Prior-Knowledge Addition to Spatial and Temporal Classification Models with Demonstration on Hand Shape and Gesture Classification

Thesis approved by
the Department of Computer Science
Technische Universität Kaiserslautern
for the award of the Doctoral Degree
Doctor of Engineering (Dr.-Ing.)

to

Aditya Tewari

Date of Defense:	25.10.2019
Dean:	Prof. Dr.-Ing. S. Deßloch
Reviewer:	Prof. Dr. P. Lukowicz
Reviewer:	Prof. Dr. D. Stricker

D 386

Contents

List of Figures

List of Tables

Acronym

ADAS Advanced Driving Assistant Systems. 3

ANN Artificial Neural Networks. 53

C3D 3D Convolutional Neural Network. 8, 115, 116

CNN Convolutional Neural Network. 15, 16, 79

HCI Human Computer Interaction. 1, 3

IMU Inertial Measurement Unit. 128

LLE Locally Linear Embedding. 12, 27, 37

LSTM Long Short-Term Memory. 17, 116, 118–123, 125–127

MLP Multi Layer Perceptron. 13, 14

PCA Principal Component Analysis. 27, 35, 37

ReLu Rectified Linear Unit. 75

RNN Recurrent Neural Networks. 8, 16, 17

SFA Slow Feature Analysis. 6, 12, 27, 33, 35, 37, 38, 52

ToF Time of Flight. 4, 34, 65, 139

Abstract

The neural networks have been extensively used for tasks based on image sensors. These models have, in the past decade, consistently performed better than other machine learning methods on tasks of computer vision. It is understood that methods for transfer learning from neural networks trained on large datasets can reduce the total data requirement while training new neural network models. These methods tend not to perform well when the data recording sensor or the recording environment is unique from the existing large datasets. The machine learning literature provides various methods for prior-information inclusion in a learning model. Such methods employ methods like designing biases into the data representation vectors, enforcing priors or physical constraints on the models. Including such information into neural networks for the image frames and image-sequence classification is hard because of the very high dimensional neural network mapping function and little information about the relation between the neural network parameters. In this thesis, we introduce methods for evaluating the statistically learned data representation and combining these information descriptors. We have introduced methods for including information into neural networks. In a series of experiments, we have demonstrated methods for adding the existing model or task information to neural networks. This is done by 1) Adding architectural constraints based on the physical shape information of the input data, 2) including weight priors on neural networks by training them to mimic statistical and physical properties of the data (hand shapes), and 3) by including the knowledge about the classes involved in the classification tasks to modify the neural network outputs. These methods are demonstrated, and their positive influence on the hand shape and hand gesture classification tasks are reported. This thesis also proposes methods for combination of statistical and physical models with parametrized learning models and show improved performances with constant data size. Eventually, these proposals are

tied together to develop an in-car hand-shape and hand-gesture classifier based on a Time of Flight sensor.

Acknowledgements

The period of this thesis has been a unique and productive experience for me. I extend my gratitude to all those who helped me complete this project. First and foremost, I would like to thank Prof.Dr. Didier Stricker for supervising my thesis and providing the opportunity of working with the Augmented Vision Group. I am particularly grateful towards Dr. Bruno Mirbach and Dr. Fréderic Grandidier for supervising me on the behalf of IEE SA Luxembourg and providing essential support and advice that guided my choices during the project.

I wholeheartedly thank Dr. Bertram Taetz who supported me through out the period of the project. Thanks Bertram, not just for the extremely useful insights, criticisms and comments, but also for the patient listening and constant motivation that you provided during this period. I wish to express my thankfulness towards all the colleagues at the Augmented Vision group for frequent lively and intelligent conversations and for help provided during many data recording campaigns. Special thanks to Jason Rambach and Kripa Sindhu Sarkar for collaborating in various capacities, and for introducing me to the concepts and ideas from their fields of expertise. I am also thankful towards Master's students and interns who helped with varied components of the thesis project.

I take this opportunity to thank my parents and brother for their love and constant encouragement. I must acknowledge and appreciate my girlfriend Harshita for the motivation, care and counsel, often from a distance, that kept me going during the hardest times. I am thankful to my friends who were always around with their support.

Finally, I thank the Technische Universität Kaiserslautern, IEE SA Luxembourg and DFKI for providing the infrastructure and academic environment for the project. I also thank Luxembourg Nation Research Fund (FNR) for the generous funding of the thesis project.

Dedication:

To my parents and public spending on research.

"You don’t collect much data when your ship is in port, tied to the dock."

Maurice Ewing

Chapter 1

Introduction

The strategy for developing solutions of vision based Human Computer Interaction (HCI) depends on the nature of camera, the available computing power, and the environment of the tasks being completed. Utilizing the environment information and working within the constraints that the task enforces may be useful while designing solutions for such problems. The varying environment and nature of problem forces redesigning of the entire solutions. However, methods which are based on learning statistical models and inferences from recorded data promise relative independence of algorithm from the camera model or the physical environment on which the task is conducted. However, such solutions impose the requirement of large and varied datasets while training the solution.

This thesis will focus on first understanding the process of feature learning for classification of images. The experiments will be made on hand shape dataset and publicly available datasets. These methods for better utilization of the learnt latent variables and mutually independent information will be used for hand shape classification in cars.

Further, this thesis will discuss methods for better classification of hand shape and

hand gesture classification within the constraints of database size. These methods will be built using various architectures of the neural networks. Broadly two methods have been proposed for this. First, those which incorporate the information about the task and data into the architecture of the neural network algorithm. Second, identifying the features that carry distinct information and methods to combine them for better performance on the specific tasks. The goal is to develop methods where existing datasets or combination of datasets, and model information can help in improving the neural network algorithm performance. This is an attempt at substituting the requirement for more data with either priors learnt from existing datasets or priors and constraints forced by the tasks.

This thesis will extensively discuss hand shape and hand gesture classification problem using machine learning based approaches. It will sometime venture into problems distinct from hand gesture and shapes. Experiments made on specific tasks in this thesis are made keeping in mind that they should be generalized into solving other tasks involving machine learning as one of the components of a solution pipeline. The basic tools for creating hand shape and hand gesture tasks already exist. This document will discuss methods developed to extend the applicability of these tools. The thesis develops a hand shape and hand gesture classification system, in the process it works on two relevant problems of machine learning algorithm based solutions. It seeks to demonstrate that,

1. The data even when unlabelled carries important information about the problem, learning the biases in the data can improve the learning performance. It can simultaneously hamper the performance of the learning algorithm on the out-of-sample data points . Judiciously learning and utilizing these biases improves the system performance without new data.

   Similar biases can also be introduced by the choice of the learning system

architectures and constrained combinations of the outputs of such systems.

2. Learning algorithms learn different feature representations, and these different representations can be combined to improve the learning performance. The correct choice of the features that are combined and the strategy for the combination improves the information that the neural network learning algorithm captures.

This thesis also deals with the tools for evaluation of the proportion of the contribution of similar-property neural network features in completing a classification task. It also demonstrates improved classification performance by utilizing the information addition in features that carry independent properties.

1.1 Motivation

One of the important methods for HCI are hand gestures without touching the sensor citefreeman1995orientation, SHG. These methods have shown to be particularly useful in Advanced Driving Assistant Systems (ADAS) in cars [1, 2]. These methods are often developed using vision based sensors specifically developed for car-interiors. The hand shape and gesture vocabulary, camera position, sensor type change the hand gesture classification problem and the required experimental datasets [3]. The nature of hand gestures also changes based on the context of the feedback or conversation [4]. Further, the problem of gesture response by the individual, and the detection and identification by the algorithm is also influenced by the world in which the gesture is made [5]. This creates a situation where the requirement for repeated data collection is created, or the existing data must be re-modelled into the specific problem that a developer wishes to solve. It is well known that the neural networks require large datasets for training and this along

with the variance discussed above further complicates the solution for a hand gesture classification system based on neural network.
The popular hand-pose dataset [6] is recorded forward facing hands with the camera placed horizontally in front of the hand. The Viva dataset [7] is recorded in a car with an RGBD sensor. The task of hand shape and gesture recognition that this thesis deals with uses a Time of Flight (ToF) sensor placed vertically above the hand location in the car. Thus, large annotated data available publicly cannot be used directly unless we develop methods to do so. This encourages developing methods where these datasets can be used to improve the performance of the proposed hand shape and gesture classification models.

1.2 Problem Statement

It can be argued that the development of a learning based solution comprises of three steps, the collection of data and its preprocessing, training a learning algorithm, and validating these algorithms. Many solutions for hand pose [8] and gesture [9, 10] identification have been proposed. These methods are based on training neural networks with labelled data. During the experiments conducted on an application for hand shape and gesture in the class it was repeatedly identified that the classification accuracy worsened when the data used to train the neural network was reduced. The motivation of this work includes reducing the requirements of specifically labelled data. To this end it is proposed that prior experiences about the task environment and the domain independent properties of the data are employed to choose better initializations and network models.
The car interiors have fast changing light conditions, also the interiors of different cars are usually distinct. The above mentioned problems can be resolved by careful preprocessing based on the properties of the car environment. The other issue that applications based on sensors which are cheaper, less popular, often built ex-

clusively for one application is the absence of existing large datasets. This leaves someone trying to develop machine learning based solutions with two problems. Firstly, generating large datasets for the tasks using the corresponding sensors. Secondly, existing strategies for transferring features learnt on large public dataset may not work off the shelf. This created a challenge where, it was important to find methods for reducing data requirement, and finding methods for targeted transfer of usable learnt knowledge.
Effectively, the three problem statements that the thesis attempts to answer are,

1. Is it possible to demonstrate the advantages of combining orthogonal information on a learning task? And, is it possible to create a quantitative measure for the feature contribution on the task performed by a neural network?

2. What are the priors that can be included in the architecture of neural network based learning systems for improved performance, or can data properties independent of the task be used as priors, does this help in domain adaptation?

3. What are the applications where multi-modal data combination improves learning performance? Can concept of multi-modal data combination can be used in multi-property feature combination?

These three questions and the demonstration of the experiments made to explore these questions have been discussed in the Chapter 3,Chapter 4 and Chapter 5 respectively. The relative advantages of following the proposed learning of features and information combination strategies will be demonstrated in each chapter. The distinctions and advantages with the existing methods will be mentioned in the sections within these chapters.

1.3 Contributions

This section briefly describes the contributions presented in each of the following chapters of this thesis.

Chapter 3 addresses the issue of improving feature learning methods and evaluating the contributions of the neural network features originating at different layers of the network for the classification task. The chapter presents a Slow Feature Analysis (SFA) based classification scheme which exploits the orthogonality of feature vectors. The applicability of SFA as an information extraction technique is ascertained. This work has been reported in [11] and the proposed extension of the SFA is shown to perform considerably better than [12] on classification of hand written digits and hand shapes.
Further, a method developed for quantitative estimates of neural network methods [13] is reported in this chapter. Random forests are used as feature selection tools and a statistical measure for feature contribution to output decision is made. This quantitative feature evaluation tool for neural networks validated various neural network related intuitions.

Publications

[11] *Tewari A., Taetz B., Grandidier F., and Stricker D., Using mutual independence of slow features for improved information extraction and better hand-pose classification. Journal of WSCG, pages 35-43, 2015.*

[13] *Tewari A., Gu F., Grandidier F., and Stricker D., Quantitative Evaluation of Salient Deep Neural Network Features Using Random Forest. 13th IEEE Image, Video, and Multidimensional Signal Processing Workshop (IVMSP), Zagori,Greece. IEEE, 2018.*

Chapter 4 presents methods for introducing some model constraints to the conventional and recurrent neural network systems [14, 15]. These models are compared against the conventional models, and models reported to work well on the hand shape problem [16] on other datasets. Further, it is demonstrated that the hand-shape class probability outputs can help gesture prediction with the support of a weak-experience model [17]. This chapter also includes experiments on domain adaptation methods where neural network is trained on statistical and model properties of the data with unlabelled data to learn data priors, it is demonstrated that transfer learning is possible using this method.

Publications

[14]*Tewari A., Taetz B., Grandidier F., and Stricker D., Two phase classification for early hand gesture recognition in 3D top view data. In Lecture Notes in Computer Science, volume 10072, pages 353-363. Springer, 2016.*

[15]*Tewari A., Grandidier F., Taetz B., and Stricker D., Adding model constraints to CNN for top view hand pose recognition in range images. In Proceedings of ICPRAM, ISBN 978-989-758-173-1), pages 170-177, 2016.*

[17]*Tewari A., Taetz B., Grandidier F., and Stricker D., A probabilistic combination of CNN and RNN estimates for hand gesture based interaction in car. In 16th IEEE International Symposium on Mixed and Augmented Reality (ISMAR), Nantes, France 2017.*

Chapter 5 evaluates the combination of neural networks to solve classification

tasks. First the experiments are carried out with multi-modal datasets on human action datasets and the utility of simultaneous training with independent sensor sources. This is then extended to a system that combines different learning properties of a hand gesture recognition system that uses a combination of 3D Convolutional Neural Network (C3D) and Recurrent Neural Networks (RNN) for identifying gestures at different delays from the start of gestures. This work combines the ideas of [18] with those of [19]. Results are demonstrated on the VIVA challenge dataset [7] and improvement in performance is noticed and reported on short incomplete sequences of the gesture. The combination model performs better on half and quarter length incomplete gesture sequence.
Finally, a system for fusing the LSTM estimate with a Kalman filter output is presented to show that the regression estimates of RNN are improved in a fusion system [20].

Publications

[21]*Tewari A., Taetz B., Grandidier F., and Stricker D., Combination of Temporal Neural Networks for Improved Hand Gesture Classification, Journal of WSCG 2018, Pilsen,May 28 - June 1, 2018.*

[20]*Rambach J.R., Tewari A., Pagani A., and Stricker D., Learning to fuse: A deep learning approach to visual-inertial camera pose estimation. In Proceedings of 15th IEEE International Symposium on Mixed and Augmented Reality., Nantes, France 2016.*

Chapter 6 presents contributions towards developing an in-car hand shape and gesture recognition systems. Concepts discussed earlier in the Thesis are used. The final architecture that allows real time classification on a moderately powered CPU and the compromises made for the application is introduced here. Finally, the chapter also describes the sensor, the recording set up and preprocessing steps before the data is fed into the neural networks.

