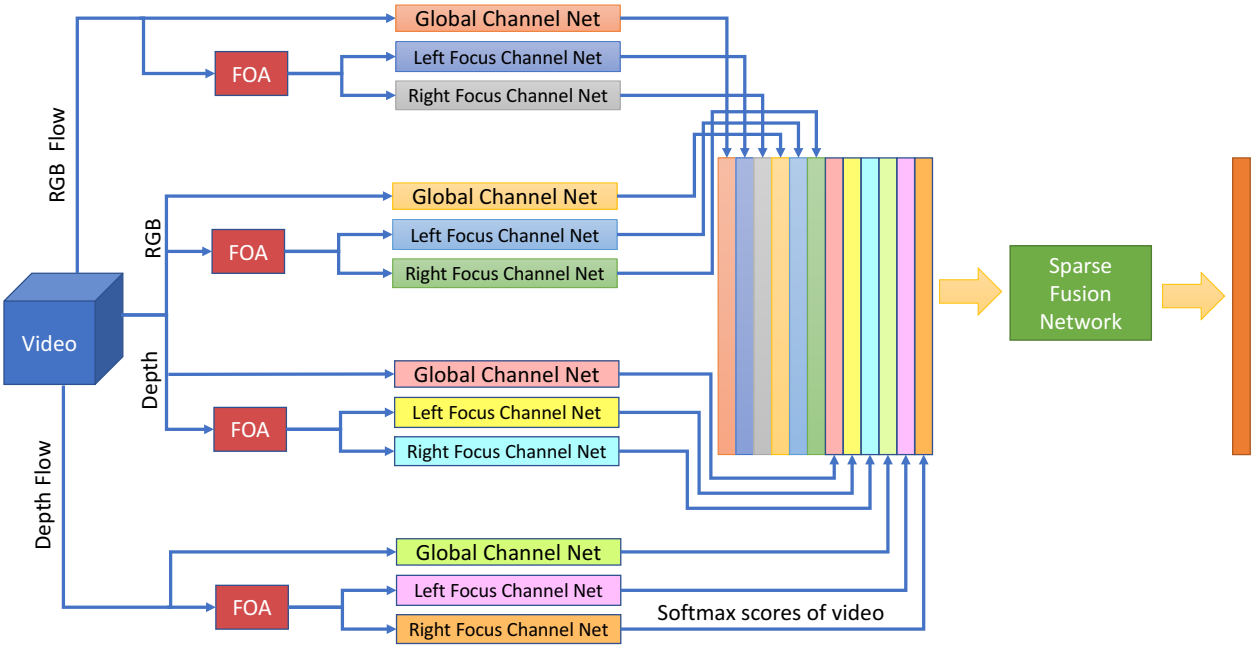

Figure 1.1: The FOANet Network Architecture instantiated for the ChaLearn IsoGD dataset. The architecture consists of a separate channel for every focus region (global, left hand, right hand) and modality (RGB, depth, RGB flow and depth flow). To create the focus channels, an FOA module is used to detect hands. The video level softmax scores from 12 channels are stacked together. Sparse fusion combines softmax scores according to the gesture type.

The $M \times L$ channels of FOANet are only useful if the data can be fused back together. It is tempting to train a neural net to fuse the high dimensional features vectors learned by the Convolutional neural networks inside each channel, but with only 35K training videos in IsoGD (and far fewer in NVIDIA), there is not enough data to avoid overfitting. Even training a fully connected layer to fuse the softmax output vectors (which have far fewer dimensions than the feature vectors) from $M \times L$ channels would be problematic. This is why many multi-channel systems simply average the channel outputs, e.g. [82, 123, 136]. Instead, FOANet uses a sparsely connected neural

layer with one weight per label $\times$ channel. For every type of gesture, the sparse fusion layer learns to combine information from the different spatial regions and data modalities. This allows the network to learn the relative importance of different spatial channels (e.g. right hand vs. left hand) and different modalities (e.g. RGB vs. depth) for each gesture, while keeping the total number of weights small and therefore minimizing the risk of overfitting.

The proposed FOANet architecture is a multi-channel architecture. Multi-channel architectures are becoming increasingly common, setting the state-of-the-art for performance in gesture recognition challenges. Unfortunately, we lack a clear explanation of why multi-channel architectures outperform single channel ones. Chapter 6 considers two hypotheses. The *Bagging* hypothesis says that multi-channel architectures succeed because they average the result of multiple unbiased weak estimators in the form of different channels. The *Society of Experts* (SoE) hypothesis suggests that multi-channel architectures succeed because the channels differentiate themselves, developing expertise with regard to different aspects of the data. The fusion layers then selectively combine complementary information. We report two experiments to distinguish between these hypotheses. The first measures the drop in individual channel performance when the input is degraded by removing high frequency, color, or motion information. The second looks at fusion weights relative to gesture properties. Both experiments support the SoE hypothesis, suggesting multi-channel architectures succeed because of channel specialization.

In summary, FOANet is an extension of the two stream architecture of Simonyan and Zisserman [107] as depicted in Figure 1.1. FOANet processes four modalities instead of two because the gesture recognition datasets being analyzed contain depth as well as RGB data. The contribution of this thesis in terms of FOANet, however, lie in three major components. The first is in introducing channels for spatial attention. Chapter 3 describes this in more detail, and evaluates its contribution to the overall FOANet architecture. The second contribution lies in the temporal fusion mechanisms applied within channels, as described and evaluated in Chapter 4. The third is the sparse fusion network for combining information from many channels. This is described and

analyzed in Chapter 5. Finally, Chapter 6 analyzes how the multiple channels in FOANet relate to each other.

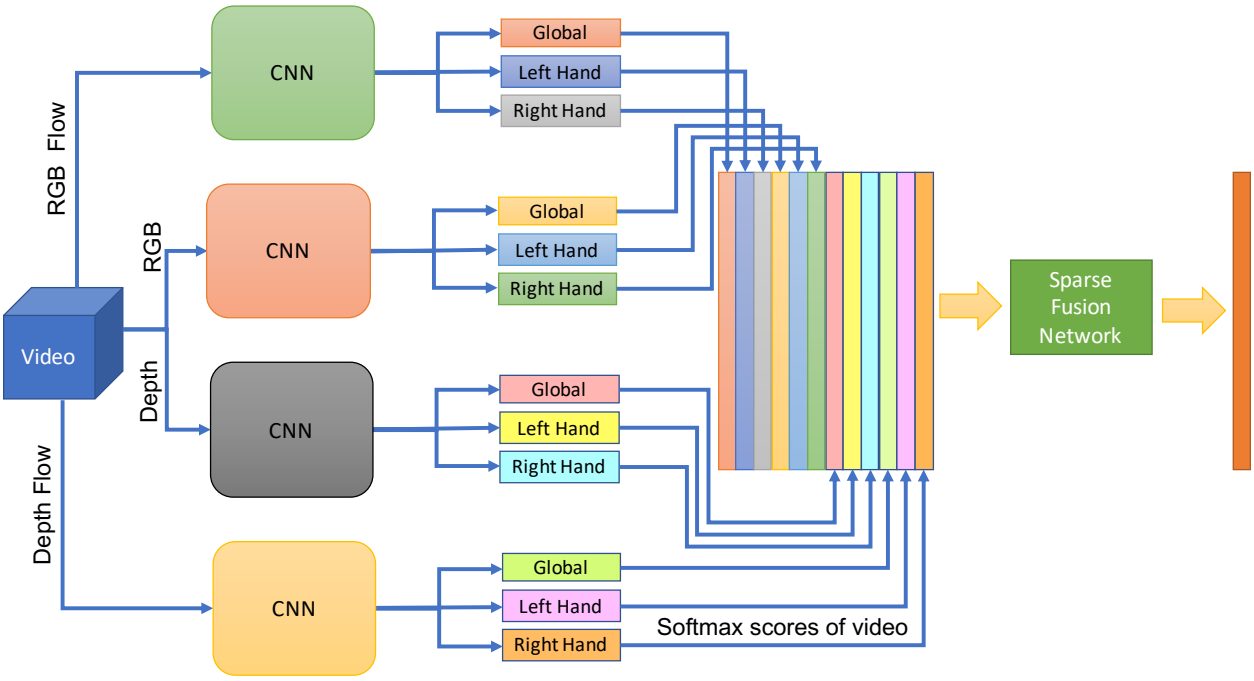

Figure 1.2: The CANet Network Architecture instantiated for the ChaLearn IsoGD dataset. The architecture consists of a CNN for every modality (RGB, depth, RGB flow and depth flow) and the CNNs have a separate classification head for every focus region (global, left hand, right hand). The video level softmax scores from 12 channels are stacked together. Sparse fusion combines softmax scores according to the gesture type.

CANet is the second architecture proposed in this thesis. CANet is inspired by FOANet, in particular the success of spatial attention and sparse fusion. Unlike FOANet that divides processing into $M \times L$ channels (M is the number of modalities and L is the number of spatial focus locations), CANet combines the attention channels resulting in only M channels. Although the attention channels are merged, CANet still preserves the concept of spatial attention by outputting L classification scores - one for each focus region as shown in Figure 1.2. This results in $M \times L$ classification scores similar to FOANet, however with fewer parameters and operations than FOANet. These classification scores are produced from the focused features generated by zeroing out the features outside the spatial focus of the focus region. The attention channels are merged to exploit the complementary information between different attention channels and to reuse the features for

different attention regions. The resulting CANet architecture is inspired by human vision, and the result is a system slightly more accurate and much more efficient than FOANet. More specifically, CANet achieves an accuracy of 84.79% on ChaLearn IsoGD dataset outperforming FOANet by 2.72% with only $\approx$ 35% of FOANet parameters and $\approx$70% of FOANet operations.

The ideas proposed in this thesis can be applied to real time gesture recognition. We demonstrate a real world application of gesture recognition in the context of human computer interaction. We developed a prototype system with a limited form of peer-to-peer communication in the context of blocks world in which the signaler is a person but the builder is an avatar with a virtual table and virtual blocks. The signaler can see a graphical projection of the virtual world, and communicate to the avatar through speech and gesture. The avatar communicates back through speech, gesture, and action, where an action is to move a block. The prototype allows the users to communicate with the avatar using gestures and speech and make the avatar build virtual block structures.

1.3.1 Contributions

This thesis presents two new architecture, FOANet and CANet, that explicitly focus their attention on hands. The main contributions of this thesis are:

1. State of the art recognition accuracies on the ChaLearn IsoGD [119], NVIDIA [82] and ChaLearn ConGD [119] datasets.

2. Two novel gesture recognition architectures with focus of attention.

3. A novel sparse network architecture for fusing channels.

4. New insights about temporal fusion for continuous gesture recognition, resulting in different fusion strategies for different channels.

5. Analysis of multi-channel networks for gesture recognition.

6. A real time demonstration of gesture recognition in a sophisticated HCI context.

1.3.2 Scope

Many factors can impact the performance of a gesture recognition system. Unfortunately, the evaluation of all factors is beyond the scope of this thesis. We choose not to focus on the evaluation or optimization of any of the following:

- Different CNN architectures: We choose ResNet-50 as the CNN architecture inside FOANet for practical reasons, as discussed in Section 2.2.2.
- Location features: Focus channels are provided with 14 location features (7 for each hand) to give them information about the location and orientation of the hands. These features provide location information to the spatial focus channels, but no experiments are performed to select the best location features from the space of all possible location features.
- Location features are passed through one fully connected layer with 14 neurons and a tanh activation function before being appended to ResNet-50 2048 feature vector. This seems to work well, but alternative subnets are not investigated.
- Hand detection: For ChaLearn IsoGD dataset, we use hand detection results by Liu *et al*. [68] and for NVIDIA dataset, we use HandSegNet of Zimmermann and Brox [138]. No evaluation or optimizations are done on hand detection.
- Combinations of temporal and channel fusion: Experiments are not performed on all combinations of temporal fusion and channel fusion techniques.
- Data augmentation techniques: Only horizontal flipping and random crops are used as discussed in Section 3.3.1.
- Learning rate schedules.

1.3.3 Document Outline

The rest of the document is organized as follows. Chapter 2 discusses the role of gestures in human communication, introduces datasets, and provides a greater discussion of related research

in attention, action recognition, and gesture recognition domains. Chapter 2 also includes a discussion on sensors. Global and focus channels are discussed in Chapters 3. Temporal fusion strategies are discussed in Chapter 4. Chapter 5 discusses fusion strategies for combining information from multiple channels. Chapter 6 analyzes multi-channel networks for gesture recognition. CANet is introduced in Chapter 7. Chapter 8 discusses the real world human computer interaction system where the ideas of this thesis are used for gesture recognition. Finally Chapter 9 concludes this thesis with a discussion about future work.

Chapter 2

Related work

This thesis touches on many diverse topics. Since the goal of this chapter is to provide the necessary background to understand the chapters that follow, it too touches on many topics. Readers may want to read just the sections that pertain to their interests.

Section 2.1 discusses the role of gestures in human communication. This material comes mostly from the field of psychology. This is followed by an overview of existing computer vision techniques to recognize gestures and actions in Section 2.2. This overview explains the two general approaches to action/gesture recognition: hand crafted features based recognition and deep neural network based recognition. A discussion of attention in neural networks follows in Section 2.3.

The second half of this chapter is more tightly focused on the background to understand specific experiments. Section 2.4 discusses the datasets used in this thesis, including the current state-of-the-art approaches for each dataset. Section 2.4 also discusses hand detection methods for each dataset (Section 2.4.1, 2.4.2). Hand detection is an important component of our system built largely on prior work by others. While not itself a contribution of this thesis, understanding FOANet as a whole requires understanding how hands are detected. Finally, a discussion of sensors in Section 2.5 concludes the chapter. This section also discusses methods to compute flow fields - another important component of our system built on prior work by others.

2.1 Role of Gestures

Gestures are a natural form of human communication. Gesturing is a "robust phenomenon, found across cultures, ages and tasks; even in individuals blind from birth" [31]. When accompanied by speech, gestures becomes "imagistic and analog" [31]. The gestures accompanied by speech are non-codified informal hand movements. Such gestures have the potential to reflect thoughts that may themselves be relatively unexamined by both speaker and listener. When gestures stands on their own and carry the full burden of communication, "they assume a language-like

form, with structure at word and sentence levels" [31]. This becomes extremely important in noisy environments, at a distance, and for people with hearing impairments.

In addition to communication, gestures serve a cognitive purpose in aiding in lexical access and retrieval and verbal working memory [30, 55]. Gestures reduce the cognitive burden by shifting from verbal to spatial memory, thereby freeing up resources that can be allocated to other tasks. For example, pointing improves young children's performance on counting tasks. Gestures also bring novel or contradictory information into a learner's repertoire without disrupting the current communication. As neither the listener nor speaker are consciously aware of the novel/contradictory ideas expressed by gestures, gesture is an ideal medium to consider notions that are not fully developed.

Gesture is a precursor to speech. Gestures serve as a vehicle for expressing thoughts to prelinguistic children. Children gesture starting around 10 months by pointing (deictic gestures) [3]. At the same time, children also use gestures to represent physical entities (iconic gestures). They represent a fish by opening and closing their mouth, or a bird by flapping their hands [2, 48]. Children combine gesture and speech to increase their communicative range. Interestingly, "children begin producing gesture-speech combinations prior to their first two-word utterances" [32].

Sign languages such as ASL are codified linguistic systems and are structured at phonological, morphological and syntactic levels. Such systems assume the full burden of communication. Interestingly, even deaf children of hearing parents who have not been exposed to sign language primarily communicate with gestures [84, 113]. More interestingly, their non-codified gestures assume the language-like forms similar to a codified communication system such as ASL. This non codified gesture systems is invented by deaf children by looking at the spontaneous gestures that their hearing parents use as they talk. "While the deaf childrens' gesture systems have language-like structure, the hearing parents' gestures do not have such structure" [31].

Gestures that can substitute speech and have meaning independent of speech are called emblems [18]. "Emblems are used to insult, praise or regulate the behavior of a communication partner" [52]. Emblems are used and understood consciously in communication. Speakers produce

emblems consciously, and listeners are consciouly aware of having seen an emblem produced. Emblems occur on their own and not "combined into gesture strings to make longer and more complex sentences. Emblems do not form a linguistic system" [31].

Gestures that accompany speech offer a different view into the mind of the speaker by reflecting speakers' feelings and emotions [27]. In addition, these gestures convey substantive information about a task - information that may not easily conveyed by the spoken language. The gestures can be used to depict notions ranging from the concrete (big, small, tall) to the abstract (mathematical notions, and even limits in calculus [73]). These spontaneously produced gestures have meaning associated with them. Interestingly, the meanings of these gestures are reliably interpreted by the listeners and this interpretation matches the meanings the speaker intends [28]. So, "gestures are not random hand movements but rather reveal substantive beliefs about the task at hand" [31]. McNeill classifies these spontaneously produced gestures as follows [74]:

1. **Iconic** gestures are the gestures that amplify or modulate the semantic content of co-occurring speech. They depict aspects of spatial images, actions, people, or objects by adding detail to the mental image that the speaker is trying convey.

2. **Metaphoric** gestures are visual representation of abstract ideas and categories that are impossible to represent directly.

3. **Beat** gestures are the hand movements that moves along with the rhythmical pulsation of speech and emphasize the speech. Beats convey minimal or no extra information and have the same form regardless of the content.

4. **Deictic** or pointing gestures indicate real, implied or imaginary persons, objects, directions, etc. These gestures are strongly tied to the conversational space.

In collaborative tasks, gestures are as important as speech. Wang *et al*. explored the use of gestures in natural interaction between two people when working together in a constrained, simplified task (specifically a physical construction task) [126]. In this study, pairs of participant were

asked to collaboratively build different pre-determined structures using wooden blocks. Participants were put in separate rooms and could communicate through live video (and audio) with each other. One participant was given the role of builder and was provided with a set of 12 wood cubes. The other participant was given the role of signaler and was given a layout of these blocks. The signaler was assigned the task of communicating to and directing the actor to replicate the layout; the actor needed to respond to the signaler's commands by placing and arranging the blocks on the table. The experiments were conducted in three different conditions: 1) **Speech Only** - participants were not able to see each other, and could only rely on speech to communicate (the signaler could only see the blocks on the table) 2) **Gesture Only** - participants could only see but not hear each other, restricting participants to using only nonverbal communication 3) **Gesture and Speech** - both audio and video were enabled, allowing participants to use any natural combination of speech and gesture they wished.

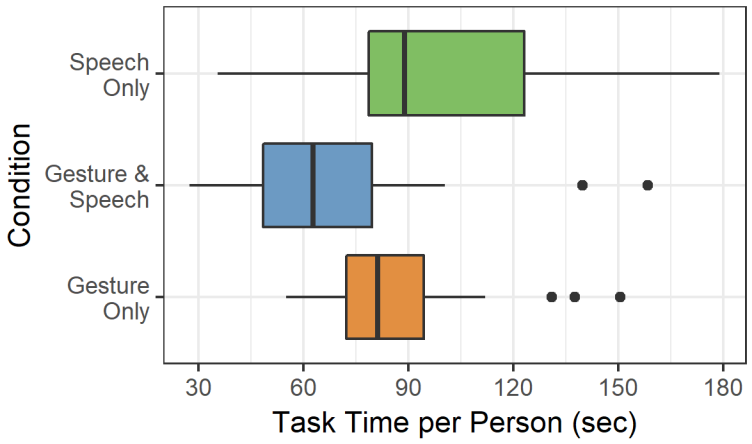

Figure 2.1: Average task time per person for Gesture Only, Speech Only, Gesture and Speech conditions [126].

The box plot in Figure 2.1 shows the average task/trial time for the Gesture Only, Speech Only, Gesture and Speech conditions. The average task time for the Gesture Only condition is 90.3 ± 26.5 seconds whereas the average for the Gesture and Speech condition was 69.6 ± 33.5 seconds, and for Speech Only condition, the average was 97.4 ± 39.5 seconds. The statistical

significance results using ANOVA and Turkey HSD correction reveals that the Gesture and Speech condition was significantly faster than Speech Only ($p < 0.05$) and there is no difference between Gesture Only and Speech only conditions. This suggests that gestures are as important as speech in collaborative physical tasks.

2.2 Recognition of Gestures and Action

The previous section stressed the importance of gestures in human communication. This section focuses on how gestures can be automatically recognized by computers. Gesture recognition is a challenging domain because: 1) there is a large diversity in how people perform gestures, 2) some gestures involve gross motions, whereas other gestures involve localized fine motions or poses, 3) environmental conditions such as background, illumination, clothing and skin color may be uncontrolled, and 4) many gestures look similar. Gesture recognition is an active field of computer vision, and a large literature has developed on gesture recognition; see [10, 42, 101] for surveys. However, most of the methods in the literature do not distinguish between gesture recognition and action recognition and directly apply action recognition methods to gesture recognition tasks.

Gestures are not the same as actions, although they are related. While actions involve whole body, gestures tend to be localized on specific body parts, usually hands. As action recognition methods are unaware of the localization aspect of gestures, they may pay attention to the parts of the video that are unrelated to the gesture, resulting in poor performance.

This section discusses the state-of-the-art methods used in action recognition. In the literature, these methods are directly applied to gesture recognition domain. Section 2.2.1 discusses two recent hand crafted features used for action recognition. Deep learning based methods automatically learn features that can classify actions. Such methods are discussed in Section 2.2.2.

2.2.1 Hand Crafted Features

Before the advent of deep learning, hand crafted features dominated the field of computer vision. Good features are hard to craft by hand. Hand crafted features can take years of research and

are often domain specific. Hand crafted features have become almost obsolete as deep learning methods that automatically learn features have become mainstream. However, recently two new hand crafted features were proposed for action recognition, a domain that is similar in many ways to gesture recognition. These two new hand crafted features are near state-of-the-art for action recognition, and can be used in FOANet as additional channels. However, using hand crafted features as additional channels is beyond the scope of this thesis. Improved dense trajectories is the current state-of-the-art hand crafted feature based action recognition method on RGB videos [122], whereas super normal vectors is the current state-of-the-art hand crafted feature based action recognition method on depth videos [133]. These methods are discussed below.

Improved Dense Trajectories

Wang *et al*. proposed dense trajectories as an efficient video representation method [120]. Their method samples dense points from each frame for several spatial scales and tracks the dense points based on displacement information from dense optical flow. Motion boundary descriptors along the dense trajectories are used to remove camera motion. Histograms of Oriented Gradients (HOG) [14], Histograms of Optical Flows (HOF) [62] and Motion Boundary Histograms (MBH) [121] descriptors are computed for each trajectory in the space-time volume aligned with the trajectory. HOG captures the static appearance information whereas HOF and MBH capture motion information.

The improved dense trajectories method (iDT) improves the previous dense trajectory method by explicit camera motion estimation [122]. The camera motion is estimated based on the assumption that two consecutive frames are related by a homography. First, the correspondence between two frames is calculated using Speeded-Up Robust Features (SURF) [4] descriptors and dense optical flow. Then RANdom SAmple Consensus method (RANSAC) [26] matching is used to estimate homography and rectify the image to remove camera motion. In addition, a human detector is employed to remove outlier matches from the human body, as human motion is not constrained by the camera. The dense trajectories calculated after removing camera motion using homogra-

phy are called improved dense trajectories and is the current state-of-the-art hand crafted feature representation method for RGB videos.

Improved dense trajectories achieved state-of-the-art results on multiple action recognition datasets outperforming the previous hand crafted feature based action recognition methods by a significant margin [122]. On Hollywood2 dataset [71], iDT achieved an accuracy of 66.8% outperforming the previous best by over 4%. iDT achieved an accuracy of 60.1% on HMDB51 dataset [59] which is an improvement of nearly 5% over the previous best. iDT also achieved state-of-the-art results on Olympic Sports [86] and High Five datasets [90] with an accuracy of 90.4% (5.4% improvement) and 69.4% (7% improvement) respectively. Molchanov *et al*. applied improved dense trajectories to NVIDIA dataset [82]. iDT achieved an accuracy of 73.4% using color and optical flow streams of NVIDIA dataset. The current state-of-the-art method on this dataset that uses deep learning techniques (C3D + RNN) achieved an accuracy of 79.3% using the same modalities. Although, iDT didn't outperform C3D + RNN on this dataset, the fact that hand crafted features performance is comparable to deep learning based methods is impressive.

Super Normal Vectors

As depth maps have different properties than color images, features designed for color videos are unsuited for depth videos and don't perform well on them. Recently, Yang and Tian proposed a new feature representation for depth maps called super normal vectors [133]. A depth sequence can be viewed as a 4-dimensional hypersurface and a normal to this hypersurface captures both shape information and motion cues. The normals in a local spatio-temporal subvolume can be concatenated together to form a polynormal p.

A visual polynormal dictionary and its corresponding coefficients are computed using a bag of keypoints feature coding approach [13]. The weighted differences between polynormals and visual words are aggregated into vectors. For each visual word, the difference vectors are further aggregated by spatial average pooling and temporal max pooling. The aggregated vectors of all visual words are concatenated to form a feature vector. A depth sequence is subdivided into a set of

space-time cells using an adaptive spatio-temporal pyramid based on motion energy. The feature vectors from each spatio-temporal cells are concatenated to form a super normal vector.

Super normal vectors achieved the state-of-the-art results on MSRAction3D Dataset [66] (93.45%), MSRGesture3D [127] Dataset (94.74%), MSRActionPairs3D Dataset [88] (100%) outperforming the previous hand crafted feature based methods by nearly 2%. In addition, it achieved the state-of-the-art accuracy of 86.25% on MSRDailyActivity3D Dataset [128], outperforming the previous best by 0.5%. Molchanov *et al.* applied super normal vectors to depth modality of NVIDIA dataset [82] and achieved an accuracy of 70.7%. In comparison, C3D + RNN achieved an accuracy of 80.3% on depth modality of NVIDIA dataset. Albeit the fact that super normal vectors are hand crafted features, their performance is still comparable to the deep learning based methods.

2.2.2 Deep Neural Network

Over recent years, the practice of computer vision has been advanced through the adoption of Convolutional Neural Networks (CNNs) and other deep learning based methods (e.g. [43, 108, 110]). Deep learning based methods have become popular in part because they can automatically learn feature representations, thus avoiding time-consuming engineering. However, deep learning methods require large training sets. CNNs perform well on still images, thanks to large public datasets for training such as Imagenet [16]. However, deep learning based methods do not perform equally well on videos. The time domain in videos introduces additional complexity, and there is no equivalently large video datasets for training. This section discusses deep learning methods for videos.

CNNs for Still Images

Lenet-5 is the first CNN architecture proposed by LeCun *et al.* [64] in 1998. Lenet-5 architecture consists of 7 layers (3 convolution, 2 pooling, 2 fully connected layers). Although Lenet-5 architecture performed well on hand digit recognition task, CNNs were hard to train due to limited training data and computing resources.

CNNs gained a lot of momentum after a deep CNN was used to win the Imagenet [16] challenge in 2012. The winning architecture by Krizhevsky *et al.* [58] called ***AlexNet*** consists of 13 layers (5 convolution, 3 max pool, 2 normalization and 3 fully connected layers). AlexNet made the first use of the ReLU non linearity function. ReLU solves the vanishing gradient problem and is the most common non linear function used in CNNs today. Data augmentation and dropouts were introduced to combat the problem of overfitting. The development of GPUs made AlexNet training feasible. Zeiler and Fergus further improved AlexNet by tweaking the architecture hyperparameters. The architecture called ***ZFNet*** improves on AlexNet by expanding the size of the middle convolutional layers and making the stride and filter size on the first layer smaller [135].

VGGNet by Simonyan and Zisserman [108] is the next influential architecture in CNNs. VGGNet strictly used 3x3 convolution filters. A stack of three 3x3 convolution layers has the same effective receptive field as one 7x7 convolution layer. This in turn simulates a larger filter while increasing the depth of the net and including more non linearities. Moreover, smaller filters have fewer parameters than large filters. Figure 2.2 shows the architecture of VGG16 network.

Figure 2.2: Architecture of VGG16 network [1].

Szegedy *et al.* proposed an inception module (network in network architecture) in ***GoogLeNet*** [111]. An inception module applies multiple convolution filters of different receptive field sizes (1x1, 3x3, 5x5) in parallel along with a max pool operation. Different receptive size filters capture information at different levels of granularity. Figure 2.3 shows an inception block. Nine inception modules are stacked together, and a single fully connected layer is used to do the classification as shown

in Figure 2.4. The number of parameters is reduced as GoogLeNet uses a single fully connected layer, unlike AlexNet and VGGNet that uses multiple fully connected layers. GoogLeNet has 12x fewer parameters than AlexNet. Inception-3 [112] and Inception-4 [110] are further extensions of GoogLeNet architecture.

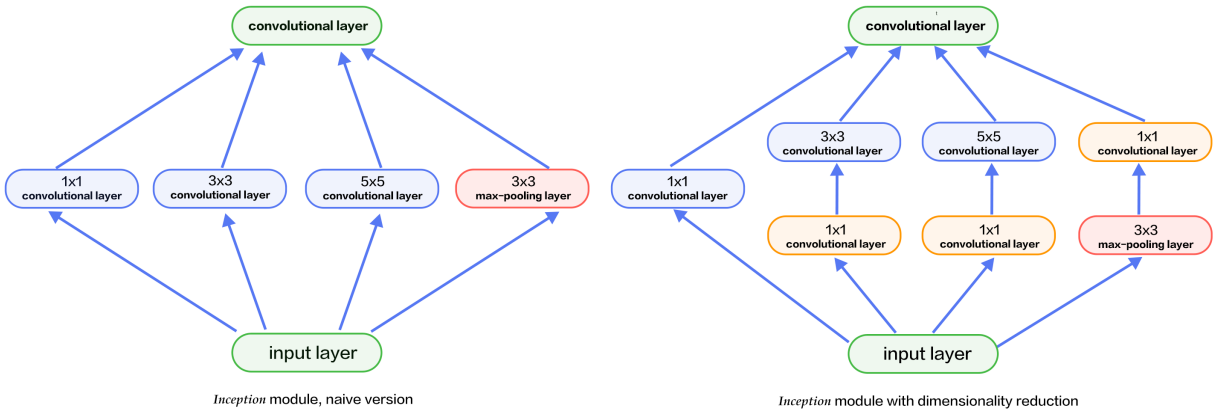

Figure 2.3: Inception Block [1].

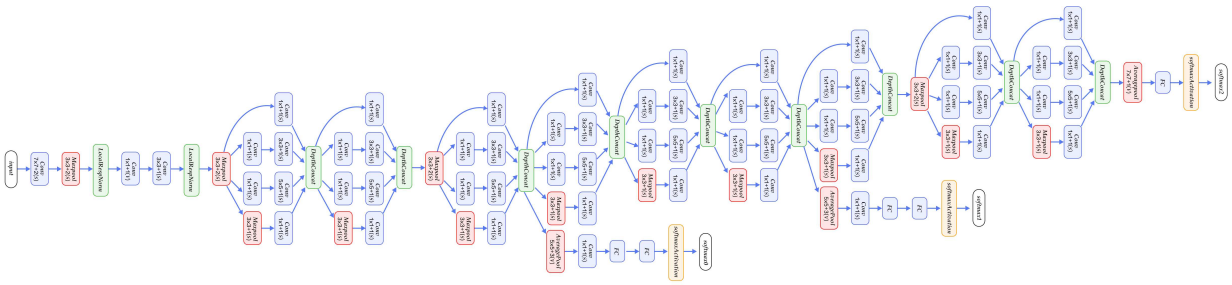

Figure 2.4: Architecture of GoogLeNet [111].

In 2015, He *et al*. proposed a very deep network using residual connections called ***ResNet*** [43]. Deep residual networks redefined state-of-the-art and won the first place in all five main tracks of the ImageNet and COCO 2015 competitions. The residual architecture has been proven as robust and is being used in many areas including image classification, object detection, semantic segmentation, speech, and language. Figure 2.5 shows the architecture of ResNet.

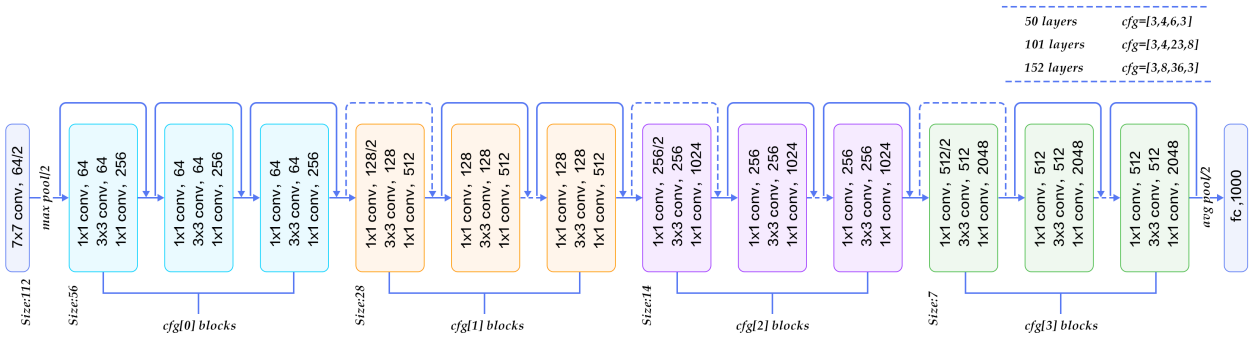

Figure 2.5: Architecture of ResNet [1].

As deep networks are harder to optimize, the training and test errors increase as we go deeper. In principle, deeper models are able to learn anything learned by shallower models. A proof by construction to this problem is to copy the learned layers from a shallower model and set the additional layers to be identity mappings. This is the motivation behind residual connections. The residual blocks try to fit a residual mapping instead of directly fitting a mapping as shown in Figure 2.6. The residual mapping is easy to optimize compared to the unreferenced mapping. If an identity mapping were optimal, the network would learn to push the residual to zero, which is easier than to fit an identity mapping by a stack of nonlinear layers.

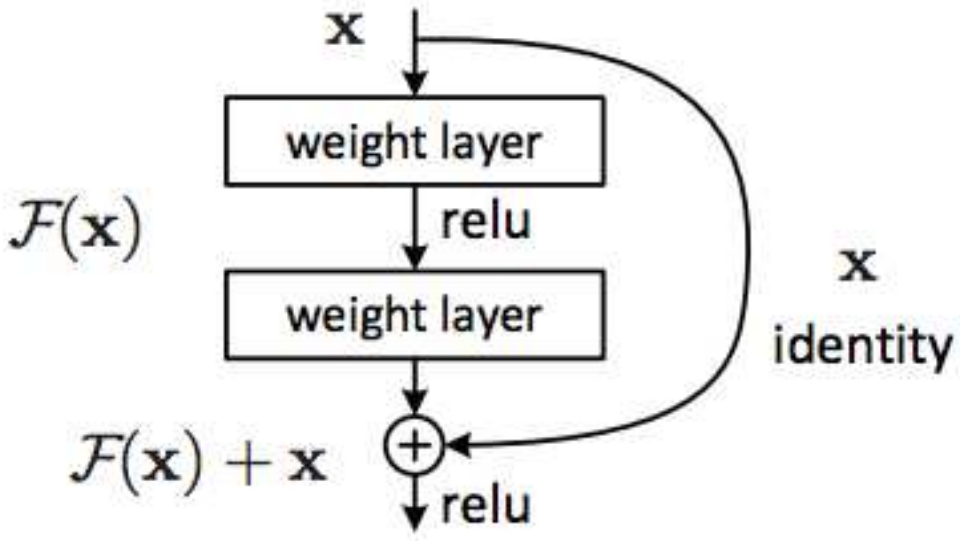

Figure 2.6: Residual block - fundamental building block of ResNet [43].

Different ResNet architectures of 34, 50, 101 and 152 layers deep are proposed. He *et al*. also proposed a 1001 layer deep ResNet by using identity shortcut connections and identity after-addition activations in the network [44].

CNNs for Videos

The CNN architectures discussed above are used for image classification. The simplest way to extend these architectures to videos is to run a CNN on each frame and then average the softmax scores for video classification. Another method to extend CNNs for video is to run a CNN on each frame of the video, extract the features, and then pool the features. A fully connected layer on top of the pooled features can be used for video classification. The advantage of the above methods is that they can take advantage of ImageNet pretraining. However, these methods ignore temporal structure. For example, it is hard to distinguish between clockwise and counter clockwise rotation.

We use these two methods of extending CNNs for videos in this thesis, however, with a slight variation. Instead of running a CNN on each frame, we run a CNN on sliding window of 10 frames. Ten frames are stacked together to form a 30 channel input image, which is then processed by a CNN. The advantage of using 10 frames is that the CNNs can learn motion information without losing the advantage of ImageNet pretraining.

CNN + RNN

The two methods of extending CNNs for videos discussed in previous section ignore temporal structure. Another approach to modeling temporal sequences is to add a recurrent neural network (RNN) after last pooling layer (in the place of fully connected layer) of a CNN. The RNN can model temporal structure and capture long-range dependencies. RNNs are usually placed after the last convolution/pooling layer so that the CNN can act as a feature extractor while the RNN models temporal structure as shown in Figure 2.7.

Vanilla RNNs that map input sequences to hidden states struggle to learn long-term dynamics due to the vanishing and exploding gradients problem [45]. Long Short Term Memory networks (LSTMs) address the problem of long-term dependency by using memory units that explicitly al-

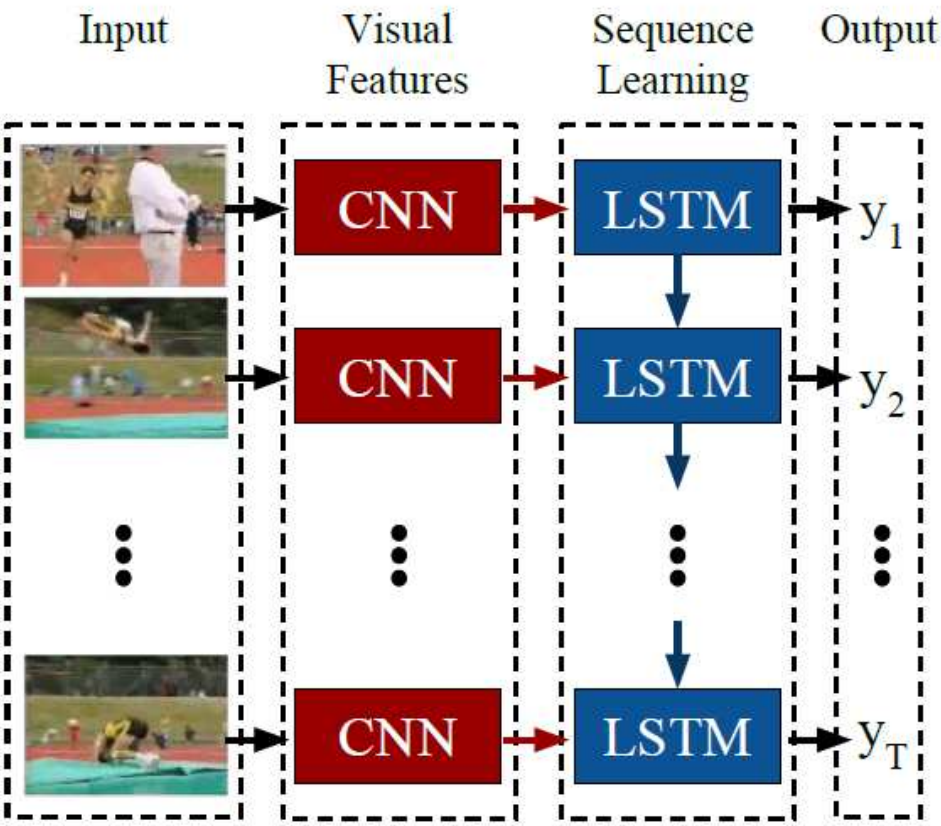

Figure 2.7: CNN + RNN architecture. This architecture can process variable-length video (left) with a CNN (middle left), whose outputs are fed into RNNs (LSTMs, middle-right), which finally produce a variable-length prediction (right). Both the CNN and LSTM weights are shared across time [17].

low the network to learn when to forget and when to update the previous information given new information [45]. Gated Reccurent Units (GRUs) are a simpler variant of LSTMs that achieves comparable performance to LSTMs with less parameters [11]. However, LSTMs always outperformed GRUs in our experiments.

3D Convolutional Networks

The CNNs discussed in the "CNNs for Still Images" section extract spatial features from single images. However, the features they extract lack motion information. 3D Convolutional Networks are a natural extension of 2D Convolutional Networks for videos that have the capability to capture spatio-temporal information. More importantly, they can create hierarchical representations of spatio-temporal data.

Tran *et al*. proposed a deep 3D Convolutional Network (C3D) that models appearance and motion simultaneously [114]. The C3D architecture takes video clips as input and consists of 8 3D convolution layers, 5 3D pooling layers, followed by two fully connected layers, and a softmax output layer. All the 3D convolution filters are of 3 x 3 x 3 dimensions.

C3D can also be used as a feature extractor for other video analysis tasks. To extract C3D features for a video, a video is split into 16 frame long clips with a 8-frame overlap between consecutive clips. These clips are passed through the C3D network and the activations from the first fully connected layer are extracted. The clip level activations are averaged to form a 4096D video descriptor [114].

Tran *et al*. further extended the C3D network by using residual connections (ResC3D) [115]. ResC3D outperforms C3D on multiple datasets while being 2 times faster at inference time, 2 times smaller in model size, and having a more compact representation. Although 3D Convolutional Networks capture spatio-temporal features, they have more parameters than 2D Convolutional Networks and are harder to train. For example, it was reported that, it took about two months to train a C3D on Sports-1M dataset [114].

Two stream networks

The two stream hypothesis of the human visual system is that human vision contains two pathways: the ventral stream to perform object recognition and the dorsal stream to recognize motion [34]. Inspired by this hypothesis, Simonyan and Zisserman [107] proposed a two stream convolutional network architecture for action recognition in videos. Their architecture consists of an RGB stream to capture information about scenes and objects in a video, and an optical flow stream to capture motions of the camera and objects. The optical flow stream has two values per pixel, one for horizontal motion and another for vertical. They average the ConvNet predictions from a single RGB image and a stack of 10 optical flow images. This method takes advantage of transfer learning from still image datasets like ImageNet.

Although the two stream architecture uses both motion and appearance information, it didn't register spatial cues with temporal cues (what is moving where). Feichtenhofer *et al*. extended

the two stream architecture by fusing the temporal and spatial streams after the last convolutional layer [25]. The fusion is done by stacking the two feature maps from temporal and spatial streams and then performing a 1x1 convolution to retain the feature map dimensions. They also recommend doing a 3D convolution followed by 3D pooling to fuse the two streams temporally after the fusion layer. Moreover, they observed that after fusing the motion stream into the appearance stream, one should retain the motion stream and reuse it at the end to increases performance.

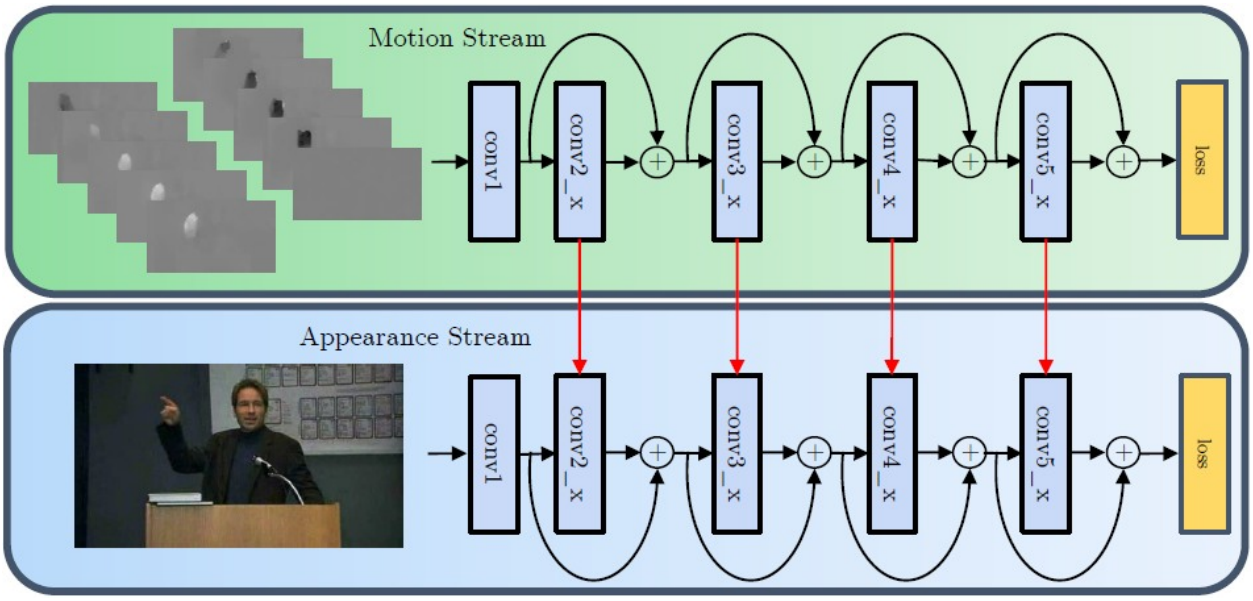

Figure 2.8: Architecture of two stream network with residual connections. The two networks separately capture spatial (appearance) and temporal (motion) information to recognize the input sequences [23].

