Fundamental Algorithmic Solutions Informed by Cognitive Assistance for First Responders

A

Thesis

Presented to

the faculty of the School of Engineering and Applied Science

University of Virginia

in partial fulfillment

of the requirements for the degree

Doctor of Philosophy

by

Lahiru Nuwan Wijayasingha

August 2024

APPROVAL SHEET

This

Thesis

is submitted in partial fulfillment of the requirements
for the degree of

Doctor of Philosophy

Author: Lahiru Nuwan Wijayasingha

This Thesis has been read and approved by the examing committee:

Advisor: John A. Stankovic

Committee Member: Lu Feng

Committee Member: Brad Campbell

Committee Member: Seongkook Heo

Committee Member: Homa Alemzadeh

Accepted for the School of Engineering and Applied Science:

[illegible]

Jennifer L. West, School of Engineering and Applied Science

August 2024

Fundamental Algorithmic Solutions Informed by Cognitive Assistance for First Responders

by

Lahiru Wijayasingha

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
in the

Department of Computer Science
University of Virginia
2024

Doctoral Committee:

Advisor	Professor Emeritus	John A Stankovic
Committee Chair	Associate Professor	Lu Feng
	Assistant Professor	Bradford Campbell
	Assistant Professor	Homa Alemzadeh
	Assistant Professor	Seongkook Heo

ABSTRACT

Emergency Medical Service (EMS) Providers perform various emergency protocols on their patients under stressful and noisy conditions. We introduce several fundamental algorithmic solutions which will be useful in creating a cognitive assistant that can be used by the EMS providers to solve several challenges they face. Cardiac arrest is one of the most critical EMS protocols. Cardiopulmonary resuscitation (CPR) is arguably the most important step in cardiac arrest-based emergency protocols. High-quality CPR with the appropriate compression rate and depth is essential for the survival of cardiac arrest patients. In reality, the CPR administered in many cases can be of low quality due to the suboptimal quality of feedback the EMS providers receive during training which is caused by human biases and errors. There is a need for automatic estimation of CPR depth and rate which is critical in providing real-time feedback to the EMS providers. We collected a multi-modal dataset of 24 participants performing CPR which can be used to develop, evaluate, and test algorithms that can be used to estimate CPR performance by estimating the CPR compression depth and rate. We showed that the inertial sensors in a smartwatch can be used to reliably estimate CPR compression depth and rate. It is preferable to estimate the CPR quality using cameras as well since some EMS providers might not be wearing a smartwatch all the time. We showed that a single camera can be used to reliably estimate the CPR rate. However, estimating the CPR depth requires improvements in monocular depth perception. We showed that defocus blur can be used to accurately estimate depth in general scenes. Any visual depth perception technique is sensitive to the changes of the camera. We developed a novel technique to estimate camera parameters of a given new camera which can be used to eliminate this camera dependency. Therefore our depth estimation technique can be used by a range of different cameras. We collected a second dataset which consists of hand images and their depth maps. This dataset was used to show that our defocus blur-based method can be used to reliably estimate the distance to the hands from the camera. We used this technique to estimate CPR depth. The depth estimation errors were higher than the results from the smartwatch and we identified possible causes for these errors so they can be alleviated in future research. We introduce a novel few-shot-learning technique that can be used to detect activities with just a very small amount of data. This can be used to detect various EMS steps in the future. Our noise-robust emotion recognition techniques can be used to detect vocal emotions under environmental noise. These vocal emotion recognition methods can be utilized in the future by a cognitive assistant to detect if an EMS provider is stressed and potentially provide feedback to alleviate the stress.

ACKNOWLEDGEMENTS

Throughout my PhD I received help from many. First I would like to thank my advisor Professor John Stankovic for the enormous support and the guidance throughout the whole process. He helped on how to tackle problems in a research oriented mind set and helped me with my focus. He helped me through a pandemic and many other many issues I faced. Without Professor Stankovic, I could not have completed my PhD.

I am profoundly grateful to all the committee members for their unwavering support and insightful feedback from the day I submitted my proposal to the completion of my degree. A special thanks to Professor Alemzadeh for her dedicated collaboration on the EMS-related work.

I extend my heartfelt gratitude to the University of Virginia Computer Science and Link Lab communities for their unwavering support and camaraderie, which were invaluable during challenging times like the pandemic.

Lastly, I want to extend my heartfelt thanks to my parents for molding me into the person I am today.

Table of Contents

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

1.1 EMS providers

Emergency Medical Services (EMS) providers are a critical component in the healthcare system. They are trained to quickly respond to emergency calls in the community and provide pre-hospital care for patients with illnesses and injuries and prepare them for transport if necessary. EMS providers physically examine the patient and assess the condition to decide on a course of treatment. They may use a wide array of medical equipment to assess or treat patients. EMS providers may administer medications, provide life care, cardiovascular life support, or other important medical procedures.

Over 30 million emergency medical incidents are reported every year in the U.S., and nearly 70% of the fire department calls were for medical emergencies [1][2]. Various EMS organizations have put forth guidelines for EMS providers, administration, and medical direction on treatment, transportation, and management of patients under many medical emergencies. The purpose of these protocol guidelines is to provide excellent prehospital care for patients. According to the Old Dominion EMS Alliance (ODEMSA) [3], the responsibility for the EMS care is shared between three parties; the Operational Medical Director (OMD), the online Medical Control physician (MC), and the EMS provider. EMS providers are allowed to provide a set of authorized services to the patients while certain services are allowed only after approval by the MC. Becoming a qualified EMS provider is very involved and requires continuous training and EMS certifications to keep up with the latest medical practices and technologies. They have to be proficient in various emergency protocols including cardiovascular emergencies, medical, trauma, toxicology, etc.

1.1.1 Problems faced by EMS providers

Due to the nature of the EMS work, the providers are facing several hardships related to their work. In this dissertation, we focus on several critical challenges that need to be addressed to ensure the well-being of EMS providers and to support an excellent level of patient care.

1.1.2 Working conditions are stressful

Due to the hazardous and hectic working environments and shift work nature, EMS providers have to regularly work under stress [4] [5]. Various levels of anxiety, anger, and depression were detected to be caused by the stress experienced by the EMS providers [6]. A national survey involving 658 emergency medical technicians performed by Cydulka et al. [7] found very high-stress levels in EMS providers. These stress levels manifested as somatic stress (stress that would affect bodily systems), organizational stress (stress experienced as a result of working conditions), and job dissatisfaction. High levels of continuous emotions such as anger, fear, anxiety, and guilt can result in mental health problems and changes in corticosteroid levels in blood which can cause physiological stress and immune response [8]. Previous attempts taken in order to relieve the stress in the work life of EMS, such as introducing new schedule patterns, had no significant positive results [9]. Careful attention must be paid by EMS organizations and EMS medical directors at an individual level to mitigate this issue.

1.1.3 EMS protocols are complicated and difficult to perfect

Many emergency protocols are complicated, difficult to follow and memorize, and require thorough and continuous training. However, it is critical that the first responders perform the medical procedures with the correct techniques and protocols. Let's take a cardiac arrest situation as an example. EMS personnel will arrive at the scene and start the necessary medical procedures to support the patient. EMS protocol for cardiac arrest is complicated and can include interventions such as ECG, ventilation, cardiac monitoring, chest decompression, Cardiopulmonary resuscitation (CPR), defibrillation, and intubation. High quality, timely CPR and attempted defibrillation are paramount to successful cardiac life support and recovery of the patient [3] [10]. It is crucial to provide high-quality CPR with the appropriate amount of chest compression rate with complete chest recoil, correct depth, and minimal interruption of compressions [11]. Adherence to the correct techniques and protocols during medical procedures such as Cardio-pulmonary Resuscitation (CPR) is critical in saving the lives of patients [10]. But it is sometimes difficult to both memorize the protocol steps and deliver CPR according to the accepted guidelines [12]. For example, CPR delivery is often sub-optimal in practice because the compressions are too shallow or deep and the frequency of the compressions can be inadequate. It has been shown that successfully administering EMS protocols requires tedious and continuous training which involves accurately following the required steps and developing the talents required to administer proper CPR which is difficult with the hectic working schedule of the EMS providers and lack of proper feedback.

1.2 Cognitive assistant as a solution for the EMS challenges

A cognitive assistant that utilizes many sensors and AI/ML techniques can be used to alleviate many of the problems faced by the EMS providers that were mentioned before. Ongoing work on cognitive assistants can be divided into two main fronts: application and algorithmic foundation areas. The work on application areas tends to build pipelines that integrate sensors and various techniques to build overall solutions. Algorithmic foundation work tends to solve generic and fundamental problems that can be integrated into the pipelines for cognitive assistants. **The overarching goal of this thesis is to contribute to the algorithmic foundations for EMS providers.** It is beyond the scope of this thesis to incorporate them into the full pipelines of a cognitive assistant. With recent swift advancements of AI, medical sciences experience benefits along many avenues of AI-supported disease diagnosis, biomedical information processing, and intelligent assistance [13]. Intelligent cognitive assistants are being developed to enhance human capabilities and to reduce cognitive load experienced by humans especially when they are working in complicated environments and tasks. For example, Wolters et al. [14] developed such a system to help people with dementia in their daily activities. A cognitive assistant system called CLAICA was developed to help human workers process a large amount of information efficiently [15]. Angulo et al. describes another cognitive assistant system that can help human operators process waste amounts of information generated by various intelligent devices efficiently [16]. Researchers have started developing cognitive assistants for the EMS domain as well. EMS providers are required to fill-out various forms that documents the condition of the patient and the types of interventions performed at an emergency scene. This can increase the cognitive load of the EMS personnel. An NLP-based cognitive system has been proposed by Alemzadeh et al. which can analyze clinical notes, patient records, and clinical guidelines to identify the state of a disease [17]. Methods to extract EMS-related information from patient care reports were developed by Kim et al. [18]. Cognitive EMS was developed by Preum et al. [19] to extract EMS-related information from the voice which can be used to help EMS providers. GRACE is a cognitive assistant that was developed to automate the process of form filling using speech transcripts recorded from conversations among the EMS providers [20]. emsReACT [21] and EMSContExt [22] was developed to analyze real-time audio transcripts to give feedback on various medical protocols. A cognitive assistant system that uses a smart glass to perform real-time data collection and analysis was developed to perform voice analysis and to provide voice-based feedback was developed [23]. Shu et al. [24] propose to use a mobile rescue robot as a virtual assistant that can perform speech recognition and use Natural Language Processing and machine learning to understand the surrounding of an EMS situation and provide feedback. An end-to-end real-time, multi-modal, wearable virtual assistant was introduced by Weerasinghe et al. [25]. This system consists of AR glasses to collect and analyze data. As seen from the numerous past work, important progress has been made which pushes the field towards a realistic and helpful cognitive assistant. But many open questions remain to be solved.

The future cognitive assistant that we envision is expected to use one or all of the following: a wearable camera, a microphone, and a smartwatch. In addition, it may also utilize an external camera. We investigate various important problems that need algorithmic solutions that must utilize these sensors. For example, a future cognitive assistant may use sensors shown in Figure 1.1.

Cameras will be an integral part of a cognitive assistant solution for EMS providers. Although inertial sensors in smartwatches can be used for activity recognition and evaluating the quality of certain emergency procedures (e.g. measuring the rate and depth of CPR) they have limitations. Cameras can be used, possibly alongside smartwatch-based solutions to perform tasks that cannot be performed with smartwatches and as a parallel modality to improve the performance. Using cameras will provide the following advantages over using smartwatches.

- Facilitate object recognition
- Can be used to recognize people (e.g through facial recognition)
- When combined with depth estimation, can estimate the sizes of various objects
- A camera worn by one EMS provider can be used to estimate the quality of an intervention performed by another EMS provider. This may solve the problem of occlusion often faced by cameras.

Note that cameras also have certain disadvantages such as being bulky, privacy issues, and the possibility of occlusion. Body cameras are widely used by the state and local law enforcement agencies in the United States. This means that the above issues have been alleviated to some extent. Therefore the adoption of cameras by EMS providers may not be far-fetched.

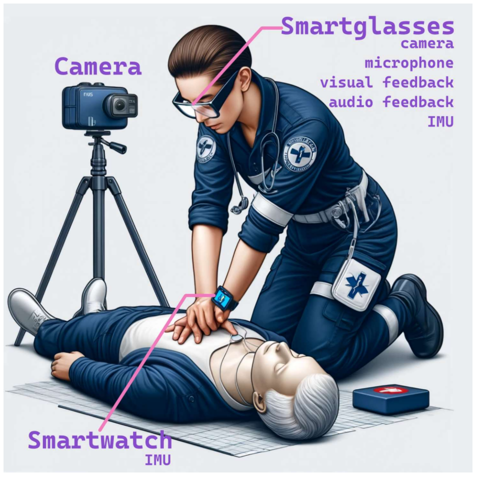


Figure 1.1: Sensors used for a cognitive assistant

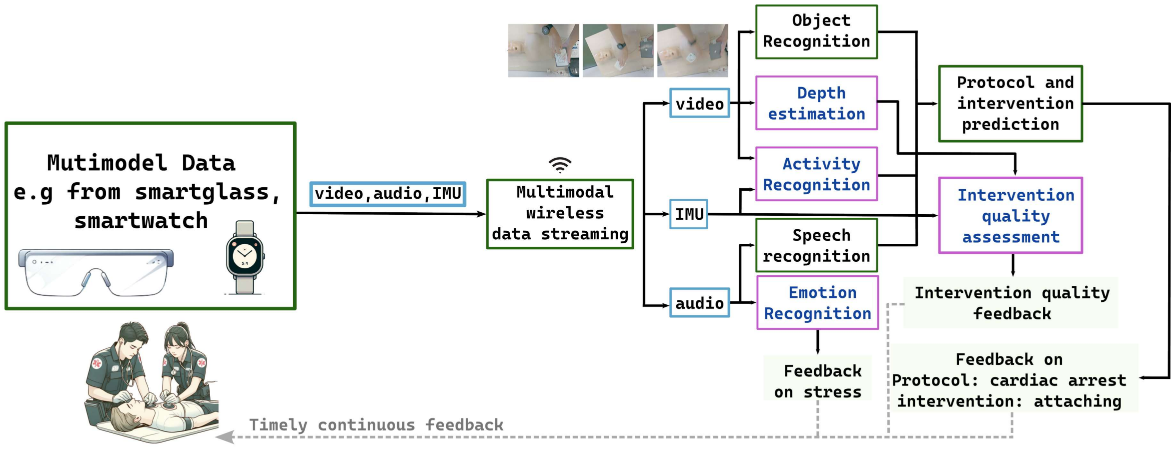

Mutimodel Data
e.g from smartglass, smartwatch
video,audio,IMU
Multimodal wireless data streaming
video
IMU
audio
Object Recognition
Depth estimation
Activity Recognition
Speech recognition
Emotion Recognition
Protocol and intervention prediction
Intervention quality assessment
Intervention quality feedback
Feedback on stress
Feedback on Protocol: cardiac arrest intervention: attaching
Timely continuous feedback

Figure 1.2: Overall Architecture of the vision for a cognitive assistant, adapted from Weerasinghe et al. [25]. The boxes in pink show our contributions.

Out of the many emergency protocols, we identify cardiac arrest as a critical protocol that many lives depend on a daily basis. For this reason, we selected to build a set of technologies that can be part of a cognitive assistant to help EMS providers train cardiac arrest-based medical protocols. We believe that a similar methodology can be utilized to expand the cognitive assistant into other protocols in the future.

1.2.1 Application Challenges

Motivated by the paper by Weerasinghe et al. [25] we propose that a future cognitive assistant may have an architecture as shown in Figure 1.2. The video and audio data from a smart glass, body camera, or external camera will be transmitted to an edge device running inference algorithms. Video feeds have been used for tasks such as object recognition and activity recognition as proposed by the paper. Audio data can be used to perform speech recognition. The results from object and speech recognition can be used for the protocol (e.g. cardiac arrest) and intervention (e.g. CPR) prediction. The predicted protocols and interventions can be used to provide feedback to the EMS providers. The IMU data from smartwatches have been used for recognizing activities [26] and estimating the quality of interventions such as CPR [27]. A cognitive assistant may use the IMU sensors in the smartwatch such as accelerometer and gyroscope to detect activities that are performed by the EMS providers. These sensors will also be used to measure the quality of interventions such as CPR. For example, these sensors will be used to measure the CPR compression frequency and depth. These measurements can be used to provide real-time feedback to the EMS provider who is performing CPR. Depth estimation is an essential capability of a cognitive assistant. Depth estimation can be used for situational awareness (to determine which objects are closer to the EMS provider and which objects are further away), to measure the sizes of various objects (e.g. to measure the size of syringes and to estimate the amount of liquid contained in them) and to measure the quality of certain EMS interventions (e.g. to measure the CPR compression depth). Since EMS providers work in hectic environments, they often suffer from high levels of stress. Voice has been recognized as an important modality to detect stress [28]. It is important to detect the emotional state of EMS providers and provide feedback. This feedback can be used by EMS providers to attempt to reduce their stress (e.g. through breathing techniques) or to exchange with another EMS person. There have been several attempts (e.g. the paper by Weerasinghe et al. [25]) to build a complete cognitive assistant which incorporates a subset of what was described above. However a complete cognitive assistant equipped with all the different capabilities mentioned above with real-time feedback is still lacking. Building such pipelines are out of the scope of this thesis since we focus on solving algorithmic challenges in each of these areas as discussed in the next section. These algorithmic improvements in the future will help researchers to build a more comprehensive cognitive assistant.

1.2.2 Algorithmic Challenges

Five main research gaps were identified as necessary to realize the vision of a cognitive assistant mentioned in the previous section. The gaps, their relationship to EMS, and the novel algorithmic solutions are discussed next.

1.2.2.1 Monocular Depth Perception

A highly accurate visual depth perception is essential for many visual tasks the cognitive assistant will face. For example, when the EMS provider is performing CPR, estimating CPR compression depth with vision demands a highly accurate depth perception method that can be used to measure depth variation of the patient's chest or the hands of the EMS provider. Such solutions can also be utilized to measure the depth of intubation, the amount of compression of bag valve masks and to measure the size of objects such as syringes. Although active methods such as structured light and Time-of-Flight (TOF) sensors can be used to measure depth, these methods require specialized hardware and are power-hungry [29] [30]. Stereo vision can also be used to measure depth in an accurate manner. However, this requires accurate point matching between the two images taken by two different cameras. Accurate depth perception in this manner suffers from errors caused by the point-matching algorithm. Further, the necessity of two cameras adds weight and increases power requirements. Defocus blur can be used to accurately measure depth with just a single camera [31] which is applicable to a future cognitive assistant for EMS providers. These depth estimation techniques are developed for a given specific camera. The models trained with images from one camera will not work on another camera. However different emergency response centers may use different cameras. It is important for our models to be camera-independent. We are the first to introduce such a technique for depth estimation from defocus blur. We introduce a quantity called K_{cam} that encapsulates various camera parameters that affect the defocus blur and introduce a novel technique to estimate the K_{cam} for a new camera which we call ”Defocus Calibration”. This estimated value for K_{cam} can be used in our deep learning model to eliminate the dependency of the model on a specific camera. The outcome is a solution that can be used for depth perception in various first responder sites and using different cameras. This solution can be helpful in other areas such as in augmented reality gaming, estimating depth in microscopic scenes, and in endoscopic videos.

1.2.2.2 EMS related Datasets

Research into quality evaluation for EMS protocols such as cardiac arrest, especially for CPR is lacking [12] [10]. The main reason for this is the clear lack of data to train machine learning models. We collected the first multi-modal dataset of CPR quality evaluation of 24 participants performing 292 CPR sessions of roughly one minute. External cameras and a body camera, and a smartwatch was used to gather data. A depth sensor installed in a CPR manikin was used to obtain ground truth CPR compression depth and rate. This novel dataset can be used by EMS researchers to make progress in estimating CPR quality with smartwatch and computer vision techniques. [add the link]

1.2.2.3 Quality estimation for CPR

CPR is one of the most important steps in cardiac arrest-related EMS protocols. It is critical to maintain the appropriate rate and depth of CPR to achieve high-quality CPR. Developing techniques that can automatically measure the CPR depth and rate would significantly benefit the EMS providers both in training and in the field. We show that our solution allows us to use a smartwatch to measure the CPR quality by estimating depth and rate in an accurate manner (the rate and depth estimation errors were 5.4 compressions/min and 6.6 mm (RMSE)). This means our solution can be used in the future to provide valuable feedback to the EMS responders thereby improving their CPR performance.

Under certain circumstances, EMS providers may not wear a smartwatch and they may already have a camera at the scene (e.g. in a smartglass or as an external camera). It is beneficial to explore the possibility of using a camera to evaluate the CPR performance as well. We developed a novel technique to measure CPR depth and rate with cameras. We estimated the CPR rate by tracking the wrist position on the hand and estimated the CPR depth by estimating the distance to the hand from the camera using the novel defocus blur technique we developed. The average CPR depth estimation accuracy over all the participants and sessions to date is 25.60 mm (RMSE). The average rate estimation error is 16 compressions per minute. This level of error is larger than the smartwatch-based solution. We observed that the performance of the vision-based CPR quality estimation method vary depending on the participant and the CPR rate. Higher CPR rates resulted in a larger error on average for both depth and rate estimation. Under certain conditions (for certain participants and for medium to lower rates), the error becomes significantly lower (less than 5 compressions/min and 11 mm). We have identified several sources of error which resulted in the accuracy drop compared to the smartwatch solution (more details can be found in Chapter 4). These results of error can guide future researchers to further improve the performance of the camera-based CPR evaluation techniques.

1.2.2.4 Emotion recognition in noisy environments

EMS providers sometimes operate in a very noisy and chaotic environment. For example, they might work on sidewalks where traffic noise is commonplace, or in buildings with significant ambient noise. Both human speech (other than the user) and environmental noise sources may pollute the audio signal observed by the cognitive assistant. The cognitive assistant must make inferences about the emotional state of the user in this noisy environment. Such a system can be used to provide feedback to the EMS provider in order to reduce stress. Current Speech Emotion Recognition (SER) systems fail to infer emotions with high accuracy under noise [28]. Noise-robust emotion recognition models must be developed for the cognitive assistant to handle noisy environments. We introduced three techniques to mitigate the adverse effects of noise on vocal emotion recognition systems. First, we showed that by combining both magnitude and phase features (in the form of magnitude and Modified Group Delay (MGD) spectrograms) as the input to the model improves the noise robustness. Training the model with input voice samples mixed with Gaussian noise also improved the noise robustness of our model. The third technique is to use an attention mechanism in our model. We saw that the attention mechanism can help our model to give more attention to the clean part of the input and pay less attention to noise-corrupted sections which resulted in a model with higher noise robustness. Apart from the EMS domain, these general techniques may be helpful in areas such as telehealth, helping caregivers of dementia patients, helping PTSD patients, and assessing the emotional state of callers to an emergency call center.

1.2.2.5 Activity Recognition with Limited Data

It is important to detect various steps in EMS protocols such as cardiac arrest. A cognitive assistant can potentially use the detected step/activity to provide feedback to the (trainee) EMS responder on missing steps or possible next steps, or to help automatically generate reports on which EMS protocols were performed. There is a lack of research trying to detect various steps due to the lack of data available. We introduce a novel few-shot learning technique to build models that can be used to detect activities with a limited amount of data. We showed that including a center loss while training our model helped to generate embeddings that have a better separation between different classes. The model adaptation technique that we introduced can further improve the performance of the model on a few shot classification tasks. The efficacy of these techniques was demonstrated on several public activity recognition datasets. These general techniques can be used in the future to detect various EMS-related activities and their steps. Note that the usage of these techniques is not limited to the EMS domain and they can be used in other domains such as rehabilitation monitoring, elderly assistant, fitness tracking, and gesture control.

1.3 Thesis Statement

By gathering an extensive dataset on specific EMS-related protocols and enhancing the current technology in areas such as depth perception, learning from minimal examples, and noise-resistant emotion recognition, we can overcome several key barriers that hinder the development of an intelligent cognitive assistant aimed at addressing the challenges faced by EMS providers.

1.4 Contributions

This dissertation makes the following key contributions to the fundamental algorithm field of research in emergency response:

- We create the first multi-model dataset that can be used to evaluate CPR performance. 24 participants were asked to perform CPR at various compression depths and compression rates using a manikin equipped with a sensor to collect ground truth on CPR compression depth and rate. Data was recorded with a chest-worn camera (also equipped with an IMU), two external stationary cameras, a smartwatch, and a depth sensor placed inside the manikin. The collected dataset can be found here[1]. We collected a dataset of hand images from 17 participants, using two cameras and their ground truth depth maps. This dataset was used to train and evaluate depth models that can estimate hand depth using defocus blur. The hand dataset can be downloaded here[2].

- We demonstrate that a smartwatch or an external camera can be used to measure CPR compression depth and rate accurately. Our smartwatch-based model achieved an error of 5.4 compressions per minute when measuring CPR compression rate and an error of 6.6 mm when measuring CPR compression depth. The external camera-based method displayed an error of 16 compression per minute when measuring the CPR compression rate and 25.6 mm when measuring the CPR compression depth. Although the depth estimation errors are significantly higher than the smartwatch-based model, we observed that the performance is much better under lower CPR rates and for certain participants (e.g. errors of less than 5 compressions/min and 11 mm for certain cases). We identified several sources of error and plan to improve our methods aiming to lower the errors.

- We showed that defocus blur can be used as a highly accurate tool for measuring depth using a single camera. Our methods achieved lower depth estimation errors than the state-of-the-art methods. The error reduction is around 3cm under the DDFF12 dataset, 7cm under the NYU depth v2 dataset and around 5cm for the synthetic dataset we created. This method was used to measure CPR compression depth with an external stationary camera. The method is sensitive to the characteristics of the camera used. A novel calibration method was developed in order to use this depth estimation method across many different cameras. Our research on camera-independent depth estimation with defocus blur was published in IEEE/CVF WACV [32].

[1]`https://drive.google.com/drive/folders/16f-1YaFIpey53A1BGeSe2LREzIYF-8AH?usp=sharing`
[2]`https://drive.google.com/drive/folders/1dFCp7cc1Ai7xwaDgr0BBVkcWkaPP2YC9?usp=sharing`

- We develop a novel Few Shot Learning (FSL) algorithm that uses the IMU of the smartwatch for activity recognition. Our model could recognize 7 activities on the PAMAP2 dataset with around 60% accuracy under a 1-shot setting while achieving 78% on a 5-shot setting. This is a general result that can be applied to steps in protocols used by first responders. The findings were published in an IEEE PerCom workshop [33].
- We develop several methodologies to build a noise-robust emotion recognition model for the estimation of the emotional state of the EMS providers in a noisy environment. Our methods showed an average accuracy of 76% on the Berlin Database of Emotional Speech. This is a significant improvement over the previous state-of-the-art model which had an accuracy of 56%. These contributions were published in the Journal Smart Health volume 19 [34].

1.5 Overview of this Dissertation

The next sections of this dissertation will begin with a discussion of the background and related work. The next chapter will present the theory on using defocus blur for depth estimation followed by the details on how to eliminate camera dependency of this method. Next, the performance of the model will be presented. The next chapter will provide details on the collection of the CPR dataset followed by a discussion on how smartwatches and cameras were used to estimate CPR compression depth and rate. The few shot-learning techniques that can be used to recognize activities with a limited amount of data will be presented next followed by evaluation results. The next chapter presents details on improving the noise robustness of the vocal emotion recognition models. The dissertation will conclude with several final remarks.

CHAPTER 2

BACKGROUND AND RELATED WORK

This chapter presents the most relevant work in the main topics discussed in this dissertation. These topics are monocular depth estimation, EMS protocol quality assessment, few shot learning, and noise robust emotion recognition.

2.1 Monocular depth estimation with defocus blur

Computer vision based depth estimation has many applications such as augmented and virtual reality (AR and VR) [35], autonomous robotics [36], background subtraction, and changing the focus of an image after it was taken [37][38][39]. Techniques such as structure from motion, structure from shading, shape from structured light, shape from defocus blur, depth from focus, multi-view stereo and Time-of-Flight (ToF) sensors can be used to estimate the depth of a scene [29, 30]. Active methods such as structured light and ToF sensors need specialized hardware and are power hungry. Stereo techniques measure depth by relying on multiple cameras to take several pictures of the scene. Techniques such as structure-from-motion and depth-from-focus [40] require several images of a static scene to estimate it's structure. Also, the assumption about a static scene does not hold when the scene is changing over time. Furthermore, structure from motion can only recover the depth of a scene up to a scale and cannot measure the absolute depth.

Single image defocus blur based depth estimation is a fairly under-explored topic in the literature [37] which utilizes the phenomena that certain objects in a photo appear more blurred than the others depending on the distance to those objects from the camera. Therefore, measuring the amount of defocus blur at a point of an image can provide a way to recover the depth to the respective point in the real 3D world. As we will show in Section 3, this method is effective for close range depth measurements (typically under 2 to 3 meters). This makes defocus blur-based depth estimation techniques ideal for measuring depth under many situations including in microscopic scenes [41, 42] and measuring depth to hands and nearby objects for a wearable camera. Certain animals in the natural world have the capability to estimate the depth of objects using the defocus blur. Nagata et al. have observed that certain parts of the principal eyes of the jumping spider is capable of focusing the light in different ways and this creates a blurred and a non-blurred images of the scene. Their brain can compare the blurred and the well-focused image in order to estimate the depth of the environment in front of them [43].

2.1.1 Depth from RGB images

Estimating depth maps from images can use various characteristics of images such as semantics, stereo matching, blur or differences in blur over a stack of images. [44–46]. Although stereo matching based and blur based depth measurements are seen as completely separate methods, Schechner and Kiryati [47] showed that both of them can be understood under the same mathematical formulation. A depth map can be estimated for the given image based on the domain knowledge on the structure of the objects in the image embedded in the estimation model [48–52]. Methods such as ZoeDepth [51] and VPD [52] have pushed the state-of-the art to be very accurate in measuring depth. However, a problem with these methods is that the estimated depth is only an approximation based on the structure of the objects. This makes these models sensitive to domain changes [29]. Also, techniques that can recover 3D structure from RGB images such as structure from motion can only estimate relative depths in a given scene [53].

2.1.2 Depth from defocus blur

The amount of defocus blur can be used to measure the depth of a scene from images. Since these methods rely more on blur which is a local feature of the image to estimate the depth, they are more robust to domain changes [29]. Shape/depth from focus methods aim to measure depth to a scene given a stack of images of different focus levels. A measure of the sharpness of each pixel of the images over the stack is calculated. The depth of a point is taken as the focus distance with the sharpest pixel. Various methods such as the Laplacian or sum-modified-Laplacian, gray-level variance and gradient magnitude squared were traditionally used to measure the sharpness [54, 55]. Modern methods utilize deep learning to automatically learn the sharpness measure from focal stacks [37, 40, 56]. But deep learning based techniques require a large amount of data to train [38].

Depth from focus methods that use a focal stack of the same scene has several drawbacks. First they assume the scene is static during the time needed to acquire several images with different focus (focal stack). Second, an accurate registration of the images in the focal stack is needed due to focal breathing (slight change of the filed-of-view of the camera due to changes of focal distance) or small movement of the camera and/or the scene [57]. Therefore, more investigation on depth estimation with a single image is necessary. Depth from defocus/blur relies on measuring the exact blur on a single image to estimate the depth and cannot use the relative variation of sharpness/blurriness of a focal stack. Due to this, depth from blur can be used to estimate the depth from a single blurred image [29, 38, 58–60]. Certain works are also concerned about removing the blur at the same time as estimating depth [38, 58, 61].

Estimating depth from the amount of the blur of a single image is ill-posed. This is due to having two possible depth values for a given blur [29]. Researchers have taken two different paths to solve this.

One solution is hardware based. One example for this is changing the shape of the aperture (coded aperture) of the camera to a shape that can help avoid the ambiguity. Ikoma et al. [62] used deep learning to learn the optimal shape for an aperture and came up with a prototype camera to measure depth from blur. Another example is to use a lightfield camera which takes many pictures with closely spaced micro lenses placed inside the camera [63]. The second approach is to use the domain knowledge (e.g. the shape and sizes of objects in the scene) of the scene to remove the ambiguity. Our research falls into this category. Gur and Wolf created a model which can generate the depth map of a scene given the blurred image and the All-in-Focus (AiF) image [29]. They also make certain assumptions about the shape of the blur circle. Usage of both AiF image and blurred images in making prediction makes this model less useful in certain situations because both of these images are not usually available from regular cameras.

Many methods in the literature first estimate the blur of a given blurred image and secondly estimate the depth from the blur. Physical consistency between the estimated blur and depth has been used as a form of domain knowledge by Zhang et al. [60]. Lu et al. create two separate models to estimate the blur and the amount of focus (sharpness) of a given blurred image. They claim that this method provides better estimates of depth due to the capability of estimating both blur and the sharpness of an image [64]. But since sharpness is just the inverse of blur, a question remains that by estimating blur aren't we also estimating the (inverse of) sharpness.

Ban et al. [41] extend depth from blur to microscopic images. Certain works focus just on blur estimation from a blurred image. Tai and Brown use hand crafted features of an image to estimate the blur map [65] while Zhuo and Sim assume that the edges in the images are step edges [66]. Cun et al. estimated the blur of a given image to separate the blurred and focused areas from a blurred image [67].

While all of the above methods assume that the blur is a single parameter (e.g. Gaussian or disk shape) Liu et al. expand our understanding by introducing a two parameter model. This model is also helpful in removing errors due to pattern edges [57].

2.1.3 Camera dependency of depth from blur

Certain characteristics of blur depend on the camera that is being used to acquire the images. The blurred image of a point has the same shape (but scaled) as the lens aperture. For example, if the aperture is circular, the blur of a point is also circular theoretically. But in practice this is a Gaussian due to diffraction [59]. In this research we assume all the apertures are circular in shape.

The size of the blur of a point depends on many other parameters of the camera. The f-number, focal length, pixel size of the image sensor (if the camera is digital), camera output scale and focal distance all affect the size of the blur [29] as shown in Chapter 3. Depth from defocus blur techniques estimate the blur of a given image as an intermediate step when estimating the depth. This makes these models sensitive to the variations due to camera parameters. We show evidence supporting this in our evaluation section.

But no papers in the literature address this problem. Gur and Wolf [29] use camera parameters in their model to recreate a blurred image. It was not used directly to predict depth and they do not test their model under different cameras. Although Maximov et al. [38] evaluate their model on several simulated datasets generated with different camera parameters they do not explicitly address or propose a solution to this problem.

2.2 Few Shot Learning

Few shot learning (FSL) is a technique that can be used to teach models new tasks with very few examples. This is different from traditional supervised learning where the modes can learn knowledge from large datasets. This is important for applications where collecting a large amount of data is prohibitive. Human activity recognition (HAR) using wearable devices has many applications including in fields such as healthcare, security, entertainment and human-computer interaction [68][69]. Deep learning based methods are being applied successfully to solve this problem. However, these models need a large amount of data to achieve a high level of performance.

Data for common activities such as walking and running are easy to obtain from public datasets, while data for activities outside of those areas are difficult to obtain. For example, publicly available data sets might not contain data for horseback riding or gymnastics because they are less common than walking. Also, there may be certain personalized activities that are unique to a certain individual. For example, the way certain individuals cook may be different from others. Therefore data for these activities are usually unavailable during the training time. Unlike in the domain of computer vision, HAR lacks data due to the difficulties in annotating data [70]. Consequently, HAR models which can detect activities with very few training samples are needed.

The field of HAR using wearable devices is very active due to the advancement, smaller size and lower cost of micro electronics and computer systems with high computational power. Their smaller size, privacy-preserving nature and being always closer to the user makes wearable devices more suitable to HAR than other types of sensors such as cameras [68] [71].

The performance of deep learning has surpassed the capabilities of other types of machine learning models in the field of HAR [72] [73]. Convolutional Neural Networks (CNNs) are proved to be more robust to the changes in underlying data distributions compared to Recurrent Neural Networks [74]. Fully Convolutional Neural Networks (FCNs) have been shown to demand less computational and memory cost while performing better than other techniques when significant class imbalance is present. Also, FCNs can accommodate variable input lengths which makes it more suitable for processing various activities with different duration. FCNs have been used successfully for HAR with IMU data with above advantages evident [75][76].

Class imbalance is also a challenge in HAR [77]. This stems from the fact that there is more data for common activities such as walking and running, but a limited amount of data for uncommon activities such as grabbing a box [72]. The lack of data for certain activities can hinder the performance on these classification tasks while creating over fitting and reducing the robustness of the classifier. Also the models need to learn from a large amount of data to model personal variations accurately [78].

The significant time and labor costs related to collecting data only exacerbates the problem [79]. In some cases data from certain classes may be unavailable during the development phase. In these cases, the knowledge from the classes with many data samples can be utilized to learn general knowledge that can be used to classify rarely seen classes [79].

Few-shot learning (FSL) is a machine learning technique that specializes in learning from a few examples. It's aim is to learn the ability to make inferences on new classes not seen during training [80] [81]. FSL has shown its use in several domains such as drug discovery, character generation, robotics, image classification, gesture recognition and neural architecture search [81]. One popular example of FSL is prototypical networks [80] where they learn a non-linear mapping from input space to an embedding space. Source class prototypes are calculated as the mean values of embeddings from each class. Learning is performed by minimizing the softmax over distance from training samples to their embedding class prototypes. Under the FSL setting, prototypes for target classes are calculated similar to prototype calculations during training. A test sample from the target domain is classified to the closest target prototype.

In essence, prototypical networks aim to cluster the embeddings of the same classes together while making the distance between clusters of different classes further from each other. Similar to prototypical networks [80], meta-learning [82] learns an embedding function. These embeddings are then classified with a SVM. Meta-learning optimizes the embedding function using the loss obtained from the SVM classifier [82].

Most of the FSL literature is only concerned about classifying target classes but for a practical application, source class classification may also be needed. Generalized FSL addresses this problem by devising methods to classify both source and target samples [83] [84][85].

The research done in FSL in the HAR domain is limited. In a FSL based HAR paper [79] a long short-term memory (LSTM) model is used to extract features and classify activities. It was trained with data from source domain and the network parameters obtained were transferred to the model that classified samples in the target domain. For each sample in the target domain parameters were only transferred from similar classes to avoid negative transfer.

2.2.1 Terminology

Base dataset: In many FSL settings, a model is first trained on a relatively larger dataset. During this learning phase, the model may learn knowledge that can be generalized to the new FSL domain. The base dataset may be either labelled (supervised setting) or unlabelled (unsupervised setting).

Support set: Once the model learns a general knowledge from the base dataset, it is usually given a very few examples from the new domain. The new domain consists of a new task to be solved. In the classification problem, the new task can be an unseen set of classes. The set of such examples are called the support set.

Query set: Once the model learns from the support set examples, it is given a set of samples called query set that the model must used to make inferences. For the classification problem, the model must predict the class labels for these unseen samples. The FSL performance of the model is evaluated on the prediction accuracy on query samples.

Number of shots and ways: The number of classes in the query set is called the number of ways (N). The number of samples in each class in the query set (K) is called the number of shots. The FSL problem can be called a N-way-K-shot task.

Next, several research areas that are related to Few Shot learning are mentioned.

2.2.2 Meta learning

Meta Learning or "learning to learn" technique tries to find a set of model parameters that can be quickly adapted to a new task by training over a range of different tasks.

2.2.2.1 Metric-Based Methods

Metric-based methods learn a embedding space so that samples of the same place will assume locations that are closer together in the embedding space while samples of different classes will occupy embeddings that are further away from each other. In Model-Agnostic Meta-Learning (MAML), first the model is trained on a range of tasks. During this "task-specific adaptation" phase, the model is adapted to each task with one or several gradient decent updates. Next, during the "meta-update" phase, the model parameters of the original model is updated so that the performance on each individual tasks are improved. The weights of the model are updated so that the change that the model parameters must undergo during the task-specific adaptation is minimized. This requires obtaining the second-order gradients of the model parameters [86]. Similar to MAML, the L2L-by-GDGD technique by Andrychowicz et al. learns how to update the optimization parameters of a model given a small number of samples [87]. Reptile is a simplification to MAML where rather than taking the second-order gradients of the parameters, gradients are updated at each step of task-specific adaptation. This makes the Reptile algorithm simpler and less computationally intensive [88]. Meta-SGD is an extension to MAML where not only the initial parameters, but also the learning rates and update directions are learned for each parameter [89]. LEO (Latent Embedding Optimization) learns an encoder-decoder pair to better optimize the model weights. The encoder creates a latent embedding for the model weights. The decoder can reproduce the model weights. LEO learns to optimize the model weights in the latent space through learning a range of tasks. Regularization methods are employed to restrict the latent space of the weight embeddings into some distribution. The ability to optimize model weights in a lower dimentional space can solve overfitting and improve efficiency [90]. Prototypical networks learns a metric space where the embeddings of the samples belonging to the same class are clustered close to each other while embeddings of samples that do not belong to the same class are pushed further apart. For a new task, class prototypes are defined as the centroids of the embeddings of the samples in each new class (taken from the support set). An unseen sample instance (from the query set) is classified to one of the classes in the support set based on the class label of the prototype that is the closest to embedding of the unseen sample. The training of the prototypical networks is conducted using episode based training where each episode simulates a FSL task. Here a number of tasks from a distribution of tasks are selected as the base task which a non-overlapping set of other classes were selected as the support and query sets [91]. Vinyals et al. introduced matching networks where an attention mechanism is used to focus only on the relevant examples of the support set when making a prediction on the query samples. This method utilized full context embeddings because the inference process takes into account the whole support set when making a prediction [92]. A Siamese network consists of two identical branches for predicting image embeddings. This model is trained with pairs of samples. When the sample pair belongs to the same class the embeddings are trained so that they are located in close proximity to each other. On the other hand when the sample pair belongs to different classes, the embeddings must be further apart from each other. Contrastive loss is used to train Siamese networks to achieve the desired operation [93]. Triplet networks contain three identical branches for prediction. The input to the Triplet network consists of three inputs; the anchor sample, positive sample and a negative sample. The positive sample belongs to the same class as the anchor sample. The negative sample belongs to a different class from the anchor sample. Triplet network can in theory learn richer representations because the network learns knowledge on both positive and negative sample at the same time. Both Siamese and Triplet networks can learn rich representation and operate with lower computational demand. But the positive and negative sample selection process highly affects the performance of the networks. Therefore, sophisticated sample selection methods may be needed for good performance on FSL problems [94]. Relation networks contains two modules made up with neural networks; embedding network and relation network. Embedding network, similar to prototypical and matching network creates embeddings for given input samples. The relation network can output a measure of similarity between two given embeddings. A sample from query set will be classified to the class of the embedding which is the most similar to the query embedding. The similarity scores provided by the Relation network is utilized to make the comparisons between the query sample and the samples in the support set [95]. While the feature extractor of Relation networks are independent of the support set, TADAM proposes a feature extractor that depends on the support set data and a learnable scaling factor to scale the similarity measure [96]. Garcia et al. [97] used graph neural networks to solve the FSL task. Data samples are represented by nodes and the edges represented by learned similarity. Though message passing techniques, the network predicts the labels on query nodes.