Chapter 2

Background Theory

This thesis focuses on improving performance of machine learning, especially neural network, based methods using prior mathematical and intuitive understanding of the task or data or both. Methods will be developed for supervised learning on camera images and videos along with inertial measurements. The application of these methods will focus on a specific cases of hand and body gesture and shape classification. Often, the proposed methods will be validated for performance on public datasets and then shown to work for hand and body pose or gesture problems.

For a long period of time, any research on practical applications of learning algorithms attempted to create representations that mapped into a low dimension, high information space [22]. The creation of a feature space that captures maximum information has remained one of the important focus of the research. A breadth of mathematical and practical knowledge to create such representations have been developed. Identifying the relationships and the contributions of the components of information is an important part of such work.

The recent advances in machine learning and computer vision have been driven by

neural networks, particularly convolutional neural networks has helped in solving various problems in vision and speech [23, 9, 24]. The advances in neural network based algorithms have changed the procedures of a learning algorithm. A reasonably large proportion of computer vision research had earlier focussed on the methods for designing features. This includes choosing features by prior understanding of salient component of data like peaks [25], or designing algorithms that choose the most informative components of a particular kind of data like in case of SIFT [26]. The neural network based methods that will be discussed in this thesis combines the process of feature learning and the identification of a partition boundary between classes. This document, will propose various methods for positively influencing the feature learning process and better use of these features in prediction tasks.

In the following section a brief introduction to feature extraction and learning methods is presented Section 2.1. This chapter will then shortly discuss important contributions as well as recent developments in Neural Networks Section 2.2. Further, a report on the attempts at including physical model information to the Neural Networks is presented in Section 2.3. A short survey on information increment in learning algorithms, and use of such strategies in neural networks is presented in the Section 2.4. Finally, some successful methods for hand shape and gesture classification are listed with special focus on methods that use neural network based algorithms.

2.1 Features and Feature Learning

The performance of machine learning methods has always depended on the feature input. The features have been defined as prior information or hint about the tasks by [27], they further go on to state that "prior knowledge is meant as any information about the problem that can be used to impose constraints on the learning systems

or help it arrive at a better solution." Throughout this thesis this definition of the prior information shall be used. As mentioned below, there are two broad classes of feature representation of the data. Both the hand crafted and learned feature representation are drawn by imposing certain constraints.

2.1.1 Hand Crafted Feature

The Histogram of Oriented Gradient (HOG) and the Scale Invariant Feature Transform (SIFT) [28, 29] features remained popular for solving vision tasks. The HOG and SIFT features are based on the assumption that these features by using the local gradient are capable of capturing the local shapes.[22] have shown procedure for building feature representations using the Haar Wavelets [30]. This works by accumaliting features on increasing scales to make decisions on progressively complex regions. Similar approaches of hand designing low-level feature representations have been used in the GLOH and SURF features [31, 32]. Integral Channel Features (ICF) [33] have also been used to calculate the local features on integral images.
It is harder to create representation that are the mid-level representations of the data using hand designed strategies. Designing combination of the low-level features to create the mid-level representation is often done using learning schemes [34].

2.1.2 Learning Feature Representations

The hand crafted features described earlier have been used as a common approach for various classification and detection problems. These features are motivated from domain based applications, thus it is hard to use the same strategies on diverse data problems. Such strategies require re-engineering the design for use outside the domain in which the features were originally designed. It is hard to extend these

features to sensor modalities like scans, texts or to videos from the same sensors [35]. To develop methods for feature space identification which are independent of the data and application domain, various learning based feature representation algorithms are proposed.

Linear Factor Models are the simplest feature models and can be used a smaller blocks for large combinations [36] that describe the data. The Independent Component Analysis is linear factor analysis based method designed for source separation and has been used in various applications like background separation, face identification [37, 38]. The principal component analysis based on eigen value analysis [39] has been used for latent variable generation. The eigen value analysis based feature vector calculation is used in the famous work on Eigenfaces[40]. These methods are also identified as dimensionality reduction techniques and have the limitation that they only capture the properties in a linear subspace.

The linear factor models work best when the data resides close to a linear manifold. Methods have been developed to create the feature subspace for data residing on a curved manifold. The Locally Linear Embedding (LLE) is defined such that the data is approximated to lie on a linear surface in a small neighborhood, and each point is a combination of these local linear spaces [41]. This can be extended into scenarios where the data is distributed over disconnected regions by following the scheme proposed by [42].

Isomap projections [43] is used for identifying the non-linear degrees of freedom of a dataset. It identifies the data subspace by identifying the euclidean surface most similar to the geodesic distances calculated on the data. These methods have been used for human motion recognition [44], image classification [43]. In [45] the Local Binary Patterns (LBPs) [46] which are hand designed features have been used for identifying the texture in an image, the LBPs are analyzed using an isomap projection.

Later in the thesis, SFA [11] is discussed in some detail. The SFA is a linear factor

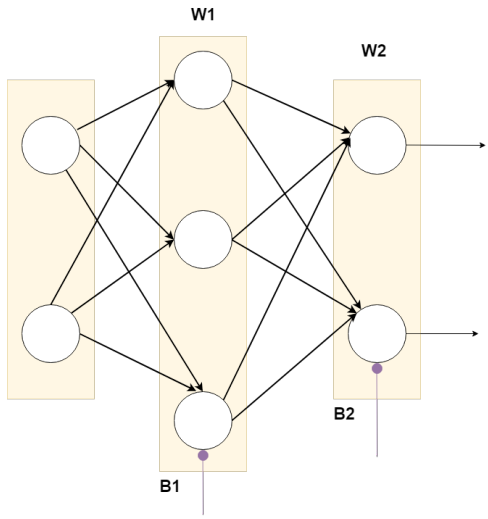


Figure 2.1: A Multi Layer Perceptron

model and is based on the principle of slowness. Slowness principle means that those characteristics that change very slowly compared to individual components of the data are better descriptor about the combined information from the data.

2.2 Artificial Neural Networks

The neural networks as computational logic were introduced by [47]. Later, [48] formalised the Artificial Neural Networks also called the feed-forward neural network or Multi Layer Perceptron (MLP) as learn-able models as a system of connected cells Figure 2.1. Each cell of the perceptron is returns a weighted addition of the impulses received from connected cell units. A set of cells are called layer and for a feed-forward network the layer with associated weights as W and bias B and input $\mathbf{x}$ has the output $\sigma(B+\mathbf{W}^T\mathbf{x})$. Where σ is the activation function which adds non-linearity to the models.

The MLP has one hidden layer before the outer cells, when activated with vector $\mathbf{x}$ the MLP returns an output vector $\mathbf{f}(\mathbf{x})$ such that,

$$\mathbf{f}(\mathbf{x}) = \sigma(B^2 + \mathbf{W_2}^T(\sigma(B^1 + \mathbf{W_1}^T\mathbf{x}))) \tag{2.1}$$

W_i are the weights connecting the inputs to the cells in the layers. A deep neural network has multiple layers and each layer outputs a multidimensional vector or scalar which is used as the output or is feed into a following layer. As shown in the Figure 2.2 a feed-forward neural network is thus a mapping form $\mathbf{x}$ to $\mathbf{y}$ such that the functions g^1, g^2, g^3 .... g^n act on the input $\mathbf{x}$ in series. Eventually, a feed-forward neural network can be represented as:

$$\mathbf{y} = g^n(g^{n-1}(...(g^3(g^2(g^1(\mathbf{x})))))) = f_{\mathbf{W}}((x)) \tag{2.2}$$

The parameters of the functions g^i are represented by the vector $\mathbf{W^i}$ and $\mathbf{B^i}$. The parameters $\mathbf{W^i}$ and $\mathbf{B^i}$ are learnt. These control parameters of the neural network are traditionally trained by a back-propagation of error [49]. The control parameters of the neural network model are called weights and the optimisation procedure for neural network was described by [50]. This document will often use $f_{\mathbf{W}}$ as a functional representation of a neural network model.

The neural networks have performed successfully on various applications in speech processing and computer vision. These algorithms have won various benchmarking in contests in the recent years. This section will briefly discuss major contributions and progresses in the deep supervised learning by neural networks.

A neural network is defined as large parallel connections of processor (neurons) that act as a collective [51]. Each neuron produces a certain activation value which is further propagated in the connections. The parameters connecting these neurons are called weights and these weights are adjustable. The weight tuning of neural networks is done using the back-propagation algorithm [52, 53]. Formal framework for use of back-propagation in neural network is described in [54], a parallel dis-

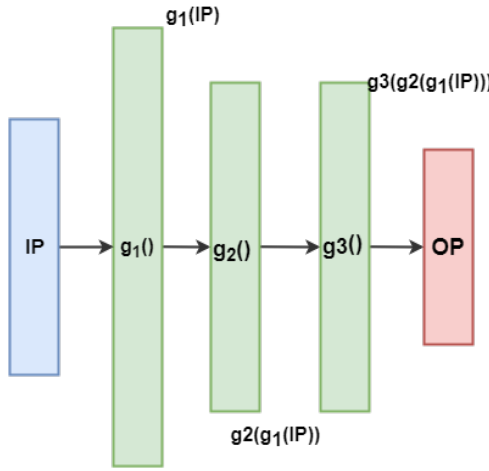


Figure 2.2: An example feedforward network

tributed approach to back-propagation was introduced in [55]. The parameters of the neural network are updated at multiple iterations of the algorithm. The neural networks frequently use the stochastic gradient descent algorithm [56] for weight adaptation using the back propagation of error method.

2.2.1 Convolutional Neural Networks

The Convolutional Neural Network (CNN) is the neural network architecture which is commonly used to process input data from a 2D-matrix. The CNN includes what are called the convolutional layers. The output of these layers are calculated by 2-dimensional filtering on the channels of the input images(2D matrices). The feed-forward equation without bias addition in the CNN for input matrix $G(c,x,y)$ is,

$$\mathbf{f}(\mathbf{x},\mathbf{y}) = \sum_{C}\sum_{U}\sum_{V} K(c,u,v)G(c,x-u,y-v). \tag{2.3}$$

The kernel K of the filter of size (U,V) is parametrised by weight vector $\mathbf{W}$ of the size $c \times U \times V$.

These layers generate feature maps by convolution with parameters learnt during the optimization procedure. A sub-sampling layer called the Pooling layer is often used in conjunction with the convolution layer in these networks. The convolutional layers are connected weights which act as convolutional filters. There have been numerous applications of convolutional networks, early applications included a delayed speech recognition system [57], document reader [58] and optical character recognition system [59].

More recently the CNNs have successfully solved problems of optical flow, segmentation, localization and human pose identification [60, 61]. The CNNs benefit from the general property of neural networks that creates hierarchical feature representation [62]. When the convolution layer is combined with a pooling layer the aggregation of multiple local features in the space is improved. [63]. This aggregation is similar to the scheme used for combination of the local Histogram of oriented gradients(HOG) and Scale-invariant feature transform(SIFT) like features in various non neural network based algorithms for computer vision. Variations of convolution layers have been proposed for segmentation of images [64, 65]. One example is the deconvolution layer, which is essentially transposed convolution layer. It has been used in semantic segmentation tasks [66], such segmentation networks generally use a network constructed only of 2 dimensional layers.

2.2.2 Recurrent Neural Networks

RNN process on input sequences of vectors in time. The RNN units have states. At time t the state is given by h_{t-1} and these states can theoretically remember past for an indefinite period.

$$\mathbf{f_t}(\mathbf{x}) = \sigma(B_t + \mathbf{W_x}^T \mathbf{x_t} + \mathbf{W_h}^T \mathbf{h_{t-1}}) \tag{2.4}$$

The RNN adds the weighted past state to the current output. In the above equation the feedback weights are represented by $\mathbf{W_h}$ and the feed-forward weights are $\mathbf{W_x}$. A longstanding deep learning problem of ineffective learning by back-propagation was exaggerated in the RNNs. It was observed that typical neural networks are constrained by problem of vanishing or exploding gradients [67].

The back-propagation error is observed to either grow exponentially or shrink rapidly. This problem in the RNNs has been alleviated by adding a memory cell with constant error carousel (CEC) in the Long Short-Term Memory (LSTM) architecture. It uses gated units that decide the magnitude of change that the present state makes on the past accumulated states. Various LSTM topologies have been developed which includes evolving structure of the memory cell such as [68]. The LSTM networks have proven to be considerably better at learning temporal patterns on time sequences longer than five to ten frames. This has been demonstatred by their comparitive performance while solving the problem of speech recognition [69], for robot localization [70], human action classification [71, 72] and other applications. Other RNN architectures that use memory units have been proposed. The Gated Recurrent Units [73] are simpler memory cells than the LSTM and have shown to perform well compared to the LSTMs on some tasks [74]. Proposals [75] for using external memory include, coupling the neural network controller with an external memory bank, the recurrent unit in this design can selectively read and write from a memory matrix. The work by [76] proposes use of four learn-able components and a static memory matrix. To distinguish from the recurrent networks from the feed-forward networks, such models will be represented by $f^r_{\mathbf{W}}$.

2.3 Model Information and Neural Network with Priors

Generally, learning for tasks inherently creates a bias learning [77]. This means that learning models for discrimination and regressions specific tasks introduces biases in models. As discussed earlier the learning compromises of prior knowledge of the data and task, and the training data. A neural network ($f_{\mathbf{W}}$) with **n** wieghts **W**, it learns a mapping between the input and expected output. Some prior knowledge or hints can reduce the appropriate **n** to achieve required performance [27]. [27] described the prior knowledge for neural network like connected graphs as, "... any information about the problem that can be used to impose constraints on the learning system or help it arrive at a better solution". While various model and data based constraints that are used often will be discussed further in this section, it is important to identify that simple choices like the network architectures, input pre-processing and the output coding also imposes certain biases.
The priors and constraints can be enforced on a neural networks by introducing several architectural constraints or by using prior biases based on some data property. The methods that have been used to introduce such information or hint before starting the training of machine learning models, specifically the neural networks are now discussed.

2.3.1 Models with prior knowledge introduced by data

A model initialized by information about the data naturally carries the biases from the data. This has been formalized as sample selection bias, its influence on learnt classifier models has been discussed in detail by [78]. This property creates the constraint that a model learnt on a sub-sample of data does not generalize outside this set; this effectively creates a need for multiple large datasets. Such constraints

have inspired work on using the prior knowledge introduced by the data in a learnt model. Two important fields of research on this front include transfer learning and domain adaptation training.