Feichtenhofer *et al*. [22] extended this architecture further by using residual networks [43] and injecting residual connections from the motion stream into the appearance stream at multiple levels as shown in Figure 2.8. Moreover, they extend the temporal receptive field by adding temporal residual connections. The ResNet architecture is extended with temporal convolutions by transforming spatial filters in residual paths to temporal filters. This allows the network to take advantage of pre-trained ResNet on ImageNet. Figure 2.8 shows the two stream architecture with residual connections. Feichtenhofer *et al*. [23] further extended this architecture by using multiplicative gating for fusion instead of addition.

Similar to the two stream networks, our proposed architecture also divides processing among appearance (RGB and depth) and motion streams (RGB flow and depth flow). RGB flow and depth flow channels of our proposed architecture uses a stack of 10 images as input, similar to the motion stream of two stream network. However, we also use a stack of 10 images as input to RGB and depth channels (equivalent to appearance stream in two stream network), unlike the two stream architecture that uses a single image as input to appearance stream. More importantly, we divide processing spatially as well as by modality, unlike the two stream network that divides the processing only by modality.

CNN Architecture Used in this Thesis

The FOANet architecture consists of CNNs to process a sliding window of 10 frames stacked together as a 30 channel input image. A CNN is run on each 10 frame sliding window of the video and the information from all the sliding windows is temporally fused. There are many state-of-the-art CNN architectures and we choose ResNet-50 as the default CNN in this thesis.

Figure 2.9 shows the relative accuracies of many of the best known CNNs on the ImageNet dataset. It shows clearly the progress that has been made since the early days of AlexNet. Accuracy is not the only metric of interest, however. The more parameters a network has, the more data it needs to avoid overfitting, and the more time it needs for training. The more operations it performs per image, the slower it is in real-time applications. Figure 2.10 plots accuracy (vertical axis) versus G-ops per image (horizontal axis) and parameter count (circle size). In general, as networks get more accurate they also get larger and more complex, although there are exceptions: VGG-16 and VGG-19 are exceptionally poor choices in terms of complexity and parameter count.

The choice of a CNN for processing single images is important for FOANet. FOANet uses a CNN in every channel. For example, looking back at Figure 1.1, there are 12 CNNs in the FOANet for the ChaLearn IsoGD dataset, because there are 12 channels. In principle, any CNN would do. In practice, we want nets that are as accurate as possible, while still being resistant to overfitting (because the gesture datasets are relatively small) and efficient in terms of training (because we have to train so many of them).

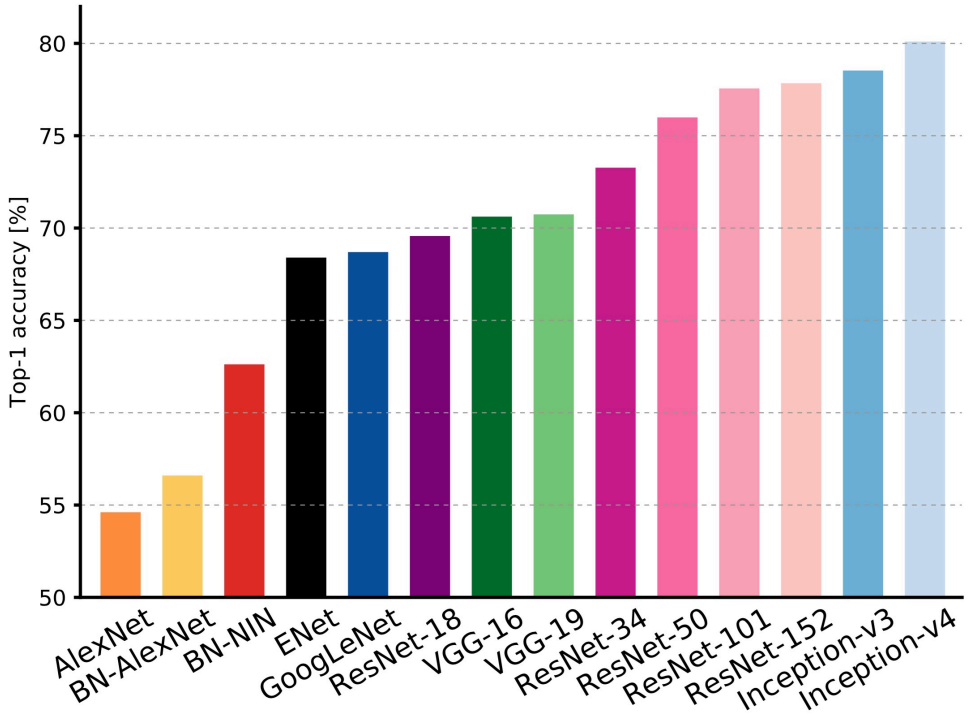

Figure 2.9: Top1 vs. network. Single-crop top-1 validation accuracies for top scoring single-model architectures on ImageNet [7]

.

For the experiments in this thesis, we use ResNet-50 as the default CNN. It is the best performing network in Figure 2.9 that fits on a single GPU in our lab. Moreover, the inference on ResNet-50 running on a Titan X GPU in our lab can be done in realtime (30 FPS) on a 128x128 image. One could substitute any of the less accurate nets to the left of ResNet-50 in Figure 2.9, but we would expect the accuracy of FOANet to drop. Given sufficient resources, one could alternatively substitute one of the four more accurate networks to the right of ResNet-50. In this case we would expect the performance of FOANet to improve, although only slightly since the performance of ResNet-50 is within 4% of the best performer.

2.3 Attention

Although there is a large literature on deep learning based methods for image and action recognition, there is not a lot of work in focus of attention in convolutional nets. Larochelle and Hinton

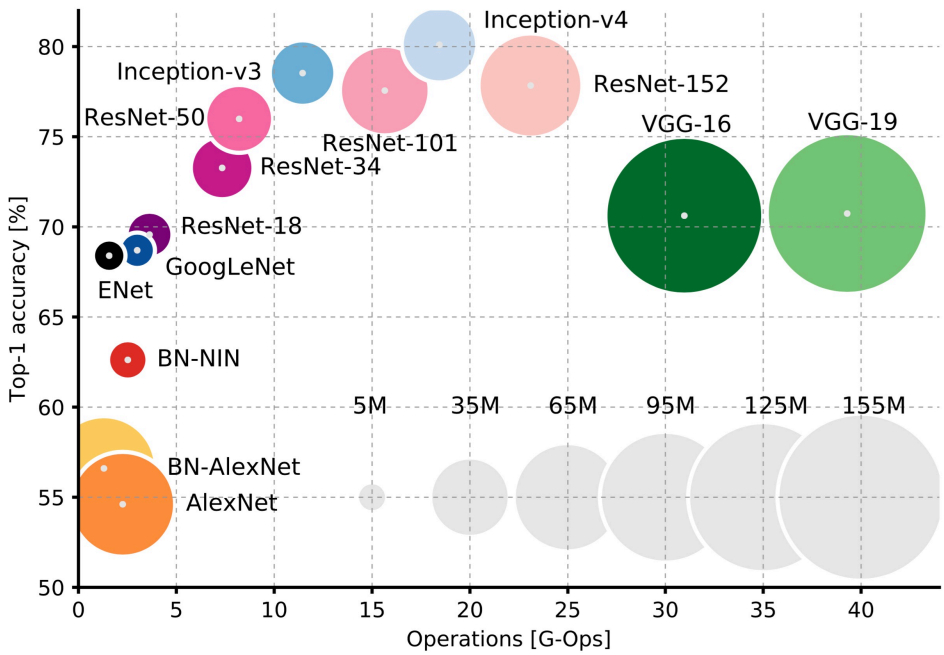

Figure 2.10: Top1 vs. operations, size α parameters. Top-1 one-crop accuracy versus amount of operations required for a single forward pass. The size of the blobs is proportional to the number of network parameters. The time to infer labels from a single image is proportional to the number of operations in single forward pass [7]

proposed a biologically inspired human vision system for object recognitionn [63]. Human vision uses a retina, which is a small area with high resolution and the resolution falls off rapidly with eccentricity. So, the human vision relies on a intelligent, top-down "fixation point strategy" for sequentially fixating parts of the scene that are relevant for the task at hand. Based on this model, the model proposed by Larochelle and Hinton uses a retina that only has enough high resolution pixels to cover a small area of the image. The model must therefore learn to focus on the relevant parts of the image. Moreover the model must decide on a sequence of fixations. Their proposed model combines the features at each fixation (glimpse), with the location of the fixation, before integrating the information with information from other glimpses of the same object. Larochelle and Hinton achieved comparable performance to the then state-of-the-art methods on two datasets.

Karpathy *et al.* proposed a multiresolution CNN that consisted of two streams - a low-resolution context stream and a high-resolution fovea stream [51]. The context stream received the downsampled frames at half the original spatial resolution, while the fovea stream receives the center region

at the original resolution. The size of the center region is equal to half the original spatial resolution. Fixing attention to the center of the frame takes advantage of camera bias in which objects of interest often occupy the center of the frame. The motivation of Karpathy *et al.* behind multiresolution CNN is to speed up training without decreasing accuracy. The idea of having a context stream and fovea stream is similar to our idea of training nets on global and focus channels. However, the idea of fixing attention to the center of the frame by relying on camera bias does not work on datasets like ChaLearn IsoGD (Section 2.4.1) and NVIDIA (Section 2.4.2) where subjects are not centered on the frame. Moreover, the attention in gesture recognition domain should be localized on a semantic object such as hands, rather than taking a center crop of the image.

Jaderberg *et al.* proposed spatial transformer networks that can spatially transform feature maps without extra training supervision [49]. They introduced a spatial transformer module that can be inserted into existing CNN architectures. The spatial transformer module has the ability to scale, crop, rotate and non-rigidly deform input feature maps. This allows networks to select regions of interest of an image (attention) and transform those regions to a canonical, expected pose to simplify recognition in the following layers. The spatial transformer module consists of three parts. The first part is a localization network that consists of a number of hidden layers followed by a regression layer to output the parameters of the spatial transformation (e.g. 6 parameters for affine transformation). The second part is a grid generator that creates a sampling grid based on the predicted transformation parameters. The sampling grid is a set of points where the input map should be sampled to produce the transformed output. The third part is the image sampler that takes feature map and the sampling grid as inputs, and produces the output map sampled from the input at the grid points. The spatial transformer module is differentiable and can be inserted at any point and in any number, into a CNN.

2.4 Datasets

This section gives an overview of datasets used in this thesis and describes the current state-of-the-art approaches relative to those datasets. The ChaLearn IsoGD dataset is introduced in

Section 2.4.1, and the NVIDIA dataset is discussed in Section 2.4.2. Section 2.4.3 presents the continuous ChaLearn ConGD dataset. Finally, other relevant datasets are discussed in Section 2.4.4.

2.4.1 ChaLearn IsoGD

ChaLearn has been conducting a series of challenges since 2011 to advance the state of the art in gesture recognition [19–21, 38–40]. The current ChaLearn LAP RGB-D Isolated Gesture Dataset (IsoGD) [119] dataset is one of the largest and most varied gesture datasets available, with 249 gestures from a variety of domains including mudras (Hindu/Buddhist hand gestures), Chinese numbers, diving signals, referee signals, traffic signals, helicopter signals, surgeon signals, gang signals, sign language, and italian gestures. The dataset is derived from the ChaLearn Gesture Dataset (CGD) and has 249 gesture labels performed by 21 different individuals.

The ChaLearn IsoGD dataset consists of both one handed and two handed gestures. Each RGB-D video is segmented (from ChaLearn ConGD dataset discussed below in section 2.4.3) to represent one gesture instance. The RGB and depth videos of the dataset are collected using the Kinect^{TM} camera at a resolution of 240 x 320 pixels at 10 frames per second. Figure 2.11 shows a single RGB and corresponding depth frame from 5 different videos of the ChaLearn IsoGD dataset. The dataset is split into three subject disjoint subsets: training, validation, and test. The training set consists of 35,878 videos from 17 subjects, the validation set consists of 5,784 videos from 2 subjects, and the test set consists of 6,271 videos from the other 2 subjects.

RGB and depth videos of the ChaLearn IsoGD dataset are not aligned as can be seen in Figure 2.11. However, the pixel coordinates from RGB images can be mapped onto depth images by the following transformation: $D = \frac{R-14}{0.93}$, where R is a coordinate in RGB image and D is its corresponding location in depth image.

ChaLearn 2017 Looking at People ICCV Challenge

The ChaLearn 2017 looking at People ICCV challenge attracted competitors from across the world [118], and the results of that challenge can be reasonably interpreted as reflecting the current state of the art in gesture recognition. The challenge consists of three tracks, and isolated gesture

Figure 2.11: A single frame from 5 different videos of the ChaLearn IsoGD dataset. Top row shows the RGB frames whereas the bottom row shows the corresponding depth frames.

recognition challenge is one of those tracks. The participants of ChaLearn IsoGD challenge are required to classify videos as one of 249 gestures. Participants were given access to a set of 35,878 labeled training videos, and a second set of 5,784 labeled validation videos. Participants were encouraged to develop the best system they could, training on the training videos and testing on the validation videos. At the conclusion of the challenge, participants were given access to a previously sequestered set of 6,271 labeled test videos. They were asked to evaluate their system on the test videos without modification.

Miao *et al.* [77] won the 2017 challenge by using a ResC3D network [114] and Temporal Segment Network [129] to extract features from RGB, depth and flow fields. First, a video enhancement pre-processing step is performed to account for illumination variations in RGB videos and noise in depth videos. Illumination normalization in RGB videos is done by Retinex [61] and noise in depth videos is removed using a median filter. Then, a key frame attention mechanism is used to resize the videos to a fixed size (32 frames). When resizing the videos, motion information is preserved by weighing different parts of the video according to the intensity of movement (measured using optical flow) in those parts. Features are extracted from the resized videos by passing them through ResC3D network. Features within each modality are fused using canonical correlation analysis, and an SVM classifies videos based on the fused features from the different modalities.

The SYSU ISEE team finished second in the 2017 ICCV challenge by processing skeleton data in addition to RGB, depth and flow fields. They used a combination of rank pooling, LSTMs and temporal streams, and fused the streams using average fusion. Other participants in the challenge [123, 136] also used C3Ds and some form of LSTM for temporal fusion. The resulting channels are fused together by averaging softmax scores.

The closest method to ours is the heterogenous networks of Wang *et al*. [123]. They use two types of networks: 3D ConvLSTMs to recognize gestures from videos and CNNs to recognize gestures from dynamic images constructed by rank pooling. They apply these networks at two spatial levels, namely body and hand level. These networks are run on RGB and depth modalities and finally scores from the 12 modalities are averaged together. The idea of having body level networks to capture global motion and hand level networks to recognize fine-grained gestures is similar to our global and focus channels, although the approach is significantly different. Wang *et al*. detect bounding boxes around the hands in every frame using F-RCNN [102]. They then compute a single large bounding box that contains the smaller bounding boxes from each frame, and set the pixels outside this box to zero. For gestures involving big motions and/or two hands, the bounding box may approach the full size of the image, in which case the purpose of the focus channel is defeated. In contrast, we detect the right and left hands and select attention windows around them, so that our focus nets are always focused on hands alone. Figure 2.12 compares the inputs to hand level net of Wang *et al*. and our focus channels. The top row shows five different frames from a video. These frames serve as an input to the body level nets of Wang *et al*. and our global nets. The second row shows the corresponding inputs to hand level nets of Wang *et al*. We can see that, even for a simple one handed emblem, the bounding box almost covers the entire height of the image. On the contrary, our focus channels always focus on the hands as shown in the third row of Figure 2.12. In addition, Wang *et al*. fuse channels by averaging their responses, whereas our sparse network fusion technique learns different weights for each gesture.

Although our system was developed after the ChaLearn 17 deadline, the experiments on ChaLearn IsoGD dataset in this thesis mimicked the experimental design of the challenge as closely as pos-

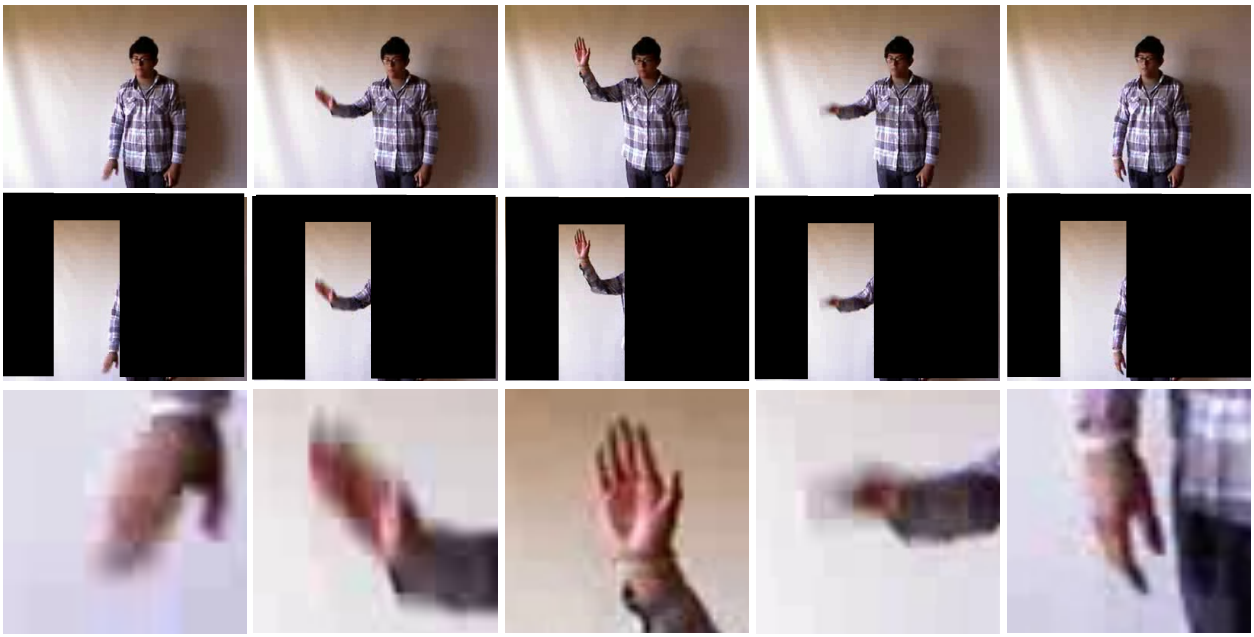

Figure 2.12: Comparison of inputs to the hand level nets of Wang *et al*. [123] and our focus channels. The top row shows five different frames from a video, which will be the inputs to the body level nets of Wang *et al*. and our global nets. The second row shows the corresponding inputs to hand level nets of Wang *et al*. The third row shows the inputs to our focus channels.

sible. The test videos were internally sequestered, and were not used during development. The system is incrementally developed by training on the training videos and testing on the validation videos. At the end, the system is evaluated only once and without modification on the test videos.

Hand Detection

For ChaLearn, we use the hand detection results provided by Liu *et al*. [67, 68] based on a two stream Faster R-CNN. Although the RGB and depth videos are captured from the same sensor in the ChaLearn IsoGD dataset, they are not registered. Therefore, the original depth video is aligned to RGB video by the mapping relationship between the color and the depth coordinate spaces. The frames from aligned videos are then passed through convolutional neural networks to extract feature maps. Feature maps from RGB and depth frames are stacked together. A Region proposal network [102] and an object classifier is run on the stacked feature maps to detect hands as shown in Figure 2.13. The bounding boxes from RGB images are mapped onto depth images

by the following transformation: $D = \frac{R-14}{0.93}$, where R is a coordinate in RGB image and D is its corresponding location in depth image.

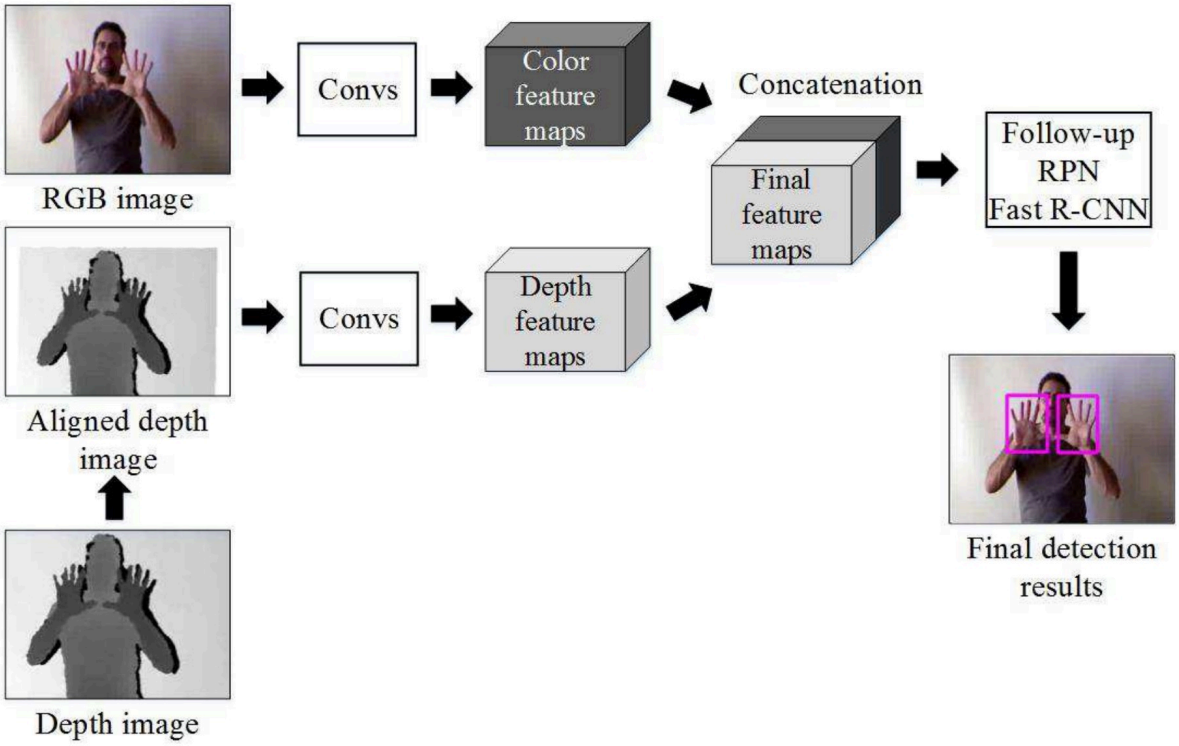

Figure 2.13: Hand detection pipeline of two-streams Faster R-CNN [68].

The hand detection results provided by Liu *et al.* do not differentiate between right and left hands. Skeletons[2] extracted from RGB frames using multi-person pose estimation code by Cao *et al.* [8] are used to distinguish left from right hands. The 2D pose estimation method associates body parts with individuals in the image using a non parametric representation called part affinity fields.The architecture jointly learns part locations and their association in a greedy bottom-up way.

Every frame in the video is passed through the real time pose estimation network to extract the skeletons. The skeletons extracted may have missing joints in some frames. Skeleton estimates are interpolated and extrapolated when necessary to fill in missing skeleton joints as necessary. For a

[2]A skeleton is a collection of joint locations. Multi-person pose estimation code by Cao *et al.* [8] estimates the positions of 18 joints.

Table 2.1: Gestures in NVIDIA Dataset

1. Move Hand Left	6. Two Fingers Right	11. Open Hand	16. Push Up	21. AntiClockwise
2. Move Hand Right	7. Two Fingers Up	12. Shake	17. Push Down	22. Two down
3. Move Hand Up	8. Two Fingers Down	13. One	18. Push Out	23. Close hand
4. Move Hand Down	9. Click	14. Two	19. Push In	24. Thumbs Up
5. Two Fingers Left	10. Beckon	15. Three	20. Clockwise	25. Ok

given frame, the hand bounding boxes provided by Liu *et al.* are mapped to the closest wrist joint to differentiate between right and left hands.

2.4.2 NVIDIA

The ChaLearn IsoGD dataset discussed in the previous section includes gestures from a variety of domains. The downside of the ChaLearn IsoGD dataset is that it is not closely tied to any specific HCI application. The NVIDIA driving gesture dataset, on the other hand mimics touchless interfaces in cars.

The NVIDIA dataset consists of 25 human computer interface gestures [82]. These gestures are captured from multiple sensors and viewpoints in a car simulator. The dataset consists of 1532 instances of dynamic hand gestures performed by 20 subjects under both bright and dim artificial lighting as shown in Figure 2.14. Subjects performed gestures with their right hand while observing the simulator's display and controlling the steering wheel with their left hand. Table 2.1 lists the 25 gestures in this dataset.

The NVIDIA dataset consists of front view color and depth videos acquired using the SoftKinetic DS325 sensor. It also consists of a pair of stereo-IR streams captured using top mounted DUO 3D sensor. The dataset provides 70-30 train test split, resulting in 1050 training and 482 test videos.

Molchanov *et al.* 's recurrent three-dimensional convolutional neural network is the best reported method on this dataset [82]. Molchanov *et al.* use a 3D-CNN to extract local spatial-temporal features from short clips. These features are input to a recurrent network, which aggregates transitions across several clips. A softmax layer on top is used to do the classification. A

Figure 2.14: A single frame from 5 different videos of NVIDIA dataset. Top row shows the RGB frames whereas the bottom row shows the corresponding depth frames.

connectionist temporal classification (CTC) cost function is used to train the network. The CTC cost function allows the network to predict class labels from in progress gestures in unsegmented input streams. They use RGB, depth, optical flow, IR image and IR disparity streams; the streams are fused by averaging the softmax scores.

As the SoftKinetic DS325 sensor used to acquire depth videos has a short range (0.15 m - 1.0 m), the depth videos are clean without any background. Even the color videos are relatively clean with a plain background. Although the color and depth videos are not spatially aligned, they can be aligned by an affine transformation. The affine transformation matrix is obtained by doing a pseudo inverse on 50 manually marked corresponding points on color and depth frames. The resulting affine matrix is

$$M = \begin{bmatrix} 1.274e+00 & 4.146e-02 & -3.033e+01 \\ 1.656e-02 & 1.307e+00 & -3.981e+01 \end{bmatrix}$$

We use color and depth videos and discard the stereo-IR streams for the experiments in this thesis. In addition, we use the dense optical flow [6, 89] computed from the color and depth stream (See Section 2.5.1). As a validation set is not provided with the dataset, we choose 1 subject from the training set to be the validation set. We follow the same experimental design as in ChaLearn by

incrementally developing our system by training on the training videos and testing on the validation videos. At the end, the system is evaluated only once and without modification on the test videos.

Hand Detection

For the NVIDIA dataset, hand detection results were not available. However, the hand is the closest object to the camera in the dataset. So, hands are detected in depth videos by considering the closest object to the camera. To segment hands in RGB videos, we use the HandSegNet of Zimmermann and Brox [138]. HandSegNet is a 16 layer network that is based on and initialized by the person detector of Wei *et al*. [130]. For a given RGB frame, HandSegNet returns a two channel image, one of which is a hand mask and the other one is the background mask.

2.4.3 ChaLearn ConGD Dataset

The ChaLearn ConGD dataset is the continuous version of the ChaLearn IsoGD dataset. It is a multimodal (RGB-D) dataset with 47,933 gesture instances in 22,535 videos. Similar to the ChaLearn IsoGD dataset, there are 249 gestures labels performed by 21 different individuals. Each video consists of one or more gestures performed with either one hand or two hands, and the goal of the dataset is to recognize gestures within continuous videos. Figure 2.15 shows various gestures in a single continuous video of ChaLearn ConGD dataset.

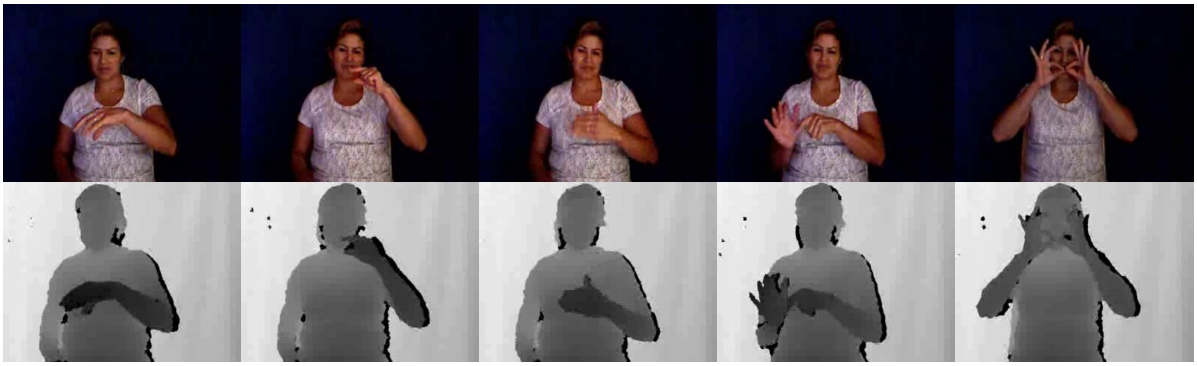

Figure 2.15: Different gestures in one video of ChaLearn ConGD Dataset. Top row shows RGB frames and bottom row shows the corresponding depth frames.

The dataset is split into three mutually exclusive subsets: training, validation, and test. The training set consists of 30,442 gestures from 14,314 videos performed by 17 subjects, the validation set consists of 8,889 gestures from 4,179 videos performed by 2 subjects, and the test set consists of 8,602 gestures from 4,042 videos performed by other 2 subjects.

The Jaccard index is used as the evaluation criterion for ChaLearn ConGD Dataset. The Jaccard index (the higher the better) measures the average relative overlap between true and predicted sequences of frames for a given gesture. For a class i in sequence s, let $G_{s,i}$ and $P_{s,i}$ be the set of ground truth and predicted frames respectively. The Jaccard Index $J_{s,i}$ for the i^{th} class in sequence s is calculated as:

$$J_{s,i} = \frac{|G_{s,i} \cap P_{s,i}|}{|G_{s,i} \cup P_{s,i}|} \tag{2.1}$$

The value of $J_{s,i}$ is set to zero when $|G_{s,i} \cup P_{s,i}|$ is 0. The Jaccard index for sequence s with l_s labels can be calculated as:

$$J_s = \frac{1}{l_s} \sum_{i=1}^{L} J_{s,i} \tag{2.2}$$

The mean Jaccard index $\overline{J_S}$ for the entire dataset is calculated as:

$$\overline{J_S} = \frac{1}{n} \sum_{j=1}^{n} J_{s_j} \tag{2.3}$$

ChaLearn 2017 Looking at People ICCV Challenge

As previously mentioned in Section 2.4.1, ChaLearn 2017 looking at People ICCV challenge consists of three tracks; continuous gesture recognition is another one of them [118]. The results of that challenge can be reasonably interpreted as reflecting the current state of the art in continuous gesture recognition. The participants of ChaLearn ConGD challenges are required to localize and classify gestures as one of 249 gestures from videos that consists of one or more gestures. Participants were given access to a set of 14,314 labeled training videos, and a second set of 4,179 labeled validation videos. Participants were encouraged to develop the best system they

could, training on the training videos and testing on the validation videos. At the conclusion of the challenge, participants were given access to a previously sequestered set of 4,042 labeled test videos. They were asked to evaluate their system on the test videos without modification.

Liu *et al*. won the ConGD challenge by pre segmenting the video and using C3D networks to classify the segmented clips [69]. Videos are segmented based on the observation that a subject raises hands up when beginning to sign a gesture and puts hands down after performing the gesture. However, this method of segmenting video into isolated gestures is specific to ChaLearn ConGD dataset and can't be generalized for other datasets. Hands are detected using two streams Faster R-CNN, as discussed in Section 2.4.1. A stable hand position is calculated from the initial several frames and is used as height threshold to determine the gesture boundaries. A gesture boundary starts when a hand is above the height threshold and ends when both hands are lower than the height threshold. The segmented video clips are then resized to 32 frames length and are passed through a C3D to extract FC6 features. Features from RGB and depth videos are stacked together and a SVM is used for classification. Before the segmented videos are passed to C3D, the regions other than hands and head are blocked to alleviate the distractor regions, such as background, clothing, body and so on. Although the idea of focusing on hands is similar to our focus of attention, blocking other regions may result in strong edges and add noise.

Wang *et al*. segment the video into gesture frames and transition frames by running a two class CNN on every frame of a video [124]. The middle frame of a continuous segment of transitional frames is considered as the gesture boundary. The segmented gestures are then used for gesture recognition. The segmented gestures from depth modality are converted into Dynamic Images and Motion Dynamic Images and a CNN is used for classification. Saliency maps are extracted from RGB videos, and segmented RGB and saliency maps are classified using a C3D followed by a Convolutional LSTM. The softmax scores from various networks are then averaged together.

Camgoz *et al*. used a probabilistic forced alignment approach to jointly segment and recognize gestures [12]. They train a 3D CNN to classify gestures (plus a silence class to identify neutral gestures) by extracting windows from a prior distribution of likely regions. The trained 3D CNN

is used to calculate the posterior distribution which is used as the prior to sample windows for the next training stage. This process is repeated until the recognition performance of the network has converged on validation set. During inference, they run the 3D CNN on all windows and segment the video at frames where the silence class probability is maximum. The softmax scores in individual segments is averaged to assign a label to the segment.

2.4.4 Other Datasets

There are many other datasets in gesture recognition domain such as:

- UTD Multimodal Human Action Dataset (UTD-MHAD) [9]: UTD-MHAD is a multimodal dataset collected using Kinect sensor with 27 different actions (20 of them are hand gestures). Recently, they released a second version of their dataset that consists of 10 hand gestures.

- Elicited Giant Gallery of Naturally Occurring Gestures Dataset (EGGNOG) [125]: EGGNOG is a large dataset with naturally occurring human gestures. The dataset contains over 8 hours of RGB video, depth video, and Kinect v2 body position data of 40 subjects. The dataset provides the physical motions and poses of the head, torso, arms and hands and the intent of the signaler.

- 20BN-Jester Dataset [117]: The 20BN-JESTER dataset is a large collection of densely-labeled video clips that show humans performing 27 pre-defined hand gestures in front of a laptop camera or webcam. The dataset consists of only RGB videos. So, it is not a multi modal dataset.

- VIVA [87]: VIVA is a multimodal hand gesture dataset captured using a Kinect device under real-world driving settings. The dataset consists of 19 hand gesture classes performed by 8 subjects. The gestures were performed either with the right hand by subjects in the driver's seat or with the left hand by subjects in the front passenger's seat.

2.5 Sensors

The previous section discussed various gesture recognition datasets. The data in these datasets are captured using different sensors. This section discusses sensors used for data collection in gesture recognition. Section 2.5.1 covers RGB sensors whereas Section 2.5.2 covers depth sensors. Infrared sensors are discussed briefly in Section 2.5.3.

2.5.1 RGB

An RGB camera or visible light camera is a device that forms an image using visible light. RGB sensors are the most common and ubiquitous sensors available today. They capture images in three channels: Red, Green and Blue. Most RGB sensors capture videos at 30 FPS, whereas some sensors can capture at 60 and 120 FPS. The resolution of RGB cameras vary from 320 x 240 to 10 Megapixels.

Frames from RGB video capture static information. Optical flow can be used to capture motion information between two adjacent frames. Optical flow is the pattern of motion of objects between two images. It is 2D vector field where each vector is a displacement vector showing the movement of points from first image to second. Optical flow works on the assumption that the pixel intensities of an object do not change significantly between consecutive frames and neighboring pixels have similar motions.

Optical flow can be computed from two adjacent frames sampled using pyflow [89] - a python wrapper for dense optical flow [6]. As it is computationally not feasible to calculate optical flow on the fly, we pre-compute flow fields. Moreover, we store the optical flow values as RGB images to make it easy to store and work with the optical flow values. To store the flow fields as RGB images, the horizontal and vertical components of the flow values are clipped at (-20, 20) and the magnitude of the motion is calculated. The horizontal, vertical and magnitude components are rescaled to [0, 255] range independently and saved as red, green, blue channels respectively of an RGB image. Figure 2.16 shows the optical flow computed on two adjacent frames from five different videos.

Figure 2.16: Optical flow computed from two adjacent RGB frames of five different videos. The top row shows a frame t from 5 different videos and the middle row shows the next frame $t + 1$ for the respective videos. Optical flow calculated from these adjacent frames is shown in bottom row.

2.5.2 Depth

Recent advance in depth sensors bring advantages to the fields of action and gesture recognition as depth cameras have many advantages compared to conventional RGB cameras. Depth maps[3] captured from the depth sensors provide additional shape cues which are helpful to get more informative geometric description, Moreover, it is easy to recover skeleton joints from a single depth map. In addition, absence of color and texture makes it easy to detect and segment humans in depth maps. Depth maps are robust to lighting and can even work in completely dark environments.

Depth sensors can be classified as structured light sensors, time of flight sensors and stereo sensors based on the technique used to estimate depth of a scene. More information regarding these methods is provided below.

Structured Light: Structured light cameras operate by projecting a sequence of known patterns onto a scene, and then observing the deformed patterns from a camera at a different direction. Depth information can be extracted by analyzing the disparity of the observed pattern from the

[3]Depth map is an image whose pixel values store the distance values of the surfaces of objects in the scene from camera.

original projected pattern. Kinect v1 is an example of a depth sensor that used structured light for depth estimation. ChaLearn IsoGD and ChaLearn ConGD are collected using Kinect v1.

Time of Flight: Depth sensors that work on Time of Flight (ToF) approach estimate depths by measuring the time taken for the light emitted by the sensor to travel to an object and be reflected back to the sensor. ToF sensors actively illuminates the scene using near infrared (NIR) intensity-modulated, periodic light. Various objects in the scene cause a time shift $\phi[s]$ in the optical signal and the time shift is transformed into depth as $d = \frac{c\phi}{4\pi}$, where c is the speed of the light. Kinect v2 and SoftKinetic DS325 are the examples of ToF depth cameras. NVIDIA dataset is collected using SoftKinetic DS325 whereas EGGNOG is collected using Kinect 2.

Stereo: Stereo vision is the extraction of 3D information from two digital images. Stereo images are obtained by using two or more cameras displaced horizontally, to obtain two different views of a scene. A two cameras setup mimics human binocular vision. By comparing two images capturing the different views of a scene, 3D information can be extracted in the form of a disparity map, which encodes the difference in horizontal coordinates of corresponding image points. The values in this disparity map are inversely proportional to the scene depth at the corresponding pixel location.

Although the depth is calculated in terms of distance units, it is not always encoded in the same way. To make it easy to save and distribute the datasets, depth maps are often encoded as grayscale images by transforming the distance units to image range (0-255). Although encoding the depth data as gray scale images saves memory, precision in the depth data is lost. EGGNOG is the only dataset that provides depth data in depth units. Other datasets such as NVIDIA, ChaLearn IsoGD and ChaLearn ConGD encodes the depth maps as grayscale images. The ChaLearn dataset goes a step further and encodes depth images as a video which further decreases the precision of the depth data and introduces artifacts as video compression is lossy.

Similar to RGB videos, we also calculate optical flow on depth videos as shown in Figure 2.17. As the optical flow is designed only for RGB images, we can see that the optical flow computed on depth images is not as clean as the optical flows computed on RGB images (see Figure 2.16).

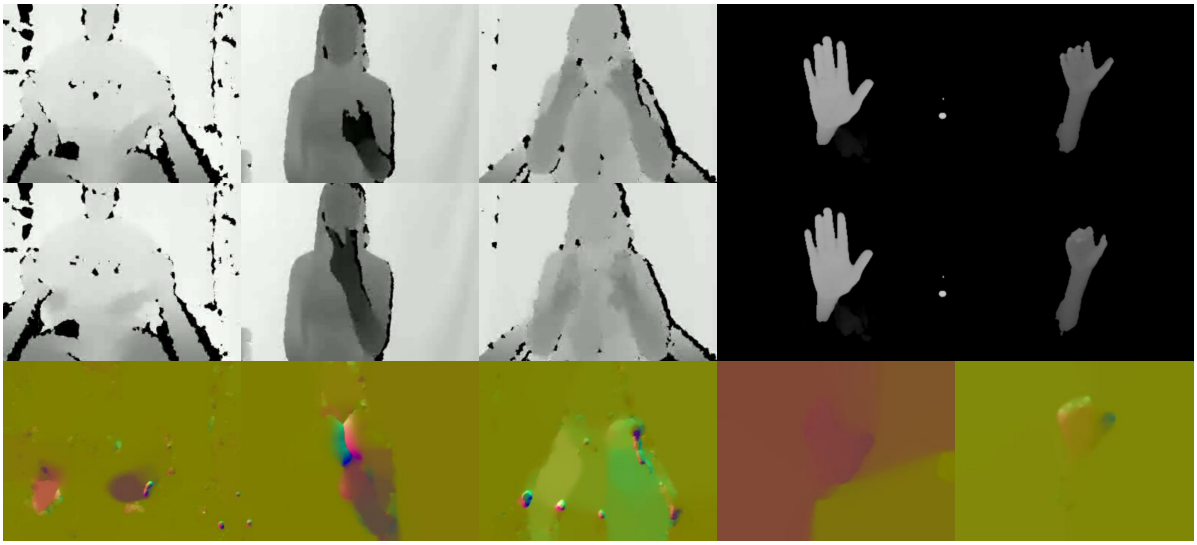

Figure 2.17: Optical flow computed from two adjacent depth frames of five different videos. The top row shows a frame t from 5 different videos and the middle row shows the next frame $t + 1$ for the respective videos. Optical flow calculated from these adjacent frames is shown in bottom row. The frames shown in this figure directly corresponds to the RGB frames shown in Figure 2.16

.

2.5.3 InfraRed

Infrared cameras (also called thermographic cameras or thermal imaging cameras) are devices that form images using infrared radiation. Infrared cameras operate in wavelengths as long as 14,000 nm (14 μm). All objects emit a certain amount of black body radiation as a function of their temperatures. The higher the object's temperature is, the more infrared radiation is emitted as black-body radiation. An infrared camera can detect this radiation and convert it into an electronic signal, which is then processed to produce a thermal image. The advantage of infrared cameras is that they can work even in total darkness. The NVIDIA dataset is the only dataset that provides a pair of stereo-IR streams using a top mounted DUO 3D camera. As IR data is not a common modality and is included in only one of the datasets described above, IR data is not used in this thesis.

Chapter 3

Attention

We propose a novel architecture called FOANet that achieves state-of-the-art results on the ChaLearn IsoGD and NVIDIA datasets. Figure 3.1 shows the FOANet architecture with 12 channels instantiated for the ChaLearn IsoGD dataset. FOANet embodies three major ideas. The first is to expand the multi-channel approach and reintroduce the old idea of spatial focus of attention. Existing multi-channel approaches uses different channels to process different data modalities. FOANet expands on this idea by dedicating a channel to every pair of modality and attention regions. In Figure 3.1, we depict 3 spatial attention regions L: one for the whole scene (the global channel), and one for each of the hands (focus channels). The idea is to create an architecture that reflects the structure of gestures, which are combinations of large body movements and fine hand motions. Moreover, for each spatial attention channels, separate modality M (RGB, depth, RGB flow and depth flow) channels are dedicated. The result is M $\times$ L channels processing different types of localized data, as shown in Figure 3.1.