2.2.2.2 Model-Based Methods

Model-based meta learning methods uses specialized architectures or memory structures to facilitate FSL. RL^2 improves reinforcement learning algorithms to learn faster with a smaller amount of data, i.e for FSL. A slow RL algorithm learns to solve different tasks while updating the weights of a recurrent neural network (RNN). This RNN encodes the parameters required to generate a fast RL algorithm. A fast RL algorithm to solve a given new task can be generated by the learned RNN [98]. SNAIL introduces a novel architecture with temporal convolution networks and attention mechanism to solve the FSL problem [99]. Meta networks utilizes an external memory and a meta-learner to quickly learn new tasks with a small amount of data. The base learner learns to solve various tasks similar to many other meta-learning methods. When samples for a new task is presented, the encoded version of these samples are stored in the external memory. A specialized section of the meta network is used to query the external memory given a query sample and generate a set of new weights for the base learner thereby making the base learner specialized to solve the new task [100].

2.2.2.3 Bayesian meta-learning

These methods model various parameters as random variables and obtain their distributions. Hierarchical Variational Autoencoders (HVAE) can be used to learn the statistics of a dataset with a hierarchy of latent variables. This hierarchical nature lets the HVAE to adapt fast to a new data distribution with only a few samples. Given the data samples from the support set of a new distribution, the statistics for each class can be used. These statistics can be used to classify a new query sample to one of the classes [101]. Bayesian MAML extends MAML to consider the Bayesian principles to improve model robustness, uncertainty estimation and FSL performance. Standard MAML finds a set of initial parameters for the base model that can be quickly adapted to a new task, Bayesian MAML finds distributions for these parameters. This improves the robustness of the model and also provides a method to estimate the uncertanity of the prediction by predicting with parameters sampled from the distributions [102][103]. Deep kernels combine kernel based methods with deep learning. A deep neural network extract features from the inputs. Then a radial basis function or some other kernel function is used to calculate the pair-wise similarity between input pairs. The features and the calculated pairwise similarities are used to make predictions by using a Gaussian process framework which also outputs prediction uncertainties. For each new task, parameters of both the deep neural network and kernel function are adjusted in meta-learning fashion [104].

Some recent works have shown that simple whole-classification training, where the model is trained on all the available tasks rather that training on just a subset of tasks as done in meta-leaning can produce comparable or even better performance than meta-learning. Chen et al. [105] showed that first performing whole-classification training and then doing meta-learning can produce better results that earlier FSL methods.

2.2.3 Transfer Learning for Few Shot Learning

Transfer learning can be used to adapt a pretrained model to a new domain or a task. Researchers have demonstrated that features learned by deep neural networks are transferable among a wide variety of domains and tasks [106] [107][108]. It has shown that in certain cases transfer learning outperforms training from scratch [109]. Transfer learning has shown to outperform more complicated meta-learning based FSL techniques [110] [111] [112] [106] [113] specially when the domain shift between source and the target domains are large [114] [115].

Meta-transfer learning aim to combine the best characteristics of both meta-learning and transfer learning techniques [116]. Meta-transfer learning defines new parameters scale and shift for each existing weight and bias of a network that is pre-trained on a large dataset. These scale and shift are adjusted based on a given meta-task. The number of parameters adjusted (shift and scale) are much less than the original number of parameters of the network which reduces overfitting. Having a separate scale and a shift for each weight facilitates a very fine-grained learning for each meta-task. The model still keeps the original weights which prevents catastrophic forgetting of the original knowledge. During meta-learning episodes, harder tasks are given priority over easier tasks and fed to the model more often than the easier tasks. Hard task are defined as the tasks where the model performs worse. Meta-transfer learning achieved state-of-the-art results in miniImageNet and Fewshot-CIFAR100 datasets. Chowdhury et al. show that using an ensemble of CNN models pretrained on diverse and large datasets can outperform more complicated and well-established meta-learning algorithms. The ensemble of models are combined with a simple feed-forward network trained with L2 regularization [117]. They show that their method of using a collection of relatively smaller CNN models trained on relatively smaller but diverse datasets performs better than a single much larger CNN model trained on a much larger dataset as mentioned in the Big transfer paper [118]. Chowdhury et al. highlights the importance of the diversity of both the model architectures used and the datasets. Different models can produce complementary features for the same input data. This means at least some of the pre-trained models may produce features that are relevant to the FSL task at hand. The library of models must be pre-trained on a diverse set of data which can reduce biases and improve the generalization across domains and to facilitate robust feature extraction. Shen et al. pointed at some drawbacks of transferring all the knowledge of a pre-trained model to another domain. This is because some knowledge of the first domain may be relavent on the second domain, some knowledge may be detrimental to the task on the second domain. To solve this Shen et al. introduces the partial transfer techniques where certain layers of the pre-trained network are frozen while the other layers are fine-tuned on the FSL task. Their evolutionary search method can determine the optimum set of layers to freeze. This technique achieved state-of-the-art performance on several widely used FSL datasets [119]. A self-supervised transfer learning technique has been proposed to solve FSL by Medina et al. [114]. They use transformations with a random component to create augmentations of the input signal. The model is trained to create embeddings that are clustered together for these augmentations. Note that this process is completely self-supervised. Their technique which is named ProtoTransfer produced comparable results with supervised methods while out-performing other unsupervised methods. Dumoulin et al. created a unified framework that can be used to compare transfer learning and meta learning techniques by combining the Visual Task Adaptation Benchmark (ViTAB) and Meta-Dataset (MD). The resuls show that large-scale transfer methods (models which are trained on vast datasets) outperform competing approaches including meta learning [115]. Guo et al. created a new benchmark to evaluate FSL methods called ”Broader Study of Cross-Domain Few-Shot Learning (BSCD-FSL)” and found that earlier meta-learning methods outperformed the newer ones on the benchmark. Further, simple fine-tuning strategies out performed more complicated meta-learning techniques by a significant margin. Surprisingly, even randomly initialized models (rather than pre-training on a large dataset), when fine-tuned outperformed meta-learning techniques on some occasions [112]. STARTUP uses a pre-trianed model on a large dataset and adapts this to a smaller target domain that is significantly different from the source domain [120]. The authors assume no labelled data from the target domain that can be used for training. They create a teacher and student models where the teacher is pre-trained on the larger labelled source dataset. The teacher model creates soft-labels for the target domain. STARTUP does not assume these labels to be very accurate. The student model learn to predict the labels of the source domain and the match the prediction distribution of the teacher model under the target domain (by matching distributions through KL divergence loss). It also tries to minimize a self-supervised loss on the target dataset. This self supervise loss is obtained with the SimCLR framework which tries to bring embeddings of the augmentations of the same input closer together while pushing the augmentations of different inputs further apart [121]. GFL model utilizes Graph Neural Networks (GNN) to solve the FSL problem. Given a new task, the nodes of the GNN represents the data samples. Some nodes may have labels. The GNN can infer the labels of the rest of the nodes based on the structure of the graph. GFL utilizes a target graph and an auxiliary graph. The auxiliary graph has learned knowledge from a a larger dataset which is potentially significantly different from the target dataset. The target graph models the target dataset. The auxiliary graph can transfer important knowledge to the target graph that can improve the performance of on the FSL task [122]. Chen et al. found that including a self-supervised task between pretraining and the FSL task can improve the FSL performance under Natural language processing tasks. The self-supervised tasks include masked word prediction tasks related to next sentence prediction [123]. Most of the FSL techniques are called inductive FSL where the model is first trained on a labeled base dataset and then adapted to the target dataset with a very small number of labeled data. In contrast, transductive FSL techniques can utilize the unlabelled target data along with labelled base data when training the base model. The Shot in the dark paper by Chen et al. shows that self-supervised training on unlabelled data alone from either or both the base and the target dataset can outperform the best trasnductive FSL methods on datasets like mini ImageNet and tiered Image Net. These resuls builds a compelling case for self-supervision for FSL [124]. The category traversal module from the paper by Li et al. can consider the whole support set when predicting the label for a given sample for the FSL task. This differs from the traditional FSL solutions which only consider a single sample from the support set to be classified. But considering the whole support set, this solution can generate a set of features that are more suitable for the given task [125].

2.2.3.1 Test-Time Adaptation

Test-Time Adaptation (TTA) techniques introduces methods to adapt a pre-trained network during the inference time into a new domain which typically only contain a scarce amount of data. This is different from the traditional setting where the network is once trained on a base dataset and fixed thereafter. Many recent TTA techniques assume that they will not have access to the training data or labels for the target domain during TTA. TTA adaptation differs from similar techniques such as domain generalization which needs specialized training techniques and domain adaptation which demands access to both source and target domain data at the same time [126]. TTA can be divided into several main areas depending on the main problem it is trying to address. Test-time domain adaptation utilizes all the test data before generating the final prediction. Test-time batch adaptation adapts the original model to a single target batch at a time. Online test-time adaptation adapts the original model in an online manner. Here target batches are used by the model in an online manner. The model may see a target batch only once. But it can utilize the knowledge gained through a target batch for the batches encountered in the future batches [126].

Hu et al. uses a backbone network that was trained on a large dataset as the feature extractor. They note that when this model is used on a new task, the feature distribution deviates from being a Gaussian-like distributions to a more skewed distribution. They introduce a power transform that can convert the feature distribution back into a Gaussian distribution thereby improving the performance of the model on the new domain [127]. TENT [128] introduces a novel technique for domain adaptation. A pre-trained model is adapted to a new domain or a task by optimizing the scaling and shift parameters of the batch normalization layers of the model. They assume no labels for the new domain. An entropy minimization loss is used to minimize the entropy of the model on the test dataset. This is based on the assumption that low entropy of the predictions of a network is associated with higher accuracy. TENT showed a significant improve in performance in areas of image classification (including corrupted images and domain shifts) and semantic segmentation. Conventional batch normalization (CBN) uses running statistics (mean and standard deviation) from the source dataset to normalize batched data at batch normalization layers.

Transductive Batch Normalization (TBN) methods such as used in TENET uses statistics from the test set in the BN layers. TTN by Lim et al. improves the TTM by combining TBN and CBN by calculating and adaptive version of statistics by interpolating between the statistics of the source and the target domains [129]. Note that TTN does not seek to optimize the scale and shift parameters like TENT does. Although direct comparison between TENT and TTN are lacking, TTN might be more appropriate for cases where the labelled target data is noisy. But TTN method might add additional complexity due to the weighted interpolation between the source and target statistics. TENT on the other hand adds not additional complexity since it relies on regular back propagation based training on the test dataset. However TTN is shown to outperform other methods including TENT for smmaller batch sizes. RoTTA address the dynamic nature of test time adaptation. It assumes the test data shifts in a dynamic manner. For examples the images captured by an autonomous vehicle may gradually shift domains from a sunny day time to night time. In a method that has some similarities to TTN, RoTTA starts the BN layers of their model with statistics from the source domain and updates them gradually with the running statistics for the target data it encounters [130]. Segu et al. uses TTA for the domain of multiple object tracking. They tested this technique under various domain shifts including sim-to-real, outdoor-to-indoor and indoor-to-outdoor [131]. Niu et al. showed that minimizing entropy and adjusting batch normalization layers similar to TENT can produce unstable results under several realistic conditions. TENT assumes that the test data only contains a single domain. But in certain realistic situations, this might be not true. For example, an image recognition model might encounter both out-of-focus images and images captured with different lighting conditions compared to the source dataset. They also showed that TENT fails when TTA is performed with single samples instead of using a batch. The next failure case encountered by TENT is when the TTA is performed with imbalanced number of labels in the batch. Niu et al. posits that these failures may be due to the mean and standard deviation calculated by the batch normalization layers being biased under the test conditions. They show that using layer and group normalization which do not depend on the batch statistics rather than utilizing batch normalization can sometimes solve these failure cases. They also introduce a method to filter out noisy samples in the test data by avoiding the samples that generate gradients that are too high. They also improve the entropy loss from TENT to be a sharpness-aware entropy minimization method. Niu et al. expriment on the ImageNet-C dataset which contains 4 main categories of image corruptions (noise, blur, weather and digital) which signifies the domain shifts [132]. AdaContrast [133] first obtains a model pre-trained on the source dataset. Then it generates pseudo labels for the target dataset. These labels may be noisy due to the domain shift. To alleviate the problems with the noisy pseudo labels, AdaContrast utilizes a voting scheme while considering the nearest neighbours of embeddings features of a given target input. The paper evaluate AdaContrast on image recognition datasets (VisDA-C and DomainNet-126) and achieve state-of-the-art results beating other methods including TENT.

Most of the TTA methods described above adapt a model that is already trained on a source domain to a target domain when there are no target labels are available. But the task that the model must solve is the same. For example for the case of classification, the class labels do not change. But during FSL, the classes do change. Previous literature have not utilized TTA techniques to solve the FSL problem itself. Bennequine at.al [134] takes an important step towards this direction. But their usage of TTA is aimed to address the domain shift between the support and query sets. They utilize prototypical networks to address the FSL problem itself. Liu et al. introduces a graph-based solution where all the target samples are mapped to a graph. In this graph, support set samples are assigned with a label. The model predicts the labels for the query set after considering the whole target dataset. This is different from regular FLS setting where only a single batch is considered at a time. This can be considered as a TTA based FSL technique because the model structure being adapted to the whole target set [113].

2.3 EMS protocol quality assessment

2.3.1 EMS provider training and use of augmented reality solutions

Although regular training was provided to EMS personal, a significant number of them are often find themselves unprepared at an emergency situation. To overcome this, cognitive assistant, VR and AR solutions have been proposed [135] [136] [137]. Rahman et al. [20] [138] developed a cognitive assistant based solution to extract vital information such as about patient condition and medical history from the conversations among EMS providers and patients. Preum et al. proposed a method to automatically curate domain specific knowledge (e.g. EMS domain) using natural language processing that can be used to develop cognitive assistants [22]. Sharon and Daniel used a VR environment in early 2000s where the users could interact with virtual objects which prepared emergency responders for bio-terrorist attacks [135]. Wilkerson et al.. created a similar solution where the feedback they received suggested that VR can be successfully used to train first responders [137]. Koutitas et al. devised both AR and VR solutions to train first responders to operate in an ambulance bus environment [136] [139]. After training with the AR/VR experience created with the head mounted displays, the participants experienced better preparedness to tackle real life emergency situations. One of the limitations of VR based solutions is that the user does not get to touch and feel the real life objects. AR based solutions can fuse the benefits of VR and the realities of actual physical interaction with the real world.

2.3.2 Cardiac arrest and CPR protocols

The EMS protocol for cardiac arrest is complicated and can include interventions such as ECG, ventilation, cardiac monitor, chest decompression, CPR, defibrillation, and intubation. It is critical that the first responders perform the medical procedures with the correct techniques and protocols. Adherence to the correct techniques and protocols during medical procedures such as Cardio-pulmonary Resuscitation (CPR) can save lives [10]. Regulatory bodies have put forward protocols and guidelines on how to best perform Cardio-pulmonary Resuscitation (CPR). It is difficult to memorize the protocol and deliver CPR according to the accepted guidelines [12]. Hence, CPR delivery is often sub optimal and the compressions are too shallow. Proper CPR technique can significantly improve the survival of the patient[10]. Therefore, good feedback on the CPR procedure for the trainees is important. Providers monitoring a CPR procedure can provide feedback on the quality of the CPR based on visual observation. But this feedback can have significant biases. For example, the accuracy of such feedback depends on the viewing angle [140]. It has been observed that health care providers often have an erroneous estimate of the quality of their own CPR performance. Visual human real-time CPR feedback and recent CPR training (or Just-In-time Training; JIT) can improve the perception of CPR depth, but did not improve other CPR quality measures such as rate of Chest Compression Fraction (CCF). This demands more objective measures of CPR quality during training and actual resuscitation [141]. In a study conducted spanning nine pediatric institutions found that despite JIT CPR training and real-time visual feedback or none of these interventions, there is a significant variability in CPR compression rate and depth. This calls for high quality real-time CPR feedback [142]. In a study conducted with 244 lifeguards revealed that while most of them adhered to the CPR guidelines, there are issues with ventilation and compression depth. Also most of them had not received pediatric-specific CPR training and felt more confident performing CPR on adults compared to children. The study recommends use of electronic feedback manikins during training [143]. It has been shown that using real time feedback from a pediatric manikin can significantly improve the CPR compression quality in terms of depth and rate in a study involving 75 healthcare providers [144]. Another study showed that both JIT CPR training and real-time feedback from a device called CPRcard improved CPRO depth and rate performance of 108 three-person teams [145].

2.3.3 CPR quality metrics

Detection of chest compression is a preliminary requirement of assessment of CPR quality. Quantities such as frequency and number of chest compressions, chest compression fraction (CCF) and chest compression rate (CC-rate) can be derived from the chest compression detection. In addition, compression depth (CD) and exerted force can be used to evaluate CPR. Studies have shown that there is a correlation between these parameters and survival rate, spontaneous circulation and restoration of the patient [10]. Among various CPR quality measures CPR compression depth and rate have been used for many studies. A positive relationship between good CPR rate and compression depth with improved physiological response has been shown [146]. In this study we use CPR compression depth and rate as our CPR quality measures.

2.3.4 CPR quality measurement solutions

A human observer cannot measure certain quantities such as compression depth with high accuracy, but can only give a rough estimate of the adequacy of the compression depth under visual observation of a trainee performing CPR. To solve such problems, an automated system that can evaluate the quality of various medical procedures is needed. It has been shown that when given real time feedback on CPR quality, participants displayed a significantly better CPR quality in terms of CPR compression rate and depth [147].

CPRcard is a credit-card-sized device equipped with accelerometers that can be placed on a CPR manikin which CPR is administered. This device can measure the CPR rate and depth and provide feedback to the healthcare provider. The feedback is provided via two separate sets of LED that signals the user of adequate/inadequate compression depth and rate. This device can provide CPR compression rate measurements with an error rate of around ±5% compressions per minute and depth estimation error of $\pm5mm$ or 10%, whichever is greater [145] [148]. CPRcard has featured in several research studies [149] [150] and has shown to help improve the CPR quality of its users. TI signals recorded with defibrillation pads is a great signal to calculate the CPR compression rate and is shown to be in par with the the gold standard CPR compression rate which is CPR rate calculated using compression force or combined ECG and TI signals [151]. Ayala et al. used a modified Heartstart 4000 defibrillators to record ECG, Thoracic Impedance (TI) signals from the regular defibrillation pads while also recording chest force and acceleration signals through modifications. They calculated a chest compression depth using the TI, force and acceleration signals and showed that both TI and and a combination of chest force and acceleration can be used to accurately measure the CPR compression rate and Chest compression fraction [10]. Lu et al. presented a method to calculate the CPR compression depth based on a measure of the magnitude of acceleration obtained with a smartwatch and the ground truth rate of CPR compressions obtained with a CPR manikin [152].

2.3.5 Data scarcity for CPR

Although previous research considered automatically evaluating the quality of CPR, they have not considered visual modality or wearable devices [10][12]. Smart devices like smartphones, smartglasses and smartwatches have become cheap and ubiquitous. They can provide a cheap and accessible solution for providing feedback for trainee first responders on medical procedures such as CPR. But to train models on such modalities, a large amount of training data is required. To solve the lack of training data, A comprehensive dataset of participants performing CPR on a dummy was collected. Data from various modalities including vision, audio and IMU sensors were collected.

2.4 Noise Robust Emotion recognition

2.4.1 CNNs for SER

Deep learning has become the state-of-the-art of many audio classification tasks including SER [153]. Within the deep learning domain, CNNs have become popular for computer vision tasks [154]. But if any set of features can be represented as images or stack of images (multi dimensional arrays), CNNs can be used to classify them. Magnitude spectrograms obtained via Discreet Fourier Transformation (DFT) of audio signals is commonly used as the input to CNNs in audio classification solutions. Because spectrograms are 2D images, the audio classification problem may be treated similar to a regular image classification problem using a CNN [154] [155] [156] [157]. [157] showed that CNNs are better suited than Deep Neural Networks (DNN) and Long Short-Term Memory (LSTM) networks for the SER problem. Although it is not straightforward to compare state-of-the-art models in SER due to the variety of datasets and evaluation methods, CNNs seems to be performing better than other types of classifiers for SER.

2.4.2 Inputs for CNNs

In terms of CNNs various types of inputs qualify for the task of SER. [158] found evidence that frequency domain features are better suited than time domain features for acoustic event classification using deep learning. Various frequency domain inputs such as DFT components [158], magnitude spectrograms [159] [156] [160] [155] [161] [157] [162] [163] [153] [164] and wavelet features [165] have been used in the literature. Spectrograms can retain more information than most hand-crafted features and are low dimensional than raw audio [155] and are one of the most commonly used feature types for audio related tasks.

Spectrograms are obtained by performing Discrete Fourier Transformation (DFT) on the audio signal. DFT of a signal around the sample n can be represented by

$$X_n(\omega) = \sum_{m=-\infty}^{+\infty} x(n)w(n-m)e^{-j\omega m} \tag{2.1}$$

Here the frequency of the component $X_n(\omega)$ is ω and w is a windowing function. $X_n(\omega)$ is a complex number and can be represented in terms of magnitude and phase angle as.

$$X_n(\omega) = |X_n(\omega)|e^{jarg[X_n(\omega)]} \tag{2.2}$$

Here $arg[X_n(\omega)]$ is the angle between imaginary and real parts.

If DFT of a signal was taken with sliding windows a distribution of frequency characteristics along time axis can be obtained. This is called a spectrogram. If the magnitude part $|X_n(\omega)|$ from equation 2 is used, this is called the magnitude spectrogram. If the part $arg[X_n(\omega)$ is used, it is called a phase spectrogram. To recreate the original signal from the DFT components, both magnitude and phase parts of the signal should be used [166].

But traditionally only the magnitude spectrogram is used and the phase spectrogram is ignored [167]. This is mainly due to the fact that the phase is discontinuous due to mathematical problems occurred when calculating the inverse of trigonometric functions. A simple correction called phase unwrapping should be performed to mitigate this problem. Researchers are beginning to explore the effectiveness of the phase based features for various tasks such as speech and speaker recognition. They found that combining magnitude with phase spectrogram improves the performance [168] of SER systems. [169] deals with detecting gunshots from audio data. They calculate statistical features from magnitude and phase spectrogram of the audio signal and conclude that combining these features improves performance than just including features from the magnitude spectrogram. [170] lists different phase representations those are being used in literature. They are relative phase shift, time-frequency derivatives of phase, phase dispersion and phase distortion.

Modified Group Delay (MGD) is another quantity that has seen a recent rise in usage. It combines magnitude and phase components of the signal. There exists some research that investigate the effectiveness of MGD for speech and speaker recognition. But only a limited number of literature investigate this for emotion recognition. Research shows that MGD performs better than traditional features for whispered emotion recognition [171]. Modified Group delay can be calculated using the DFT. Let

$$y(n) = nx(n) \tag{2.3}$$

Here $x(n)$ is the n^{th} sample of the speech signal. If $X(\omega)$ and $Y(\omega)$ are the Fourier transformations of $x(n)$ and $y(n)$ at any given n
Define

$$\tau'(\omega) = \frac{X_R(\omega)Y_R(\omega) + X_I(\omega)Y_I(\omega)}{|S_c(\omega)|^{2\gamma}} \tag{2.4}$$

Here X_R and Y_R are real parts of X and Y. X_I and Y_I are imaginary parts of X and Y. To obtain $S_c(\omega)$, first squared magnitude $|X(\omega)|^2$ of the signal $x(n)$ is obtained and then it is cepstrally smoothed. Modified Group Delay (MGD) can be obtained as follows.

$$\tau(\omega) = \frac{\tau'(\omega)}{|\tau'(\omega)|}(|\tau'(\omega)|^{\alpha}) \tag{2.5}$$

In these equations α and γ are parameters where $(0 < \alpha \leq 1)$ and $(0 < \gamma \leq 1)$. These parameters should be tuned depending on the application. Obtaining MGD with a moving window yields a MGD spectrogram.

Although many frequency domain feature types are being used in literature, their comparative performance and the reason to choose a particular feature type is rarely studied [172]. But there is evidence that different features can yield different performance levels. Even slightly changing the representation of features can change the performance. For example, [164] use CNNs for snore sound classification. They input magnitude spectrograms obtained with 3 different color maps. Different color maps gave different performance.

Previous studies have considered combining different feature types as inputs to deep learning based audio classification models. [161] [168] and [165] found that combining various frequency domain features improved the performance of their audio classification models. There are different methods to combine feature types as inputs to CNN models. [161], [173] and [174] combines each different spectrogram as a different channel when they are input to the CNN. This is similar to the way R,G and B channels are treated in RGB images. Different from this method, [168] and [174] concatenates two different spectrogram representations side-by-side to form a single image. Although it is intuitive to assume that different methods of combining the features may yield different results very limited research is done in this regard.

2.4.3 CNNs accepting variable input sizes

The height of a spectrogram depends on the range of frequency considered and the length depends on the time interval considered. Since people may speak utterances of variable length, an emotion classification models should have the capability to handle inputs with variable lengths. But traditional CNNs only accept a fixed size of input.

One method to overcome this is to combine a CNN with a Recurrent Neural Network (RNN) model [153] [154] [160]. In these studies the CNN acts as a feature extractor and the RNN (in this case it is a LSTM) layer learns the time sequence relationships in these features. [153] showed that combining CNN with LSTM better performance can be obtained than just using a CNN. This may be due to the fact that LSTM can model the sequential characteristics of the input.

Another method to handle variable input lengths is to use a Fully Convolutional Neural Network (FCNN). A FCNN does not explicitly model the sequential nature of the input. Instead it considers the whole input (of variable length) and make inferences. A FCNN can be generated starting from a traditional CNN by replacing all fully connected layers with convolution layers. This can also be beneficial because it reduces the number of trainable parameters in the CNN. [175] uses FCNN for spoken emotion recognition task and saw that it outperformed a model which combined CNN and LSTM.

2.4.4 Performance under noise

SER systems can be adversely affected by environmental distortions. Noise and reverberation are two main sources of distortions that could degrade the quality of the audio signal. Reverberation depends on the characteristics of the environment that the sound is produced. Noise is generated due to sound sources other than the one we are interested in. When the distance form the speaker to microphone increases, increased noise relative to signal can be observed [28]. Noise and reverberation can affect the original clean speech signal as shown in equation 6.

$$y(t) = h(t) * s(t) + n(t) \tag{2.6}$$

Here h(t) is the room impulse response between the microphone and the speaker. s(t) is the clean speech and n(t) is the background noise. * is convolution operation. Note that reverberation is represented as convolution with the original signal. Noise is represented as addition [176].

Studying both reverberation and noise for SER is important because both of these distortions can affect SER systems in an adverse manner. Previous studies have attempted to study/reduce these effects from both noise and reverberation. For example [177] studied the effects of reverberation and [28] studies the effects of both noise and reverberation for SER. [163] use CNN with amplitude spectrograms as input and observed performance degradation when noise is added to the inputs. [159] use magnitude spectrogram and provides reasoning for spectrograms may be better for handling noisy data. [178] studies the effects of the distance to the microphone from the speaker on a SER system because increasing the distance also increases the amount of noise with respect to that of the signal.

In this study we aim to find solutions for the adverse effects of noise on SER systems. Although the problem of reverberation is also very important, certain solutions for that problem can be obtained easier than the problem of noise. For instance, if we have prior knowledge that the SER system will operate in a certain environment (e.g. living room of a house) it can be assumed that majority of the living rooms have similar reverberation characteristics. Therefore if the SER system was trained with data recorded in living rooms, this SER may be able to solve the problem of reverberation for operation inside similar living rooms. The problem of background noise on the other hand is harder to solve. Although there are certain types of noises such as air conditioner which is present in most of the house holds, there are many other unpredictable noises such as a noise of a vehicle outside the house or a ring of an alarm. Therefore it is not realistic to assume that all the possible noise conditions can be known beforehand. Therefore we focus on solving the problem of background noise in this study.

2.4.5 Methods to mitigate effects of noise

Several methods can be used to reduce the adverse effects of noise on SER systems. Some of these methods are described below.

One method is to remove the noise from raw speech signal or calculated features. This is generally called speech enhancement. Several main categories of speech enhancement can be identified from literature. They are spectral subtraction based [160], subspace based, statistical-model-based, Wiener-filter based algorithms [179] and de-noising autoencoders [180] [181]. The aim of this research is to find noise robust features, CNN architectures and methods of training them. Once these are found, speech enhancement can also be applied on top of our solution to further improve the performance. Future studies may be necessary to quantify these further improvements.

The second method is using features which are robust to noise. Some features are more robust to reverberate and noisy conditions than others [28]. According to [182] different frequency domain features may have varying robustness for noisy data for speech recognition task. The noisy environments they consider are background music, white noise and reverberate environments. [182] use 1st and 2nd derivative of Mel-Frequency Cepstral Coefficients (MFCC) features for speech recognition. They found that including the derivatives of MFCC features improves performance in both noisy and quiet environments. [166] described MGD which is calculated using both magnitude and phase of the Fourier transformation of the signal. MGD is analytically shown to be robust under additive noise [183] [166] [184]. Practically, features calculated with MGD is shown to be robust under noise for speech recognition [184] [166] and for voice activity detection [183]. We explore this dimension in this study. Only a very limited number of research is done regarding noise robust features in the context of SER. We explore the noise robustness of several DFT based features and their combinations.

The lack of performance of models under noisy conditions can be attributed to the difference between training and testing data distributions. Researchers have tried to reduce this problem by trying to bring the training data distribution closer to that of the testing data by artificially injecting noise or reverberations to the training dataset. [28] and [177] utilize this method to improve model performance under reverberate conditions. The same methods can be applied to noisy conditions. Limited number of research is done exploring this dimension with regard to SER systems. In this study we explore the effect of adding synthetic noise to training data on the noise robustness of these models.

There are evidence that certain NN architectures are more robust to noise than others. For example [185] shows that CNN with attention mechanisms could be more robust to noise compared to other types of models (e.g. LSTM). Attention mechanism used with CNNs can be used to improve the model performance by teaching a model to explicitly pay attention to important parts of the input and to ignore unimportant parts. [186] used attention mechanism with a CNN for image captioning tasks and observed an improvement in performance. Similar models can be used for SER [175]. [175] uses FCNN with attention mechanism and saw an improvement of performance compared to other models. But a detailed analysis on the noise robustness of CNNs with attention mechanism was not performed in the literature. In this research we implement an attention mechanism for a Fully Convolutional Neural Network (FCNN) and show that adding the attention mechanism improves the noise robustness properties of the model.

CHAPTER 3

MULTIMODAL DATASETS THAT ARE USEFUL TO CREATE A COGNITIVE ASSISTANT FOR EMS PROVIDERS

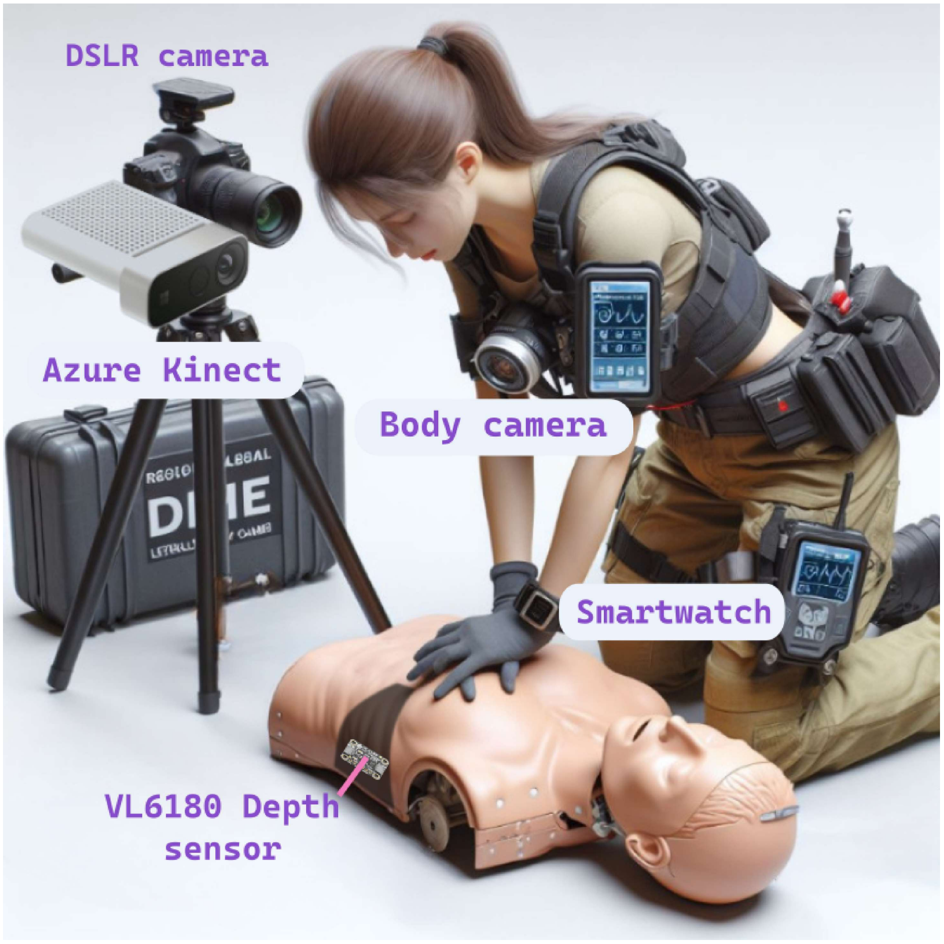


Figure 3.1: Data collection setup

3.1 Introduction

As mentioned in the previous sections, we aim to develop methods that can be used by augmented reality systems to evaluate CPR performance. Although previous research considered automatically evaluating the quality of CPR, there are very limited considerations given for wearable devices and they have not considered visual modality [10] [12]. Smart devices like smartphones, smartglasses, and smartwatches have become cheap and ubiquitous. They can provide a cheap and accessible solution for providing feedback for training first responders on medical procedures such as CPR. But, to train models on such modalities, a large amount of training data is required. To solve the lack of training data, we collected a comprehensive dataset of 24 participants performing CPR on a dummy. We collected data from various modalities including vision, audio, and IMU sensors. There are no previous research done which uses vision as a modality to measure CPR quality. Measuring CPR compression depth is a critical component of measuring CPR quality. Therefore, depth estimation to hands is a critical component of our solution. There are some datasets containing hand images and their depths. But, the range of the depth in these datasets is limited. Furthermore, there are no existing datasets that are suitable for depth-from-defocus-blur methods. Therefore, we collected a dataset which contains hand images from two cameras and their corresponding depth maps.

3.2 Data collection setup

3.2.0.1 CPR dataset

For the CPR dataset, the sensors were setup as shown in Figure 3.1. The different types of data that were collected with the different types of sensors are mentioned in Table 3.1. Data from Azure Kinect, Canon DSLR camera, smartwatch, and the VL6180 sensor are stored in real-time on a laptop while data from the GoPro camera was transferred after each participant finished to the same laptop. The collected dataset can be found at
https://drive.google.com/drive/folders/16f-1YaFIpey53A1BGeSe2LREzIYF-8AH?usp=sharing

3.2.0.2 Hand depth dataset

For the hand depth dataset, only the Kinect and Canon cameras were used. The participants were asked to move their hands, one hand at a time back and forth in-front of the Canon and the Kinect cameras displaying various hand poses as shown in Figure 3.3. The hand dataset can be downloaded from
https://drive.google.com/drive/folders/1dFCp7cc1Ai7xwaDgr0BBVkcWkaPP2YC9?usp=sharing

Sensor	Data type
Azure Kinect	color images, depth images
Canon DSLR camera	color images
Chest worn GoPro camera	color images, IMU, audio
Smartwatch	IMU
VL6180 sensor	Time-of-flight dpeth sensor data

Table 3.1: Data sources

3.2.1 Azure Kinect camera

Azure Kinect camera can be used to obtain aligned color and depth maps which contains a time-of-flight (TOF) depth sensor. There are multiple settings to choose from which provide various resolutions and frames per second settings for the camera. We utilized the frame rate to be 30 frames per second. This resulted in 30 color and depth images per second. Color image resolution was 1280x720. The field of view of the color camera was 90 x 59 degrees in the horizontal and vertical directions. The color image output format was MJPEG. The exposure time for the camera was set automatically. The pixel resolution for the depth camera was set to 512 x 512 with frame rate of 30 fps which was synchronized with the color images. The operating range for the depth sensor under these settings was 0.25 - 2.88m. The depth maps were aligned with the color images using the C++ Azure Kinect SDK. All our subjects in the experiments were within this depth range [187]. For depths between 0.5 - 3.0m, it was observed that the Azure Kinect camera has a depth measurement accuracy of under 2mm [188] [189] which is sufficient for our needs. Azure Kinect SDK [187] and Open3D [190] python library was used record data from Azure Kinect and to align depth images to the color image coordinate frame and resolution.

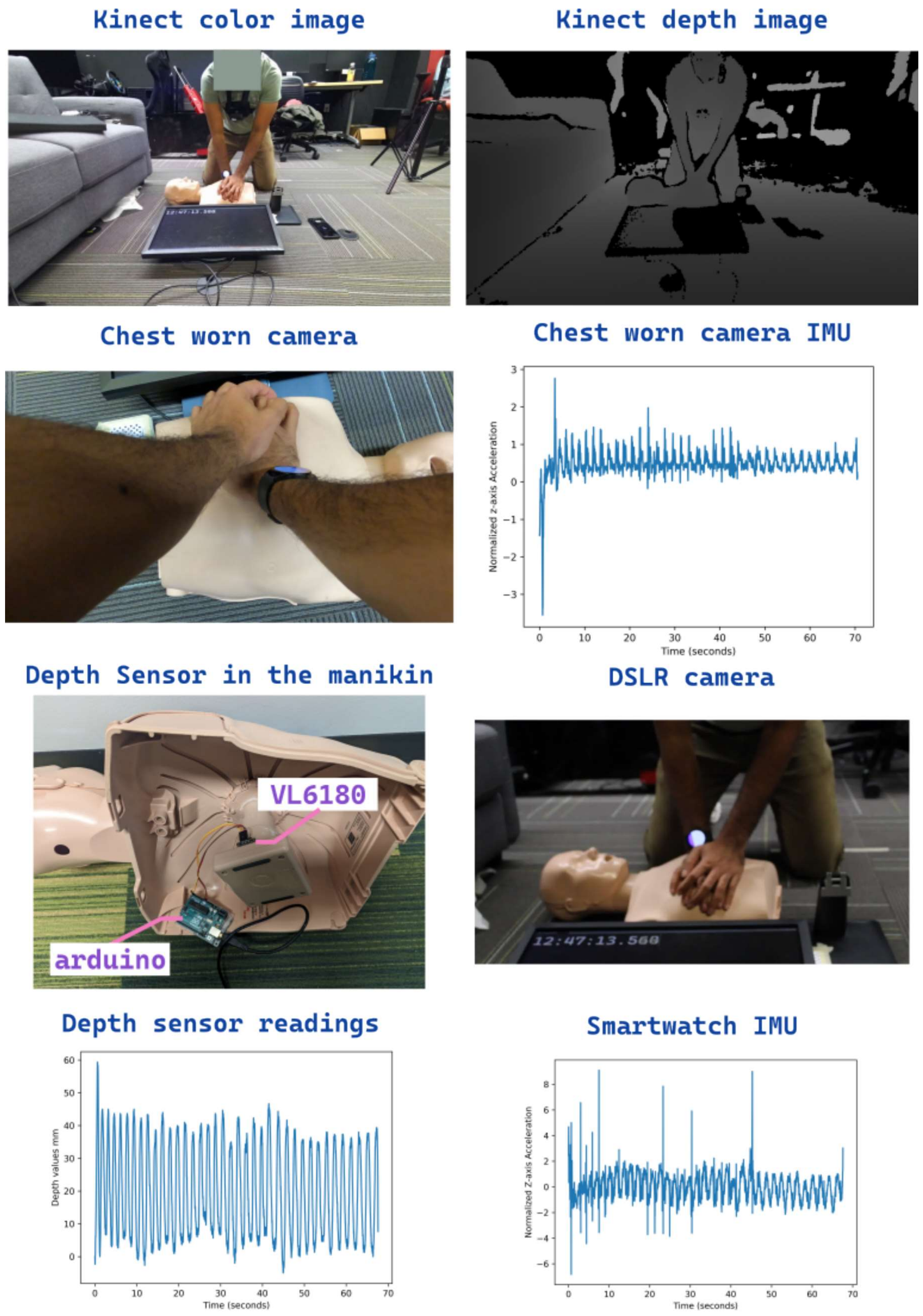


Figure 3.2: Data sources

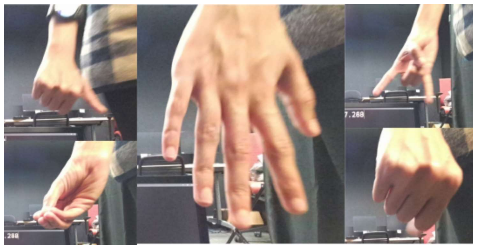


Figure 3.3: Hand poses for hand depth dataset

3.2.2 Canon DSLR camera

EOS Rebel T100 camera with the lens EFS 18-55mm was used to capture video of the experiments. The focal length was fixed to around 1m. The focal length was adjusted to that the manikin, hands and the time-stamp monitor will fill the field-of-view of the camera. The frame rate of the camera was set to 30 FPS. The image resolution of the camera was set to 1920 x 1080 pixels.