The machine learning community has worked on developing methods for predictions on the future data using statistical models that are trained on previously collected labeled or unlabeled training data; this procedure is identified as transfer learning. Methods based on domain adaptation have been proposed. In [79] the classifier is adapted from one language corpus to another. In [80] the correspondence between words in separate datasets is learnt for better transfer learning.

The other approach selects transferable properties from one model to another. One of the methods proposed by [81] includes generating class labels for unlabeled data by clustering and then transferring specific discriminative information from supervised knowledge to the unlabeled samples in other classes. In [82] presents a method for learning auxiliary problems from large unlabeled data and used it on a target problem. The Information Vector Machine of [83] learns the most informative features in a data by performing multiple tasks and creating models that conserve such features.

One of the primary reasons of the popularity of the CNN based transfer learning [84], is that transfer learning provides the ability of CNNs to share weights and learn robust image features and transfer these to new image-sets. Transfer learning using AlexNet [24] lead to various important developments in the problem related to classification, detection and localization. Like other machine learning models The pre-training of the weights in the neural network creates bias for solving problems on a certain kind of data. This assertion will be further re-visited in Chapter 4. Efforts have been made to transfer biases across domains. Using the ideas from unsupervised learning like autoencoders [85], the work of [86] tries to obtain domain-adaptation with unsupervised learning. Unsupervised methods like [87] use sparse coding to extract high level features to support supervised (transfer) learning. There

are approaches that use synthetic data for pre-training of NNs, e.g. [88]; which can be formally interpreted as model based reasoning [88].
Further, works on domain adaption focusing on neural network based adaptation have focused on maximizing feature correlation from separate datasets [89] and [90]. Some other approaches like [91] simultaneously learn domain confusion and adaptation while solving a task on labelled data from one dataset and unlabelled data from another. These methods use shared losses and shared weights concept for transfer of bias from one domain to another. These are strictly not based only on the data prior but also constraints from the network influence the transfer process.

Influence of Model Architecture on Learning.

Constraint based priors can also be forced by neural network architectures, these constraints can be forced in various ways. Shared weights [92] were first used as regularizers for the network, funnel shape networks [93] was used for better visualization and simplification of decision understanding, networks connected at the output are common examples of system where architecture of the network creates constraints on the input data based on the task. Some methods also include decomposing the network to create a neural network system [94] with the knowledge that the tasks can later be combined to understand a higher level abstraction. Some applications have also attempted to design the model such that it understands the physical property of the task domain [95].

2.4 Information Combination in Neural Networks

It has been observed that ensemble algorithms [96] based on combination of weak models perform well on generalization tasks. These algorithms are capable of

combining independent information, the importance of orthogonal or independent source of information for making decision has already been discussed in Section 2.1.2. Multi-modal learning is the process of learning to combine the features from one system (say a visual feature extractor) with feature from another system (inertial measures or audio). The multi-modal learning is inspired by ensemble methods but it attempts to learn optimum methods for combination of features.

In many applications of machine learning use of different input channel is helpful. It is possible to capture useful but different representations from separate input channels. These representations can be combined to learn about classification or recognition tasks that the machine learning problem attempts to solve. A guide to creating useful multi-modal combination is provided in [97], they suggest that the different representation should have the property that "similarity in the representation space implies similarity of the corresponding concepts", that absence of one information should be compensate-able. They then propose a Boltzmann machine based classification and retrieval application.

The Flickr retrieval solution challenge contributed to various multi-modal approaches [98, 99]. These solutions involved combination of the text and image information. Another area in which combination of separate sensor information is prevalent is the speech recognition. It is identified [100] that the combination of audio-visual cues improves the speech detection performance of any algorithm. Detailed experiments by [101] show the usefulness of such approaches, they show that the performance of the deep learning algorithms can be improved using the multi-modal learning in parallel.

These methods of information selection have also been used in creating encoders for image-sentence embedding using RNNs [102]. Similar methods have been used for combination of audio cues with skeleton images for human gesture identification [103]. Multi-modal methods have shown to be useful for sensor fusion, especially on discriminatory tasks [104]. These methods allow a flexible combination

of independent features.

2.5 The Hand Shape and Gesture Application

Most vehicles are shipped with touch and tactile based interfaces for various systems. The number of interactive systems in a car have rapidly increased over the past few years, this increases the overall complexity of controlling the operations for these systems. The touch and tactile based systems cause visual distraction which affects the attention of the driving [105]. The experiments simulated in [106] show that simpler and natural interactions with these devices can improve driver's safety. [107] has compared various methods of interactions and reported that the gesture based interaction requires least eye contact, though the study also reveals that gestures were not entirely attention free. Work by [108] also shows that the performance of the driver can degrade sharply with small increase in the shift of attention. It can thus be argued that a robust, touch-sensor free gesture based interaction can improve the safety of the driver. It is important that the chosen gestures for such application are natural gestures such that they do not distract the driver. This makes touch-less vision based methods particularly suited applications of Automotive interfaces.

The Vision based Hand Gesture identification techniques can be distributed into two broad classes. One class of solutions recognizes a static pose of hand [109, 110] and the second class uses the hand pose and hand motion over frames. For the later scheme, hand-poses over these frames may or may not change. The second scheme provides the flexibility of designing various hand-gestures. This approach allows a use of a more natural set of gestures and the number of possible actions is also increased.

The early solution for hand gesture recognition used Finite State Machines (FSM) [111], a gesture was distributed into phases and set of twelve gestures were classi-

fied. Inspired by the results on handwriting recognition [112] and speech analysis [113], the multi-modal approach by [114] does simultaneous detection and classification based on HOG and HOG^2 features. The other methods employed for action recognition use trajectory shapes, boundaries and motion structures in a bag of features approach [115].

Another branch of solution includes sequential neural networks as well as RNNs. Human and hand pose identification and hand shape classification was approached using CNNs [116]. In [117] a graphical model enforced constraints for human pose classification using CNN. These networks are trained on RGB-D dataset and can be utilized to initialize various other models, to tackle hand related problems. RNNs were used for gesture classification by [118]. More recently, various Neural Network architectures have been used for human activity and gesture recognition problems. For instance, [19] used an LSTM architecture for action recognition in videos. A C3D network [10] was trained with data from multiple vision sensors and radar. Another similar network is described in [18]. This work used multi-column C3D architecture with variation in the input scale of the two columns.

Using a neural network with convolution layers usually requires a large dataset and relatively large training time. The convolutional layers extract spatial features from the input images. As various researchers work with a multitude of sensing modalities, they can often not use datasets like [119] or [120] to initialize (pre-train) their models, due to data from a different sensing modality. This avoids the possibility to benefit from the additional variations in existing datasets to solve the task of hand-pose or hand-shape recognition more appropriately.

Chapter 3

Feature Representation

3.1 Introduction

Feature engineering is one of the most important components in a Machine Learning or Computer Vision system. Unsupervised feature representation changes the data representation process in two important ways, firstly it removes the tedious process of handcrafting the features, then evaluating them, and redesigning [121]. Secondly, the process of unsupervised learning provides some insulation against the biases of those who design the handcrafted features [122]. Obviously, such methods usually also remove expert experiences from the data representation process. In this chapter and thesis the phrases learning representations, feature learning and feature representation will be used interchangeably. Bengio et al [27, 123] listed the properties that make a useful learning representation. Some of these desired properties are re-formalized here. To learn the mapping $\mathbb{X} \rightarrow \mathbb{Y}$ on the dataset $\mathscr{D}$ of sample size N, a set of representations $\mathbb{F}$ for the set of concepts $\mathscr{O}$ are explored. Each representation function $\mathbf{f}_i \in \mathbb{F}$ provides a concept $o_i : i \in [k]$, where k is the number of representations. The members of the representation function set $\mathbb{F}$ should be such that,

1. Representation functions $\mathbf{f}_i$ are smooth, where smoothness is defined as $x \approx y \Rightarrow \mathbf{f}_i(x) \approx \mathbf{f}_i(y)$

2. The concepts o_i and o_j mapped from $\mathbf{f}_i$ and $\mathbf{f}_j$ are not necessarily mutually exclusive but are sufficiently dissimilar

3. The concepts $\mathbf{O} \subseteq \mathscr{O}$ explaining $\mathbb{P}(\mathbb{X})$, the distribution of $\mathbb{X}$, is useful for understanding a probability mapping $\mathbb{P}(\mathbb{Y}|\mathbb{X})$ for $\mathbb{X}$

4. Different values of the target vectors, especially in case of classification tasks, may be associated with different subsets of the concepts

5. The distance between the feature at time T and $T + \delta_T$ or at space S and $S + \delta_S$ is a small value $\varepsilon \to 0$.

This thesis will often return to these formalizations to justify the choices of the chosen learning representation schemes. Many modifications to the schemes for feature explorations introduced in this text are motivated from these formalizations.

3.1.1 Factor analysis

When modelling an input as $p_{model}(\mathbf{x} \mid \mathscr{D})$ the feature representation can be interpreted as a set of latent variables $\mathscr{H}$ with each latent variable identified as $\mathbf{h}_i$. The process of formation of $\mathbf{x}$ from the latent variable can be explained as a weighted sum of the infinite latent variables,

$$\mathbf{x} = \sum_{i=0}^{i \to \infty} W_i \mathbf{h}_i. \tag{3.1}$$

A set of N latent variables $\mathscr{H}$ such that $N \ll \infty$ can be used to estimate $\mathbf{x}$, this approximation of $\mathbf{x}$ is defined as the linear factor model for $\mathbf{x}$.

$$\bar{\mathbf{x}} \approx \mathbf{W}\mathscr{H} + \mathbf{z}(\sigma), \tag{3.2}$$

where **z** is Gaussian noise. This is merely an extension of the 3.1.
The latent variable $\mathscr{H}$ if represented as a conditional distribution $\overline{\mathscr{H}} : \overline{\mathscr{H}} = \mathscr{N}(\mathscr{H}; 0, \mathbf{I})$ can allow a probabilistic model based explanation for the linear factor analysis. The a priori suppositions on the $\overline{\mathscr{H}}$ and the choice of noise model influences the explored latent variables and the weights **W** of the linear factor model. Various methods are used to calculate the linear components of 3.2. These methods attempt to find a finite number of latent variables whose linear combination can best approximates the variable **x**.
Of the various Linear Factor Analysis models, the Section 3.2 of this chapter discusses the SFA. A new scheme for classification using SFA is introduced in the Section 3.2.3. The SFA is modified such that the property of orthogonality is better exploited. It is later demonstrated that this scheme of SFA works better than the existing methods and also performs better than other latent variable (Principal Component Analysis (PCA), LLE) based classifiers.
The Section 3.3 introduces the neural networks as feature learning algorithms. In the Section 3.3.1 the methodologies that were developed for understanding the contribution of neural network features to classification are described. This section uses random forests to identify the classification contributions of the features that are not necessarily mutually exclusive but carry sufficiently dissimilar concepts. The evaluations made by these methods and the resulting conclusion are described and discussed in the Section 3.3.3.

3.2 Slow Feature Analysis

This section describes the SFA, one of the various linear factor models, in some detail. The prior assumption in the exploration of the SFA is that a feature that does not vary rapidly, yet has a slow consistent change promises to describe the behavior of a function in better detail [124].The SFA allows unsupervised learning

of invariant or slowly varying features. It can learn translation, scale and rotational invariances [125]. Low level features are short duration features and are often misleading meanwhile, the high level features of the data carry information that extends beyond small neighbourhoods. SFA learns functions that represent such high level features. These high level representation can better explain the property of the data space. The slow features thus provide a consistent trend in the data.
The SFA has been modified to provide consistent trends within elements belonging to a static dataset [12]. It has also been modified to achieve supervised learning to achieve classification [12]. It provides mutually orthogonal features thus the prominent features carry independent information about the data even though they remain invariant to size, rotation and translation. Another important property of the SFA is the guaranteed optimization to the slowest changing function which allows for easy extension when learning a new class.
In this section procedure for learning several slow feature functions is introduced. This further improve the classification performance for each class . To achieve this the property of mutual orthogonality is applied on features learnt from each class. The mutually orthogonality of SFA features result in aggregation of information thus it increases the effective information that a classifier receives. The results are compared against the earlier proposed methods of [125] and [12] and it is shown that the proposed SFA scheme performs better on classification tasks.

3.2.1 Defining the SFA

SFA was originally formulated on temporal data. It was proposed as a solution for the problem of detecting trends in temporal data [125]. If a vectorial input $\mathbf{X}(\mathbf{t}) \in \mathbb{R}^{\mathrm{d}}$ is a time series, one of the slow features is the function $\mathbf{g}(\cdot)$, such that $\mathbf{y}_t = \mathbf{g}(\mathbf{X}(\mathbf{t}))$, varies as slowly as possible while avoiding trivial responses.

The problem is formally described by [126] as minimizing the absolute differential

$$\Delta(y_j) := \langle \dot{y_j}^2 \rangle. \tag{3.3}$$

Here y_j is the j^{th} component of $\mathbf{y}_t$ and $\dot{y_j}$ is the derivative of y_j with respect to time t and $\langle \cdot \rangle$ denotes average over time. The absolute differential is minimized under the following conditions:

$$\langle y_j \rangle = 0 \tag{3.4}$$

$$\langle y_j^2 \rangle = 1 \tag{3.5}$$

$$\langle y_i y_j \rangle = 0 \quad i \neq j. \tag{3.6}$$

While the minimization selects invariant features, (3.5) forces some variance and removes the possibility of obsolete solutions like a constant function and (3.6) forces independence among the calculated slow features. These constraints are forced by sphering the data [127].