The second idea is to process a sliding window of 10 frames using 2D CNNs and temporally fuse information within channels. Compared to images, videos pose an additional challenge in the form of variable temporal duration. Unlike images that have fixed dimensions (W $\times$ H), videos have variable dimensions due to variable temporal duration T (W $\times$ H $\times$ T). As CNNs can't process variable dimensional data, some recent works normalize the temporal lengths of the videos to a fixed length [77, 80, 81]. Temporal normalization is done by resampling videos using nearest neighbor interpolation (NNI) by dropping or repeating frames. Temporally normalized videos are then processed using 3D CNNs. Although temporally normalizing the videos works for gesture recognition in segmented videos, this technique can't be extended to continuous gesture recognition in streaming data. The input to the focus channels is a stack of 10 spatial image windows focused around one of the hands (in the 5th frame of 10 frame sliding window). As a 10 frame sliding window is around $\frac{1}{3}$rd of a second for gestures captured at 30 FPS, the hands don't move a

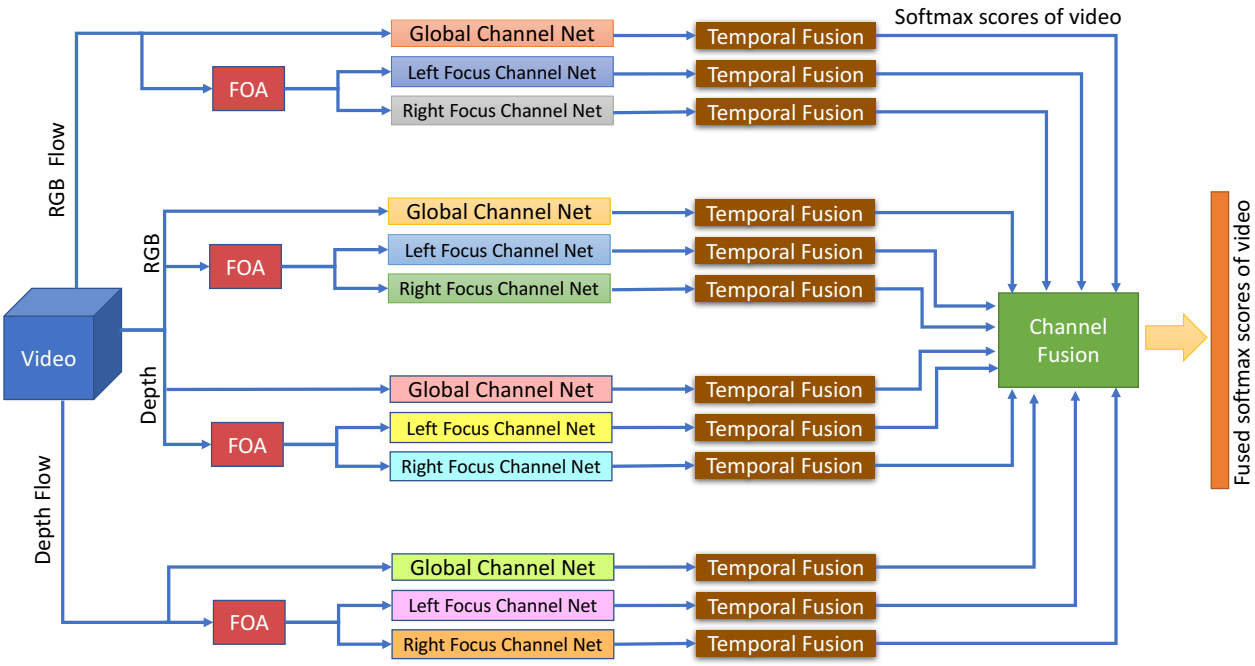

Figure 3.1: The FOANet Network Architecture instantiated for the ChaLearn IsoGD dataset. The architecture consists of a separate channel for every focus region (global, left hand, right hand) and modality (RGB, depth, RGB flow and depth flow). FOA module is used to detect hands. The temporal fusion module is used to combine the frame level predictions of the video. The video level softmax scores from 12 channels are combined by a channel fusion module. Temporal fusion module is discussed in Chapter 4 and channel fusion module is discussed in Chapter 5.

lot in this small time frame, and remains focused in the 10 frame sliding window. As the 10 frame sliding window has the advantages mentioned above, we process a sliding window of 10 frames using 2D CNNs. 2D CNNs are relatively easy to train than 3D CNNs, and can take advantage of ImageNet pretraining[4]. The information from different sliding windows is temporally fused within channels either by pooling the FC features, averaging the softmax scores, or using a RNN to encode temporal information from all sliding windows.

The multi-channel approach of FOANet divides the processing into M $\times$ L channels. The M $\times$ L channels are only useful if they can be fused back together. On one extreme, the predictions from multiple channels can be averaged together, as in [82, 123, 136]. Although this technique produces better results than using a single channel, it doesn't make the best use of the information

[4]It is very rare to train CNNs from scratch with random initialization, because it is rare to have a dataset of sufficient size. Instead, it is a common practice to pretrain CNNs on ImageNet dataset, and then use the weights learned by CNNs to initialize CNNs for other tasks.

available in the multiple channels. On the other extreme, features from multiple channels can be stacked together and a fully connected neural layer on top can be trained to do the classification. The number of weights to be learned by this method are proportional to the number of features times the number of channels. This is infeasible for large number of channels and is susceptible to overfitting even for a few channels. We propose a novel fusion method with relatively few parameters called sparse network fusion (see Section 5.1.3). This is the third idea of FOANet.

These three major ideas are discussed in three different chapters of this thesis. This chapter discusses the CNN architectures to process global and focused spatial focus of attention regions. Chapter 4 discusses different temporal fusion strategies within channels. Chapter 5 discusses strategies to fuse information from multiple channels. The reason for dividing the architecture into three chapters is to focus and optimize each piece, and to evaluate its contribution to FOANet as a whole. For this chapter, that means evaluating just the impact of the separate channels before temporal fusion. This evaluation is done both at frame level and video level (by averaging across time).

This chapter explains the CNN architecture that processes global (section 3.1) and focus (section 3.2) channels. The global and focus channels are CNNs modeled after ResNet [43], except that focused channels have additional features to record the positions of the attention windows. This is followed by experimental results on ChaLearn IsoGD (section 3.3.1) and NVIDIA (section 3.3.2) datasets. The experiments section also includes the experimental design, training process and inference process of respective datasets. Section 3.4 concludes the chapter.

3.1 Global Channel

Global channels are based on 50 layer deep residual networks [43, 44]. ResNet-50 is a high-performing network on the ImageNet challenge [16]. Although there are deeper versions of ResNet (ResNet-101, ResNet-152, ResNet-1001) and better performing architectures on ImageNet like and Inception-V3 [46], and Inception-V4 [110], ResNet-50 is selected for practical reasons, as discussed in Section 2.2.2. Unlike the original ResNet that takes a single image as input, the

input to a global channel is a stack of images. More precisely, the input is a temporal window of 10 image frames that captures local motion information. Let w and h be the width and height of the video. For an arbitrary frame t, we stack 10 consecutive frames around t (frames between [t-4,t+5]) to form a 30 channel input volume $I_g^{w \times h \times 30}$. The first 4 and last 5 frames of the video are discarded. The first layer of ResNet-50 is modified to process 30 channel input by having a convolutional layer of $7 \times 7 \times 30$ dimensions. Other than the first layer, the convolution and pooling layers of the global channel are the same as in ResNet-50, and produce a 2048 dimensional feature vector (FC) as shown in figure 3.2. Also as in ResNet-50, a fully connected layer followed by softmax produces one output per label from the 2,048 feature vector.

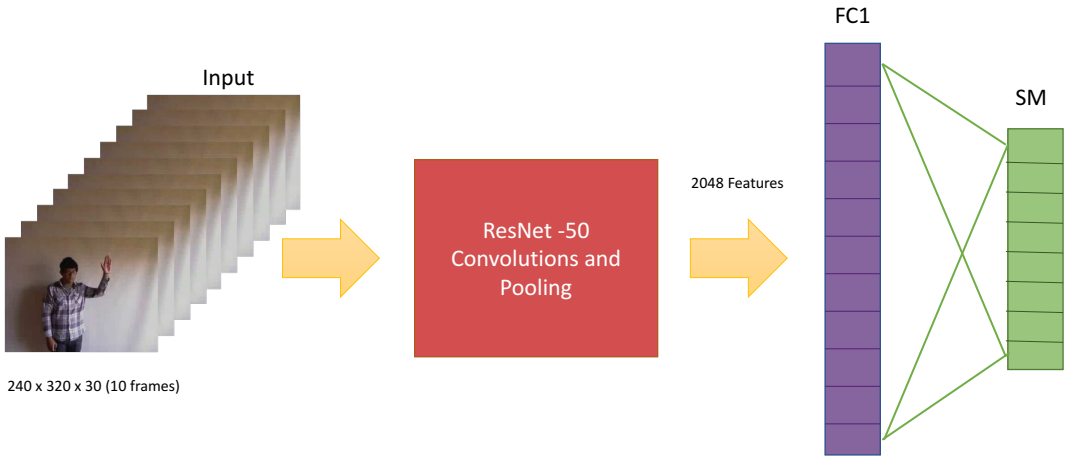

Figure 3.2: Network Architecture of Global Channels. The input to the network is a stack of 10 images resulting in a $240 \times 320 \times 30$ volume. The input volume is passed through ResNet-50 convolution and pooling layers resulting in 2048 features. A fully connected layer on top produces a vector of softmax scores.

Global channels are trained for four modalities: RGB, depth, optical flow fields from RGB and optical flow fields from depth images. The details regarding optical flow fields were previously discussed in Section 2.5. Figure 3.3 shows a window of 10 consecutive frames that forms an input to global channels. The first row of the figure shows a window from RGB modality whereas the next three row shows the same window in depth, RGB flow and depth flow modalities respectively.

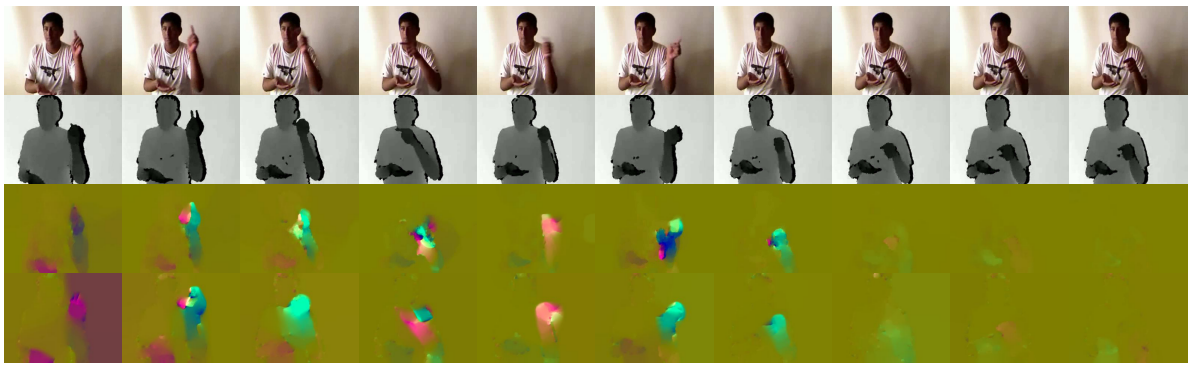

Figure 3.3: A window of 10 consecutive frames from an arbitrary video of ChaLearn IsoGD dataset. The figure shows four different modalities that global channels are trained on. The first row is the RGB modality, whereas the second row is the depth modality. RGB flow and depth flow modalities are shown in rows 3 and 4 respectively.

3.2 Focus Channel

Similar to the global channels, focus channels take a stack of images as input and use 50 layer deep residual networks [43, 44] as the network architecture. ResNet-50's first layer is modified to have $7 \times 7 \times 30$ dimensional convolutions to process 30 channel input, similar to global channels. Unlike the global channels, the input image stack is not a stack of whole image, but instead is a stack of spatial image windows focused around one of the hands. For an arbitrary frame t, let (x_1, y_1) and (x_2, y_2) be the two corners of the bounding box centered on a hand. Let $s = max(x_2 - x_1, y_2 - y_1)$ be the maximum side of the bounding box. An input volume$I_f^{s \times s \times 30}$ that is centered on the bounding box is cropped from $I_g^{w \times h \times 30}$. The cropped image stack I_f is then resized to $I_f^{128 \times 128 \times 30}$ and is given as input to the focus channels. The details about hand detection were previously discussed in Sections 2.4.1 and 2.4.2. Figure 3.4 shows an arbitrary window from ChaLearn IsoGD and its corresponding focused right and left hand windows. The bounding box is focused on the hands in 5^{th} frame. So, the entire hand is visible in the 5^{th} frame and the other frames capture the motion into and out of the 5^{th} frame bounding box. The windows focused on hands, as shown in Figure 3.4, do not capture other parts of the image that are not useful in classifying the gesture.

To tell the focus channel where the hands are, we provide 14 additions location features (7 for each hand). The location features are: (x, y) locations of top left and bottom right corners of the

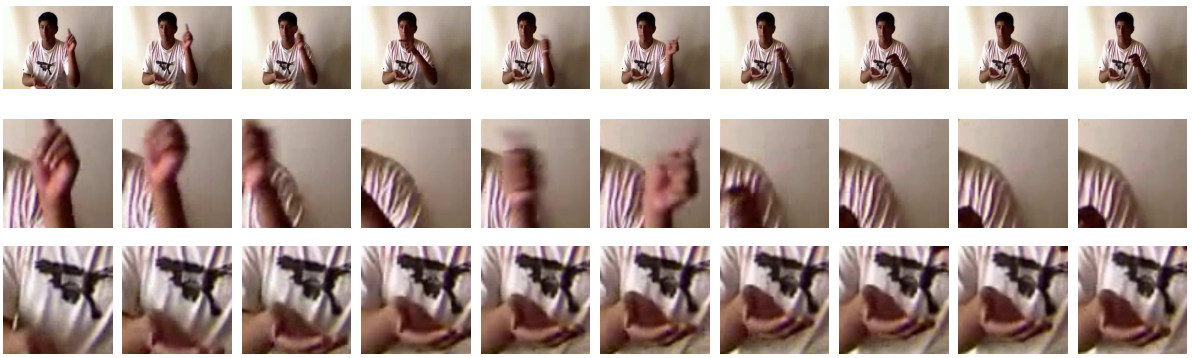

Figure 3.4: A window of 10 consecutive frames from RGB modality of Figure 3.3. The top row shows the whole window whereas the middle and bottom row shows the window focused around right and left hands respectively. The top row serves as an input to the global channels discussed in Section 3.1 whereas the middle and bottom rows serve as an input to right and left hand focus channels respectively. The hand bounding box is focused around the hand in 5^{th} column.

bounding box, width and height of the bounding box, and the ratio between the width and height. If only one hand is visible, we set the features of the other hand to zeros. These 14 features are passed through one hidden layer of 14 nodes with a $\tanh$ activation function, and the resulting 14 features are appended to ResNet-50's features as shown in Figure 3.5. The resulting feature vector (FC) is passed to a fully connected layer for classification.

A separate focus net is trained for each pair of hand and modality. For applications that involve only one hand, as in the NVIDIA data set, a single focus net is trained for each modality. Similar to global nets, focus nets are trained on four modalities: RGB, depth, optical flow from RGB images and optical flow from depth images, resulting in 4 or 8 focus nets depending on the number of hands. Figures 3.6 and 3.7 shows different modalities of the right and left hand focused windows respectively. These images correspond to the focus channels from Figure 3.4. The windows from Figures 3.3, 3.6, 3.7 together serve as input for 12 different channels of FOANet.

3.3 Results

The previous two sections (Section 3.1 and 3.2) introduced the global and focus channels to create an architecture that reflects the structure of gestures, which are combinations of large body movements and fine hand motions. This section evaluates the global and focus channels on multi-

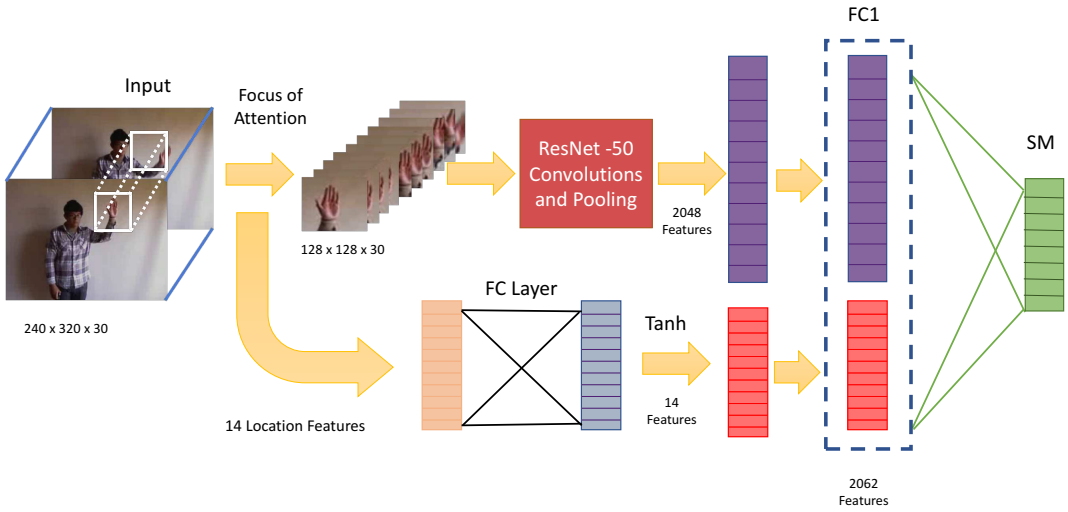

Figure 3.5: Network Architecture of Focus Channels. The input to the network is a cropped volume centered on hands. The input volume is passed through ResNet-50 convolution and pooling layers. In addition, 14 location features are passed through a fully connected layer of 14 neurons with a $\tanh$ non-linearity. These 14 features are concatenated on to the ResNet features, and a fully connected layer on top produces a vector of softmax scores.

ple modalities. We evaluate individual channels at frame level to see if they are capturing meaningful information, which is above random (random performance is $\approx$0.4% for the ChaLearn IsoGD dataset and $\approx$4% for the NVIDIA dataset). Moreover, it will give us an estimate on the relative importance of channels based on the information being captured by individual channels. We also do a correlation analysis between channels at frame level to check whether the channels are capturing different information, as there is an advantage in fusing multiple channels only if they are capturing different information.

However, frame level accuracy is not a good estimator of performance as not all frames contain information to classify a gesture. So, we also evaluate the channels at video level by temporally fusing the information from all sliding windows. Global and focus nets take a window of 10 consecutive frames as input and output a vector of softmax scores. The global and focus nets output $T - 9$ softmax vectors for a video of length T. To evaluate the channels at video level, we average the softmax scores from all windows of a video (see Chapter 4 for other temporal fusion methods). Evaluating the channels at video level tells us the relative importance of channels at video level, and allows us to compare our results with the previous state-of-the-art (which is

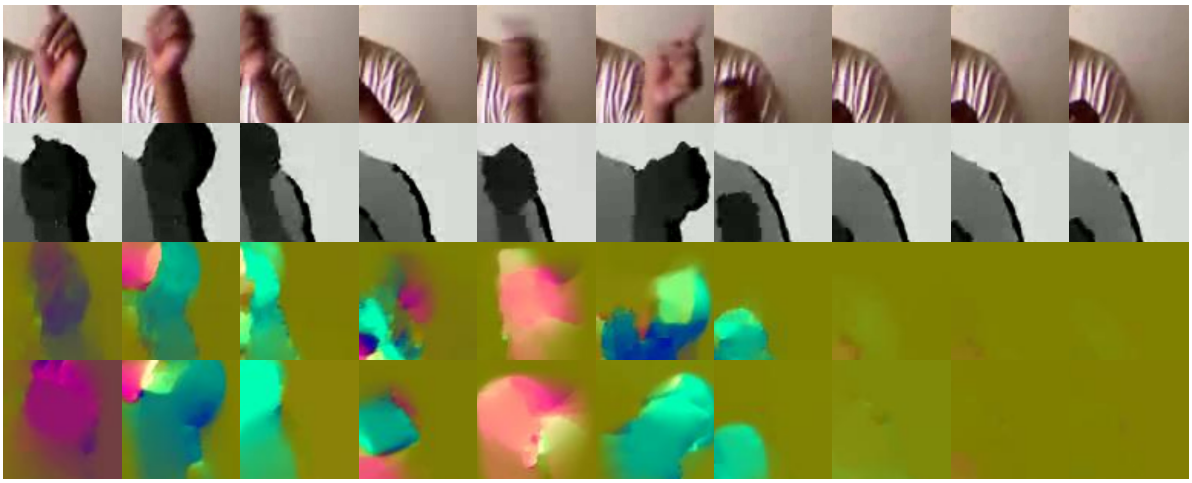

Figure 3.6: Different modalities of right hand focused window from Figure 3.4. The first row is the RGB modality, whereas the second row is the depth modality. RGB flow and depth flow modalities are shown in rows 3 and 4 respectively.

64.40% on the validation set, and 67.71% on the test set of the ChaLearn IsoGD, and 83.8% on the NVIDIA dataset).

Section 3.3.1 presents the experimental results on ChaLearn IsoGD dataset and Section 3.3.2 presents the experimental results on NVIDIA dataset. In the following sections, we evaluate the accuracies of different channels both at frame level and video level, and we do a correlation analysis between channels at frame level. These sections also include experimental design, training and inference process for the respective datasets.

3.3.1 ChaLearn IsoGD Experiments

Experimental Design

The 2017 ChaLearn IsoGD challenge asked participants to classify videos as one of 249 gestures [118]. Participants were given access to a set of 35,878 labeled training videos, and a second set of 5,784 labeled validation videos. Participants were encouraged to develop the best system they could, training on the training videos and testing on the validation videos. At the conclusion of the challenge, participants were given access to a previously sequestered set of 6,271 labeled test videos. They were asked to evaluate their system on the test videos without modification.

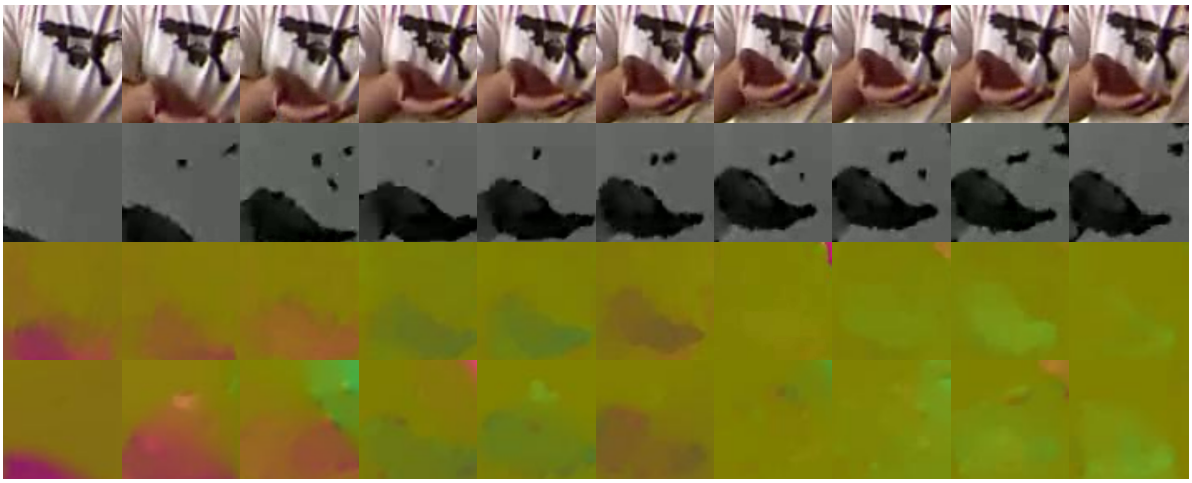

Figure 3.7: Different modalities of left hand focused window from Figure 3.4. The first row is the RGB modality, whereas the second row is the depth modality. RGB flow and depth flow modalities are shown in rows 3 and 4 respectively.

As already discussed in Section 2.4.1, we mimicked the experimental design of ChaLearn IsoGD challenge as closely as possible. Our system was incrementally developed by training on the training videos and testing on the validation videos. Test videos were internally sequestered and were used only to evaluate our system.

Participants in the challenge generally report two sets of numbers: performance on the validation data, and performance on the test data. We do the same in the following sections.

Training Process

The convolutional nets inside the global and focused channels are trained using various forms of "warm starts". The convolutional nets in global channels are fine-tuned from ResNet-50 pretrained on ImageNet [16]. The pretrained ResNet-50 takes 3 channel images as input, whereas our global channel nets takes 30 channels as input (a 10 image stack with 3 bands per image). To account for this, the first pretrained convolutional layer weights $(7 \times 7 \times 3 \times 64)$ are repeated 10 times and stacked together $(7 \times 7 \times 30 \times 64)$. The last fully connected layer weights are randomly initialized and the nets are trained end to end using mini-batch stochastic gradient descent with momentum (set to 0.9) and a random batch of size 64. The input volume is randomly cropped to a $224 \times 224 \times 30$ volume and random flipping is performed for data augmentation. Color aug-

mentation is not performed to keep the training methodology the same on all channels, since color augmentation can't be performed on flow fields or depth images. The learning rate lr is initially set to 0.0002 and is decayed exponentially with a decay factor df of 0.7 and decay steps ds of 40,000. The decayed learning rate dlr at a step is calculated as $dlr = lr * df^{\frac{step}{ds}}$.

The global channel convnets took 9 days to fine-tune on the ChaLearn dataset using a single Titan X GPU. We used the fine-tuned global channel as a warm start for the respective focus channels. For example, the RGB left hand and RGB right hand focus channels are trained by fine-tuning the RGB global channel. The convolution weights for focus channels are initialized from the pretrained global channels and the fully connected layer weights (location and last fully connected layers) are randomly initialized. The input volume is randomly cropped to a $100 \times 100 \times 30$ volume and random flipping is performed by flipping left hand and using it to train right hand nets and vice versa. Similar to global channel nets, focus channel nets are also trained end to end using mini-batch stochastic gradient descent with the same momentum term, batch size and learning rate rules. The focus channel convnets took 1 day to train when fine-tuned global channel weights are used as warm starts.

Inference Process

During inference, data is passed through the convolutional networks without augmentation (cropping or flipping). For global channels, the input volume is $240 \times 320 \times 30$; for focus channels, the input volume is $128 \times 128 \times 30$. The output of CNNs is a softmax vector of length 249. The argmax of the softmax vector is the predicted gesture.

Analysis of Channels at Frame Level

Table 3.1 shows the frame level accuracy of various channels on ChaLearn IsoGD validation and test sets. The shaded row represent average accuracy of global and focus channels across different modalities whereas the shaded column represents average accuracy of modalities. By comparing the results in Table 3.1, we can say that RGB flow and depth are the two best modalities on ChaLearn IsoGD dataset. However, there is no clear winner between these two modalities.

Table 3.1: Frame level accuracies (percent correct) on all channels of ChaLearn IsoGD validation and test set. Rows represent different modalities whereas columns represent global and focus channels (right and left hand). The shaded column represents the average frame accuracy of different modalities whereas shaded row represents average accuracy of global and focus channels. The global channel accuracy is calculated on all frames of the validation and test set whereas focus channel accuracy is calculated on the frames where the respective hand is visible.

	Validation Set				**Test Set**			
	Global	**Left**	**Right**	**Average**	**Global**	**Left**	**Right**	**Average**
RGB	24.00	14.68	22.09	21.14	27.95	11.29	23.21	21.40
Depth	22.58	24.25	31.56	26.34	28.02	16.25	34.27	26.82
RGB Flow	32.32	22.79	28.29	28.58	32.53	15.27	27.47	25.63
Depth Flow	22.74	20.92	25.06	23.19	28.02	13.80	28.08	23.87
Average	25.40	20.66	26.75		29.12	14.15	28.26	

RGB flow modality performs best on validation set with average frame level accuracy of 28.58%, whereas depth modality is the second best with an average frame level accuracy of 26.34%. However on test set, depth modality performs best with average accuracy of 26.82% followed by RGB modality with an average accuracy of 25.63%. RGB modality has the least accuracy on both validation and test set, possibly because it is susceptible to illumination and background variation. Unlike RGB, the depth and RGB flow modalities don't capture background and are unaffected by illumination, so they are the top two modalities on ChaLearn IsoGD. As optical flow is not designed for depth, depth flow doesn't perform on par with RGB flow and comes in third.

Among global and focus channels, the right hand is the best channel on validation set with an average accuracy of 26.75% followed by global channel with an average accuracy of 25.40%. On the test set, however, the global channel performs best with an average accuracy of 29.12% followed by right channel with an average accuracy of 28.26%. As the ChaLearn IsoGD dataset is heavily biased with right hand participants, the left hand channel comes last among global and focus channels. Moreover, on the validation set the difference between the left hand channel performance and the other channels' performance is nearly $5 - 6\%$, whereas on test set the difference is around $14 - 15\%$. Based on this observation, we can deduce that the test set has higher right hand bias than the validation set.

By looking at individual cells of Table 3.1, the RGB flow modality on the global channel and the depth modality on the right hand are the top two performing channels. Similar to the pattern observed in modalities and global/focus channels, there is no clear winner between these two channels. "Global RGB flow" channel is better on validation set whereas "depth right" channel is better on test set. "RGB left" channel is the least performing channel on both validation and test sets.

Correlation of Frame Level Predictions among Channels

We do a correlation analysis between channels to see if the channels are predicting the same frames correctly/incorrectly. To perform correlation analysis, a boolean vector for a video is obtained by taking the argmax of the softmax vector for each sliding window and comparing it with the ground truth gesture class. For each channel and video in the ChaLearn IsoGD dataset, similar boolean vectors are obtained. These vectors are stacked together within channels, resulting in 12 long boolean vectors, one for each channel. Correlation analysis is done between 12 channels by calculating correlation between pairs of channels. As both the left and right hands are not visible in all frames, the correlation between different spatial channels is done only on overlapping frames. At the beginning and end of the video, the subjects will be in neutral poses, and all channels tend to do incorrect predictions. Although the channels might be highly correlated during neutral pose, the overall effect on the dataset as a whole is not significant, as neutral poses tend to occur for relatively short periods.

Correlation values between different channels on the validation set of the ChaLearn IsoGD dataset are shown in Figure 3.8. The correlation values on the validation set range between [0.14, 0.44]. The channels are not highly correlated to each other and there is no negative correlation between channels. By looking at the correlation values in Figure 3.8, clear patterns emerge. Global channels are more correlated to other global channels, followed by right hand, and left hand channels. Right hand channels are more correlated to other right hand channels, followed by global, and left hand channels. Left hand channels are more correlated to other left hand channels, followed by right hand, and global channels. As all global channels process the whole image, they

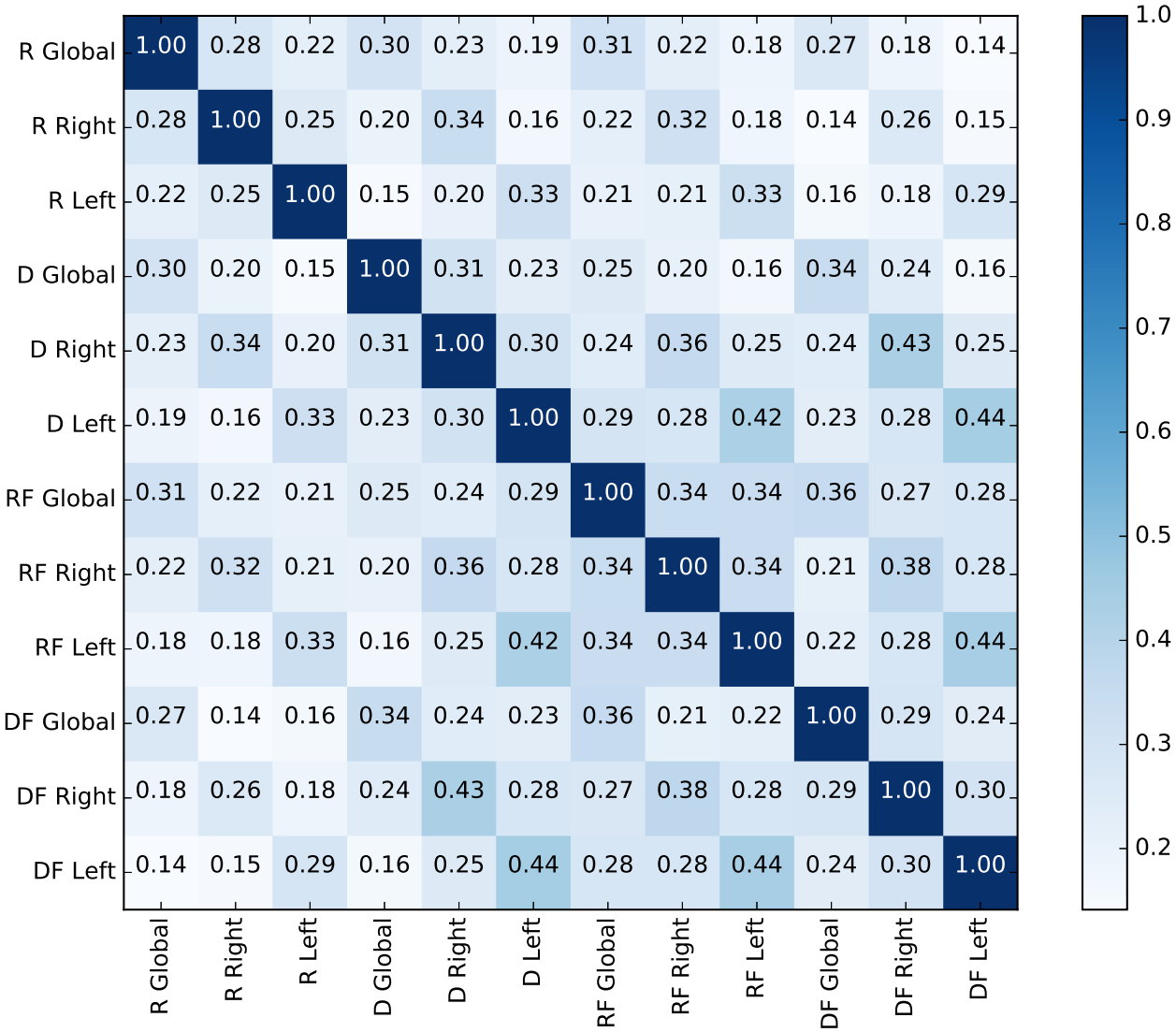

Figure 3.8: Frame level correlation among different channels on validation set of ChaLearn IsoGD dataset.

tend to correlate more with each other compared to other focus channels that process only hands. Global channels are more correlated to right hand channels than left hand channels as the left hand is generally only used for two handed gestures. Similarly, right hand channels are more correlated to right hand channels as all right hand channels process right hand data. Right hand channels tend to correlate more to global channels than left hand channels due to the inherent right hand bias in the dataset. Interestingly. left hand channels correlate more to right hand channels than global channels. As already mentioned, the left hand is used only to perform two-handed gestures. When performing two-handed gestures, the right hand channels has as much information as the left hand

channels to classify the gesture. So, left hand and right hand channels tend to correlate more on two-handed gestures. Therefore, left hand channels correlate more to right hand channels than global channels.

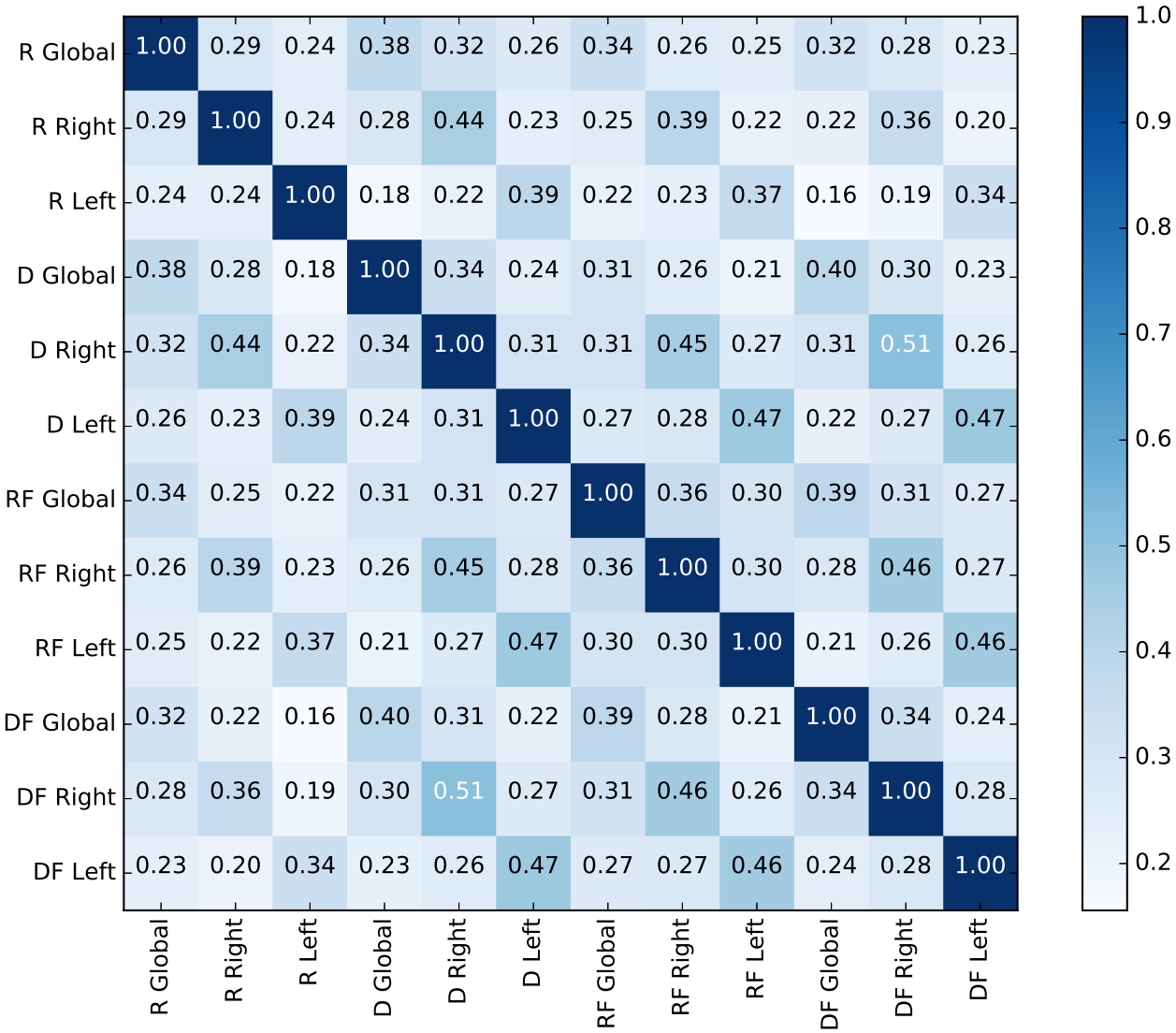

Figure 3.9: Frame level correlation among different channels on test set of ChaLearn IsoGD dataset.

The highest correlations on the validation set is between "depth left" and "depth flow left" channels, and "RGB flow left" and "depth flow left" channels. As left channels tend to be idle most of the time and a gesture can't be predicted by processing idle hands, these channels might be highly correlated due to incorrectly predicting the gesture most of the time. When looked at the

average correlation of the channels, "depth left" channel has the highest average correlation and "RGB right" has the lowest average correlation.

The correlation values between different channels on the test set of the ChaLearn IsoGD dataset is shown in Figure 3.9. The correlation values on the test set range between [0.16, 0.51], which is higher than the validation set. The correlation patterns of global and focus channels evident in the validation set are also evident here. Global correlate more with other global channels; right hand channels correlate more with other right hand channels; left hand channels correlate more with other left hand channels. On test set, "depth right" and "depth flow right" channels have highest correlation with a correlation score of 0.51. "Depth right" channel has highest average correlation to other channels and "RGB left" has the lowest average correlation to other channels. This pattern is also evident in the frame level accuracies of test set as shown in Table 3.1. On test set, "Depth right" channel has the highest frame level accuracy (34.27%), and "RGB left" has the lowest frame level accuracy (11.29%).

In conclusion, the correlation values between different channels of ChaLearn IsoGD dataset are low, which indicates that the channels are capturing different information. So, there is an advantage in fusing information from all channels. Moreover, there is a clear pattern of global channels being relatively highly correlated with other global channels, and focus channels being relatively highly correlated with respective focus channels.

Analysis of Channels at Video Level

The previous two sections analyzed the 12 channels of ChaLearn IsoGD dataset at frame level. However, frame level accuracy is not a good estimator of performance as not all frames contain information to classify a gesture. Gestures usually consists of three phases: preparation, nucleus, and retraction [82]. Nucleus contains the information to classify a gesture whereas preparation and retraction can look similar for different gestures. In addition, hands can be in idle pose. As the frame level accuracy takes into account the predictions from preparation, retraction phases, and idle poses, it is not a good estimator of performance. So, we analyze the performance of channels at video level in this section.

Table 3.2: Individual channel accuracies on ChaLearn IsoGD validation and test set at video level. The softmax scores from all the sliding videos of a video are averaged together and the argmax of average softmax vector is used as the gesture class prediction for a video. The numbers represent the accuracies on all videos of validation and test set. However, not all videos have both hands visible. The accuracies in brackets shows the accuracies on the videos where the particular hand is visible.

	Validation Set			**Test Set**		
	Global	**Left**	**Right**	**Global**	**Left**	**Right**
RGB	33.22	16.17 (23.41)	41.60 (41.76)	41.27	16.63 (19.55)	47.41 (47.44)
Depth	27.98	23.76 (34.40)	54.91 (55.12)	38.50	24.06 (28.29)	64.44 (64.48)
RGB Flow	46.22	24.14 (34.95)	54.60 (54.81)	50.96	24.02 (28.23)	59.69 (59.73)
Depth Flow	31.66	21.84 (31.62)	48.32 (48.51)	42.02	22.71 (26.70)	58.79 (58.83)

To analyze the channels at video level, the information from sliding windows should be fused temporally across a video. Chapter 4 discusses different methods to temporally fuse the information. However, we use a simple temporal fusion strategy of averaging the softmax scores across time in this section.

Table 3.2 shows the video level accuracy of each channel on the IsoGD validation and test sets. Unfortunately, the left and right hands are not visible in all videos. Right hands are visible in 5,762 of 5,784 validation videos and 6,267 of 6,271 test videos, or in about 99% of the videos. In contrast, left hands are only visible in 3,994 of 5,784 validation videos and 5,334 of 6,271 test videos, or about 77% of the videos. The numbers in brackets in Table 3.2 refer to the classification accuracies of focused channels when limited to videos in which the corresponding hand is visible.

The classification accuracies in Table 3.2 range from 16.17% to 64.44%. Although the classification accuracy of all channels is above random (which is 0.4%), none of the channels outperformed the previous state-of-the-art results on the ChaLearn IsoGD dataset (which is 64.40% on the validation set, and 67.71% on the test set of the ChaLearn IsoGD). However, the previous state-of-the-art system achieved the classification accuracy of 67.71% by combining multiple channels (RGB, depth, flow) [77].

Comparing the columns in Table 3.2, a clear pattern emerges: Right outperforms Global and Global outperforms Left in all eight cases. Presuming performance is a guide to where the most useful information resides, the most useful information is in the right hand. This outcome may not

be surprising since the dataset is composed primarily of right handed participants, and participants tend to use their left hand only for gestures involving both hands. So, even when the left hand is visible, it is often idle. However, overall performance is best when all channels are combined (see Section 5.2.1), suggesting that the left hand is important for the subset of two-handed gestures.

When we compare the rows in Table 3.2, the contributions of the different data modalities are more complex. Global channels perform best when they process flow fields extracted from RGB data. This is consistent with the idea that global channels are looking for gross movements. Right hand channels perform best on depth data, suggesting that many of them may be poses rather than motions, although they also perform well on RGB flow fields. Left hand channels perform roughly the same on depth and RGB flow field data. We also note that flow fields extracted from depth data don't perform on par with flow fields extracted from RGB data. This may be attributable to the fact that flow field extraction algorithms are designed for RGB images, not depth images, and suggests a research opening for better flow field from depth algorithms.

3.3.2 NVIDIA Experiments

Experimental Design

Recently, NVIDIA published a dataset of 25 gesture types intended for touchless interfaces in cars as discussed in Section 2.4.2. The dataset consists of 1,532 instances of dynamic hand gestures performed by 20 subjects. RGB, depth and a pair of stereo-IR streams are provided for each hand gesture, although we use only the RGB and depth streams. The data is split by subject into 1,050 training and 482 test videos. As a validation set is not provided with the dataset, we choose 1 subject from the training set to be the validation set. We follow the same experimental design as in ChaLearn by incrementally developing our system by training on the training videos and testing on the validation videos. We evaluated the system only once and without modification on the test videos.

Training and Inference

Figure 3.10 shows the FOANet architecture with 8 channels instantiated for the NVIDIA dataset. Unlike the ChaLearn IsoGD dataset, that has both hands visible, all gesture in NVIDIA dataset are performed by right hand only. So, there are only four focus (right hand) channels for the NVIDIA dataset.