3.2.3 GoPro chest-worn camera

Although we planed to use a smartglass to capture ego-centric videos, from several pilot studies we realized that the videos captured by the smartglass is extremely shaky due to the motion of the wearer's head. Furthermore, the wearer might look around towards other EMS providers or objects and this will break the view away from the manikin. Therefore, we decided to use a chest-worn camera. We used the GoPro Hero 9 Black camera which can record color images (video), audio and IMU data from accelerometer and gyroscope. The color image resolution is 2704 x 1520 and the frame rate is 60 FPS. The audio was recorded at 48 KHz. Accelerometer and gyroscope data were recorded at 150 Hz.

3.2.4 Smartwatch

Accelerometer, gyroscope and magnetometer readings are recorded with Google Pixel Watch 2 at a rate of around 50Hz. We developed an Android app called sensor_stream that can stream IMU data from the smartwatch to the computer in real time through WiFi. A python program can read this stream of data from the laptop. This app can be accessed here: https://github.com/sleekEagle/sensor_stream

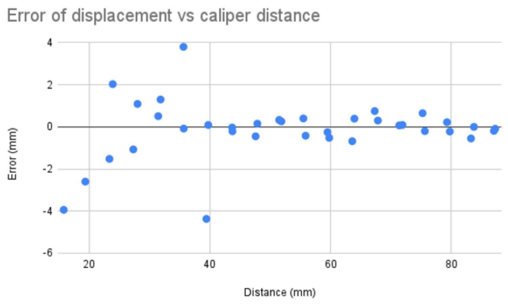
3.2.5 VL6180 depth sensor

This sensor that was embedded in the manikin is used to get the ground truth CPR compression depth and rate. VL6180 sensor uses an infrared based Time-of-Flight sensor to measure depth. VL6180 has an effective range of 1-160 mm and a sampling rate of around 100Hz. According to the specs the depth error of the sensor is around 1mm. To verify this we performed an experiment as shown in Figure 3.4. First, the sensor was attached to a vise as shown in (a) and the ground truth distance was measured with a vernier caliper. The VL6180 was attached to an Arduino Uno REV3 and the data was recorded into a computer. A C program running in the Arduino reads the depth values from the sensor which a python program running in the laptop is receiving these data from the Arduino. We get the mean distance reading considering different window sizes. Figure 3.4 (b) shows the error when the window is 25ms while (c) is for 50ms. It can be seen that for distances between 30mm and 90mm, the error is less than 1mm. (d) shows the standard deviation of 2000 samples and its variation for different actual distances. It can be seen that the standard deviation increases when the actual distance increases. The nominal distance that we measure is around 40 mm and we mean-filter the depth readings with a window length of 33 ms which falls under the region where the error is less than 1mm.

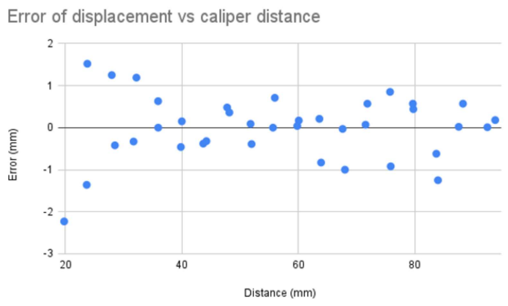
3.3 Methods

24 participants contributed to the dataset. Each participant was asked to perform several CPR sessions of around one minute at different depths and rates. Most of the participants contributed to the hand depth dataset which was done at the same time-slot right after or before the CPR experiment.

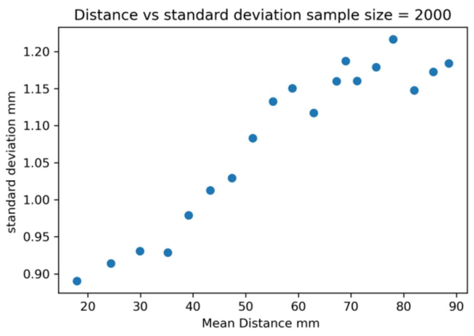
(a) setup



(b) error: 25 ms



(c) error: 50 ms



(d) error vs distance

Figure 3.4: VL6180 accuracy validation

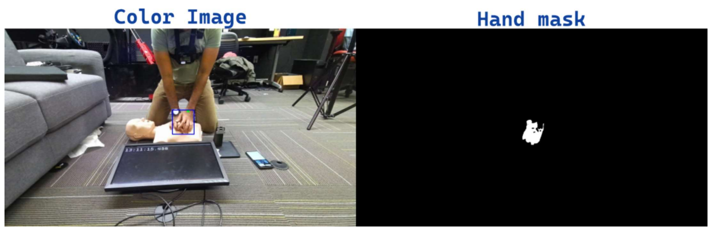


Figure 3.5: Hand transformation

3.3.1 Using timestamps

Different methods were used to obtain the timestamps for different sensors and devices. A display with a timestamp with the format hh:mm:ss was kept next to the manikin so the Kinect and Canon cameras can see it. This timestamp was visible to the chest work GoPRo camera at certain times. This timestamp was generated by the laptop which is used to collect data. For each color image from the Kinect and Canon camera, the timestamp was obtained by performing Optical Character Recognition (OCR) using Google vision API. One image per every second was sent to the Google vision API and the timestamps for the other images were obtained by interpolation. The output from the Gopro camera contains a timestamp and the timestamp displayed was used to synchronize the Gopro clock and the clock used by Kinect and Canon cameras. The sensor_stream app sends the smartwatch timestamp at each sensor event occurrence. The python program that stores the IMU data also stores the local time of the laptop when the data sample was received along with the IMU sensor readings. For all the purposes in this study, the laptop timestamps are used. The python program that is reading the depth sensor readings from the VL6180 sensor stores the local timestamp of the laptop when the depth sensor reading was received. Note that there is a slight discrepancy between the actual times of the smartwatch IMU and depth sensor readings and the stored timestamps for these devices, this difference was assumed to be negligible due to fast data transfer rates.

3.3.2 Data Selection

Valid sections of CPR was detected by filtering for standard deviation of the depth sensor readings. Each valid section of CPR was termed a session in this study. All the sensor readings for each session were stored in separate directories.

3.3.3 Data Processing

For each color image from the Kinect and Canon cameras we obtain several derivative quantities as shown in Figure 3.5. First, the bounding box for the hand that is visible (top hand) is obtained with Grounding Dino [191]. Next, hand segmentation maps (mask) were generated with segment anything model [192]. The wrist position of the hands were detected using two methods. First we utilize Google mediapipe [193] to detect the wrist position of the visible hand over all of the image frames of a given CPR video. We initially detected 21 hand keypoints for a given hand. But for our study we only use the wrist keypoint. The wrist position given by this method can be used to estimate the CPR compression rate and depth after measuring depth to these pixels. We experiment with a second method to track the wrist position which is based on optical flow. The wrist position is manually detected for the first frame of the CPR video for all the CPR sessions. This position was tracked with Lucas-Kanade based optical flow [194] implemented in OpenCV library [195] for the rest of the images in the video. Note that for practical application, this method required human intervention for each CPR session and can be cumbersome in the real world. We do this for the purpose of comparison.

To create the ground truth CPR compression depth and rate, we analyzed the depth from the VL6180 sensor. Peaks and valleys are detected with Python Scipy library [196] as seen on Figure 3.6. The mean value of the number of peaks and valleys are taken as the number of compressions in a given time. CPR rate is taken by dividing the number of CPR compressions by the time. To get the ground truth depth, we subtract the valley depths from the consecutive peak depths.

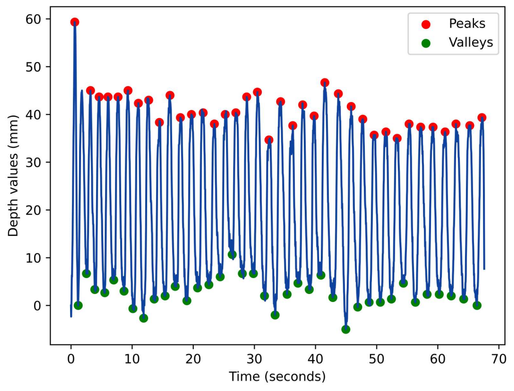


Figure 3.6: Peak Detection

3.3.4 Data interpolation and synchronization

3.3.4.1 Smartwatch data

The data rate of the IMU of the smartwatch varies around 50Hz. But a constant data rate is demanded by the machine learning models that were trained to predict depth and the CPR rate (more on these are described later). Hence, all the smartwatch data was interpolated to be 100Hz. To synchronize all the data, the local timestamp of the laptop was used.

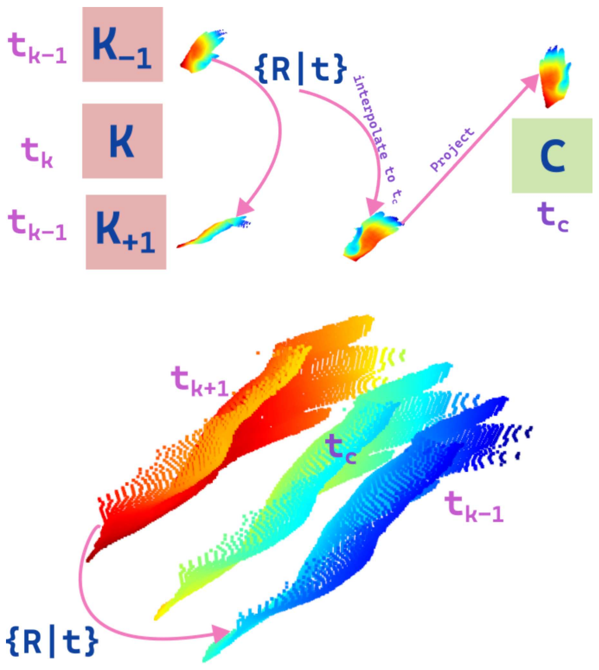


Figure 3.7: Hand transformation

3.3.4.2 Hand depth dataset

Special considerations for data synchronization was necessary for the hand depth dataset. As mentioned before, we obtained color and depth images from Kinect camera and color images from the Canon camera. The color and the depth images from the Kinect camera are synchronized at hardware level. But, the images from Canon and Kinect are not synchronized. But, in order to obtain color and ground truth depth image pair for the Canon camera, the images from the canon and the Kinect must be synchronized. This process is depicted in Figure 3.7. First, a color image from the Kinect camera is selected. This is depicted as K. Then the timestamp of this image is obtained (t_k). Next, the Canon image with the closes timestamp to t_k is selected (Canon image C with timestamp t_c). Next, the Kinect images with a timestamp immediately lower than t_c (K_{-1}) and immediately greater than t_c (K_{+1}) were selected. Next, 3D point clouds of the hands from these images are obtained by projecting 2D coordinates of the hand settings on the image into 3D space with the intrinsic matrix of the Kinect camera. Next, the transformation between the two points clouds obtained from the images K_{-1} and K_{+1} is obtained with open3D iterative closest point algorithm (ICD) [190]. Next, the point clouds obtained with the image K_{-1} was transformed into the timestamp t_c using the transformation matrix obtained in the previous step. Then this transformed point cloud is projected into the Canon 2D image plane with the Canon camera's intrinsic matrix to obtain the hand depth map for the Canon camera.

3.3.5 Camera calibration

The Azure Kinect camera provides the color images and aligned depth images. In this study we are interested in measuring depth to the hands of our participants. This depth will be used to obtain the 3D coordinates of the wrist. The 3D motion of wrist can be analyzed to estimate the CPR compression depth. When we obtain the color images and the depth images, we can utilize the intrinsic camera matrix to project 2d (x,y) coordinates along with the depth at that point to 3D coordinates (X,Y,Z). We can estimate the intrinsic matrix of a camera by camera calibration. Similarly, the intrinsic matrix of the canon DSLR camera can be estimated. The Azure Kinect camera provides us with a depth map. But, the Canon camera does not have a depth sensor. We meed to measure the depth of the hands of our participants with respect to the Canon camera. To do this we need to estimate the relative position and rotation of the Canon camera with respect to the Kinect camera. Then a 3D point $(X, Y, Z)_{kinect}$ (a 3D points from Kinect camera coordinate frame) can be transformed into Canon coordinate frame $(X, Y, Z)_{canon}$. The relative position and orientation of Canon camera with respect to the Kinect camera can be obtained by obtaining image pairs of a calibration pattern from both Kinect and Canon cameras and performing stereo camera calibration. The same set of images of the calibration patterns can be used to calibrate the cameras to obtain the intrinsic matrices. We use Matlab camera calibration [197] and stereo camera calibration [198] toolboxes to do the calibration. An example of an image pair is shown in Figure 3.8. The Azure Kinect, Canon intrinsic calibration matrices (k_{kinect} and k_{canon}) and the stereo calibration matrix T are obtained as follows.

$$k_{kinect} = \begin{bmatrix} 615.87 & 0 & 640.80 \\ 0 & 615.91 & 365.54 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

$$k_{canon} = \begin{bmatrix} 1646.60 & 0 & 634.93 \\ 0 & 1647.30 & 361.86 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

$$T = \begin{bmatrix} 0.994 & 0.011 & 0.107 & 99.045 \\ -0.011 & 0.999 & -0.004 & -34.41 \\ -0.107 & 0.003 & 0.994 & 29.12 \end{bmatrix} \tag{3.3}$$

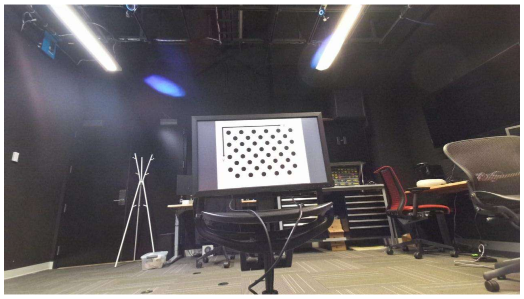
Now the 3D points in the Kinect coordinate frame $(X, Y, Z, 1)_{kinect}$ can be transformed to a 3D point in the Canon coordinate frame $(X, Y, Z, 1)_{canon}$ as

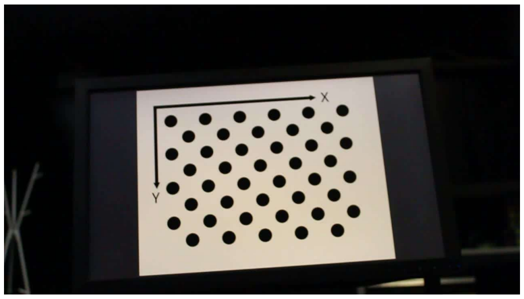
$$(X, Y, Z, 1)_{canon} = T.(X, Y, Z, 1)_{kinect} \tag{3.4}$$

3.4 Dataset Description

3.4.1 CPR dataset

The CPR dataset contains data from 24 participants and 239 CPR sessions. The distribution of the average CPR compression depth, rate and the length of these sessions are calculated as mentioned in section 3.3.3 are shown in Figure 3.9. Note that we calculate the ground truth CPR compression depth and rate are calculated based on the VL6180 sensor embedded in the manikin.


(a) Image from Kinect



(b) Image from Canon

Figure 3.8: Calibration image pair

3.4.2 Hand depth dataset

2720 Kinect and Canon image pairs and their respective depth maps were collected from 17 participants. Statistics of the depth values of the dataset is shown in the Figure 3.10. In the histogram of the Figure 3.10 the frequencies were counted as the number of pixels of a given depth that belonged to a region of a hand.

3.5 Models, prediction and results

3.5.1 CPR quality evaluation with smartwatch

In this section we provide details on the model that we call SW-CPR model to predict the CPR compression depth and rate. We created a dataset (a subset of the CPR dataset) that consists of 5 second segments of smartwatch IMU data and ground truth depth values from VL6180 sensor with 50% overlap. For each 5 second segment our model outputs two quantities. It predicts the mean CPR compression depth. It also tries to recreate the ground truth depth signal. The first loss is calculated by the mean squared error between the mean predicted depth and ground truth depth (L_{depth}). The second loss is calculated by the mean squared error between the depth signal recreation and the ground truth depth signal (L_{rec}). The CPR compression is calculated by detecting peaks and valleys of the recreated depth signal as mentioned in section 3.3.3. The structure of our model which just 5 layers is shown in Figure 3.11. The model contains two convolution layers, a single LSTM layer and two fully connected layers.

3.5.1.1 Training and evaluation

The loss of the model was defined as

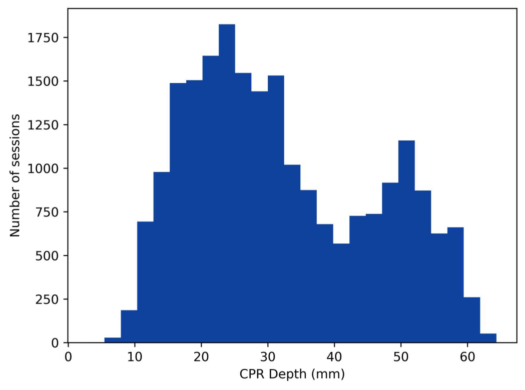
$$L = L_{depth} + L_{rec} \tag{3.5}$$

The model was trained on data from 12 participants which amounted for 2368 5-second samples. It was evaluated on data from the other 12 participants which contained 2091 samples. The evaluation results are shown in Table 3.2. Our SW-CPR model can predict the CPR compression depth with an error of 6.6mm and the compression rate with an error of 5.4 compressions per minute. We compared the compression rate estimation with two other methods. Python numpy based peak detection algorithm performed with a error of 22.8 compressions per minute and Fast Fourier Transform based (FFT-based) method had an error of 21.3 compressions per minute. Note that the two latter methods used the Z-axis accelerometer readings from the smartwatch to detect peaks and valleys to calculate the CPR compression rate. FFT based method converted the raw accelerometer readings into the Fourier domain and gave the most dominant frequency as the output after discarding the DC-component. Figure 3.12 shows further analysis of the performance of the SW-CPR model.

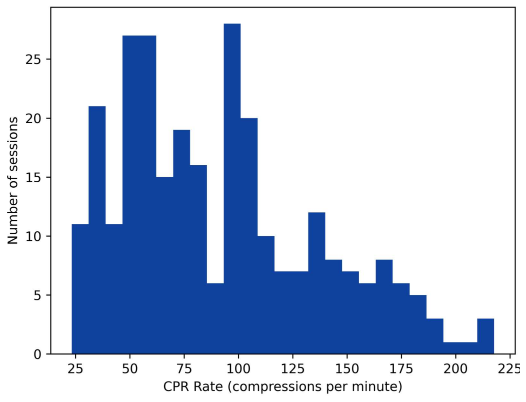
3.5.2 CPR quality evaluation with vision

3.5.2.1 Detection and tracking of the wrist position

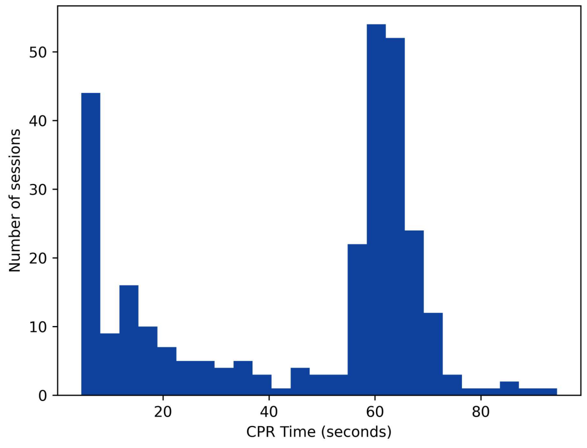
We experiment with two methods to detect and track the wrist position of the hand when a participant is performing CPR on the manikin.



(a) Depth histogram



(b) Rate histogram



(c) Time histogram

Figure 3.9: CPR dataset stats

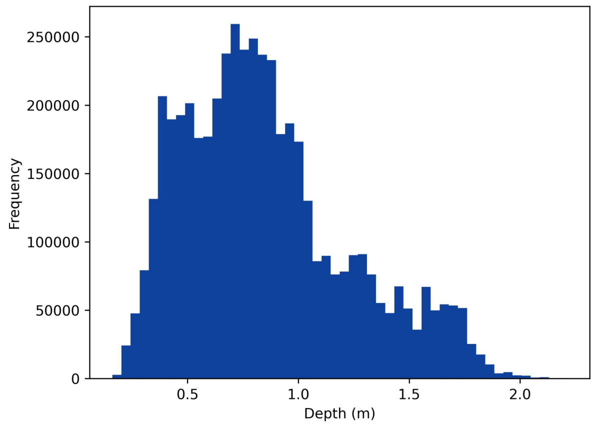


Figure 3.10: Depth dataset statistics

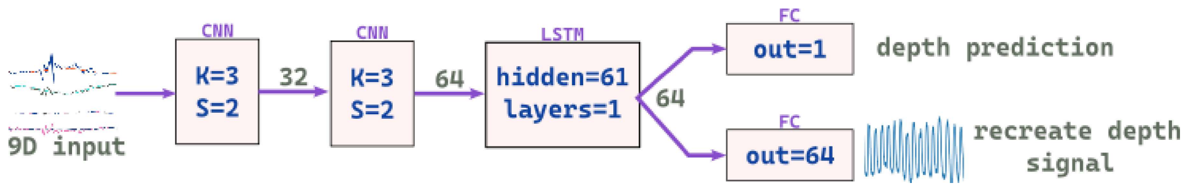


Figure 3.11: SW-CPR model

Prediction	**Method**	**Error**
Depth	SW-CPR model	6.6 mm
Rate	SW-CPR model	5.4 compression/min
	np peak detection	22.8
	FFT-based	21.3

Table 3.2: Performance of the SW-CPR model

3.5.2.2 CPR compression rate

In order to measure the CPR compression rate and depth with vision, the wrist position was detected from the image as mentioned in section 3.3.3. The Y-axis variation of the pixel location of the wrist position was obtained as show in Figure 3.13. The peaks and valleys are detected with Python Scipy library [196]. The CPR rate was calculated as mentioned in equation 3.6.

$$CPR_{rate} = (npeaks + nvalleys) \times 0.5/time \tag{3.6}$$

3.5.2.3 CPR compression depth

The 3D position of the wrist can be obtained with x and y pixel position of the wrist and the depth at this pixel. The ground truth depth from the Kinect depth map can be used for the depth. First, the peaks and valleys are detected as in section 3.5.2.2. The x and y coordinates of the wrist was projected to the 3D points X,Y and Z using the corresponding depth at that pixel. We then take the distances between the 3D positions at peak and valley locations. For a given CPR session we take the average depth of all these distances to get average CPR depth.

Table 3.3 shows the CPR compression rate and depth errors for all the participants and the sessions. CPR rate error is in compressions per minute while the depth error is in mm.

Participant	Session	Kinect	
		Rate error (comp/min)	Depth error (mm)
P0	0	1.3	0.29
	1	5.2	1.1
	2	34.4	4.66
	3	38.7	3.02
	4	7.6	2.03
	5	20.5	2.34
	6	5	0.43
	7	3	0.25
	8	46.5	0.38
	9	4.3	4.07
	10	67.9	42.24
P1	0	69.6	14.4
	1	36.5	13.2
	2	7.36	73.4
	3	20.45	121.9
	4	2.35	0.6
	5	8.92	5.8
	6	21.4	58.6
	7	24.5	12.8
P2	0	5.09	54.68
	1	24.65	4.93
	2	4.57	4.75
	3	6.7	4.71
	4	5.47	4.75
	5	5.57	3.87
	6	12.69	15.12
	7	3.19	22.36
	8	7.03	3.02

	9	2.71	5.45
	10	31.84	4.77
P3	0	14.2	105.93
	1	43.3	12.45
	2	1.5	22.9
	3	31.7	58.1
	4	42.4	37.9
	5	9.7	32.8
	6	4.7	5.1
	7	6.5	11.2
	8	30.7	65.1
P4	0	0.89	1.55
	1	18.72	27.63
	2	3.17	29.51
	3	4.4	46.25
	4	86.4	22.6
	5	14.44	55.9
	6	35.63	2.36
	7	42.36	1.02
P5	0	1.42	67.71
	1	7.39	81.95
	2	14.83	13.67
	3	0.42	40.01
	4	37.75	2.06
	5	4.67	47.72
	6	61.38	4.36
	7	7.7	78.41
	8	9.28	50.18
	9	1.74	35.4
	10	11.68	10.2
P6	0	10.51	2.83
	1	19.06	52.19
	2	46.97	32.76
	3	34.91	47.14
	4	0.54	11.5
	5	8.76	25.11
	6	77.76	54.41
	7	54.64	19.83
	8	97.47	5.2
P7	0	12.63	12.59
	1	8.51	2.12
	2	16.23	0.56
	3	1.41	2.32
	4	40.3	30.1
	5	0.29	0.57
	6	9.35	7.09
	7	8.99	12.2
	8	11.58	9.39
P8	0	1.56	1.17
	1	3.91	49.97
	2	34.7	1.36
	3	29.99	0.15
	4	3.06	0.67
	5	0.81	1.16
	6	2.68	0.15
	7	1.32	0.42
	8	9.88	2.97

	9	9.49	1.71
	10	11.56	1.74
	11	24.99	13.47
P9	0	116.35	13.34
	1	120.84	26.13
	2	41.65	0.11
	3	31.9	7.08
	4	25.96	9.13
	5	27.39	10.09
	6	65.85	16.57
	7	33.13	19.54
	8	33.13	39.6
P10	0	41.11	83.83
	1	14.11	35.26
	2	16.75	44.62
	3	31.4	33.91
	4	15.98	3.48
	5	23.72	1.32
P11	0	9.49	0.63
	1	4.82	0.77
	2	9.08	0.99
	3	8.73	0.41
	4	12.54	3.03
	5	9.02	3.95
	6	8.13	292.65
	7	42.15	5.17
	8	2.21	1.65
	9	4.65	8.84
	10	0.02	173.61
P12	0	8.32	8.23
	1	29.54	0.5
	2	18	0.86
	3	1.81	1.77
	4	10.33	0.64
	5	2.96	0.38
	6	2.77	4.99
	7	1.23	0.01
P13	0	20.78	23.58
	1	2.1	7.53
	2	1.9	4.62
	3	4.16	8.14
	4	0.46	0.59
	5	13.32	2.54
	6	0.84	2.21
	7	0.03	1.76
	8	11.77	1.98
	9	0.25	24.8
P14	0	6.35	18.28
	1	5.79	15.87
	2	3.95	14.3
	3	11.6	4.04
	4	11.05	7.23
	5	4.06	18.42
	6	31.74	2.79
	7	59.21	6.1
	8	52.32	15.42
P15	0	4.51	9.25

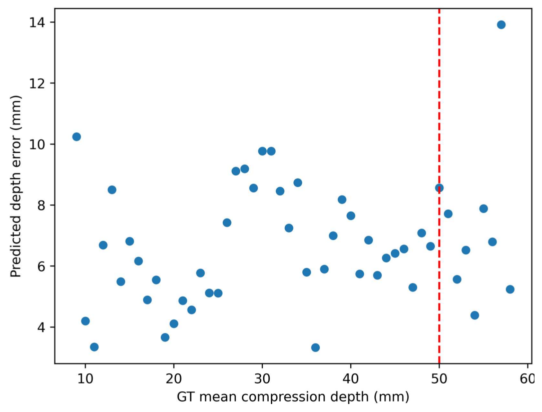
	1	10.95	12
	2	14.04	11.46
	3	3.71	13.24
	4	7.77	32.09
	5	13.76	12.31
	6	3.87	6.65
	7	21.55	35.13
	8	6.09	6.7
	9	0.15	33.33
	10	12.23	9.15
P16	0	2.32	3.44
	1	0.31	2.39
	2	13.52	0.61
	3	28.41	1.63
	4	22.69	0.81
	5	9.14	8.61
	6	61.14	22.87
	7	0.89	0.96
P17	0	27.24	6.86
	1	58.98	2.71
	2	0.19	1.52
	3	10.92	1.1
	4	0.57	5.27
	5	3.67	2.14
	6	1.33	5.31
	7	10.44	2.37
	8	0.17	3.85
	9	9.28	2.46
P18	0	20.79	2.34
	1	4.03	1.47
	2	11.07	7.47
	3	3.77	4.69
	4	1.43	8.33
	5	1.35	3.67
	6	0.19	3.35
	7	7.84	9.83
	8	10.41	3.26
	9	36.35	3.56
	10	2.3	2.18
	11	11.33	2.63
P19	0	6.74	1.51
	1	1.97	0.6
	2	6.16	5.6
	3	1.43	10.02
	4	1.76	12.19
	5	39.58	7.7
	6	15.26	12.22
	7	2.39	6.26
P20	0	11.7	1.64
	1	5.57	10.58
	2	5.51	3.56
	3	7.22	4.89
	4	7.37	9.52
	5	3.26	15.11
	6	73.12	6.1
	7	2.77	6.85
	8	17.68	2.76

	9	21.77	5.14
	10	9.17	4.84
	11	1.96	9.41
	12	3.67	5.73
P21	0	6.94	8.89
	1	21.45	16.09
	2	16.08	12.09
	3	9.38	7.85
	4	8.13	6.89
	5	5.49	8.89
	6	2.42	9.84
	7	28.66	3.99
	8	7.59	15.04
	9	2.56	4.17
	10	0.49	13.46
P22	0	8.17	1.2
	1	13.92	0.73
	2	0.8	3.14
	3	2.33	0.65
	4	13.02	1.71
	5	26.02	2.84
	6	4.05	0.25
	7	0.83	1.73
	8	0.09	1.19
	9	4.29	1.81
	10	4.32	0.33
	11	1.32	0.65
	12	7.86	1.75
P23	0	3.31	2.32
	1	6.52	2.73
	2	46.38	1.07
	3	16.31	2.65
	4	14.95	4.14
	5	6.39	0.39
	6	14.03	1.45
	7	18.51	1.76
	8	4.18	0.49
	9	37.88	2.55
	10	19.97	0.04
	11	39.24	2.31
Average		16.26	14.62

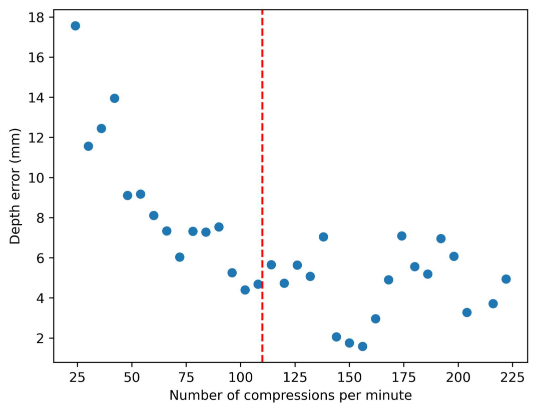
Table 3.3: CPR compression rate and depth performance

3.6 Conclusions

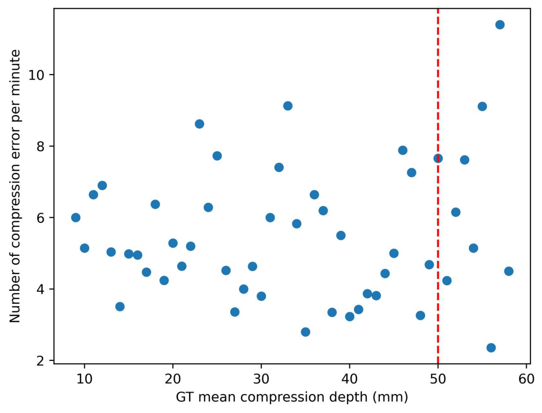
This Chapter describes collection, processing, and analysis of the multi-modal CPR quality estimation dataset and hand depth estimation dataset. CPR quality estimation dataset was collected to build and evaluate methods that can be used to estimate CPR compression depth and rate with smartwatches and with using cameras. Depth perception is an essential requirement for estimating CPR depth from cameras. But there are no suitable datasets that can be used to train models that can be used to accurately estimate depth and track hands. To close this research gap, a dataset that contain hand images and their respective depth maps were collected. Models that can be used to estimate CPR compression depth and rate with both smartwatch data and video were built and evaluated. This analysis shows that smartwatch can be used to accurately measure both of these quantities. Video based models show promise and requires further improvement to improve their accuracy.



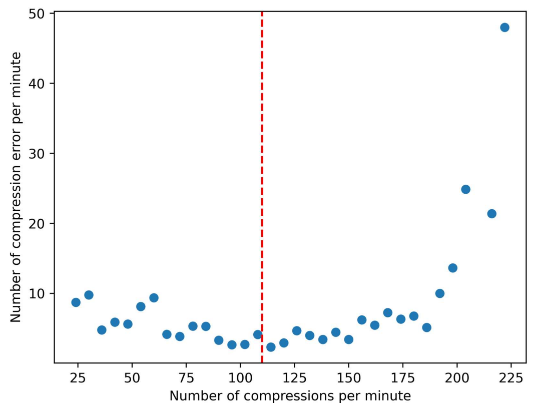
(a) Depth error vs GT depth



(b) Depth error vs GT rate



(c) Rate error vs GT depth



(d) Rate error vs GT rate

Figure 3.12: Performance of the SW-CPR model

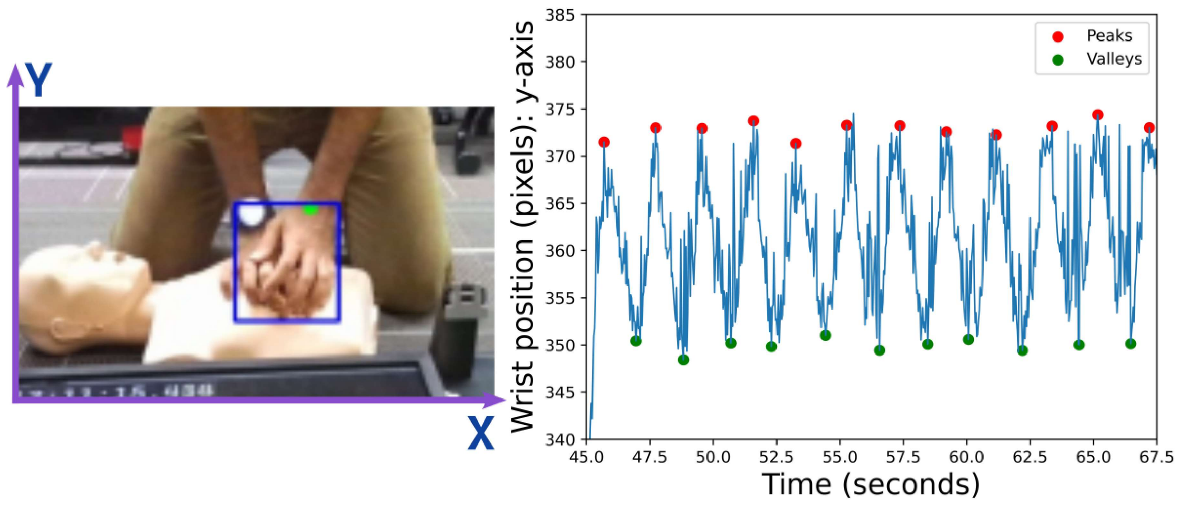


Figure 3.13: Visual CPR rate estimation

CHAPTER 4

MONOCULAR DEPTH ESTIMATION WITH DEFOCUS BLUR

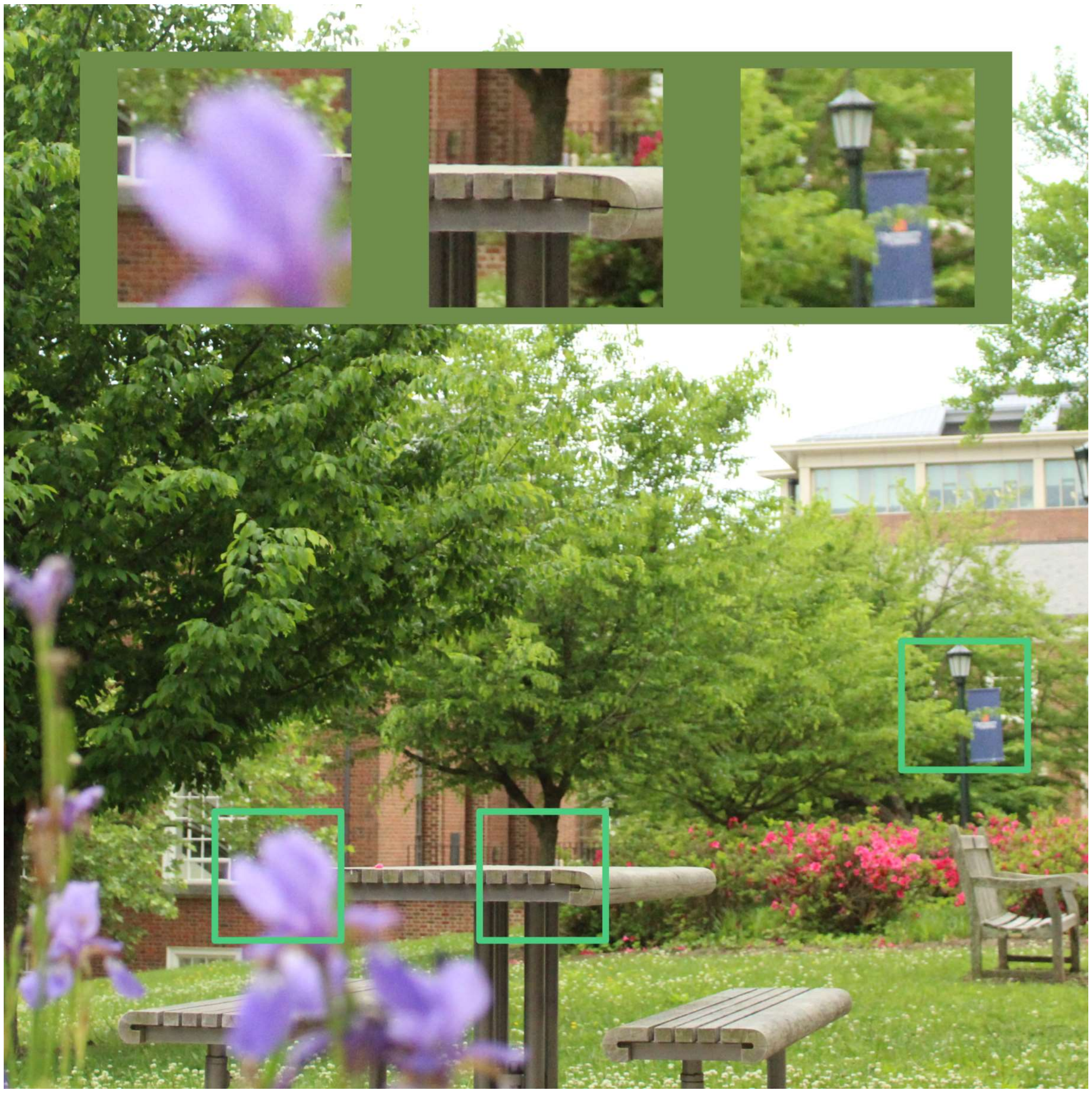


Figure 4.1: Defocus blur of a photo

4.1 Introduction

Depth perception is a critical component of cognitive assistant solutions for EMS providers. Computer vision based depth estimation has many applications such as augmented and virtual reality (AR and VR) [35], autonomous robotics [36], background subtraction, and changing the focus of an image after it was taken [37][38][39]. Techniques such as structure from motion, structure from shading, shape from structured light, shape from defocus blur, depth from focus, multi-view stereo and Time-of-Flight (ToF) sensors can be used to estimate the depth of a scene [29, 30]. Active methods such as structured light and ToF sensors need specialized hardware and are power hungry. Stereo techniques measure depth by relying on multiple cameras to take several pictures of the scene. Usage of multiple cameras increase the weight and power consumption of the wearable devices. Furthermore, stereoscopic depth estimation relies on point matching of the two images to measure depth. Errors in the point matching algorithms could result in increased errors of the depth estimation. Techniques such as structure-from-motion and depth-from-focus [40] require several images of a static scene to estimate it's structure. Also, the assumption about static scene does not hold when the scene is changing over time. Furthermore, structure from motion can only recover the depth of a scene up to a scale and cannot measure the absolute depth.