Sphering of $\mathbf{X} \in \mathbb{R}^d$ means we transform $\mathbf{X}$ such that the covariance matrix of the transformed random variable $\mathbf{X}^*$ is an identity matrix. $\mathbf{X} = (x_1, x_2 ..., x_n)$, represents a data matrix and $x_1, x_2, x_3, ..., x_n$ are n vectors belonging to it. If $(\mathbf{X} - \mu)$ and Σ are respectively the centered data matrix and the covariance matrix, then the sphered data is expressed as:

$$\mathbf{X}^* = B_n(\mathbf{X} - \mu), \quad with \quad B_n^T B_n = \Sigma^{-1}. \tag{3.7}$$

The sphered data $\mathbf{X}^*$ is projected into a quadratic space, resulting in data $\mathbf{Z}$. The derivative $Z(t+1) - Z(t)$, is represented by $\dot{Z}$. Let $\mathbf{W} = [w_1, w_2 ...., w_n]$ be the eigenvectors of the covariance matrix of the derivative matrix $\dot{\mathbf{Z}}$ and λ_W be the corresponding eigenvalues,

$$\langle \dot{\mathbf{Z}} \dot{\mathbf{Z}}^{\mathbf{T}} \rangle \mathbf{W} = \lambda_{\mathbf{W}}. \tag{3.8}$$

The eigenvectors corresponding to the smallest eigenvalues are the direction of the slowest change in differential of the data. These eigenvectors compose the slow

feature functions. These functions are the weighted linear sums over the components of the expanded signal, where weights are the components of eigenvectors w,

$$g_j(x) = {w_j}^T.Z(t). \tag{3.9}$$

Where w_j is the j^{th} eigenvector. The m smallest eigenvalues correspond to the m primary slow feature functions: $g_1, g_2, g_3 \ldots g_m$. This set of slow features given by functional $\mathbf{G}$ is the $\mathbb{F}$ of features which was introduced earlier in the Section 3.1 and the output of these functions are the latent variables $\mathscr{H}$ calculated on the dataset $\mathscr{D}$. The condition of orthogonality introduced by eigenvectors enforces the mutual exclusivity of the elements of $\mathbf{G}$.

3.2.2 SFA for Classification

The slow features describe intrinsic properties of a time series. It is the property of slow features to conserve variations over time, this property can be exploited for classification. The data for classification is not temporal and thus the absolute differential described in 3.3 is modified to perform a supervised classification. To perform a supervised classification, functions resulting in minimum inter-element difference within each class must be identified. As in case of time series SFA, the conditions of zero mean, constant variance and linear independence are imposed. Once again these conditions are satisfied by sphering the data. Furthermore, the optimization process tries to increase the variance outside a class, to identify the slow feature functions.

For the dataset $\mathscr{D}$, we define a matrix $\mathbf{Z}$, such that $\mathbf{Z}$ is the quadratic expansion of the sphered transform of $\mathscr{D}$. Accordingly, the differential term for a vector z^{el} belonging to the expanded dataset $\mathbf{Z}$ is represented as:

$$\nabla_{el} := \sum_{C=1}^{N} \left(\sum_{n=1}^{N_C} (z_C^n - z^{el})^2 \right)^{\frac{1}{2}}. \tag{3.10}$$

Thus average differential for the data **Z** can be re-represented as:

$$\nabla := \langle \nabla_{el} \rangle. \tag{3.11}$$

Where, z^{el} is the vector corresponding to the element for which the differential is calculated. z_C^n is the n^{th} element of a class C, N is the number of classes and N_C is the number of data points in the class C. The value of ∇ is now minimized. This minimization condition returns functions that forces slow variance within classes. Each of the slow features correspond to one of the classes, to further improve the extracted feature functions, 3.11 is extended to maximize the variance between classes while minimizing it within the class [128].

To achieve this we subtract the average of the absolute difference of the in-class element with elements outside the class (∇_{el}^o) from the average differential within the class (∇_{el}), that yields

$$\nabla_{el}^o := \sum_{C=1}^{N} \left(\sum_{\{c=1, c\neq C\}}^{N} \sum_{n=1}^{N_c} (z_c^n - z^{el})^2 \right)^{\frac{1}{2}}. \tag{3.12}$$

This difference of in-class and out-class variance is used as the cost function. Effectively, the slow feature function of 3.11 is modified to minimizing the cost function O, where O is defined as:

$$O = \langle \nabla_{el} \rangle - \langle \nabla_{el}^o \rangle. \tag{3.13}$$

The minimisation process is described in the next section.

3.2.3 Using Orthogonality to Increase Information

The classification process described above returns (N=number of classes) functions. These functions are learnt from the entire dataset using the optimization function of 3.13. This procedure results in a set of functions which provide low variance response. The constraint of decorrelation between different slow features creates the possibility of learning many functions corresponding to one class.

The ready availability of features after doing an SFA procedure, and there mutual independence motivates us to find more features within a class. Thus we calculate multiple slow features corresponding to each class. Rather than learning slow features over the entire dataset we learn a set of functions for every class. Slow features are learnt by restricting the dataset to elements of one class, this is repeated for all classes.

As each function is orthogonal, we have more than one function representing intrinsic properties of the specific class. These linear functions are decorrelated on the expanded space. Learning slow features in every class requires a larger training dataset, meanwhile it also results in adding information for classification. The optimization function 3.13 is further modified to minimize variance within a class, while maximising out-of-class variation using all other classes (3.15). This modification extends 3.12 as follows:

$$\nabla_{el_C} := \left(\sum_{n=1}^{N_C} (z_C^n - z^{el_C})^2 \right)^{\frac{1}{2}}, \tag{3.14}$$

$$\nabla^o_{el_C} := \left(\sum_{\{c=1, c\neq C\}}^{N} \sum_{n=1}^{N_c} (z_c^n - z^{el_C})^2 \right)^{\frac{1}{2}}, \tag{3.15}$$

where, $\nabla^o_{el_C}$ is the sum of out-of-class variances calculated over the training dataset.

$$O_C = \langle \nabla_{el_C} \rangle - \langle \nabla^o_{el_C} \rangle. \tag{3.16}$$

el_C represents that the calculation for the differential is done for elements belonging to the class C. The optimisation for class C is achieved by minimizing O_C.

The functions are collected as matrix $\mathbf{W_C}$ where C is the class for which these functions are learnt. $w_{\lambda_{C_j}}$ is the vector corresponding to the j^{th} eigenvalue λ_{C_j} of class matrix $\mathbf{W_C}$. For a test input vector $\mathbf{P}$ the functional $\mathbf{G}$ returns an output vector $\mathbf{G}(\mathbf{W}, \mathbf{P})$. The functional $\mathbf{G}$ has m linear functions in the space corresponding to the dimension of vector expanded in data space,

$$G(\mathbf{W_C}, \mathbf{P}) = \mathbf{P} \cdot \mathbf{W_c}^T. \tag{3.17}$$

The variance for the output of the function is calculated as,

$$Var_C \quad = \sum_j (\mathbf{P} \cdot w_{\lambda_{C_j}})^2 \quad = \sum G(\mathbf{W_C}, \mathbf{P})^2. \tag{3.18}$$

The final classification is performed as follows:

$$class = \underset{C}{\operatorname{argmin}}(Var_C). \tag{3.19}$$

While doing an N class classification using m functions for each class, we have Nm functions.

The value of functions corresponding to a class when applied to an element from the same class is centred around a constant value. When a function is applied on a mismatched class, the result is random. This randomness likely results in a wrong identification.

In the case of multiple centred functions, corresponding to a class, the resulting output for a matching sample has all the function outputs centred around zero. Some functions from non-matching classes may return centred responses close to zero but, the aggregated variance for a mismatch element is higher, resulting in clearer distinction from the matching class.

The next section discusses SFA and the proposed classification approach in light of the earlier discussed (Section 3.1) properties of good latent variables.

3.2.4 Classification with SFA

MNIST dataset [129] is one of the classical dataset for classification problems. To validate the proposed method of further tested and compared both methods of using SFA for classification described earlier on the same dataset. Earlier, [12] described the original classification technique on the MNIST Hand written digit dataset.

Two experiments have been performed. First, ten slow features functions are calculated for the full dataset. Second, ten slow feature functions for each class are

calculated according to the proposed method of Section 3.2.3. It was observed that the identification performance for every class improved when the property of in-class orthogonality is used to calculate slow feature functions. The comparative results are listed in Table 3.1.

Table 3.1: Digit classification accuracy. The second row values are accuracy percentages when slow feature functions are learnt from the entire dataset, the third row shows the accuracy percentages when several functions are learnt.

Class	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Full Dataset	81	93	79	83	77	72	77	80	73	84
Class Separation	91	96	82	85	79	81	89	91	83	84

Hand shape data and classification

A 3D ToF, PMD-Nano camera has been used to collect a dataset of hand-poses. The camera is fixed vertically above the palm. The output of the PMD-Nano ToF camera is an 120x165x2 image. The two channels of the image are the amplitude value and the depth map image data. We cover the arm region with absorbent clothing and use the reflectance of skin to segment the palm. The reflectance constraint does not entirely remove the background and thus the closest contour greater than a threshold area is chosen as the palm region. The segmented palm region is then converted into a binary image which is further used for hand-pose identification.
The Hand pose dataset is used for a classification task on classes labelled as "Fist", "Flat", "Index", "Open" and "Grab". Overall, fifteen thousand samples of hands were collected and the slow features learnt. Thousand frames in each class are randomly selected and rotated in either direction, by an angle between 10° and 20°. These rotated frames are added to the training dataset along with the original frames. Note that, this spreads the poses such that they cover the whole rotational

axis, it also increases the dataset and generates samples which train the SFA for rotational invariances.

Before learning slow features from the dataset of segmented hands, the image is scaled down to one-third of its original size. This is followed by a PCA which reduces each image to a thirty-five dimension vector that is projected to its quadratic space to allow the learning of non-linear invariances in the principal components of the training data.

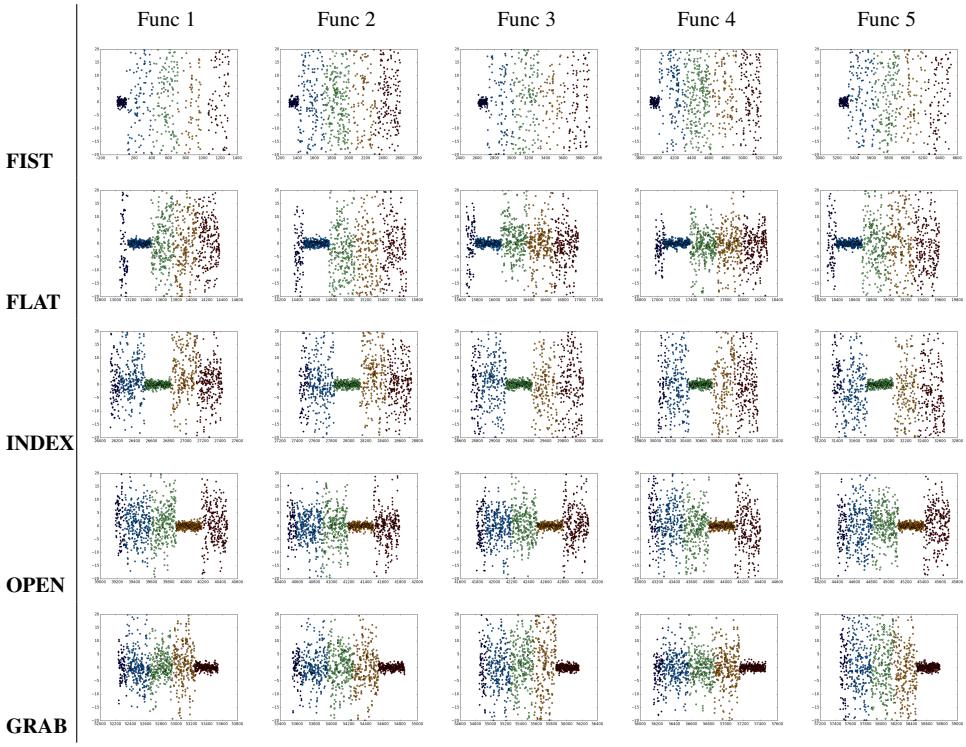
Three hundred frames are selected for each class through random partitioning of the original dataset. These samples are used as test dataset, while the remaining original dataset is used for training. The preprocessing follows the same procedure as described for the training dataset.

During the SFA learning process the covariance matrix of the differential data as well as the eigenvectors corresponding to the largest eigenvalues are recorded.These vectors correspond to linear functions used for classification. Once learnt, these functions are used to compute the corresponding function-mean over the training samples. Each function is centred around the mean values learnt during the training process, and the variance over the functions is then calculated.

It is observed that the samples of matching classes are tightly spread around the mean values of the classes. The class which corresponds to the function has much smaller variance as compared to other classes. Figure 3.1 shows the response of the test dataset on the most prominent function of the "Fist" class. The data points for each class are represented by a unique color. The "Fist" class which is represented by blue in the figure has relatively tight packing of the data-points as compared to any other class. Like in the previous figures the X axis of the plot represents the data points which are arranged by their labels, and the Y axis represents the centred value of the learnt function.

This pattern is visible over the entire set of Nm functions, $5 \times 10 = 50$ in the present case. Table 3.2 shows the response of each data-point to the first five of the ten learnt

Table 3.2: Scatter map showing the value for 5 SFA functions for every class on the test dataset, different colors represent different classes.



functions of each class. Each rows represents a set of functions corresponding to the class.

It can be observed that the functions learnt for one class have lower variance in the same class, while higher variance in other classes. This observation is used to differentiate classes. Thus we calculate the variance of the function response over all the functions calculated for a class.

The three hundred frames of each class in the dataset are used for evaluations. While learning models that are saved include, PCA mapping for each class, the sphering matrix, m eigenvectors and the covariance matrices for each class.

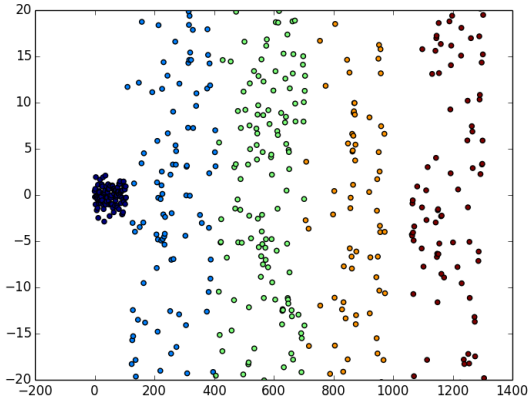


Figure 3.1: Slow function response for class 'Fist'.

3.2.5 Classification Accuracy Comparison

The results of the classification using SFA were compared against results from KNN on physical features extracted from each frame. The physical features include coordinates of the tip of the finger (or the tip of the palm), the coordinates of the palm centroid, the convex ratio and the concave depth of the image and the polar and azimuth angle of the finger [130, 131, 132] . Comparisons are also made against the results to KNN applied on the PCA of the data and the low dimension manifold of the raw binary image [133].