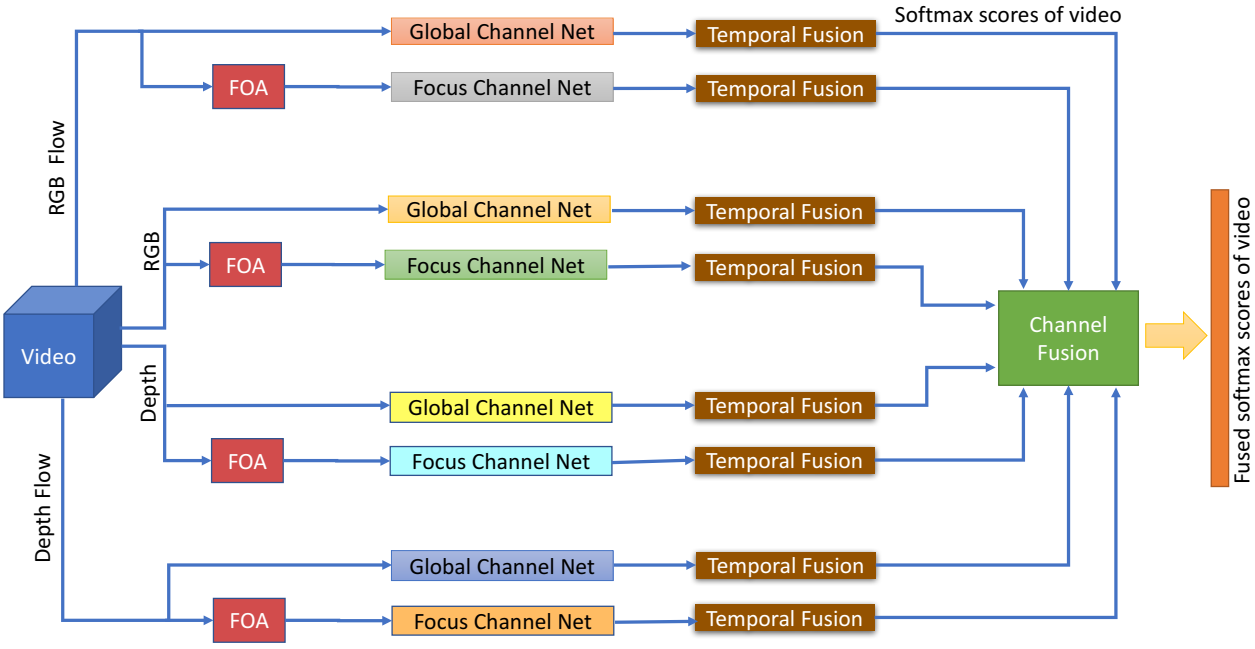

Figure 3.10: The FOANet Network Architecture instantiated for the NVIDIA dataset. The architecture consists of a separate channel for every focus region (global, right hand) and modality (RGB, depth, RGB flow and depth flow). As only right hand is visible in this dataset, there is only one focus channel per modality. FOA module is used to detect hands. The temporal fusion module is used to combine the frame level predictions of the video. The video level softmax scores from 8 channels are combined by a channel fusion module. Temporal fusion module is discussed in Chapter 4 and channel fusion module is discussed in Chapter 5.

The CNNs for the NVIDIA dataset are trained in a similar way to the CNNs for the ChaLearn dataset (See Section 3.3.1), except for the following three differences: 1) the CNNs are fine-tuned from the respective channel nets trained on ChaLearn IsoGD; 2) flipping is not used to augment the training set, as all gestures are performed with the right hand only; and 3) only right hand focus channels are trained, since the left hand is never visible. The inference process is similar to the process for ChaLearn as discussed in Section 3.3.1.

Table 3.3: Frame level accuracies on all channels of NVIDIA test set. Rows represent global and focus channels whereas columns represent different modalities. The shaded row represents the average frame accuracy of different modalities whereas shaded column represents average accuracy of global and focus channels.

	RGB	**Depth**	**RGB Flow**	**Depth Flow**	**Average**
Global	27.23	43.49	42.40	28.46	35.39
Focus	34.71	46.11	39.60	35.56	38.99
Average	30.97	44.8	41.00	32.01	

Analysis of Channels at Frame Level

Table 3.3 shows the frame level accuracy on the test set of NVIDIA dataset. As the left hand is never visible in the dataset, there is only one focus channel. The shaded column represents the average accuracy of global and focus channels across different modalities whereas the shaded row represents the average accuracy of modalities.

By comparing the average accuracy of different modalities, we can say that depth is the best modality with an average frame level accuracy of 44.8%, followed by RGB flow (41.00%), depth flow (32.01%) and RGB (30.97%) modalities. This pattern is the same as the frame level accuracies of test set of ChaLearn IsoGD dataset, as discussed in Section 3.3.1. Among global and focus channels, focus channels outperform global channels with an average accuracy of 38.99% compared to the average accuracy of 35.39% on global channels. This pattern of focus channels performing better than global channels was also evident on validation set of ChaLearn IsoGD dataset (see Section 3.3.1).

By looking at the individual accuracies, we can see that "depth focus" channel is the best performing channel with an accuracy of 46.11%. This is followed by "depth global" channel with an accuracy of 43.49%, and "RGB flow global" channel with an accuracy of 42.40%. Interestingly, "RGB flow global" and "depth focus" are the top two performing channels on ChaLearn IsoGD dataset as well. The "RGB global" channel has the least accuracy (27.23%) among all channels.

Correlation of Frame Level Predictions among Channels

Similar to the ChaLearn IsoGD dataset, we do correlation analysis between pairs of different channels of NVIDIA dataset. The correlation values between different channels is shown in Figure 3.11. The correlation values on NVIDIA dataset range from [0.12, 0.52]. The channels are not highly correlated to each other. Interestingly, the pattern of global channels being highly correlated to other global channels and focus channels being highly correlated to other focus channels that was evident in ChaLearn IsoGD dataset is not evident here.

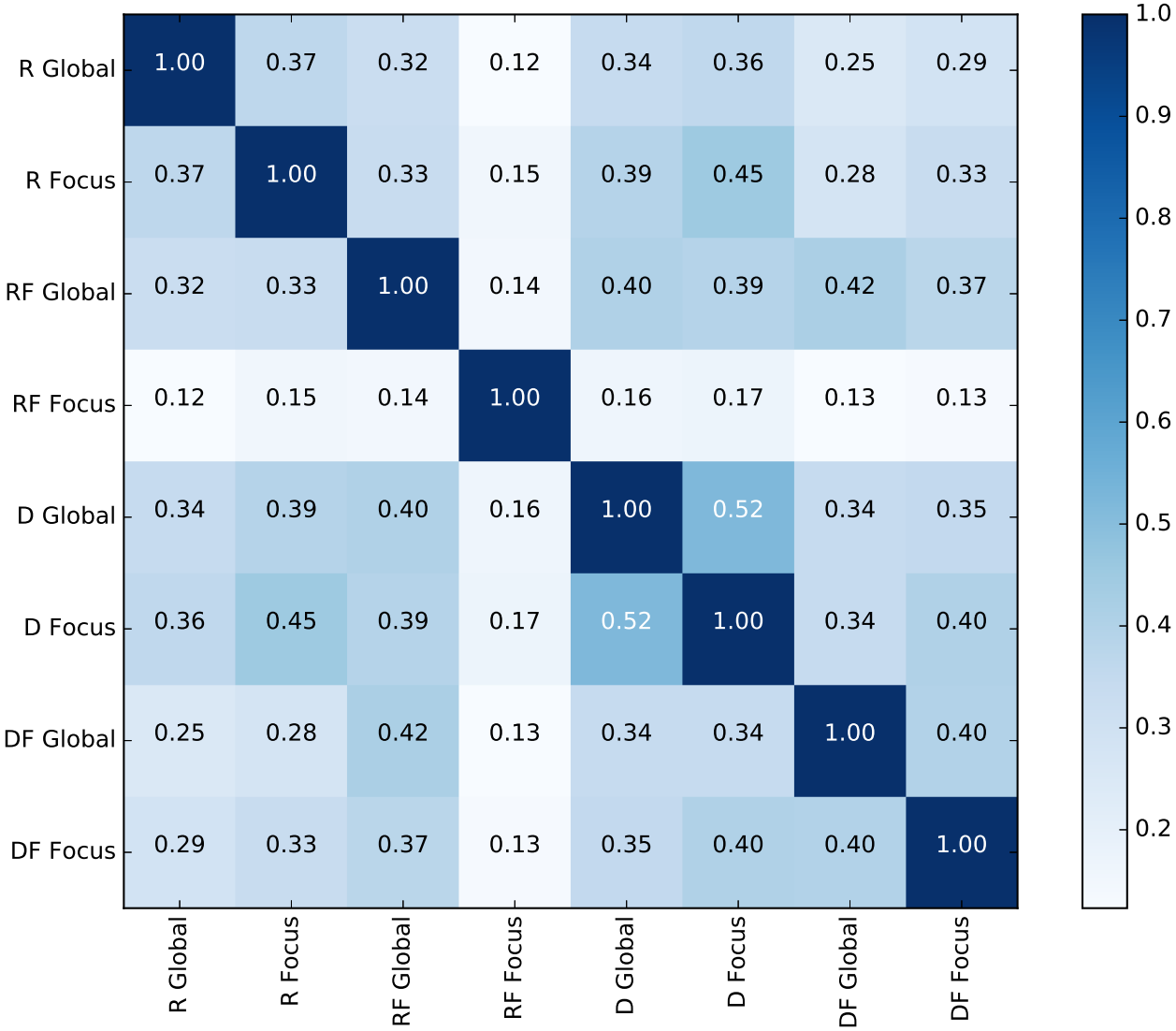

Figure 3.11: Frame level correlation among different channels on NVIDIA dataset.

Table 3.4: Video level accuracies of individual channels on NVIDIA test set. The softmax scores from all the sliding videos of a video are averaged together and the argmax of average softmax vector gives the gesture class prediction for a video.

	RGB	**Depth**	**RGB Flow**	**Depth Flow**
Global	43.98	66.80	62.66	58.71
Focus	58.09	70.12	77.18	73.65

When the correlations values of a channel to all other channels are averaged together, "depth focus" channel has the highest average correlation and "RGB flow focus" channel has the lowest average correlation. Looking back at Table 3.3, "depth focus" channel has the highest frame level accuracy of 46.11%. However, "RGB flow focus" channel doesn't have the lowest frame level accuracy. Interestingly "RGB flow focus" has the highest video level accuracy as seen in Table 3.4.

Analysis of Channels at Video Level

As the frame level accuracy is not a good estimator of performance, we analyze the channels at video level, by averaging the softmax scores across time. Table 3.4 shows the video level accuracies of various channels on NVIDIA dataset.

The classification accuracies in Table 3.4 range from 43.98% to 77.18%. Although the classification accuracy of all channels is above random (which is 4%), none of the channels individually outperformed the previous state-of-the-art results of 83.8% on the NVIDIA dataset. However, the previous state-of-the-art system achieved the classification accuracy of 83.8% by combining multiple channels (RGB, depth, flow, IR) [82].

Comparing the rows in Table 3.4, a clear pattern emerges: Focus channels always outperform global channels on a given modality. This outcome is consistent with the observations on ChaLearn IsoGD dataset, stressing the importance of focus channels.

When we compare the columns in Table 3.4, the contributions of the different data modalities are more complex. Global channels perform best when they process depth modality, followed by flow modalities. In contrast, global channels performed best on flow modalities on ChaLearn IsoGD dataset. This may be attributable to the fact that the depth modality in NVIDIA dataset is clean without any background, unlike ChaLearn IsoGD dataset that has complex backgrounds

visible even in depth. Focus channels perform best on flow modalities, followed by depth, and RGB modalities. Presuming performance is a guide to where the most useful information resides, in the absence of background, RGB flow has the most useful information. This result is not surprising as optical flow captures motion information that is useful for distinguishing gestures, and optical flow algorithms are specifically designed for RGB images.

3.4 Summary

This chapter proposed a multi-channel approach to gesture recognition based on the spatial focus of attention. In addition to different data modalities, the proposed multi-channel approach also includes 3 spatial attention regions for each modality: one for the whole scene (global channel), and one each for the hands (focus channels). This architecture reflects the structure of gestures, which are combinations of large body movements and fine hand motions. The architecture of global channels and focus channels was discussed in Section 3.1 and 3.2 respectively. Experimental results in Sections 3.3.1 and 3.3.2 evaluated different channels on the ChaLearn IsoGD and NVIDIA datasets respectively.

The experimental results on the ChaLearn IsoGD and NVIDIA datasets shows that global and focus channels are different, and are capturing different information. Global and focus channels have low correlations across all modalities. Also, individual spatial attention channels have low correlations with themselves across different modalities. As spatial focus of attention channels and modalities are capturing different information, there is an advantage in our FOANet architecture of dividing the processing into spatial attention channels $\times$ modalities, and finally combining the information from them. Chapter 5 discusses various methods to fuse information from multiple channels and we will see that the overall performance is best when the information from all channels is combined.

As the frame level accuracies are not an actual testament of the performance of an algorithm as it is not practical to classify the whole video based on a window of 10 frames, we evaluated the channels at video level by averaging the softmax scores from all windows of a video. At the

video level, all channels (combinations of modality cross spatial attention) perform way above random. However, none of the channels outperform the previous state-of-the-art individually. In the ChaLearn IsoGD dataset, left channels doesn't perform on par with right hand channels and global channels, as the dataset is heavily biased by right hand participants and left hand is used only to perform two-handed gestures.

Moreover, focus channels are better than global channels, when spatial location channels are compared. When data modalities are compared, the contribution of different data modalities is more complex. In the presence of complex background (ChaLearn IsoGD dataset), global channels perform best on flow modalities, and focus channels perform best on depth data. However, in the absence of complex backgrounds (NVIDIA dataset), global channels perform best on depth data and focus channels perform best on flow modalities.

Global and focus nets discussed in this chapter work on a window of 10 consecutive frames. The information from these windows should be temporally fused to make predictions at video level. This chapter presented results on averaging - the simplest temporal fusion strategy, and we noticed that even the simplest temporal fusion strategy increased the performance significantly. So, the next chapter discusses different methods to temporally fuse information in a video and analyzes their performance.

Chapter 4

Temporal Fusion

The previous chapter introduced the multi-channel approach of FOANet to divide processing into L spatial attention regions $\times$ M modalities. These channels are evaluated at frame level and video level on two isolated gesture recognition datasets: ChaLearn IsoGD [119] and NVIDIA [82]. Isolated gesture recognition is a video classification problem. Every video contains one and only one gesture, and the goal is to assign the correct gesture label to every video. The CNNs in each channel classify every frame[5] as belonging to one of N gestures. These labels must be fused together temporally to assign a label to the video. As isolated gesture recognition datasets have a single gesture per video, simple temporal fusion strategies like averaging can be used to temporally fuse information as discussed in the previous chapter.

Isolated gesture recognition helps to develop better systems for gesture recognition. However, isolated gesture recognition systems can't be directly employed in the real world, as real world systems receive continuous streams of unsegmented data. Continuous gesture recognition is a harder video labeling task. In continuous gesture recognition, videos contain sequences of gestures, and the goal is to assign a label to every frame. Figure 4.1 shows the distinction between isolated and continuous gesture recognition.

Continuous gesture recognition poses an additional challenge in the time domain, as gestures should be detected and classified simultaneously. The simplest approach is to pre-segment the gestures from video and apply isolated gesture recognition methods [69, 118, 124]. Although pre-segmentation introduces a new source of error and a significant time lag, it is the dominant paradigm in continuous gesture recognition challenges. Unfortunately, gesture recognition methods that rely on pre-segmentation can't be directly adopted to real-time gesture recognition.

[5] Actually, every frame is taken as the center of a 10 frame temporal window of frames, and the window of frames is classified.

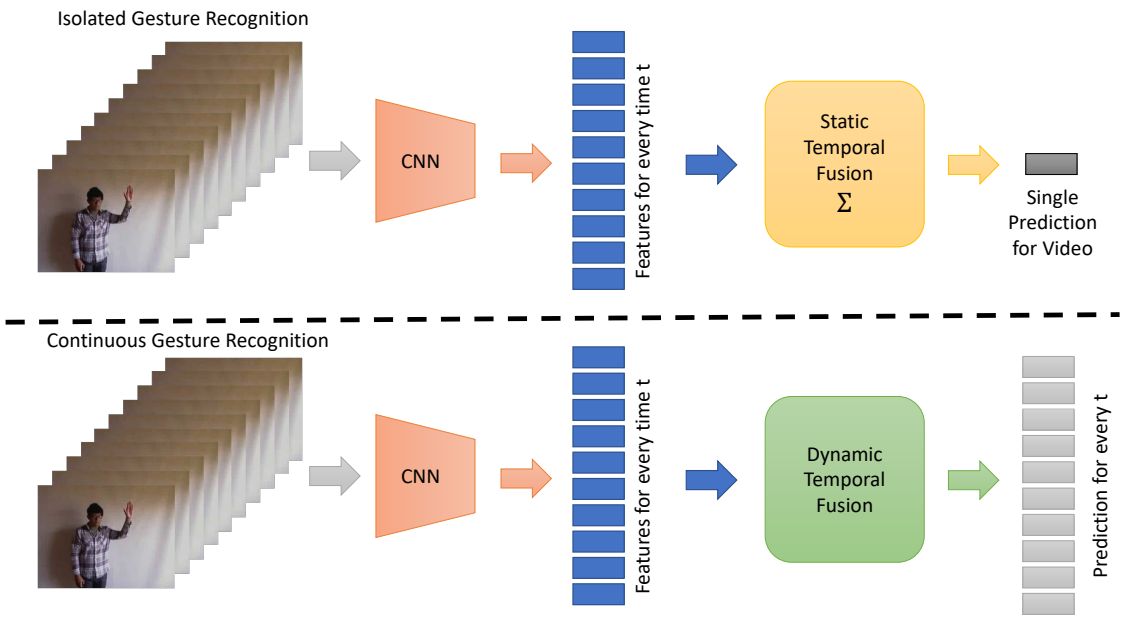

Figure 4.1: Difference between isolated and continuous gesture recognition. Isolated gesture recognition is a video classification problem where a video contains a single gesture. A simple temporal fusion strategy like averaging would suffice for isolated gesture recognition. Continuous gesture recognition is a video labeling problem where a video contains a sequence of gestures, and the goal is to assign a label to every frame of the video. Continuous gesture recognition requires more sophisticated temporal fusion methods.

Unlike approaches that temporally normalize the video to a fixed duration [80, 81], FOANet processes a sliding window of 10 frames, making the FOANet architecture suitable for continuous gesture recognition from streaming data. However, 10 frames is a very small temporal footprint ($\frac{1}{3}rd$ of a second for gestures captured at 30 FPS). The information from overlapping sliding windows must be temporally fused so as to permit transitions between gestures while generating a label for every frame.

This chapter introduces three dynamic temporal fusion methods to temporally fuse information within channels for continuous gesture recognition: *Late Pooling, Gaussian Pooling and LSTMs*. *Late Pooling* temporally fuses the information at softmax level. *Gaussian Pooling* temporally fuses the information one layer earlier than *Late Pooling*, at FC feature level. LSTMs temporally encodes information from both FC features and softmax scores.

When experimenting with these three temporal fusion strategies for continuous gesture recognition, we noticed a striking pattern. Different channels performed best using different temporal

fusion strategies, and the best fusion strategy is determined by the target (global vs. left/right hand) and modality (RGB, depth or flow field) of the channel. Using terms that will be defined in Section 4.2.1, global channels perform best with *Gaussian Pooling*, flow field channels focused on the left or right hand perform best with *LSTMs*, and RGB and depth channels focused on the left or right hand perform best with *Late Fusion*. Sections 4.2.2 and 4.2.3 shows the data supporting this surprising conclusion, while Section 4.2.2 conjectures about its cause.

The interplay between channel type and the optimal temporal fusion methods for continuous gesture recognition was so striking that we wanted to see whether it also held for isolated gesture recognition. We experimented with temporal fusion strategies for isolated gesture recognition that are analogous to the dynamic temporal fusion methods for continuous gesture recognition in Section 4.3. However, the pattern did not hold for isolated gesture recognition task. We hypothesize that segmentation removes the need for intelligent temporal information fusion.

The rest of the chapter is organized as follows. Section 4.1 discusses the model selection criteria for different temporal fusion methods. Section 4.2 discusses temporal fusion strategies for continuous gesture recognition. Section 4.3 concentrates on temporal fusion methods for isolated gesture recognition. Section 4.4 summarizes and concludes the chapter.

4.1 Model Selection

At the heart of every FOANet channel is a convolutional neural network trained using standard deep learning algorithms and techniques. One of these techniques is model selection. CNNs are trained iteratively. One typically observes that the training accuracy increases steadily over time, while the validation accuracy plateaus and then eventually decreases. The standard way to select the best model is to choose the model where validation accuracy is maximum [35].

However, the model that maximizes the validation accuracy is not the best model for all temporal fusion strategies. In pilot studies, we observe that the maximum validation accuracy model is best for temporal fusion strategies that don't involve any additional training (e.g., averaging or late pooling), but not for methods that involve additional training (e.g., LSTM, max pooling, Gaus-

sian pooling). Our experiments suggest that earlier models selected when the validation accuracy first starts to plateau overfit less and are better for temporal fusion methods that involve additional training. This model can be found by fitting a line to the last $\frac{1}{4}$th of validation accuracies by using least squares approximation and taking the first model to cross this line, as shown in Figure 4.2.

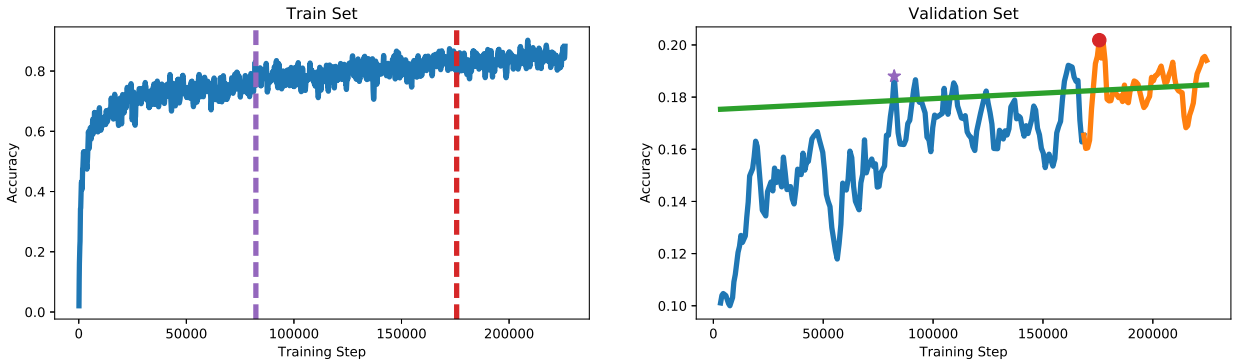

Figure 4.2: Model selection for temporal fusion. The left frame shows shows training accuracy as function of training steps for one CNN, while the right frame shows validation accuracy as a function of training steps. Temporal fusion methods that don't need any additional training are optimized by selecting the model that maximizes validation accuracy, shown as a red dot in the right frame. On the other hand, methods that involve additional training are optimized by less overfit models produced earlier in the training. For these techniques, we fit a line (shown in green) to the last quarter of the validation accuracy points (shown in orange). The first model that crosses the green line is selected.

Figure 4.2 shows the frame level accuracy on training and validation datasets as the training progresses for one channel of ChaLearn ConGD dataset. The standard way to select the best model is to chose the model with maximum validation accuracy (red circle in Figure 4.2). This model is indeed the best model for temporal fusion methods that don't require additional training, but not for methods that require additional training. We can observe that when the training accuracy of this model is highest (red vertical line in Figure 4.2), the features are overfit and temporal fusion methods trained on these features don't perform well. It is better to select a model from an earlier stage of training and this model can be picked by fitting a line (green line in Figure 4.2) on the last $\frac{1}{4}$th of validation accuracies (orange curve in Figure 4.2). The first model that crosses this line (purple star in Figure 4.2) is selected for training temporal fusion methods. We can see that the training accuracy at this model (purple vertical line) is nearly 10% less (less overfitting) than the

one at maximum validation accuracy. So, this model is best suited for training temporal fusion strategies. Pilot experiments show that the model selection strategy discussed in this section improves the individual channel accuracies on validation sets of multiple datasets (ChaLearn IsoGD, NVIDIA, ChaLearn ConGD) by 4% on an average.

4.2 Temporal Fusion Methods for Continuous Gesture Recognition

Continuous gesture recognition videos contain multiple unsegmented gestures and the task is to assign a gesture label for every frame. FOANet channels process sliding windows of 10 frames and classify each window. However, it is hard to distinguish a gesture from just 10 frames, so information should be fused across windows to label a gesture. As the continuous gesture recognition task requires a label at every frame, the temporal fusion methods should be dynamic. This section discusses three dynamic temporal fusion mechanisms for continuous gesture recognition. Section 4.2.1 introduces the late pooling, Gaussian pooling and LSTM methods. These methods are evaluated on ChaLearn ConGD and NVIDIA continuous datasets in Sections 4.2.2 and 4.2.3 respectively.

4.2.1 Dynamic Temporal Fusion

Dynamic temporal fusion methods are methods that temporally fuse information while outputting a gesture label at every tumestep. We discuss three dynamic temporal fusion mechanisms in this thesis: *Late Pooling*, which combines information at the label (softmax) level, *Gaussian Pooling*, which smooths features prior to the fully connected layer, and recurrent networks in the form of *LSTMs*. This section gives more information about these three techniques, but it is also worth noting that other techniques were evaluated and rejected, including Max Pooling over small temporal windows, GRU recurrent networks [11], stacked LSTMs [37] and nested LSTMs [83]

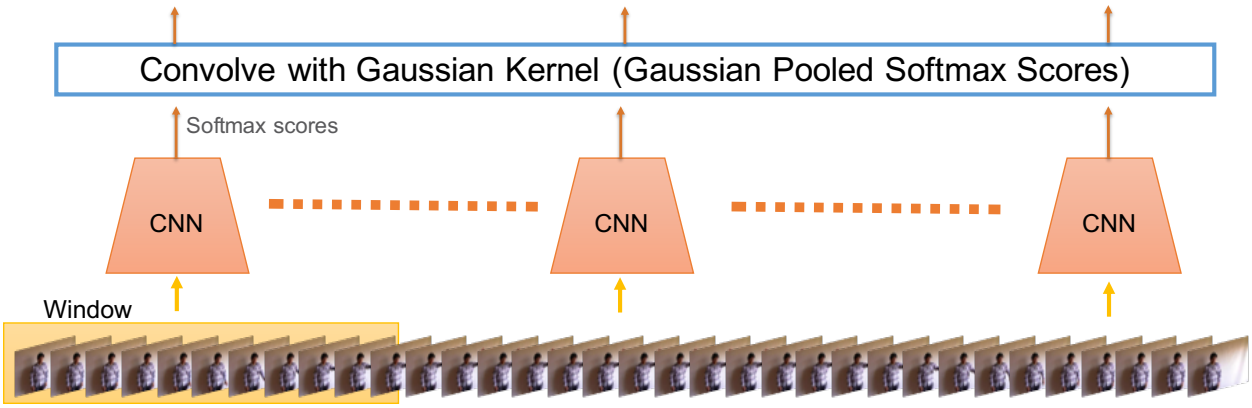

Figure 4.3: Late pooling architecture. A sliding window of 10 frames is processed by CNNs (orange) and the softmax scores from CNNs convolved over time with a Gaussian kernel resulting in a pooled softmax vector at each time step.

Late Pooling

Late Pooling fuses information across time by convolving the sequence of softmax vectors generated for every frame with a 1-D Gaussian kernel. More formally, for every frame t, a channel produces a vector of C softmax scores, where C is the number of classes. These vectors can be stacked together to form a matrix S of dimensions $C \times T$. Let G be the 1-D Gaussian kernel of length 5σ+1, calculated as $G(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$, where σ is the standard deviation and μ $(= 2.5\sigma + 1)$ is the mean of the Gaussian kernel. The pooled softmax scores S_{lp} are calculated as $S_{lp} = S * G$, and the argmax of S_{lp} at time t gives the predicted gesture label at time t. The value of σ is set as $\frac{1}{4}$th of the average gesture length in the dataset, so that 2 standard deviations of the Gaussian kernel covers the average length of the gesture.

Gaussian Pooling

Whereas Late Pooling fuses labels, Gaussian Pooling fuses convolutional features (FC features) before they are converted into labels by a fully connected layer. Similar to Late Pooling, the FC features are convolved with a 1-D Gaussian kernel. A fully connected layer is then trained to classify the pooled features, producing a logit vector that is fed into a softmax function. More formally, for every frame t, a channel produces a feature vector FC of length $|FC|$ (2048 for global nets and 2062 for focus nets). These vectors can be stacked together to form a matrix

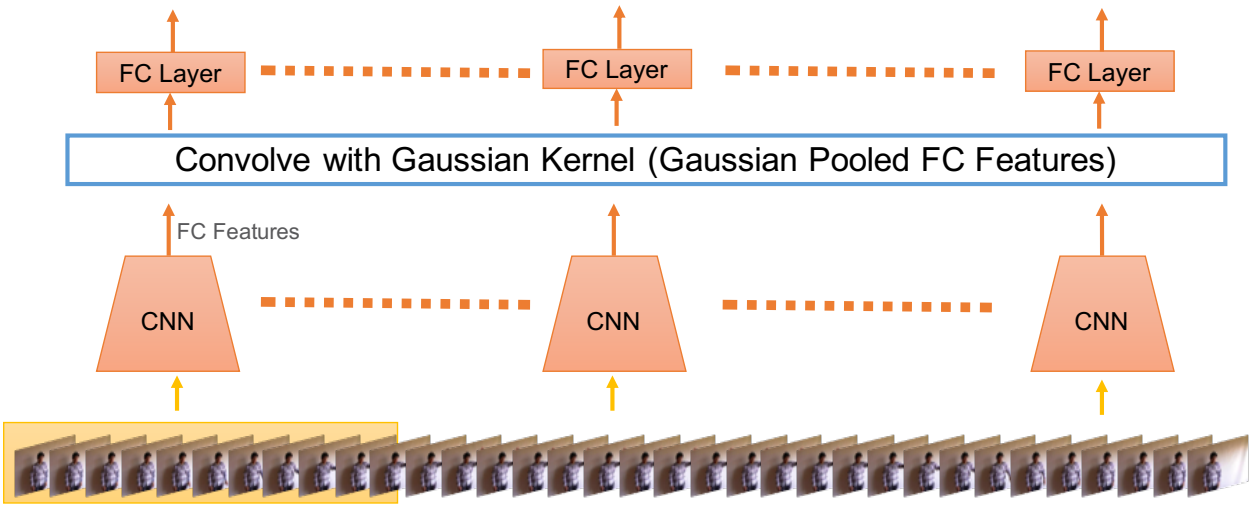

Figure 4.4: Gaussian pooling architecture. A sliding window of 10 frames is processed by CNNs resulting in a FC feature vector for every input window processed. These FC features are convolved with a Gaussian kernel over time resulting in a pooled FC feature vector at each time step. The pooled FC vector is the classified by a fully connected layer.

FC_T of dimensions $|FC| \times T$. Let G be the 1-D Gaussian kernel of length 5σ+1, calculated as $G(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$, where σ is the standard deviation and μ $(= 2.5\sigma + 1)$ is the mean of the Gaussian kernel. The pooled features FC_{Pool} are calculated as $FC_{Pool} = FC_T * G$. A fully connected layer is then trained to classify these FC_{Pool} features.

LSTMs

Unlike Gaussian Pooling and Late Pooling, recurrent neural networks explicitly model sequences of variable length and take temporal orderings into account. We model RNNs as LSTM networks [29, 45] applied to the concatenation of the convolutional features of the CNNs (as in Gaussian Pooling) and the softmax score (as in Late Fusion), as shown in Figure 4.5. The LSTM outputs are passed through a softmax classifier to label every sliding window. Based on pilot experiments, we use a 1024 memory cell LSTM.

More formally, for every frame t in video v (excluding the first 4 and last 5 frames), a channel produces a FC feature vector of length F, where F is the number of features (2048 for global nets and 2062 for focus nets) and a softmax vector of length C, where C is the number of gestures (249 for ChaLearn IsoGD dataset and 25 for NVIDIA dataset). The FC feature vector and softmax

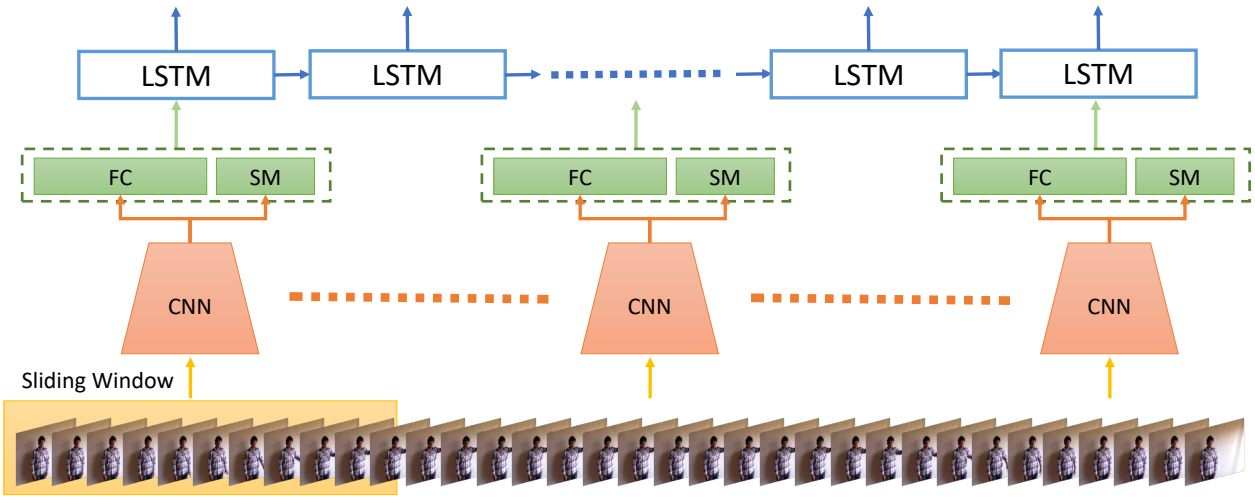

Figure 4.5: Architecture of Recurrent Neural Networks. A sliding window of 10 frames is processed by CNNs (orange) and the FC features and softmax scores from CNNs are stacked together (green), These stacked features are processed forward through time by LSTMs. A softmax layer predicts the gestures at each time step.

vector are concatenated together and passed to a LSTM with 1024 neurons as shown in Figure 4.9. The LSTM outputs feature vectors of length 1024 for each timestep t which are passed to a fully connected layer for classification. The weights of this fully connected layer are shared across the timesteps. The argmax of the softmax scores at each timestep gives the gesture label at every timestep.

4.2.2 ChaLearn ConGD Experiments

This section compares the performance of the above dynamic temporal fusion methods on ChaLearn ConGD dataset at the level of individual channel accuracy. As we are reporting the results on ChaLearn ConGD dataset for the first time in this thesis, we also discuss the training and inference model of CNNs in individual channels in addition to the training and inference of different temporal fusion methods. The results on ChaLearn ConGD show that there is an interplay between channels and the best temporal fusion method for that channel. We also conjecture about its cause in this section.

Training Process

To accelerate training, we “warm start" the weights of the focus channel nets from ResNet-50 [43] trained on ImageNet [16], and then use the weights of the trained right hand nets to fine-tune the global channels. To warm start focus channels from ResNet-50, the first pretrained convolutional layer weights $(7 \times 7 \times 3 \times 64)$ are repeated 10 times and stacked together $(7 \times 7 \times 30 \times 64)$ to account for the different number of input channels (3 vs 30) between ImageNet and focus channels. The fully connected layer weights are randomly initialized and the nets are trained end to end using mini-batch stochastic gradient descent with momentum (0.9) and a random batch of size 64. The input volume is randomly cropped to $100 \times 100 \times 30$ and flipping is performed so that a single sample can help train both right hand and left hand nets. The learning rate is initially set to $2e-4$ and decays exponentially with a decay factor of 0.7 and decay steps of 40,000.

The fine-tuned right hand focus channels are used as a warm start for the respective global channels. For example, the RGB global channel is trained by fine-tuning the RGB right hand focus channel. The global channel nets are also trained end to end using mini-batch stochastic gradient descent with the same momentum term, batch size and learning rate rules as focus channel nets. The input volume is randomly cropped to a $224 \times 224 \times 30$ volume and random flipping is performed for data augmentation.

When training temporal fusion methods, the FC features and softmax scores from all channels are precomputed. Gaussian pooling is trained by first convolving the FC features with a Gaussian kernel and then training a fully connected layer weights using the Adam optimizer [54] with a batch size of 64. The initial learning rate is set to 0.01 for first 10,000 steps, and is decreased to 0.001 till 20,000 steps and is further decreased to 0.0001. Training stops after 100,000 iterations. LSTMs are trained by stacking the FC features and softmax scores as input and training LSTM weights using the Adam optimizer [54] with a batch size of 64 and a dropout of 0.25. The cost function is calculated over all the frames of input and the gradients are backpropagated at each frame. The learning rate rule and training iterations is similar to Gaussian pooling. Sparse network fusion weights are learned by precomputing the softmax scores of all the channels and convolving these

scores with a Gaussian kernel of size 51. The weights are trained using the Adam optimizer [54] with a batch size of 32. The learning rate rule and training iterations are similar to Gaussian pooling.

Inference Process

During inference, data is passed through the convolutional networks without augmentation (cropping or flipping). Features and softmax scores are calculated at every timestep. The FC features from global channels are convolved with a Gaussian kernel and passed through the fully connected layer trained by Gaussian Pooling. The FC features and softmax scores from focused flow field channels are stacked together and passed through the LSTM. The softmax scores from all channels are convolved with a Gaussian kernel (Late Pooling) and the scores are stacked together and multiplied by the fusion layer weights and the diagonal of the resulting matrix is extracted. These softmax scores are further fused using Late Pooling and the argmax of the softmax scores is the predicted gesture label for the timestep.

Results

Table 4.1 looks at the interplay between processing channels and temporal fusion mechanisms. Each row corresponds to one of the 12 processing channel of FOANet. The values in a row show the mean Jaccard Index scores of the channel given one of the three temporal fusion strategies (Late Pooling, Gaussian Pooling or LSTMs) on the validation and tests data sets. The Jaccard index measures the average relative overlap between true and predicted sequences of frames for a given gesture and is used as the standard performance measure for continuous gesture recognition [119].

Two quick observations can be made based on Table 4.1. The first is that no single temporal fusion method is the best on all channels. The second is that when performance is averaged across all 12 channels, LSTMs are a better temporal fusion mechanism than Late Pooling or Gaussian Pooling as shown in the last row of Table 4.1.

The most significant finding in Table 4.1 lies in the pattern of maximum values, highlighted with light blue backgrounds. For all four global channels, for both the validation and test data sets,

the highest mean Jaccard Index is achieved using Gaussian Pooling. Focused channels behave differently, however. Focused channels of “raw" data, i.e. channels that process windows of RGB or depth values, achieve the highest mean Jaccard Index using Late Pooling on both the validation and test sets. Focused channels of flow fields achieve their highest values with LSTMs.

Table 4.1: Mean Jaccard Index scores of channels fused by different temporal fusion mechanisms on validation and test set of ChaLearn ConGD. The last row shows the average mean Jaccard index across 12 channels. The numbers in bold with blue backgrounds represent the best scores for a given channel.

	Valid			Test		
Temporal Fusion / Channels	**Late Pooling**	**Gaussian Pooling**	**LSTM**	**Late Pooling**	**Gaussian Pooling**	**LSTM**
RGB Global	0.2210	**0.2501**	0.2320	0.2164	**0.2379**	0.2323
Depth Global	0.2784	**0.3286**	0.3084	0.3221	**0.3423**	0.3353
RGB Flow Global	0.396	**0.4408**	0.3983	0.3735	**0.4188**	0.4012
Depth Flow Global	0.308	**0.3476**	0.2980	0.3387	**0.3684**	0.3204
RGB Left	**0.2337**	0.2042	0.2207	**0.2162**	0.1822	0.2126
RGB Right	**0.2745**	0.2482	0.2455	**0.3954**	0.3455	0.3557
Depth Left	**0.3104**	0.2865	0.3018	**0.2836**	0.2704	0.2831
Depth Right	**0.4225**	0.3690	0.3948	**0.5293**	0.4754	0.5178
RGB Flow Left	0.3336	0.3225	**0.3383**	0.3038	0.2735	**0.3041**
RGB Flow Right	0.4439	0.4235	**0.4522**	0.4912	0.4709	**0.4918**
Depth Flow Left	0.2715	0.2893	**0.3104**	0.2728	0.2643	**0.2753**
Depth Flow Right	0.3693	0.3896	**0.4194**	0.4955	0.4632	**0.5130**
Average	0.3219	0.3250	**0.3266**	0.3457	0.3427	**0.3536**

This is a strong pattern, even though some of the numeric differences are small. We have three types of channels: global, focused raw and focused flow field. We have four channels of each type, and each channel is tested on two data sets. We therefore have 8 3-way comparisons among fusion mechanisms for each channel type, for a total of 24 3-way comparisons. Across all of these comparisons, every channel type has a single fusion mechanism that is always best, it’s just that the best mechanisms depends on the channel type. The data set is also very large, so none of the differences can be attributed to noise. When comparing temporal fusion techniques, all 3 algorithms are applied to the same samples and produce answers that are either correct or

incorrect. We can therefore test for statistical significance using McNemar's test [76]. In all 24 cases, the maximum value is significantly greater than the other two at a level of $p = 1e - 05$.

Analysis

Table 4.1 reveals a clear relationship between channel type and the optimal temporal fusion strategy, but it doesn't explain why. Since the architectures of the channels are the same, there must be some difference in the data being processed that makes one fusion model more or less appropriate than another. We have looked at various properties of the FC features and softmax vectors being produced by the CNNs inside the information channels. Only one seems to correlate to the choice of temporal fusion strategy: *prediction stability*. We define prediction stability as the frequency with which a CNN predicts the same label (whether right or wrong) on two consecutive time steps. Figure 4.6 shows the prediction stability of each of the 12 channels, with global channels shown in blue to the left, raw focused channels shown in orange in the middle, and focused flow field channels shown in purple on the right.

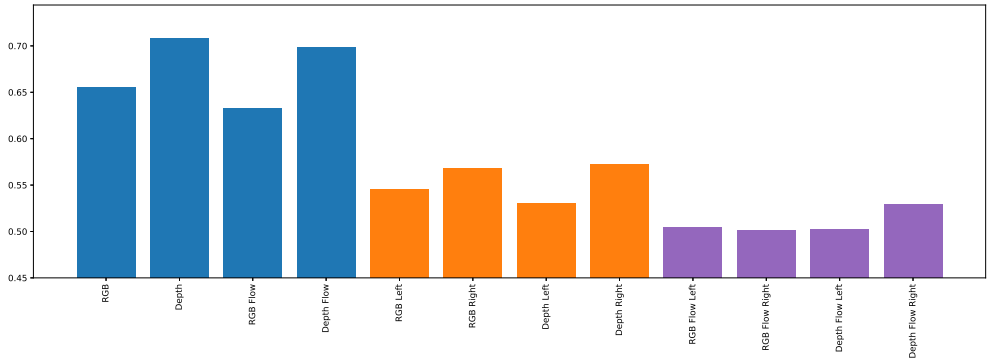

Figure 4.6: Bar plots showing prediction stability of CNNs for all 12 channels. Prediction stability is defined as the frequency with which a CNN predicts the same label on two consecutive time steps. Global channels show the most prediction stability, while raw focused channels show intermediate stability and focused flow field channels are the least stable.

Figure 4.6 suggests that the CNNs within individual channels are not as stable as one might guess. Even the global depth channel, which is the most stable, only predicts the same label for two consecutive time steps about 75% of the time. Other channels are as low as 50%. Some of this can

be explained by the large label set (the likelihood of picking the same label twice at random is $\frac{1}{249}$) and the large number of frames between gestures during which no label is obvious. Nonetheless, the lack of stability reinforces the importance of temporal fusion to smooth out data over time.

More importantly for this analysis, prediction stability correlates to the choice of temporal fusion strategy. Global channels are the most stable, and they perform best with Gaussian Pooling. Raw focused channels have intermediate stability and perform best with Late Pooling. Focused flow field channels are the least stable, and perform best with LSTMs.

We can only speculate as to why prediction stability correlates to the relative performance of temporal fusion strategies. LSTMs model long-term dependencies, which may allow them to detect long-term patterns in highly noisy data. Ng *et al.* also noticed that LSTMs outperform pooling-based strategies when processing optical flow fields [85] and suggested that LSTMs process "optical flow in a manner which lends itself to late model fusion" [85].

Gaussian pooling, on the other hand, is designed to filter white noise. Global channels are the most stable channels, as can be seen in Figure 4.6. We hypothesize that their errors are better modeled as temporal white noise, so that Gaussian Pooling performs best on global channels. We cannot prove this conjecture, however, nor can we explain why Late Pooling outperforms other methods on raw focused channels.

4.2.3 NVIDIA Experiments

The pattern in Table 4.1 was so striking that we wanted to make sure it would hold for a different dataset. We chose NVIDIA dataset as another continuous dataset. There is both a isolated (segmented) and continuous version of NVIDIA dataset. Chapter 3 used the isolated version; now we use the continuous one. In the continuous NVIDIA dataset, every subjects starts in a neutral position (hands on steering wheel), performs a gesture, and then returns to the neutral position. Every video therefore has three labels: drive, gesture, and drive. All gestures are performed with the right hand, and the left hand is never in the field of view. Therefore there are only 8 (instead of 12) FOANet channels. Although every video in NVIDIA dataset has three labels: drive, gesture,

and drive, the timing is too regular: subjects always start gesturing around frame 130 (± 20) and the gestures last for 80 frames. To make the data more realistic, we select random start and end points for each video in the dataset such that all videos remain at least 60 frames long.