Single image defocus blur based depth estimation is a fairly under-explored topic in the literature [37] which utilizes the phenomena that certain objects in a photo appear more blurred than the others depending on the distance to those objects from the camera. Figure 4.1 shows an examples image with defocus blurring. This image is focused at the wooden table and the flower is blurred due to its proximity to the camera. The lamp post with the flag is also blurred because its further away from the focal point (wooden table). Measuring the amount of defocus blur at a point of an image can provide a way to recover the depth to the respective point in the real 3D world. As we will show in Section 4.4.3, this method is effective for close range depth measurements (typically under 2 to 3 meters) and out performs the state-of-the-art semantic based monocular depth estimation methods. This makes defocus blur-based depth estimation techniques ideal for measuring depth under many situations including in microscopic scenes [41, 42] and measuring depth to hands performing CPR and nearby objects used by first responders for a wearable camera.

Single image depth from defocus blur methods are not robust to changes of cameras. As we will show in our experiments, the performance of existing methods degrades significantly when they are trained on images taken from one camera and evaluated on images taken from another camera (even when they both image the same scene). This is due to the fact that different cameras will produce defocus blurs with different characteristics.

In this Chapter we describe a novel technique to estimate depth from defocus blur in a camera-independent manner. We exploit the optical physics equations that describe the relationships between various camera parameters and the amount of defocus blur. Our method can be used to train a deep learning model in a supervised manner on a dataset containing defocus blurred images taken from a single or multiple camera/s and respective ground truth depth maps. This trained model can be used to predict depth using images taken with a wide range of other cameras with a slight modification to the model (depending on the particular camera parameters of the new camera) and without the need for retraining. We also describe a novel method to estimate the camera parameters of a given camera with an easy to use calibration process. This will be particularly useful when the parameters for a certain camera cannot be obtained (certain manufacturers do not provide all the parameters in the data-sheets and/or the values are only provided as approximations). Having a general solution is important for first response applications because different first responder stations will be using different cameras.

Our main **contributions** are as follows:

- We show that depth from defocus technique can measure depth more accurately than the state-of-the-art techniques.
- We show that existing depth from defocus methods are not robust to changes of cameras the images are acquired with.
- This work is the first to devise a relationship between defocus blur and the blur created due to pixel binning.
- We present a novel depth from defocus blur method which is robust to images taken from a wide range of cameras, given camera parameters that describe a particular camera.
- We present a novel calibration technique to estimate the camera parameters based on several images taken from a given camera.

- Our methods have less estimation error than the state-of-the-art when performing depth from blur in a camera-independent manner. The error reduction is around 3cm under the DDFF12 dataset, 7cm under the NYU depth v2 dataset and around 5cm for the synthetic dataset we created.

4.2 Related Work

4.2.1 Depth from RGB images

Estimating depth maps from images can use various characteristics of images such as semantics, stereo matching, blur or differences in blur over a stack of images. [44–46]. Although stereo matching based and blur based depth measurements are seen as completely separate methods, Schechner and Kiryati [47] showed that both of them can be understood under the same mathematical formulation. A depth map can be estimated for the given image based on the domain knowledge on the structure of the objects in the image embedded in the estimation model [48–52]. Methods such as ZoeDepth [51] and VPD [52] have pushed the state-of-the art to be very accurate in measuring depth. However, a problem with these methods is that the estimated depth is only an approximation based on the structure of the objects. This makes these models sensitive to domain changes [29]. Also, techniques that can recover 3D structure from RGB images such as structure from motion can only estimate relative depths in a given scene [53].

4.2.2 Depth from defocus blur

The amount of defocus blur can be used to measure the depth of a scene from images. Since these methods rely more on blur which is a local feature of the image to estimate the depth, they are more robust to domain changes [29]. Shape/depth from focus methods aim to measure depth to a scene given a stack of images of different focus levels. A measure of the sharpness of each pixel of the images over the stack is calculated. The depth of a point is taken as the focus distance with the sharpest pixel. Various methods such as the Laplacian or sum-modified-Laplacian, gray-level variance and gradient magnitude squared were traditionally used to measure the sharpness [54, 55]. Modern methods utilize deep learning to automatically learn the sharpness measure from focal stacks [37, 40, 56]. But deep learning based techniques require a large amount of data to train [38].

Depth from focus methods that use a focal stack of the same scene has several drawbacks. First, they assume the scene is static during the time needed to acquire several images with different focus (focal stack). Second, an accurate registration of the images in the focal stack is needed due to focal breathing (slight change of the filed-of-view of the camera due to changes of focal distance) or small movement of the camera and/or the scene [57]. Therefore, more investigation on depth estimation with a single image is necessary. Depth from defocus/blur rely on measuring the exact blur on a single image to estimate the depth and cannot use the relative variation of sharpness/blurriness of a focal stack. Due to this, depth from blur can be used to estimate the depth from a single blurred image [29, 38, 58–60]. Certain works are also concerned about removing the blur at the same time as estimating depth [38, 58, 61].

Estimating depth from the amount of the blur of a single image is ill-posed. This is due to having two possible depth values for a given blur [29]. Researchers have take two different paths to solve this. One solution is hardware based. One example for this is changing the shape of the aperture (coded aperture) of the camera to a shape that can help avoid the ambiguity. Ikoma et al. [62] used deep learning to learn the optimal shape for an aperture and came up with a prototype camera to measure depth from blur. Another example is to use a light field camera which takes many pictures with closely spaced micro lenses placed inside the camera [63]. The second approach is to use the domain knowledge (e.g. the shape and sizes of objects in the scene) of the scene to remove the ambiguity. Our research falls into this category. Gur and Wolf created a model which can generate the depth map of a scene given the blurred image and the All-in-Focus (AiF) image [29]. They also makes certain assumptions about the shape of the blur circle. Usage of both AiF image and blurred images in making prediction makes this model less useful in certain situations because both of these images are not usually available from regular cameras. Many methods in the literature first estimate the blur of a given blurred image and secondly estimate the depth from the blur. Physical consistency between the estimated blur and depth has been used as a form of domain knowledge by Zhang et al. [60]. Lu et al. create two separate models to estimate the blur and the amount of focus (sharpness) of a given blurred image. They claim that this method provides better estimates of depth due to the capability of estimating both blur and the sharpness of an image [64]. But since sharpness is just the inverse of blur, a question remains that by estimating blur aren't we also estimating the (inverse of) sharpness. Ban et al. [41] extend depth from blur to microscopic images. Certain works focus just on blur

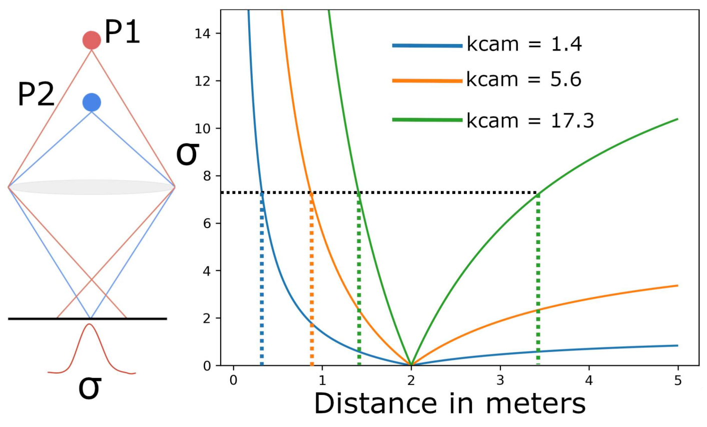


Figure 4.2: Left:Image formation in a simple camera system. Right:Blur vs. distance

estimation from a blurred image. Tai and Brown use hand crafted features of an image to estimate the blur map [65] while Zhuo and Sim assume that the edges in the images are step edges [66]. Cun et al. estimated the blur of a given image to separate the blurred and focused areas from a blurred image [67]. While all of the above methods assume that the blur is a single parameter (e.g. Gaussian or disk shape) Liu et al. expand our understanding by introducing a two parameter model. This model is also helpful in removing errors due to pattern edges [57].

4.2.3 Camera dependency of depth from blur

Certain characteristics of blur depend on the camera that is being used to acquire the images. The blurred image of a point has the same shape (but scaled) as the lens aperture. For example, if the aperture is circular, the blur of a point is also circular theoretically. But in practice this is a Gaussian due to diffraction [59]. In this research we assume all the apertures are circular in shape.

4.3 Approach

This section starts with a theoretical introduction to estimating depth from defocus blur then establishes the challenges faced by this technique and finally our solution.

4.3.1 Theory and Techniques

When imaging a scene with a camera, the points that are not in focus appear blurred and the points that are perfectly in focused appear sharp in the image. This phenomenon is called defocus blurring. To illustrate this, in the left side of the Figure 4.2, the point $P2$ that is in focus appears as a point in the image plane of the camera. A point $P1$ that is not in focus appears as a blur in the image plane where the pixel intensity is the highest at the center and gradually falls of as we move away. This can be modelled with a 2D Gaussian function as denoted in equation 4.1 with σ as the standard deviation, x and y are image coordinates.

$$G(x,y) = \frac{1}{2\pi\sigma} e^{-\frac{1}{2}\frac{x^2+y^2}{\sigma^2}} \tag{4.1}$$

σ depends on the distance to the point $P1$ from the camera center and several other camera dependent factors as shown in equation 4.2.

$$\frac{|s_1 - s_2|}{s_2} \cdot \frac{1}{(s_1 - f)} \cdot \frac{f^2}{N} \cdot \frac{1}{p} \cdot \frac{out_{pix}}{sensor_{pix}} = k_r \cdot \sigma \tag{4.2}$$

In equation 4.2, s_2 is the depth (distnace to the)f is the focal length of the camera, N is the f-number, p is the pixel width, out_{pix} is the number of pixels in the final image, $sensor_{pix}$ is the number of pixels in the image sensor, s_1 is the focus distance, k_r is a constant that depends on the camera [199]. Many cameras allow user to change s_1 thereby focusing the camera at different distances. we define a camera dependent parameter k_{cam} as shown in equation 4.3.

Camera/device	Lens	f (mm)	N	K_{cam}
Cannon EOS Rebel T7	EF-S	18	4	15.54
		55	5.6	105.58
	EF 50mm	50	1.2	406.16
	EF 70-300mm	70	5.6	172.35
Nikon D7500	Nikon AF-S	50	1.8	240.40
	AF-S DX NIKKOR	18	3.5	15.76
		18	5.6	9.85
		55	3.5	149.98
		55	5.6	93.73
Sony Alpha 7 IV	FE PZ 16-35mm	16	4	8.92
		35	4	43.13
		16	22	1.62
	FE 70-200 mm	70	2.8	250.94
		200	2.8	2196.48
		70	22	31.93
Google Pixel 7 Pro	wide	25	1.85	50.39
	telephoto	120	2.55	1577.82

Table 4.1: k_{cam} of some popular cameras

$$\frac{|s_1 - s_2|}{s_2} \cdot k_{cam} = \sigma \tag{4.3}$$

where $k_{cam} = \frac{1}{(s_1-f)} \cdot \frac{f^2}{N} \cdot \frac{1}{p} \cdot \frac{out_{pix}}{sensor_{pix}} \cdot \frac{1}{k_r}$

$G(x, y)$ is the response of the camera system to a point target and is called the point spread function (PSF). We can obtain the defocus blurred image $B(x, y)$ by convolving the the perfectly focused image $F(x, y)$ with PSF $G(x, y)$ as show in equation 4.4.

$$B(x, y) = G(x, y) * F(x, y) \tag{4.4}$$

Equation 4.4 explains the blur solely due to defocus blurring. An additional blurring can occur due to various other reasons such as filtering in the camera hardware (e.g. to reduce noise), pixel binning, color filter mosaics, analog/digital image processing, analog to digital conversion, etc. [200]. This additional blurring can also be modelled as a convolution with another Gaussian function $Q(x, y)$ having a standard deviation γ which we assume to be constant for a given camera stetting. The final image can be obtained by

$$I(x, y) = Q(x, y) * G(x, y) * F(x, y) \tag{4.5}$$

All we can observe is the final image I. We show that the combined blurring (from defocus and due to other reasons described above) can also be modelled with a Gaussian PSF (refer the Appendix) and we can estimate the standard deviation of this PSF (λ) at each pixel in I. After estimating λ for a given image and when γ is known we can obtain σ as shown in the equation 4.6.

$$\sigma = \sqrt{\lambda^2 - \gamma^2} \tag{4.6}$$

Substituting equation 4.6 into equation 4.3 we can obtain equation 4.7.

$$\frac{|s_1 - s_2|}{s_2} \cdot k_{cam} = \sqrt{\lambda^2 - \gamma^2} \tag{4.7}$$

The right side of the Figure 4.2 shows the variation of σ with different distances (s_2) and under different cameras. For a given camera (hence for a given k_{cam}), we can estimate the σ from a given image and then estimate s_2. For certain sections of the curve (e.g. curve of $k_{cam} = 17.3$ at the shown value of σ), estimating s_2 is ambiguous since there will be two s_2 values for a given σ. This limitation can be mitigated by using a learning based model to estimate s_2. Another observation is that the value of σ depends on k_{cam}. This poses the main problem that we are addressing in this chapter. If a model was trained to estimate depth using data from a camera with one k_{cam}, this model will fail to predict the depth accurately for images taken with a camera having a different k_{cam}. Furthermore, the sensitivity of σ to the distance diminishes as the distance increases. Hence the effectiveness of the defocus blur based depth measurement techniques will also lessen with increasing distance. This limits the effectiveness of depth from defocus blur techniques to close range; as a rule of thumb, to distances less than 2m.

Table 4.1 shows some camera models and their k_{cam} values based on the particular lens/settings used. Please refer to the Appendix for a more detailed calculation and for k_{cam} values for more cameras/settings.

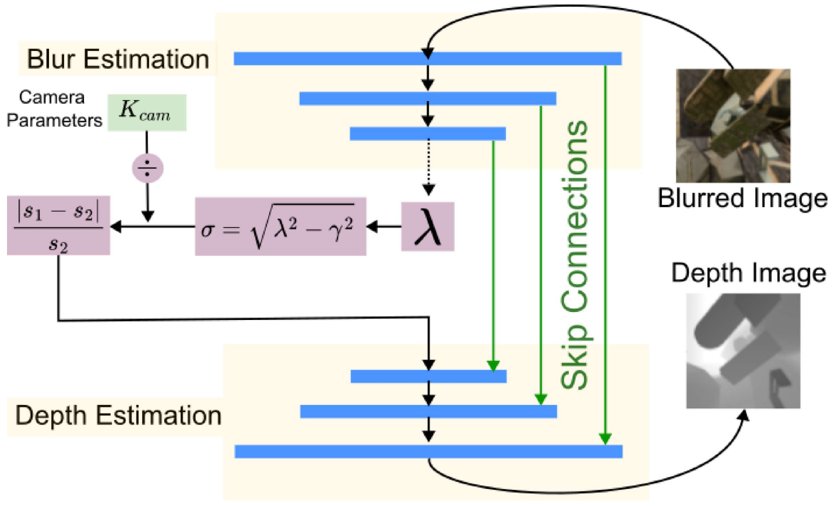


Figure 4.3: Our Model

4.3.2 Our solution

The operation or our model is shown in Figure 4.3. Both Blur Estimation and Depth estimation sections are CNN based neural networks inspired by the defocusnet [38]. Given a defocus blurred image, the blur estimation model estimates the PSF standard deviation λ at each pixel of the image. Then we calculate the standard deviation of the PSF solely due to defocus blurring σ according to equation **??**. Next we divide the obtained σ with k_{cam} to obtain $\frac{|s_2-s_1|}{s_2}$ which does not depend on either k_{cam} or γ. Then $\frac{|s_2-s_1|}{s_2}$ is sent to the depth estimation model to estimate s_2. Note that we require a learning based model such as a CNN to estimate s_2 due to the ambiguity explained in the previous section and to impart semantic domain knowledge of the image into the estimation process via the skip connections.

When we train our model, we calculate two types of losses; blur estimation loss and the depth estimation loss. Blur estimation loss (L_b) is calculated at the prediction of λ. Ground truth L_b can be obtained with equation 2 with known camera parameters at the training time. Depth prediction loss (L_d) is calculated comparing the predicted and ground truth depth maps. The final loss is obtained by,

$$L_{total} = L_d + b\_weight \cdot L_b \tag{4.8}$$

where b_weight is a parameter used to scale L_b.

4.3.3 Defocus Blur Calibration

Assume we train our model with images from a certain camera and need to use this already trained model to estimate depth using images from another camera with a different k_{cam} and γ. In this section we present our novel method that can be used to estimate these parameters for a given camera. We call this method the ”Defocus Blur Calibration”. Note that defocus blur calibration is different from but requires the conventional camera calibration where camera intrinsic and distortion coefficients are estimated. The steps for defocus blur calibration are as follows.

1. Fix the focal distance of the camera at s_1 (we used $s_1 = 2m$ in our experiments) and calibrate the camera (in a conventional sense) with a calibration pattern [201]. We have used an asymmetric circular pattern as can be seen in Figure 4.4. Maintain a rough distance of around $\frac{s_1}{2}$ from the camera to the calibration pattern. After this calibration, we can estimate the distance to a given point on the calibration pattern that is visible in a given image.

2. Capture two images of a circular calibration pattern (preferably the same pattern that was used in step 1) while maintaining a distance of $\frac{s_1}{2}$ (we used $1m$) from the camera to the pattern. The first image is obtained with the camera focused on the pattern ($s_1 = 1m$) and the second image is obtained while maintaining $s_1 = 2m$. Since the first image is focused on the calibration pattern, the circles on the pattern will appear sharp as shown in the upper part of Figure 4.4. The second image will look

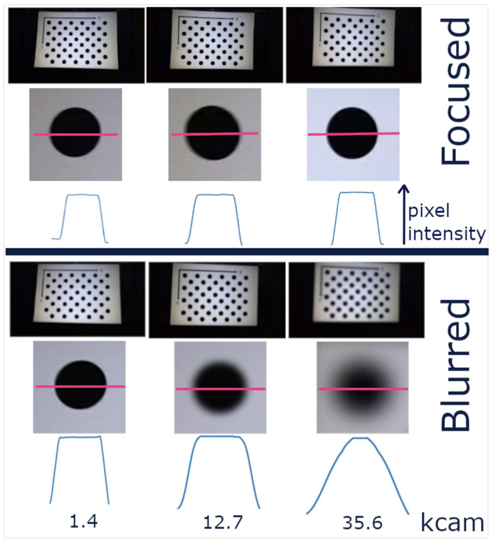


Figure 4.4: Defocus Blur Calibration

blurred as shown in the bottom half of the Figure 3. According to Figure 4.4, the images of circle edges on the focused images have a steeper slope (A Gaussian with a lower std). The slight blurring in these images are solely due to pixel binning. The edges on the blurred images have a more gradual slope. Also the slope becomes even more gradual as k_{cam} is increased.

3. Estimate the std of the Gaussian function of the circle edges from the focused image. We horizontally slice the image of the circle as seen in Figure 4.4 and obtain the distribution of pixel intensities. These are flat-top Gaussian functions. The flat top nature is due to the intensity being constant inside the circle. It falls gradually at the edges of the circles. We scale these intensity values into the range from zero to one. We consider all the values less than a threshold (we used 0.95) as belonging to the falling edges. We then integrate the resulting distribution (one dimensional Gaussian). According to equation 4.9, we can estimate γ after obtaining the integral J for a constant y.

$$J = \int_{-\infty}^{\infty} G(x)dx = \int_{-\infty}^{\infty} e^{-\frac{1}{2}\frac{x^2+y^2}{\gamma^2}} dx = \gamma\sqrt{2\pi} \tag{4.9}$$

4. Estimate the std of the Gaussian of the falling edges (λ) of the circles from defocus blurred images similarly to step 3. Note the the blurring of the defocused images are due to both defocus blurring and pixel binning. We can estimate the std of the Gaussian of the falling edges due to defocus blurring with equation 4.6.

5. Estimate the distance to each circle center from camera using the defocus blurred images using the camera intrinsic matrix generated with calibration in step1. This is a well-established procedure that is available in most of the computer vision libraries. We can write a separate version of equation 4.7 for each circle in the calibration pattern. With λ and γ already estimated, we can estimate the k_{cam} for the given camera using equation 4.7. Here we have assumed that the distance from the camera to each circle center is approximately equal to the distance to the edges of the circle. This can be justified because the distance to the circles from the camera (around 1m) is much larger than the diameter of the circles (around 4cm in our case).

6. To improve the accuracy of the estimate, we can repeat steps from 2 to 5 several times. See the evaluation section for further details on the experiments.

We can estimate the k_{cam} for a given camera with the above steps. The estimated k_{cam} can be used as shown in Figure 4.3 to predict depth with the images taken from this new camera.

4.3.3.1 Convolution of a 2D Gaussian Function with another 2D Gaussian Function

we have shown that the defocus-blurring can be modelled with a convolution of a 2D Gaussian function; the Point Spread Function (PSF) having a standard deviation of σ with the respective image in perfect focus (in-focus image). This Gaussian PSF can be denoted as below.

$$G(x,y) = e^{-\frac{1}{2}\frac{x^2+y^2}{\sigma^2}} \tag{4.10}$$

The blurred image B can be obtained as follows by convolving the in-focus image F with the blur function (PSF) G.

$$B(x,y) = G(x,y) * F(x,y) \tag{4.11}$$

Additional blurring of the image occurs due to phenomenon such as pixel binning and post processing which can be modelled as an additional convolution with a Gaussian function with a standard deviation of γ which we can be represented as follows.

$$Q(x,y) = e^{-\frac{1}{2}\frac{x^2+y^2}{\gamma^2}} \tag{4.12}$$

Let's consider the convolution of the already defocus-blurred image B with the additional blurring function due to pixel binning and post processing Q. The final image we can observe will be denoted by I

$$I = Q(x,y) * B(x,y) \tag{4.13}$$

$$I = Q(x,y) * (G(x,y) * F(x,y)) \tag{4.14}$$

Due to the associative nature of convolution

$$I = (Q(x,y) * G(x,y)) * F(x,y) \tag{4.15}$$

Let's explore the quantity $Q(x,y) * G(x,y)$
$Q(x,y) * G(x,y)$
$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} Q(k,m) \cdot G(x-k, y-m) dkdm$
$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-\frac{1}{2}\frac{k^2+m^2}{\gamma^2}} \cdot e^{-\frac{1}{2}\frac{(x-k)^2+(y-m)^2}{\sigma^2}} dkdm$
$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-\frac{1}{2}(\frac{k^2+m^2}{\gamma^2}+\frac{(x-k)^2+(y-m)^2}{\sigma^2})} dkdm$
$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-\frac{1}{2}(\frac{\sigma^2(k^2+m^2)+\gamma^2[(x-k)^2+(y-m)^2]}{\sigma^2\gamma^2})} dkdm$
$= e^{-\frac{1}{2}\frac{\gamma^2(x^2+y^2)}{\sigma^2\gamma^2}} \cdot$
$\int_{-\infty}^{\infty} e^{-\frac{1}{2}\frac{(\sigma^2+\gamma^2)k^2-2\gamma^2xk}{\sigma^2\gamma^2}} dk \int_{-\infty}^{\infty} e^{-\frac{1}{2}\frac{(\sigma^2+\gamma^2)m^2-2\gamma^2ym}{\sigma^2\gamma^2}} dm$
$= e^{-\frac{1}{2}\frac{\gamma^2(x^2+y^2)}{\sigma^2\gamma^2}} \cdot$
$\int_{-\infty}^{\infty} e^{-\frac{(\sigma^2+\gamma^2)}{2\sigma^2\gamma^2}(k-\frac{\gamma^2x}{\sigma^2+\gamma^2})^2+\frac{x^2\gamma^2}{2\sigma^2(\sigma^2+\gamma^2)}} dk \cdot$
$\int_{-\infty}^{\infty} e^{-\frac{(\sigma^2+\gamma^2)}{2\sigma^2\gamma^2}(m-\frac{\gamma^2y}{\sigma^2+\gamma^2})^2+\frac{y^2\gamma^2}{2\sigma^2(\sigma^2+\gamma^2)}} dm$

The integral of a Gaussian function,
$\int_{-\infty}^{\infty} e^{-a(x+b)^2} dx = \sqrt{\frac{\pi}{a}}$
Therefore,
$\int_{-\infty}^{\infty} e^{-\frac{(\sigma^2+\gamma^2)}{2\sigma^2\gamma^2}(k-\frac{\gamma^2x}{\sigma^2+\gamma^2})^2+\frac{x^2\gamma^2}{2\sigma^2(\sigma^2+\gamma^2)}} dk = \sqrt{\frac{2\sigma^2\gamma^2\pi}{\sigma^2+\gamma^2}}$
and
$\int_{-\infty}^{\infty} e^{-\frac{(\sigma^2+\gamma^2)}{2\sigma^2\gamma^2}(m-\frac{\gamma^2y}{\sigma^2+\gamma^2})^2+\frac{y^2\gamma^2}{2\sigma^2(\sigma^2+\gamma^2)}} dk = \sqrt{\frac{2\sigma^2\gamma^2\pi}{\sigma^2+\gamma^2}}$
The convolution becomes,

$$Q(x,y) * G(x,y) = 2\pi\frac{\sigma^2\gamma^2}{\sigma^2+\gamma^2} \cdot e^{-\frac{1}{2}\frac{(x^2+y^2)}{\sigma^2+\gamma^2}} \tag{4.16}$$

Therefore the convolution of a 2D Gaussian function with another 2D Gaussian function is also a Gaussian function. Let λ be the standard deviation of the resultant Gaussian function.

$$\lambda^2 = \sigma^2 + \gamma^2 \tag{4.17}$$

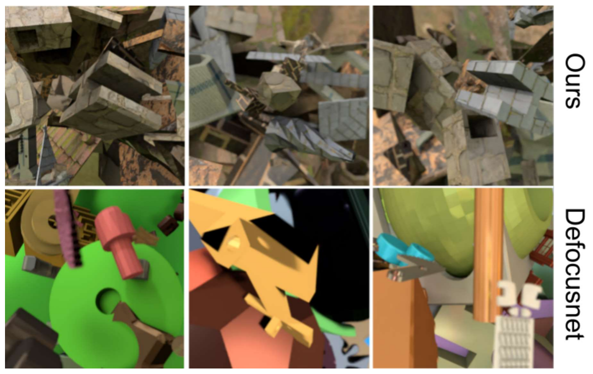


Figure 4.5: Synthetic dataset samples.

Since we can observe the image I, we can measure its blur (λ). The depth of each pixel on the image I is related the defocus blur (which is described with σ). We can find σ given λ (which we can measure from I) and γ (which we can measure with the calibration process described in the main paper) according to the equation below.

$$\sigma = \sqrt{\lambda^2 - \gamma^2} \tag{4.18}$$

4.4 Experiments

4.4.1 Datasets

Defocusnet dataset. We use the synthetic dataset generated by Maximov et al. [38] to train one of our models. This dataset was created with a virtual camera having several K_{cam} values of 0.15,0.33,0.78,1.59 and 2.41. This dataset has 500 focal stacks, each with 5 images with different focal distances.
Synthetic Blender dataset. We create a new synthetic dataset by expanding the defocusnet dataset [38]. This new dataset has various textures (to make them realistic) mapped to the 3D objects that was not present in the original dataset. We use several simulated cameras with K_{cam} of 0.08, 0.15, 0.23 and 0.33. This dataset has a focal distance of 1.5m and contains 400 defocus blurred images. We use the script provided by Maximov et al. (modified) to create our dataset. The images we generate in this dataset are 256 x 256 pixels.

We create this dataset to show that the models have the ability to generalize its learned knowledge to a new dataset and also it can adapt to different camreas. The original dataset did not have the textures mapped to the 3D objects. We found that this is because the 3D models used are of the STL format (commonly used for 3D printing) which only stores the geometric shape of an object and does not support texture or color information. We convert some of the downloaded STL models into OBJ format after UV mapping to facilitate textures. Different from the defocusnet dataset [38] we create focal stacks taken from several virtual cameras instead of just one camera. Some examples of the defocusnet dataset and our dataset is shown in Figure 4.5. We vary the virtual cameras in two respects; f-stop and focal length. The f-stops of the simulated cameras were chosen to be 1.0,1.1,1.2,1.5,1.8,2.0,2.2,2.8,3.0,5.0,8.0 and 10.0 to create the $blender_{testN}$ dataset. The focal lengths of the camera were taken to be 3mm, 4mm, 5mm, and 6mm to create the $blender_{testF}$ dataset while keeping the F-number 2. We train our models with the data from the decocusnet dataset and test on the images coming from $blender_{testN}$ and $blender_{testF}$ datasets. $blender_{testN}$ dataset contains 1491 focal stacks. $blender_{testF}$ contains 400 focal stacks. In addition to this we also create another dataset ($blender_{train}$) with a virtual camera with a focal length of $2.9mm$ and F-number of 1. This dataset has 1000 focal stacks. Each focal stack in all the datasets we created contains 6 blurred images and additionally an all-in-focus (AIF) image. The blurred images are focused at distances of 0.1, 0.15, 0.3, 0.7, 1.5 meters and at infinity. We do not use the images focused at infinity in evaluations of this paper. We use the same 20 3D models, 10 textures and a single environment map to create all the datasets. We use the script provided by Maximov et al. (modified) to create our dataset. Note that in the paper we have only used the $blender_{testF}$ dataset.

Both the Synthetic blender and the defocusnet datasets also have a perfectly focused image per each defocus blurred image.

Method	K_{cam}			
	0.08	0.14	0.23	0.33
in-focus	0.099	0.081	0.082	0.100
No K_{cam}	0.062	0.050	0.056	0.085
GT K_{cam}	0.045	0.037	0.052	0.061

Table 4.2: Performance on Blender dataset (MSE)

DDFF12 dataset. We also use the DDFF12 dataset provided by Hazirbas et al. [56] which contains 720 images created with a lightfield camera. We use the two real world datasets (DDFF12 and the NYU dataset described next) so that we can show that our models can work under real world images and deal with the domain gap between real and synthetic images [53]. After pre-processing the images as mentioned by Hazirbas et al. we obtained the blurred images which are focused at various distances.
NYU depth v2 dataset. The NYU depth v2 dataset [202] contains 1449 pairs of aligned RGB and depth image pairs. Following previous papers [38] [203] we create the training and testing splits. We create artificially defocus blurred images from this dataset using the method described by Carvalho et al. [31]. We have fixed certain drawbacks in their Matlab script in order to produce more realistic defocus blurred images as further discussed later in the chapter. We used K_{cam} values for training and testing as shown in Table 4.3. Images of 480 x 480 pixels were used for training and 480 x 640 were used for testing.

4.4.2 Experimental Setup

We use PyTorch [204] to implement the neural networks. We use the Adam optimizer [205] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ and a learning rate of 10^{-4}. Mean Squared Error was used as the loss function for both the blur and the depth to train all the models. We evaluate our depth predictions with the metrics absolute relative error (REL), mean-squared error (MSE), Root-Mean-Squared error (RMSE) and average log10 error. We also report threshold accuracy δ_n which is the percentage of pixels which satisfy the condition $max(d_i/\hat{d_i}, \hat{d_i}/d_i) < 1.25^n$. We train our models on the defocusnet dataset for 400 epochs and a batch size of 20. Our NUY depth models were train for 800 epochs with a batch size of 8.

4.4.3 Performance

Table 4.2 shows the performance of the model trained on the defocusnet [38] dataset and evaluated on our Blender dataset with different k_{cam} values of simulated cameras. All three methods; in-focus, No K_{cam} and GT K_{cam}, use the same deep learning architecture to predict depth. The only exception is that the GT K_{cam} model performs the K_{cam} correction as shown in Figure 4.3. We use the K_{cam} values that were used to generate the data and these can be called the Ground Truth K_{cam} values (GT K_{cam}). The In-focus model was both trained and tested on perfectly focused images. The No K_{cam} model does not consider the effect of K_{cam} during either training or testing (similar to defocusnet [38] model). This means the No K_{cam} model does not divide the output of the blur estimation model with K_{cam} as shown in Figure 4.3. The GT K_{cam} model on the other hand considers the effect of K_{cam} and behaves as shown in Figure 4.3 during both training and testing. According to Table ref4.2 the performance of both the No K_{cam} and the GT K_{cam} models are better (by around 0.025) than that of the in-focus method which shows that considering defocus blur is valuable when estimating depth. Our models perform better when considering the effect of K_{cam} (GT K_{cam}) compared to when not considering it (No K_{cam}) by around 0.015 in MSE. This shows that we can transfer the knowledge learned with the trained model into a new domain (images taken with a different K_{cam}) just with one parameter K_{cam}.

Table 4.3 shows the performance on the defocus blurred NYU depth dataset. Here we use a single trained model to evaluate the performance under various settings. The model was trained on data refocused with a K_{cam} of 8.79 and 35.61 and tested on the rest under the distance range of 0 to 2 m. The VPD model was trained and tested on in-focus images with no defocus blurring. Our GT and est K_{cam} methods outperform the state-of-the-art depth estimation model (VPD) on the NUY depth v2 dataset [52] by around 0.04 in RMSE. This converts to a reduction of error of around 4cm in the depth estimation. This proves again the importance of defocus blurring in depth estimation. We evaluate our models under three methods which depend on the nature of the K_{cam} values used. The method column, "GT K_{cam}" means we have used the K_{cam} values that were used to defocus blur the particular dataset which can be considered as Ground Truth K_{cam} values. "est K_{cam}" represents the K_{cam} values that were estimated with the defocus

K_{cam}	method	$\delta_1 \uparrow$	$\delta_2 \uparrow$	$\delta_3 \uparrow$	REL $\downarrow$	RMSE $\downarrow$	log10 $\downarrow$
in-focus	VPD[52]	0.953	0.992	0.999	0.052	0.154	0.027
8.79	GT K_{cam}	0.976	0.997	0.999	0.046	0.082	0.019
8.79	No K_{cam}	0.912	0.975	0.998	0.095	0.161	0.037
35.61	GT K_{cam}	0.976	0.997	0.999	0.046	0.082	0.019
35.61	No K_{cam}	0.962	0.995	0.999	0.054	0.101	0.023
12.69	GT K_{cam}	0.969	0.999	0.999	0.068	0.123	0.088
12.69	est K_{cam}	0.970	0.999	0.999	0.068	0.122	0.030
12.69	No K_{cam}	0.853	0.963	0.999	0.127	0.193	0.050
22.67	GT K_{cam}	0.980	0.998	0.999	0.068	0.117	0.028
22.67	est K_{cam}	0.980	0.998	0.999	0.069	0.118	0.028
22.67	No K_{cam}	0.896	0.994	0.999	0.105	0.165	0.043

Table 4.3: Performance on NYU dataset

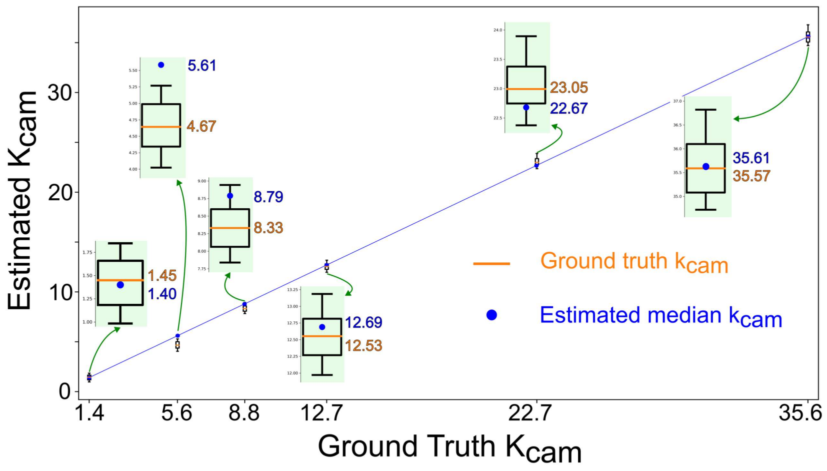


Figure 4.6: Estimation of K_{cam} values of different cameras

calibration method described in section 3.3. No K_{cam} models do not consider the effect of K_{cam}. We describe further details of the estimation process in the subsequent sections.

Table 4.3 shows the performance of our model under the DDFF12 dataset [56]. All the models were first trained on the defocusnet dataset [38] (the same model we used to evaluate the Blender dataset as shown in Table 4.3). The In-focus model was trained and tested on well focused images. The No K_{cam} and est K_{cam} models were trained and tested on defocus blurred images. Since we do not have ground truth K_{cam} for the DDFF12 dataset, we performed a linear search of the K_{cam} which predicts the best depth using the ground truth depth maps provided in the training set. The results in Table 4.4 are the performance of our model under the test set using the K_{cam} value found above. Both the No K_{cam} and est K_{cam} models perform better than the in-focus model for depth prediction. Also using the appropriate K_{cam} to transfer the model to the new domain of images significantly improves the performance compared to the no K_{cam} model which does not perform a correction that depends on the camera.

4.4.4 Defocus Blur Calibration Performance

We expand the discussion on defocus blur calibration in this section. These experiments were performed on the refocused NYU depth v2 dataset. We have created refocused data with K_{cam} values of 1.39, 5.61, 8.79, 12.69, 22.67, 25.61. Note that we have used the additional K_{cam} values (1.39 and 5.61) that were not used to evaluate the performance of depth estimation in Table 4.3. We obtain several photos of the asymmetric circular pattern shown in Figure 4.4 with the Microsoft Kinect camera and refocus them with the above mentioned K_{cam} values. We used from 19 to 20 different image pairs (an in-focus image and a defocus-blurred image) for each K_{cam} value. Then we perform the defocus blur calibration procedure described in section 4.3.3. Estimated K_{cam} values vs. the actual values (ground truth K_{cam}) are shown in Figure 4.6. The relationship between the ground truth and estimated K_{cam} values are very linear as expected. We estimate one K_{cam} value per one circle from an in-focus and defocus blurred image pair. Since there are 44 circles in the pattern, for 20 image pairs we obtained 880 estimated K_{cam} values. The

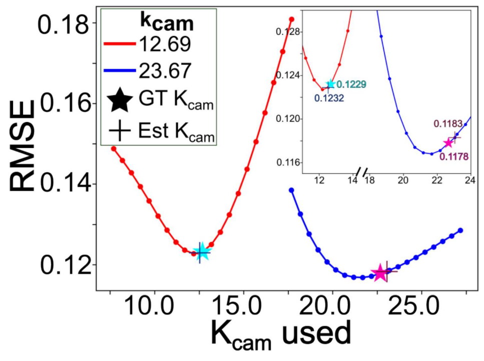


Figure 4.7: K_{cam} Estimation Error

Method	MSE
in-focus	0.0640
No K_{cam}	0.0139
est K_{cam}	0.0096

Table 4.4: Performance on DDFF12 dataset

K_{cam}	Image size	RMSE
12.69	Original size	0.123
	Resized	0.438
22.67	Original size	0.117
	Resized	0.399

Table 4.5: Effect of FOV

results in Figure 4.4 were obtained after removing outliers and calculating the median from the estimated K_{cam} values. We show box plots with interquartile range, median, minimum and maximum values of these estimations along with the ground truth K_{cam} values.

4.4.4.1 Sensitivity of depth estimation performance to K_{cam}

In this section we explore how the variation in estimated K_{cam} values affect the depth estimation performance. We use the same model that we used to obtain the results in Table 4.3 that was trained on data from K_{cam} values of 8.79 and 35.61 and evaluate them on data from K_{cam} values of 12.69 and 22.67. As can be seen in Figure 4.7, we use a range of numbers centered on the actual K_{cam} values for the respective datasets and obtain the RMSE error of depth estimation. It can be seen that the error response of the model to the variation of K_{cam} used has a clear minimum. The error increases if the values used in the place of K_{cam} deviates from the actual value. For example, the error of the response of K_{cam} = 23.67 increases by around 16% if the K_{cam} used deviates positively from the GT values by 18%. Figure 4.10 shows some examples of predicted depth maps when the model was provided with an unseen virtually blurred image from a camera with $K_{cam} = 22.67$. Agreeing with the Figure 4.7, the predictions get distorted faster when the K_{cam} used lowers than the ground truth K_{cam} and distorts slower when it increases.

4.4.5 Effect of the blur weight

We change the scaling parameter b_weight from equation 4.8 while training several models on data from defocus blurred NYU depth v2 dataset with K_{cam} values of 8.79 and 35.61. The performance on the two evaluation datasets (with K_{cam} values of 12.69 and 22.67) are shown in Figure 4.9.