These calculations are demonstrated to show the performance of the SFA based latent variable against the latent variables calculated using well known linear factors methods like LLE and PCA. The KNN models for the physical features are generated using 1500 samples from each class and are modelled by simple euclidean distances. The manifold is learned by Isomap algorithm [43] and the learning is

Table 3.3: Confusion matrix for SFA classification.

%	FI	FL	IN	O	G
FI	**97.0**	1.0	0.0	1.7	0.3
FL	0.0	**96.7**	2.3	1.0	0.0
I	0.0	0.0	**98.7**	1.3	0.0
O	1.0	0.0	1.3	**97.6**	0.0
G	0.7	2.3	0	0.3	**96.7**

Table 3.4: KNN Confusion Matrix for physical features.

%	FI	FL	I	O	G
FI	**97.0**	0.7	1.3	0	1.0
FL	0.7	**95.7**	3.0	0	0.7
I	2.7	5.7	**91.7**	0.3	0.0
O	3.0	2.3	0	**94.3**	0.3
G	0.7	4.7	0	0.3	**94.3**

Table 3.5: KNN Confusion Matrix for 35-D PCA.

%	FI	FL	IN	O	G
FI	**78.3**	12.2	2.9	3.8	2.9
FL	1.3	**80.7**	6.3	6.6	5.0
I	0.0	3.3	**81.7**	2.0	14.0
O	0.0	7.7	4.7	**85.3**	2.3
G	0.3	3.3	7.7	3.0	**85.7**

Table 3.6: KNN Confusion Matrix for 9-D isomap.

%	FI	FL	IN	O	G
FI	**97.0**	0.3	2.0	0.7	0
FL	0.3	**98.3**	1.3	0	0.3
I	3.7	0.3	**96.0**	0	0
O	1.7	0.0	1.0	**96.3**	1.0
G	2.7	0.3	0.3	0.7	**96.0**

done by the same training data as used for SFA.

The SFA based classification works better than the physical feature based classification evaluated in the KNN model. It also outperforms the KNN evaluation done with 35-dimensional (35-D) PCA and 9-dimensional (9-D) manifold representation of the dataset. A 35-D PCA is chosen because it is used as the basis for SFA calculation and 9-D isomap because the classification by KNN performs best for it.

Table 3.3 shows the confusion matrix for the SFA based classification and Table 3.4 shows the confusion matrix for classification on KNN model trained on the hand crafted physical features. Table 3.5 is the confusion matrix for classification results from KNN model trained on the 35-D PCA representation of the image data. While classifying on the 9-D element vector received from the isomap done on the palm region as described earlier, the results are improved as compared to KNN on physical features and PCA based KNN. Table 3.6.

The results from the SFA are considerably better than the results from the physical features. These features are carefully selected for hand-pose estimation. This

underlines the ability of the method to search for relevant features in a class. This improvement also suggests that SFA is capable of reducing the effect of local noise and distortion.

Comparison of SFA is made against KNN on the lower dimension representation of the data computed by PCA. The confusion matrices of Table 3.3 and Table 3.5 clearly demonstrate that SFA performs far better. Thus the process of calculating the slow feature functions after doing PCA on the data further refines the knowledge that we are able to extract from the dataset.

SFA classification also performs better than a KNN model trained on manifold representation of the dataset. While the identification of the "Flat" hand-pose is better than the SFA in case of the isomap representation, the overall performance of SFA is superior. This result suggests that SFA is capable of managing non-linearities in the data, this can be attributed to the step in which the PCA data is projected onto a quadratic space.

The improvement from PCA to Isomap modeling is a result of better handling of non-linearities in the data. The KNN model based on euclidean distances suffers from the inability to compensate for non-linearities, this is overcome when we use the isomap projection. It is also important to note that while the KNN model is learnt over the isomap projection, SFA classification provides better results by simple variance calculations. It is worth mentioning that the performance improvement in the quality of classification was minimal when we scaled the palm region by distances. This observation can be attributed to the characteristic of SFA that, it explores multidimensional linear functions which encompasses the invariances over the data points.

3.3 Evaluation of CNN as a Feature Extractor

The calculation of the SFA, as described in the Section 3.2, involves SVD decomposition. As, the dataset becomes larger the calculation involves large matrices. Thus the experiments on the SFA were made on a relatively smaller dataset. The neural network on the other hand learns by iteratively evaluating the input datasample, this allows the use of larger dataset.

The CNNs as feed forward neural networks were introduced in the section Section 2.2. It was mentioned that these networks may have one or more convolutional layer connected to fully connected layers. The combinational output of the convolutional layer are used as input to the traditional multi layer perceptron. The CNN learns a combination of suitable filters which are used to extract features from images. The large number of features provided by the convolutional neural network provide separate information. Visualization of feature property is one of the often used methods to do so. It is easy to identify that these features rise on the scale-space as they propagate through the neural network.

Unlike the SFA from the last section, it is not possible to design mutually exclusive features by training a CNN. But, it is important to understand the properties of these features. The CNN features include the properties of the data type, data domain and task. In the SFA, the orthogonality, the in-class invariance and the divergence among features from different classes is forced by an optimization constraints, the cross entropy cost function does the same optimization for maximizing the accuracy performance of the neural network.

Table 3.7: Hand Shape classification accuracy

%	Flat	Point	Open	Fist	Join
CNN	89	88	88	89	96

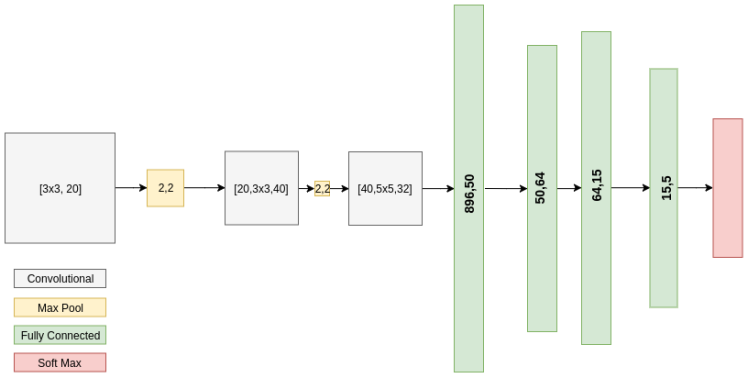


Figure 3.2: Hand shape classifier

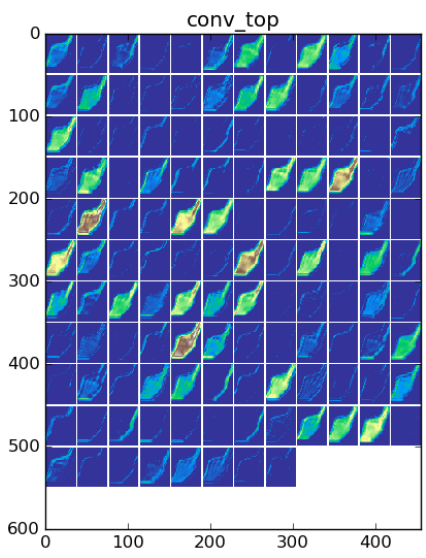
A simple hand shape classifier CNN shown in Figure 3.2 is trained. The training for this neural networks is completed in ten thousand batch iteration of sixty-four samples per batch. The classification of the hand-shapes and the class-wise accuracy values for the large dataset are given in the Table 3.7.

To develop and compare the tools for feature analysis the experiments on the hand shape dataset are conducted with the network shown in Figure 3.2. The features in the neural network are not separable and thus various methods there analysis have been proposed. The visualization and node-activation based methods for neural network are important in understanding the underlying nature of the network. In this section the methods for a quantitative measure of saliency are proposed and described. The Guided Random Forests (GRRF) [134] are used to identify regions carrying relatively independent information. Further, modifications in the entropy measure are made to develop tools for identifying important and unimportant features for the specific task. Along with the GRRF, the concept of shadow features [135] is used to identify features informative for the classification tasks, importance feature ratio from each class is calculated and presented. The neural network features extracted from the hand shape dataset are shown in Figure 3.3.

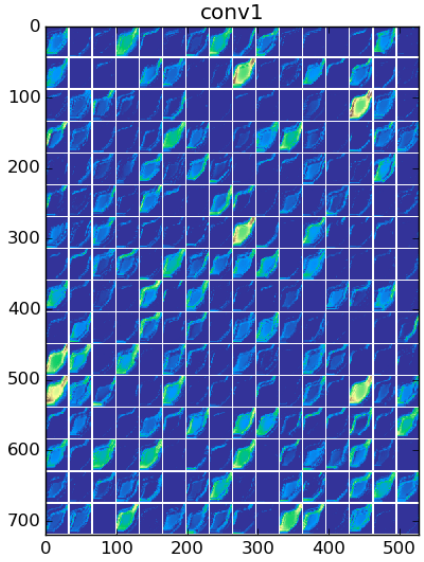
In this sections the experiments are made by using the features extracted from a neural network shown in Figure 3.2. A hand shape dataset with one hundred five thousand samples of hand shapes in training set and ten thousand samples in the test set is used for the experiments of hand shape classification. The dataset is later described in detail in the Section 4.5.1. For experiments, feature datasets are created by extracting features produced by all images of the datasets. These features include outputs from all non-output layers, and carry an identifier of the layer of origin. As the feature vectors create an extremely large dataset, the experiments and the evaluations are conducted on one-fourth of the training data images are randomly sampled from the dataset.
Experiments are also conducted on the classical LeNet [136] model on the hand written digit MNIST dataset.

3.3.1 Feature saliency Measure With Random Forests

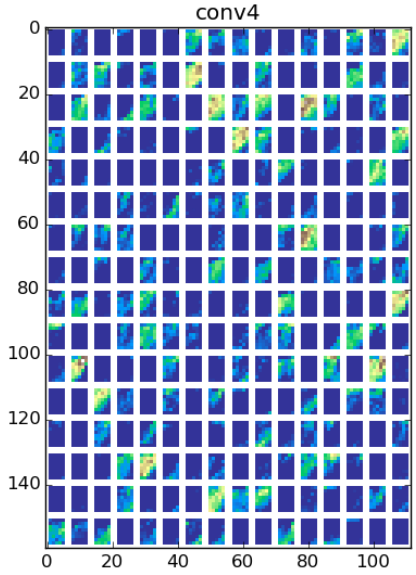
Samples of the features resulting from the trained neural networks are shown in the Figure 3.3, these features contribute to the classification task. The task now is to identify which components of the network are learning relatively independent features, and which part carry important informations. To this end a Random Forests algorithm based on the idea of growing a large number of trees from random samples of training data is used. Each tree of the Random Forests votes for a class and the Forests chooses the class with the highest vote. A decision tree, at every node, makes a binary partition such that the total information gain, $gain(\cdot)$, is maximized. The information gain is the increase in normalized entropy, $\mathbb{H}$ after every partition. This strategy for growing a tree and the random sampling from training data to create multiple trees makes Random Forests classifier a tool for measurement of feature saliency. When high dimensional feature vector is input into the Random Forests the probability of uninformative feature selection and multiple splitting of



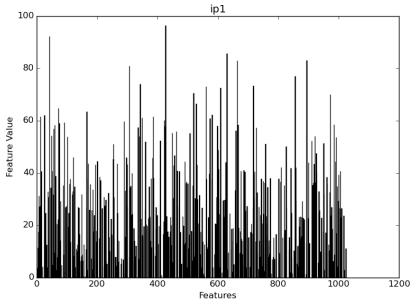
(a) Sample of channels from input convolution layer



(b) Sample of channels from first convolution layer



(c) Sample of channels from last convolution layer



(d) Sample of channels from first fully connected layer

Figure 3.3: Samples of channel ouptput from different layers of the network.

branch is increased. The two possible solution proposed in earlier research shall be shortly discussed and will be used with Random Forest on Neural Network features to identify the informative regions of the neural network and there contribution to classifier feature set.

Guided Regularized Random Forests

A saliency measure for each feature that contributes to the classification in Random Forest is calculated. The saliency score Ψ_i for a feature $\mathbf{f}_i$ in the Random Forest input is calculated as,

$$\Psi_i = \frac{1}{n_{tree}} \sum_{\eta \in \mathscr{N}_i} gain(\mathbf{f}_i, \eta) \tag{3.20}$$

Where $\mathscr{N}_i$ is the set of nodes where partition decision is made by feature $\mathbf{f}_i$ and n_{tree} is the number of trees in the Forests.

The Regularized Random Forest (RRF) are designed such that they use minimum number of features while creating Random Forests for a task. An initial set of features $\mathbb{F}_0$ is chosen. This set is iteratively increased in size. At each test step t the set is modified to $\mathbb{F}_t$. This modification is achieved by adding a new feature $\mathbf{f}$ if the information gain by a partition using the feature $\mathbf{f}$ is at least $\frac{1}{\lambda}$ fractions higher than gain from features already existing $\mathbb{F}_{t-1}$. At the end of the iterations of the features the feature set, $\mathbb{F}$, is created.

λ is used to normalize the gain and create the regularized gain term, the $gain_R(f)$ is defined as:

$$gain_R(f_i) = \begin{cases} \lambda_i \cdot gain(\mathbf{f}_i) & \mathbf{f}_i \notin \mathbb{F} \\ gain(\mathbf{f}_i) & \mathbf{f}_i \in \mathbb{F}. \end{cases} \tag{3.21}$$

The number of samples used to train the deeper levels of the tree nodes are comparatively much smaller than the numbers used to train a classic Random Forests. The smaller data size used to create the partition boundary in these scenarios makes the *gain* evaluation deceptive. Thus a gain regularizing factor λ_i is used to restrict the

size of the tree. It also ascertains that only those features that add substantially to the decision tree information are added to the final $\mathbb{F}$. Unlike 3.21 where the gain regularizer λ is a constant, the GRRF uses a parameterized λ which is defined as:

$$\lambda_i = (1-\gamma) + \gamma \cdot \bar{\Psi}_i \tag{3.22}$$

Where the normalized saliency score $\bar{\Psi}_i$ of the feature f_i is obtained by normalization of the saliency score Ψ_i by the maximum saliency score Ψ^* in the feature set $\mathbb{F}$. It is to be noted that the normalized saliency is calculated by first training a classical Random Forests on the training data.