Table 4.2: Mean Jaccard Index scores of channels on the NVIDIA dataset for different temporal fusion strategies.

	Global				**Focus**			
	RGB	**Depth**	**RGB Flow**	**Depth Flow**	**RGB**	**Depth**	**RGB Flow**	**Depth Flow**
Late Pooling	0.4632	0.4836	0.5549	0.4895	**0.4792**	**0.5265**	0.5154	0.4924
Gaussian Pooling	**0.5402**	**0.5532**	**0.5701**	**0.5364**	0.4137	0.5048	0.5071	0.4314
LSTM	0.5047	0.5326	0.5618	0.5304	0.4537	0.5092	**0.5621**	**0.5149**

Table 4.2 shows the results of FOANet on the NVIDIA dataset. Note that this time channels are columns and fusion strategies are rows (the opposite of Tabel 4.1). The CNNs, temporal fusion methods and sparse fusion network are trained in the same way as ChaLearn ConGD (see Section 4.2.2), except that random flipping is not performed and the focus channels are only trained on right hands. Table 4.2 displays the same pattern as Table 4.1, albeit with fewer channels. Once again, the global channels perform best with Gaussian Pooling, the (now 2) raw focused channels perform best with Late Pooling, and focused flow field channels perform best with LSTMs.

4.3 Temporal Fusion Methods for Isolated Gesture Recognition

The results in the previous section show that there is a clear interplay between channel type and temporal fusion method for continuous gesture recognition. This section evaluates whether the same pattern holds for isolated gesture recognition. Isolated gesture recognition videos have a single gesture in the video and the task is to assign a single gesture label to the whole video. As previously discussed, FOANet divides the processing into multiple channels, one for each com-

bination of spatial attention region and modalities. To assign a single gesture label to the entire video, the information (features and/or softmax scores) from these sliding windows must be fused temporally. This section discusses three temporal fusion strategies for isolated gesture recognition. Section 4.3.1 introduces the average, pooling and RNN temporal fusion methods. These methods are evaluated on ChaLearn IsoGD and NVIDIA datasets in Sections 4.3.2 and 4.3.3 respectively.

4.3.1 Static Temporal Fusion Methods

Static temporal fusion methods are methods that fuse the information from a whole video and summarize it into a single feature vector or softmax score. Average, pooling and Recurrent Neural Networks are the three static temporal fusion methods discussed in this thesis and they are described as follows:

Average

FOANet channels produce a vector of softmax scores at each time step. To create a single softmax vector for the whole video, one approach is to average the softmax scores from each window across time as shown in Figure 4.7. This is a "weak" technique in the sense that it requires neither training nor *a priori* knowledge. More formally, for every frame t in video v (excluding the first 4 and last 5 frames), a channel produces a vector of softmax scores of length C, where C is the number of classes. These vectors can be stacked together to form a matrix S of dimensions C x T . Let $j = [\frac{1}{T}, \frac{1}{T}, \frac{1}{T}, ..., \frac{1}{T}]'$ be a column vector of dimensions T x 1. The softmax scores for the video v denoted by S_v are calculated by taking the mean across the time axis as $S_v = Sj$. The argmax of the resulting mean softmax vector gives the gesture class prediction. Note that this is the segmented (isolated gesture recognition) equivalent of Late Pooling.

Averaging the softmax scores is the simplest temporal fusion strategy. It doesn't take into account the temporal structure of the gesture, so it is hard for this method to distinguish between gestures such as rotate clockwise and rotate counter clockwise. Moreover, averaging is a late fusion strategy in the sense that the fusion is done after 2D CNNs independently classify the

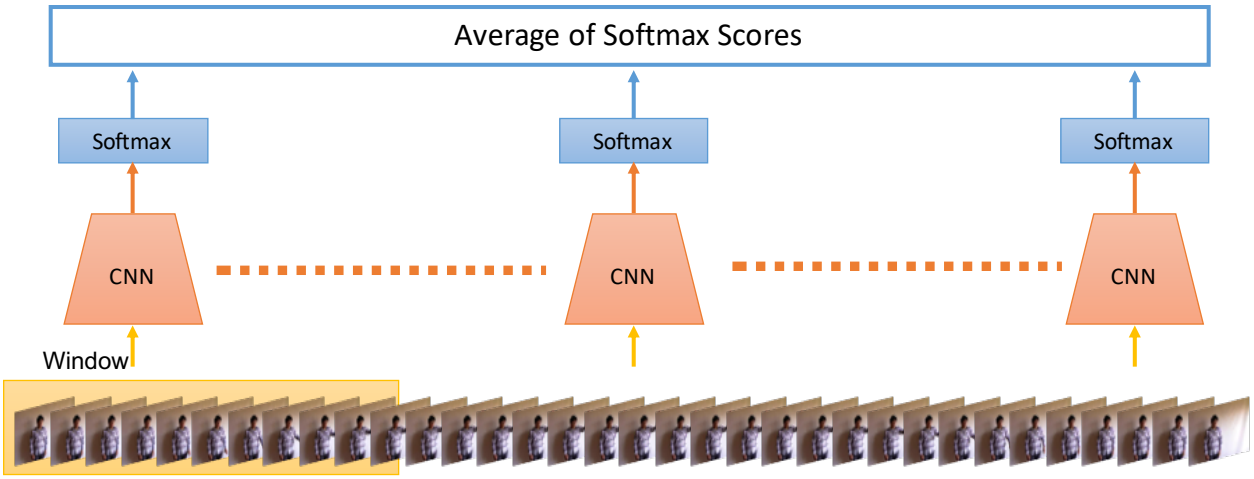

Figure 4.7: Temporal fusion strategy of averaging the softmax scores from all windows. Global and focus channels take a 10 frame sliding window as input and outputs a softmax vector of length C, where C is the number of classes in the dataset. The softmax scores from all the windows are averaged together to get softmax scores of the video.

sliding windows. As the sliding windows are fused late in the processing, this method can't make full use of the information present in different sliding windows.

Pooling

Pooling is another strategy to temporally fuse information. Here, the information is fused one step earlier, compared to the late fusion model of averaging. Fully connected features (FC) can be extracted from FOANet channels for each sliding window. For a video, FC features from all overlapping sliding windows can be pooled together either by max pooling or average pooling. A fully connected layer classifies the resulting pooled FC features as shown in Figure 4.8. More formally, for every frame t in video v (excluding the first 4 and last 5 frames), a channel produces a FC feature vector of length F, where F is the number of features (2048 for global nets and 2062 for focus nets). These vectors can be stacked together to form a matrix FC of dimensions F x T. Let $j = [\frac{1}{T}, \frac{1}{T}, \frac{1}{T}, ..., \frac{1}{T}]'$ be a column vector of dimensions T x 1. Then the average pooled FC features denoted by $FC_{Average}$ are calculated as $FC_{Average} = FCj$. Similarly, the max pooled FC features denoted by FC_{Max} are calculated as $FC_{Max} = \max_{1 \leq t \leq T} FC_{ft}$. This temporal pooling layer is followed by a fully connected layer and a softmax function. The argmax of the resulting

softmax vector gives the gesture class prediction. Note that this is the segmented (isolated gesture recognition) equivalent of Gaussian Pooling

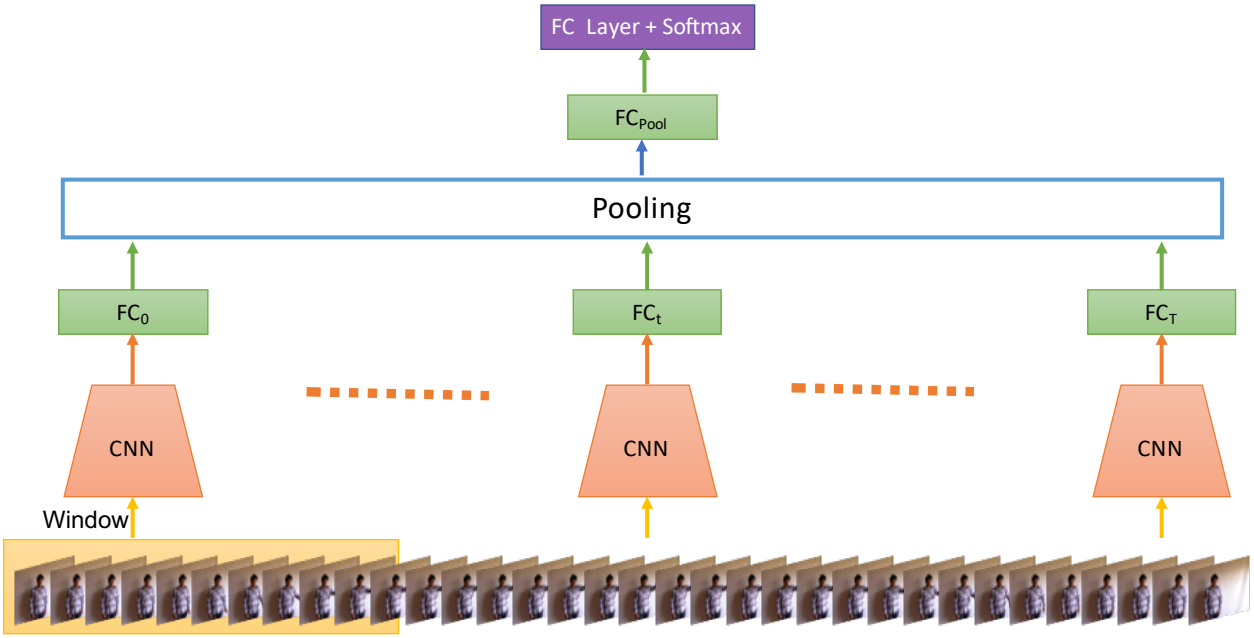

Figure 4.8: Temporal fusion strategy of pooling the FC features from all windows. Global and focus channels take a 10 frame sliding window as input and outputs a FC feature vector of length F(2048 for global nets and 2062 for focus nets). The FC features from all the windows are pooled together to get a single FC feature vector for the video. Average pooling and Max pooling are two pooling strategies. A fully connected layer stacked on top of pooled features is used for gesture classification.

Similar to the averaging strategy discussed previously, pooling doesn't take into account the temporal structure of the gesture, so it can't distinguish between clockwise and counter clockwise rotation gestures. As pooling methods fuse the information from sliding windows a layer earlier, compared to averaging, pooling methods have the capability to capture more information from sliding windows than averaging.

Recurrent Neural Network

Unlike the temporal fusion strategies discussed previously that ignore temporal structure, recurrent neural networks (RNNs) capture the temporal structure of gestures and long range dependencies as well. Similar to RNNs in continuous gesture recognition, we use a LSTMs with 1024

neurons for isolated gesture recognition. The only difference is that the softmax scores from all timesteps is averaged to get a single output per video as shown in Figure 4.9.

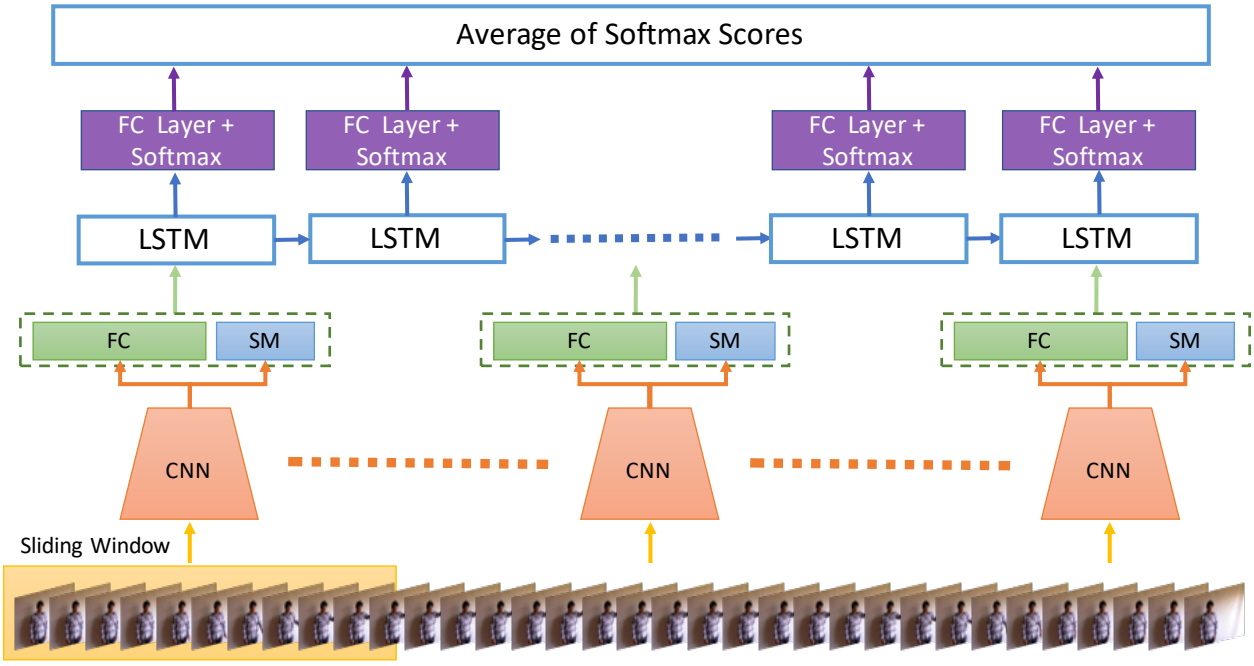

Figure 4.9: Recurrent neural network for isolated gesture recognition. Global and focus channels take a 10 frame sliding window as input and outputs a FC feature vector of length F(2048 for global nets and 2062 for focus nets) along with a softmax vector of length C (249 for ChaLearn IsoGD and 25 for NVIDIA). FC feature vector and softmax feature vector are concatenated together and is passed as input to LSTMs. The features from LSTM are classified by a fully connected layer and the softmax scores across all timesteps is averaged together.

4.3.2 ChaLearn IsoGD Experiments

The experiments in this section evaluate average, pooling and RNN methods on the ChaLearn IsoGD dataset. The temporal fusion methods are evaluated to select the best method on individual channels and the dataset as a whole. These experiments are also meant to provide intuitions on how pooling methods interact with modalities and channels on the ChaLearn IsoGD dataset.

Training and Inference

The first method of temporal fusion simply averages the softmax scores from all windows and requires no training. During inference, the softmax scores of all the overlapping windows from a

video are calculated. These softmax scores are averaged together and the argmax of this average vector gives the predicted class label.

Training of pooling based temporal fusion method is done by first extracting the FC features from all windows of dataset. This is done to prevent extracting the features from the same window multiple time when training the fully connected layer on pooled features. The last fully connected layer weights are randomly initialized and the weights are trained using the Adam optimizer [54] with a batch size of 32. A random batch is selected by randomly choosing 32 videos form the training set of ChaLearn IsoGD dataset. For each random video, random number of windows (between 1 and T, where T is the total number of windows in the video) are extracted from the video and their FC features are pooled together. The initial learning rate is set to 0.01 for first 10,000 steps, and is set to 0.001 till 15,000 steps and is further decreased to 0.0001. The training is stopped after 20,000 iterations. During inference time, all windows of the video are pooled together and the trained fully connected layer is used for classification. During the training of pooling strategies, a training model is saved every 500 training steps. The best model is selected based on its accuracy on the validation set and the best model is used to report accuracy on test set.

Similar to pooling strategy, FC features and softmax scores are pre-computed and concatenated for all windows in dataset to train LSTMs. The LSTM weights and fully connected feature weights are randomly initialized and are trained using the Adam optimizer [54] with a batch size of 64. A random batch is selected by randomly choosing 64 videos form the training set of ChaLearn IsoGD dataset and within each video choosing atleast 25% of frames while preserving the order. The initial learning rate is set to 0.01 for first 10,000 steps, and is set to 0.001 till 20,000 steps and is further decreased to 0.0001 till 60,000 steps. The learning rate is then set to 0.00001 and the training is stopped after 100,000 iterations. A dropout of 25% is applied to LSTM neurons and the network is trained as a labeling network (the costs are calculated at every timestep), so that the LSTMs try to predict the correct label at every timestep. During inference, FC features and softmax scores from all windows of a video are passed to LSTM and the softmax scores from all

timesteps are averaged together. Accuracy on validation set is calculated every 1000 training steps and the model with highest validation accuracy is used to classify the test set.

Results

Table 4.3 looks at the interplay between processing channels and static temporal fusion mechanisms on the ChaLearn IsoGD dataset. Each row corresponds to one of the 12 processing channels. The values in a row show the video level accuracy of the channel obtained by temporally fusing with different temporal fusion strategies (average, max pool, average pool or LSTMs) on the validation and test sets.

The results in Table 4.3 show that there is no single temporal fusion method that is best on all channels. We see that different channels perform best when fused using different temporal fusion methods. However, there is no clear pattern evident in a group of channels preferring one temporal fusion method over another on both validation and test sets.

All left hand channels gain maximum accuracy on the validation set when they are temporally fused by average pooing. However, on the test set this pattern is observed only on three of four modalities. On test set, RGB left hand channels performs best when fused by LSTMs.

We see that right hand channels from RGB, depth and depth flow modalities perform best when softmax scores are averaged across time. However, on test set depth right channel performed best when fused using average pool instead of average. RGB flow right channel that achieved high accuracy when fused using average pool on validation set performed best on test set when average is used instead.

Similarly, there is no clear pattern in global channels as well. RGB and depth global channels perform best when fused by max pool method on validation set. However, on test set RGB global channels prefers average pool over max pool. RGB flow global channels achieves best accuracy with average method both on validation and test sets. Depth flow global channel prefers LSTMs on validation set and average on test set.

In summary, there is not a single temporal fusion channel that performs best on all channels, and there is no pattern of a temporal fusion method outperforming others on similar attention or

Table 4.3: Different temporal fusion accuracies on individual channels of ChaLearn IsoGD validation and test set. The numbers in bold with blue backgrounds represent the best scores for a given channel.

Validation					
Modality	**Channel**	**Average**	**Max Pool**	**Average Pool**	**LSTM**
RGB	**Global**	33.22	**37.45**	37.02	36.19
	Left	16.17	15.77	**16.31**	15.23
	Right	**41.60**	38.23	41.13	38.69
Depth	**Global**	27.98	**32.57**	32.23	31.30
	Left	23.76	23.48	**24.14**	23.51
	Right	**54.91**	53.74	54.59	54.14
RGB Flow	**Global**	**46.22**	38.80	45.77	41.03
	Left	24.14	24.88	**26.63**	26.28
	Right	54.60	51.97	**55.04**	51.13
Depth Flow	**Global**	31.66	30.24	32.37	**33.12**
	Left	21.84	21.94	**23.18**	21.91
	Right	**48.32**	45.43	47.69	47.23
Average		35.37	34.54	**36.34**	34.98
Test					
Modality	**Channel**	**Average**	**Max Pool**	**Average Pool**	**LSTM**
RGB	**Global**	41.27	43.41	**45.63**	44.17
	Left	16.63	16.24	17.26	**18.78**
	Right	**47.41**	44.28	46.82	45.39
Depth	**Global**	38.50	**44.02**	41.73	43.68
	Left	24.06	23.31	**24.67**	23.91
	Right	64.44	63.28	**65.50**	64.19
RGB Flow	**Global**	**50.96**	41.35	50.74	50.39
	Left	24.02	23.72	**25.94**	25.78
	Right	**59.69**	54.74	57.96	54.89
Depth Flow	**Global**	**42.02**	37.11	41.49	41.50
	Left	22.71	21.23	**23.02**	23.31
	Right	**58.79**	54.69	56.11	57.29
Average		40.87	38.94	**41.41**	41.10

modality channels. We can see that average and average pooling are two best temporal fusion methods (in no particular order) in terms of number of channels that they perform best on. When the performance of all 12 channels are averaged, average pooling is the best temporal fusion as can be seen in Table 4.3. As average method doesn't involve any additional training, we use average temporal fusion method for ChaLearn IsoGD experiments in Chapter 5.

4.3.3 NVIDIA Experiments

The training and inference of different temporal fusion strategies on the NVIDIA dataset is similar to the training and inference of ChaLearn IsoGD dataset as discussed in Section 4.3.2. Table 4.4 shows the accuracies of different temporal fusion strategies on 8 channels of the NVIDIA test set.

Looking at Table 4.4, we can see that no temporal fusion method performed best on all channels similar to ChaLearn IsoGD (see Table 4.3). Max pool is the best temporal fusion method for NVIDIA dataset as it outperforms other temporal fusion methods on five out of eight channels. Max pool is the best temporal fusion strategy even when the average performance across 8 channels is considered as can be seen in last row of Table 4.4. Average pool comes second by outperforming other methods on two channels. Average is not the best temporal fusion method for any channel.

Table 4.4: Accuracies of different temporal fusion strategies on individual channels of NVIDIA test set. The numbers in bold with blue backgrounds represent the best scores for a given channel.

Modality	Channel	Average	Max Pool	Average Pool	LSTM
RGB	**Global**	43.98	**52.12**	47.68	47.75
	Focus	58.09	59.13	56.18	**60.61**
Depth	**Global**	66.80	**72.74**	69.59	69.18
	Focus	70.12	**78.22**	75.98	76.35
RGB Flow	**Global**	62.66	**72.78**	71.33	65.71
	Focus	77.18	**78.13**	78.09	75.31
Depth Flow	**Global**	58.71	73.40	**73.53**	34.27
	Focus	73.65	77.01	**75.52**	73.47
Average		42.60	**46.96**	45.66	41.89

These results are in complete contrast to the performance of temporal fusion methods on ChaLearn IsoGD dataset. On ChaLearn IsoGD dataset average is the best temporal fusion method and max pool is the third best method. The experiments on ChaLearn IsoGD and NVIDIA datasets shows that there is no interplay between channels and static temporal fusion methods that is consistent across isolated gesture recognition datasets.

4.4 Summary

Previous chapter introduced the FOANet attention mechanism to divide processing into multiple channels and they are evaluated on two isolated gesture recognition datasets. This chapter presented temporal fusion methods to extend FOANet architecture to do continuous gesture recognition. Continuous gesture recognition refers to the task of assigning a label to every frame for a video with multiple unsegmented gestures. This chapter discussed three dynamic temporal fusion methods for continuous gesture recognition - late pooling, Gaussian pooling and LSTMs. These dynamic temporal fusion methods are evaluated on ChaLearn ConGD and continuous NVIDIA datasets. As these methods are used for continuous gesture recognition, perhaps not surprisingly, recurrent neural networks were the best overall technique, averaged across all 12 channels. What is surprising, however, is that different channels performed best using different temporal fusion strategies, and that the best fusion strategy is determined by the target (global vs. left/right hand) and modality (RGB, depth or flow field) of the channel. More specifically, global channels perform best with *Gaussian Pooling*, flow field channels focused on the left or right hand perform best with *LSTMs*, and RGB and depth channels focused on the left or right hand perform best with *Late Fusion*.

As the results of continuous gesture recognition revealed an interesting interplay between channel type and temporal fusion methods, we also did experiments to see if the pattern holds for isolated gesture recognition. As isolated gesture recognition refers to the task of assigning a single gesture label to a video with only one gesture, this chapter discussed three static temporal fusion methods for isolated gesture recognition - average, pooling (max and average) and LSTMs. These static temporal fusion methods are evaluated on ChaLearn IsoGD and NVIDIA datasets. Experimental results shows that the relative performance of static temporal fusion strategies doesn't have a common pattern across datasets and channels. On the ChaLearn IsoGD dataset, average softmax was the best performer in most cases. However on the NVIDIA dataset, average softmax never outperformed other methods. Max pool mechanism that was the third best temporal fusion method on IsoGD dataset is the best temporal fusion method when applied to NVIDIA dataset. As LSTMs

came last on both datasets, they might not be suitable temporal fusion models for isolated gesture recognition.

Chapter 5

Channel Fusion

The architecture of FOANet introduced in Chapter 3 divides processing into M $\times$ L channels, where M is the number of modalities and L is the number of spatial locations (body parts) focused on. More specifically, we use 4 modalities (RGB, depth, RGB flow, and depth flow), and 3 spatial locations (global, right hand, left hand). This leads to 12 global and focused channels (8 for datasets that have only one hand visible). Chapter 4 discussed different methods to temporally fuse information within a channel. These M $\times$ L channels are only useful if the data can be fused back together to produce a single gesture label per video for isolated gesture recognition and a single gesture label per frame for continuous gesture recognition.

The temporal fusion methods discussed in previous chapter are useful within channels. These temporal fusion methods can't be used to fuse information across channels, however (except for averaging). This is because channel fusion requires combining information from dependent channels that are trained independently. There is no temporal ordering, as required for LSTMs. Moreover, there is no correspondence between features from independently trained channels, so feature pooling strategies can't be used either.

Section 5.1 discusses strategies to fuse information across channels. Most significantly, Section 5.1.3 discusses sparse network fusion - a novel method for channel fusion that learns the relative importance of the different spatial regions and data modalities for every gesture type. These channel fusion methods are evaluated on ChaLearn IsoGD, NVIDIA and ChaLearn ConGD datasets in Section 5.2. Section 5.3 summarizes and concludes the chapter.

5.1 Channel Fusion Methods

After FOANet divides the processing into multiple channels and the information within channels is temporally fused, the information must be fused across channels. On one extreme, these channels can be fused by a late fusion model by simply averaging the softmax scores from all

channels as discussed in Section 5.1.1. On the other extreme these channels can be fused a layer earlier by extracting FC features from all channels, concatenating them, and training a fully connected layer to do the classification. This concatenation based fusion strategy is discussed in Section 5.1.2. Since concatenation fusion tends to overfit on gesture datasets, we propose a more directed learning mechanism called sparse network fusion. This is a main contribution of this thesis, and is introduced in Section 5.1.3.

5.1.1 Average Fusion

The 12 global and focused channels (8 for datasets that have only one hand visible) shown in Figures 1.1 and 3.1 produce 12 response vectors for every input (i.e. for each sliding window of 10 frames). These channels produce a single softmax vector per video after temporal fusion using the techniques discussed in Chapter 4. Similar to the average temporal fusion strategy discussed in Section 4.3.1, softmax vectors from multiple channels can be averaged. This is a "weak" technique in the sense that it requires neither training nor *a priori* knowledge. More formally, for an arbitrary video v, every channel h in our gesture recognition architecture produces a vector of softmax scores of length C, where C is the number of classes. These vectors can be stacked together to form a matrix S of dimensions C x H, where H is the number of channels in our architecture. Let $j = [\frac{1}{T}, \frac{1}{T}, \frac{1}{T}, ..., \frac{1}{T}]'$ be a column vector of dimensions T x 1. The average softmax scores for the video v from multiple channels, denoted by S_v^h, are calculated by taking the mean across the time axis as $S_v^h = Sj$. The argmax of the resulting mean softmax vector gives the gesture class prediction. Average fusion is the most widely used channel fusion technique in the literature [81, 82, 107, 123, 124]

5.1.2 Concatenation

An alternative to average fusion is to concatenate the FC features and train a fully connected layer to classify these stacked features. For a video, a single FC feature vector per channel can be obtained by temporally fusing FC features from overlapping sliding windows, using one of the pooling strategies discussed in Section 4.3.1. More formally, for every frame t in video v

(excluding the first 4 and last 5 frames), a channel produces a FC feature vector of length F, where F is the number of features (2048 for global nets and 2062 for focus nets). These vectors can be pooled together resulting in a FC feature vector FC_{pool} of length F. The FC_{pool} features from all channels can be concatenated together resulting in a FC vector FC_{concat} of length $F \times H$, where H is the number of channels in our architecture. A fully connected layer followed by a softmax function can be trained on top of this concatenated feature vector FC_{concat}. The argmax of the resulting softmax vector gives the gesture class prediction. This technique is used by systems with only few channels, e.g. [68, 77].

Unfortunately there is not enough training data to train fully connected layers to fuse large number of channels. With 12 channels, the concatenated feature vector would be over 24,000 elements long, and the fusion layer as a whole would have to learn over 6 million weights. With only about 35K training videos in the IsoGD dataset, networks overfit as shown in Section 5.2.1.

5.1.3 Sparse Network Fusion

The fusion strategies discussed above have disadvantages. Averaging does not make the best use of the available information; see Section 5.2 below. Concatenation method overfits, as discussed in Section 5.1.2.

We propose a more directed learning mechanism. The goal is to learn the properties of gestures. For example, the diving gestures in ChaLearn were designed to be seen at a distance through murky water, so they involve large arm motions. Mudras, on the other hand, are small dexterous motions of one hand. Our goal is to learn how much weight to assign to a channel, given a gesture, so that global channels are emphasized for diving gestures while the right hand channels are dominant for mudras. We therefore fuse channels using a sparsely connected network with one weight per gesture $\times$ channel.

Let n be the number of channels and C be the set of classes. For video v, let $S = [s_1, s_2, s_3, ..., s_n]$ be the softmax scores, where s_i is a vector of length $|C|$. If a hand is never visible or doesn't move throughout the video (average movement less than 4 pixels), the corresponding softmax vector is

set to all zeros. For each class $c \in C$, the weight vectors $W_c = [w_{c1}, w_{c2}, w_{c3}, ..., w_{cn}]$ should be calculated to weigh the different channels according to their importance to gesture c.

We pose this problem as a perceptron learning problem where class weights are learned in tandem. Let $W = [W_1, W_2, W_3, ..., W_{|C|}]$ be the weight matrix to be learned. The dimensions of softmax score matrix S are C x n and the dimensions of weight matrix W are n x C. These two matrices can be multiplied to create F: $F = SW$. The dimensions of F are $|C|$ x $|C|$, and the diagonal elements of F represent the softmax scores of classes multiplied with their corresponding class weights ($F_{ii} = S_{i,*} \cdot W_{*,i}$), whereas the off diagonal elements represent softmax scores of a class multiplied with weights of different classes ($F_{ij} = S_{i,*} \cdot W_{*,j}$). The off diagonal elements of matrix F are therefore discarded using a Hadamard product of F with a $|C| \times |C|$ identity matrix I. A softmax function is applied to the diagonal elements of FI. The weight matrix W is learned by back propagation using cross entropy loss and mini-batch gradient descent. As the off-diagonal elements are zeroed out by the Hadamard product with I, they do not produce derivatives and the weights of a class are effected only by their corresponding softmax scores.

Figure 5.1 explains the sparse network fusion with a toy problem of four gesture classification by fusing softmax scores from three channels. As this is a four class classification problem, each channel produces a softmax vector of length four as shown in Figure 5.1. The figure shows that these softmax vectors from three channels are sparsely connected to the fused softmax vector of length four. For example, the softmax scores corresponding to gesture 1 from all channels is connected only to gesture 1 of fused softmax vector, not others. This way sparse network fusion learns the importance of individual channels to a gesture with only one weight per gesture $\times$ channel.

Figure 5.2 shows the weights learned by sparse network layer for 12 channels of FOANet for a one hand static gesture from ChaLearn IsoGD dataset. As the gesture in Figure 5.2 is a right handed gesture, we can see that sparse network fusion learned high weights for right hand channels stressing the importance of right hand channels in classifying the gesture. The network also learned to ignore left hand channels by learning negative weights for left hand channels. Global channels

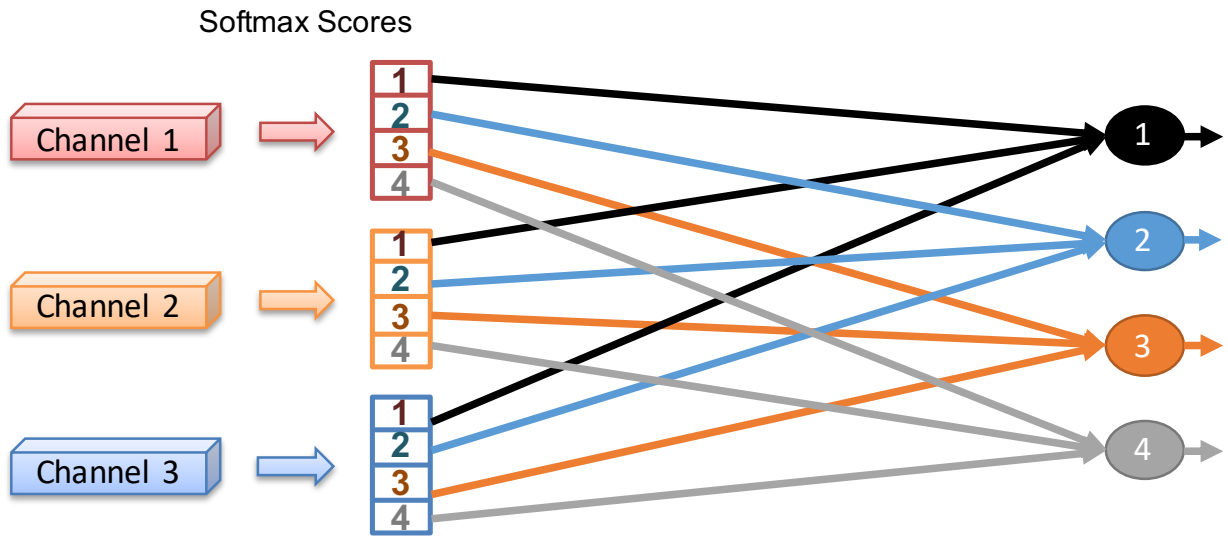

Figure 5.1: Demonstration of sparse network fusion on a toy problem with 3 channels and 4 gestures. For comparison concatenation is fully connected, whereas average is the average of individual softmax scores.

were assigned small positive weights as these channels also carry the information to classify the gesture, but are not as important as right hand channels.

Figure 5.3 shows the weights learned by sparse network layer for 12 channels of FOANet for a two handed dynamic gesture from ChaLearn IsoGD dataset. As the gesture involves large motions of two hands, sparse network fusion assigned high weights to all channels. As the gesture involves motion of arms which is not entirely captured by focus channels, global channels are important to classify this gesture and the sparse network fusion learns exactly the same. Of the three attention channels, sparse network fusion learned to prioritize global channels, followed by right hand and left hand channels.

5.2 Results

The previous three sections (Section 5.1.1, 5.1.2, and 5.1.3) discussed different fusion strategies to fuse information from multiple channels. This section evaluates these methods on ChaLearn IsoGD and NVIDIA datasets. This section also presents the sparse network fusion results on ChaLearn ConGD dataset. Section 5.2.1 presents the experimental results on the ChaLearn IsoGD

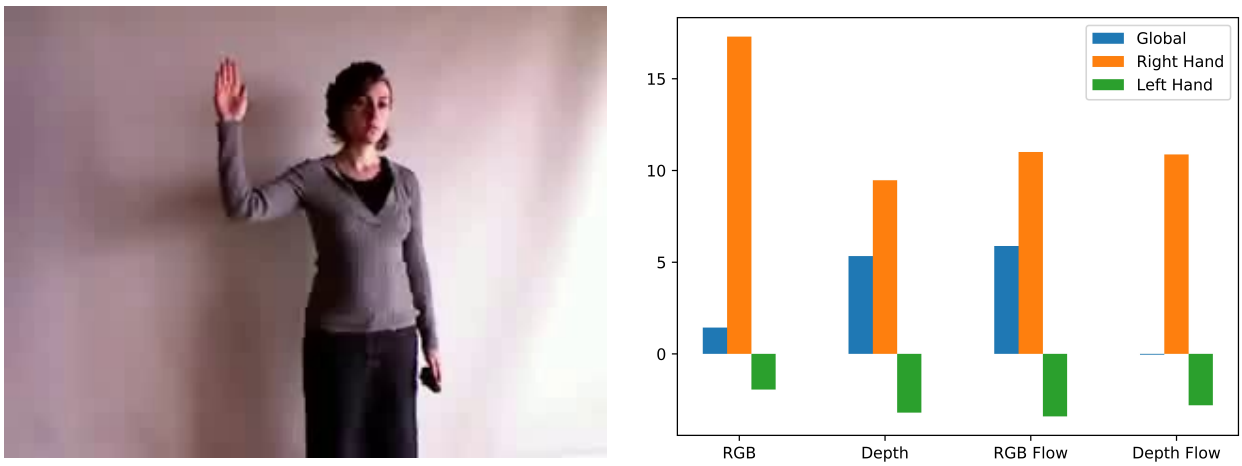

Figure 5.2: One handed static gesture from ChaLearn IsoGD dataset and the corresponding weights learned by sparse network fusion layer for 12 channels of FOANet.

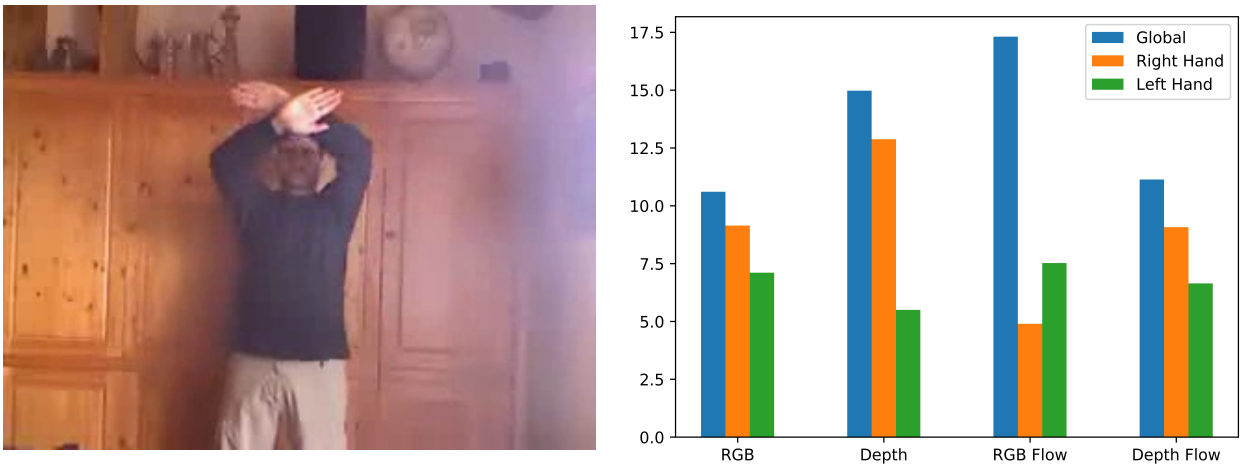

Figure 5.3: Two handed dynamic gesture from ChaLearn IsoGD dataset and the corresponding weights learned by sparse network fusion layer for 12 channels of FOANet.

dataset and Section 5.2.2 presents the experimental results on the NVIDIA dataset. Sparse network fusion results on ChaLearn ConGD dataset are discussed in Section 5.2.3.

In addition to comparing three channel fusion strategies, this section also compares the results of FOANet with previous state-of-the-art results. In addition to fusing all channels, we will also analyze the performance of fusing a subset of channels to know the combinations that are capturing more information than others.

Chapter 4 discussed different temporal fusion strategies for isolated and continuous gesture recognition tasks. As there was no evident pattern on isolated gesture recognition datasets, average method is used as the temporal fusion strategy for isolated gesture recognition datasets in this chapter. As the best temporal fusion strategy depends on the type of data being processed for continuous gesture recognition datasets, we use the best temporal fusion strategy based on the channel for ChaLearn ConGD dataset (Section 5.2.3).

5.2.1 ChaLearn IsoGD Experiments

Training and Inference

ChaLearn IsoGD dataset consists of both one handed and two handed gestures. In some of the one handed gesture videos, only one hand is visible. In such cases, only 8 out of 12 channels are active. This doesn't pose a problem to average fusion method that doesn't involve training. However, concatenation and sparse network fusion methods always expect information from 12 channels. To be consistent across different fusion strategies, if a hand is never visible, the corresponding FC features and softmax scores for that channel are set to all zeros. This ensures that every video has information from all 12 channels.

In a one hand gesture video that has both hands visible, one hand will be performing the gesture whereas the other hand is always idle. The channels focusing on this idle hand can't classify the gesture and only adds noise to the classification. To prevent this, if the average movement of a hand is less than 4 pixels throughout the video, the corresponding FC features and softmax scores for that channel are set to all zeros.

For an arbitrary video v and channel c, softmax scores are calculated for every overlapping sliding window. These scores are averaged across the video, resulting in a single softmax vector per channel. The softmax vectors thus obtained are used in average fusion technique and sparse network fusion technique.

There is no additional training involved for average fusion. During inference time, the softmax scores from all channels are averaged together, resulting in a single softmax vector for a video. The argmax of the softmax vector is the predicted gesture label.

To learn the weights of the sparse network fusion layer, the softmax scores of different channels of training data are precomputed. The weights are then trained using the Adam optimizer [54] with a batch size of 32. The initial learning rate is set to 0.01 for first 10,000 steps, and is decreased to 0.001 till 20,000 steps and is further decreased to 0.0001. The training is stopped after 50,000 iterations.

During inference time, for an arbitrary video v and channel c, softmax scores are calculated at every timestep. These scores are averaged across the video, resulting in a single softmax vector. All the scores are stacked together and are multiplied by the fusion layer weights and the diagonal of the resulting matrix is extracted. The argmax of the diagonal is the predicted gesture label.

Results

The proposed method which is a combination of focus of attention and sparse network fusion achieves the state-of-the-art performance on the ChaLearn IsoGD dataset, as shown in Table 5.1. Table 5.1 also shows the top performing entries from the ChaLearn 2017 competition [118]. On the validation data we outperform the previous state-of-the-art by 16.5%, with an accuracy of 80.96% compared to the previous best of 64.4%. On the test set we achieve an accuracy of 82.07%, outperforming the previous state-of-the-art by 14.3%.

As already stated, focus of attention and sparse network fusion are the keys to our method. To evaluate the contribution of sparse network fusion, we replace it with average fusion, i.e. averaging the output of the softmax layers of the 12 channels. The average fusion version of FOANet achieves better results than previous methods (67.38% vs 64.40% on validation set and 70.37% vs 67.71%

Table 5.1: ChaLearn IsoGD 2017 results. Entries are ordered by their performance on test data. Results on systems other than ours were previously reported in [118].

System	Valid	Test
FOANet (Sparse fusion)	**80.96%**	**82.07%**
Miao *et al*. [77] (ASU)	64.40%	67.71%
SYSU_IEEE	59.70%	67.02%
Lostoy	62.02%	65.97%
Wang *et al*. [123] (AMRL)	60.81%	65.59%
Zhang *et al*. [136] (XDETVP)	58.00%	60.47%

on test set), as shown in Table 5.2. Therefore, sparse network fusion improves performance by 11.7%.

Another way to interpret this result, however, is that focus of attention channels are surprisingly powerful. The other entries in Table 5.1 use 3D convolutions and RNNs. Our approach with spatial attention channels outperforms these techniques using only 2D convolutions, averaging across time, and averaging across channels.

When concatenation is used to fuse channels, the resulting accuracy on ChaLearn IsoGD validation and test sets are 56.03% and 59.44% respectively. Interestingly concatenation method performed worse than averaging and its performance can be attributed to the fact that the method overfits (achieves a training accuracy of 96.32%).

To probe further, we applied averaging to all possible subsets of the 12 channels on validation set. With averaging as the fusion mechanism, the best performance was achieved by a subset of 7 of the 12 channels: 3 RGB flow channels, 2 depth focus channels, the RGB right hand channel, and the depth flow right hand channel. If we average these 7 channels together, the accuracy is 69.06% on the validation set and 71.93% on the test set, as shown in the 3rd row of Table 5.2. This is roughly 1.5% better than averaging all 12 channels, and suggests that 5 of the channels produce as much noise as information. We see a different pattern with sparse network fusion, however. By using only 7 channels with sparse network fusion, the accuracy decreases to 77.31% on the validation set and 78.9% on the test set. With sparse network fusion the system learns which

Table 5.2: Comparison of fusion strategies. Accuracies are shown for FOANet using sparse network fusion versus channel averaging, for 12 channels (maximal for sparse nets) and 7 channels (optimal for averaging). The numbers in bold represent the results that are better than the previous state-of-the-art.

Fusion	Valid		Test	
	12 Channels	7 Channels	12 Channels	7 Channels
Sparse Fusion	**80.96%**	**77.31%**	**82.07%**	**78.90%**
Average Fusion	**67.38%**	**69.06%**	**70.37%**	**71.93%**
Concatenation	56.03%	55.29%	59.44%	58.84%

Table 5.3: Individual channel accuracies on ChaLearn IsoGD validation and test set. These numbers are similar to the numbers in Table 3.2 and the third column (average of softmax) of Table 4.3.