4.4.6 Effect of the Field of View

Field of view of a camera can be defined in several ways. One way is to define it as the size of an object at a given distance from the camera that would completely fill the image sensor. In Figure 4.9, s is the length of the sensor, f is the focal length of the lens, d is the distance to the object and w is the length of the object. The size w of an object that would completely fill the sensor will be given by $w = \frac{s}{f} \cdot d$. It can be seen that w is inversely proportional to f. Cameras with a smaller focal length have a larger Field of View and vise-versa. In all the experiments that we performed including the NYU dataset, we have assumed that the cameras have a fixed FOV even when the k_{cam} (and therefore f) changes. While this is helpful to analyze the performance of blur based depth estimation methods, it is important to investigate the effect

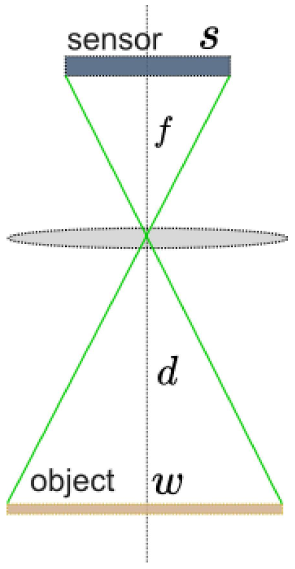


Figure 4.8: FOV of a camera

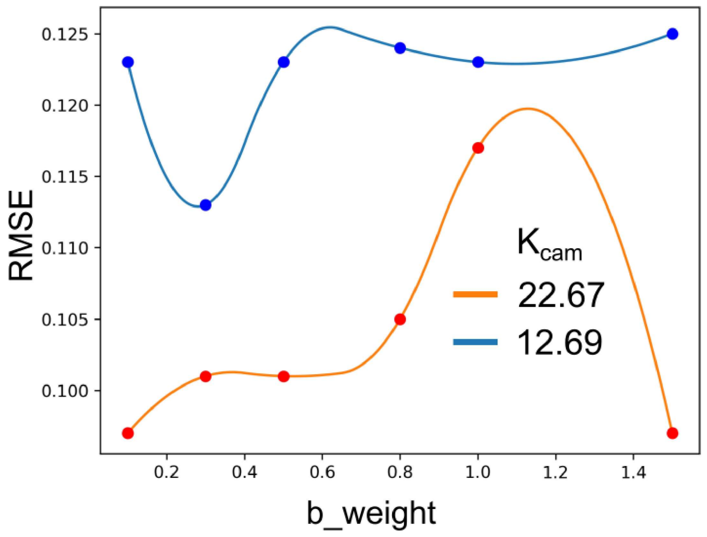


Figure 4.9: RMSE Vs b_weight

Model	RMSE (mm)
our defocus blur model	22.43
Depth Anything [206]	161.27

Table 4.6: Performance of hand depth estimation model

of the FOV change on the performance of the models. We created a dataset by scaling down the images in the NYU depth dataset by a factor of 0.6 and then refocusing with a k_{cam} (respective f is 30mm) of 12.69. The model has been trained with data having k_{cam}s of 8.79 and 35.61 (they had focal lengths (f) of 20mm and 50mm). We scaled down the images of $f = 30mm$ with respect to $f = 50mm$ which is 0.6. Note that the images have the same amount of blur as the images of original size; only the size of the objects visible have changed. From Table 4.5, it can be seen that the performance drops significantly by more than three folds when we perform the resizing. The reason for this can be understood from Figure 4.3. The depth estimation section receives two inputs. One is the blur and the second is the image features in the form of skip connections. Although we account for the change of blur through division by the respective k_{cam}, we do not modify image features to reflect the change of FOV. This is a limitation of our work and needs to be addressed in the future.

4.5 Hand Depth Prediction from Defocus blur

This section applies the depth from defocus blur techniques to estimate depth to hands. The dataset collected in section 3.4.2 was used to train a depth prediction model used to predict depth from defocus blur. The same model and training setting used in this chapter was used to train the model. The images from the Canon camera was used as inputs for this model since these images provided more significant defocus blur than the images from the Kinect camera. The depth to hands were also predicted with the state-of-the-art model Depth Anything [206] and the results are shown in Table 4.6. Note that the Depth Anything model was used as is and not fine-tuned on our hand depth estimation dataset. Our model which uses defocus blur achieves low error level of just around 22mm while the depth anything model shows an error of 161 mm.

Next we used the trained depth from defocus blur model to predict the depth to hands in order to predict the CPR compression depth. The results are shown in Table 4.7. Results which used the ground truth depth from the Kinect camera is also repeated for reference.

Participant	**Session**	Kinect		Blur	
		Rate error (comp/min)	Depth error (mm)	Rate error (comp/min)	Depth error (mm)
P0	0	1.3	0.29	0.86	11.63
	1	5.2	1.1	4.91	12.68
	2	34.4	4.66	34.43	15.85
	3	38.7	3.02	38.39	13.93
	4	7.6	2.03	7.52	12.85
	5	20.5	2.34	20.41	13.53
	6	5	0.43	4.66	11.48
	7	3	0.25	2.77	11.26
	8	46.5	0.38	46.40	11.66
	9	4.3	4.07	4.28	14.76
	10	67.9	42.24	68.04	48.86
P1	0	69.6	14.4	69.23	25.41
	1	36.5	13.2	36.40	24.22
	2	7.36	73.4	7.24	84.39
	3	20.45	121.9	20.24	132.99
	4	2.35	0.6	1.97	11.37
	5	8.92	5.8	8.77	16.50
	6	21.4	58.6	21.15	69.98
	7	24.5	12.8	24.31	24.43
P2	0	5.09	54.68	4.85	65.74
	1	24.65	4.93	24.19	16.14
	2	4.57	4.75	4.32	15.69
	3	6.7	4.71	6.43	15.74
	4	5.47	4.75	5.22	15.94
	5	5.57	3.87	5.63	14.55
	6	12.69	15.12	12.45	26.00
	7	3.19	22.36	3.09	33.35
	8	7.03	3.02	6.97	14.09
	9	2.71	5.45	2.71	16.27
	10	31.84	4.77	31.64	16.22
P3	0	14.2	105.93	14.14	109.72
	1	43.3	12.45	43.25	23.74
	2	1.5	22.9	1.04	33.74
	3	31.7	58.1	31.42	69.12
	4	42.4	37.9	42.36	49.46
	5	9.7	32.8	9.51	42.97
	6	4.7	5.1	4.52	16.50
	7	6.5	11.2	6.34	22.53
	8	30.7	65.1	30.42	70.07
P4	0	0.89	1.55	0.58	12.46
	1	18.72	27.63	18.58	38.59
	2	3.17	29.51	3.03	40.17
	3	4.4	46.25	4.39	57.29
	4	86.4	22.6	86.44	33.46
	5	14.44	55.9	13.86	67.12
	6	35.63	2.36	35.53	13.58
	7	42.36	1.02	42.10	12.28
P5	0	1.42	67.71	1.46	78.76
	1	7.39	81.95	7.30	92.57
	2	14.83	13.67	14.57	24.77
	3	0.42	40.01	0.03	50.94
	4	37.75	2.06	37.40	12.51
	5	4.67	47.72	4.38	58.37
	6	61.38	4.36	61.34	15.19

	7	7.7	78.41	7.48	89.54
	8	9.28	50.18	8.86	61.07
	9	1.74	35.4	1.50	46.52
	10	11.68	10.2	11.53	21.17
P6	0	10.51	2.83	10.11	13.74
	1	19.06	52.19	18.89	63.33
	2	46.97	32.76	46.76	43.91
	3	34.91	47.14	34.60	58.03
	4	0.54	11.5	0.31	22.25
	5	8.76	25.11	8.56	35.98
	6	77.76	54.41	77.69	65.87
	7	54.64	19.83	54.12	30.85
	8	97.47	5.2	97.46	16.27
P7	0	12.63	12.59	12.40	23.11
	1	8.51	2.12	8.40	12.94
	2	16.23	0.56	16.32	11.13
	3	1.41	2.32	1.20	13.46
	4	40.3	30.1	40.00	40.80
	5	0.29	0.57	0.16	11.70
	6	9.35	7.09	8.89	17.52
	7	8.99	12.2	8.63	23.52
	8	11.58	9.39	11.40	20.27
P8	0	1.56	1.17	1.47	12.07
	1	3.91	49.97	3.69	61.62
	2	34.7	1.36	34.36	12.11
	3	29.99	0.15	29.69	11.24
	4	3.06	0.67	2.81	11.85
	5	0.81	1.16	0.54	12.11
	6	2.68	0.15	2.67	11.53
	7	1.32	0.42	1.36	12.06
	8	9.88	2.97	9.52	13.64
	9	9.49	1.71	9.40	13.00
	10	11.56	1.74	11.39	12.15
	11	24.99	13.47	24.75	10.77
P9	0	116.35	13.34	116.11	24.58
	1	120.84	26.13	120.76	37.15
	2	41.65	0.11	41.40	11.44
	3	31.9	7.08	31.96	17.39
	4	25.96	9.13	26.08	20.27
	5	27.39	10.09	27.55	21.16
	6	65.85	16.57	65.60	27.18
	7	33.13	19.54	32.74	31.08
	8	33.13	39.6	33.03	48.72
P10	0	41.11	83.83	40.80	95.51
	1	14.11	35.26	14.02	46.34
	2	16.75	44.62	16.48	55.86
	3	31.4	33.91	31.18	45.03
	4	15.98	3.48	15.86	14.09
	5	23.72	1.32	23.58	12.29
P11	0	9.49	0.63	9.33	11.77
	1	4.82	0.77	4.74	11.71
	2	9.08	0.99	9.05	12.36
	3	8.73	0.41	8.46	11.59
	4	12.54	3.03	12.46	13.55
	5	9.02	3.95	8.80	14.56
	6	8.13	292.65	8.03	304.06
	7	42.15	5.17	41.99	16.20

	8	2.21	1.65	1.61	12.76
	9	4.65	8.84	4.54	19.69
	10	0.02	173.61	0.26	184.13
P12	0	8.32	8.23	8.25	19.73
	1	29.54	0.5	29.46	17.71
	2	18	0.86	17.66	12.32
	3	1.81	1.77	1.72	12.61
	4	10.33	0.64	10.31	12.35
	5	2.96	0.38	2.74	11.56
	6	2.77	4.99	2.73	15.44
	7	1.23	0.01	0.94	10.96
P13	0	20.78	23.58	20.63	34.25
	1	2.1	7.53	1.85	18.95
	2	1.9	4.62	1.69	15.77
	3	4.16	8.14	3.91	19.58
	4	0.46	0.59	0.21	11.77
	5	13.32	2.54	13.03	13.08
	6	0.84	2.21	0.69	13.58
	7	0.03	1.76	0.13	13.09
	8	11.77	1.98	11.65	11.37
	9	0.25	24.8	0.20	36.12
P14	0	6.35	18.28	6.18	29.28
	1	5.79	15.87	5.65	26.23
	2	3.95	14.3	3.33	24.95
	3	11.6	4.04	11.54	15.26
	4	11.05	7.23	10.70	18.70
	5	4.06	18.42	3.92	29.24
	6	31.74	2.79	31.56	13.56
	7	59.21	6.1	59.13	17.28
	8	52.32	15.42	52.26	26.45
P15	0	4.51	9.25	4.21	20.08
	1	10.95	12	10.81	25.92
	2	14.04	11.46	14.12	22.82
	3	3.71	13.24	3.47	23.90
	4	7.77	32.09	7.44	43.02
	5	13.76	12.31	13.45	22.81
	6	3.87	6.65	3.60	17.81
	7	21.55	35.13	21.25	46.50
	8	6.09	6.7	5.68	17.47
	9	0.15	33.33	0.20	44.08
	10	12.23	9.15	12.05	20.14
P16	0	2.32	3.44	1.99	15.17
	1	0.31	2.39	0.05	13.95
	2	13.52	0.61	13.38	11.46
	3	28.41	1.63	28.13	12.68
	4	22.69	0.81	22.34	12.53
	5	9.14	8.61	8.86	19.87
	6	61.14	22.87	61.17	33.88
	7	0.89	0.96	0.58	12.31
P17	0	27.24	6.86	27.09	18.11
	1	58.98	2.71	58.62	15.11
	2	0.19	1.52	0.04	12.43
	3	10.92	1.1	10.65	11.73
	4	0.57	5.27	0.51	15.69
	5	3.67	2.14	3.88	13.05
	6	1.33	5.31	1.02	16.52
	7	10.44	2.37	10.21	13.54

	8	0.17	3.85	0.16	14.73
	9	9.28	2.46	8.88	16.59
P18	0	20.79	2.34	20.40	13.48
	1	4.03	1.47	3.90	12.71
	2	11.07	7.47	10.84	21.56
	3	3.77	4.69	3.74	15.76
	4	1.43	8.33	1.35	19.24
	5	1.35	3.67	1.11	14.98
	6	0.19	3.35	0.04	13.99
	7	7.84	9.83	8.01	22.61
	8	10.41	3.26	10.14	14.26
	9	36.35	3.56	36.36	14.77
	10	2.3	2.18	2.29	13.35
	11	11.33	2.63	11.11	13.72
P19	0	6.74	1.51	6.50	12.55
	1	1.97	0.6	1.85	11.65
	2	6.16	5.6	6.06	17.17
	3	1.43	10.02	1.07	21.35
	4	1.76	12.19	1.52	22.95
	5	39.58	7.7	39.44	18.67
	6	15.26	12.22	14.85	23.81
	7	2.39	6.26	2.20	17.55
P20	0	11.7	1.64	11.64	13.48
	1	5.57	10.58	5.50	21.96
	2	5.51	3.56	5.34	14.67
	3	7.22	4.89	7.05	17.98
	4	7.37	9.52	7.01	20.94
	5	3.26	15.11	2.99	22.80
	6	73.12	6.1	72.85	16.58
	7	2.77	6.85	2.50	17.73
	8	17.68	2.76	17.46	13.56
	9	21.77	5.14	21.58	15.59
	10	9.17	4.84	8.73	15.90
	11	1.96	9.41	1.88	20.18
	12	3.67	5.73	3.30	16.47
P21	0	6.94	8.89	6.76	19.98
	1	21.45	16.09	21.52	27.21
	2	16.08	12.09	15.89	23.63
	3	9.38	7.85	8.97	20.00
	4	8.13	6.89	7.82	17.42
	5	5.49	8.89	5.30	19.61
	6	2.42	9.84	2.14	20.81
	7	28.66	3.99	28.35	15.41
	8	7.59	15.04	7.22	26.43
	9	2.56	4.17	2.43	15.52
	10	0.49	13.46	0.34	24.84
P22	0	8.17	1.2	8.01	12.27
	1	13.92	0.73	13.65	11.65
	2	0.8	3.14	0.58	13.98
	3	2.33	0.65	2.04	11.59
	4	13.02	1.71	12.96	12.61
	5	26.02	2.84	25.77	13.98
	6	4.05	0.25	3.71	11.04
	7	0.83	1.73	0.59	13.19
	8	0.09	1.19	0.08	12.04
	9	4.29	1.81	4.23	14.41
	10	4.32	0.33	3.99	11.55

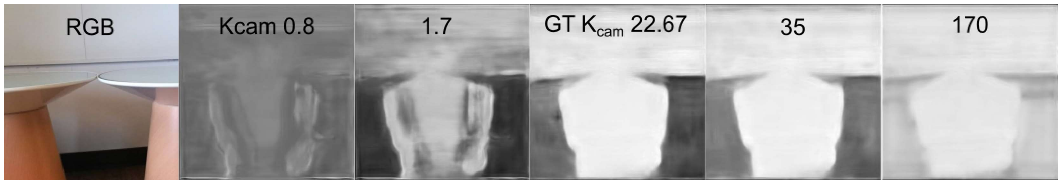


Figure 4.10: Examples of from the camera with $K_{cam} = 22.67$ predicted using various K_{cam} values

	11	1.32	0.65	1.25	11.90
	12	7.86	1.75	7.60	12.49
P23	0	3.31	2.32	3.10	13.53
	1	6.52	2.73	6.23	14.49
	2	46.38	1.07	46.24	11.74
	3	16.31	2.65	16.08	14.15
	4	14.95	4.14	14.66	14.56
	5	6.39	0.39	5.93	11.93
	6	14.03	1.45	13.66	12.54
	7	18.51	1.76	18.30	12.96
	8	4.18	0.49	3.85	11.18
	9	37.88	2.55	37.69	13.59
	10	19.97	0.04	20.04	10.66
	11	39.24	2.31	39.00	12.88
Average		16.26	14.62	16.06	25.60

Table 4.7: CPR compression rate and depth performance

4.6 Conclusions

We show that estimating depth from defocus blur is significantly superior to conventional semantic based depth prediction provided that the camera is suitable for it. But this technique is sensitive to the camera. Our novel approach performs a simple correction to an already trained depth prediction model using the camera parameters of a given camera. We show that this correction can alleviate the sensitivity of the model to the camera. Our novel defocus blur calibration technique can estimate the camera parameters using several images taken by a given camera. We show that our approach beats the state-of-the-art for several datasets. We show some limitations of our work and suggest future improvements. We trained a model that uses defocus blur that can be used to measure hand depth with the dataset we collected in Chapter 3. This model was used to predict CPR depth and we present the results. The results still need improvements over the smartwatch-based results from Chapter 3. We may improve the performance in the future after considering the suggestions mentioned in Chapter 3.

CHAPTER 5

GENERALIZED FEW-SHOT LEARNING FOR WEARABLE SENSOR-BASED HUMAN ACTIVITY RECOGNITION

5.1 Introduction

Human activity recognition (HAR) using wearable devices has many applications including in fields such as healthcare, security, entertainment, first response, and human-computer interaction [68][69]. Deep learning based methods are being applied successfully to solve this problem. However, these models need a large amount of data to achieve a high level of performance. Data for common activities such as walking and running are easy to obtain from public datasets, while data for activities outside of those areas are difficult to obtain. For example, publicly available data sets might not contain data for horse riding or gymnastics because they are less common than walking. Also, there may be certain personalized activities that are unique to a certain individual. For example, the way certain individuals cook may be different from others. Therefore data for these activities are usually unavailable during the training time. Unlike in the domain of computer vision, HAR lacks data due to the difficulties in annotating data [70]. Consequently, HAR models which can detect activities with very few training samples are needed.

In Few-Shot Learning (FSL) setting, it is important to make the distinction between source and target domain. Source domain is a set of data samples from classes where data is abundant (e.g. walking and running). Target domain is data samples where only very limited amount of data is available from each class (e.g. horse riding and gymnastics). In the problems considered in this paper, target and source domain classes are completely exclusive from one another. FSL utilizes knowledge from source domain and generalize this knowledge to classify the samples from the target domain [207] [80]. To so this very few samples from the each class in the target domain is used. For example, consider an HAR system that is used to detect activities performed by an elderly person living alone. This system may be trained with common activities such as walking, sleeping, sitting and climbing stairs. Now, the elderly person would like their device to recognize other activities such as cooking, playing an instrument, doing exercises, and walking with a mobility aid. It is impossible for the manufacturer to account for all of these activities because they may be personalized to each individual. For example, each elderly person may be doing very unique exercises due to injury or pain and the mechanics of walking with aid depends heavily on the device used. The FSL system should be able to be trained with just a few examples of these new activities while being economical because the FSL system will be running on the end user's resource constrained mobile devices.

A practical FSL solution should classify samples from the target domain as well as the source domain. This is important because while the user is interested in classifying the samples from the target domain, it should not stop classifying samples from the source domain. This task is called generalized Few-Shot Learning (GFSL) and most FSL papers do not address it [83]. Building a classifier for source domain is straight-forward and in this thesis we present the methods to train a FSL classifier to classify samples from the target domain and another classifier to distinguish if a sample is coming from source or target domain, which would make this a GFSL solution. Although FSL has been applied in drug discovery, character generation, robotics, image classification, gesture recognition and neural architecture search [81], application of FSL in HAR is very limited. Domains such as image classification has the luxury of enormous datasets. In comparison, datasets in HAR are minuscule. Therefore, FSL methods developed in the image classification domain cannot be applied directly to HAR.

We developed a generalized FSL solution for HAR using wearable devices. Influenced by previous work on FSL [80] we used a deep learning based embedding function to project input data from wearable inertial sensors into an embedding space. Previous work has shown that encouraging an embedding model to generate features which are tightly clustered around their respective class centroids improves their FSL performance [80] [208].

We use several methods to reduce the intra-class distances and increase inter-class distances of the embeddings. We use center loss which is used successfully in facial recognition systems [208] as our loss function. We also modified prototypical networks, which is a popular FSL method [80] from the literature. We generate embedding with these two methods and used them to train CLS_{FSL}. Finally we show that test-time adaptation methods can be used after the mode is trained to improve the accuracy of the FSL performance.

This work is the first to demonstrate how to build a generalized FSL solution for HAR. We ran experiments on three publicly available datasets. Fine tuning was done on the UTWENTE dataset[209] and evaluations were performed on the OPP[210] and PAMAP2[211] datasets. The methods on average outperformed state-of-the-art FSL models by around 11% and 6% on the PAMAP2 and OPP datasets, respectively, under FSL conditions. Finally we evaluated the model dataset that combines several activities from PAMAP2 dataset and CPR activity which is important in the context of Emergency Medical Service (EMS) providers.

5.2 Related Work

Convolutional Neural Networks (CNNs) are proved to be more robust to the changes in underlying data distributions compared to Recurrent Neural Networks [74]. Fully Convolutional Neural Networks (FCNs) have been shown to demand less computational and memory cost while performing better than other techniques when significant class imbalance is present. Also, FCNs can accommodate variable input lengths which makes it more suitable for processing various activities with different duration. FCNs have been used successfully for HAR with IMU data with above advantages evident [75][76].

Lack of data for certain classes in HAR [77] can cause probles such as performance degradation, overfitting and reduced robustness [78]. Lack of data could arise due to the fact that there is more data for common activities such as walking and running, but a limited amount of data for uncommon activities such as grabbing a box [72] or certain classes being simply unavailable during the development phase as described in section 1. The significant time and labor costs related to collecting data only exacerbates the problem [79]. In these cases, the knowledge from the classes with many data samples can be utilized to learn general knowledge that can be used to classify rarely seen classes [79].

Unsupervised representation learning aims to generate clustering-friendly embedding from input data without using any labelled data. A common approach is to use an encoder-decoder architecture to indirectly learn a low-dimensional latent representation of the input data [212] [213]. After generating clusters they may be mapped to an existing activity class using a small amount of labelled data [212] [213]. Although unsupervised representation learning has similarities to our problem, we need a different solution to address the situation where we have sufficient data from several classes and extremely limited amount of data from some other classes.

Few-shot learning (FSL) is a machine learning technique that specializes in learning from a few examples. It's aim is to learn the ability to make inferences on new classes not seen during training [80] [81]. FSL has shown its use in several domains such as drug discovery, character generation, robotics, image classification, gesture recognition and neural architecture search [81]. One popular example of FSL is prototypical networks [80] where they learn a non-linear mapping from input space to an embedding space. In essence, prototypical networks aim to cluster the embeddings of the same classes together while making the distance between clusters of different classes further from each other. Similar to prototypical networks [80], meta-learning [82] learns an embedding function. These embeddings are then classified with a SVM. Meta-learning optimizes the embedding function using the loss obtained from the SVM classifier [82]. Most of the FSL literature is only concerned about classifying target classes but for a practical application, source class classification may also be needed. Generalized FSL addresses this problem by devising methods to classify both source and target samples [83] [84][**Socher2013Zero-shotTransfer**].

The research done in FSL in the HAR domain is limited. In a FSL based HAR paper [79] a long short-term memory (LSTM) model is used to extract features and classify activities. It was trained with data from source domain and the network parameters obtained were transferred to the model that classified samples in the target domain. For each sample in the target domain parameters were only transferred from similar classes to avoid negative transfer.

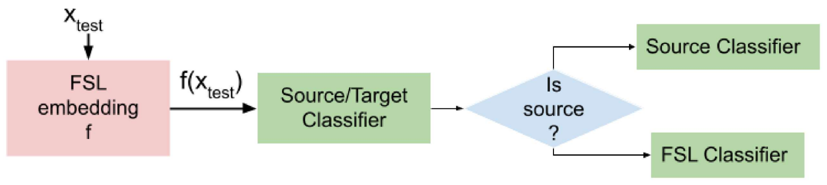


Figure 5.1: Operation of the FSL system

From the literature it can be seen that generating embedding which have low intra-class variation and high inter-class variation improved the FSL performance. We use center loss along with softmax loss [208] to create such embeddings [208]. According to a taxonomy introduced by a survey on HAR [81], our solution fell under model based methods which performed task-invariant embedding learning. We also implemented prototypical loss [80] with some modifications for the task of FSL based HAR. The model FSHAR from early work on FSL based HAR [79] is used as a baseline for comparison.

5.3 System Operation

Fig. 1 shows the operation of our FSL system after it is trained. Data flow is shown in Figure 2. A FSL embedding function is denoted by f. CLS_{ST} distinguishes data from source and target domain. CLS_S classifies data in source domain while CLS_{FSL} classifies data from target domain. We denote source domain data as $S = \{(s_i, y_i)_{i=1}^{N_s}\}$. Where each $s_i \in \mathbb{R}^D$ is the D-dimensional feature vector of a sample. $y_i \in C_S = \{1, ..., |C_S|\}$ are the corresponding labels. We divide S into S_{train} and S_{test}. We train the embedding function f with data from S_{train} and perform testing on S_{test}. We embed S_{train} and S_{test} with f and get Σ_{train} and Σ_{test}. While training, the source classifier CLS_S is attached at the end of f. CLS_S is trained to classify the embeddings generated by f into one of the classes present in S. The error used in training CLS_S is calculated from the class probability generated by it. We learn an embedding function $f(., \theta_{src}) : \mathbb{R}^D \to \mathbb{R}^M$ using data from S_{train} which embeds each sample s_i to an embedding $\sigma_i \in \mathbb{R}^M$. Here θ_{scr} is the set of parameters of the embedding function f. This process outputs embedded source data $\Sigma = \{(\sigma_i, y_i)_{i=1}^{N_s}\}$. These embeddings and corresponding class labels are used to calculate a loss (explained in next section) which, in turn, is used to train f using back propagation. FSL (CLS_{FSL}) and source/target (CLS_{ST}) classifiers are trained with the data from the user. In Figure 1, x_{test} denotes a data sample to be classified. This would be a multi dimensional vector of IMU data. The embedding of the x_{test}, $f(x_{test})$ is generated with the trained embedding function f (CNN model). $f(x_{test})$ is sent to CLS_{ST}. This outputs whether the input sample is from a source class or a target class. If x_{test} comes from the source data, $f(x_{test})$ is send to CLS_S for classification. This would give the probabilities of x_{test} being belonging to each class in the source classes. If x_{test} belongs to target data, $f(x_{test})$ is sent to CLS_{FSL} which would classify it to one of the target classes. When the user encounters new classes, the system should be capable to incorporate these into the CLS_{ST} and CLS_{FSL} by retraining them. f and CLS_S require no modification after the initial factory training.

For the FSL tasks we have the data from target domain. $T = \{(t_i, z_i)_{i=1}^{N_T}\}$. Where $t_i \in \mathbb{R}^D$. $z_i \in C_T = \{1, ..., |C_T|\}$ are the corresponding labels. We can generate the set of embeddings from T by sending all $t_i \in T$ through the embedding function f. This way we obtain $\mathbb{T} = \{(\tau_i, z_i)_{i=1}^{N_T}\}$ where each $\tau_i \in \mathbb{R}^M$ and $\tau_i = f(t_i, \theta_{src})$. To train the FSL classifier we extract K number of samples from each class randomly from the data in $\mathbb{T}$. The number of classes in T is $|C_T|$. This is considered a $|C_T|$-way-K-shot classification. We use rest of the data in $\mathbb{T}$ for testing. To classify an unseen sample $(t_{test}, ?)$, we get the embedding of t_{test} as $f(t_{test}, \theta_{src}) = \tau_{test}$. Then $CLS_{FSL}(\tau_{test})$ classifies the test sample t_{test} to one of the classes in T. If the embedding function f embeds the data to a lower dimension, that is if $M < D$, a simpler classifier can be built to classify the embedding of t_{test} into a class. CLS_{FSL} is preferred to be a simpler classifier since the amount of data used to train CLS_{FSL} is very small.

We build a source/target classifier CLS_{ST} to distinguish samples from S and T. We select a K number of samples from each class in both Σ_{test} and $\mathbb{T}$. We train C_{ST} with these data. C_{ST} is tested with the rest of the data from Σ_{test} and $\mathbb{T}$.

5.3.1 Calculating Class Centers

Both prototypical networks and center loss based networks utilize class centers. Class centers for source data S can be calculated as

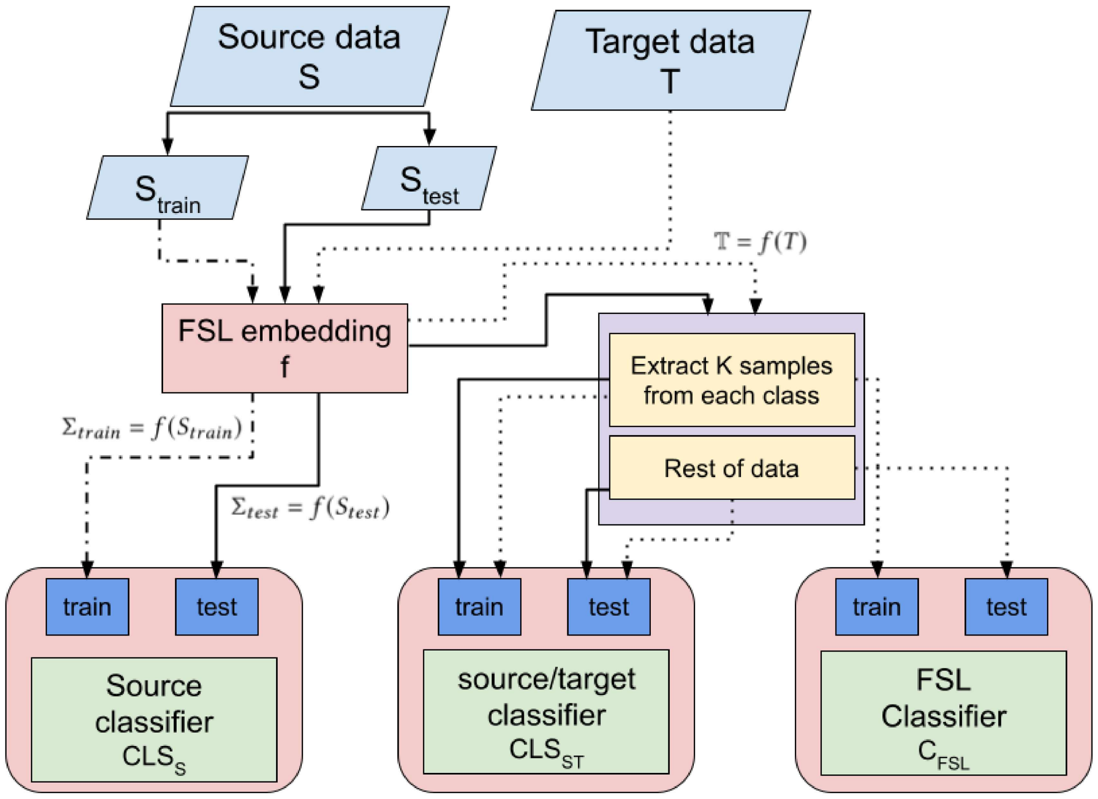

Source data S
Target data T
S_{train}
S_{test}
$\mathbb{T} = f(T)$
FSL embedding f
Extract K samples from each class
Rest of data
$\Sigma_{train} = f(S_{train})$
$\Sigma_{test} = f(S_{test})$
train
test
Source classifier CLS_S
train
test
source/target classifier CLS_{ST}
train
test
FSL Classifier C_{FSL}

Figure 5.2: Training workflow

$$CENTER_k = \frac{1}{|S_k|} \sum_{(s_i,k)\in S_{train}} f(s_i) \tag{5.1}$$

Here $CENTER_k$ is the center of class k. S_k denotes all samples belonging to class k. In prototypical networks these centers are called "prototypes" [80]. Center loss paper calls them "class centers" [208]. Ideally, the entire training set S_{train} should be taken into account when calculating the centers. But in practice, the centers are calculated only considering mini-batches.

5.3.2 Prototypical networks

Prototypical networks from [80] use a neural network based function to derive embeddings for input data. For each class in S, they calculate a "prototype" by taking the mean of the embedding of each class as shown in equation 1. Given a distance function d, prototypical networks generate a distribution over classes for a sample point x based on softmax over distances.

$$p_\Phi(y = k|x) = \frac{exp(-d(f(x), CENTER_k))}{\sum_{k'} exp(-d(f(x), CENTER'_k))} \tag{5.2}$$

If x belongs to class k, the distance between the class center $CENTER_k$ and x should be low and $p_\Phi(y = k|x)$ value for class k should be the maximum among all other $p_\Phi(y = k'|x)$ for all the other classes k'.

Learning aims to minimize the negative log-probability $J(\Phi) = -log[p_\Phi(y = k|x)]$. Since f is a neural network, the loss J can be used for training f by back propagation. Details of the training procedure follows from prototypical networks [80].

5.3.3 Center Loss

To create embeddings with low intra-class variation and high inter-class variation, we use the center loss as described in [208]. It is described as shown in the equation 3. The training data were taken from source domain S_{train}.

Algorithm 1 Training episode loss computation. C_S is the set of classes in source set. C_{cl} and C_{sl} are the sets of classes selected to calculate center loss and softmax loss. Here $|C_s| = |C_{cl}| + |C_{sl}|$. m is the number of samples used to train per class. S^k denotes the subset of S where all elements (s_i, y_i) are such that $y_i = k$. RANDOMSAMPLE(A,B) denotes a set of B elements chosen uniformly at random from set A without replacement. Σ^p_{train} denotes all data points $\in \Sigma_{train}$ which belongs to class p. Here $softmaxloss$ is the softmax loss calculated according to equation (4).

```
Input : source set $S = \{(s_1, y_1), ..., (s_N, y_N)\}$ where each $y_i \in C_s$
output: The loss J for a training episode.
  $P = \leftarrow RANDOMSAMPLE(C_s, |C_{cl}|)$
  for $p$ in $P$ do
    $A_p \leftarrow RANDOMSAMPLE(\Sigma^p_{train}, m)$
    $CENTER_p \leftarrow \frac{1}{m}\sum_{(s_i,y_i)\in A_p} f(s_i)$
    $CL \leftarrow CL + \frac{1}{2}\sum_{(s_i,y_i)\in A_p} ||f(s_i) - CENTER_p||^2_2$
  end for
  for $q$ in $\{\{C_S\}\backslash P\}$ do
    $B_q \leftarrow RANDOMSAMPLE(\Sigma^q_{train}, m)$
    $SM \leftarrow SM + \sum_{(s_i,y_i)\in B_p} softmaxloss(f(s_i), y_i)$
  end for
  $J = SM + \lambda CL$
```

$$L_C = \frac{1}{2}\sum_{i=1}^{m} ||f(x_i) - CENTER_i||^2 \tag{5.3}$$

Here m is the size of mini-batch. $CENTER_i$ is the center of class of x_i. The softmax loss for the mini-batch can be defined by equation 4.

$$L_S = -\sum_{i=1}^{m} log\frac{e^{W^T_{y_i}x_i + b_{y_i}}}{\sum_{j=1}^{n} e^{W^T_j x_i + b_j}} \tag{5.4}$$

The final loss of the model is the summation of the two losses with the hyperparameter λ to control L_C

$$L = L_S + \lambda L_C \tag{5.5}$$

Note that in equation 3, L_c is the scaled average of euclidean distance of data points from their class centroid. Research on FSL with prototypical networks [80] show that euclidean distance works better than other types of distances such as cosine similarity for FSL problems.

We modified the training procedure to generate more generalized embeddings. During each training episode, a subset of $|C_{cl}|$ number of classes are selected from C_S. Samples in the mini-batch belonging to C_{cl} are used to obtain the center loss L_c. The rest of the data in the mini-batch is used to obtain the softmax loss L_s. Pseudocode to compute loss J is shown in Algorithm 1. We use separate classes to calculate center loss and softmax loss so that f would be adapted to unseen classes.

5.3.4 Test time adaptation for Few shot learning

We expand our FSL framework by incorporating the test time adaptation philosophy. As mentioned in Section 2 Batch Normalization (BN) layers can be used to adapt a model to the test data distribution by adjusting the scaling and shifting parameters of the BN layers. BN layers operate as show in equation 5.6. x is the inputs to the BN layer. $E[x]$ and $Var[x]$ represents the mean and variance of the inputs to the BN layer. γ is the scaling parameter and β is the shifting parameter.

$$y = \frac{x - e[x]}{\sqrt{Var[x] + \epsilon}} * \gamma + \beta \tag{5.6}$$

We show that the reason the FSL performance is low is the difference of distributions between the source data and the target data. The scaling and the shift parameters of the BN layers can be used to adapt a model easily to a new data distribution. Since there are just two parameters (scale and shift) this adaptation can be done with very few data samples. Therefore this is very suitable for the FSL setting. Although BN based test time adaptation have been used to adapt a model to a new domain, it has not been tested for FSL setting. In order to adapt the model under FSL setting, we perform the regular back propagation on the model after freezing the whole model except for the BN layers thereby adjusting the scaling and shifting parameters of all the BN layers. We calculate a novel loss based on the model outputs as shown in equation 5.8. Here we use two different types of losses. L_C is the centerloss used previously. In addition to this we use another loss that we call Distribution Loss while measures the intra-sample distance of the prototypes as shown in Equation 5.7. This loss is utilized to increase the separation of the different classes. We also use regularization on the features to prevent the features from getting too large.

$$L_{dist} = \frac{1}{\frac{1}{N}\sum_{(i,j)\in Li\neq j}^{m}(f(x_i) - f(x_j))^2} \tag{5.7}$$

$$L = L_C + L_{dist} \tag{5.8}$$

We further fine-tune our model with the UTWENTE dataset under the Test-time Adaptation (TA) and obtained the model in Figure 5.3. We use this model in all the TA based experiments.

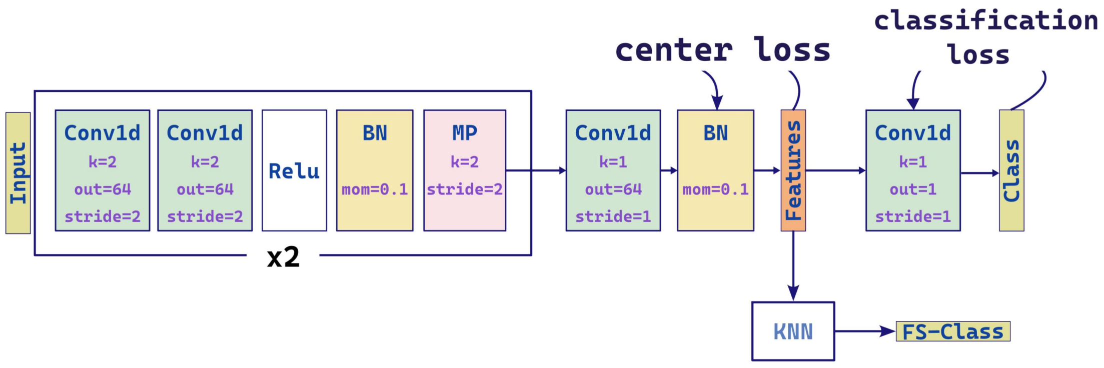

classification loss
center loss
Input
Conv1d k=2 out=64 stride=2
Conv1d k=2 out=64 stride=2
Relu
BN mom=0.1
MP k=2 stride=2
x2
Conv1d k=1 out=64 stride=1
BN mom=0.1
Features
Conv1d k=1 out=1 stride=1
Class
KNN
FS-Class

Figure 5.3: TA model

Test vs train statistics for the BN layers: When normalizing the outputs with, BN layers can either use the statistics $E[x]$ and $Var[x]$ calculated from the current batch or use the statistics calculated from the whole train set. In addition, a balance of these two quantities can be used by calculating running statistics as mentioned in Equation 5.9. Here $\hat{\mu}$ is the current mean (calculated from the train set) and μ_t is the statistic from the current batch (in the FSL setting this is from the target samples).

$$\hat{\mu}_{new} = (1 - momentum) \times \hat{\mu} + momentum \times \mu_t \tag{5.9}$$

Note that when $momentum = 1$ the statistic is calculated solely from the target samples and when $momentum = 0$, it is calculated from just the train set.

5.4 Experimental Evaluation

5.4.1 Dataset Information

We experiment on 3 publicly available HAR datasets. We extract data from The Physical Activity Monitoring dataset (PAMAP2) [211] and The Opportunity Activity recognition dataset (OPP) [210] as mentioned in literature [79] [74]. Complex human activity recognition using smartphone and wrist-worn motion sensors (the UTWENTE dataset) [209] contains 13 activities from 9 participants. We extract accelerometer and gyroscope data from wrist-placed smartphones. This provides us with time series data with 6 dimensions.

For each dataset, we select certain classes as the source and others as the target. These target and source activity splits are shown in Table 5.1. Note that the split of Datasets PAMAP2 and OPP are the as same

Table 5.1: Source/target split for activities

source activities	target activities
PAMAP2	
Lie,Stand,Walk,Run,Rope Jump Ascend Stairs,Vacuum Clean	Sit,cycle,Nordic Walk,Iron Descend Stairs
OPP	
Open Door 2,close door 2 close fridge,clear table,drink from cup close drawer 1,2,3,toggle switch close dishwasher	Open Door 1,close door 1 open dishwasher open drawer 1,2,3 open fridge
UTWENTE	
Walk,stand,type,drink,talk,smoke,eat ascend stairs	jog,sit,bike,write descend stairs

as the paper by Feng and Duarte [79]. We create our own split for the UTWENTE dataset. To experiment with the effect of variations caused by users, we divide the participants of PAMP2 and OPP datasets as mentioned in Feng and Duarte [79] into 3 and 4 groups. We do not apply this to the UTWENTE dataset due to missing information on participants. We perform two types of testing on OPP and PAMAP2 datasets. They are when source and target data is drawn from the same group and different groups as mentioned in Feng and Duarte [79]. We fine-tune our system using the UTWENTE dataset for FSL performance and use OPP and PAMAP2 for evaluations. When preparing data we break the data sequences into sliding windows of 1 seconds with 50% overlap and standardized. No other pre-processing was performed.