Shadow Features

The shadow feature method avoids selection of features which do not provide information about the task(classification here) to be completed by the Random Forests. In these experiments, the sample feature are tested by creating duplicates of the original samples and creating a random distribution of labels for the duplicate samples. Thus, those features of the duplicate samples should not provide any important information to the prediction task. The procedure described in [135] is then followed to choose the important feature set $\mathbb{F}_i$. The training set containing all the shadow samples is trained by different random forests. N Random Forests are trained and the i^{th} Random Forests selects a features set $\mathbb{F}_i^s$. The saliency measure is calculated by the repetition frequency of these features. The best saliency value of each randomized sample in the shadows is chosen to form the final sample representing the uninformative features, which is called shadow saliency sample in the following context. Those real features whose contribution to the classifier is same as shadow sample features are discarded. Thus a subset of the salient features set is selected. The similarity of the features is calculated by the Wilcoxon check sum test [137].

These two methods are used for the saliency analysis of Neural Network features in the experiments described in the following sections.

3.3.2 Experiments and Observation

The analysis is first performed on the features from a convolution and pooling layers to decide if features from one of the two layers can be used to represent same information.The Random Forests are built using convolutional features from each convolutional and pooling layers of the neural network. In each case, multiple forests are constructed with a defined maximum depth. The results from this experiment confirm the theoretical expectation that the corresponding convolutional and pooling layers provide features of similar information but because of reduced features size in pooling layer, the per-feature performance of the pooling layers is better than the convolution layer features. This result will be used to reduce the feature dimension in the following analysis. Hereafter only the pooling layer features shall be used as the proxy for the features from the corresponding convolutional and pooling layers.

Important Feature Frequency in Random Forests

In the following experiments the regions of the neural network that contribute higher to completing the task of classification are identified. This is a simple measure of frequency expectation of the source of salient features. Random Forests consisting of hundred trees and twenty maximal depth is trained. During the training procedure, the features providing the maximum splitting gain at each node are counted. The Random Forests is trained using the features from the convolutional features from the final two 2-D layers and the first fully connected dense layer.
Figure 3.4 shows the share of features from each neural network layer used for

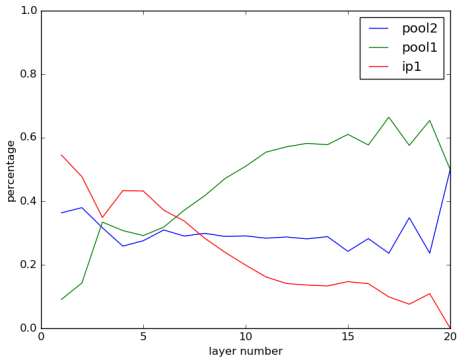


Figure 3.4: Salient Feature Contribution from layers at various decision tree levels.

splitting the dataset at nodes of the Random Forests when tested with the MNIST dataset. The first fully connected layer contributes extensively to splitting the nodes on the first level of the Random Forests trees. It was also observed that as the trees grow deeper the contribution of the features from the pooling layers increase. The outermost layer contributes (58%) the largest to the top layer($= ip1$) of the decision Forests which means it contributes the highest entropy for the required classification task. The less discriminatory but yet important features for making decisions steadily increase on traversing down the layers of the decision forests to being 50% each at the twentieth layer.

3.3.3 Salient Feature Reduction Using Random Forests

The last experiment uses all features of the neural network which is a very large dimension vector, it thus suffers from the problem of not providing a credible estimate of the importance value. The stated goal of these experiments is to identify importance of salient features or neurons that are salient for the particular task. In order to achieve this the salient and essential features are identified.

The innermost fully connected layer and the outermost 2-D layers are used as feature source for training the neural networks. The experiment to identify and evaluate the salient regions of the neural network are now presented.These experiments are conducted with GRRF and shadow features.

Experiment with Guided Regularised Random Forests

A simple Random Forest is first trained on the training dataset created from the neural network features. This is done to calculate the normalized saliency score. The value of $\bar{\Psi}_i$, for each feature is calculated. These normalized saliency scores are then used for training a GRRF. The GRRF is trained using the same training set and the normalized saliency score scheme. The GRRF selects a subset **f** of the feature set $\mathbb{F}$ and provides a classification output. In these experiments the size of the subset **f** depends on the parameter γ introduced in 3.22. Larger γ value makes the addition of new features to RF classifier harder. Thus, the larger the γ smaller the size of the selected subset **f**.
The performance of the GRRF trained Random Forest is compared to the Random Forest trained on the entire features of the neural network. The accuracy of Random Forest trained on full feature set is 98.5% for the MNIST and 91.6% for the Hand shape dataset. When the value of γ is much less than hundred the GRRF eliminates almost half of the features without a considerable reduction in the accuracy performance. In fact in the MNIST experiment for the γ value of 0.005 and 0.015 the accuracy of over 98.6% is achieved which is higher than the accuracy from the full feature set. The reduction in the feature size and influence on the share from different layers is shown in Table 3.8 and Table 3.9.
A feature survival rate for each layer can now be defined. *The ratio of the features which originate in a particular layer accepted by the GRRF to the total features provided by the layer* is the feature survival rate. The accuracy performance, the

percentage size of the feature vector *wrt* the original feature vector and the layer-wise survival rate with increasing γ are recorded in the Table 3.8 and Table 3.9. It is apparent that while the inner layers of the network are less dominant in the contributing to the classification property, the decay rate for the contributing features for such layers in substantially larger than the outer fully connected layers.

Table 3.8: MNIST: GRRF accuracy and layer wise survival

			IP	1^{st} Pool	2^{nd} Pool
γ	Accuracy(%)	Size(%)	Survival Rate(%)		
0.005	98.66	28	58	36	26
0.010	98.62	25	51	37	22
0.015	98.65	23	52	34	20
0.03	98.62	19	42	30	15
0.06	98.54	12	29	16	11
0.10	98.40	10	21	9	6
RF trained with all features has accuracy 98.5%					

Table 3.9: HAND data: GRRF accuracy and layer wise survival

			IP	3^{rd} Conv	2^{nd} Pool
γ	Accuracy(%)	Size(%)	Survival Rate(%)		
0.005	91.6	33	64	34	27
0.010	91.4	28	61	26	22
0.015	90.8	24	56	22	18
0.03	90.8	22	51	20	18
0.06	91.2	18	32	16	16
0.10	90.4	12	26	13	9
RF trained with all features for Hand Shape has accuracy 91.6%					

Experiment with Shadow Features

The features generated by the CNN and collected earlier are duplicated. The duplicated set of features are labeled with incorrect and randomized labels. These features are thus information-less. The training set of feature vectors is thus corrupted with information-less features and the combined feature dataset is used for

training $N = 200$ Random Forests.

Using the earlier strategy from RRF a subset of salient features are collected, this subset of salient features is again tested with Wilcoxon rank sum method. While training a random forest only parts of the features are used. On the entire set of the N Random Forests the impact of real and duplicate features on the output is compared. As mentioned earlier, the measure of this similarity is created using the pValue calculated from the Wilcoxon test.

Those features which are similar to the shadow features in influencing the output are considered uninformative features and are filtered out. Thus, If the pValue in the Wilcoxon rank sum test of a feature and its shadow is bigger than a chosen threshold, this feature is discarded from the salient feature set $\mathbb{F}$.

In the experiment, saliency scores of 200 different random forests are gathered to perform the Wilcoxon rank sum test. The results are shown in Table 3.10 and 3.11. The relatively lower fall in the size of the vector shows that most features learnt by the CNN contain task information. It also suggests that while the features contain useful information but they contain information which is repeated, as the reduction in the useful feature size in these experiments is considerably less than the experiment done with GRRF.

Table 3.10: MNIST: GRRF accuracy and layer wise survival using the shadow features

			FC	1^st Pool	2^nd Pool
$-\log(pValue)$	Accuracy	Size(%)	Survival Rate(%)		
10	98.5	96.7	100	94	98.2
20	98.5	93.8	100	91.5	98
30	98.7	92.4	100	89.5	97
40	98.5	90.7	100	87	97
50	98.4	89	100	84	96
100	98.4	68.8	100	72	92
125	98.6	51.9	95	48	83.4
150	96.6	45.8	93	43	72.4

Table 3.11: HAND Data: GRRF accuracy and layer wise survival using the shadow features

			FC	2^{nd} Pool	3^{rd} Conv
$-\log(pValue)$	Accuracy	Size(%)	Survival Rate(%)		
10	91.6	97.1	100	95	98
20	91.6	96.5	100	91	97
30	91.4	91.2	100	92	96.7
40	90.4	89.6	100	87	96.3
50	90.4	88.8	100	88	95
100	90.8	72.0	100	72	82
125	89.8	62.0	100	42	63.4
150	90.2	52.0	98	33	55.4

In both experiments it was demonstrated that the feature size can be reduced considerably without loss of accuracy, it indicates repetition of information. The size can be reduced to almost 50% with less than 1% loss in prediction accuracy.

While the CNN performed better than the Random Forests trained to classify the MNIST dataset, it was observed that the performance of the Hand classification improved when all neural network features were used to train the CNN Table 3.12. It is possible that this is because of the non-optimal choice of the learning parameters for the neural network. This observation suggests that a shorter training of Random Forests can be used to optimize the hyper-parameter choices while training a neural network.

The experiments show consistent properties of neural network layers. Two meth-

Table 3.12: Accuracy for hand shape classification with CNN and Random Forests on CNN features

%	Flat	Point	Open	Fist	Join
CNN	89	88	88	89	96
RF on CNN	**91**	88	**92**	**91**	95

ods for validating these properties were introduced and the experiments were com-

pleted on two separate datasets to check for consistencies. The utility of these experiments is proven by validation of various qualitative arguments. The argument that the outer layer contains properties which increase information about the task is validated by these experiments, that the pooling layer increases information density is also demonstrated. Overall these methods for quantitative evaluation of features demonstrate that various features of the deep neural network share information. Reducing the features to half the original size only has a minuscule influence on the performance of the network. The higher discriminatory nature of the outer fully connected layers is quantitatively demonstrated. The importance of pooling layers as information intensifying layer is also validated.

3.4 Conclusion

This chapter introduced the information increment through increasing the number of extracted orthogonal feature vectors in a dataset. This was done using two examples, the new scheme for SFA and the evaluations made on different but not mutually exclusive neural network features.

It was demonstrated that increase in orthogonal slow feature component improved the performance on a classification problem. As orthogonality could not be forced on the features learnt from a trained neural network feature vectors, the condition of independence was sought to be imposed using the Random Forests. A feature importance measure and a feature survival measure was developed to identify important regions of the neural network.

These experiments provide insights on independence and separated information carried by feature inputs. These will be utilized in the Chapter 4 for improving the classification using combination of neural network architectures which either have separate properties or are trained on distinct data sources.

Chapter 4

Knowledge Inclusion for Better Neural Network Learning

4.1 Introduction

The Chapter 3 introduced the neural network as a combinational model of non linear function and a multi-scale feature extractor. It was also mentioned that the $\mathscr{H}$ of a data can be selected by choosing a model prior for the data and the noise distribution. Any feature transfer done on neural network presented in the Chapter 3 can be represented as a solution for better prior modeling [138]. This chapter will review a formulation for transfer learning in the Bayesian settings, present the Bayesian interpretation of Artificial Neural Networks (ANN)s and then present various experiments to demonstrate the influence of model priors on the learning process.
Priors are created on weights by the transfer learning methods used in neural networks. In neural networks transfer learning is the process of using the statistical information learned on one domain of a problem and reuse it on a new domain of

the problem. Deep neural networks require large datasets for training, it is assumed that the large amount of data and large parameters of neural networks create a generalization over a large data distribution, this creates a problem where some local minimas of optimization are reasonably good solutions [139]. Given the large data requirement for solving such tasks, transfer learning between task domains is desirable.

This chapter will present experiments focusing on classification by various neural network architectures. All classification tasks will attempt at minimizing the cost $\mathscr{C}$ given by the cross-entropy loss of the network estimates and negative log likelihood(NLL) loss for binary classification with 0 and 1 as output is, and multiclass(K) classification respectively is,

$$\mathscr{C}(f_{\mathbf{W}}(\mathbf{x}),y) = -(y\log(f_{\mathbf{W}}(x)) + (1-y)\log(1-f_{\mathbf{W}}(x)), \tag{4.1}$$

for multiclass classifier with input vector $\mathbf{x}$ and output vector $\mathbf{y}$ the NLL is written as,

$$\mathscr{C}(f_{\mathbf{W}}(\mathbf{x}),y) = -\sum_{k=0}^{K-1} \mathbf{y}_k \log(f_{\mathbf{W}}(\mathbf{x}_k)). \tag{4.2}$$

Thus the average NLL loss over a set of N data samples $\{X,Y\}$ is,

$$\mathscr{C}(f_{\mathbf{W}}(X),Y) = -\frac{1}{N}\sum_{i=0}^{N-1}\sum_{k=0}^{K-1} \mathbf{y}_{i,k} \log(f_{\mathbf{W}}(\mathbf{x}_{i,k})). \tag{4.3}$$

where $\mathbf{x}_{i,k}$ and $\mathbf{y}_{i,k}$ are the input and output vector values of the k^{th} component of the i^{th} element of the set $\{X,Y\}$.

The optimization of these neural network models is done using gradient back propagation.

This chapter deals with improving the neural network performance using prior informations. For this, three strategies are employed,

1. Introducing architectural changes in the neural network model for incorporating constraints forced by the physical models of the data,

2. Including the statistical and physical biases of the data into the neural network as weight priors,

3. Drawing neural network based systems such that they do task separation and such that their estimates can be corrected using task priors.

In this chapter, a bayesian interpretation for using a prior learning on the neural networks is first presented in the Section 4.2, a definition of the transfer-learning and the constraints this definition imposes is presented in Section 4.2. A brief introduction of the information inclusion strategies for neural network are described in Section 4.3, the physical model constraints, the weight priors and the task separation are introduced. Section 4.4 introduces a constituent learning procedure which is a proposal for intra-domain transfer learning.
The dataset used for experiments demonstrating these concepts is described in the Section 4.5. Section 4.6 introduces a hand-shape classification problem and the experiments are made with a hand model neural network training procedure for using badly processed data, and non-specific classes for better initilsation of weight priors is also presented in this section. Finally, it is demonstrated that weight priors learnt in one pre-processed dataset can be transferred to same dataset but pre-processed with another scheme. In the Section 4.7 the experiments on the intra-domain transfer learning method of Section 4.4 are made. Then the combination of the Section 4.3 and Section 4.4 is shown in the Section 4.7.4. These experiments conclusively demonstrate the advantages of the information combination.
The Separation of task and their eventual combination to impose task model constraints is discussed in the Section 4.8, the experiments are conducted to demonstrate improved performance on a hand-gesture classification task. Finally, in Section 4.8.2 a combination of task model and task priors enforced by a combination of neural network models is shown and proposed as an optimum solution for the hand-gesture problem.