	Validation Set			Test Set		
	Global	**Left**	**Right**	**Global**	**Left**	**Right**
RGB	33.22	23.41	41.76	41.27	19.55	47.44
Depth	27.98	34.40	55.12	38.50	28.29	64.48
RGB Flow	46.22	34.95	54.81	50.96	28.23	59.73
Depth Flow	31.66	31.62	48.51	42.02	26.70	58.83

channels to include for each gesture type, with the result that sparse network fusion benefits from the presence of channels that hurt performance when averaging channels.

Analysis of channels

Here we analyze the performance of various combinations of 12 channels of ChaLearn IsoGD to measure the contribution of different channel types to FOANet. Table 5.3 shows the accuracy of each channel on the IsoGD validation and test sets. This table is similar to the third column (average of softmax) of Table 4.3. However, the column in reshaped such that the rows show different modalities and columns show different focus of attention channels.

We combine different combinations of channels using sparse network fusion, as shown in Table 5.4. From the first two fusion columns, we can see that the combination of focus channels is better than the combination of global channels. In fact, the fusion of focus channels is the best combination, short of combining all channels. Moreover, most of the information from focus is contributed by the right hand alone which can be attributed to the right handed bias in the dataset. We also notice that the fusion of RGB and RGB flow nets is better than the fusion of depth and depth flow nets on validation set. However on the test set, depth + depth flow performed better.

Looking back at Table 5.3, we can see that “Depth Right" outperforms all other channels on the test set and that contributed to depth modality's overall performance. Next, we see that the fusion of RGB and depth channels performs on par with the fusion of RGB flow and depth flow channels. We also note that all of the columns in Table 5.4 except for the global column outperform the previous state-of-the-art which is 64.40% on validation set and 67.71% on test set.

	Validation	Test	Global	Focus	Right	RGB	Depth	Raw	Flow	All
RGB Global	33.22	41.27	✓			✓		✓		✓
RGB Left	23.41	19.55		✓		✓		✓		✓
RGB Right	41.76	47.44		✓	✓	✓		✓		✓
Depth Global	27.98	38.50	✓				✓	✓		✓
Depth Left	34.40	28.29		✓			✓	✓		✓
Depth Right	55.12	64.48		✓	✓		✓	✓		✓
RGB Flow Global	46.22	50.96	✓			✓			✓	✓
RGB Flow Left	34.95	28.23		✓		✓			✓	✓
RGB Flow Right	54.81	59.73		✓	✓	✓			✓	✓
Depth Flow Global	31.66	42.02	✓				✓		✓	✓
Depth Flow Left	31.62	26.70		✓			✓		✓	✓
Depth Flow Right	48.51	58.83		✓	✓		✓		✓	✓
Validation			61.4	**76.76**	**72.64**	**71.41**	**68.56**	**70.69**	**70.49**	**80.96**
Test			67.5	**77.61**	**74.46**	**75.41**	**76.39**	**75.29**	**74.39**	**82.07**

Table 5.4: Results of fusing different combinations of channels. 'Raw' refers to input from a stack of unprocessed images, whereas 'flow' refers to input of a stack of flow field images. The last column matches the first row of Table 5.1. Bold-face numbers represent results that are higher than the previous state-of-the-art (64.40% on validation set and 67.71% on test set). Note that all combinations involving focus channels beat the previous state-of-the-art.

5.2.2 NVIDIA Experiments

Training and Inference

The training and inference of different channel fusion strategies is similar to the training and inference for ChaLearn IsoGD dataset (See Section 5.2.1) except for one difference. As the NVIDIA dataset involves only one hand gestures, left hand is never visible. All videos has information from 8 channels and there is no necessity to zero out features/scores from hands that are idle or not visible.

Results

FOANet performance surpasses both the previous best result and human accuracy, as shown in Table 5.5. Our method achieves an accuracy of 91.28%, a 7.5% increase over the best previous result [82], and an increase of 8.9% over the best previous result not using IR data. FOANet even surpassed human level accuracy by 2.9%.

The accuracy of FOANet drops to 85.26% when sparse network fusion is replaced by average fusion, emphasizing the importance of sparse network fusion even in domains with only one hand and no significant background changes. However, the accuracy of 85.26% is still better than the previous SOA, reaffirming the importance of focus of attention channels. Concatenation method achieves a very low accuracy of 56.84% reaffirming the fact that the method overfits (training accuracy of 98.23%).

Table 5.5: Results on NVIDIA test set. The bold-face numbers represent results that are higher than previously reported results.

Method	Channels	Accuracy
FOANet	FOA + Sparse Fusion	**91.28**
FOANet	FOA + Avg. Fusion	**85.26**
Human	Color	88.4
Molchanov [82]	All (including IR)	83.8
Molchanov [82]	Depth + Flow	82.4

Similar to ChaLearn IsoGD dataset, we applied averaging to all possible subsets of the 8 channels. With averaging as the fusion mechanism, the best performance was achieved by combination of RGB flow and depth flow modalities on focus channels. The combination of these two channels gives an accuracy of 86.92%, as shown in the 3rd row of Table 5.6. This is roughly 1.7% better than averaging all 8 channels, and suggests that 6 of the channels produce as much noise as information. We see a different pattern with sparse network fusion, however. By using only 2 channels with sparse network fusion, the accuracy decreases to 88.38%. With sparse network fusion the system learns which channels to include for each gesture type, with the result that sparse network fusion benefits from the presence of channels that hurt performance when averaging channels. This

Table 5.6: Comparison of fusion strategies. Accuracies are shown for FOANet using sparse network fusion versus channel averaging, for 8 channels (maximal for sparse nets) and 2 channels (optimal for averaging).

Fusion	**Valid**	
	8 Channels	2 Channels
Sparse Fusion	**91.28%**	**88.38%**
Average Fusion	**85.26%**	**86.92%**
Concatenation	56.84%	72.26%

pattern is consistent with the results on ChaLearn IsoGD dataset as discussed in Section 5.2.1. However, unlike results in Section 5.2.1, the performance of concatenation fusion increased significantly (72.26%) when only 2 channels are fused. As previously discussed concatenation method overfits due to the large number of weights to learn. The number of weights that concatenation method learns is proportional to the number of channels to fuse. When only two focus channels of NVIDIA dataset are fused, concatenation method learns only 102,750 weights ($2055 \times 2 \times 25$) compared to 410,300 weights when all 8 channels are combined. As the weights are reduce by $\approx$ 25%, the performance of concatenation method increases significantly when only two channels are combined.

Analysis of channels

Here we analyze the performance of various combinations of 8 channels of NVIDIA dataset to analyze the contribution of different channel types to FOANet. We combine different combinations of channels using sparse network fusion, as shown in Table 5.7. The accuracies shown in second column are the individual channel accuracies using average softmax temporal fusion strategy (see Table 4.4). From the first two fusion columns, we can see that the combination of focus channels is better than the combination of global channels. In fact, the fusion of focus channels is the best combination, short of combining all channels. We also notice that the fusion of RGB and RGB flow nets is better than the fusion of depth and depth flow nets. Next, we see that the fusion of flow channels outperforms the fusion of RGB and depth channels. We also note that half of the combinations in Table 5.7 outperform the previous state-of-the-art. One difference between the combinations that outperformed previous state-of-the-art and the combinations that did not is the

	Test	Global	Focus	RGB	Depth	Raw	Flow	All
RGB Global	43.98	✓		✓		✓		✓
RGB Focus	58.09		✓	✓		✓		✓
Depth Global	66.80	✓			✓	✓		✓
Depth Focus	70.12		✓		✓	✓		✓
RGB Flow Global	62.66	✓		✓			✓	✓
RGB Flow Focus	77.18		✓	✓			✓	✓
Depth Flow Global	58.71	✓			✓		✓	✓
Depth Flow Focus	73.65		✓		✓		✓	✓
Test		72.82	**89.62**	**84.02**	79.46	75.10	**88.38**	**91.28**

Table 5.7: Results of fusing different combinations of channels on NVIDIA dataset. 'Raw' refers to input from a stack of unprocessed images, whereas 'flow' refers to input of a stack of flow field images. The last column matches the first row of Table 5.5. Bold-face numbers represent results that are higher than the previous state-of-the-art. Note that all combinations involving focus channels beat the previous state-of-the-art.

presence of RGB flow focus channel. This is the best performing individual channel and all the combinations that has this channel outperformed the previous state-of-the-art.

5.2.3 ChaLearn ConGD Experiments

The previous two subsections (5.2.1 and 5.2.2) evaluated sparse network fusion on two isolated gesture recognition datasets. As there was no relationship between channels and the performance of temporal fusion methods, the results in previous two subsections are based on using average as temporal fusion method. However, there is a clear relationship between temporal fusion methods and channels on continuous gesture recognition datasets as can be seen from the results in Section 4.2.

For continuous gesture recognition, we see that the best temporal fusion strategies in multi-channel networks depends on the modality (RGB vs depth vs flow field) and target (global vs left hand vs right hand) of the channel. More specifically, global channels perform best with *Gaussian Pooling*, flow field channels focused on the left or right hand perform best with *LSTMs*, and RGB and depth channels focused on the left or right hand perform best with *Late Fusion*. So, FOANet can be extended to temporally fuse different channels by different temporal fusion methods rather than having a single temporal fusion method for all channels.

We propose the Selective FOANet (S-FOANet) architecture, as shown in Figure 5.4. Although the figure looks complex at first, the extension lies inside the dashed red boxes in Figure 5.4. This is where information is fused over time. The surprise is that different types of channels are handled differently. Temporal information in global channels is fused using *Gaussian Pooling*, while focused RGB and depth channels are combined through *Late Fusion* and focused flow field channels are fused using *LSTMs* [45]. The motivation for using three different temporal fusion strategies is previously discussed in Section 4.2.

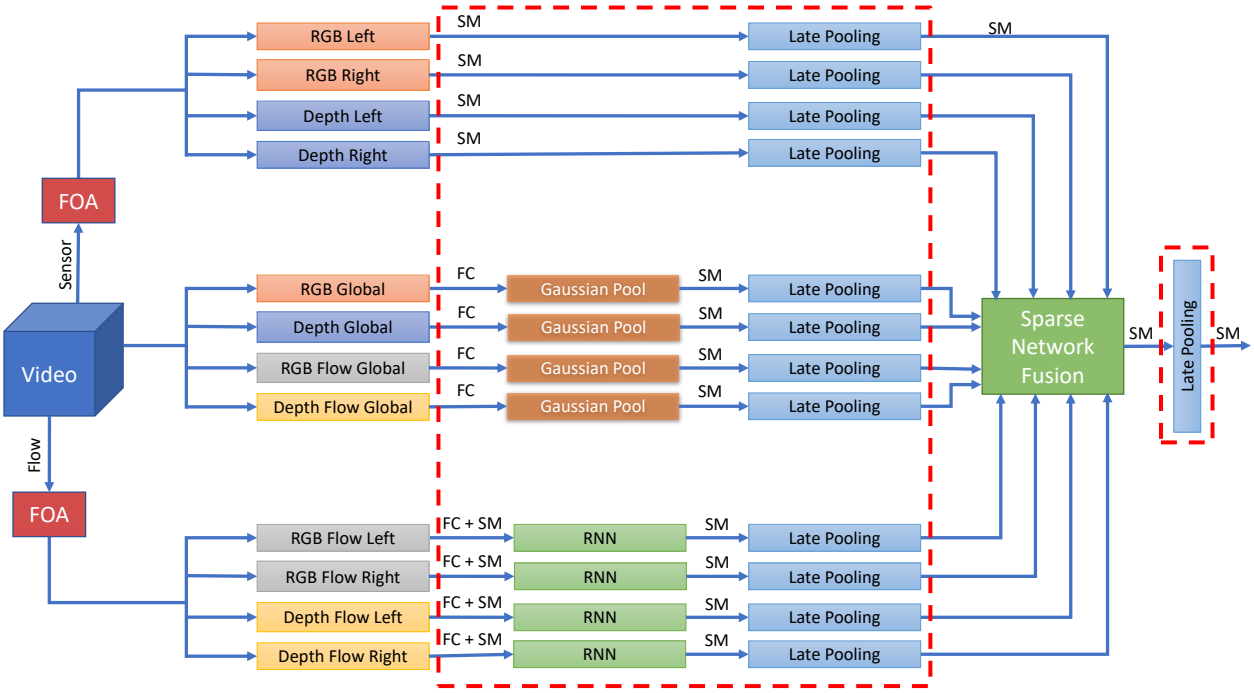

Figure 5.4: The S-FOANet Architecture. The parts inside the dashed red boxes are novel to S-FOANet. The red dashed box in the middle shows how information is fused over time within channels. The surprise is that global channels are fused differently from local channels, and local RGB and depth channels are fused differently from local flow field channels. S-FOANet also adds a final late pooling module to improve temporal coherence among the final labels.

Results

S-FOANet achieves state-of-the-art performance on the ChaLearn ConGD dataset, as shown in Table 5.8. On the validation data, S-FOANet outperforms the previous SOA with a mean Jaccard

Table 5.8: ChaLearn ConGD 2017 results. Entries are ordered by their performance on test data. Results on systems other than ours were previously reported in [118].

System / Data	Selective FOA	ICT_NHCI [69]	AMRL [124]	PaFiFA [12]	DeepGesture [92]
Validation Set	**0.7791**	0.5163	0.5957	0.3646	0.3190
Test Set	**0.7740**	0.6103	0.5950	0.3744	0.3164

Table 5.9: Mean Jaccard Index scores of temporal fusion on the validation and test sets of ChaLearn ConGD. S-FOANet combines the best temporal fusion strategies for each type of channel: Gaussian Pooling for global channels, Late Pooling for raw focused channels, and LSTMs for focused flow field channels.

	Valid				Test			
	Late Pooling	Gaussian Pooling	LSTM	Selective FOANet	Late Pooling	Gaussian Pooling	LSTM	Selective FOANet
Jaccard Index	0.742	0.7387	0.765	**0.7791**	0.7481	0.7036	0.7615	**0.7740**

Index of 0.7791, compared to the previous best of 0.5957. On the test set it achieve a mean Jaccard Index of 0.7740, outperforming the previous state-of-the-art of 0.6103.

Table 5.9 shows the results of fusing information across all 12 channels. The Late Pooling, Gaussian Pooling and LSTM columns show the result of using a single temporal fusion mechanism across all channels. The S-FOANet column show the result of using Gaussian Pooling on global channels, Late Pooling on raw focused channels, and LSTMs on focused flow field channels. Not surprisingly given the results in Table 4.1, the best result is achieved by selecting the fusion mechanism based on the channel type - using Gaussian Pooling for global channels, LSTMs for focused (left hand or right hand) flow field channels, and late Pooling for focused RGB and depth channels.

5.3 Summary

The framework introduced in Chapters 3 and 4 leads to 12 global and focused channels (8 for datasets that has only one hand visible). This chapter discussed different strategies to fuse information from multiple channels. A main contribution of this thesis, sparse network fusion, is discussed in this chapter and the experimental results showed the effectiveness of this fusion strategy.

We showed that when spatial channels are focused on the hands, gesture recognition improves significantly, particularly when the channels are fused using a sparse network. Using this technique, we improved performance on the ChaLearn IsoGD dataset from a previous best of 67.71% to 82.07%, and on the NVIDIA dynamic hand gesture dataset from 83.8% to 91.28%, and on the ChaLearn ConGD dataset from 0.6103 to 0.7740.

The experiments also showed that concatenation based strategy fails to scale and overfits with the increase in number of channels to be fused. Although, averaging doesn't involve any additional training, it outperformed concatenation strategy every time.

Chapter 6

Analyzing Multi-Channel Networks for Gesture Recognition

The FOANet architecture proposed in this thesis divides processing by modality and spatial attention, and information from these channels is combined by sparse network fusion (see Chapter 5). This multi-channel architecture achieves state-of-the-art results on multiple gesture recognition datasets. Moreover, the gain in performance over the previous state-of-the-art reported in this thesis is not small: the error rates are almost halved on all datasets.

FOANet achieved these impressive results by following the general trend of multi-channel architectures in gesture recognition, where two or more convolutional networks process different versions of the same video in parallel [22, 23, 25, 51, 80, 82, 91, 107]. Most of these methods divide processing into two to five channels. FOANet pushed this trend to extremes by dividing the processing into 12 channels, and the results of FOANet show the advantage of the 12 channel architecture. However, the justification for the 12 channel architecture is not immediately obvious. On one extreme, it is easy to encode RGB data, depth data, flow field vectors and more as bands of input images, allowing all the same information to be fed to a single channel, where the data can be combined without restrictions. On the other extreme, the number of channels could be increased beyond 12 by dividing the data into multiple subsets and training 12 channels on each subset.

Performance improvements are also evident in other multi-channel architectures where multi-channel approaches outperform single channel methods. But there is no clear explanation as to why. This is in line with the field of deep learning in general where the practice is ahead of the theory. Recently, Feichtenhofer *et al*. visualized the spatiotemporal representations learned by deep two-stream networks for action recognition [24]. Their work focuses on cross-stream fusion between two channels and shows that cross-stream fusion enables the learning of spatiotemporal features. However, most multi-channel networks are trained independently without cross-channel

interactions and with fusion performed at the FC or softmax levels. In this chapter, we focus on the information learned by different channels of FOANet where channels are trained independently.

More specifically, we consider two possible explanations. One is what we call the *Bagging* hypothesis. It is motivated by the well-known result that strong classifiers can be created by averaging (a.k.a. bagging) the results of many weak classifiers, as long as the weak classifiers are unbiased [5]. The idea is that every channel is a weak classifier, and the performance of the multi-channel system is improved by the bagging effect. In this model, every channel solves essentially the same problem, and the role of the different modalities and/or attention targets is to keep the channels from being too strongly correlated. The implications of the Bagging hypothesis are that (1) more channels will always be better than fewer, and (2) channel outputs can be combined by averaging, as in [82, 123, 136]. The alternative explanation is what we call the *Society of Experts* (SoE) hypothesis, named in the spirit of Marvin Minsky [79]. The SoE hypothesis asserts that channels specialize to the modalities and/or attention targets they are assigned to. Each channel specializes in particular sources of information and therefore becomes better at recognizing some gesture classes at the expense of being worse at others. The implications of the SoE hypothesis are that (1) adding a channel is only beneficial if it adds a new source of information, and (2) the outputs of channels should be combined selectively, usually through trained fusion networks as in [25, 51, 77].

The Bagging and SoE hypotheses are actually two points along a spectrum of possible models. In the bagging hypothesis, all channels share similar expertise and the difference between them is estimation variance. In the SoE hypothesis, channels have unique expertise. In between are models with varying degrees of overlapping expertise among channels. Most real systems probably lie somewhere between the two extremes, particularly systems with cross-stream fusion [22, 23, 25]. Nonetheless, it is important to know whether bagging or SoE is the closer model because of the architectural implications. To the extent that systems are bagging, we should add as many channels as possible. We could even have multiple channels processing the same data stream, as long as some other technique (e.g. data partitioning) is used to make sure the channels are not strictly

redundant. If systems are societies of experts, on the other hand, we should only add channels for strong new sources of information, and we should train fusion networks instead of just averaging results.

We explore the Bagging and SoE hypotheses in the context of the 12-channel FOANet architecture on the ChaLearn IsoGD data set [119]. FOANet has more channels than any other architecture we know of and divides channels by both modality and focus of attention target. This chapter first looks at how channels respond to properties of their input data. For example, it measures how the performance of a channel degrades when high frequency information is removed by pre-processing the input videos with a low-pass filter. It also looks at how channel performance degrades when color or motion information is removed. The Bagging hypothesis suggests that channels should exploit all the data available to them, and therefore their performance should degrade similarly when information is removed. The SoE hypothesis, on the other hand, suggests that channels specialize and exploit only specific aspects of the input signal. Therefore some channels should degrade more severely than others when a specific kind or source of information is removed.

This chapter also measures specialization with regard to gesture properties. Some gestures, for example, are one-handed, while others are two-handed. Similarly some gestures involve large arm motions, while others involve localized hand or finger motions and others are defined by static poses. The SoE hypothesis predicts that channels specialize and should be more useful for gestures with some properties than others. The Bagging hypothesis, on the other hand, predicts that channels are generalists and their performance should not be tied to specific gesture properties.

In both sets of experiments, we find evidence that FOANet channels are highly specialized, and that they often specialize in explainable and interesting ways. For example, RGB channels specialize in a way that is analogous to differences between human foveal and peripheral vision (see Section 6.1.2). Both sets of experiments can therefore be interpreted as preferring the SoE hypothesis, at least in the context of FOANet.

6.1 Relating Channels to Data Properties

This section investigates the extent to which channels specialize in the information they extract from videos. The methodology can be thought of as a data ablation study. FOANet channels are trained on RGBD videos from the ChaLearn IsoGD data set. In each experiment we remove a source of information from the videos and measure the drop in performance for each channel. For example, in one experiment we remove high-frequency information, while in another we remove color. The Bagging hypothesis suggests that channels consume all the information available to them, so the relative performance drop when a source of information is removed should be similar. The SoE hypothesis, on the other hand, predicts that channels specialize in the information they exploit, so some channels should degrade much more than others when a source of information is removed.

6.1.1 FOANet Channels

FOANet has 12 independent channels, one for every combination of modality and attention focus targets. Specifically, the 4 modalities of RGB, depth, RGB flow and depth flow are crossed with 3 focus of attention selections: attention to the entire video frame (global), attention focused on just the right hand, and attention focused on just the left hand. The modality of a channel determines the type of input. RGB and depth channels consume 3 band RGB and depth images respectively. Flow field channels also consume 3 band images, but the bands are *dx, dy,* and *magnitude*. The difference between RGB flow and depth flow is whether the flow fields are computed from RGB images or depth images.

The focus of attention targets in FOANet are less standard. Whereas Karpathy *et al*. positioned their attention window in the middle of the scene, FOANet actively detects and tracks the positions of the left and right hands throughout the video. The left and right hand channels (collectively, the focused channels) are created by selecting an image window that bounds the estimated position of the hand [67, 68]. The image window is extracted from the RGB, depth and flow field images and re-scaled to 128×128 pixels.

Every channel in FOANet has access to motion information. FOANet channels analyze 10-frame temporal sliding windows stacked together as 30 bands, similar to Simonyan and Zisserman [107]. Flow field channels therefore have potential access to acceleration data, i.e. changes in motion over time.

6.1.2 Methodology

To understand how performance degrades when a source of information is removed, we look at a channel's response to every 10-frame temporal window. The ChaLearn data set has 249 gesture classes, so channels produce a 249 element softmax vector in response to every input window. To establish a baseline for each channel, we measure the accuracy of the argmax of this vector on the ChaLearn IsoGD validation and test videos. We then alter the validation and test videos to remove a source of information and measure the relative drop in performance. A total of three experiments are run, removing three different sources of information: motion, high spatial frequencies, and color. The experiment using color is restricted to RGB channels.

Removing Motion

FOANet channels classify windows of 10 consecutive frames stacked together as a 30 band image. These windows capture motion information over a 10-frame interval. We remove motion information by taking the middle frame of the window and replicating it 10 times as shown in second row of Figure 6.1.

Removing High Frequency Information

High frequency information is removed from RGB and depth images by convolving them with a Gaussian mask of sigma 3. RGB flow and depth flow images are computed from the smoothed RGB and depth images. Right-hand and left-hand channels are extracted and rescaled from the smoothed images or from the flow fields extracted from smoothed images. The third row of Figure 6.1 shows an arbitrary RGB window after high frequency information is removed.

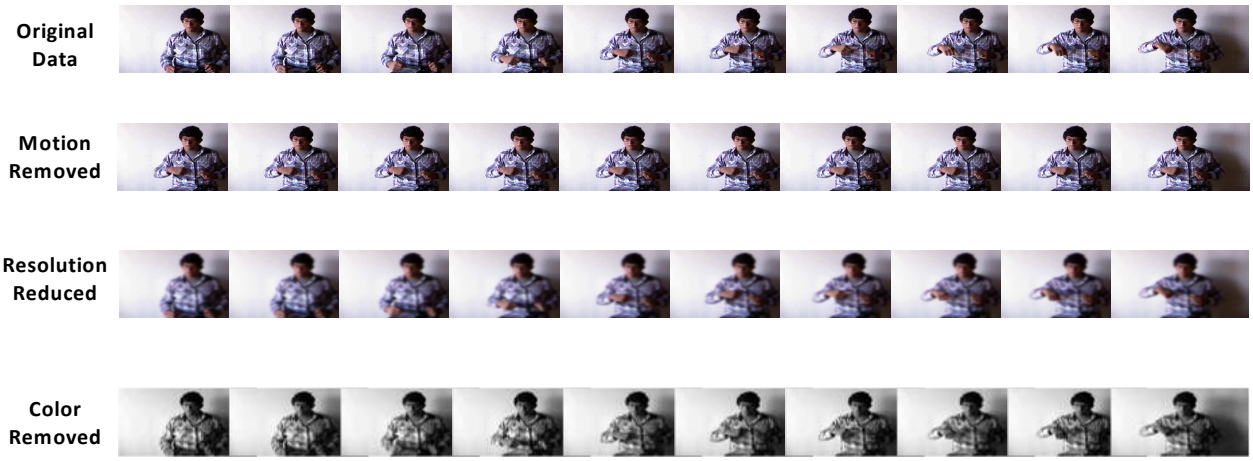

Figure 6.1: Removing different sources of information. The first row shows an arbitrary 10 frame sliding window. The second row shows the resulting window after motion is removed (fifth frame is replicated). The third row shows the sliding window after high frequency information is removed. The last row shows the sliding window in gray scale (color information is removed).

Removing Color

Color information is removed from the inputs to RGB channels by transforming RGB images to gray-scale images. As the other modalities (depth, flow fields) do not have color bands, color information is only removed for RGB channels. Last row of Figure 6.1 shows an arbitrary window after removing color information.

6.1.3 Results

Figure 6.2 shows the results of data ablation on the RGB global and RGB right hand channels. The vertical axis is channel accuracy, measured relative to baseline performance on unaltered data. The gray bar, which represents baseline performance, is therefore always 100% for all channels. The blue bars show the performance of the RGB global channel when motion (left), high frequency (middle) and color (right) data is removed. The orange bars show the same for the RGB right hand channel.

In Figures 6.2 through 6.4, data is shown for the global and right hand channels, but not the left hand channel. This is to simplify the figures, because the relative performance of the left hand channel mimics the relative performance of the right hand channel. In Section 6.2, the performance of the left and right hand channels will be broken out.

The most striking result in Figure 6.2 is shown on the right. When color information is removed, the accuracy of the RGB right-hand channel is cut roughly in half. Its performance drops by 53.9%, while the performance of the left hand channel (not shown) is similar and drops by 54.01%. The RGB global channel, on the other hand, is almost unaffected. It drops by less than 7%. This suggests that the right and left hand channels depend on color information, but the global channel does not.

The middle of Figure 6.2 shows the relative performance of global and focused channels when high frequency information is removed. This time, the performance of both the global and the focused channels drop significantly, suggesting that both rely on high frequency information. They do not rely on it to the same extent, however. The performance of the global channel drops by half (45.4%), but the right hand channel is even more strongly effected. Its performance drops by 80.99% (the left hand drops by 81.45%). Thus while all channels depend on high frequency information, the right- and left-hand channels rely on it more than the global channel does.

The left most comparison between global and right-hand channels in Figure 6.2 reveals a dependence upon motion that is the opposite of that seen for color and resolution. When motion information is removed the performance of all channels drops, but this time the right and left-hand channels drop by less than half (38.7% and 40.63% respectively), while the performance of the global channel drops by over 80%. This suggests that the global channel is more concerned with motion than the focused channels are.

There is an interesting analogy here to foveal and peripheral vision. The right and left hand channels are both focused on a target, and they learn to rely on color and high frequency information, while making less use of motion, much like human foveal vision. The global channel looks at the whole scene and primarily exploits motion information, while making less use of color and high-frequency information, much like human peripheral vision. Obviously, human visual specialization is dictated by the anatomy of the eye [93]. The fovea contains densely-packed cones providing high-frequency color information to ganglion cells that largely feed the slow but color-sensitive α-channel in LGN. The peripheral retina contains mostly loosely-packed rods connected

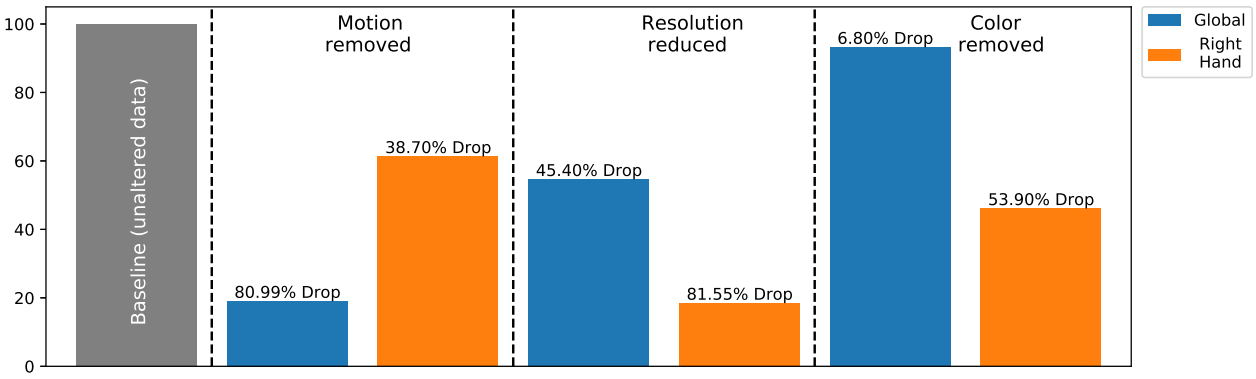

Figure 6.2: Bar chart showing the result of data ablation on RGB modality. The vertical axis is relative accuracy, compared to baseline performance on unaltered data. Blue and orange bars show the accuracy of the global and right-hand channels respectively. The pairs of blue and orange bars show relative accuracy after motion information is removed (left), high frequency information is removed (middle), and color information is removed (right).

to ganglion cells that feed the faster-responding β-channel. In contrast, all channels in FOANet receive the same types of information. It is interesting to speculate, however, whether the eye evolved the way it did because focused vision demands high-frequency color information, while peripheral vision concentrates on motion information.

Figure 6.3 shows the results of data ablation on depth channels. When motion information is removed, the relative performance of the global depth channel drops by 41.82%, while the drop in performance for the right-hand depth channel is similar at 36.77% (The left hand drops by 38.21%). This suggests that all depth channels are sensitive to motion information, but none rely on it as their primary source of information. When high frequency information is removed, on the other hand, the relative performance of the global depth channel drops by only 25.70%, whereas the relative performance of the right hand depth channels drops by 77.31% (left: 80.66%). The focused depth channels seem to rely on high frequency information, whereas the depth global channel can make do with low frequency information.

Figure 6.4 shows the results of data ablation on flow field channels. There are two types of flow fields: those computed on RGB images (shown in solid colors) and those computed from depth images (shown with cross-hatching). When only a single flow field is provided (instead of a window of 10), global flow field channel performance drops by 80.14% for depth flow and 86.88%

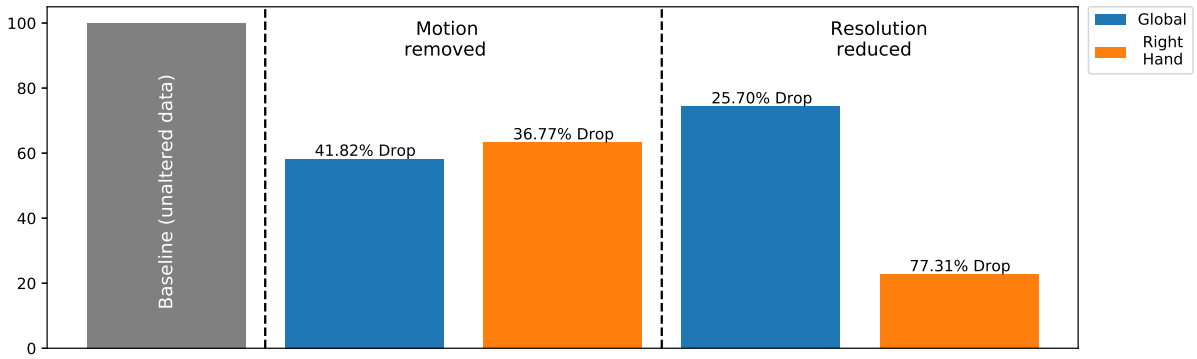

Figure 6.3: Bar chart showing data ablation results on the depth modality with motion removed and resolution reduced. The figure formatting otherwise matches that used in Figure 6.2.

for RGB flow. The right hand flow channels performance drops by 66.87% and 68.15%. This is somewhat counterintuitive, since even one frame of a flow field represents motion information. Apparently, these channels rely on multiple frames of flow field data, either to smooth noise or to calculate accelerations. When high frequency information is removed, on the other hand, global channel performance drops by 35.99% and 41.77%, while right-hand channel performance drops by 42.10% and 48.45%. Flow channels are not as sensitive to high frequency information as they are to multi-frame motion.

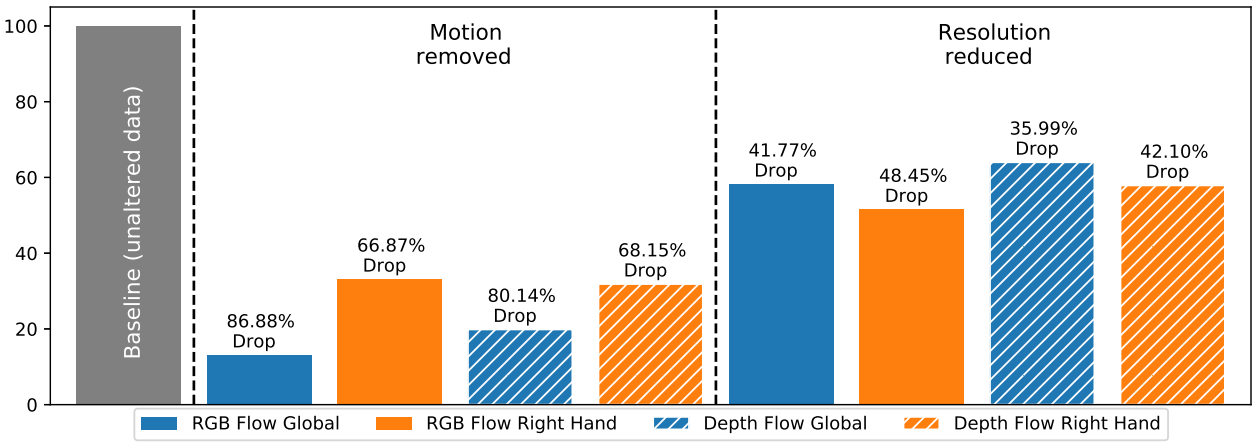

Figure 6.4: Bar chart showing the results of data ablation on flow field channels. Solid bars represent flow field channels computed from RGB images, while dashed bars represent flow fields computed from depth images. The first four bars show the result of removing motion information (i.e. using only a single flow field, not a window of 10). The next four show the result of removing high frequency information.

Table 6.1: Information favored by different channels of FOANet

	RGB	**Depth**	**Flow**
Global (Whole Frame)	Motion	Low Frequency	Motion
Focus (Left/Right Hand)	High Frequency and Color	High Frequency	Motion

Table 6.1 summarizes the results of Figures 6.2 through 6.4. Global RGB channels favor motion information whereas RGB focus channels (i.e. right hand and left hand channels) favor high frequency and color information. Global depth channels prefer low frequency information and focused depth channels prefer high frequency information. Both global and focus channels of flow fields capture motion information. We can see clear patterns of degradation that correspond to modality and attention target. This suggests that channels specialize and capture different information, leading us to prefer the SoE hypothesis over the Bagging hypothesis as an explanation of multi-channel behavior.

6.2 Relating Channels to Gesture Properties

Section 6.1.2 suggests that FOANet channels specialize in the sources of information they exploit, but does this specialization cause channels to be better at recognizing some types of gestures than others? To answer this question, we inspect the fusion layer of FOANet. For every gesture class, the fusion layer learns 12 weights, one per channel. These weights give the relative importance of a channel to a gesture class. The fusion layer weights can be examined to relate channels to gesture properties by seeing if some channels are better at recognizing some gesture types over others.

The methodology employed here is to have volunteer raters assign a binary (yes/no) label to all 249 ChaLearn IsoGD gestures classes for each property being examined. Then, to see whether a channel is tuned to a property, the fusion layer weights are compared for the gestures with the property versus gestures without the property. Statistically significant differences in weights tells us that the corresponding channel (a combination of modality and attention) is important for recognizing gestures with the associated property.

Table 6.2 shows the mean weights by channel for gestures with and without significant arm motions (Table 6.2a), with and without significant motion of the hand or fingers (Table 6.2b) and with and without a significant static characteristic pose (Table 6.2c). Within each of these tables are twelve columns, one for each channel, labeled by modality and attention target. In each column, the first row is the average weight for gestures with arm motions, and the second row is the average weight for those without. The third row is the signed difference between the means, while the fourth row is the p-value from Student's t-test, indicating whether the difference is statistically significant.

Without exception, flow fields channels are more important to gestures with arm motions than to those without. This confirms our intuition that flow field channels specialize in motion information. At the same time, the lack of a significant difference for the other channels suggests that although RGB and depth channels are given access to 10-frame image windows their performance in not impacted (negatively or positively) by large motions.

Smaller hand and finger motions are slightly different. Flow fields computed from RGB images are important for detecting hand motions, but flow fields computed from depth images are not. We presume the depth sensor lacks the resolution to resolve such fine motions. Static poses, on the other hand, create an opposite scenario from arm motions. Flow field channels, whether from RGB or depth images, have significantly lower weights for gestures with static poses, meaning that flow field channels are less important to gestures with static poses than for those with movement. Once again, the RGB and depth channels are unaffected.

Table 6.3 is similar in layout to Table 6.2 and presents weights for two additional properties. Table 6.3a shows the difference in fusion weights between one handed and two handed gestures. Since most ChaLearn participants are right-handed, one-handed gestures are usually performed with the right hand only. Examining the difference in weights in Table 6.3a reveals that left-handed channels are discounted for one-handed gestures. This makes sense, since the left hand is not used in these cases. More interestingly, the RGB global and RGB right-hand channels are more important for one-handed gestures than two-handed gestures. The same is true for the right-hand

Table 6.2: Fusion network weights compared for gestures with and without specific properties. Significantly larger weights for a modality attention combination suggest the combination is relied upon for recognizing gestures with the associated attribute.

Weights comparison summary with and without Global Motion, i.e. arm motions												
Modality →	**RGB**			**Depth**			**RGB Flow**			**Depth Flow**		
Attention →	**Global**	**Right**	**Left**	**Global**	**Right**	**Left**	**Global**	**Right**	**Left**	**Global**	**Right**	**Left**
Arm Motion	5.361	5.420	1.575	4.795	8.249	1.459	6.951	8.038	0.043	5.178	7.591	2.536
No Arm Motion	4.886	5.356	0.779	4.564	7.430	-0.630	5.360	6.412	-4.251	3.522	5.006	-1.434
Difference	0.475	0.064	0.796	0.232	0.819	2.089	1.591	1.626	4.294	1.656	2.525	3.970
P Value	0.385	0.911	0.561	0.709	0.117	0.116	0.002	0.006	0.004	0.008	0.000	0.004

(a)

Weights comparison summary with and without Local Motion, i.e. hand and finger motions												
Modality →	**RGB**			**Depth**			**RGB Flow**			**Depth Flow**		
Attention →	**Global**	**Right**	**Left**	**Global**	**Right**	**Left**	**Global**	**Right**	**Left**	**Global**	**Right**	**Left**
Hand Motion	4.842	5.454	-0.792	4.401	7.791	0.720	6.999	9.214	3.364	4.351	6.815	2.064
No Hand Motion	5.136	5.361	-3.084	4.725	7.737	0.032	6.915	6.744	0.444	4.115	5.833	-0.440
Difference	-0.293	0.093	2.292	-0.324	0.054	0.688	1.308	2.470	2.920	0.236	0.981	2.505
P Value	0.649	0.890	0.267	0.657	0.930	0.660	0.032	0.000	0.039	0.751	0.129	0.127

(b)

Weights comparison summary with and without Static Poses, i.e. fix pose briefly held fixed												
Modality →	**RGB**			**Depth**			**RGB Flow**			**Depth Flow**		
Attention →	**Global**	**Right**	**Left**	**Global**	**Right**	**Left**	**Global**	**Right**	**Left**	**Global**	**Right**	**Left**
Static Pose	5.051	5.171	-4.135	4.728	7.477	-0.454	5.026	5.896	0.258	3.415	4.794	-1.386
No Static Pose	5.090	5.587	-1.059	4.582	8.014	0.807	6.910	8.650	1.901	4.902	7.276	1.576
Difference	-0.039	-0.416	-3.076	0.146	-0.537	-1.261	-1.884	-2.754	-1.643	-1.487	-2.482	-2.962
P Value	0.943	0.455	0.072	0.810	0.293	0.331	0.000	0.000	0.018	0.015	0.000	0.029

(c)

Table 6.3: Fusion network weights compared for gestures: a) using only one hand versus two hands, and b) with significant movements along the optical axis.

Weights comparison summary with only one hand versus two hands												
Modality →	**RGB**			**Depth**			**RGB Flow**			**Depth Flow**		
Attention →	**Global**	**Right**	**Left**	**Global**	**Right**	**Left**	**Global**	**Right**	**Left**	**Global**	**Right**	**Left**
One Hand	5.757	6.152	-7.752	5.054	8.615	-3.933	6.285	7.642	-2.676	4.281	6.200	-3.957
Two Hand	4.114	4.307	4.636	4.096	6.541	5.924	5.554	6.798	6.338	4.009	5.841	5.786
Difference	1.643	1.844	-12.387	0.958	2.075	-9.857	0.732	0.844	-9.014	0.272	0.359	-9.743
P Value	0.002	0.001	0.000	0.118	0.000	0.000	0.156	0.121	0.000	0.663	0.510	0.000

(a)

Weights comparison summary with gestures with and without significant motion toward/away from the camera												
Modality →	**RGB**			**Depth**			**RGB Flow**			**Depth Flow**		
Attention →	**Global**	**Right**	**Left**	**Global**	**Right**	**Left**	**Global**	**Right**	**Left**	**Global**	**Right**	**Left**
Depth - Yes	6.217	6.545	-1.216	5.664	9.078	-2.550	7.314	7.911	2.967	3.645	6.778	3.550
Depth - No	4.908	5.216	-2.772	4.510	7.560	0.573	5.790	7.201	1.666	4.242	5.946	0.918
Difference	1.309	1.330	1.555	1.154	1.518	-3.123	1.524	0.710	1.301	-0.597	0.831	2.632
P Value	0.105	0.114	0.549	0.208	0.049	0.044	0.047	0.384	0.212	0.522	0.307	0.256

(b)

depth channel. Our guess is that fine hand details are more important in one-handed gestures than two-handed gestures.

Table 6.3b shows the weights for gestures with or without motion toward or away from the camera. It shows that the right-hand and left-hand depth channels and the global RGB flow field channel are important for motion along the optical axis. It surprises us that the focused RGB flow field channels are not significant for this task, and none of the depth flow field channels are significant. We note, however, that unlike in previous cases the p values are close to our significance cutoff threshold of 0.05, suggesting that this result is not as strong as the others.

6.3 Summary

The literature suggests that multi-channel architectures outperform single channel ones, but does not explain why. After all, the same information can be packed into a single multi-band input, allowing a single-channel architecture to combine information across modalities without restrictions. This chapter compares two possible explanations for the success of multi-channel systems. The Bagging hypothesis suggests that the benefit is from averaging the results of unbiased classifiers. In this model, every channel is general, and the goal of differentiating the channel inputs is to prevent redundancy. The Society of Experts (SoE) hypothesis suggests that channels specialize, with each channel becoming expert at a different aspects of the data. In this model, the benefit comes from selectively fusing information across experts. We explore these hypotheses in the context of FOANet, a 12-channel architecture with state of the art performance on the ChaLearn IsoGD data set.

We find evidence that the SoE hypothesis is a better description of FOANet than the Bagging hypothesis. By altering the input streams, we show that channels specialize in the sources of information they pay attention to. For example, the global RGB channel favors motion information, while the left hand and right hand RGB channels prefer high-frequency color information. By analyzing fusion weights, we show that channels specialize to gesture properties. For example, one-handed gestures discount left-hand channels entirely while emphasizing RGB data and focused

depth data over global depth data or flow field data. Collectively, our experiments suggest that channels specialize to specific sources of information, making them more or less relevant to specific types of gestures.

Chapter 7

Covert Attention

7.1 Introduction

The previous four chapters developed FOANet, a system that achieves state-of-the-art performance on gesture recognition tasks by expanding on the theme of multi-channels architectures. FOANet divides processing by modalities and spatial attention regions and fuses information from all channels using sparse network fusion. Chapter 3 introduced the idea of channels based on spatial focus of attention, Chapter 4 discussed various temporal fusion methods within these channels, and Chapter 5 addressed the cross-channel fusion issues created by creating new spatial attention channels. Chapter 6 analyzed some properties of the resulting channels and system.