5.4.2 Extended source target splits and EMS related dataset

For the previous section a single source/target splits were used. This section extends this by adding three more source/target splits for OPP and PAMAP2 datasets.

Table 5.2: Data splits

dataset		Source classes	Target classes
OPP	split1	Open Door 2,Close Door 2,Close Fridge,Close Dishwasher,Close Drawer 1,Close Drawer 2,Close Drawer 3,Clean Table,Drink from Cup,Toggle Switch	Open Door 1,Close Door 1,Open Fridge,Open Dishwasher,Open Drawer 1,Open Drawer 2,Open Drawer 3
	split 2	Close Door 2,Close Drawer 1,Close Drawer 2,Close Drawer 3,Close Door 1,Open Drawer 1,Open Drawer 2,Open Door 2,Open Door 1,Open Drawer 3	Close Dishwasher,Close Fridge,Open Fridge,Drink from Cup,Open Dishwasher,Clean Table,Toggle Switch
	split 3	Close Dishwasher,Close Fridge,Open Fridge,Drink from Cup,Open Dish-washer,Clean Table,Toggle Switch,Open Door 1,Close Door 1,Open Drawer 1,Close Drawer 1	Close Door 2,Close Drawer 2,Close Drawer 3,Open Drawer 2,Open Door 2,Open Drawer 3
	split 4	Close Door 2,Close Fridge,Close Dish-washer,Close Drawer 1,Close Door 1,Close Drawer 2,Close Drawer 3,Clean Table,Drink from Cup,Toggle Switch	Open Door 2,Open Door 1,Open Fridge,Open Dishwasher,Open Drawer 1,Open Drawer 2,Open Drawer 3

In addition to the public datasets, the FSL methods were evaluated for EMS related activities mentioed below.

5.4.3 Classifiers used for S_{FSL}

We use KNN which had been used for FSL settings [214] due to its simplicity [78] [82]. We also use a Distance Classifier (DC) to classify $f(x_{test})$ to the class centroid with the closest distance to it following prototypical networks [80]. We use euclidian distance as the measure of distance because it has proven to be more effective that some other distance measures in FSL setting [80]. Finslly we also use SVM classifier because sometimes it has show to out-peform simpler classifiers [215].

5.4.4 Fine-tuning

We fine-tuned the hyper parameters of the embedding function f separately for both center loss and prototypical loss based methods using the UTWENTE dataset. We modified a CNN architecture which proved to be successful for HAR with IMU data [216]. The FCN structure obtained is shown in Fig. 4. Here the embeddings are calculated from the $C3$ layer. The output from the last pooling layer goes to the source classifier CLS_S which uses a convolutional layer followed by a softmax layer for classification of source data. Parameters found for both of the methods are shown in Table 5.3.

Table 5.3: Hyperparameters used

Parameter	proto	CL
samples per class	10	8
embedding size	128	128
learning rate	0.001	0.001
discount	0.7	0.9
C1 kernel size	2	2
C2 kernel size	4	1
C3 kernel size	128	128
selected classes	6	2
λ	-	0.0001
support samples	7	5
Optimizer	Adam	Adam
num. episodes	1000	1000

Table 5.4: Performance on UTWENTE with KNN classifier for CLS_{FSL}

	1-shot	5-shot
proto	77.81	91.54
CL	83.63	93.25

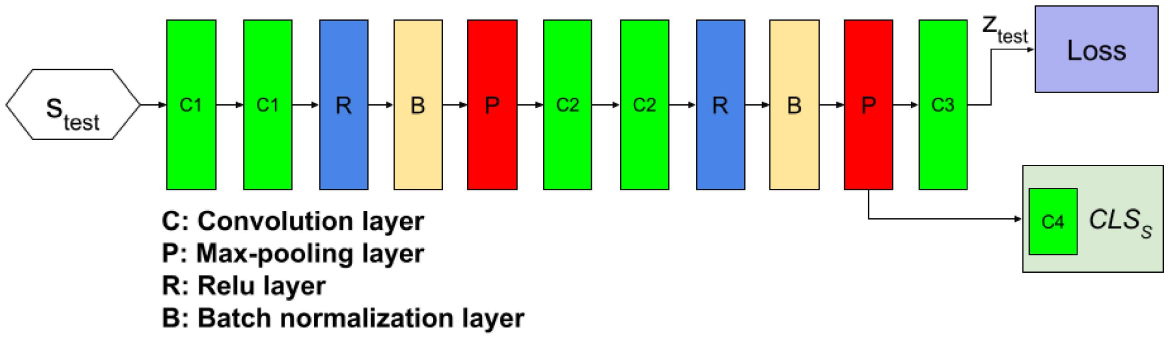


Figure 5.4: FCNN architecture

5.4.5 Performance of the Models

In this section we report the accuracy of C_{FSL}, C_S and C_{ST} for both prototypical and center loss based models, and where applicable we compare it with a weight transfer based method FSHAR from literature [79].

Table 5.5: Performance on PMAP2 using KNN for CLS_{FSL}

		1-shot			5-shot		
		1	2	3	1	2	3
base	same	50.87	57.67	58.98	65.95	62.84	70.08
	different	54.59	50.58	63.70	67.91	59.22	77.18
proto	same	59.33	63.84	62.67	74.92	79.18	83.76
	different	62.62	58.82	51.87	75.20	76.84	80.31
CL	same	58.65	62.22	64.23	74.01	78.38	84.11
	different	58.68	59.84	53.07	74.69	74.84	78.98

Table 5.6: Performance on OPP using SVM for CLS_{FSL}

		1-shot				5-shot			
		1	2	3	4	1	2	3	4
base	same	55.92	53.24	58.62	54.75	67.08	64.50	67.68	69.05
	diff	48.02	49.70	48.82	49.06	62.29	61.88	62.08	60.64
proto	same	57.99	58.12	59.65	56.53	75.82	73.08	76.72	72.50
	diff	51.62	50.93	48.41	49.22	70.22	68.43	62.38	67.00
CL	same	69.82	63.15	70.34	64.22	82.33	76.93	82.46	77.85
	diff	56.85	54.21	52.22	53.17	72.26	68.42	67.55	68.70

5.4.5.1 *FSL performance (CLS_{FSL})*

We show the number of training samples used to train CLS_{FSL} per each target class (1 or 5) and whether source and target data are generated by the same participant (for OPP dataset)/group of participants (for PAMAP2 dataset). Each value in this section is averaged for over 100 models. Tables 5.4, 5.5 and 5.6 shows these results. Note that for each dataset we only show the results of the best FSL classifier. KNN was the best option for CLS_{FSL} under both UTWENTE and PAMAP2 datasets while SVM was the best for OPP.

Number of shots: A common trend that can be seen from the results is that the performance level improves when we increase the number of training samples seen by CLS_{FSL} (number of shots). For example in Table 5.4, under the KNN classifier of CL model, the accuracy improves from 89% to 93% when number of shots is changed from 1 to 5. This can be expected because when CLS_{FSL} gets more data, it can make a more informed decision.

Training data used for the FSL classifier: For the OPP and PAMAP2 datasets, the average performance of CLS_{FSL} is higher when it is trained and tested with data from the same participant opposed to when they are trained and tested with different participants. This can be observed from Tables 5.5 and 5.6 and can be more clearly seen from Table 5.7. Table 5.7 shows the average performance of all the classifiers for CL model. We used KNN for all the CLS_{FSL} and CLS_{ST} classifiers except for CLS_{FSL} under OPP dataset where we used SVM due to its higher performance. In Table 5.7, the 5-shot average performance of CL model trained on OPP data when source and target data are from the same participant is 80% and when data comes from different participants, this value drops to 69%. This is true for all the participants/groups except for group 1 under 5-shot evaluation in PAMAP2 where these performance metrics are roughly equal as can be seen from Table 5.5. This characteristic where FSL performance is greater when source and target data are drawn from the same participants/groups than when they are different can also be observed under the FSHAR model evaluated on the OPP dataset. But, this is not so for the PAMAP2 dataset as can be seen in the FSHAR paper [79]. Following the reasoning from FSHAR paper [79] we can conjecture that this is because the OPP train and test data from the same participant have the same marginal distribution. Marginal distribution of data selected from different participants must be different. The PAMAP2 dataset on the other hand does not display the above behaviour. This might be because we group 3 participants into the same group for this dataset which may make the two marginal distributions of the same group dissimilar. We can conjecture that our centroid based models can learn relevant concepts from data better even when marginal distributions of train and test data are different.

Different methods of training embedding function From the results in Table 5.7, it can be seen that the centroid based models perform significantly better than the FSHAR method [79]. The average improvement of CL method over the FSHAR method [79] is around 10% for both the OPP and PAMAP2 datasets under 5-shot setting. The improvement is significantly greater when 5-shot setting is used opposed to using just 1 sample. For the OPP dataset, the average 1 shot improvement when using CL model over FSHAR method is around 8%. For 5-shot setting, this is over 10%. On the PAMAP2 dataset, 1-

Table 5.7: average performance of the CL model with 95% confidence intervals

		CL		from [79]	
		1-shot	5-shot	1-shot	5-shot
UTWENTE					
CLS_{FSL}		83.63 ± 0.69	93.42 ± 0.17		
CLS_S		84.87 ± 0.44			
CLS_{ST}		71.28 ± 0.80	75.06 ± 0.66		
OPP					
CLS_{FSL}	same	66.40 ± 0.81	79.87 ± 0.37	55.63	67.07
	different	54.12 ± 0.71	69.24 ± 0.45	48.90	61.72
CLS_S		96.21 ± 0.12			
CLS_{ST}		74.69 ± 0.60	83.45 ± 0.37		
PAMAP2					
CLS_{FSL}	same	60.57 ± 0.98	78.66 ± 0.66	55.84	66.29
	different	59.08 ± 0.98	76.79 ± 0.51	56.29	68.10
CLS_S		94.55 ± 0.17			
CLS_{ST}		77.95 ± 0.57	86.73 ± 0.31		

Table 5.8: Classification accuracy among target classes UTWENTE

	sim		SVM		KNN	
	1	5	1	5	1	5
sitting	78.88	90.57	78.88	90.40	78.88	88.97
biking	73.66	79.85	73.66	87.72	73.66	91.93
jogging	83.70	94.40	83.70	95.00	83.70	95.27
descending stairs	90.35	95.68	90.35	96.05	90.35	96.17
writing	91.52	95.16	91.52	94.71	91.52	94.74

shot improvement is around 3.7% and 5-shot improvement is over 10%. These improvements going from 1 shot to 5 shot settings can be attributed to CLS_{FSL} classifiers and to the quality of the embedding generated by f. Also the FSL performance improvement is higher when source data comes from the same participant/group than when the data comes from a different participant/group. For example, on OPP dataset considering the CL model, the performance improvement over the FSHAR method from the literature is around 12% when data comes from the same participant. This is only 6% when the data comes from a different participant. For PAMAP2 dataset these values are around 8% and 5%. This trend is also visible for prototypical networks. We can conclude that the centroid based methods learn more useful details from data than the FSHAR model does. Furthermore, the improvement in performance is larger when training data comes from a similar distribution to the test data. Also, we can see that in most cases CL model performs the best. This is true for the UTWENTE dataset, for all the settings on the OPP dataset and for certain cases of the PAMAP2 dataset.

Class-wise breakdown of the CLS_{FSL} performance can be seen in Tables 5.8 to 5.11. We also show the performance of all three classifiers used for FLS_{FLS} and the number of shots.

Table 5.9: Classification accuracy among target classes on OPP

	DC		SVM		KNN	
	1	5	1	5	1	5
open door 1	60.63	75.51	60.63	75.57	60.63	73.91
close door 1	71.71	79.82	71.71	80.18	71.71	79.66
open fridge	51.89	68.53	51.89	70.83	51.89	65.26
open dishwasher	46.71	61.48	46.71	62.26	46.71	62.37
open drawer 1	60.60	75.94	60.60	76.78	60.60	73.45
open drawer 2	54.28	70.68	54.28	70.74	54.28	70.99
open drawer 3	76.04	85.29	76.04	85.57	76.04	84.77

Table 5.10: Classification accuracy among target classes on PAMAP2

	DC		SVM		KNN	
	1	5	1	5	1	5
sitting	48.43	62.43	48.43	66.91	48.43	74.45
cycling	76.24	90.11	76.24	90.17	76.24	89.35
Nordic walking	78.04	85.24	78.04	89.02	78.04	91.57
descending stairs	43.23	59.87	43.23	64.82	43.23	65.99
ironing	53.18	65.25	53.18	69.08	53.18	67.26

Table 5.11: source class accuracy on UTWENTE

source class	accuracy
walk	96.88
stand	77.15
ascending stairs	98.32
type	90.00
drink	82.88
talk	81.33
smoke	70.56
eat	81.07

Table 5.12: source class accuracy on PAMAP2

source class	accuracy
lying	93.98
standing	93.90
walking	94.87
running	94.87
ascending stairs	92.08
vacuum cleaning	93.35
rope jumping	96.84

5.4.5.2 *Source classifier performance*

This section discuss the performance of the classifier which classify source samples (CLS_S) as can be see in Table VI. CLS_{FSL} of UTWENTE is around 93% while that of CLS_S is around 85%. But for OPP and PAMAP2, CLS_{FSL} performance levels are in mid-seventies while CLS_S are mid nineties. We can surmise that the embedding function might have overfit to the source data for OPP and PAMAP2 datasets. But UTWENTE models seem not to suffer from this. Therefore it is important to handle the overfitting when creating a FSL solution.

Interesting relationships between performance levels of source and target classes can be observed. From Table 5.8, it can be seen that descending stairs class has the highest 5-shot performance (for CLS_{FSL}) around 96% for the KNN classifier for the UTWENTE dataset. The highest performing source class is ascending stairs at 98% (see Table 5.11). We can surmise that this is due to the similarities between these two activities. Since the embedding function learned higher quality embeddings for the source class ascending stairs, this knowledge was easily transferred to the target domain. Sitting on the other hand has the lowest 5-shot performance at 89% among target activities under KNN (Table 5.8). This might be because there are no similar activities in the set of source classes of UTWENTE so the embedding function did not learn concepts similar to sitting. For PAMAP2 dataset, the highest achieving target activity (under 5-shot and KNN classifier) is Nordic walking which has an accuracy of 91.5% as can be seen from Table 5.11. On the set of source classes, the highest performing activity is rope jumping at almost 97% (see Table 5.12). We can hypothesize that this is due to the inherent similarities between the two activities.

Table 5.13: Classification accuracy on source classes on OPP

source class	accuracy
Close Dishwasher	96.62
Close Drawer 3	97.12
Close Drawer 2	90.07
Close Door 2	96.92
Close Drawer 1	92.26
Close Fridge	97.25
Toggle Switch	98.46
Open Door 2	95.94
Drink from Cup	97.38
Clean Table	99.42

Table 5.14: Classification accuracy of source classes vs. target classes

		DC		SVM		KNN	
		1	5	1	5	1	5
UTWENTE	S	56.79	57.29	32.83	47.12	60.79	72.08
	T	86.70	87.22	99.41	98.40	81.77	86.47
OPP	S	56.27	60.51	22.59	70.17	66.62	79.09
	T	64.89	62.84	95.60	87.71	82.77	87.81
PAMAP2	S	69.02	72.55	44.00	83.29	74.13	85.53
	T	70.10	73.33	91.25	85.07	81.76	87.94

Both activities involve significant and synchronized arm and leg movements. Ascending stairs from the set of source classes in PAMAP2 dataset and descending stairs from the target classes both are the lowest performing activities in their respective domains (Tables 5.11 and Table 5.12). We can infer that the activity descending stairs did not perform well in the target domain due to the embedding function did not learn concepts related to ascending stairs well in the source domain.

5.4.5.3 *Source/target classifier performance*

Table **??** demonstrates that performance of CLS_{ST} on all 3 datasets. For each dataset we show the accuracy of detecting both target (T) and source (S) samples under 3 different types of classifiers DC, SVM and KNN, under 1-shot and 5-shot setting. For 5-shot setting, each classifier is trained with 5 samples from each class in source and target domain and then tested on the rest of data. It is evident that KNN is the best overall choice for CLS_{ST} which has the average performance of 75%, 83% and 87% on the UTWENTE, OPP and PAMAP2 datasets. A summary result of this can also be seen in table 5.7. We have only reported CLS_{ST} performance under KNN dues to its highr performance.

It can be seen that the performance of our CLS_{FSL} models surpasses those from literature [79]. Although the 1-shot accuracy values are not sufficiently high for any practical use, 5-shot accuracy values show promise. Therefore we can recommend to use the model CL with 5-shot criterion for a FSL-based HAR system.

5.4.6 Test time adaptation for Few Shot Learning

This section extends the previous experiments using test time adaptation as shown in Table 5.15. It can be seen that using test time adaptation improves the performance significantly specially in 5 shot setting. The improvement is more significant when BN momentum is 0; that is when we use the train data statistics instead of the target dataset statistics. This may be due to the statistics derived with just 5 samples in the target dataset may be too biased. The improvement under 1 shot setting is not significant; in some cases the performance decreased. Therefore, under 1 shot setting, using TA is not necessary.

Next we evaluated the model trained on PAMP2 data on target data which contains all the target classes from the PAMAP2 and also the data from a CPR data session. The IMU data we collected during CPR contained 6 dimensions (3 axis accelerometer and 3 axis gyroscope). Therefore we trained the model with the same sensor outputs from PAMAP2 dataset and evaluated on the combined dataset. The results are shown in Table 5.16. The new activity, CPR can be recognized with a high accuray.

5.5 Conclusion

We present the first generalized FSL system for human activity recognition with wearable devices. We show that our methods outperform the state-of-the art in the FSL task. Our system has real practical applications, especially as these applications evolve over time. For example, consider elderly people living alone. We can use wearable devices which are programmed to recognize certain set of activities. But as the condition of the elderly change over time, they might end up doing different activities to what the device is previously programmed for. We can use our system to detect these new activities with just a few samples from each new activity.

Table 5.15: TA evaluation on OPP dataset

split	Method	1 Shot	5 Shot
split 1	Base	52.79	62.55
	CL	56.11	66.79
	TA mom = 0.0	57.37	75.93
	TA mom = 0.5	55.79	75.13
	TA mom = 1.0	56.83	75.64
split 2	Base	52.41	62.55
	CL	58.57	66.79
	TA mom = 0.0	57.76	75.93
	TA mom = 0.5	59.63	66.79
	TA mom = 1.0	60.24	66.79
split 3	Base	59.70	62.55
	CL	58.61	66.79
	TA mom = 0.0	57.51	75.93
	TA mom = 0.5	56.55	66.79
	TA mom = 1.0	57.73	66.79
split 4	Base	65.91	62.55
	CL	67.34	66.79
	TA mom = 0.0	64.95	75.93
	TA mom = 0.5	65.89	66.79
	TA mom = 1.0	64.61	66.79

Table 5.16: FSL evaluation on data with EMS data

Activity	5-shot FSL accuracy
sitting	0.76
cycling	0.56
Nordic walking	0.75
Descending stairs	0.28
Ironing	0.40
CPR	0.85

CHAPTER 6

ROBUSTNESS TO NOISE FOR SPEECH EMOTION RECOGNITION

6.1 Introduction

This chapter explores solutions to develop noise-robust speech emotion recognition system for the EMS providers. To this end, we conduct experiments with several techniques for vocal emotion recognition under several different environmental noise conditions on public emotion recognition datasets. We hope the techniques that we develop in this chapter can be expanded into real world EMS provider setting with minimal modifications.

There are other domains that Automatic emotion recognition (AER) can be used [153]. For example, it can be used for interaction between human and machines such as robot health assistants [156]. There are many applications for emotion understanding in health such as diagnostic tools for therapists, helping caregivers of dementia patients, helping post traumatic stress disorder patients, assessing the emotional state of callers to a emergency call center, and even outside of health such as to assess emotional state of drivers in cars [156] and for media [154].

One of the weaknesses of the state-of-the-art SER systems is they are very sensitive to noise. These noises are caused by other sound producing sources in the vicinity of the SER system which can corrupt the speech signal the SER system is analysing. But only very limited amount of research was performed addressing this problem in the context of SER. Therefore, in this chapter, we develop, analyze, compare and propose several potential solutions to solve this problem.

Convolutional Neural Network (CNN) models were used for SER since they have the state-of-the-art performance in the literature. Several spectrogram based feature types as inputs to the CNN models were used, compared and combined in this study. All of these spectrograms were obtained by performing Discrete Fourier Transformation (DFT) on the speech signals. Traditionally, SER models use only magnitude spectrograms for audio classification. But in this study we also use Modified Group Delay (MGD) spectrograms [171] and unwrapped phase spectrograms obtained with python package scipy [**scipy**].

The main contributions of this chapter are:

1. Majority of state-of-the-art SER models are CNNs and use magnitude spectrogram as the input. But performance of these models degrade significantly with background noise. There are evidence from literature that Modified Group Delay (MGD) may be robust under noise. We combine magnitude spectrogram with MGD spectrogram and show that the Fully Convolutional Neural Network (FCNN) models trained with this combined input is more robust to noise than just using magnitude spectrogram. We perform initial experiments on the Berlin Database of Emotional speech. Using other techniques such as training with artificially added noise and attention mechanism alongside the combined input we show that an average improvement of 15% accuracy (F1 of 0.16) can be obtained under noisy conditions when compared to a traditional model which only uses magnitude spectrogram as input.

2. We show that including synthetic noise in the training data improves the noise robustness of all the models considered. Interestingly, the model with combined magnitude and MGD spectrograms as inputs saw a larger improvement than those used individual spectrogram inputs. Doing this step improved the accuracy by 10% (F1 by 0.11) over the model which used magnitude spectrogram alone and did not train with noisy data.

3. Including an attention mechanism to the FCNN model with combined input and training with noisy data also improved the noise robustness by an accuracy of 5% (F1 by 0.05). We also provide evi-

dence that attention mechanisms can ignore noisy data sections of speech and pay more attention to important and cleaner sections of the speech.

4. Our best model showed an average accuracy of 76% (F1 of 0.73) over all the noise levels (Signal to noise ratios from 10 to 35 and clean speech) and all the noise types considered. If the performance under AWGN is considered our model had an accuracy of 76% over all the noise levels (including clean speech). This performance is a significant improvement over the model mentioned in [217] which reports an accuracy of 56%.

5. The above results were obtained with the Berlin Database of Emotional speech. Next the best fine-tuned model architecture, noise robust features and the hyperparameters chosen from those steps were used and trained on the RAVDESS dataset [218]. This model obtained an accuracy of 91% (F1 - 0.91) under clean speech and 81% average accuracy (F1 - 0.81) under all noise types and levels considered. This shows that our solution can be generalized to other datasets and emotions.

The remainder of this chapter is organized as follows. First we discuss the methods and solution framework that was used in Section 6.2. Then, we present the experiments and results in Section 6.3. A discussion and conclusions are presented in Sections 6.4 and 6.5.

6.2 Methods and solutions

The aim of this study is to find features, training procedures and model architectures that are robust to noise for the task of SER. This section describes the methods used to achieve this, problems faced and solutions.

6.2.1 Data Sets

For the first sections of this study audio speech from the Berlin Database of Emotional Speech is used [219]. It contains around 500 utterances spoken in German. The emotions happiness, anger, anxiety/fear, disgust, boredom, sadness and neutral are present in the dataset. First the hyperparameters of the models were tuned and appropriate input features were selected using the Berlin dataset. Afterwards the model with those hyperparameters and input features were tested on the speech audio data from the RAVEDESS dataset [218]. This dataset contains two intensities of emotion expression; strong and normal. Only the strong emotional expressions were used. For this study, the emotions calm, happy, sad, angry, fearful and surprised were used from the RAVDESS dataset. 575 utterances were present in the selected data from RAVDESS.

6.2.2 Noise

Noise types used. Different types of noise can affect the performance of a SER system. But an analysis of SER performance under different noise types is not performed in an adequate manner in the literature. Therefore, a set of 9 common indoor noises were selected for this study from the Freesound Dataset [220]. They are the sounds of crickets, alarms, kettle whistling, rain, steps (person walking), thunder with rain, traffic noise, vacuum cleaner, and air conditioner. In addition, additive white Gaussian noise (AWGN) was used.

These noise types are selected because they are very common sources of noises in indoor environments. For example, air conditioner may be always on during summer. Also sounds with various characteristics were selected. For example, unlike air conditioner hum, steps is of intermittent nature and alarm sound has a higher and well defined set of frequencies. Note that some of these noise types such as steps, rain, traffic noise are very common in EMS situations.

Additive white Gaussian Noise. This kind of noise can be added (arithmetic element-wise addition) to the signal. Also its mean value is zero (randomly sampled from a Gaussian distribution with mean value of zero; standard deviation can vary). It contains all the frequency components in an equal manner. AWGN is easier to model and easier to generate. Since AWGN contains noise equally in all the frequency bands, researchers (e.g. [221]) use it to approximate different types of noises and to compare the performance of different models and features.

The SER models trained during this study were evaluated with all of these noise types. It is hypothesized that a model which is competent under these noise conditions will be able to handle a wide variety of indoor noise types which are not used here.

Signal to noise ratio. For the experiments in this study, one of the qualities that was evaluated is the performance of a classifier under noisy conditions. Signal to noise ratio (SNR) is a method to quantify how much noise is present in a signal. For the purposes of this study, signal is the speech. Noise is one of the noise sources described in the last section. SNR can be defined as follows.

$$SNR = 10log(\frac{RMS_{signal}^2}{RMS_{noise}^2}) \tag{6.1}$$

where RMS_{signal} is the RMS value of signal and RMS_{noise} is that of noise. log represents the logarithm of 10. In the experiments performed, noise was added to the clean signal at different SNR levels and the performance of models were observed.

6.2.3 Feature selection

Magnitude spectrogram is the most commonly used input type for CNN based SER systems. So, it will be included in this study. Since there are evidence suggesting including phase information makes models perform better, we also use unwrapped phase spectrogram as an input. Also, since modified group delay (MGD) was theoretically proven to be robust to noise as described in section 2.4.5 we include MGD spectrogram in our study. All 3 of these input types are Fourier transform based.

Different input types and their combinations may have different noise robustness characteristics. Therefore, models were trained taking each of these input types and some of their combinations as inputs and their performance under noise is compared. This way, the best performing input types/input combinations can be selected.

6.2.4 Creating the FCNN architecture

To find the best CNN architecture for each feature type, the auto ML package Autokeras [222] was used. To build the classifier with Autokeras, ImageClassifier class with $max\_trials = 20$ and fit function with $epochs = 20$ were used. An interesting observation is that the optimum CNN architecture found by Autokeras was the same for all the situations (Different input types and their combinations). After obtaining a CNN architecture, it was converted into a FCNN by replacing all the fully connected layers with convolutional layers. The architecture is shown in Table 1. Note that there are 7 filters in the last convolutional layer. These filters correspond to the 7 different emotion classes in the Berlin dataset which is used in initial experiments.

Table 6.1: FCNN architecture

Layers
conv2D 32 3x3 1
Activation relu
Max Pooling 2x2 2
conv2D 64 3x3 1
Activation relu
Max Pooling 2x2 2
conv2D 16 3x3 1
Activation relu
Dropout 0.5
conv2D 7 3x3 1
Global Average Pooling
Activation softmax

6.2.5 Calculating spectrograms

First, start and end silence sections were removed from voice samples. All the voice samples were rescaled so they will be between $[-1, 1]$. Then features were calculated from these samples. Note that the length of these voice samples varied since FCNN can handle variable length inputs. Note that MGD has the parameters α and γ from Section 2.4.2. After some trial and error $\alpha = 0.6$ and $\gamma = 0.5$ was selected and kept constant throughout this study.

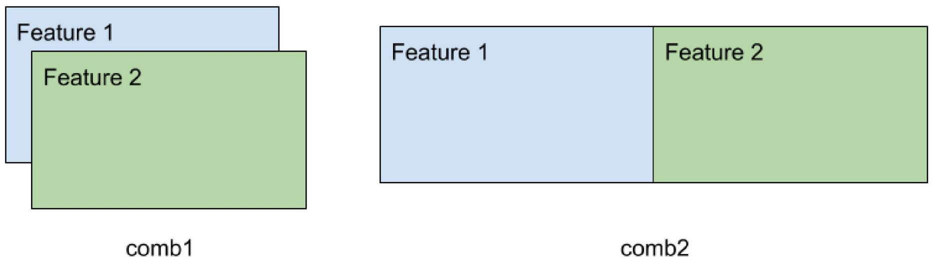


Figure 6.1: Combining Features

The input types used are magnitude, MGD and unwrapped phase spectrograms. To calculate these, the voice signal was broken into blocks of desired length (depending on the desired DFT window length) and then applied a hamming window. Afterwards DFT was performed. From the DFT results, magnitude, MGD and unwrapped phase were calculated. DFT was performed with python library scipy.

DFT window length. Different lengths for the blocks (this is called DFT window length in this study) were chosen. The frequency and time resolution of the spectrograms depend on the chosen DFT window length. Larger lengths results in higher frequency resolution and lower time resolution and vice-versa. Therefore different DFT lengths may react to various noises in different ways. The noise performance of the models may depend on the DFT window length chosen. These lengths are chosen to be 25 ms, 50 ms, 75ms and 100ms. A separate model for each of these DFT window lengths were trained.

6.2.6 Combining Input spectrograms

The goal of this section is to find the effects of combining different inputs on noise robustness of SER. It is hypothesized that different methods of combining different inputs may yield varying levels of SER performance and combining inputs may provide better results than just using individual inputs. To test this some input types were combined in two different ways and compared.

Two different methods of combining two feature types were used in this study. They are combining them as different channels of an image (called comb1) and combining them side-by-side (called comb2). Figure 6.1 shows the two methods of combining feature types. If both $Feature\_1$ and $Feature\_2$ are 2D arrays of dimension $h * w$, the dimension of comb1 would be $h * w * 2$ and dimension of comb2 would be $h * 2w$. A CNN can be trained using regular methods with both of these input types.

6.2.7 Model training

All the FCNN models were trained with Keras deep learning library with the optimizer adam. The starting learning rate was 0.001. To train all the FCNN models except under batch training in section 6.3.5 and when training on RAVDESS in section 6.3.8 batch size of 1 was used because keras can only have fixed length inputs in the same batch. For details on the batch training refer section 6.3.5. Learning rate was decayed by $1 * 10^{-6}$ once every epoch. Class weights were used while training due to the class imbalance. These class weights were inversely proportional to the number of instances present in each class.

6.2.8 Model testing

In the initial experiments, for each feature and DFT window length 10 fold cross validation was performed on the Berlin Database of Emotional Speech. First each model was evaluated under clean speech test set. In order to evaluate the noise robustness of these models, they were evaluated under clean speech clips mixed with various noise types at various noise levels (measured by SNR). SNRs used are 10, 15, 20, 25, 30 and 35. For example when evaluating a model under noise of crickets, cricket noise was mixed to the original clean test data set at different SNRs. The same procedure was performed for all the different noise types including AWGN. Afterwards the model accuracy and F1 score was obtained. F1 score is used to report most of the results because it is more useful for evaluating datasets with class imbalance. For the RAVDESS dataset, train data was prepared with 80% of the data and testing was performed on the rest. Procedure for testing under noisy conditions was identical to that of the Berlin dataset.

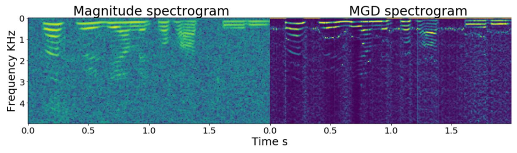


Figure 6.2: comb2 with AWGN

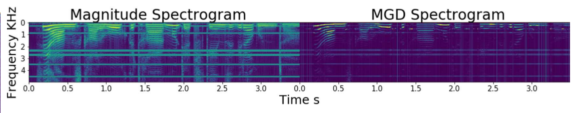


Figure 6.3: comb2 with noise at several frequency bands

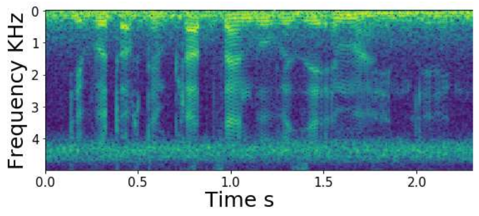


Figure 6.4: Magnitude spectrogram with cricket noise

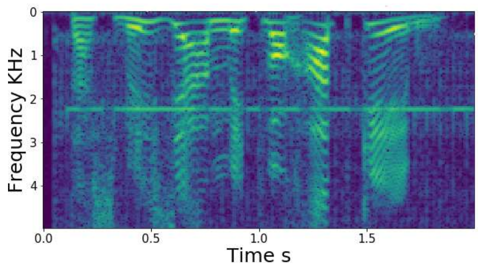


Figure 6.5: Magnitude spectrogram with alarm noise

6.2.9 Training with noise

One reason that the models perform worse on the testing data is that the distributions of train and test data are different. If models are trained with clean speech and used in real world noisy environments, the model performance degrades because the test data contains noise which was not present in train data. To bridge this gap, noise may be included in the training data. It is hypothesized that by injecting noise to training data, the performance of the model on test data under noise can be improved.

To prepare the training data, 3 data sets were generated from the clean speech data. One is the clean speech data itself. A second data set was prepared by injecting AWGN of SNR 40 to clean speech. Figure 6.2 shows one instance of these samples. A third data set was created by injecting random noise at several random frequency bands. Figure 6.3 shows one of the samples with this type of noise. Note that Figures 6.2 and 6.3 show magnitude and MGD spectrograms combi mb2.

6.2.10 Adding attention mechanism

An attention mechanism was added as shown in the Figure 6.6. It functions according to equations 6.2 and 6.3. This attention mechanism is influenced by and modified from the one in [186]. [186] implements the attention mechanism for a traditional CNN architecture with fully connected layers. This had to be modified to fit FCNN which only contains convolution layers as feature extractors. A convolution layer is used for extracting attention weights. There was no previous research done regarding combining FCNN models with attention mechanism and evaluating its noise robustness. Although [186] used both spatial and channel-wise attention this study only uses spatial attention due to increased complexity involved in implementing both types of attention at the same time. Due to the smaller nature of the datasets used in this study, it can be imagined that increasing the number of model parameters will increase over fitting.

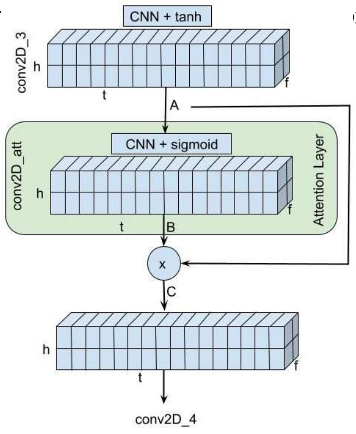


Figure 6.6: Attention Mechanism

$$B = sigmoid[conv2D_{att}(A)] \tag{6.2}$$
$$C = A * B \tag{6.3}$$

Attention mechanism can be used to teach the model in an explicit way to focus on relevant sections of the input and ignore irrelevant parts in the time-frequency space. The motivation behind using attention mechanism to create noise robust SER models is that certain noises affect localized regions in the time-frequency space. For example Figures 6.4 and 6.5 show voice clips mixed with cricket and alarm sound. Majority of these noises are contained in some particular frequency ranges. Alarm noise in this example corrupts a narrow frequency range around 2.1KHz. Cricket sound has corrupted some lower frequencies and higher frequencies around 4-5KHz. If the model learns through attention mechanism to focus on clean sections of the input and ignore the noisy regions it is hypothesized that the model will be more robust to these types of noises.

6.3 Experimental Results

6.3.1 Training FCNN

The FCNN architecture shown in Table 6.1 was trained with spectrograms generated from voice recordings of variable lengths. Figures 6.7, 6.8 6.9 and 6.10 shows the performance of the FCNN models under various levels of noise. The SNRs used are 10,15,20,25,30 and 35 as mentioned in section 6.2.8. SNR of infinity corresponds to clean speech. In Figures 6.7, 6.8 and 6.9 at each SNR, the average performance of each model under all of the noise types mentioned in section 2.2 are shown. A separate model was trained for each DFT window length and feature type. The DFT window length used is shown in the legend. Figure 6.10 shows the performance breakdown under different noise types for the best model trained under each feature. For each of these noise types the average performance under the various SNR were taken.

Figure 6.7 shows the mean accuracy of the FCNN model trained with magnitude spectrogram as input. The model trained with DFT window length of 75ms shows the best overall performance under noisy conditions. Under clean speech, it shows an F1 score of 0.71. Interestingly the worst performing model under noise (25ms model) performs best under clean speech. The performance levels of all the models drop significantly with the addition of noise. Figure 6.8 shows the performance of the MGD model. 50ms model shows best performance under noise and clean speech. It shows an F1 score of 0.75 under clean speech. Figure 6.9 shows the performance of the model which uses phase spectrogram as input. The best overall performance was achieved when DFT window length is 75ms and was 0.42 under clean speech. Figure 6.10 compares the best models from each feature. Here the magnitude, MGD and phase features were derived using DFT window sizes 75ms, 50ms and 75ms respectively.

From these plots it can be seen that different features may have different performance for clean speech and under various noise levels. Also using different DFT window length may yield different performance both for clean and noisy speech. The DFT window length which gives the best performance under clean speech may not be a good choice for SER under noisy speech. From the features evaluated in this section, MGD shows the most significant drop of performance going from clean speech to noisy speech. For example the performance of the 50ms MGD model drops from 0.75 to 0.49 when going from clean speech to speech with SNR of 35. But its performance is very stable around 0.50 until SNR is 15. From Figure 6.10 it can be seen that for certain noise types MGD performs better and for the others mag is better. MGD performs better under noise types air conditioner, rain, vacuum cleaner and white noise. Note that these noise types

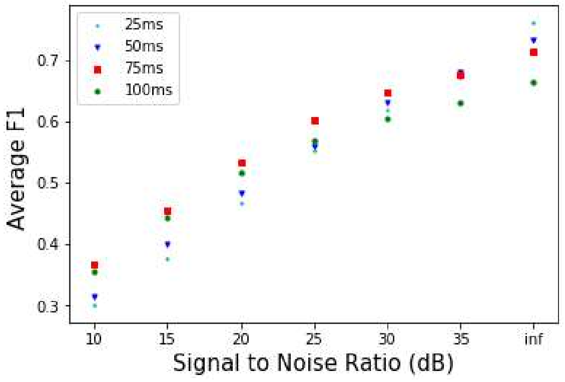


Figure 6.7: Performance of magnitude models under average noise

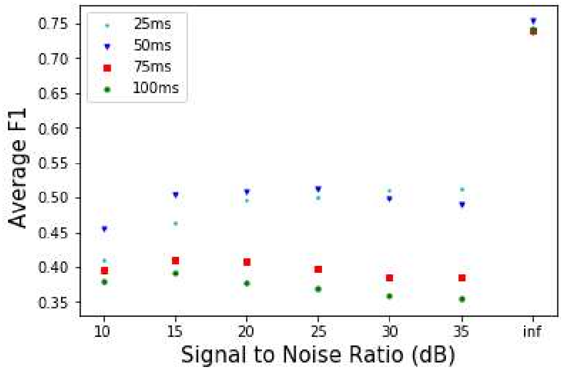


Figure 6.8: Performance of MGD models under noise

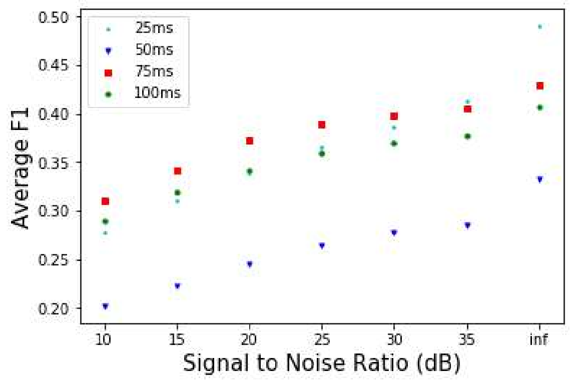


Figure 6.9: Performance of phase models under average noise

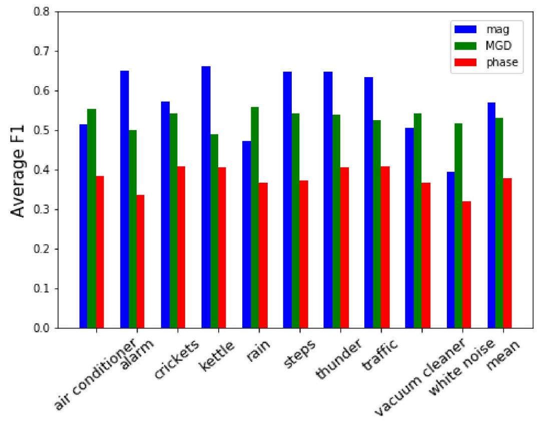


Figure 6.10: Comparative performance under all types of noises

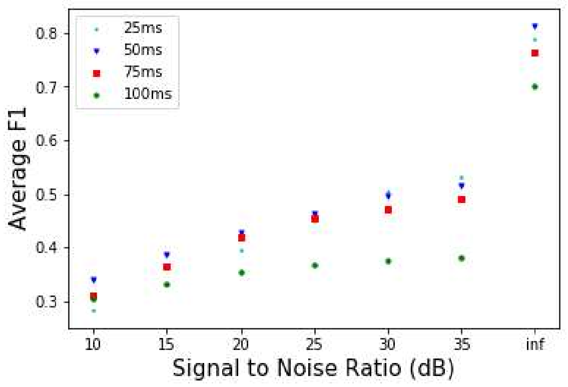


Figure 6.11: performance of comb1 models under noise

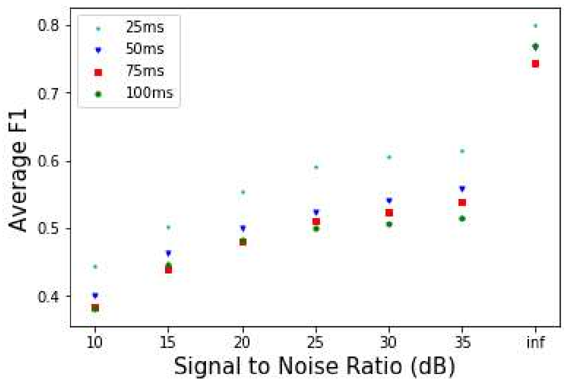


Figure 6.12: performance of comb2 models under noise

are more similar to white noise and corrupts broad frequency bands when compared to other types of noises used here. Phase always performs the worst under all the noise types. Therefore only magnitude and MGD features were selected for further analysis. From this section it can be seen that not only magnitude but also phase based features like MGD can be used for SER.