4.2 Defining the Learning Process

Given the transformation $y = f(\mathbf{x}, \Theta)$, the linear regression can be formalized by Bayesian treatment. Assuming that the prior probability distribution over the parameters Θ is Gaussian distributed as $\mathcal{N}(\mu_o, \omega_o)$. If the conditional variance for $f(\mathbf{x}, \Theta)$ is ω_c, the relation with the posterior distribution $p(\Theta \mid \mathscr{D}, \omega_m)$ can be give as,

$$p(\Theta \mid \mathscr{D}, \omega_m) \propto p(\Theta \mid \omega_o) p(\mathscr{D} \mid \Theta, \omega_c). \tag{4.4}$$

For $\mathscr{D}$ of N observations and using the assumption of independent Gaussian distribution, the likelihood of observation $\mathbf{y} = y_1, y_2, y_3 ...., y_N$ is given by

$$p(\mathscr{D} \mid \Theta, \omega_c) = \prod_{n=1}^{N} \mathcal{N}(y_n \mid f(x_n, \Theta), \omega_c). \tag{4.5}$$

In case of classification the log likelihood $\mathscr{L}$ of Θ given the outcome is $\mathscr{L}(\Theta \mid y)$, the above method is used to solve for Θ such that the average log likelihood $\hat{\mathscr{L}}$ is maximized over the dataset $\mathscr{D}$. Given a neural network $f_{\mathbf{W}}(\cdot)$ with $\mathbf{W}$ representing the multidimensional parameters of the neural network. The Bayesian approach for linear regression can not be directly applied for the neural networks. The multiple non-linearities introduced by the squashing functions at the neural network layers create non-linear dependencies between $\mathbf{x}$ and $f_{\mathbf{W}}(\mathbf{x}) : \mathbf{x} \in \mathscr{D}$. A closed solution and an exact convex optimization of the log loss of neural network output is not possible. Thus the posterior distribution is no more a Gaussian distribution.

To overcome these constraints [140, 141] proposed an analogy between hyper-parameter optimization in neural networks and the Bayesian optimization for simple linear regression. The non-linear nature of the neural networks and necessitates assumptions of small posterior distribution variance in comparison with the $\mathbf{W}$. This assumption allows a posterior in the Gaussian form.

Transfer Learning

The classical machine learning algorithms solve isolated problems and methods for identifying and transferring the task knowledge are not very popular. Separate tasks have different decision boundaries and formal methods for modification of the boundaries are hard to define. But, when a learning agent learns a certain task, it should be able to apply the knowledge gained from the learning to a distinct new task. This process of transferring information from one task to the another is described as transfer learning.

In [142] transfer learning has been defined as maximum *posterior* adaption of maximum entropy models. A existing model which has been trained to be maximum information model is adapted such that the adapted log likelihood is maximized. During the adaption phase the prior distribution of the parameters is centered at around a mean μ_0 estimated from the background data. For a feature-set $\mathbb{F}_b$ learnt on a background task, [143] explains the process of knowledge transfer to feature-set $\mathbb{F}_a$. The overall probability estimation when the transfer learning is made between the dataset $\mathscr{D}_s$ and $\mathscr{D}_t$ is thus given by the relation between the two datasets, the datasets can be weighted by a weighing factor w such that for a pair of input and label x, y, it is defined as,

$$w(x) = \frac{p(x, y_t \in \mathscr{D}_t \mid \omega_c)}{p(x, y_s \in \mathscr{D}_s \mid \omega_{c'})}. \tag{4.6}$$

for the known variance ω_c and $\omega_{c'}$. Eventually, given $p_S = p(y_s \mid x_s)$, the $p_T = p(y_t \mid x_s)$ is desired which can be estimated using the proportional relationship,

$$p_T(y \mid x_t) \propto w(x_t) p_S(y \mid x_t) p_S(x_t). \tag{4.7}$$

Here, y_s and y_t are the labels for the dataset $\mathscr{D}_s$ and $\mathscr{D}_t$ respectively. And, y is the output of the model. It can be argued from (4.6) that this formulation of transfer learning enforces similarity of the data sources.

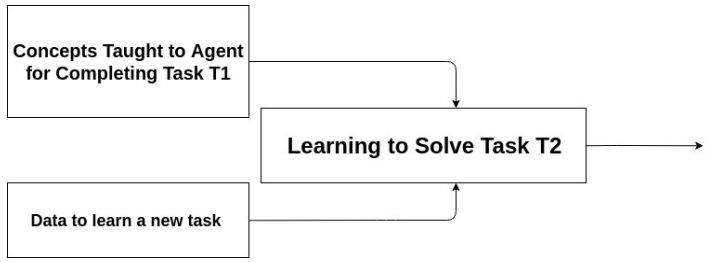


Figure 4.1: The concept for learning from priors and the data.

4.3 Neural Network Priors and Constraints

The Bayesian analogy for the neural network optimization allows interpretation of the pre-trained network weights as biases and neural network architecture as constraints. This assumption allows various methods for inculcating model informations while training. A short introduction to the concepts that form the basis for the experiments in this chapter is presented in this section. The earlier mentioned three strategies can be enforced by architectural and weight priors. This is explained in some detail here, before the experiments are described.

4.3.1 Weight Priors

The classical transfer learning as described in the Section 4.2 creates a constraint of similar data-space for two datasets. A method for transfer learning over data with different property domains is proposed in the next Section 4.4. It is done by finding a common space on which certain properties of both datasets are mapped. It is proposed that data from, different sensing modalities like the RGB camera, depth camera or Infra-red cameras, recording the same object or event can be used to initialize neural networks for problems on other dataset.

4.3.2 Architectural Constraints

The probabilistic interpretation of the neural network allows us to think of the initialization of the connection weights and the choice of network model as a prior. The training procedure modifies the weight prior in the process of optimization. A constraints set by the neural network architecture as in [92] and [93] changes the learning behavior. It is argued here, that the nature of data can be encoded into the architecture of the neural network. In the experiments later, it will be demonstrated that this may improve the performance of a neural network.
While a physical model of data introduces a bias, a similar approach can be used by enforcing the known-information about the separate properties that form the components of a task for better learning. It will be demonstrated that separation of independent learning tasks and their combination thereafter, creates a system that performs better than a single network.

4.4 Constituent Transfer Learning

Domain adaptation is an important field of machine learning but apart from few recent works based on adversarial networks [144], [145] their are only few attempts for using neural networks for attempting to solve the tasks like domain adaptation and mapping described in [146] and [147] are few. Here, inspiration from these works is used to produce solutions on neural networks.
Let us begin with considering two datasets $\mathscr{D}_\alpha$ and $\mathscr{D}_\beta$ from two sensors measuring on separate principles. The two datasets have recorded either the same or a similar events (e.g. a hand shape). Now, let us assume that the data type on which we wish to solve the problem is of the kind that matches $\mathscr{D}_\beta$ and the data type $\mathscr{D}_\alpha$ is available in larger quantity. The problem is now to find a mapping on the data that

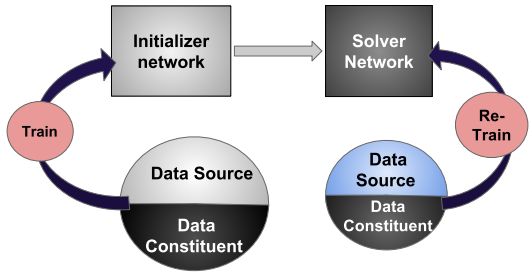


Figure 4.2: The description of the proposed transfer learning, where the data constituent remain the same but source changes.

extracts useful features from $\mathscr{D}_\alpha$, such that a network learned on this representation can be used as warm start for a faster learning progress on $\mathscr{D}_\beta$, than if it would only be learnt on $\mathscr{D}_\beta$ (from scratch). The principle is also illustrated in Figure 4.2.

Earlier in Chapter 3, it was mentioned that the concepts $\mathbf{O} \subseteq \mathscr{O}$ explaining $\mathbb{P}(\mathbb{X})$ are useful for understanding $\mathbb{P}(\mathbb{Y}|\mathbb{X})$, in this section this property of the latent variables will be employed for an intra-domain transfer learning. The properties derived from the unsupervised learning of image dataset $\mathscr{D}_\alpha$ and the reasoning in the model which are common to both $\mathscr{D}_\alpha$ and $\mathscr{D}_\beta$ are used for transfer learning. The training procedure is developed such that the constituents of the image models can be transferred over independent recording domains, with similar models.

Given a set of property which is represented by a numerical proxy $\mathbb{C}$. This numerical representation lies on the same space $\mathbb{R}^N$, and to which both $\mathscr{D}_\alpha$ and $\mathscr{D}_\beta$ behave similarly. This property is used to learn a mapping from $\mathscr{D}_\alpha$ to $\mathbb{C}$. Learning this mapping provides an initial point for optimization to solve a new problem on the $\mathscr{D}_\beta$. This can be formalized as:

Given that the $\mathbf{c}$ is vector of matrix $\mathbb{C}$ which is the numerical representation of the properties of all samples in $\mathscr{D}_\alpha$, then a f_p is learnt such that,

$f_p(x) = \mathbf{c} \forall x \in \mathscr{D}_\alpha$,

Eventually, a mapping f_m between the elements of $\mathscr{D}_\beta$ and the corresponding task labels $\mathbf{y}$ is learnt, such that,
$f_m(f_p(x^{'})) = \mathbf{y} \forall x^{'} \in \mathscr{D}_\beta$

4.4.1 Constituent Learning

This section describes an unlabelled initialization for transfer learning. It is specific to data describing events composed of similar *constituents*, i.e. similar events, described via different modalities. The reasoning for the choice of various data embedding schemes used for training an initializer neural network are presented. Finally, a similar scheme with labelled data is presented. The principal objectives of constituent learning are

- Learning a representation of data properties.
- Data properties that remain consistent over source and target recording schemes should be captured.
- The learned features generalize well to new tasks.

Unlabeled constituent learning

A neural network $f_{\mathbf{W}}(\cdot)$, when trained to perform classification or regression task, maps the input data sample to an independent output space. A classifier learns to compose discriminatory features. Some networks magnify or suppress features depending on the task targets. It is proposed to train the neural network to learn the constituent properties of the problem space. This is done by training a neural network to learn a mapping for the *target representation* of the data.The neural network $f^p_{\mathbf{W}}(\cdot)$ thus learns the mapping f_p on $\mathscr{D}_\alpha$. Such representations are conducive

to transfer learning when the data-sets are similar in some physical constitutions but originate from different sensor sources. More specifically, the weights w of the $f^p_{\mathbf{W}}(\cdot)$ learning f_p are trained such that the following loss function L over the dataset is minimized,

$$\min_{w} L(f_p(X), f^p_{\mathbf{W}}(X)), \tag{4.8}$$

$X \subset \mathscr{D}_\alpha$ is the combined representation for a subset of elements from the $\mathscr{D}_\alpha$ of size n. $f_p : \mathbb{R}^{\mathrm{D}} \longmapsto \mathbb{R}^{\mathrm{d}}, \mathrm{d} << \mathrm{D}$ is the accurate transformation which creates a descriptor for the dataset and transposes the data to Y. A squared l_2 distance is used as a loss function. It is recommended that while using an unbounded loss function, both the output activations of the network and the target vectors are normalized to a unit sphere before the loss calculation. Effectively, the loss function $L(\cdot)$ is,

$$L(X) = \frac{1}{n}\sum_{i=1}^{n} l^i, \tag{4.9}$$

with

$$l^i = \|\overline{f_p(x^i)} - \overline{T(x^i)}\|^2 \tag{4.10}$$

and $\bar{x}$ is the spherical-normalization operator and x^i is the n^{th} element of set X.

Choice of target representation

The mapping that creates the target representation should

- Preserve the properties which are consistent over the datasets.
- Try to conserve information.
- Diminish the variance caused due to outliers or noise.

If the transformation f_p acts on the input X such that $Y = f_p(X)$. The choice of f_p is to be made such that it assists the network in learning self-contained properties of the data rather than the correlation between the input X and the output Y in some

independent target space. Apart from this property, a low dimension projection scheme such as Principal Component Analysis (PCA) transformation satisfies the earlier mentioned requirements of alleviating noise while simultaneously preserving transferable properties and the information held in the data. PCA also handles the normalization problem as the data is spherised during transformation process, also the PCA optimization does not involve local minima [148]. This contributes to the important property that the transformation f_p is constant for the dataset and does not depend on initialization of the algorithm or the order in which data is presented to the algorithm. A neural network, however is capable of non-linear transformations and training the network to mimic a PCA transformation forces a linear mapping constraint on models capable of learning non-linear transformations. The Locally Linear Embedding (LLE) [41] for dimensionality reduction also employs a non-linear optimization and satisfies requirements discussed earlier. Thus, the two transformations used for training an initializer neural network are the PCA and LLE. The low dimensional representation from these is the numerical representation of the constituents of the data.

Labeled constituent learning

Learning a neural network with constituent information about the data is also possible with a scheme where the network is trained with labels that represent self-contained properties of the input data. This necessitates datasets with ground-truth. Though, it is important that the ground-truth could be approximately modeled entirely from the input data. The model, albeit, may be extremely complex.
Later in an experiment with the hand skeleton structure it is stated that, the skeleton structure of a hand is a valid low dimensional representation and the location vector of the skeletal points relative to the palm center is a self-contained property. The skeletal point vector of the hand is the numerical representation $\mathbf{c}$ which represents

some property of hand shape. The neural network $f^p_{\mathbf{W}}(\cdot)$ in this case will learn the mapping f_p which is a function that projects image input to skeletal point locations.

4.4.2 Constituent Learning Procedure

The classical transfer learning scheme is such that, an (*initializer network*) is first trained either in a supervised or unsupervised manner on a large database ($\mathscr{D}_\alpha$). The trained network is then re-trained to solve a separate problem or to work with a new dataset($\mathscr{D}_\beta$) of the same kind, or both. If the initial neural network acts on the input as $f^p_{\mathbf{W}}(\cdot)$, the retraining process modifies the neural network to $f^m_{\mathbf{W}}(\cdot)$, which is subsequently called *solver network*.