Having extended the state-of-the-art by expanding the number and use of channels, t“““““““‘ The rest of this chapter is organized as follows. Section 7.2 reviews the human attention mechanism. Section 7.3 explains the architecture of CANet. Experimental results of CANet on ChaLearn IsoGD dataset are presented in Section 7.4. Finally, Section 7.5 summarizes and concludes the chapter.

7.2 Human Attention

The two stream architecture of Simonyan and Zisserman was originally motivated by the human vision system [106]. Early human vision has what are commonly called α and β channels [78]. The α channel begins with fast responding $P\alpha$ ganglion cells that project to the magnocellular layers of LGN; this pathway responds transiently to motion. The β channel begins with slower responding $P\beta$ ganglion cells that project to parvocellular layers in LGN. The β pathway responds to appearance more than motion. At a conceptual level, therefore, the two-stream architecture by Simonyan and Zisserman was inspired by human vision.

In addition to α and β channels, human vision also makes extensive use of spatial attention. Human sensory systems are presented with vast numbers of stimuli at every moment. However, only some of this stimuli is important. To efficiently process the information, human vision 'attends' to the important stimuli and ignores the rest using selective spatial attention. Thus, selective attention amplifies or intensifies the important information to facilitate action or perception [50].

Human vision attends to important regions of the scene by a combination of overt and covert attention. Overt attention selects locations through gaze, i.e. by moving the eyes, head and shoulders. Covert attention is the act of selecting data within the visual field. There is evidence that people can attend to up to 4 or 5 locations at a time [99, 103, 104]. Unlike with motion, however, attention is not implemented through separate neural channels. Instead, attention is implemented through top-down excitation of neurons corresponding to specific spatial locations within the α and β channels.

There are at least two explanations for why nature does not use separate channels for each attention region. One is efficiency. It is more efficient to process information using a single channel rather than having separate channels for every attention region. The second is information fusion. There is complementary information in different spatial regions and this information can be exploited when processing different attention regions together.

7.3 CANet

Inspired by human vision, the proposed CANet (Covert Attention Net) architecture merges attention channels while preserving the concept of spatial attention. As the attention channels are merged together, CANet has only 4 CNNs compared to 12 CNNs in FOANet. The four CNNs in CANet process four different modalities (RGB, Depth, RGB Flow, Depth Flow). The CNNs of CANet process a modality and output three softmax scores - one for each attention region (global, left hand, and right hand). As CANet merges and processes all attention channels of a modality using a single CNN, it has fewer weights and is more computationally efficient than FOANet. Moreover, the training time and inference time of CANet are comparatively less than

for FOANet ($\approx$ 70% of FOANet). CANet also exploits the complementary information among attention regions, resulting in increased accuracy. Figure 7.1 shows the CANet architecture for the RGB modality. The architecture for other modalities looks similar and sparse network fusion finally merges information from all modalities.

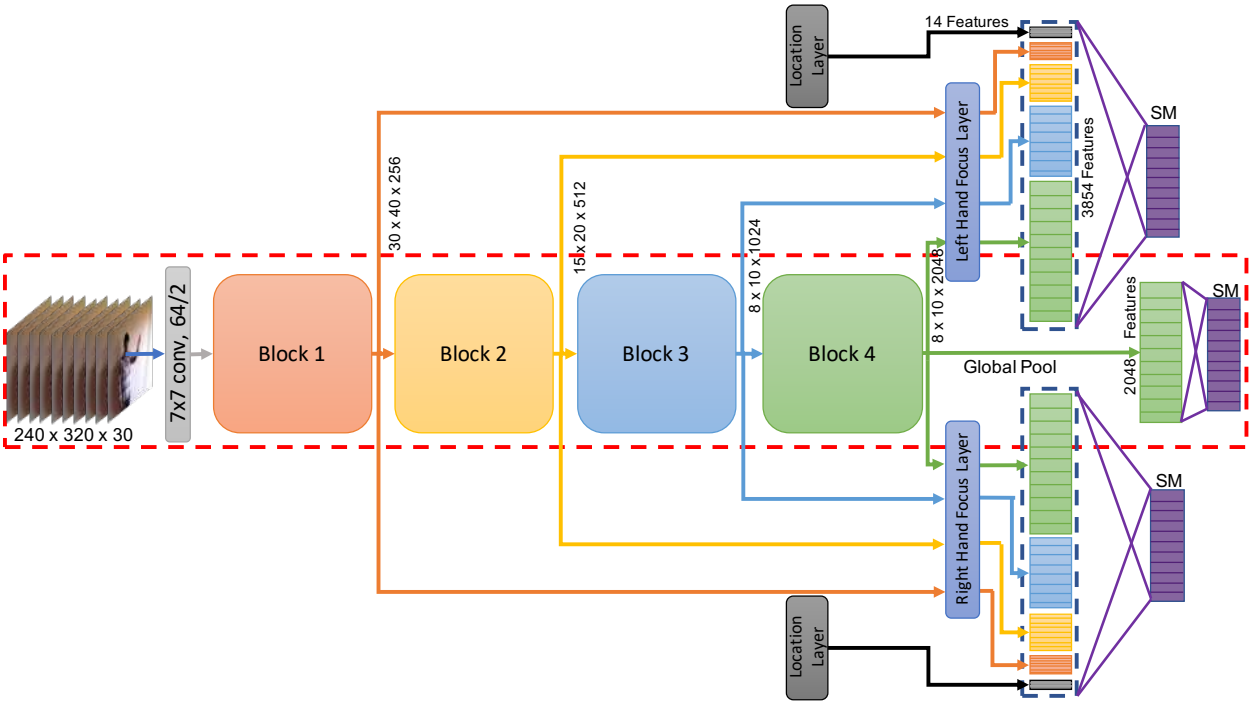

Figure 7.1: CANet Architecture for RGB modality. The input to the network is a stack of 10 images resulting in a 240 $\times$ 320 $\times$ 30 volume. The input volume is passed through ResNet-50 convolution and pooling layers. The last feature map of 8 $\times$ 10 $\times$ 2048 is global pooled resulting in 2048 global features. In addition, the feature maps after each block of ResNet are passed through right hand and left hand focus layer, where the feature maps are multiplied with respective hand masks and then global pooled to get the features that are focused on hands. Hand features from all blocks of ResNet are concatenated along with 14 location features resulting in 3854 features for each hand. These three features (global, left hand, right hand) are classified by three fully connected layers producing three vectors of softmax scores. The network inside the red dashed box is the global channels of FOANet.

The CANet architecture shown in Figure 7.1 looks similar to the global channels of FOANet with additional layers. The network in the red dashed box of Figure 7.1 is in fact the FOANet global channel that produces a softmax vector based on the features from entire field of view (FOV). In addition, CANet has two additional fully connected classification layers that classify gestures based on features from right hand and left hand, as shown in Figure 7.1. In total, CANet has three

classification layers that classify the same gesture based on different spatial attention regions. The global classification layer classifies the gestures by taking features from all spatial regions into account whereas the focus classification layers (right hand and left hand) classifies gestures by considering spatially localized features. The focus layers allow the CANet to focus attention on hands without having dedicated attention channels. As CANet is simultaneously attending to three attention regions using a single network (unlike FOANet that uses 3 independent networks), CANet is optimizing a more complex function using only $\approx \frac{1}{3}rd$ of FOANet parameters.

Focus layers are an important part of the CANet architecture. Focus layers of CANet zero out features that don't correspond to the left or right hand, allowing CANet to focus on hands without needing dedicated channels. In addition to the ten frame sliding window, two binary masks (one for each hand) are provided as input to CANet. These masks have ones in the bounding box of the hand and zeros elsewhere and correspond to the attention windows used by FOANet. These binary hand masks are resized using strided average pool to match the size of feature maps to be focused and then normalized to account for scale variations of hand. The feature maps are multiplied by pooled masks (hadamard product) resulting in the focused feature maps. Finally, the global pool of these focused feature maps gives the features that are local to the hand.

Figure 7.2 explains the focus layer of CANet on a single feature map from Block 1 of ResNet. Figure 7.2 shows the middle frame from the 10 frame sliding window and corresponding right and left hand masks. The input frame and masks are of shape 240×320. The input window is processed by Block 1 of ResNet-50 resulting in a feature map of 30×40×256 dimensions. One random feature map of shape 30×40 from Block 1 is shown in Figure 7.2. The hand masks are then resized to 30×40 dimensions using a 8×8 average pool with a stride of 8. The hadamard product of feature maps with the pooled masks gives the focused feature maps as shown in Figure 7.2. Finally, the focused feature maps are averaged spatially (global pool) resulting in features for the respective hand. The random feature map in Figure 7.2 from Block 1 of ResNet resulted in a feature value of 0.08 for left hand and 0.07 for right hand.

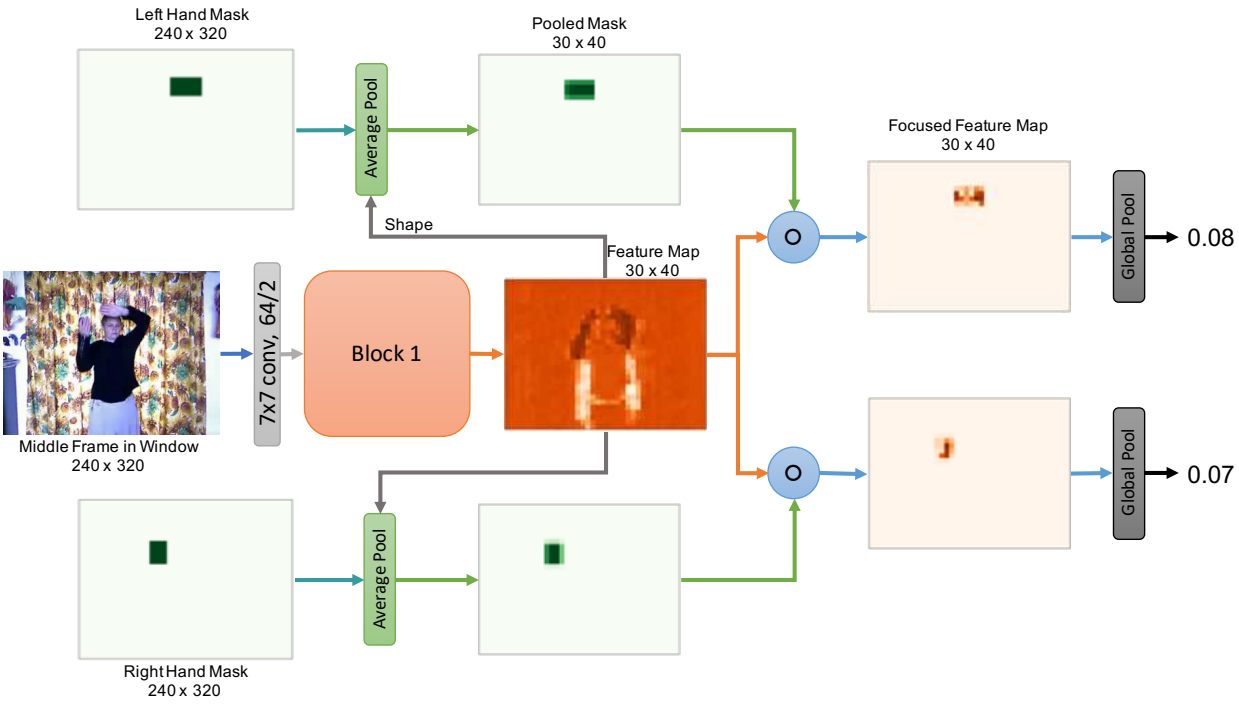

Figure 7.2: Focus Layer of CANet after Block 1 of ResNet. The figure shows a middle frame from the input window to CANet along with right and left hand masks. Input window is processed by Block 1 of ResNet resulting in $30 \times 40 \times 256$ feature maps and one random feature map is shown in the figure. The average pool layer resizes the hand masks to match the size of feature map. The hadamard product of pooled mask and feature map results in feature focused maps which are then global pooled resulting in focused features.

In addition to focus layers, CANet also has skip connections for left and right hands. Skip connections (a.k.a shortcut connections) bypass some layers and have long been used in neural networks [36, 43, 44, 47, 65, 100, 109, 111]. Skip connections can be introduced between layers or to output layers with the explicit aim of improving the flow of information. In addition, skip connections are also used to extract multi-level features, and these multi-level features have been found to be effective for various vision tasks [41, 70, 105, 132]. CANet uses skip connections for two reasons: to extract multi-level features and to reduce the effect of background.

The focus layers of CANet extract the features of the hand by multiplying the feature maps with a hand mask. The hands occupy only a small region in the image frame, which is reduced to a very small region (usually around a pixel) in the last feature map. Skip connections are used in CANet to extract features at different scales. Skip connections were not needed in FOANet

because focus channels on the other hand resizes the hands to 128 $\times$128, resulting in features at much finer scale compared to CANet focus features from last layer.

Skip connections also reduce the effect of background due to increasing receptive field with the depth of the network. Receptive field is the region of the input image that affects one pixel of a feature map, and the receptive field increases with convolutions and pooling operations. So, the receptive field increases with the depth of the network. The final feature maps of ResNet-50 have a receptive field that covers the entire image. Each value in last feature map is influenced by the entire input image. Although the focus layer of CANet removes the background features, the features in focused region still have the influence of background due to convolutions. As the initial layers of CNNs have smaller receptive fields, features from these layers are less influenced by the background. Skip connections help give these early features more influence. In addition, the backpropogation into initial layers of CNN by skip connections helps to train focused features in the initial layers, thereby benefiting subsequent layers. FOANet focus channels, on the other hand, crop and resize the hands before processing by CNN. So, FOANet focus channels doesn't face this challenge of background intervention in focus features.

ResNet-50 has 50 layers that are grouped into four blocks. Each block has multiple convolutional layers with residual connections. Feature maps are downsampled by a factor of two and feature depth is increased by a factor of two between ResNet blocks. Skip connections in CANet are initiated from these four blocks of ResNet. The feature maps after each block are passed into a focus layer and the corresponding hand features are global pooled and concatenated. For example, a 240$\times$320$\times$30 input window results in 30$\times$40$\times$256 sized feature maps after Block 1 convolutions of ResNet. These feature maps are passed to focus layer resulting in 256 features.

For each hand, the skip connections from four blocks results in 256 features from Block 1, 512 features from Block 2, 1024 features from Block 3 and 2048 features from Block 4. The receptive field after Block 1 is 35, Block 2 is 99, Block 3 is 291 and Block 4 is 483. The focused features from these four blocks are concatenated together along with 14 location features (see Chapter 3) resulting in 3854 features for each hand. As global channels don't have skip connections, they have

only 2048 features. These three feature sets are then classified using three separate fully connected layers. The total loss is the sum of softmax loss from three classification heads.

7.4 Experiments

The previous section introduced the CANet architecture. This section evaluates the performance of CANet on ChaLearn IsoGD dataset. The overall accuracy of CANet is compared to FOANet along with the previous state-of-the-art methods on ChaLearn IsoGD. This section also compares individual channel accuracy of CANet and FOANet to see the interaction between network architecture and individual channel performance. In addition, this section also compares the channel fusion results (using sparse network fusion) of different channels of CANet with FOANet.

In addition to accuracy, we will also compare CANet and FOANet architecture in terms of the number of network parameters and number of operations required for a single forward pass. The more parameters a network has, the more data it needs to avoid overfitting, and the more time it needs for training. The more operations it performs per image, the slower it is in real-time applications.

The rest of this section is organized as follows. Section 7.4.1 discusses the training and inference process of CANet on ChaLearn IsoGD dataset. The results of CANet on ChaLearn IsoGD dataset is presented in Section 7.4.2. Section 7.4.3 compares FOANet and CANet in terms of network parameters and operations. Finally, the results in Section 7.4.4 justifies the skip connections of CANet.

7.4.1 Training and Inference

CANet is trained similar to global channels of FOANet by warm starting from ResNet-50 trained on ImageNet as discussed in Section 3.3.1. Similar to FOANet global channels, the first convolutional layer weights ($7 \times 7 \times 3 \times 64$) are repeated 10 times and stacked together ($7 \times 7 \times 30 \times 64$) to account for 30 channel input window. The fully connected layer weights are randomly initialized and the nets are trained end to end using mini-batch stochastic gradient descent with

momentum (set to 0.9) and a random batch of size 64. The input volume is randomly cropped to a $224 \times 224 \times 30$ volume and random flipping is performed for data augmentation. The learning rate lr is initially set to $2e3$ and is decayed exponentially with a decay factor df of 0.7 and decay steps ds of 10,000 unlike FOANet where decay steps ds is set to 40,000. Focus channels of FOANet are fine-tuned from the respective trained global channels. As CANet merges attention channels, focus channels of CANet can't be warm started from global channels.

Sparse network fusion layer is also trained similar to FOANet as discussed in Section 5.2.1. The weights are trained by using the Adam optimizer a batch size of 32. The initial learning rate is set to 0.01 for first 10,000 steps, and is decreased to 0.001 till 20,000 steps and is further decreased to 0.0001. The training is stopped after 50,000 iterations.

During inference time, the softmax scores of a window are computed by processing the window using ResNet-50. The softmax score of the video is then calculated by averaging the softmax scores at every timestep of the video. These softmax scores from all channels are stacked together and are multiplied by the fusion layer weights. The argmax of the diagonal of the resulting matrix is the predicted gesture label.

7.4.2 Results

Our method achieves state-of-the-art performance on the ChaLearn IsoGD dataset, as shown in Table 7.1. Table 7.1 also shows the performance of FOANet along with the top performing entries from the ChaLearn 2017 competition [118]. On the validation data CANet outperforms FOANet by 3.28%, with an accuracy of 84.24% compared to the FOANet accuracy of 80.96%. On the test set CANet achieves an accuracy of 84.79%, outperforming FOANet by 2.72%. These results shows the advantage of processing global and focus channels together by a single CNN.

Table 7.2 shows the individual channel accuracies of CANet and FOANet on ChaLearn IsoGD dataset at video level. The results in Table 7.2 shows that the global and left hand channels of CANet outperformed the respective FOANet channels on all modalities. When right hand channels are compared, CANet outperformed FOANet only on RGB modality. CANet outperformed

Table 7.1: ChaLearn IsoGD 2017 results. Entries are ordered by their performance on test data. Results on systems other than ours were previously reported in [118].

System	Valid	Test
CANet	**84.24%**	**84.79%**
FOANet	**80.96%**	**82.07%**
Miao *et al*. [77] (ASU)	64.40%	67.71%
SYSU_IEEE	59.70%	67.02%
Lostoy	62.02%	65.97%
Wang *et al*. [123] (AMRL)	60.81%	65.59%
Zhang *et al*. [136] (XDETVP)	58.00%	60.47%

FOANet on 9 of 12 channels with a significant margin ranging from 2.29% to 27.3%. Although, FOANet outperformed CANet on remaining three channels, the margin is comparatively small with a range of 0.47% to 2.26%. These results shows that CANet is generally a better architecture at channel level compared to FOANet.

When global channels are processed by CANet, the performance of depth channels improved the most followed by RGB, depth Flow and RGB Flow channels. The performance of CANet depth global channel increased by 27.3% on validation set and 20.48% on test set over the respective FOANet channel. RGB global channel achieves an accuracy of 54.44% on validation set and 57.17% on test set when processed by CANet - a 21.22% increase on validation set and a 15.9% increase on test set over FOANet. When depth flow global channel is processed by CANet, it achieves an accuracy of 43.18% on validation set and 52.32% on test set compared to the FOANet accuracy of 31.66% on validation set and 42.02% on test set. Similarly, CANet increased the performance of RGB flow global channels by 4.31% on validation set and 2.29% on test set. As CANet has a single CNN with two additional heads for right and left hand channels, those additional attention heads forces the CNN architecture to focus on hands, thus improving the performance of global channels as well.

By comparing the left hand channel results of CANet and FOANet, we see that the performance of RGB channels improved the most when processed by CANet followed by RGB flow, depth flow and depth modalities. On validation set, the RGB left channel of CANet achieved an accuracy of 27.13% - a 9.96% increase over FOANet. The RGB flow left hand channel of CANet achieved an accuracy of 31.19% compared to 24.14% for FOANet. The performance of depth flow and

Table 7.2: Video level accuracies of individual channels of CANet and FOANet on ChaLearn IsoGD dataset.

		Valid		Test	
		FOANet	**CANet**	**FOANet**	**CANet**
RGB	**Global**	33.22	54.44	41.27	57.17
	Left Hand	16.17	27.13	16.63	26.20
	Right Hand	41.60	49.27	47.41	53.86
Depth	**Global**	27.98	55.28	38.50	58.98
	Left Hand	23.76	27.32	24.06	29.72
	Right Hand	54.91	54.44	64.44	63.89
RGB Flow	**Global**	46.22	50.53	50.96	53.25
	Left Hand	24.14	31.19	24.02	34.03
	Right Hand	54.60	53.91	59.69	58.17
Depth Flow	**Global**	31.66	43.18	42.02	52.32
	Left Hand	21.84	26.89	22.71	27.92
	Right Hand	48.32	46.78	58.79	56.43

depth left hand channels of CANet also increased by 5.05% and 3.56% respectively over FOANet. On the test set of ChaLearn IsoGD, the performance of RGB, RGB flow, depth flow and depth modalities increased by 9.57%, 10.01%, 5.21% and 5.66% respectively when these channels are processed by CANet rather than FOANet. The performance improvement of left hand channels of CANet can be attributed to the presence of global and right hand classification heads. As global, left hand and right hand channels are trained together, left hand channels of CANet might have benefited from other channels and resulted in the performance improvement of left hand channels.

The pattern of CANet outperforming FOANet on all modalities of global and left hand channels doesn't hold in the case of right hand channels as shown in Table 7.2. CANet outperforms FOANet only on RGB modality achieving an accuracy of 54.44% on validation set and 57.17% on test set compared to the FOANet accuracy of 33.22% on validation set and 41.27% on test set. On other three modalities, FOANet outperforms CANet. By looking at the accuracy of depth, rgb flow and depth flow modalities, we see that the performance of depth flow modality drops more followed by RGB flow and depth modalities. On validation set, FOANet outperforms CANet by 1.54%, 0.69% and 0.47% on depth flow, RGB flow and depth channels respectively. Similarly, FOANet outperforms CANet by 2.36%, 1.52% and 0.55% on depth flow, RGB flow and depth channels respectively of ChaLearn IsoGD test set. CANet and FOANet are different architectures

and the differences between these two architectures are listed previously. As discussed, CANet focus channels has background influence unlike FOANet focus channels and it might have affected CANet right hand channels' performance. Moreover, FOANet focus channels are warm started from trained global channels which is not the case with CANet.

Next we fuse different combinations of channels using sparse network fusion, as shown in Table 7.3. From the second and third fusion rows, we can see that the combination of global channels is better than the combination of focus channels. In fact, the fusion of global channels is the best combination, short of combining all channels. Interestingly the combination of focus channels performs better than combination of global channels in FOANet - the fusion of global channels is the worst combination in FOANet. Looking back at Table 7.2, we can see that the performance of global channels increased a lot by merging attention channels and that contributed to global channels overall performance. We can also see that most of the information from focus is contributed by the right hand alone which can be attributed to the right handed bias in the dataset.

We also notice that the fusion of RGB and RGB flow nets is better than the fusion of depth and depth flow nets on validation set for both CANet and FOANet. However on the test set, depth + depth flow performed better. Next, we see that the fusion of RGB and depth channels performs better than the fusion of RGB flow and depth flow channels for CANet. However, fusion of RGB and depth channels performed on par with the fusion of RGB flow and depth flow channels for FOANet. Looking back at Table 7.2, we can notice that the flow right hand channels of CANet lose to corresponding FOANet channels affecting the fusion performance of flow channels.

7.4.3 Architecture Analysis

As previously discussed, there are two motivations for CANet: accuracy and efficiency. CANet exploits complementary information from different attention regions and outperforms FOANet by 3.28% on validation set and 2.72% on test set of ChaLearn IsoGD dataset as shown in Table 7.1. This section compares CANet and FOANet in terms of number of parameters and number of floating point operations required for a single forward pass to evaluate the efficiency of CANet.

	RGB			Depth			RGB Flow			Depth Flow			Validation		Test	
	Global	**Left**	**Right**	**Global**	**Left**	**Right**	**Global**	**Left**	**Right**	**Global**	**Left**	**Right**	**FOANet**	**CANet**	**FOANet**	**CANet**
All	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	80.96	84.24	82.07	84.79
Global	✓			✓			✓			✓			61.4	78.92	67.50	79.31
Focus		✓	✓		✓	✓		✓	✓		✓	✓	76.76	77.75	77.61	78.14
Right			✓			✓			✓			✓	72.64	74.98	74.46	75.93
RGB	✓	✓	✓				✓	✓	✓				71.41	75.25	75.41	76.32
Depth				✓	✓	✓				✓	✓	✓	68.56	70.15	76.39	77.51
Raw	✓	✓	✓	✓	✓	✓							70.69	75.58	75.29	78.36
Flow							✓	✓	✓	✓	✓	✓	70.49	71.69	74.39	75.42

Table 7.3: Results of fusing different combinations of channels from FOANet and CANet. 'Raw' refers to input from a stack of unprocessed images, whereas 'flow' refers to input of a stack of flow field images.

Table 7.4: Number of parameters (in millions) and operations (in G-Ops) for a single modality of CANet. The table also shows the parameters and operations for a single modality of FOANet (combination of global, left hand and right hand channels).

	Parameters [Million]	**Operations [G-Ops]**
Common	23.604	14.073
Global	0.510	0.001
Left Hand	0.959	0.002
Right Hand	0.959	0.002
CANet	26.033	14.078
FOANet	72.350	20.013

Table 7.4 shows the number of network parameters and number of floating point operations for a single modality of CANet and FOANet. The numbers in Table 7.3 shows that both CANet and FOANet have large number of parameters (measured in millions) and operations (measured in G-Ops). For a single modality, CANet has 26.033 million parameters which is only ≈36% of FOANet parameters (72.350 million). Comparing the number of operations, we see that CANet has only 14.078 G-Ops which is ≈70% of FOANet operations (20.013 G-Ops).

As CANet merges attention channels into a single network a majority of parameters and operations are shared for all attention regions. More specifically, 23.604 million operations and 14.073 G-Ops operations are common for all attention regions as shown in the first row of Table 7.4. This accounts to 90.66% of parameters and 99.96% of operations in CANet. Global channels have additional 0.510 million parameters and right and left hand channels have 0.959 million additional parameters each. The additional parameters in right and left hand channels can be attributed to the large feature vector from skip connections. Similarly in terms of operations, global channels contribute to additional 0.001 G-Ops and each hand channel add 0.002 G-Ops operations to CANet common operations. In addition to the large feature vector in hand channels, the hadamard product of hand mask with feature maps also add additional operations for hand channels of CANet.

The results in Table 7.4 show that CANet has only ≈36% of FOANet parameters whereas the number of operations are ≈70% of FOANet. As FOANet processes a volume of 128 × 128 ×

30 dimensions for focused channels compared to 240 $\times$ 320 $\times$ 30 volume for global channels, focus channels of FOANet has only $\approx$ 21% of global channel operations. So, CANet has higher percentage of operations than parameters when compared to FOANet.

This section compared CANet and FOANet in terms of network parameters and operations. This comparison showed that CANet achieves higher accuracy than FOANet with less parameters and operations. So, CANet is an efficient architecture compared to FOANet.

7.4.4 Effect of Skip Connections

CANet architecture merges spatial attention channels of FOANet while preserving attention by multiplying feature maps with hand masks. As previously discussed, hands occupy a very small region in the entire FOV and the features from hands span a very small region in the final feature map (around a pixel). Focus channels of FOANet, on the other hand, resize the cropped hands to 128 $\times$ 128 dimensions resulting in features at a finer scale. Moreover, the last feature maps of ResNet-50 have a receptive field of 483. So the focused features from last feature maps have a very high influence of the background. In contrast, FOANet focus channels doesn't have the effect of background as the hands are cropped before being processed by ResNet. So, skip connections are introduced in CANet to get focus features at different scales and decrease the effect of background on focused features as receptive field increases with the depth of the network.

This section examines the effect of skip connections on the performance of CANet. Table 7.5 presents the accuracies of CANet channels with and without skip connections on validation and test set of ChaLearn IsoGD dataset. When all channels are combined together, CANet with skip connections outperformed non skip version by 2.47% on validation set and 2.67% on test set. In fact, non skip version of CANet outperforms FOANet by a very small margin (81.77% vs 80.96% on validation set, 82.12% vs 82.07% on test set). When individual channels are considered, the advantage of skip connections becomes more evident. The performance of all channels increased by adding skip connections. The performance gains of skip connections range from to 1.07% to 6.61%. At the level of individual channels, FOANet outperforms CANet on three right hand

Table 7.5: Accuracies of CANet channels with and without skip connections on validation and test set of ChaLearn IsoGD. Last row shows the sparse network fusion accuracy.

		Validation		Test	
		Skip Connections	**No Skip Connections**	**Skip Connections**	**No Skip Connections**
RGB	**Global**	54.44	47.83	57.17	51.03
	Left	27.13	24.09	26.20	24.12
	Right	49.27	46.33	53.86	50.29
Depth	**Global**	55.28	54.07	58.98	56.13
	Left	27.32	25.02	29.72	28.77
	Right	54.44	50.65	63.89	59.23
RGB Flow	**Global**	50.53	49.40	53.25	52.18
	Left	31.19	28.55	34.03	32.56
	Right	53.91	50.56	58.17	54.27
Depth Flow	**Global**	43.18	40.69	52.32	50.34
	Left	26.89	24.57	27.92	25.66
	Right	46.78	40.58	56.43	51.12
Sparse Fusion		84.24	81.77	84.79	82.12

channels with a small margin. Without skip connections, the difference would have been much higher on three right hand channels.

It is interesting to see that the performance of global channels also increased even though the skip connections are added only to right hand and left hand channels. Skip connections on hand channels might have resulted in better feature representation for hands which in turn helped global channels as well. We also tried a version where skip connections were added to global channels, but it negatively impacted the performance of all channels.

In summary, skip connections increased the performance of all channels and sparse fusion in CANet. Adding skip connections to hand channels is a good idea, but adding skip connections to global channels impacts the performance of all channels.

7.5 Summary

This chapter introduced a new architecture called CANet for gesture recognition. CANet has a single channel for spatial attention regions (global, left hand and right hand) unlike FOANet that has a separate channel for each attention region. CANet merges attention channels into a single channel while still preserving attention with the help of focus layer. Skip connections of CANet allows the features to be captured at different scales while reducing the influence of background on the features.

CANet achieves an accuracy of 84.24% on validation set and 84.79% on test set of ChaLearn IsoGD dataset outperforming FOANet by 3.28% and 2.72% on validation and test sets respectively. More importantly, CANet achieved these results with only $\approx 36\%$ of FOANet parameters and $\approx 70\%$ of FOANet operations. At the channel level, CANet outperformed FOANet on 9 out of 12 channels with biggest gains on global channels.

Chapter 8

Cooperating with Avatars through Gesture, Language and Action

Previous chapters presented the intellectual contribution of this thesis. We analyzed each sub component of the proposed system to understand the value of each components. However, the practical impact of this system lies in putting these components together and demonstrating it on a real world application. This chapter describes such an application, developed jointly by the author and other researchers at Colorado State University, University of Florida and Brandeis University. In particular, I would like to thank Nikhil Krishnaswamy, Isaac Wang, Rahul Bangar, Dhruva Patil, Gururaj Mulay, Kyeongmin Rim, Ross Beveridge, Jaime Ruiz, James Pustejovsky, and Bruce Draper for their contributions to the work presented in this chapter.

Advances in artificial intelligence are fundamentally changing how we relate to machines. We used to treat computers as tools, but now we expect them to be agents, and increasingly our instinct is to treat them like peers [60]. For example, we talk to them and give them personal names (e.g. Alexa, Siri, Cortana). Unfortunately, the more familiar we become with artificial agents, the more frustrated we become with their limitations. We expect them to see and hear and reason like people. *No, no, Alexa, can't you see that I ...*

This is the reason that the Defense Advanced Research Projects Agency (DARPA) funded the "Communication With Computers(CWC)" program in 2015. The goal of CWC is to push the current limits of human-computer interaction, with the eventual goal of making human-computer communication as natural as human-human interaction. Gestures are an important part of communication among humans, and gestures will aid communication between humans and computers as well. Colorado State University is one of the 18 universities and institutes participating in the CWC program, and the goal of CSU within this program is to ensure that the new paradigm includes gestures, not just words. We developed a prototype system with a limited form of peer-to-

peer communication by collaborating with Brandeis University and the University of Florida. The methods for gesture recognition presented in this thesis enable real time gesture recognition in the context of this program.

Two ideas are central for peer-to-peer communication. The first is *shared perception*. When people work together, much of the information that passes between them is contextual and derived from perception. Imagine, for example, two people cleaning a room. They might discuss high-level strategy (“you start here, I’ll start over there"), but they don’t describe every action they take to each other. They can just look at the room to see what the other person has or has not done. Only the high-level discussion is verbal, and even that is grounded in perception: the definitions of “here" and “over there" depend on knowing where the other person is. In general, when one person changes the state of the world, the other person sees it, and the goal of conversation is to add information beyond what is provided by perception.

The second idea is that reasoning about physical objects is grounded in visualization. Imagine a simple command like “put the book on the table". Is this command feasible? Yes, if there exists both a book and a table, and there is a clear spot on the table at least the size of the book, and if there is a clear path from the book’s position to the table. In general the command can be understood if it can be visually simulated.

We explore these ideas in blocks world. In particular, we consider a scenario in which one person (the *builder*) has a table with blocks on it, and another person (the *signaler*) is given a target pattern of blocks. Only the builder can move the blocks, so the signaler has to tell the builder what to do. While blocks world is obviously not a real-world application, it serves as a surrogate for any cooperative task with a shared workspace.

We begin our exploration with elicitation studies (led by Jaime Ruiz) similar to Wobbrock *et al*. [131], but with differences in how gestures are elicited. In the original elicitation study format, people are given specific actions (called referents), and asked to create a user-defined gesture (called signs) for the action. In our study, we take a more natural approach and simply present pairs of people with tasks to complete and observe the actions and gestures that naturally

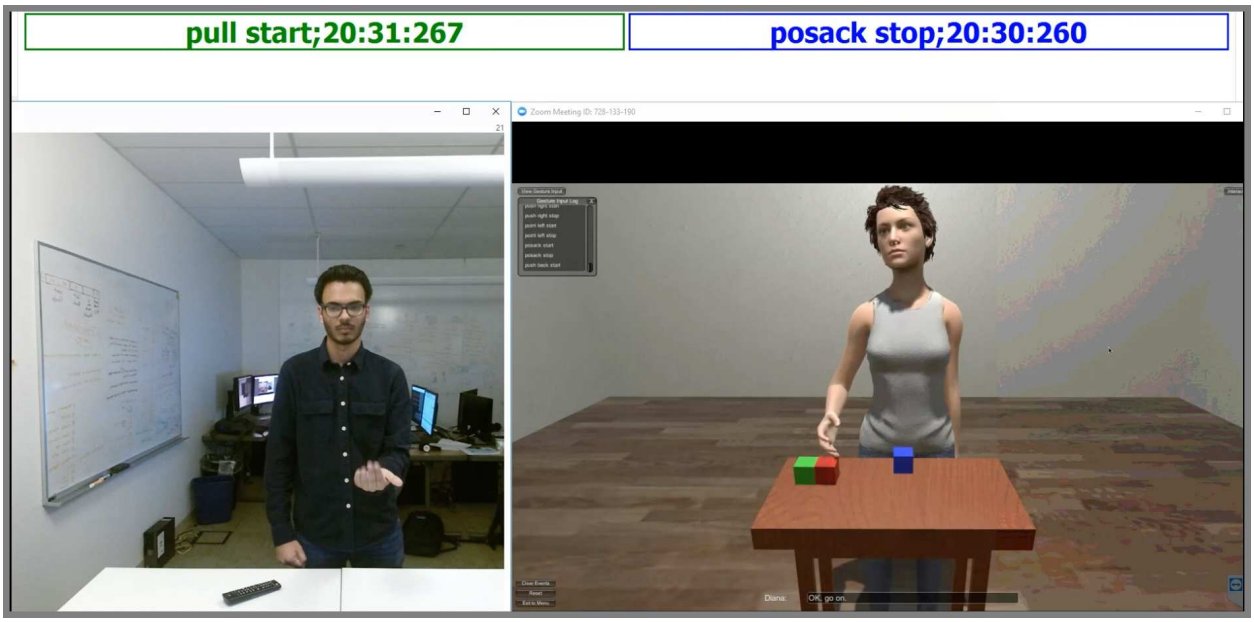

Figure 8.1: Prototype peer-to-peer interface. The signaler on the left can communicate only through gestures. The avatar on the right can communicate through language, gesture and action.

occur. The signaler and builder are people in separate rooms, connected by a video link. We vary the communication between them across three conditions: (1) the signaler and builder can both see and hear each other; (2) the signaler and builder can see but not hear each other; and (3) the signaler and builder can only hear each other (the signaler can see the builder's table and knows where the blocks are, but cannot see the builder).

Using gestures observed in the elicitation studies, we develop a prototype system in which the signaler is a person but the builder is an avatar with a virtual table and virtual blocks. The signaler can see a graphical projection of the virtual world, and communicate to the avatar through speech and gesture. The avatar communicates back through speech, gesture, and action, where an action is to move a block. Figure 8.1 shows the set up, with the signaler on the left and the avatar on the right in her virtual world.

Experience derived from using the prototype reveals important features of human-computer cooperation on shared physical tasks. For example, we learned that complex, gesture-based peer-to-peer conversations can be constructed from relatively few gestures, as long as the gesture set includes: (1) social gestures, for example acknowledgement and disagreement; (2) deictic gestures,

such as pointing; and (3) iconic gestures mimicking specific actions, such as pushing or picking up a block. When the builder and avatar are allowed to speak, words can replace the iconic gestures, but the social and deictic gestures remain important. We learned that ambiguities arise in the context of conversations not just from questions of reference, i.e. which block to pick up, but also from options among actions, for example whether to put a block down on top of another block or next to it. Fortunately, these ambiguities are easily resolved if the conversational lead is allowed to switch from the signaler to the builder (in this case, the avatar). We also came to appreciate the importance of making two or more gestures at the same time, for example nodding (a social gesture) while signaling for the builder to pick up an object. Finally, we learned how important it was for the avatar to gesture back to the signaler, even when the avatar can speak and move blocks.

From an engineering perspective, we also confirmed that the combination of depth images from inexpensive sensors (Microsoft Kinect v2s) and deep convolutional neural networks is sufficient to recognize 35 common hand poses, and that with GPUs these hand poses can be recognized in real time. The directions of arm movements are also easily detected and are needed for deixis and for supplying directions to representational actions such as push or carry.

8.1 Human/Avatar Blocks World (HAB) System

To explore the role of gesture in peer-to-peer communication with shared perception, we first conducted human subject studies. The goal of these studies is to elicit common gestures and their semantic intents for the blocks world task in order to gain insight about how they might be used by people. Overall, we collected about 12.5 hours of data and hand labeled the data at the level of left and right hand poses, left and right arm motions, and head motions. Summarizing these labels, we discovered 110 combinations of poses and motions that occurred at least 20 times and were performed by at least 4 different subjects. Of these, 29 were determined to have no semantic intent, as when a subject drops their arms to their side. Of the 81 remaining gestures, many were either minor variations or enantiomorphs of each other. For example, a participant might make the “thumbs up" sign while raising their forearm or pushing it forward. Similarly, the “thumbs

Table 8.1: Semantic gestures performed at least 20 times in total by at least 4 different human subjects.

Numeral	Representational	Deictic	Social
one	grab	point (that/there)	start
two	carry	tap (this/here)	done
three	push	this group	positive ack
four	push (servo)	column	negative ack
five	push together	row	wait for
	rotate		wait (pause)
			emphasis

up" sign might be made with the right hand, the left hand, or both. We also grouped physically different but semantically similar poses, such as the “thumbs up" and “OK" signs. After grouping similar motions and poses, we were left with 22 unique semantic gestures, as shown in Table 8.1.

The 22 semantic gestures fall into four categories. Deictic gestures, such as pointing or tapping the table, serve to denote objects or locations. Iconic gestures, such as grab, push or carry, mimic actions. Social gestures, such as head nods or thumbs up, address the state of the dialog. Numerals are a form of abstract plural reference. We note that there are broader and more inclusive schemes for categorizing gestures (see [53], chapter 6), but none are universally accepted and the simple categories above work well for describing the gestures we observed in blocks world.

The elicitation studies provide insights about gestures people use in blocks world. Our goal, however, is to explore peer-to-peer communication between people and computers using visually-grounded reasoning in the context of shared perception. To this end, we created a prototype system that replicates the elicitation study setup, except that the builder is now an avatar and the blocks and table are virtual. The human/avatar blocks world (HAB) system operates in real time, and allows the human signaler to gesture and speak to the avatar. The avatar can gesture and speak in return, as well as move blocks in the virtual world. In some tests we turn off the audio channel, thereby eliminating words and limiting communication to gestures and observation.

HAB has three major components - the perceptual module that implements gesture recognition, the grounded semantics module (VoxSim) which determines the avatar’s behavior, and the interplay between perception and reasoning. The perceptual module is described in subsection 8.1.1,

VoxSim is described in subsection 8.1.2, and the interactions between perception and VoxSim are described in subsection 8.1.3.

8.1.1 Perception

We hypothesize that shared perception is the basis for peer-to-peer communication, particularly when working in a common workspace. To implement shared perception, the human signaler and avatar builder need to be able to see each other as well as the virtual table and its blocks. Perceiving the virtual table and blocks is relatively easy. The avatar has direct access to the virtual world, and can directly query the positions of the blocks relative to the table. The human builder sees the rendering of the virtual world, and therefore knows where the blocks are as well.

More challenging is the requirement for the human and avatar to see each other and interpret each other's gestures. Based on our elicitation studies, we have a lexicon of commonly occurring gestures. We animate the avatar so that she can perform all of these gestures, and rely on the builder's eyes to recognize them. We also have an RGB-D video stream of the human captured by the Microsoft Kinect v2. The rest of this section describes the real-time vision system used to recognize the builder's gestures in real time.

Gesture recognition is implemented by independently labeling five body parts. The methods for gesture recognition presented in this thesis enabled us to do real time gesture recognition in HAB. The left and right hands are labeled according to their pose. The system is trained to recognize 34 distinct hand gestures in depth images, plus a 35th label ("other") that is used for hands at rest or in unknown poses. The hand poses are directional, in the sense that pointing down is considered a different pose than pointing to the right. Head motions are classified as either nod, shake or other based on a time window of depth difference images. Finally, the left and right arms are labeled according to their direction of motion, based on the pose estimates generated by the Microsoft Kinect [137]. As the 34 hand gestures recognized by HAB are static poses performed by a single hand, we only use focus channels introduced in this thesis to recognize these gestures in HAB.

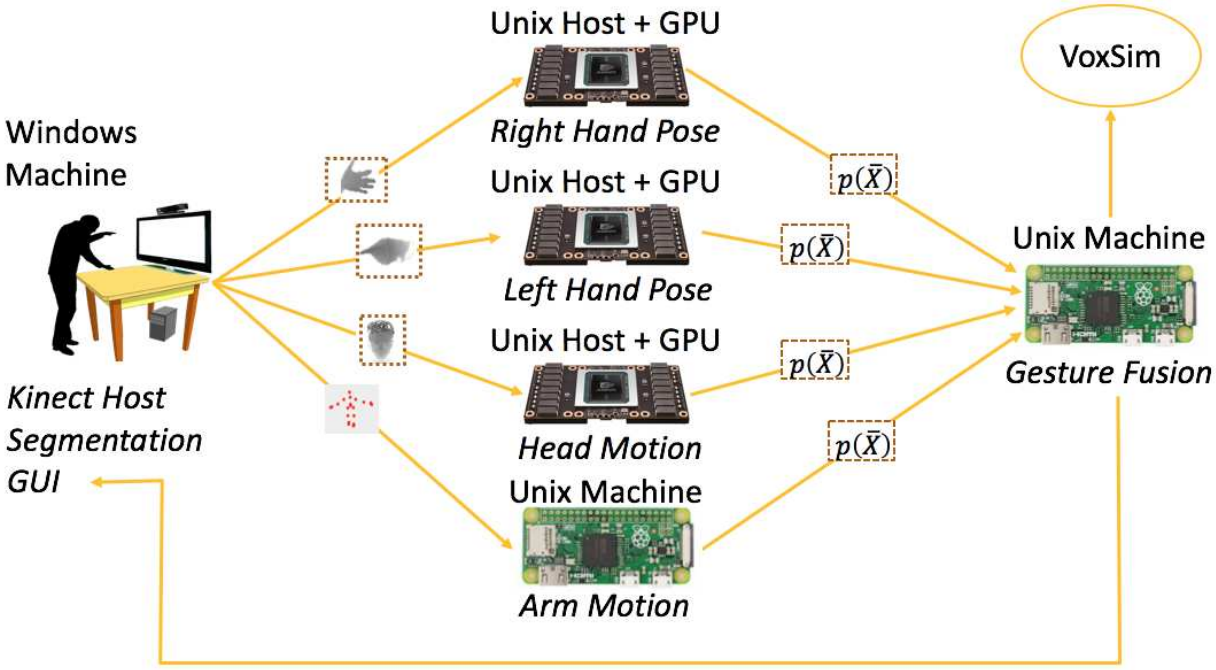

Figure 8.2: The architecture of the real-time gesture recognition module.