6.3.2 Combining spectrograms

Next the effects of combining different input types to FCNN were studied. These input types were combined according to two methods (comb1 and comb2) as discussed in section 6.2.6. From the previous section it can be seen that the performance may depend on the selected DFT window length. So for each combination method different DFT window lengths were used to train and validate the models.

From Figure 6.11 it can be seen that the DFT window length 50ms performs better for combination method comb1. Under clean speech comb1 model with 50ms DFT window length yields a F1 score of 0.81. For comb2, 25ms is the best DFT window length as can be seen from Figure 6.12. The performance of this model under clean speech is 0.8.

Next the best models for magnitude spectrogram, MGD spectrogram, comb1 and comb2 input types were compared. Figure 6.13 and Figure 6.14 shows the relative performance of these models. Figure 6.13 was obtained by taking the mean performance of the best magnitude, MGD, comb1 and comb2 models over different SNR under all the noise types. Values in Figure 6.14 were obtained by taking the mean value of performance under all SNRs under individual noise type.

Although the best performing model under clean speech is comb1 which gives a F1 score of 0.81 according to Figure 6.13, it never performs the best at any noisy condition (see Figure 6.14). Also comb1

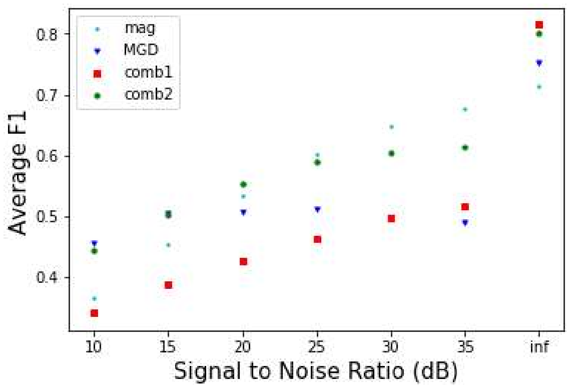


Figure 6.13: comparative performance of models under noise

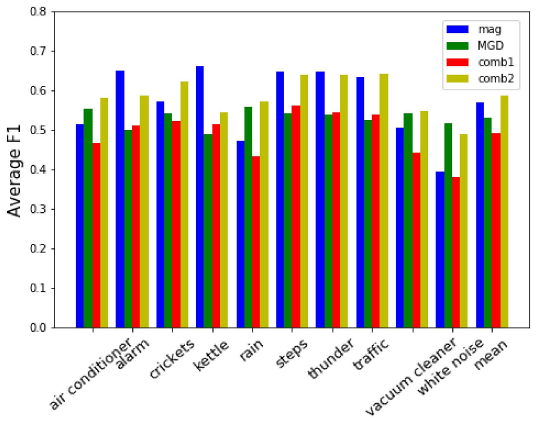


Figure 6.14: comparative performance of models under noise

is very sensitive to noise since its F1 drops to 0.51 under a little addition of noise having a SNR of 35. Therefore, we can safely eliminate comb1 from further consideration. According to Figure 6.13, the mag model shows an F1 score of 0.71 under clean speech. It has the best performance under noisy conditions among all the models until SNR drops to 20. But from Figure 6.14 it can be seen that comb2 model performs the best when we take the mean performance over all the noise types. Also a different kind of model may be the best one under a different noise type.

From the results of this section, it can be concluded that by combining phase based features such as MGD with magnitude, models more robust to noise can be built (compared to model just using magnitude as input). Also different methods of combining features can have different levels of noise robustness. Under these conditions comb2 method seems significantly better than comb1 method of combining magnitude and MGD features.

6.3.3 Training with noisy data

Previously, all the models were trained with just clean data. In this section the effects of training with noisy data is studied. Three best performing models from previous section (mag, MGD and comb2) were selected and retrained with data mixed with artificial noise as mentioned in section 6.2.9. When trained with noise, mag, MGD and comb2 models were named noise_mag, noise_MGD and noise_comb2. Figure 6.15 shows the improvement obtained by training with noise for each model. This figure shows the average performance under all noisy conditions and all SNRs. All the models saw an improvement when trained with noise. F1 score of magnitude model improved from 0.57 to 0.59. MGD model improved from 0.53 to 0.61 and comb2 improved from 0.58 to 0.68.

Figure 6.16 shows the mean performance of models under all the noisy conditions under various SNRs. The model trained with magnitude spectrograms perform the worst in general and noise_comb2 performs the best. Figure 6.17 shows the performance of the models under all the different noise types. For each noise type, the average over all the SNRs were taken. It can be seen that noise_comb2 performs the best for each and every individual noise type.

From the results of this section, it can be concluded that training with artificial noise, the noise robustness of SER models can be improved. Furthermore, the improvement is greater when the model takes both magnitude and MGD inputs compared to just using one of them. One other interesting observation from Figure 6.17 is that noise_comb2 is the best model under all the different types of noises used. From experiments in previous sections it can be observed that no model was performing the best under all the noise types like this. Therefore these results show evidence that by training under noise and using both magnitude an MGD as inputs we can build SER models that are robust to many different types of indoor noises.

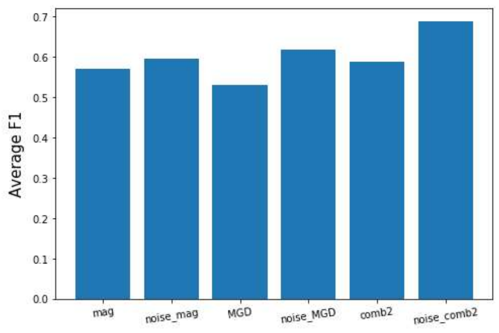


Figure 6.15: Improvement when training under noise

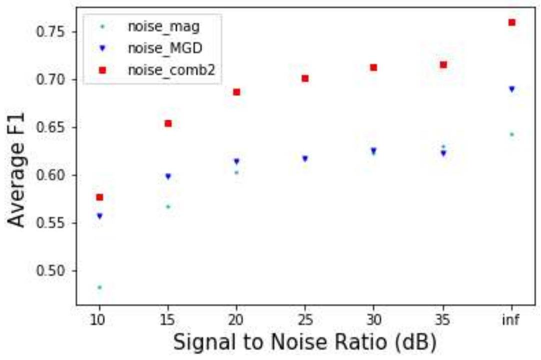


Figure 6.16: Mean performance when training under noise

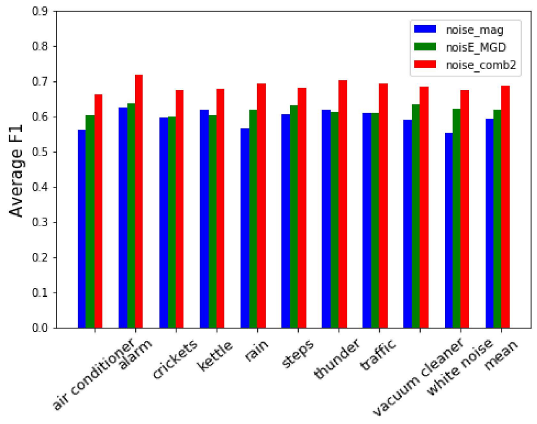


Figure 6.17: Performance when training under noise

6.3.4 Incorporating attention mechanism

To test the effectiveness of the attention mechanism, several models from the previous section were chosen and attention mechanism was added to them as described in section 6.2.10. Magnitude was not considered in this section because it was the worst performing model from previous section. All the models in this section are trained with noise as mentioned in the previous section. In addition to that, attention mechanism was incorporated. The models with attention mechanism are called att_noise_MGD and tuned_att_noise_comb2. These models use MGD and comb2 features respectively. tuned_att_noise_comb2 was fine tuned to increase the performance. During the fine tuning process, several filter sizes and number of filters were tried and the fine tuning was performed with the validation accuracy value of clean data. No such fine tuning was performed on att_noise_MGD. The reasoning behind just choosing the com2 model for fine tuning is that from the previous section it was seen that noise_comb2 performs the best under all different noise types and levels. Therefore it was assumed that com2 model will perform better with the added attention mechanism and fine tuning.

From Figure 6.18 it can be seen that the average F1 score of MGD model improves from 0.61 to 0.64 after adding attention mechanism. tuned_noise_att_comb2 performs the best. This comb2 model improved from 0.68 to 0.73. From Figure 6.19 it can be seen that for clean speech, this model shows a F1 score of 0.85.

Figure 6.20 shows the per emotion performance of tuned_noise_att_comb2. These values are taken by averaging the F1 scores over all the different SNR levels and noise types per each emotion.

This section provides evidence that incorporating attention mechanism to CNN based SER systems can improve its robustness to noise. This may be due to the ability of attention mechanism to focus on important sections of the input and ignore the rest.

Figure 6.21 compares the performance of the model tuned_att_noise_comb2 with the model W-WPCC

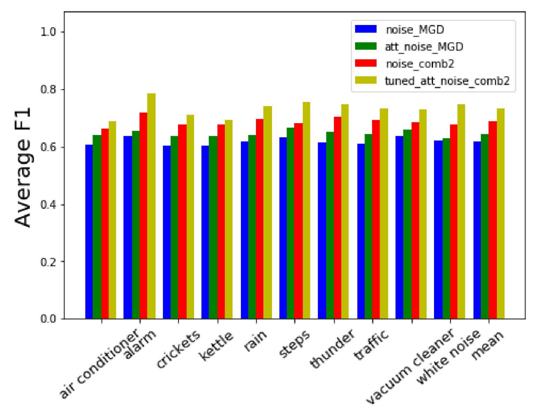


Figure 6.18: Improvement when under attention model

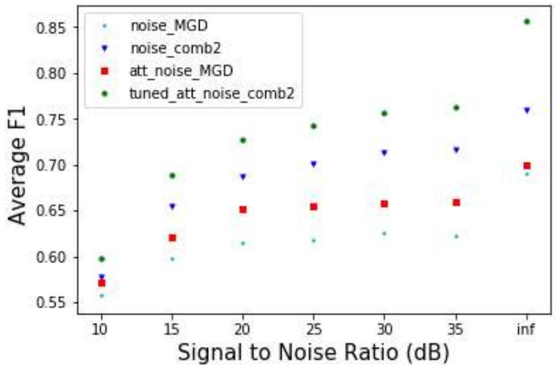


Figure 6.19: Mean performance under attention

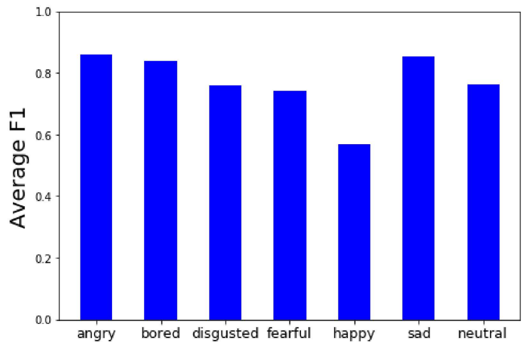


Figure 6.20: Per emotion performance

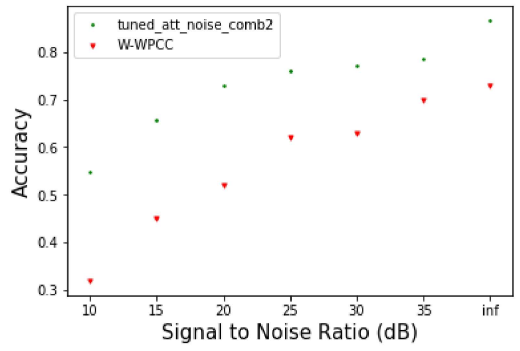


Figure 6.21: Comparison with W-WPCC model from [217]

from [217]. They use an importance-weighted support vector machine to classify features based on sub-band spectral centroid weighted wavelet packet cepstral coefficients. [217] evaluates their model only under AWGN conditions. Therefore Figure 6.21 shows the performance of both models only under AWGN. Our model performs better under both clean and noisy speech. Note these results use accuracy instead of F1 score because [217] only reports accuracy. For clean speech, tuned_att_noise_comb2 performed at an accuracy level 86% and W-WPCC was 73%. Under speech of SNR of 20, our model performed at 72% and W-WPCC performed at 52%. If the average performance under all the noise types, SNR from 15 to 35 and clean speech is considered, W-WPCC performs at accuracy of 56% and our model performs at 76%. Note that [217] uses noise robust features. But they do not employ other techniques such as training with noise or attention mechanism.

6.3.5 Effects of batch-wise training

This section explains the effectiveness of batch training of the tuned_att_noise_comb2 model. Here the model batch_tuned_att_noise_comb2 was trained batch-wise. During earlier experiments, all the models were trained with batch size of 1 since Keras does not allow variable input sizes in the same batch even with FCNN. During batch training, for each batch the inputs were cropped into a randomly chosen length (along time axis). So all the samples in a batch had the same size. The evaluation procedure is the same as before. After a few trial-and-error experiments, a batch size of 16 was selected. Figure 6.22 and Figure 6.23 compares the F1 scores obtained via batch training and instance wise training. From Figure 6.23 it can be seen that the average performance of batch training under lower noise levels (SNR $\geq$ 20) is significantly better than that of the instance wise trained model. According to Figure 6.22 the mean performance

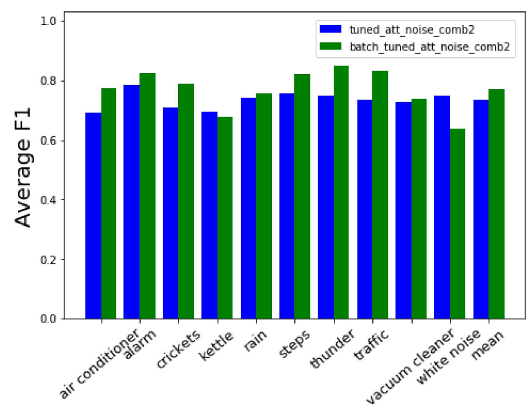


Figure 6.22: Comparison of batch training and instance wise training

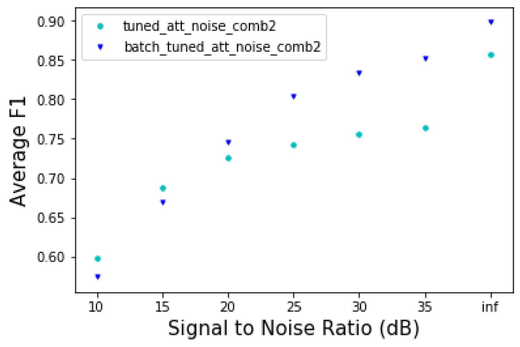


Figure 6.23: Comparison of batch training and instance wise training

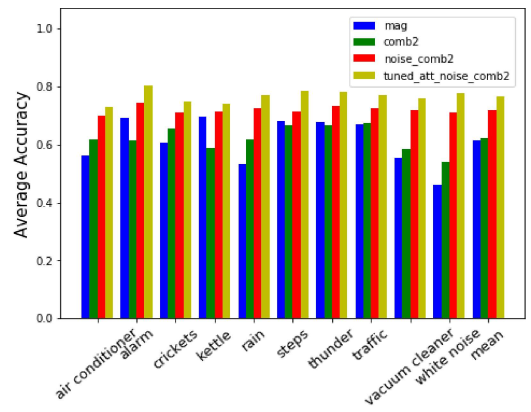


Figure 6.24: Improvement comparison of different methods

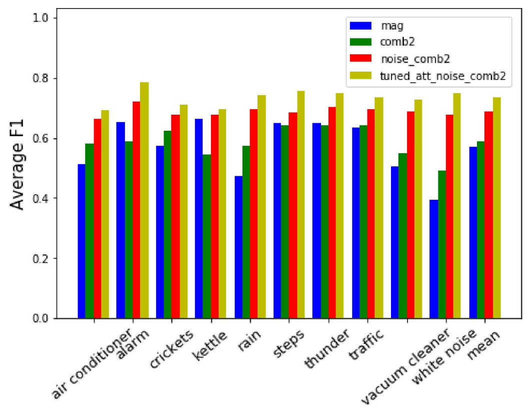


Figure 6.25: Improvement comparison of different methods

improves from 0.73 to 0.76 when trained batch-wise. But interestingly the batch trained model performs worse than the instance trained model in certain noise conditions such as white noise and kettle noise. Batch trained model is much better than the instance model under some noise conditions such as thunder and traffic noise. Therefore this batch trained model may be useful under certain noise conditions.

6.3.6 Comparison of various methods on noise robustness

This section presents the effectiveness of various methods used to improve noise robustness and their comparison. Figure 6.24 shows the accuracy levels of the models trained with just using magnitude (mag), comb2, com2 model trained with noise and comb2 model with attention trained with noise. Taking mag model as a baseline, using combined features (mag and MGD) improved the accuracy by 1%. Using combined features and training the model with noise improved the accuracy by 10%. Using comb2, noise in training data and also incorporating attention mechanism improved the accuracy by 15%. From Figure 6.25, these value in F1 score are 0.01, 0.11 and 0.16. From these results it can be seen that combining magnitude and MGD spectrogram and training the model with noisy data have the biggest impact on building a noise robust model.

6.3.7 Operation of attention mechanism

Attention mechanism explicitly instructs the CNN model to focus only on important sections of the input features. This can be demonstrated by few examples. Figure 6.26 shows one example input to the classifier.

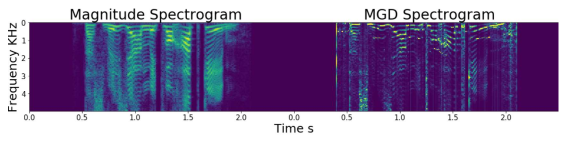


Figure 6.26: comb2 with silence

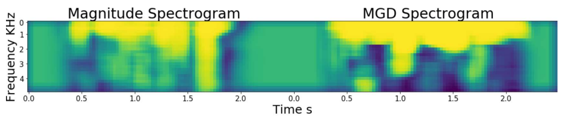


Figure 6.27: Attention map

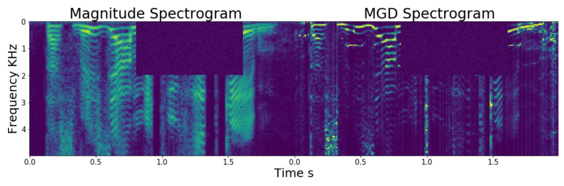


Figure 6.28: comb2 with corrupted sections

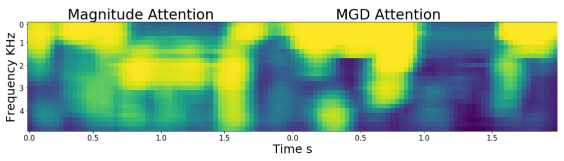


Figure 6.29: Attention map

There are silent sections in this speech clip. The silent sections are not important for classification of emotions. Figure 6.27 shows the attention map when the input is passed through the model tuned_att_noise _comb2. It can be seen that the attention mechanism has given less weights to the silent sections. Before training the models, the silent sections were removed from start and end of the speech data. Therefore, the model did not have an opportunity to experience silent sections. This might explain the model still paying some attention to silent sections.

Figure 6.28 shows another possible input to tuned_att_noise _comb2 model. In this input, some sections are hidden. These rectangular 'hidden' sections contains random noise. When this input is passed through the model, the generated attention map is shown in Figure 6.29. It can be seen that the model pays less attention to the the corrupted sections of the input. Furthermore, it can be seen that the MGD section does a better job of doing this. This gives evidence that the attention mechanism may be capable of handling data with missing/corrupted sections. For example, imagine a section of the input is corrupted with some noise. Certain noises are limited only to a certain range of frequencies and times as mentioned in Section 6.3.4. These types of noises might create artifacts which can be approximated by rectangular sections in Figure 6.28. Figure 6.29 provides evidence that attention mechanism may be able to handle situation similar to this.

6.3.8 Evaluation on the RAVDESS dataset

The model in which hyperparameters were fine tuned on the Berlin dataset was used to train and evaluate on the RAVDESS dataset. The training was performed with batches as described in section 6.3.5. A subset of emotions were selected as mentioned in section 6.2.1. Since we used 6 emotions, the FCNN architecture was modified to accommodate that. Therefore, the last convolution layer of the FCNN had 6 filters instead of 7. All other hyperparameters were kept unchanged. Figure 6.30 shows the mean performance of the model under all the different SNR levels, but under different noise types. From this diagram it can be seen that the model has a mean F1 score of 0.81. This value is the performance of the system under all SNR levels and noise types considered. It can also be seen that kettle noise has the lowest level of performance which is 0.68. Alarms and steps noises have the highest performance which is 0.88. This is similar to the results from Berlin data set as can be seen from Section 6.3.4. This provides evidence that disturbances like alarms and step noises can be handled easily with our SER solution. But noises like kettle whistle noise are harder to handle for these systems. Figure 6.31 shows the model F1 values under different SNR values. These values were averaged over all the different noise types. The model shows F1 values of 0.91 under clean speech. Figure 6.32 shows the same evaluation results in terms of accuracy. It shows an accuracy of 91% under clean speech. Figure 6.33 shows an emotion wise breakdown of model F1 values. Here the average F1 was taken over all the SNR levels and noise types. From Figure 6.33 it can be seen that sad is the lowest performing emotion while calm performs best. Although the types of emotions used in training Berlin dataset if different from RAVDESS, certain common emotions are present in both datasets. The performance on these emotions differ in the two models. This may be due to the difference in the training data, difference in the emotions and randomness involved in the training process. Therefore, the we can conclude that the performance of each individual emotion may depend on these conditions and should be taken into consideration when using these models for practical applications.

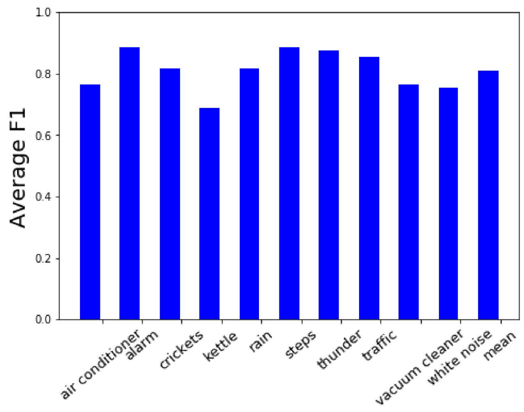


Figure 6.30: Per noise type performance on RAVDESS

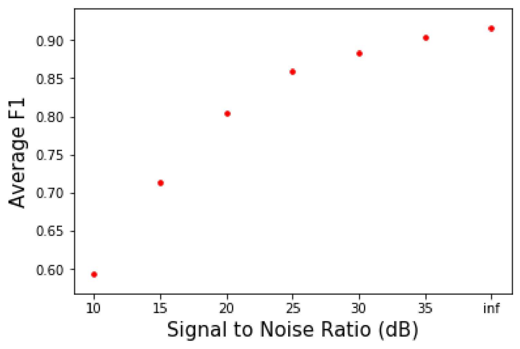


Figure 6.31: Average F1 on RAVDESS at various SNR levels

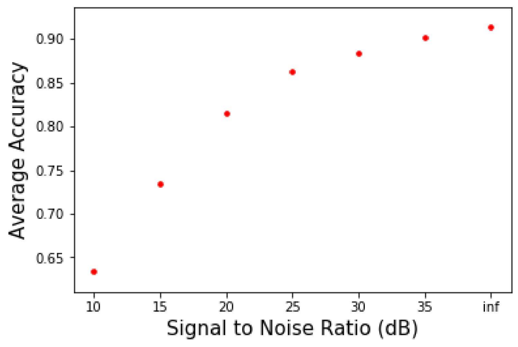


Figure 6.32: Average accuracy on RAVDESS at various SNR levels

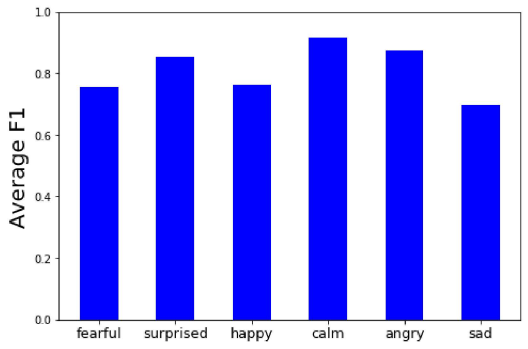


Figure 6.33: Per noise performance on RAVDESS

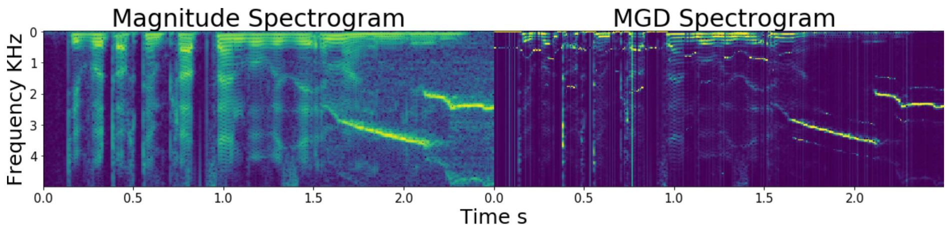


Figure 6.34: Speech with kettle noise

6.4 Discussion

From the results of this study it can be seen that the attention mechanism is capable of handling data with corrupted sections. But the training data consisted only of simple and artificial noise. If it contained other real noise types, the attention mechanism may have learned to deal with noise/corrupted data in a better manner. This hypothesis is yet to be tested.

The main motivation for using the attention mechanism is that it is capable of ignoring irrelevant sections of input and focus on the important sections. So, if the input consists of large sections of irrelevant data, the attention mechanism should pay less attention to this input. Thinking in this direction, it can be hypothesized that the attention layer by itself may be able to predict the uncertainty of the model. This has to be tested during future studies.

From figures 6.18 and 6.30 which explains the noise type performance of the models trained on Berlin and RAVDESS datasets, it can be seen that both of these models show their lowest performance under kettle noise. This may be evidence that SER systems created by the procedure we described are affected adversely by noises similar to the whistle sound of kettles. Figure 6.34 shows the kettle noise we used mixed with one of the speech samples from Berlin dataset. The time frequency characteristics of kettle noise are different from other noises in that the dominant frequency of the kettle noise shifts in time. But when our models were trained with noise, the artificial noises that were used only occupied a constant frequency ranges as can be seen from Figure 6.3. The low performance under kettle noise may be due to the fact that the models have not seen noises similar to this while training.

Referring to section 6.3.5 it can be seen that the performance of the model trained with batches decreased under white noise. This is in contrast to the performance under most of the other noise types where batch training improved performance over instance wise training. This may be due to certain characteristics of batch training. When training batch wise, the mean error it calculated per batch and then back propagated to update the weights. As explained in section 6.2.9 white noise was added to speech samples while training. Since white noise corrupts all the frequency ranges and time intervals in an equal manner, batch training might have cancelled the effect of white noise. Further analysis of this phenomena should be performed.

6.5 Conclusion

This study analyzes several ways to improve the performance of SER systems under noisy conditions. Magnitude spectrogram is the most common input feature for state-of-the-art CNN SER systems. But these are very sensitive to noise. We show that by combining magnitude spectrogram with modified group delay spectrogram as inputs to CNNs, the noise robustness of SER systems can be improved. Also it was observed that adding artificial noise during training can make models more robust to real world noises. Also, using CNN architecture characteristics like FCNN and attention mechanism can improve CNN based SER models. We first used the Berlin Database of Emotional Speech to find the noise robust features, training procedures, CNN architecture, and hyperparameters which perform well under noise. Then, we use these features, procedures, CNN architecture, and hyperparameters to train and evaluate a model on the speech section of the RAVDESS dataset.

Using the Berlin Database of Emotional Speech we showed that our final model with attention mechanism improved performance over other models considered. The 10 fold cross validation accuracy of the final model was 86% (F1 - 0.85) under clean speech and the average accuracy under all the noise types and signal to noise ratios considered was 76% (F1 - 0.73). The model trained on the speech section of RAVDESS dataset achieved an accuracy of 91% (F1 of 0.91) under clean speech and average accuracy of 82% (F1 of

0.81) under all signal to noise ratios and noise types considered. These results show that the noise robust features, training methods and the particular FCNN architecture with the attention mechanism that we obtained can indeed handle noisy data for the task of speech emotion recognition. Models trained and code used to produce results in this chapter can be found at https://github.com/sleekEagle/noise_emotion.git

CHAPTER 7

CONCLUSION

EMS providers face several challenges in their work which hiders their effectiveness when working. The emergency protocols that they must follow are very complicated and requires frequent training and skill for successful execution. Certain emergency protocols such as cardiopulmonary resuscitation (CPR) is such a protocol which require frequent training and also requires high-quality feedback during training. Accurate feedback must be provided on CPR compression depth and rate which are essential in providing high-quality CPR. Normal practice is for a human observer to provide feedback on the CPR procedure which can be sub-optimal. Work stress is another hindrance faced by EMS providers which can limit their effectiveness. An intelligent cognitive assistant can be utilized to solve these problems. To realize the vision of a cognitive assistant, several fundamental algorithmic contributions are still lacking. Through this thesis, several such fundamental algorithmic contributions were made which in the future can be combined into a capable cognitive assistant.

Depth perception is a critical component which can be used to measure distance to objects around the EMS provider. For example, it can be used to estimate the distance to the hands during CPR which is critical in measuring CPR depth. We introduced a novel technique based on defocus blur that can be used to measure depth to human hands using a single camera. All monocular depth estimation techniques are sensitive to the change of the camera. We introduce a novel technique to eliminate this sensitivity so our solution will be usable by a range of different cameras.

We introduced several techniques to estimate CPR compression depth and rate. To solve the problem of data scarcity on CPR performance evaluation, the first multi-modal dataset on CPR performance was collected. The ground truth CPR depth was obtained with a sensor implanted in a CPR manikin while the video was recorded with three different cameras. These cameras included one body-worn camera to obtain an ego-centric view and two external cameras. Microsoft Kinect camera was used to collect depth maps of the scene. Inertial measurement data was collected with a smartwatch on the wrist of the participant. The defocus-blur-based depth perception model was used to predict the depth of the hands of the participants while they were performing CPR. A smartwatch IMU-based model was developed to predict CPR depth and rate which achieved accurate results. This model could predict the CPR rate with an error of 5.4 compressions per minute and CPR compression depth with an error of 6.6 mm. The CPR compression rate could be calculated with video from an external camera with the error of 16 compressions per minute. The CPR compression depth could be estimated with the camera with an error of 25.6 mm.

A few shot learning methods were developed to teach models to recognize activities with very few samples. The methods include utilizing a special kind of loss called 'Center Loss' and batch normalization-based adaptation methods. All these techniques are useful in teaching a model with few examples. First, we experimented with several public datasets for IMU-based activity recognition to show the effectiveness of our methods which achieved the state-of-the-art performance. For example our model could recognize 7 activities on PAMAP2 dataset with around 60% accuracy under 1-shot setting while achieving 78% on 5-shot setting. On the OPP dataset the performance under 1-shot and 5-shot settings were 66% and 79% for detecting 7 classes in the target set. For the UTWENTE dataset 1-shot and 5-shot accuracies achieved 83% and 93%. We also show that the models we trained on these public datasets can be used to detect EMS activities including CPR with a high degree of accuracy.

We addressed the problem of noise robustness of vocal emotion recognition with several fronts. Regular practice is to use magnitude spectrogram as an input to the models in order to predict emotions. We showed that using a special type of spectrogram called Modified Group Delay (MGD) also to models improved there performance under noise by 15%. Training with data mixed with artificial noise was shown to improve the performance under noise for the vocal emotion recognition models. This may be due to injecting noise brings the noise-injected data samples closer to the real world noisy samples. Adding synthetic noise to the data samples improved the accuracy of the models on average by 10% as evaluated on public datasets.

In addition to the above two methods, using attention mechanism in the model can also improve the performance under noise in the models. This is due to the attention mechanism being able for focus on regions of the input that are not corrupted by noise. Adding an attention layer improved the performance of our model under noise by 5%. Our model showed an average accuracy of 76% on the Berlin Database of Emotional speech. This is a significant improvement over the previous state-of-the-art model which had an accuracy of 56%. We also experimented on the RAVDESS dataset where our model performed at 81% accuracy under noisy conditions.

Our improvements to the state-of-the-art in the above-mentioned directions alleviated several problems related to the usage of an intelligent cognitive assistant that can help EMS providers which has the potential to solve their problems that were mentioned earlier.

Still, there are many avenues for improvement in these fields. Our camera-based CPR quality estimation methods did not perform as well as the smartwatch-based models. This is due to inaccuracies in in-depth perception and tracking points on the hands. There are plenty of future works in high accurate depth estimation and highly accurate point tracking on the hand. The defocus calibration method that we invented was tested on artificially defocus-blurred images. Although the algorithms we used to create the artificial defocus blurring on the images are quite realistic, more extensive testing must be done on real datasets. Our few shot learning algorithms must be tested more thoroughly on more EMS-related data. The methods that we used to build a noise-robust emotion recognition model must be evaluated on EMS related emotional dataset. Furthermore, it must be researched to see if the same methods are applicable in the closely related area of speech recognition that can also be used to understand the stress level of EMS responders. Finally, future work includes integrating the developed sets of solutions in to a fully functional cognitive assistant

Acknowledgment: This work is supported by the award 70NANB21H029 from the U.S. Department of Commerce, National Institute of Standards and Technology (NIST).

BIBLIOGRAPHY

[1] T. McCallion, "And the survey says: NASEMSO analysis provides snapshot of EMS industry," *JEMS: a journal of emergency medical services*, vol. 37, no. 1, pp. 34–35, 2012.

[2] H. V. Duong *et al.*, "National characteristics of emergency medical services responses for older adults in the United States," *Prehospital emergency care*, vol. 22, no. 1, pp. 7–14, 2018.

[3] Charles D. Feiring BS, "2018 ODEMSA Protocols," OLD DOMINION EMERGENCY MEDICAL SERVICES ALLIANCE, Richmond, VA, Tech. Rep. [Online]. Available: `https://odemsa.net/regional-documents/`.

[4] P. D. Patterson, M. D. Weaver, and D. Hostler, "EMS provider wellness," *Emergency medical services: clinical practice and systems oversight*, pp. 211–216, 2015.

[5] A. Afshari, S. R. Borzou, F. Shamsaei, E. Mohammadi, and L. Tapak, "Perceived occupational stressors among emergency medical service providers: a qualitative study," *BMC Emergency Medicine*, vol. 21, pp. 1–8, 2021.

[6] R. Bounds, "Factors affecting perceived stress in pre-hospital emergency medical services," *Californian Journal of Health Promotion*, vol. 4, no. 2, pp. 113–131, 2006, ISSN: 1545-8725.

[7] R. K. Cydulka, C. L. Emerman, B. Shade, and J. Kubincanek, "Stress levels in EMS personnel: a national survey," *Prehospital and disaster medicine*, vol. 12, no. 2, pp. 65–69, 1997, ISSN: 1945-1938.

[8] C. E. Izard, *Human emotions*. Springer Science & Business Media, 2013, ISBN: 1489922091.

[9] R. K. Cydulka, C. L. Emerman, B. Shade, and J. Kubincanek, "Stress levels in EMS personnel: a longitudinal study with work-schedule modification," *Academic Emergency Medicine*, vol. 1, no. 3, pp. 240–246, 1994, ISSN: 1069-6563.

[10] U Ayala *et al.*, "Automatic detection of chest compressions for the assessment of CPR-quality parameters," *Resuscitation*, vol. 85, no. 7, pp. 957–963, 2014.

[11] N. W. Mick and R. J. Williams, "Pediatric cardiac arrest resuscitation," *Emergency Medicine Clinics*, vol. 38, no. 4, pp. 819–839, 2020, ISSN: 0733-8627.

[12] S. O. Aase and H. Myklebust, "Compression depth estimation for CPR quality assessment using DSP on accelerometer signals," *IEEE Transactions on Biomedical Engineering*, vol. 49, no. 3, pp. 263–268, 2002.

[13] G. Rong, A. Mendez, E. B. Assi, B. Zhao, and M. Sawan, "Artificial intelligence in healthcare: review and prediction case studies," *Engineering*, vol. 6, no. 3, pp. 291–301, 2020, ISSN: 2095-8099.

[14] M. K. Wolters, F. Kelly, and J. Kilgour, "Designing a spoken dialogue interface to an intelligent cognitive assistant for people with dementia," *Health informatics journal*, vol. 22, no. 4, pp. 854–866, 2016, ISSN: 1460-4582.

[15] S. Kernan Freire *et al.*, "Lessons learned from designing and evaluating CLAICA: a continuously learning ai cognitive assistant," in *Proceedings of the 28th International Conference on Intelligent User Interfaces*, 2023, pp. 553–568.

[16] C. Angulo, A. Chacón, and P. Ponsa, "Towards a cognitive assistant supporting human operators in the Artificial Intelligence of Things," *Internet of Things*, vol. 21, p. 100 673, 2023, ISSN: 2542-6605.

[17] H. Alemzadeh and M. Devarakonda, "An NLP-based cognitive system for disease status identification in electronic health records," in *2017 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI)*, IEEE, 2017, pp. 89–92, ISBN: 1509041796.

[18] S. Kim, W. Guo, R. Williams, J. Stankovic, and H. Alemzadeh, "Information Extraction from Patient Care Reports for Intelligent Emergency Medical Services," in *2021 IEEE/ACM Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE)*, IEEE, 2021, pp. 58–69, ISBN: 1665439653.

[19] S. Preum, S. Shu, M. Hotaki, R. Williams, J. Stankovic, and H. Alemzadeh, "CognitiveEMS: A cognitive assistant system for emergency medical services," *ACM SIGBED Review*, vol. 16, no. 2, pp. 51–60, 2019, ISSN: 1551-3688.

[20] M. A. Rahman, S. M. Preum, R. Williams, H. Alemzadeh, and J. A. Stankovic, "GRACE: generating summary reports automatically for cognitive assistance in emergency response," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 13 356–13 362.

[21] M. A. Rahman *et al.*, "emsReACT: A Real-Time Interactive Cognitive Assistant for Cardiac Arrest Training in Emergency Medical Services," in *2023 19th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)*, IEEE, 2023, pp. 120–128, ISBN: 9798350346497.

[22] S. M. Preum, S. Shu, H. Alemzadeh, and J. A. Stankovic, "Emscontext: EMS protocol-driven concept extraction for cognitive assistance in emergency response," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 13 350–13 355.

[23] S. M. Preum *et al.*, "Towards a cognitive assistant system for emergency response," in *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, IEEE, 2018, pp. 347–348, ISBN: 153865301X.

[24] S. Shu, S. Preum, H. M. Pitchford, R. D. Williams, J. Stankovic, and H. Alemzadeh, "A behavior tree cognitive assistant system for emergency medical services," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2019, pp. 6188–6195, ISBN: 1728140048.

[25] K. Weerasinghe *et al.*, "Real-Time Multimodal Cognitive Assistant for Emergency Medical Services," *arXiv preprint arXiv:2403.06734*, 2024.

[26] S. Mekruksavanich, A. Jitpattanakul, P. Youplao, and P. Yupapin, "Enhanced hand-oriented activity recognition based on smartwatch sensor data using lstms," *Symmetry*, vol. 12, no. 9, p. 1570, 2020, ISSN: 2073-8994.

[27] C. Lins, B. Friedrich, A. Hein, and S. Fudickar, "An evolutionary approach to continuously estimate CPR quality parameters from a wrist-worn inertial sensor," *Health and Technology*, vol. 12, no. 1, pp. 161–173, 2022, ISSN: 2190-7188.

[28] M. Y. Ahmed, Z. Chen, E. Fass, and J. Stankovic, "Real time distant speech emotion recognition in indoor environments," in *Proceedings of the 14th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, 2017, pp. 215–224.

[29] S. Gur and L. Wolf, "Single image depth estimation trained via depth from defocus cues," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7683–7692.

[30] M. Watanabe, S. K. Nayar, and M. N. Noguchi, "Real-time computation of depth from defocus," in *Proc.SPIE*, vol. 2599, Jan. 1996, pp. 14–25. DOI: 10.1117/12.230388. [Online]. Available: https://doi.org/10.1117/12.230388.