To recognize gestures in real time, the computation is spread across 6 processors, as shown in Figure 8.2. The processor shown on the left is the host for the Microsoft Kinect, whose sensor is mounted on top of the signaler's monitor. It uses the Kinect's pose data to locate and segment the signaler's hands and head, producing three streams of depth images. The pose data also becomes a data stream that is used to label arm directions. The hand and head streams are classified by a ResNet-style deep convolutional neural network (DCNN) [43]. Each net is hosted on its own processor, with its own NVIDIA Titan X GPU. The arm labeling process has its own (non-GPU) processor. Finally, a sixth processor collects the hand, arm and head labels and fuses them using finite state machines to detect gestures.

One challenge with HAB that we didn't have with ChaLearn and NVIDIA datasets is that the data is streamed continuously and the gestures should be recognized in real time. So, we process one frame at a time and do a simple forward smoothing. Let p^t be the softmax scores at time t. The smoothed softmax scores p^t_{smooth} at time t is calculated by the following equations.

$$p^0_{smooth} = p^0 \tag{8.1}$$

$$p^t_{smooth} = 0.5 * p^t + p^{t-1}_{smooth} \tag{8.2}$$

8.1.2 VoxSim

The avatar's reasoning system is built by Brandeis university on the VoxSim platform [56, 57]. VoxSim is an open-source, semantically-informed 3D visual event simulator implemented in Unity [33] that leverages Unity's graphics processing, UI, and physics subsystems.

VoxSim maps natural language event semantics through a dynamic interval temporal logic (DITL) [98] and the visualization modeling language VoxML [97]. VoxML describes and encodes qualitative and geometrical knowledge about objects and events that is presupposed in linguistic utterances but not made explicit in a visual modality. This includes information about symmetry or concavity in an object's physical structure, the relations entailed by the occurrence of an event in a narrative, the qualitative relations described by a positional adjunct, or behaviors *afforded* by an object's *habitat* [72, 96] associated with the situational context that enables or disables certain actions that may be undertaken using the object. Such information is a natural extension of the lexical semantic typing provided within Generative Lexicon Theory [95], towards a semantics of embodiment. This allows our avatar to determine which regions, objects, or parts of objects may be indicated by deictic gestures, and the natural language interface allows for explicit disambiguation in human-understandable terms. The movement of objects and the movement of agents are compositional in the VoxML framework, allowing VoxSim to easily separate them in the virtual world, which means that the gesture used to refer to an action (or program) can be directly mapped to the action itself, establishing a shared context in which disambiguation can be grounded from the perspective of both the human and the computer program.

8.1.3 Perception & VoxSim

To create a single, integrated system we connect the recognition module (and by extension, the human signaler) with VoxSim and its simulated world. VoxSim receives "words" from the gesture

recognizer over a socket connection, and interprets them at a contextually-wrapped compositional semantic level. The words may be either spoken or gestured by the (human) builder. For the moment, we have seven multi-modal “words":

1. *Engage*. Begins when the signaler steps up to the table or says “hello", and ends when they step back or say “goodbye". Indicates that the signaler is engaged with the avatar.

2. *Positive acknowledge*. Indicated by the word “yes", a head nod, or a thumbs up pose with either or both hands. Used to signal agreement with a choice by the avatar or affirmative response to a question.

3. *Negative acknowledge*. Indicated by the word “no", a head shake, a thumbs down pose with any combination of hands, or a stop sign gestured with the hand closed, palm forward, and fingertips up. Signals disagreement with a choice by the avatar or negative response to a question.

4. *Point*. Gestured by extending a single finger, with the optional spoken words “this" or “that". The information given to VoxSim about pointing gestures includes the spot on the tabletop being pointed to. Signals either a block to be used for a future action, or an empty space.

5. *Grab*. Indicated by a claw-like pose of the hand that mimics grabbing a block or the word “grab". Tells the avatar to grab a block that was previously pointed to.

6. *Carry*. Indicated by moving the arm while the hand is in the grab position, with the optional spoken word “carry". The information given to VoxSim includes a direction, one of left, right, forward, back, up or down. A “carry up" can be thought of as pick up, and a “carry down" is equivalent to put down.

7. *Push*. Gestured with a flat, closed hand moving in the direction of the palm, with the optional spoken word “push". Similar to carry, it includes a direction, although *up* and *down* are not allowed. As a special case, a beckoning gesture signals the avatar to push a block toward the signaler.

In addition to the multi-modal “words”, there are words that can only be spoken, not gestured. These words correspond to the block colors: black, red, green, blue, yellow and purple. Speech recognition is implemented by simple word spotting, so for example the phrase "the red one" would be interpreted as the single word “red.”

The flow of information from the avatar/builder back to human/signaler is similar. The avatar can say the words and perform the gestures mentioned above, with the additional gesture of reaching out and touching a virtual block (a gesture not available to the human builder). One important difference, however, is that the avatar can also communicate through action. Because the human signaler can see the avatar and the virtual blocks world, they can see when the avatar picks up or moves a block.

Any time the recognition module determines that one of the known “words" begins or ends, VoxSim receives a message. VoxSim responds by parsing the meaning of the gesture in context. For example, if the gesture points to a spot on the right side of the table and the avatar is currently holding a block, then the gesture is a request to move the block to that spot. Alternatively, if the avatar is not holding a block, the same gesture selects the block nearest to the point as the subject of the next action.

Gestural ambiguities are common. If two blocks are near each other, a pointing gesture in their direction is ambiguous. Which block did the signaler point to? Similarly, if the user says “the red one" when there are two red blocks on the table, the reference is ambiguous. Actions may also be ambiguous. If the user signals the avatar to put down one block near another, should it stack the blocks or put them side by side?

When presented with ambiguities, VoxSim assumes the initiative in the conversation and asks the user to choose among possible interpretations. In the case of pointing, for example, VoxSim might ask “do you mean the red block?". If the answer is negative, it might then try “do you mean the green block?". VoxSim orders the options according to a set of heuristics that favor interesting interpretations over less interesting ones. For example, if the options are to stack a red block on top

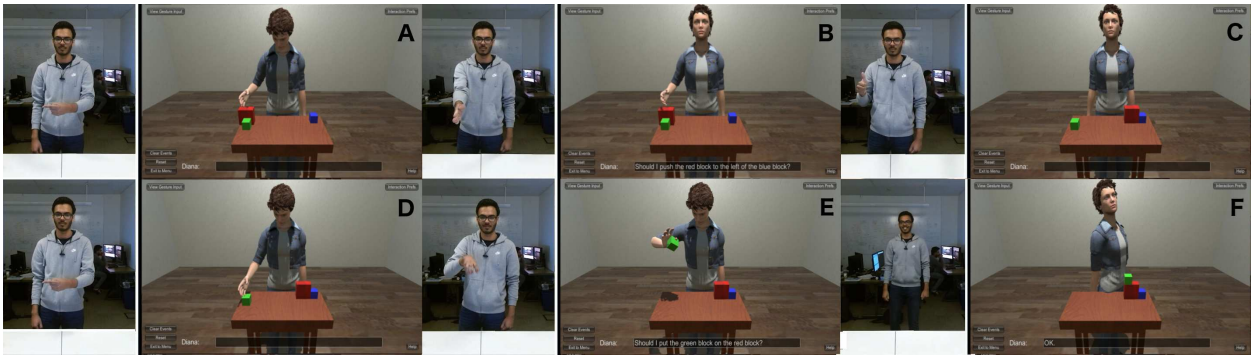

Figure 8.3: An example of building a staircase in HAB using only gestures and actions (no spoken language).

of a blue block or put them next to each other, VoxSim favors the stacking option, because stacks are interesting.

8.2 Example

We illustrate HAB with an example in which the audio has been turned off, so all communication happens through gestures and actions. The example begins with three blocks on the table: a green block and a red block to the right of the signaler, and a blue one on the left. The signaler's goal is to arrange the blocks in a staircase. The conversation begins when the signaler steps up to the table, causing an *engage* gesture to be recognized and sent to VoxSim.

The signaler points to the left, as shown in Frame A of Figure 8.3. VoxSim, interprets this gesture as selecting the blue block for the next action. The avatar moves its hand toward the blue block in anticipation; this is a gesture that serves as a form of positive acknowledgement, since it lets the signaler know what the avatar understood. The signaler then beckons, and the avatar pushes the block away from itself and toward the signaler.

Next the signaler points to his right where the red and green blocks are (Frame B of Figure 8.3). This is an ambiguous reference, so the avatar reaches toward the red block as a way of asking whether the signaler means the red block. The signaler shakes his head, sending a negative acknowledgement, so the avatar motions toward the green block. This time the signaler nods,

resolving the ambiguity. The signaler then beckons again, and the avatar pushes the green block toward the signaler.

Continuing with the example, the signaler points toward the blue block and gestures to slide it to the right (Frame C). The slide gesture is ambiguous, however. Should the avatar slide the block a little ways to the right, or slide it all the way to the green block? Sliding it to the green block is the more interesting option, so this is the one the avatar suggests, and since it is what the signaler wants, he gives a thumbs up (Frame D) and the avatar slides the block.

This style of interaction continues. The signaler selects the red block by pointing and then mimics a grabbing motion. Both the reference and action are unambiguous, so the avatar complies. Next the signaler raises his arm while keeping his hand in the grabbing pose, and brings his arm forward. The avatar responds as shown in Frame E. The signaler then lowers his arm and releases his grip, asking the avatar to put the block down. The placement is ambiguous – should the red block go on the blue block, the green block, or the table top? – but some gestural back and forth quickly clear this up, and the staircase is completed, as shown in Frame F.

8.3 Recognition Accuracy

The gesture recognition techniques presented in this thesis outperformed previous state-of-the-art results on ChaLearn IsoGD, NVIDIA and ChaLearn ConGD datasets. However, gestures in ChaLearn and NVIDIA datasets are not natural gestures. One concern that arose while designing HAB was whether the gesture recognition module would be accurate enough to support natural communication. We needed to recognize natural gestures elicited from naive users in the context of blocks world. We used the data from elicitation studies (EGGNOG dataset [125]) to train our focus channels. However, there was not enough training data to train reliable networks. We augmented the data using rotations, translations and scale for 25 hand poses. There were other hand poses, such as thumbs down, that appeared less often but that we still wanted to include in the system. We therefore supplemented the training samples for 10 more hand poses by having

volunteers perform the poses in front of the Kinect. Unfortunately, the data collected in this way turned out to be exaggerated compared to naturally occurring poses.

When the prototype was completed, signalers reported satisfaction with the gesture recognition. In practice, the system neither missed gestures nor inserted false gestures. There are many possible explanations for this, however. For example, the signalers were system designers with an interest in gesturing clearly. Furthermore, although the gesture recognition system detects 35 hand poses, only a few are fused by the finite state machines into the 7 semantic gestures. The underlying performance of hand pose recognition was therefore still unknown.

To evaluate the accuracy of hand pose recognition, we collected new data from 14 naive human subjects, using the same experimental setup and protocol as in [125]. We didn't have the resources to hand label every frame, so instead we adopted a sampling methodology. The new videos were processed through the hand focus channels. To estimate precision, we randomly sampled 60 instances of each hand pose as identified by the focus channels. We then brought in naive raters and asked them whether the detected gestures actually occurred where the focus channels said they did.

The precision results are shown in Figure 8.4. The units of the horizontal axis are seconds, so that the leftmost data point is the precision for gesture detections that lasted for $\frac{1}{5}$ of a second or less. The second data point represents detections with a duration between $\frac{1}{5}$ and $\frac{2}{5}$ of a second, and so on. The orange line represents the precision across all 35 poses, while the blue line shows the precision for the 25 poses trained on data from the human subjects studies. The plot shows that even for gestures that last for $\frac{1}{5}$ of a second or less, over 55% of the DCNNs detections are correct. Long duration gestures (one second or more) have a precision of 80%. When we limit the evaluation to the 25 naturally trained poses, these numbers go up to 60% for short gestures and 87% for long ones.

Recall was estimated using a similar procedure. In this case, naive raters were given portions of videos, and asked to label any hand poses they saw. For each pose, they selected one frame. Although not instructed to do so, they usually selected the first frame in which the pose appeared. We then measured how often the DCNN detected the hand pose at that frame, within a fifth of a

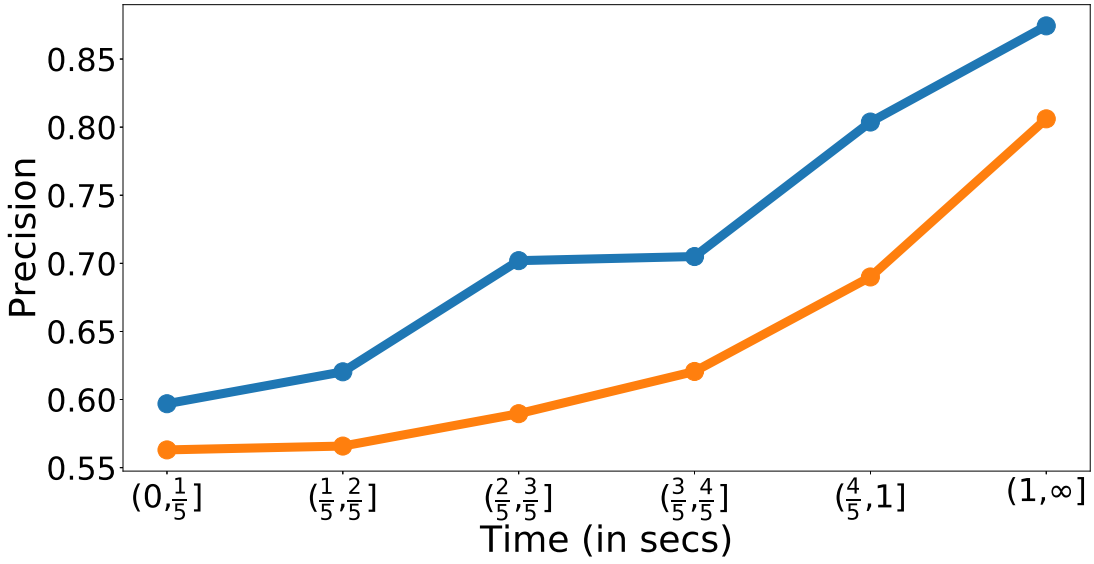

Figure 8.4: Precision of hand pose detection. The horizontal axis represents durations of detected hand poses. The vertical axis is the percent of true detections as judged by naive raters. The blue line shows the precision of the 25 hand poses trained on the human subjects data. The orange line adds 10 more poses for which additional, exaggerated training data was collected.

second of that frame, within two fifths of a second, and so on, up to a second. The resulting plot is shown in Figure 8.5, with the same color scheme as in Figure 8.4 in terms of recall for 25 or 35 poses.

Interestingly, almost half the recall omissions are the result not of mistakes by the DCNN, but of segmentation failures. The depth images passed to the DCNN are windows of the full image centered on the hand, as identified by the Microsoft skeleton. When the hand position is incorrect, the image passed to the DCNN might not contain a hand, in which case failure is inevitable.

8.4 Conclusion

This section presented a system capable of peer-to-peer communication between people and avatars in the context of a shared perceptual task. In the proposed system, people communicate with avatars using gestures and words, and avatars communicate back through gestures, words, and actions. Together, they complete tasks through mixed initiative conversations. The human

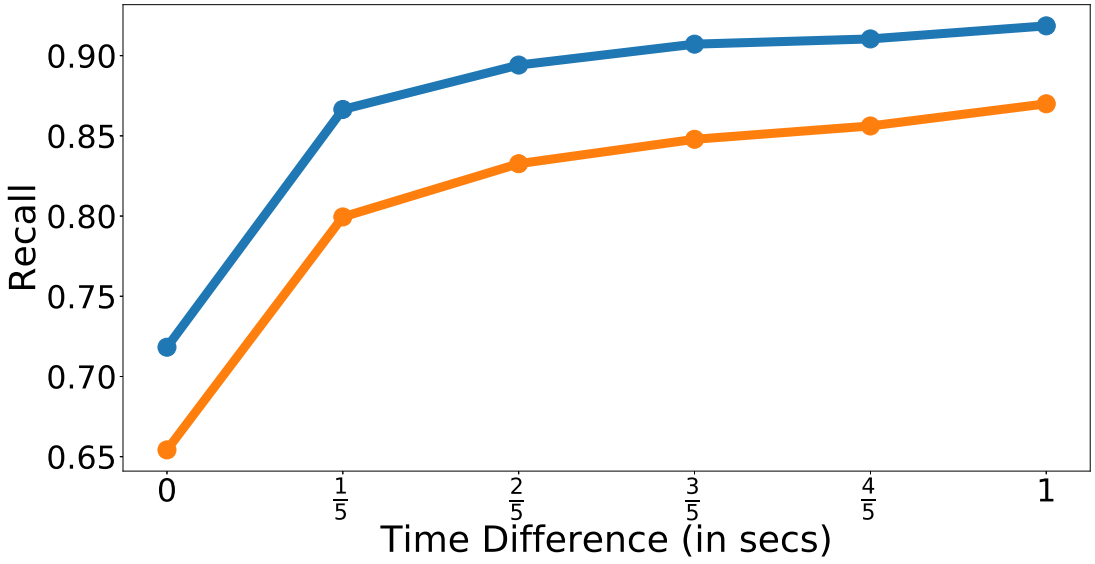

Figure 8.5: Recall of hand pose detection. The horizontal axis is the time difference between the frame selected by a naive rater and the automatic detection. The vertical axis is the percent of recall. The blue and orange lines indicate the same distinction between 25 and 35 poses as in Figure 8.4.

signaler has the initial goal and tells the avatar what to do, but when ambiguities arise the initiative shifts and the avatar asks the human for clarification. Social cues make this process flow naturally. Overall, we demonstrate an example of peer-to-peer cooperation between people and machines, through shared perception and perceptually-grounded reasoning.

The HAB system is still under development and does not make full use of the technology presented earlier in the thesis. Currently, we are only using focus channels on depth modality for gesture recognition. We would like to replace the current gesture recognition system with CANet as part of future work. CANet would be a better choice in this context as CANet uses a single CNN for all channels, thus eliminating the need for multiple GPUs for gesture recognition. HAB does demonstrate however, how gesture recognition systems like the ones developed in earlier chapters can be embedded in real-time systems to revolutionize the field of human-computer interactions.

Chapter 9

Conclusion

9.1 Summary

Gestures are an important form of communication, and gesture recognition is an important application area for computer vision. Using the ChaLearn IsoGD, ChaLearn ConGD and NVIDIA datasets as benchmarks, this thesis shows that recognition accuracy is significantly improved if convolutional channels are used to focus attention on important regions within the scene. In particular, much of the information in gestures is in the hands, and focusing attention on the hands raise recognition rates.

This thesis introduces two different architectures to focus attention on the hands. The FOANet architecture divides processing into 12 channels - one channel for each combination of four modalities (RGB, depth, RGB Flow, and depth flow) and three attention channels (global, left hand, right hand). FOANet's focus channels (right hand and left hand) focuses on the hands by processing cropped regions of the hands.

CANet, on the other hand, merges the attention channels for a single modality into a single network while still preserving attention. Focus layers of CANet zero out the features that doesn't correspond to the respective hand by multiplying the feature maps with hand masks. Focus layers in CANet allow CANet to focus on the hands without having dedicated channels for attention. As the attention channels are merged together, CANet has only 4 CNNs compared to 12 CNNs in FOANet. So, CANet has reduced weights in addition to less training and inference time making it more efficient than FOANet. CANet also exploits the complementary information in different attention regions resulting in increased accuracy of individual channels and architecture as a whole.

The two architectures proposed in this thesis - CANet and FOANet divide processing into multiple channels. These channels must be fused together for classification. This thesis proposes

a novel fusion method called sparse network fusion. Sparse network fusion intelligently fuses information from multiple channels without overfitting.

Fusing FOANet channels by sparse network fusion raises recognition rates from 67.71% to 82.07% on the IsoGD dataset, and from 83.8% to 91.28% on the more task-specific NVIDIA dataset. Moreover, the accuracy of 91.28% on NVIDIA dataset is better than the human accuracy of 88.4%. FOANet also achieves state-of-the art results on continuous ChaLearn ConGD dataset by increasing the mean Jaccard index to 0.7740 compared to the previous best result of 0.6103. CANet architecture further increases the accuracy of ChaLearn IsoGD dataset achieving an accuracy of 84.24% on validation set and 84.79% on test set.

We also studied the interaction between different temporal fusion strategies and types of channels. The results on continuous gesture datasets show that the best temporal fusion strategies in multi-channel networks depend on the modality (RGB vs depth vs flow field) and target (global vs left hand vs right hand) of the channel. More specifically, global channels perform best when fused using Gaussian Pooling, focused (left hand or right hand) flow field channels perform best when temporally fused using LSTMs, and focused RGB and depth channels achieves maximum accuracy when temporally fused by Late Pooling.

We also examined the reason for the superior performance of multi-channel architectures over single channel ones. This thesis compares two possible explanations for the success of multi-channel systems. The Bagging hypothesis suggests that the benefit is from averaging the results of unbiased classifiers. In this model, every channel is general, and the goal of differentiating the channel inputs is to prevent redundancy. The Society of Experts (SoE) hypothesis suggests that channels specialize, with each channel becoming expert at a different aspects of the data. In this model, the benefit comes from selectively fusing information across experts. We explore these hypotheses in the context of FOANet and find evidence that the SoE hypothesis is a better description of FOANet than the Bagging hypothesis. Our experiments suggest that channels specialize to specific sources of information, making them more or less relevant to specific types of gestures.

Finally, we showed a real world application of the techniques discussed in this thesis in the context of Communication with Computers (CWC) program. We presented a system (HAB) capable of peer-to-peer communication between people and avatars in the context of a shared perceptual task. In HAB, people communicate with avatars using gestures and words, and avatars communicate back through gestures, words, and actions. Together, they complete tasks through mixed initiative conversations. Gesture recognition is an important module of HAB system and we also evaluated the performance of gesture recognition module on naturally occurring gestures.

9.2 Future Work

The work presented in thesis achieved state-of-the-art results on multiple gesture recognition datasets. There are many possible extensions that might further improve the state-of-the-art on ChaLearn and NVIDIA datasets while also advancing the field of gesture recognition.

The two architectures proposed in this thesis (FOANet and CANet) use ResNet-50 as the default CNN architecture to process gesture windows. However, architectures such as ResNet-50 (2D CNNs) are not suitable to process videos for action/gesture recognition. 3D CNNs often outperform 2D CNNs on many action recognition tasks [116] and the performance of FOANet and CANet can be further improved by replacing ResNet-50 with a 3D convolution network.

Spatial attention is one of the main themes of this thesis and we showed that recognition accuracies improve significantly by focusing attention on hands. In this thesis, hands are detected either using Faster R-CNN or HandSegNet and is considered as a preprocessing step. However, we hypothesize that the features required to detect hands might also be useful to classify gestures and vice versa. So, hand detection and gesture recognition networks can be combined together resulting in a gesture recognition architecture with self attention mechanism that can do gesture recognition by automatically focusing on the hands. Such network can be trained end-to-end and might result in better gesture recognition performance. Moreover, such network will have less parameters and faster inference times.

CANet loses to FOANet on three right hand channels. One possible reason for this might be the influence of background on hand features due to the receptive field that increases with the depth of the network. Skip connections of CANet are introduced to decrease the effect of background on focus features. These skip connections are introduced very early in the network (after Block 1 of ResNet). Although the skip connections are introduced early, the receptive field at this stage is 40 pixels which is greater than the average size of the hand. So, there is already an influence of background on hand features in the first skip connection. Future work to reduce this background influence will further increase the performance of CANet. One possible way to achieve this might be to have convolution operations that are sensitive to the hand boundaries.

Another possible idea to reduce the background influence on CANet focus features might be an early fusion strategy that is inspired by biological perception - foveal and peripheral vision in the eye. The center of the retina, called the fovea, is populated with cones. So, the part of the scene where we focus our attention has the highest possible visual acuity and color sensitivity (foveal vision) [94]. The other parts of the visual field are filled with rods and have surprisingly low visual acuity (peripheral vision). This low acuity peripheral vision is used to understand the scene, and detect movement. Inspired by this model, the input image to CANet can be represented as a foveal image by encoding the hands as a fovea (placed in the focus of attention) surrounded by a set of concentric rings with decreasing resolution. As the background in such images has very low resolution ,the influence of background on focus features further decreases.

The work in this thesis achieves state-of-the-art results on multiple gesture recognition datasets. A natural extension to this work is action recognition domain as gestures and actions are related. Extending this work to action recognition domain poses interesting challenges as there might be regions other than hands that are important in action recognition domain and these regions might be dependent on action being recognized. For example, legs might be important for a kick action whereas hands might be important for punch action. So, a self attention mechanism that focuses on regions of the scene based on the action being recognized would be a very interesting extension of this work.

Bibliography

[1] *PaddlePaddle*. http://paddlepaddle.org/docs/develop/book/.

[2] Linda Acredolo and Susan Goodwyn. Symbolic gesturing in normal infants. *Child development*, pages 450–466, 1988.

[3] Elizabeth Bates. *The emergence of symbols: Cognition and communication in infancy*. Academic Press, 2014.

[4] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. *Computer vision–ECCV 2006*, pages 404–417, 2006.

[5] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[6] Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. *Computer Vision-ECCV 2004*, pages 25–36, 2004.

[7] Alfredo Canziani, Adam Paszke, and Eugenio Culurciello. An analysis of deep neural network models for practical applications. *arXiv preprint arXiv:1605.07678*, 2016.

[8] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR*, 2017.

[9] Chen Chen, Roozbeh Jafari, and Nasser Kehtarnavaz. Utd-mhad: A multimodal dataset for human action recognition utilizing a depth camera and a wearable inertial sensor. In *Image Processing (ICIP), 2015 IEEE International Conference on*, pages 168–172. IEEE, 2015.

[10] Hong Cheng, Lu Yang, and Zicheng Liu. Survey on 3d hand gesture recognition. *IEEE Transactions on Circuits and Systems for Video Technology*, 26(9):1659–1673, 2016.

[11] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using

rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[12] Necati Cihan Camgoz, Simon Hadfield, and Richard Bowden. Particle filter based probabilistic forced alignment for continuous gesture recognition. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[13] Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV*, volume 1, pages 1–2. Prague, 2004.

[14] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[15] DARPA. Communicating with computers (cwc). https://www.darpa.mil/program/communicating-with-computers, 2015.

[16] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[17] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634, 2015.

[18] Paul Ekman and Wallace V Friesen. The repertoire of nonverbal behavior: Categories, origins, usage, and coding. *semiotica*, 1(1):49–98, 1969.

[19] Hugo Jair Escalante, Víctor Ponce-López, Jun Wan, Michael A Riegler, Baiyu Chen, Albert Clapés, Sergio Escalera, Isabelle Guyon, Xavier Baró, Pål Halvorsen, et al. Chalearn

joint contest on multimedia challenges beyond visual analysis: An overview. In *Pattern Recognition (ICPR), 2016 23rd International Conference on*, pages 67–73. IEEE, 2016.

[20] Sergio Escalera, Xavier Baró, Jordi Gonzalez, Miguel Ángel Bautista, Meysam Madadi, Miguel Reyes, Víctor Ponce-López, Hugo Jair Escalante, Jamie Shotton, and Isabelle Guyon. Chalearn looking at people challenge 2014: Dataset and results. In *ECCV Workshops (1)*, pages 459–473, 2014.

[21] Sergio Escalera, Jordi Gonzàlez, Xavier Baró, Miguel Reyes, Isabelle Guyon, Vassilis Athitsos, Hugo Escalante, Leonid Sigal, Antonis Argyros, Cristian Sminchisescu, et al. Chalearn multi-modal gesture recognition 2013: grand challenge and workshop summary. In *Proceedings of the 15th ACM on International conference on multimodal interaction*, pages 365–368. ACM, 2013.

[22] Christoph Feichtenhofer, Axel Pinz, and Richard Wildes. Spatiotemporal residual networks for video action recognition. In *Advances in Neural Information Processing Systems*, pages 3468–3476, 2016.

[23] Christoph Feichtenhofer, Axel Pinz, and Richard P Wildes. Spatiotemporal multiplier networks for video action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[24] Christoph Feichtenhofer, Axel Pinz, Richard P Wildes, and Andrew Zisserman. What have we learned from deep representations for action recognition? *arXiv preprint arXiv:1801.01415*, 2018.

[25] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1933–1941, 2016.

[26] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[27] HS Friedman. The concept of skill in nonverbal communication: Implications for understanding social interaction. *Skill in nonverbal communication*, pages 2–27, 1979.

[28] Philip Gather, Martha Wagner Alibali, and Susan Goldin-Meadow. Knowledge conveyed in gesture is not tied to the hands. *Child development*, 69(1):75–84, 1998.

[29] Felix A Gers, Nicol N Schraudolph, and Jürgen Schmidhuber. Learning precise timing with lstm recurrent networks. *Journal of machine learning research*, 3(Aug):115–143, 2002.

[30] Maureen Gillespie, Ariel N James, Kara D Federmeier, and Duane G Watson. Verbal working memory predicts co-speech gesture: Evidence from individual differences. *Cognition*, 132(2):174–180, 2014.

[31] Susan Goldin-Meadow. The role of gesture in communication and thinking. *Trends in cognitive sciences*, 3(11):419–429, 1999.

[32] Susan Goldin-Meadow and Cynthia Butcher. Pointing toward two-word speech in young children. *Pointing: Where language, culture, and cognition meet*, pages 85–107, 2003.

[33] Will Goldstone. *Unity Game Development Essentials*. Packt Publishing Ltd, 2009.

[34] Melvyn A Goodale and A David Milner. Separate visual pathways for perception and action. *Trends in neurosciences*, 15(1):20–25, 1992.

[35] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[36] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[37] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE, 2013.

[38] Isabelle Guyon, Vassilis Athitsos, Pat Jangyodsuk, and Hugo Jair Escalante. The chalearn gesture dataset (cgd 2011). *Machine Vision and Applications*, 25(8):1929–1951, 2014.

[39] Isabelle Guyon, Vassilis Athitsos, Pat Jangyodsuk, Hugo Jair Escalante, and Ben Hamner. Results and analysis of the chalearn gesture challenge 2012. In *Advances in Depth Image Analysis and Applications*, pages 186–204. Springer, 2013.

[40] Isabelle Guyon, Vassilis Athitsos, Pat Jangyodsuk, Ben Hamner, and Hugo Jair Escalante. Chalearn gesture challenge: Design and first results. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, pages 1–6. IEEE, 2012.

[41] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Hypercolumns for object segmentation and fine-grained localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 447–456, 2015.

[42] Haitham Hasan and Sameem Abdul-Kareem. Human-computer interaction using vision-based hand gesture recognition systems: a survey. *Neural computing & applications*, 25(2), 2014.

[43] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[44] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pages 630–645. Springer, 2016.

[45] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[46] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. *arXiv preprint arXiv:1709.01507*, 2017.

[47] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, volume 1, page 3, 2017.

[48] Jana M Iverson, Olga Capirci, and M Cristina Caselli. From communication to language in two modalities. *Cognitive development*, 9(1):23–43, 1994.

[49] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, pages 2017–2025, 2015.

[50] William James. *The principles of psychology*. Read Books Ltd, 2013.

[51] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[52] Adam Kendon. Gesture. *Annual review of anthropology*, 26(1):109–128, 1997.

[53] Adam Kendon. *Gesture: Visible Action as Utterance*. Cambridge University Press, 2004.

[54] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[55] Robert M Krauss, Yihsiu Chen, and Rebecca F Gotfexnum. 13 lexical gestures and lexical access: a process model. *Language and gesture*, 2:261, 2000.

[56] Nikhil Krishnaswamy and James Pustejovsky. Multimodal semantic simulations of linguistically underspecified motion events. In *Spatial Cognition X: International Conference on Spatial Cognition*. Springer, 2016.

[57] Nikhil Krishnaswamy and James Pustejovsky. VoxSim: A visual platform for modeling motion language. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. ACL, 2016.

[58] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[59] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. Hmdb: a large video database for human motion recognition. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2556–2563. IEEE, 2011.

[60] Dennis Küster, Eva Krumhuber, and Arvid Kappas. Nonverbal behavior online: A focus on interactions with and via artificial agents and avatars. In *The Social Psychology of Nonverbal Communication*, pages 272–302. Springer, 2015.

[61] Edwin H Land and John J McCann. Lightness and retinex theory. *Josa*, 61(1):1–11, 1971.

[62] Ivan Laptev, Marcin Marszalek, Cordelia Schmid, and Benjamin Rozenfeld. Learning realistic human actions from movies. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[63] Hugo Larochelle and Geoffrey E Hinton. Learning to combine foveal glimpses with a third-order boltzmann machine. In *Advances in neural information processing systems*, pages 1243–1251, 2010.

[64] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[65] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. In *Artificial Intelligence and Statistics*, pages 562–570, 2015.

[66] Wanqing Li, Zhengyou Zhang, and Zicheng Liu. Action recognition based on a bag of 3d points. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*, pages 9–14. IEEE, 2010.

[67] Zhipeng Liu. Chalearn2017_isolated_gesture. https://github.com/ZhipengLiu6/Chalearn2017_isolated_gesture, 2017.

[68] Zhipeng Liu, Xiujuan Chai, Zhuang Liu, and Xilin Chen. Continuous gesture recognition with hand-oriented spatiotemporal feature. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3056–3064, 2017.

[69] Zhipeng Liu, Xiujuan Chai, Zhuang Liu, and Xilin Chen. Continuous gesture recognition with hand-oriented spatiotemporal feature. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[70] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[71] Marcin Marszałek, Ivan Laptev, and Cordelia Schmid. Actions in context. In *IEEE Conference on Computer Vision & Pattern Recognition*, 2009.

[72] David McDonald and James Pustejovsky. On the representation of inferences and their lexicalization. In *Advances in Cognitive Systems*, volume 3, 2014.

[73] David McNeill. *Psycholinguistics: A new approach.* Harper & Row Publishers, 1987.

[74] David McNeill. *Hand and mind: What gestures reveal about thought*. University of Chicago press, 1992.

[75] David McNeill. *Gesture & Thought*. University of Chicago Press, 2005.

[76] Quinn McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157, 1947.

[77] Qiguang Miao, Yunan Li, Wanli Ouyang, Zhenxin Ma, Xin Xu, Weikang Shi, Xiaochun Cao, Zhipeng Liu, Xiujuan Chai, Zhuang Liu, et al. Multimodal gesture recognition based on the resc3d network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3047–3055, 2017.

[78] David Milner and Mel Goodale. *The visual brain in action*. Oxford University Press, 2006.

[79] Marvin Lee Minsky. *The Society of Minds*. Simon Schuster, 1986.

[80] Pavlo Molchanov, Shalini Gupta, Kihwan Kim, and Jan Kautz. Hand gesture recognition with 3d convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 1–7, 2015.

[81] Pavlo Molchanov, Shalini Gupta, Kihwan Kim, and Kari Pulli. Multi-sensor system for driver's hand-gesture recognition. In *Automatic Face and Gesture Recognition (FG), 2015 11th IEEE International Conference and Workshops on*, volume 1, pages 1–8. IEEE, 2015.

[82] Pavlo Molchanov, Xiaodong Yang, Shalini Gupta, Kihwan Kim, Stephen Tyree, and Jan Kautz. Online detection and classification of dynamic hand gestures with recurrent 3d convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4207–4215, 2016.

[83] Joel Ruben Antony Moniz and David Krueger. Nested lstms. *arXiv preprint arXiv:1801.10308*, 2018.

[84] Donald F Moores. Nonvocal systems of verbal behavior. *Language perspectives: Acquisition, retardation, and intervention*, pages 377–417, 1974.

[85] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 4694–4702. IEEE, 2015.

[86] Juan Carlos Niebles, Chih-Wei Chen, and Li Fei-Fei. Modeling temporal structure of decomposable motion segments for activity classification. In *European conference on computer vision*, pages 392–405. Springer, 2010.

[87] Eshed Ohn-Bar and Mohan Manubhai Trivedi. Hand gesture recognition in real time for automotive interfaces: A multimodal vision-based approach and evaluations. *IEEE transactions on intelligent transportation systems*, 15(6):2368–2377, 2014.

[88] Omar Oreifej and Zicheng Liu. Hon4d: Histogram of oriented 4d normals for activity recognition from depth sequences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 716–723, 2013.

[89] Deepak Pathak, Ross Girshick, Piotr Dollár, Trevor Darrell, and Bharath Hariharan. Learning features by watching objects move. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[90] Alonso Patron-Perez, Marcin Marszalek, Andrew Zisserman, and Ian D Reid. High five: Recognising human interactions in tv shows. In *BMVC*, volume 1, page 2. Citeseer, 2010.

[91] Lionel Pigou, Sander Dieleman, Pieter-Jan Kindermans, and Benjamin Schrauwen. Sign language recognition using convolutional neural networks. In *Workshop at the European Conference on Computer Vision*, pages 572–578. Springer, 2014.

[92] Lionel Pigou, Mieke Van Herreweghe, and Joni Dambre. Gesture and sign language recognition with temporal residual networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3086–3093, 2017.

[93] Stephen E. Plamer. *Vision Science: Photons to Phenomenology*. MIT Press, 1999.

[94] Jan M Provis, Adam M Dubis, Ted Maddess, and Joseph Carroll. Adaptation of the central retina for high acuity vision: cones, the fovea and the avascular zone. *Progress in retinal and eye research*, 35:63–81, 2013.

[95] James Pustejovsky. The generative lexicon. *Computational linguistics*, 17(4):409–441, 1991.

[96] James Pustejovsky. Dynamic event structure and habitat theory. In *Proceedings of the 6th International Conference on Generative Approaches to the Lexicon (GL2013)*, pages 1–10. ACL, 2013.

[97] James Pustejovsky and Nikhil Krishnaswamy. VoxML: A visualization modeling language. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, May 2016. European Language Resources Association (ELRA).

[98] James Pustejovsky and Jessica Moszkowicz. The qualitative spatial dynamics of motion. *The Journal of Spatial Cognition and Computation*, 2011.

[99] Zenon W Pylyshyn and Ron W Storm. Tracking multiple independent targets: Evidence for a parallel tracking mechanism. *Spatial vision*, 3(3):179–197, 1988.

[100] Tapani Raiko, Harri Valpola, and Yann LeCun. Deep learning made easier by linear transformations in perceptrons. In *Artificial Intelligence and Statistics*, pages 924–932, 2012.

[101] Siddharth S Rautaray and Anupam Agrawal. Vision based hand gesture recognition for human computer interaction: a survey. *Artificial Intelligence Review*, 43(1):1–54, 2015.

[102] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[103] Brian J Scholl and Zenon W Pylyshyn. Tracking multiple items through occlusion: Clues to visual objecthood. *Cognitive psychology*, 38(2):259–290, 1999.

[104] Brian J Scholl, Zenon W Pylyshyn, and Jacob Feldman. What is a visual object? evidence from target merging in multiple object tracking. *Cognition*, 80(1-2):159–177, 2001.

[105] Pierre Sermanet, Koray Kavukcuoglu, Soumith Chintala, and Yann LeCun. Pedestrian detection with unsupervised multi-stage feature learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3626–3633, 2013.

[106] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[107] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, pages 568–576, 2014.

[108] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[109] Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks. In *Advances in neural information processing systems*, pages 2377–2385, 2015.

[110] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, pages 4278–4284, 2017.

[111] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[112] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[113] Bernard T Tervoort. Esoteric symbolism in the communication behavior of young deaf children. *American Annals of the deaf*, 1961.

[114] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015.

[115] Du Tran, Jamie Ray, Zheng Shou, Shih-Fu Chang, and Manohar Paluri. Convnet architecture search for spatiotemporal feature learning. *arXiv preprint arXiv:1708.05038*, 2017.

[116] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6450–6459, 2018.

[117] twentybn. The 20bn-jester dataset. https://www.twentybn.com/datasets/jester, 2017.

[118] Jun Wan, Sergio Escalera, A Gholamreza, Hugo Jair Escalante, Xavier Baró, Isabelle Guyon, Meysam Madadi, A Juri, G Jelena, L Chi, et al. Results and analysis of chalearn lap multi-modal isolated and continuous gesture recognition, and real versus fake expressed emotions challenges. In *ChaLearn LaP, Action, Gesture, and Emotion Recognition Workshop and Competitions: Large Scale Multimodal Gesture Recognition and Real versus Fake expressed emotions, ICCV*, volume 4, 2017.

[119] Jun Wan, Yibing Zhao, Shuai Zhou, Isabelle Guyon, Sergio Escalera, and Stan Z Li. Chalearn looking at people rgb-d isolated and continuous datasets for gesture recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 56–64, 2016.

[120] Heng Wang, Alexander Kläser, Cordelia Schmid, and Cheng-Lin Liu. Action recognition by dense trajectories. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3169–3176. IEEE, 2011.

[121] Heng Wang, Alexander Kläser, Cordelia Schmid, and Cheng-Lin Liu. Dense trajectories and motion boundary descriptors for action recognition. *International journal of computer vision*, 103(1):60–79, 2013.

[122] Heng Wang, Dan Oneata, Jakob Verbeek, and Cordelia Schmid. A robust and efficient video representation for action recognition. *International Journal of Computer Vision*, 119(3):219–238, 2016.

[123] Huogen Wang, Pichao Wang, Zhanjie Song, and Wanqing Li. Large-scale multimodal gesture recognition using heterogeneous networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3129–3137, 2017.

[124] Huogen Wang, Pichao Wang, Zhanjie Song, and Wanqing Li. Large-scale multimodal gesture segmentation and recognition based on convolutional neural networks. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[125] Isaac Wang, Mohtadi Ben Fraj, Pradyumna Narayana, Dhruva Patil, Gururaj Mulay, Rahul Bangar, J Ross Beveridge, Bruce A Draper, and Jaime Ruiz. Eggnog: A continuous, multimodal data set of naturally occurring gestures with ground truth labels. In *Automatic Face & Gesture Recognition (FG 2017), 2017 12th IEEE International Conference on*, pages 414–421. IEEE, 2017.

[126] Isaac Wang, Pradyumna Narayana, Dhruva Patil, Gururaj Mulay, Rahul Bangar, Bruce Draper, Ross Beveridge, and Jaime Ruiz. Exploring the use of gesture in collaborative tasks. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, pages 2990–2997. ACM, 2017.

[127] Jiang Wang, Zicheng Liu, Jan Chorowski, Zhuoyuan Chen, and Ying Wu. Robust 3d action recognition with random occupancy patterns. In *Computer vision–ECCV 2012*, pages 872–885. Springer, 2012.

[128] Jiang Wang, Zicheng Liu, Ying Wu, and Junsong Yuan. Mining actionlet ensemble for action recognition with depth cameras. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1290–1297. IEEE, 2012.

[129] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *European Conference on Computer Vision*, pages 20–36. Springer, 2016.

[130] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4724–4732, 2016.

[131] Jacob O. Wobbrock, Meredith Ringel Morris, and Andrew D. Wilson. User-defined Gestures for Surface Computing. CHI '09, pages 1083–1092, New York, NY, USA, 2009. ACM.

[132] Songfan Yang and Deva Ramanan. Multi-scale recognition with dag-cnns. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1215–1223, 2015.

[133] Xiaodong Yang and YingLi Tian. Super normal vector for human activity recognition with depth cameras. *IEEE transactions on pattern analysis and machine intelligence*, 39(5):1028–1039, 2017.

[134] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4694–4702, 2015.

[135] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[136] Liang Zhang, Guangming Zhu, Peiyi Shen, Juan Song, Syed Afaq Shah, and Mohammed Bennamoun. Learning spatiotemporal features using 3dcnn and convolutional lstm for ges-

ture recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3120–3128, 2017.

[137] Zhengyou Zhang. Microsoft kinect sensor and its effect. *IEEE MulitMedia*, 19:4–10, 2012.

[138] Christian Zimmermann and Thomas Brox. Learning to estimate 3d hand pose from single rgb images. In *IEEE International Conference on Computer Vision (ICCV)*, 2017. https://arxiv.org/abs/1705.01389.