[31] M. Carvalho, B. Le Saux, P. Trouvé-Peloux, A. Almansa, and F. Champagnat, "Deep Depth from Defocus: how can defocus blur improve 3D estimation using dense neural networks?" In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 2018, p. 0.

[32] L. Wijayasingha, H. Alemzadeh, and J. A. Stankovic, "Camera-Independent Single Image Depth Estimation from Defocus Blur," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2024, pp. 3749–3758.

[33] L. Wijayasingha and J. A. Stankovic, "Generalized Few-Shot Learning For Wearable Sensor-based Human Activity Recognition," in *2022 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, IEEE, 2022, pp. 328–334, ISBN: 1665416475.

[34] L. Wijayasingha and J. A. Stankovic, "Robustness to noise for speech emotion classification using CNNs and attention mechanisms," *Smart Health*, vol. 19, p. 100 165, 2021, ISSN: 2352-6483.

[35] J. Ping, Y. Liu, and D. Weng, "Comparison in depth perception between virtual reality and augmented reality systems," in *2019 IEEE conference on virtual reality and 3d user interfaces (VR)*, IEEE, 2019, pp. 1124–1125, ISBN: 1728113776.

[36] X. Dong, M. A. Garratt, S. G. Anavatti, and H. A. Abbass, "Towards real-time monocular depth estimation for robotics: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 10, pp. 16 940–16 961, 2022, ISSN: 1524-9050.

[37] N.-H. Wang *et al.*, "Bridging unsupervised and supervised depth from focus via all-in-focus supervision," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 12 621–12 631.

[38] M. Maximov, K. Galim, and L. Leal-Taixé, "Focus on defocus: bridging the synthetic to real domain gap for depth estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1071–1080.

[39] V. Casser, S. Pirk, R. Mahjourian, and A. Angelova, "Depth prediction without the sensors: Leveraging structure for unsupervised learning from monocular videos," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, 2019, pp. 8001–8008, ISBN: 2374-3468.

[40] F. Yang, X. Huang, and Z. Zhou, "Deep Depth from Focus with Differential Focus Volume," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 642–12 651.

[41] Y. Ban *et al.*, "Depth estimation method for monocular camera defocus images in microscopic scenes," *Electronics*, vol. 11, no. 13, p. 2012, 2022, ISSN: 2079-9292.

[42] X. Zhang *et al.*, "Particle field positioning with a commercial microscope based on a developed CNN and the depth-from-defocus method," *Optics and Lasers in Engineering*, vol. 153, p. 106 989, 2022, ISSN: 0143-8166.

[43] T. Nagata *et al.*, "Depth perception from image defocus in a jumping spider," *Science*, vol. 335, no. 6067, pp. 469–471, 2012, ISSN: 0036-8075.

[44] C. Kong and S. Lucey, "Deep non-rigid structure from motion," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1558–1567.

[45] N.-H. Wang, B. Solarte, Y.-H. Tsai, W.-C. Chiu, and M. Sun, "360sd-net: 360 stereo depth estimation with learnable cost volume," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 582–588, ISBN: 1728173957.

[46] Y.-L. Chang, W.-Y. Chen, J.-Y. Chang, Y.-M. Tsai, C.-L. Lee, and L.-G. Chen, "Priority depth fusion for the 2D to 3D conversion system," in *Three-Dimensional Image Capture and Applications 2008*, vol. 6805, SPIE, 2008, pp. 320–327.

[47] Y. Y. Schechner and N. Kiryati, "Depth from Defocus vs. Stereo: How Different Really Are They?" *International Journal of Computer Vision*, vol. 39, no. 2, pp. 141–162, 2000, ISSN: 1573-1405. DOI: `10.1023/A:1008175127327`. [Online]. Available: `https://doi.org/10.1023/A:1008175127327`.

[48] Y. Li, Y. Guo, Z. Yan, X. Huang, Y. Duan, and L. Ren, "Omnifusion: 360 monocular depth estimation via geometry-aware fusion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 2801–2810.

[49] A. Mertan, D. J. Duff, and G. Unal, "Single image depth estimation: An overview," *Digital Signal Processing*, p. 103 441, 2022, ISSN: 1051-2004.

[50] V. Patil, C. Sakaridis, A. Liniger, and L. Van Gool, "P3depth: Monocular depth estimation with a piecewise planarity prior," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 1610–1621.

[51] S. F. Bhat, R. Birkl, D. Wofk, P. Wonka, and M. Müller, "Zoedepth: Zero-shot transfer by combining relative and metric depth," *arXiv preprint arXiv:2302.12288*, 2023.

[52] W. Zhao, Y. Rao, Z. Liu, B. Liu, J. Zhou, and J. Lu, "Unleashing text-to-image diffusion models for visual perception," *arXiv preprint arXiv:2303.02153*, 2023.

[53] K. Xian, J. Zhang, O. Wang, L. Mai, Z. Lin, and Z. Cao, "Structure-guided ranking loss for single image depth prediction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 611–620.

[54] S. K. Nayar and Y Nakagawa, "Shape from focus," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 8, pp. 824–831, 1994, ISSN: 1939-3539. DOI: `10.1109/34.308479`.

[55] M Subbarao and J. K. Tyan, "Selecting the optimal focus measure for autofocusing and depth-from-focus," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 864–870, 1998, ISSN: 1939-3539. DOI: 10.1109/34.709612.

[56] C. Hazirbas, S. G. Soyer, M. C. Staab, L. Leal-Taixé, and D. Cremers, "Deep depth from focus," in *Computer Vision–ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, December 2–6, 2018, Revised Selected Papers, Part III 14*, Springer, 2019, pp. 525–541, ISBN: 3030208923.

[57] S. Liu, F. Zhou, and Q. Liao, "Defocus map estimation from a single image based on two-parameter defocus model," *IEEE Transactions on Image Processing*, vol. 25, no. 12, pp. 5943–5956, 2016, ISSN: 1057-7149.

[58] S. Anwar, Z. Hayder, and F. Porikli, "Depth Estimation and Blur Removal from a Single Out-of-focus Image.," in *BMVC*, vol. 1, 2017, p. 2.

[59] M. Subbarao and G. Surya, "Depth from defocus: A spatial domain approach," *International Journal of Computer Vision*, vol. 13, no. 3, pp. 271–294, 1994, ISSN: 0920-5691.

[60] A. Zhang and J. Sun, "Joint Depth and Defocus Estimation From a Single Image Using Physical Consistency," *IEEE Transactions on Image Processing*, vol. 30, pp. 3419–3433, 2021.

[61] S Pertuz, D Puig, M. A. Garcia, and A Fusiello, "Generation of All-in-Focus Images by Noise-Robust Selective Fusion of Limited Depth-of-Field Images," *IEEE Transactions on Image Processing*, vol. 22, no. 3, pp. 1242–1251, 2013, ISSN: 1941-0042. DOI: 10.1109/TIP.2012.2231087.

[62] H. Ikoma, C. M. Nguyen, C. A. Metzler, Y. Peng, and G. Wetzstein, "Depth from defocus with learned optics for imaging and occlusion-aware depth estimation," in *2021 IEEE International Conference on Computational Photography (ICCP)*, IEEE, 2021, pp. 1–12, ISBN: 1665419520.

[63] R. Ng, M. Levoy, M. Brédif, G. Duval, M. Horowitz, and P. Hanrahan, "Light field photography with a hand-held plenoptic camera," Stanford university, Tech. Rep., 2005.

[64] Y. Lu, G. Milliron, J. Slagter, and G. Lu, "Self-supervised single-image depth estimation from focus and defocus clues," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 6281–6288, 2021, ISSN: 2377-3766.

[65] Y.-W. Tai and M. S. Brown, "Single image defocus map estimation using local contrast prior," in *2009 16th IEEE International Conference on Image Processing (ICIP)*, IEEE, 2009, pp. 1797–1800, ISBN: 1424456541.

[66] S. Zhuo and T. Sim, "Defocus map estimation from a single image," *Pattern Recognition*, vol. 44, no. 9, pp. 1852–1858, 2011, ISSN: 0031-3203.

[67] X. Cun and C.-M. Pun, "Defocus blur detection via depth distillation," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIII 16*, Springer, 2020, pp. 747–763, ISBN: 3030586006.

[68] O. D. Lara and M. A. Labrador, "A Survey on Human Activity Recognition using Wearable Sensors," *IEEE Communications Surveys Tutorials*, vol. 15, no. 3, pp. 1192–1209, 2013. DOI: 10.1109/SURV.2012.110112.00192.

[69] M. Vrigkas, C. Nikou, and I. A. Kakadiaris, "A Review of Human Activity Recognition Methods," *Frontiers in Robotics and AI*, vol. 2, p. 28, 2015, ISSN: 2296-9144. DOI: 10.3389/frobt.2015.00028. [Online]. Available: https://www.frontiersin.org/article/10.3389/frobt.2015.00028.

[70] S. Deng, W. Hua, B. Wang, G. Wang, and X. Zhou, "Few-shot human activity recognition on noisy wearable sensor data," in *International Conference on Database Systems for Advanced Applications*, Springer, 2020, pp. 54–72.

[71] M. M. Hassan, M. Z. Uddin, A. Mohamed, and A. Almogren, "A robust human activity recognition system using smartphone sensors and deep learning," *Future Generation Computer Systems*, vol. 81, pp. 307–313, 2018, ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2017.11.029. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X17317351.

[72] F. Moya Rueda, R. Grzeszick, G. A. Fink, S. Feldhorst, and M. Ten Hompel, "Convolutional Neural Networks for Human Activity Recognition Using Body-Worn Sensors," *Informatics*, vol. 5, no. 2, 2018, ISSN: 2227-9709. DOI: 10.3390/informatics5020026. [Online]. Available: https://www.mdpi.com/2227-9709/5/2/26.

[73] S.-M. Lee, S. M. Yoon, and H. Cho, “Human activity recognition from accelerometer data using Convolutional Neural Network,” in *2017 ieee international conference on big data and smart computing (bigcomp)*, IEEE, 2017, pp. 131–134, ISBN: 1509030158.

[74] N. Y. Hammerla, S. Halloran, and T. Plötz, “Deep, convolutional, and recurrent models for human activity recognition using wearables,” *arXiv preprint arXiv:1604.08880*, 2016.

[75] R. Yao, G. Lin, Q. Shi, and D. C. Ranasinghe, “Efficient dense labelling of human activity sequences from wearables using fully convolutional networks,” *Pattern Recognition*, vol. 78, pp. 252–266, 2018, ISSN: 0031-3203. DOI: `https://doi.org/10.1016/j.patcog.2017.12.024`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S0031320317305204`.

[76] M. Kim, C. Y. Jeong, and H. C. Shin, “Activity Recognition using Fully Convolutional Network from Smartphone Accelerometer,” in *2018 International Conference on Information and Communication Technology Convergence (ICTC)*, 2018, pp. 1482–1484. DOI: `10.1109/ICTC.2018.8539419`.

[77] Y. Guan and T. Plötz, “Ensembles of Deep LSTM Learners for Activity Recognition Using Wearables,” *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 1, no. 2, Jun. 2017. DOI: `10.1145/3090076`. [Online]. Available: `https://doi.org/10.1145/3090076`.

[78] M. Fink, “Object classification from a single example utilizing class relevance metrics,” *Advances in neural information processing systems*, vol. 17, pp. 449–456, 2005.

[79] S. Feng and M. F. Duarte, “Few-shot learning-based human activity recognition,” *Expert Systems with Applications*, vol. 138, p. 112 782, 2019, ISSN: 0957-4174.

[80] J. Snell, K. Swersky, and R. S. Zemel, “Prototypical networks for few-shot learning,” *arXiv preprint arXiv:1703.05175*, 2017.

[81] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, “Generalizing from a few examples: A survey on few-shot learning,” *ACM Computing Surveys (CSUR)*, vol. 53, no. 3, pp. 1–34, 2020, ISSN: 0360-0300.

[82] K. Lee, S. Maji, A. Ravichandran, and S. Soatto, “Meta-learning with differentiable convex optimization,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 657–10 665.

[83] E. Schonfeld, S. Ebrahimi, S. Sinha, T. Darrell, and Z. Akata, “Generalized zero-and few-shot learning via aligned variational autoencoders,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8247–8255.

[84] S. Rahman, S. Khan, and F. Porikli, “A unified approach for conventional zero-shot, generalized zero-shot, and few-shot learning,” *IEEE Transactions on Image Processing*, vol. 27, no. 11, pp. 5652–5667, 2018.

[85] R. Socher, M. Ganjoo, C. D. Manning, and A. Ng, “Zero-Shot Learning Through Cross-Modal Transfer,” in *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L Bottou, M Welling, Z Ghahramani, and K. Q. Weinberger, Eds., Curran Associates, Inc., 2013, pp. 935–943. [Online]. Available: `http://papers.nips.cc/paper/5027-zero-shot-learning-through-cross-modal-transfer.pdf`.

[86] C. Finn, P. Abbeel, and S. Levine, “Model-agnostic meta-learning for fast adaptation of deep networks,” in *International conference on machine learning*, 2017, pp. 1126–1135.

[87] M. Andrychowicz *et al.*, “Learning to learn by gradient descent by gradient descent,” *Advances in neural information processing systems*, vol. 29, 2016.

[88] A. Nichol and J. Schulman, “Reptile: a scalable metalearning algorithm,” *arXiv preprint arXiv:1803.02999*, vol. 2, no. 3, p. 4, 2018.

[89] Z. Li, F. Zhou, F. Chen, and H. Li, “Meta-SGD: Learning to Learn Quickly for Few Shot Learning,” *ArXiv*, vol. abs/1707.09835, 2017. [Online]. Available: `https://api.semanticscholar.org/CorpusID:25316837`.

[90] A. A. Rusu *et al.*, “Meta-learning with latent embedding optimization,” *arXiv preprint arXiv:1807.05960*, 2018.

[91] J. Snell, K. Swersky, and R. S. Zemel, “Prototypical Networks for Few-shot Learning,” in *Neural Information Processing Systems*, 2017. [Online]. Available: `https://api.semanticscholar.org/CorpusID:309759`.

[92] O. Vinyals, C. Blundell, T. Lillicrap, and D. Wierstra, “Matching networks for one shot learning,” *Advances in neural information processing systems*, vol. 29, 2016.

[93] G. Koch, R. Zemel, and R. Salakhutdinov, “Siamese neural networks for one-shot image recognition,” in *ICML deep learning workshop*, vol. 2, Lille, 2015.

[94] E. Hoffer and N. Ailon, “Deep metric learning using triplet network,” in *Similarity-Based Pattern Recognition: Third International Workshop, SIMBAD 2015, Copenhagen, Denmark, October 12-14, 2015. Proceedings 3*, Springer, 2015, pp. 84–92, ISBN: 3319242601.

[95] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. S. Torr, and T. M. Hospedales, “Learning to compare: Relation network for few-shot learning,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1199–1208.

[96] B. Oreshkin, P. Rodríguez López, and A. Lacoste, “Tadam: Task dependent adaptive metric for improved few-shot learning,” *Advances in neural information processing systems*, vol. 31, 2018.

[97] V. Garcia and J. Bruna, “Few-shot learning with graph neural networks,” *arXiv preprint arXiv:1711.04043*, 2017.

[98] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel, “Rl $ˆ 2$: Fast reinforcement learning via slow reinforcement learning,” *arXiv preprint arXiv:1611.02779*, 2016.

[99] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel, “A simple neural attentive meta-learner,” *arXiv preprint arXiv:1707.03141*, 2017.

[100] T. Munkhdalai and H. Yu, “Meta networks,” in *International conference on machine learning*, PMLR, 2017, pp. 2554–2563, ISBN: 2640-3498.

[101] H. Edwards and A. Storkey, “Towards a neural statistician,” *arXiv preprint arXiv:1606.02185*, 2016.

[102] E. Grant, C. Finn, S. Levine, T. Darrell, and T. Griffiths, “Recasting gradient-based meta-learning as hierarchical bayes,” *arXiv preprint arXiv:1801.08930*, 2018.

[103] C. Finn, K. Xu, and S. Levine, “Probabilistic model-agnostic meta-learning,” *Advances in neural information processing systems*, vol. 31, 2018.

[104] M. Patacchiola, J. Turner, E. J. Crowley, M. O’Boyle, and A. J. Storkey, “Bayesian meta-learning for the few-shot setting via deep kernels,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 16 108–16 118, 2020.

[105] Y. Chen, Z. Liu, H. Xu, T. Darrell, and X. Wang, “Meta-baseline: Exploring simple meta-learning for few-shot learning,” in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 9062–9071.

[106] W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C. F. Wang, and J.-B. Huang, “A closer look at few-shot classification,” *arXiv preprint arXiv:1904.04232*, 2019.

[107] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How transferable are features in deep neural networks?” *Advances in neural information processing systems*, vol. 27, 2014.

[108] M. Huh, P. Agrawal, and A. A. Efros, “What makes ImageNet good for transfer learning?” *arXiv preprint arXiv:1608.08614*, 2016.

[109] S. Kornblith, J. Shlens, and Q. V. Le, “Do better imagenet models transfer better?” In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 2661–2671.

[110] A. Gupta, K. Thadani, and N. OHare, “Effective Few-Shot Classification with Transfer Learning,” in *Proceedings of the 28th International Conference on Computational Linguistics*, 2020.

[111] Y. Hu, V. Gripon, and S. Pateux, “Leveraging the Feature Distribution in Transfer-Based Few-Shot Learning,” in *Lecture Notes in Computer Science*, Springer International Publishing, 2021, pp. 487–499.

[112] Yunhui Guo, N. Codella, Leonid Karlinsky, John R. Smith, T. Simunic, and R. Feris, “A New Benchmark for Evaluation of Cross-Domain Few-Shot Learning,” *arXiv.org*, 2019.

[113] Y. Liu *et al.*, “Learning to propagate labels: Transductive propagation network for few-shot learning,” *arXiv preprint arXiv:1805.10002*, 2018.

[114] Carlos Medina, A. Devos, and M. Grossglauser, “Self-Supervised Prototypical Transfer Learning for Few-Shot Classification,” *International Conference on Machine Learning*, 2020.

[115] Vincent Dumoulin *et al.*, “Comparing Transfer and Meta Learning Approaches on a Unified Few-Shot Classification Benchmark,” *arXiv.org*, 2021.

[116] Q. Sun, Y. Liu, T.-S. Chua, and B. Schiele, “Meta-Transfer Learning for Few-Shot Learning,” in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Apr. 2019.

[117] A. Chowdhury, M. Jiang, S. Chaudhuri, and C. Jermaine, “Few-shot image classification: Just use a library of pre-trained feature extractors and a simple classifier,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 9445–9454.

[118] A. Kolesnikov *et al.*, “Big transfer (bit): General visual representation learning,” in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16*, Springer, 2020, pp. 491–507, ISBN: 3030585573.

[119] Z. Shen, Z. Liu, J. Qin, M. Savvides, and K.-T. Cheng, “Partial Is Better Than All: Revisiting Fine-tuning Strategy for Few-shot Learning,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 11, pp. 9594–9602, May 2021.

[120] C. P. Phoo and B. Hariharan, “Self-training for few-shot transfer across extreme task differences,” *arXiv preprint arXiv:2010.07734*, 2020.

[121] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A simple framework for contrastive learning of visual representations,” in *International conference on machine learning*, PMLR, 2020, pp. 1597–1607, ISBN: 2640-3498.

[122] H. Yao *et al.*, “Graph few-shot learning via knowledge transfer,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, 2020, pp. 6656–6663, ISBN: 2374-3468.

[123] M. Chen *et al.*, “Improving in-context few-shot learning via self-supervised training,” *arXiv preprint arXiv:2205.01703*, 2022.

[124] Z. Chen, S. Maji, and E. Learned-Miller, “Shot in the dark: Few-shot learning with no base-class labels,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 2668–2677.

[125] H. Li, D. Eigen, S. Dodge, M. Zeiler, and X. Wang, “Finding task-relevant features for few-shot learning by category traversal,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 1–10.

[126] J. Liang, R. He, and T.-P. Tan, “A Comprehensive Survey on Test-Time Adaptation under Distribution Shifts,” *ArXiv*, vol. abs/2303.15361, 2023. [Online]. Available: `https://api.semanticscholar.org/CorpusID:257767040`.

[127] Y. Hu, V. Gripon, and S. Pateux, “Leveraging the Feature Distribution in Transfer-based Few-Shot Learning,” in *International Conference on Artificial Neural Networks*, 2020. [Online]. Available: `https://api.semanticscholar.org/CorpusID:219531491`.

[128] D. Wang, E. Shelhamer, S. Liu, B. Olshausen, and T. Darrell, “Tent: Fully test-time adaptation by entropy minimization,” *arXiv preprint arXiv:2006.10726*, 2020.

[129] H. Lim, B. Kim, J. Choo, and S. Choi, “TTN: A Domain-Shift Aware Batch Normalization in Test-Time Adaptation,” *ArXiv*, vol. abs/2302.05155, 2023. [Online]. Available: `https://api.semanticscholar.org/CorpusID:256808572`.

[130] L. Yuan, B. Xie, and S. Li, “Robust Test-Time Adaptation in Dynamic Scenarios,” *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15 922–15 932, 2023. [Online]. Available: `https://api.semanticscholar.org/CorpusID:257757140`.

[131] M. Segu, B. Schiele, and F. Yu, “Darth: Holistic test-time adaptation for multiple object tracking,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 9717–9727.

[132] S. Niu *et al.*, “Towards Stable Test-Time Adaptation in Dynamic Wild World,” *ArXiv*, vol. abs/2302.12400, 2023. [Online]. Available: `https://api.semanticscholar.org/CorpusID:257206115`.

[133] D. Chen, D. Wang, T. Darrell, and S. Ebrahimi, “Contrastive Test-Time Adaptation,” *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 295–305, 2022. [Online]. Available: `https://api.semanticscholar.org/CorpusID:248366600`.

[134] E. Bennequin, V. Bouvier, M. Tami, A. Toubhans, and C. Hudelot, “Bridging Few-Shot Learning and Adaptation: New Challenges of Support-Query Shift,” in *ECML/PKDD*, 2021. [Online]. Available: `https://api.semanticscholar.org/CorpusID:235186995`.

[135] S. Stansfield, D. Shawver, A. Sobel, M. Prasad, and L. Tapia, "Design and implementation of a virtual reality system and its application to training medical first responders," *Presence: Teleoperators & Virtual Environments*, vol. 9, no. 6, pp. 524–556, 2000.

[136] G. Koutitas, S. Smith, and G. Lawrence, "Performance evaluation of AR/VR training technologies for EMS first responders," *Virtual Reality*, vol. 25, no. 1, pp. 83–94, 2021.

[137] W. Wilkerson, D. Avstreih, L. Gruppen, K.-P. Beier, and J. Woolliscroft, "Using immersive simulation for training first responders for mass casualty incidents," *Academic emergency medicine*, vol. 15, no. 11, pp. 1152–1159, 2008.

[138] M. A. Rahman *et al.*, "IMACS-an interactive cognitive assistant module for cardiac arrest cases in emergency medical service: demo abstract," in *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*, 2020, pp. 621–622.

[139] G. Koutitas *et al.*, "A virtual and augmented reality platform for the training of first responders of the ambulance bus," in *Proceedings of the 12th ACM International Conference on Pervasive Technologies Related to Assistive Environments*, 2019, pp. 299–302.

[140] A. Jones, Y. Lin, A. Nettel-Aguirre, E. Gilfoyle, and A. Cheng, "Visual assessment of CPR quality during pediatric cardiac arrest: does point of view matter?" *Resuscitation*, vol. 90, pp. 50–55, 2015.

[141] A. Cheng *et al.*, "Perception of CPR quality: influence of CPR feedback, just-in-time CPR training and provider role," *Resuscitation*, vol. 87, pp. 44–50, 2015, ISSN: 0300-9572.

[142] A. Cheng *et al.*, "Variability in quality of chest compressions provided during simulated cardiac arrest across nine pediatric institutions," *Resuscitation*, vol. 97, pp. 13–19, 2015, ISSN: 0300-9572.

[143] J. Webber, K. Moran, and D. Cumin, "Paediatric cardiopulmonary resuscitation: Knowledge and perceptions of surf lifeguards," *Journal of Paediatrics and Child Health*, vol. 55, no. 2, pp. 156–161, 2019, ISSN: 1034-4810.

[144] C.-Y. Lin, S.-H. Hsia, E.-P. Lee, O.-W. Chan, J.-J. Lin, and H.-P. Wu, "Effect of audiovisual cardiopulmonary resuscitation feedback device on improving chest compression quality," *Scientific Reports*, vol. 10, no. 1, p. 398, 2020, ISSN: 2045-2322.

[145] A. Cheng *et al.*, "Improving cardiopulmonary resuscitation with a CPR feedback device and refresher simulations (CPR CARES Study): a randomized clinical trial," *JAMA pediatrics*, vol. 169, no. 2, pp. 137–144, 2015, ISSN: 2168-6203.

[146] T. Eftestøl *et al.*, "A probabilistic function to model the relationship between quality of chest compressions and the physiological response for patients in cardiac arrest," in *2020 Computing in Cardiology*, IEEE, 2020, pp. 1–4, ISBN: 1728173825.

[147] T.-C. Lu *et al.*, "Using a smartwatch with real-time feedback improves the delivery of high-quality cardiopulmonary resuscitation by healthcare professionals," *Resuscitation*, vol. 140, pp. 16–22, 2019, ISSN: 0300-9572.

[148] Laerdal, *Get Familiar with CPRcard*.

[149] A. E. White *et al.*, "Measuring the effectiveness of a novel CPRcard™ feedback device during simulated chest compressions by non-healthcare workers," *Singapore medical journal*, vol. 58, no. 7, p. 438, 2017.

[150] S. Tanaka *et al.*, "Comparison of quality of chest compressions during training of laypersons using Push Heart and Little Anne manikins using blinded CPRcards," *International Journal of Emergency Medicine*, vol. 10, pp. 1–7, 2017, ISSN: 1865-1372.

[151] D. M. González-Otero *et al.*, "Chest compression rate feedback based on transthoracic impedance," *Resuscitation*, vol. 93, pp. 82–88, 2015, ISSN: 0300-9572.

[152] T.-C. Lu *et al.*, "A novel depth estimation algorithm of chest compression for feedback of high-quality cardiopulmonary resuscitation based on a smartwatch," *Journal of biomedical informatics*, vol. 87, pp. 60–65, 2018, ISSN: 1532-0464.

[153] J. Zhao, X. Mao, and L. Chen, "Speech emotion recognition using deep 1D & 2D CNN LSTM networks," *Biomedical Signal Processing and Control*, vol. 47, pp. 312–323, 2019, ISSN: 17468108. DOI: 10.1016/j.bspc.2018.08.035.

[154] W. Lim, D. Jang, and T. Lee, “Speech emotion recognition using convolutional and Recurrent Neural Networks,” in *2016 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference, APSIPA 2016*, IEEE, 2017, pp. 1–4, ISBN: 9789881476821. DOI: 10.1109/APSIPA.2016.7820699.

[155] L. Wyse, “Audio Spectrogram Representations for Processing with Convolutional Neural Networks,” *arXiv preprint arXiv:1706.09559*, 2017. [Online]. Available: http://arxiv.org/abs/1706.09559.

[156] A. M. Badshah, J. Ahmad, N. Rahim, and S. W. Baik, “Speech Emotion Recognition from Spectrograms with Deep Convolutional Neural Network,” in *2017 International Conference on Platform Technology and Service, PlatCon 2017 - Proceedings*, IEEE, 2017, pp. 1–5, ISBN: 9781509051403. DOI: 10.1109/PlatCon.2017.7883728.

[157] H. M. Fayek, M. Lech, and L. Cavedon, “Evaluating deep learning architectures for Speech Emotion Recognition,” *Neural Networks*, vol. 92, pp. 60–68, 2017, ISSN: 18792782. DOI: 10.1016/j.neunet.2017.02.013.

[158] L. Hertel, H. Phan, and A. Mertins, “Comparing time and frequency domain for audio event recognition using deep learning,” in *Proceedings of the International Joint Conference on Neural Networks*, vol. 2016-Octob, IEEE, 2016, pp. 3407–3411, ISBN: 9781509006199. DOI: 10.1109/IJCNN.2016.7727635.

[159] I. Ozer, Z. Ozer, and O. Findik, “Noise robust sound event classification with convolutional neural network,” *Neurocomputing*, vol. 272, pp. 505–512, 2018, ISSN: 18728286. DOI: 10.1016/j.neucom.2017.07.021.

[160] A. Satt, S. Rozenberg, and R. Hoory, “Efficient emotion recognition from speech using deep learning on spectrograms,” in *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, vol. 2017-Augus, 2017, pp. 1089–1093. DOI: 10.21437/Interspeech.2017-200.

[161] Y. Ma, Y. Hao, M. Chen, J. Chen, P. Lu, and A. Košir, “Audio-visual emotion fusion (AVEF): A deep efficient weighted approach,” *Information Fusion*, vol. 46, pp. 184–192, 2019, ISSN: 15662535. DOI: 10.1016/j.inffus.2018.06.003.

[162] Y. Su, K. Zhang, J. Wang, and K. Madani, “Environment sound classification using a two-stream CNN based on decision-level fusion,” *Sensors (Switzerland)*, vol. 19, no. 7, p. 1733, 2019, ISSN: 14248220. DOI: 10.3390/s19071733.

[163] Z. Huang, M. Dong, Q. Mao, and Y. Zhan, “Speech emotion recognition using CNN,” in *MM 2014 - Proceedings of the 2014 ACM Conference on Multimedia*, ACM, 2014, pp. 801–804, ISBN: 9781450330633. DOI: 10.1145/2647868.2654984.

[164] S. Amiriparian *et al.*, “Snore sound classification using image-based deep spectrum features,” in *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, vol. 2017-Augus, 2017, pp. 3512–3516. DOI: 10.21437/Interspeech.2017-434.

[165] K. Qian, Z. Ren, V. Pandit, Z. Yang, Z. Zhang, and B. Schuller, “Wavelets Revisited for the Classification of Acoustic Scenes,” in *Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2017, pp. 1–5.

[166] R. M. Hegde, H. A. Murthy, and V. R. R. Gadde, “Significance of the modified group delay feature in speech recognition,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 1, pp. 190–202, 2007, ISSN: 1558-7916.

[167] X. Huang, A. Acero, H.-W. Hon, and R. Foreword By-Reddy, *Spoken language processing: A guide to theory, algorithm, and system development*. Prentice hall PTR, 2001, ISBN: 0130226165.

[168] L. Guo, L. Wang, J. Dang, L. Zhang, H. Guan, and X. Li, “Speech emotion recognition by combining amplitude and phase information using convolutional neural network,” in *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, vol. 2018-Septe, 2018, pp. 1611–1615. DOI: 10.21437/Interspeech.2018-2156.

[169] I. Paraskevas and M. Rangoussi, “Feature Extraction for Audio Classification of Gunshots Using the Hartley Transform,” *Open Journal of Acoustics*, vol. 02, no. 03, pp. 131–142, 2012, ISSN: 2162-5786. DOI: 10.4236/oja.2012.23015.

[170] P. Mowlaee, R. Saeidi, and Y. Stylianou, “Phase importance in speech processing applications,” in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[171] J. Deng, X. Xu, Z. Zhang, S. Frühholz, D. Grandjean, and B. Schuller, "Fisher kernels on phase-based features for speech emotion recognition," in *Dialogues with social robots*, Springer, 2017, pp. 195–203.

[172] M. Dörfler, R. Bammer, and T. Grill, "Inside the spectrogram: Convolutional Neural Networks in audio processing," in *2017 12th International Conference on Sampling Theory and Applications, SampTA 2017*, IEEE, 2017, pp. 152–155, ISBN: 9781538615652. DOI: `10.1109/SAMPTA.2017.8024472`.

[173] Y. Han and K. Lee, "Acoustic scene classification using convolutional neural network and multiple-width frequency-delta data augmentation," *arXiv preprint arXiv:1607.02383*, 2016. [Online]. Available: `http://arxiv.org/abs/1607.02383`.

[174] O. Abdel-Hamid, A. R. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 22, no. 10, pp. 1533–1545, 2014, ISSN: 15587916. DOI: `10.1109/TASLP.2014.2339736`.

[175] Y. Zhang, J. Du, Z. Wang, J. Zhang, and Y. Tu, "Attention based fully convolutional network for speech emotion recognition," in *2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, IEEE, 2018, pp. 1771–1775, ISBN: 9881476852.

[176] K. Kinoshita *et al.*, "A summary of the REVERB challenge: state-of-the-art and remaining challenges in reverberant speech processing research," *EURASIP Journal on Advances in Signal Processing*, vol. 2016, no. 1, p. 7, 2016, ISSN: 1687-6180.

[177] R. F. Dickerson, E. Hoque, P. Asare, S. Nirjon, and J. A. Stankovic, "Resonate: reverberation environment simulation for improved classification of speech models," in *IPSN-14 Proceedings of the 13th International Symposium on Information Processing in Sensor Networks*, IEEE, 2014, pp. 107–117, ISBN: 1479931470.

[178] A. Salekin *et al.*, "Distant emotion recognition," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 1, no. 3, p. 96, 2017.

[179] Y. Hu and P. C. Loizou, "Evaluation of objective quality measures for speech enhancement," *IEEE Transactions on audio, speech, and language processing*, vol. 16, no. 1, pp. 229–238, 2007, ISSN: 1558-7916.

[180] M. Mimura, S. Sakai, and T. Kawahara, "Joint Optimization of Denoising Autoencoder and DNN Acoustic Model Based on Multi-Target Learning for Noisy Speech Recognition.," in *Interspeech*, 2016, pp. 3803–3807.

[181] A. Satt, S. Rozenberg, and R. Hoory, "Efficient Emotion Recognition from Speech Using Deep Learning on Spectrograms.," in *INTERSPEECH*, 2017, pp. 1089–1093.

[182] K. Kumar, C. Kim, and R. M. Stern, "Delta-spectral cepstral coefficients for robust speech recognition," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, IEEE, 2011, pp. 4784–4787, ISBN: 9781457705397. DOI: `10.1109/ICASSP.2011.5947425`.

[183] S. H. K. Parthasarathi, R Padmanabhan, and H. A. Murthy, "Robustness of group delay representations for noisy speech signals," *International Journal of Speech Technology*, vol. 14, no. 4, p. 361, 2011, ISSN: 1381-2416.

[184] P. Rajan, S. H. K. Parthasarathi, and H. A. Murthy, "Robustness of phase based features for speaker recognition," Tech. Rep., 2009.

[185] C.-W. Huang and S. S. Narayanan, "Deep convolutional recurrent neural network with attention mechanism for robust speech emotion recognition," in *2017 IEEE International Conference on Multimedia and Expo (ICME)*, IEEE, 2017, pp. 583–588, ISBN: 1509060677.

[186] L. Chen *et al.*, "Sca-cnn: Spatial and channel-wise attention in convolutional networks for image captioning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5659–5667.

[187] *Azure Kinect DK documentation.*

[188] L. Yang, L. Zhang, H. Dong, A. Alelaiwi, and A. El Saddik, "Evaluating and improving the depth accuracy of Kinect for Windows v2," *IEEE Sensors Journal*, vol. 15, no. 8, pp. 4275–4285, 2015, ISSN: 1530-437X.

[189] G. Kurillo, E. Hemingway, M.-L. Cheng, and L. Cheng, "Evaluating the accuracy of the azure kinect and kinect v2," *Sensors*, vol. 22, no. 7, p. 2469, 2022, ISSN: 1424-8220.

[190] Q.-Y. Zhou, J. Park, and V. Koltun, “Open3D: A modern library for 3D data processing,” *arXiv preprint arXiv:1801.09847*, 2018.

[191] S. Liu *et al.*, “Grounding dino: Marrying dino with grounded pre-training for open-set object detection,” *arXiv preprint arXiv:2303.05499*, 2023.

[192] A. Kirillov *et al.*, “Segment anything,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 4015–4026.

[193] F. Zhang *et al.*, “Mediapipe hands: On-device real-time hand tracking,” *arXiv preprint arXiv:2006.10214*, 2020.

[194] B. D. Lucas and T. Kanade, “An iterative image registration technique with an application to stereo vision,” in *IJCAI’81: 7th international joint conference on Artificial intelligence*, vol. 2, 1981, pp. 674–679.

[195] G Bradski, “The OpenCV Library,” *Dr. Dobb’s Journal of Software Tools*, 2000.

[196] P. Virtanen *et al.*, “SciPy 1.0: fundamental algorithms for scientific computing in Python,” *Nature methods*, vol. 17, no. 3, pp. 261–272, 2020, ISSN: 1548-7105.

[197] *Matlab Camera Calibrator.*

[198] *Matlab Stereo Camera Calibrator.*

[199] M. Subbarao and G. Surya, “Depth from defocus: A spatial domain approach,” *International Journal of Computer Vision*, vol. 13, no. 3, pp. 271–294, 1994, ISSN: 1573-1405. DOI: 10.1007/BF02028349. [Online]. Available: https://doi.org/10.1007/BF02028349.

[200] Jeff Meyer and Alex Summersby, *Image sensors explained.*

[201] OpenCV, *Camera Calibration*, 2023.

[202] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, “Indoor segmentation and support inference from rgbd images.,” *ECCV (5)*, vol. 7576, pp. 746–760, 2012.

[203] D. Eigen, C. Puhrsch, and R. Fergus, “Depth map prediction from a single image using a multi-scale deep network,” *Advances in neural information processing systems*, vol. 27, 2014.

[204] A. Paszke *et al.*, “Pytorch: An imperative style, high-performance deep learning library,” *Advances in neural information processing systems*, vol. 32, 2019.

[205] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[206] L. Yang, B. Kang, Z. Huang, X. Xu, J. Feng, and H. Zhao, “Depth anything: Unleashing the power of large-scale unlabeled data,” *arXiv preprint arXiv:2401.10891*, 2024.

[207] H. Qi, M. Brown, and D. G. Lowe, “Low-shot learning with imprinted weights,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 5822–5830.

[208] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, “A discriminative feature learning approach for deep face recognition,” in *European conference on computer vision*, Springer, 2016, pp. 499–515.

[209] M. Shoaib, S. Bosch, O. D. Incel, H. Scholten, and P. J. M. Havinga, “Complex human activity recognition using smartphone and wrist-worn motion sensors,” *Sensors*, vol. 16, no. 4, p. 426, 2016.

[210] D. Roggen *et al.*, “Collecting complex activity datasets in highly rich networked sensor environments,” in *2010 Seventh International Conference on Networked Sensing Systems (INSS)*, 2010, pp. 233–240. DOI: 10.1109/INSS.2010.5573462.

[211] A. Reiss and D. Stricker, “Introducing a new benchmarked dataset for activity monitoring,” in *2012 16th international symposium on wearable computers*, 2012, pp. 108–109.

[212] H. Ma, Z. Zhang, W. Li, and S. Lu, “Unsupervised Human Activity Representation Learning with Multi-task Deep Clustering,” *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 5, no. 1, pp. 1–25, 2021, ISSN: 2474-9567.

[213] A. Abedin, F. Motlagh, Q. Shi, H. Rezatofighi, and D. Ranasinghe, “Towards Deep Clustering of Human Activities from Wearables,” in *Proceedings of the 2020 International Symposium on Wearable Computers*, ser. ISWC ’20, New York, NY, USA: Association for Computing Machinery, 2020, pp. 1–6, ISBN: 9781450380775. DOI: 10.1145/3410531.3414312. [Online]. Available: https://doi.org/10.1145/3410531.3414312.

[214] B. Wang and D. Wang, “Plant leaves classification: A few-shot learning method based on siamese network,” *IEEE Access*, vol. 7, pp. 151 754–151 763, 2019.

[215] K. Lee, S. Maji, A. Ravichandran, and S. Soatto, “Meta-learning with differentiable convex optimization,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 657–10 665.

[216] F. Moya Rueda, R. Grzeszick, G. A. Fink, S. Feldhorst, and M. Ten Hompel, “Convolutional neural networks for human activity recognition using body-worn sensors,” in *Informatics*, vol. 5, 2018, p. 26.

[217] Y. Huang, W. Ao, and G. Zhang, “Novel sub-band spectral centroid weighted wavelet packet features with importance-weighted support vector machines for robust speech emotion recognition,” *Wireless Personal Communications*, vol. 95, no. 3, pp. 2223–2238, 2017, ISSN: 0929-6212.

[218] S. R. Livingstone and F. A. Russo, “The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English,” *PloS one*, vol. 13, no. 5, e0196391, 2018, ISSN: 1932-6203.

[219] F. Burkhardt, A. Paeschke, M. Rolfes, W. F. Sendlmeier, and B. Weiss, “A database of German emotional speech,” in *Ninth European Conference on Speech Communication and Technology*, 2005.

[220] E. Fonseca *et al.*, “Freesound datasets: a platform for the creation of open audio datasets,” in *Hu X, Cunningham SJ, Turnbull D, Duan Z, editors. Proceedings of the 18th ISMIR Conference; 2017 oct 23-27; Suzhou, China.[Canada]: International Society for Music Information Retrieval; 2017. p. 486-93.*, International Society for Music Information Retrieval (ISMIR), 2017.

[221] C. Huang, G. Chen, H. Yu, Y. Bao, and L. Zhao, “Speech emotion recognition under white noise,” *Archives of Acoustics*, vol. 38, no. 4, pp. 457–463, 2013, ISSN: 2300-262X.

[222] H. Jin, Q. Song, and X. Hu, “Auto-keras: An efficient neural architecture search system,” in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 1946–1956.