Let two datasets $\mathscr{D}_\alpha$ and $\mathscr{D}_\beta$ be recorded from two sensors of different modality. The target is to use $\mathscr{D}_\alpha$ to improve the performance of a solution on $\mathscr{D}_\beta$. Thus, properties on $\mathscr{D}_\alpha$ that can be transferred to $\mathscr{D}_\beta$ are identified. A neural network is trained to mimic the property of the model $f_p(\cdot)$ on the dataset $\mathscr{D}_\alpha$. Following the conventional transfer learning technique, the output layer of the neural network trained on $\mathscr{D}_\alpha$ is modified and the network is re-trained with $\mathscr{D}_\beta$ to solve the target problem. The procedure is described in **Algorithm** 1.

To create the initializer, the neural network, which in a crude sense is a very high dimension regressor, is trained to model a function. In such scenarios, it is highly likely that many weights of the neural network collapse to very small values. These weights of the network may then remain unaffected during the training process. To discourage this behaviour a random Gaussian noise is added to the target representation. The variance of the Gaussian noise is sampled from a exponential power distribution.

The initializers explained in the Section 4.4.1 when trained are approximations of property conservation projections. It is important to remember that on re-training with a new dataset for a new task the learnt projection function $f_{\mathbf{W}}(\cdot)$ is itself mod-

ified to $f_{w'}(\cdot)$.

4.5 Dataset for Experiments

4.5.1 Hand-Shape Dataset for Model Experiments

The next sections will discuss using the model constraints and model priors for the classification of hand-shapes. A hand-shape dataset has been recorded for these experiments, the data is recorded on a desk setup. Wrist onwards the hand has high degree of freedom. A hand can thus form various signs and symbols, some of these poses are naturally used for communication. Of these possible symbols six poses are defined and recorded as the top view of the hand. Five of the poses are 'Fist', 'Flat', 'Joined', 'Pointing', and 'Spread'. The 'Fist' is a closed fist hand with palm facing downwards. Pose 'Flat' is when the palm is open with the four fingers joined together. 'Joined' is when the hand is conically shaped and points downwards with all fingers touching each other. 'Pointing' is the index finger pointing forward. Finally 'Spread' is an open palm with fingers spread apart. Further, a class of hand-pose in the places where the hand transitions from open to close are recorded. This class can have different uses. It can be identified as class of unintended poses or one that helps describing transitions of pose in a gesture.

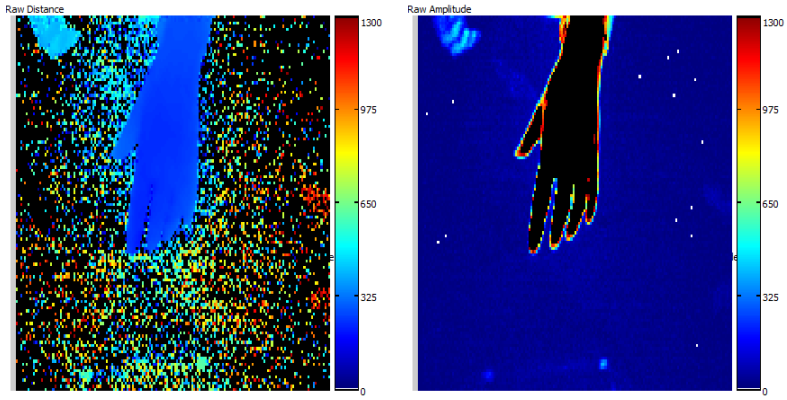
Data Recording and Segmentation

A large dataset of hand-pose is collected using a 3D-ToF camera the 'pmd camboard nano'. The datapoints are 16 bit two channel images of dimension $120 \times 165 \times 2$, Figure 4.3. The first channels of the matrix represent the amplitude of the reflected ray received by the camera and the second channel are the range values of the respective pixels, expressed in millimeters.

Algorithm 1: Procedure for training the initializer network and using it to train a new network.

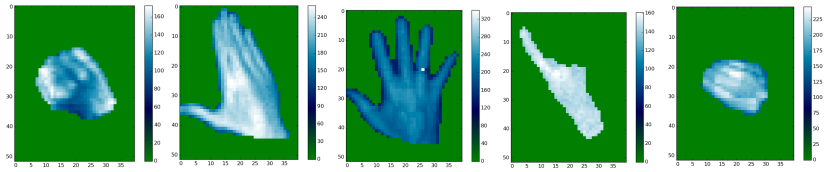
input : The Dataset $\mathscr{D}_\alpha$
input : The Dataset $\mathscr{D}_\beta$
input : Labels $\mathscr{T}_\beta$ for $\mathscr{D}_\beta$
network: neural network with random initial weights.
output : Initializer network trained to approximate $T(\cdot)$
output : neural network trained to complete the task on $\mathscr{D}_\beta$

```
/* Training Initializer network.                    */
for i in Initialepochs do
    for dataPoint in D_α do
        /* Generating targets.                      */
        targ ←CalcProjection(dataPoint);
        targ ←targ +N(0,σ);
        pred ←ForwardNN(dataPoint,iniParams);
        loss ←CalcLoss (pred,targ);
        iniParams ←BackProp (loss, iniParams);
    end
end
iniNet ← SaveNetwork (iniParams);
/* Training solver network.                         */
params ← GetParams (iniNet);
for i in epochs do
    for dataPoint, label in D_β,T_β do
        pred ←ForwardNN(dataPoint,params);
        /* Using groundtruth targets.               */
        loss ←CalcLoss (pred,label);
        params ←BackProp (loss, params);
    end
end
usedNet ← SaveNetwork (params);
```



(a) Raw Distance Data. (b) Raw Amplitude Data.

Figure 4.3: The hand shape channel sample.

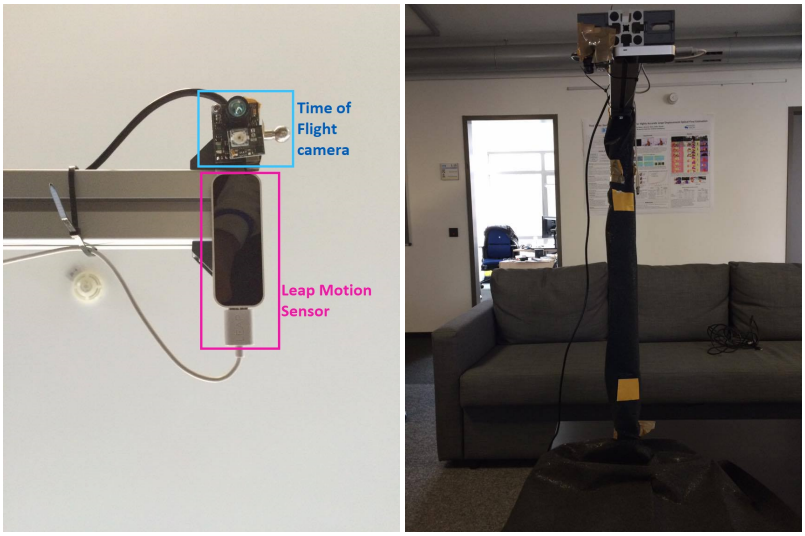


(a) Fist (b) Flat (c) Open (d) Point (e) Join

Figure 4.4: Sample of the processed and normalized Hand shapes.

Recording Setup

The data is recorded within a cuboidal space with varying heights. The ToF camera is mounted vertically above the recording region. The furthest vertical range is marked by a table top. The height of the camera from the table varies between 400 and 800 mm. The closest vertical approach to the camera is marked at 150mm from it. While recording the participants were asked to wrap an absorbing cloth on their arms, Figure 4.5.



(a) The recording ToF sensor with a Leap Motion

(b) The recording set-up device and the sensor housing

Figure 4.5: The recording setup for the on desk recordings.

Recording

Twelve participants were recorded for pose and gestures. Each participant keeps his palm as one of the defined poses, and randomly but not abruptly moves the palm within the virtual cuboidal space. This is recorded for two minutes, for all six poses. Such recording of the data adds variances in depth and variances of hand orientation in the horizontal plane. The participants are also asked to rotate their palms to add the angular variances in the vertical plane.

The recorded participants have varying skin textures and palm sizes, some of the participants are recorded while wearing rings.

Segmentation

The absorbing cloth wrapped on the arms of the participants assists in hand segmentation by thresholding the amplitude channel of the image. The reflectance constraint does not entirely remove the background and thus the closest contour greater than a threshold area is chosen as the palm region. The segmented palm region is then converted into a binary image which is used as a mask for both channels. The resulting image is a two channel 16-bit image of the palm isolated from the environment. After segmentation The depth channel values for the background are set to a fixed maximum-depth and the amplitude values are set to 0. The basic processing after segmentation involves re-normalisation of the segmented hand pixels to values between zero and two hundred fifty five.

4.5.2 Gesture Data and Features

Further, in this chapter the next sections will also discuss using the model constraints on time series classification problem. A dataset recorded in the car to solve a hand gesture classification problem has been recorded and will be used to demonstrate these experiments. In this section the definition of the gestures and the recording setup is defined.

A hand-gesture is a sequence of frames of moving palm. It can involve motion of palm without change in the hand-pose or it could be defined as a sequence of hand-poses where the occurrence of the different hand-poses have a predictable, predetermined order. For this work the recorded hand-gestures include, 'Clicking', 'Swiping' in Left and Right direction and in Up and Down motion, 'Accepting', 'Declining', 'Drop' and 'Grabbing'. 'Clicking' involves a forward horizontal motion of the pointing finger. Hand motion in horizontal left-right direction is denoted as 'Swiping' in left and right direction. The swiping motion may be repeated more

than once. Similar vertical palm motion is denoted as vertical swiping. 'Accepting' is a motion of hand outwards from the screen (relative to the camera) . 'Declining' is the motion of a hand into the screen. 'Grabbing' involves a transition of a spread hand with the palm facing vertically downwards to a position of joined fingers accompanied with some vertical motion. 'Drop' begins with joined fingers ending in a spread hand with a short downward motion.

4.5.3 Data Collection and Properties

The output frames from the camera have two channels, the depth and the amplitude. The amplitude value of the pixels are proportional to the reflectance of the surface and inversely proportional to the square of the distance values. The data is recorded with a frame rate of 25 frames per second.
We use a Photonic Mixer Device (PMD) Nano sensor with a resolution of 120×165 pixel for recording data. This ToF based 3-D camera is attached to the rear-view mirror holder protection. The experiments for hand gesture recognition inside the car are conducted with seventeen participants. The data is recorded inside the car and each participant repeats nine gestures around the sat-nav screen of the car. Every participant repeats each gesture six to twelve times. Each frame of the sequence is marked with two labels. 'Accepting', 'Declining', 'Drop', 'Grabbing', 'Clicking', 'Horizontal', and 'Vertical' are used as the primary labels. Sequences marked as 'Horizontal' are marked with a secondary label 'Left' and 'Right', and those marked with 'Vertical' are marked with secondary labels 'Up' and 'Down'.

Training and testing sampling

Before training the network, the data was shuffled such that the frames from a complete gesture sequence stay together, while the gesture sequences were placed ran-

Table 4.1: Number of gesture class samples in dataset

	Up	Down	Left	Right	Click	Accept	Decline	Grab	Drop	TOTAL
data-points	220	226	247	247	160	188	194	172	160	1814
down-sampled	44	45	49	49	32	47	48	34	32	380
Total	264	271	296	296	192	235	228	204	192	2194

domly. This shuffling was essential because the participant continuously repeated the same gesture multiple times during recording. Each frame of the gesture sequence is marked with the label for the entire sequence. This allows us to train the network in a way such that it attempts at predicting the sequence-label from the start of the gesture.

The total number of available sequences for training the model are increased by sub-sampling approximately one-fifth of sequences in time. Equal proportion of sequences from each class of gesture are reduced to half duration. Such down-sampling effectively creates sample-points on which the duration for completing a gesture is shorter than the average gesture sequence. The start and end of each sequence including the sub-sampled once are marked. Both training and testing phase of the algorithm use these sequence markers. Table 4.1 gives a description of the distribution of the data-samples over classes and the number of sub-sampled sequences created for each class.

For testing a leave-two cross-validation was performed on the data. This was completed by training the dataset collected from all but two sets recorded subjects and was tested on the two subjects left out during training. The data from the seventeen participants was distributed into eight sets of two participants and one set of one. The average accuracy is reported on nine-fold repetition of cross-validation. The set with only subject when used as test data uses a leave-one cross-validation. The sub-sampled sequences are separately divided into nine groups and then used in training and testing accordingly.

Table 4.2: Description of features used for the experiments

Type	Feature Names	Description
Location	Finger Coordinates	The X,Y,Z coordinates of the tracked pixel closest to the screen.
	Hand Coordinates	The X,Y,Z coordinates of the tracked palm centroid.
	Finger Azimuth	Polar angle of the Principal component vector of the finger cluster of the palm.
	Finger Polar	Azimuth angle of the Principal component vector of the finger cluster of the palm.
Velocity	Finger Velocity	The X,Y,Z components of the tracked pixel closest to the screen.
	Hand Velocity	The X,Y,Z components of the tracked palm centroid.
Shape	Concave Depth	The maximum distance between convex hull and edge of the segmented palm region.
	Convex Ratio	The ratio of the size of the convex hull around the palm and the segmented palm region.
	Active Pixels	The number of pixels in the segmented palms provides an indication of palm-size.

4.5.4 Segmentation and Feature Extraction

These experiments are conducted by identifying the hand gesture by feature extraction. These features are calculated on the palm region segmented by creating a virtual cuboidal space in the region where the hand-gestures are observed.

A background was generated by recording a video in the car and keeping the consistent pixels. This 3-D background image was then removed from each incoming images of the video sequences. Furthermore, the palm pixel closest to screen is tracked. Hand region is segmented by calculating the largest connected contour and, another a threshold is used for palm-hand segmentation. This threshold is estimated by by K-means clustering on a sample of data. The Mahalanobis distance is used as measure for calculating the K-means clusters for palm-finger segmentation. The hand palm centroid and finger-tip are estimated and tracked using a Kalman

filter. Features are further described in Table 4.2. The features were centred and normalized such that the mean of each feature element over the training data was zero and the variance was unity.

The foremost location of the palm was tracked in the segmented video frames. The coordinates of the palm centroid were estimated. Moreover, the azimuth and polar angle of the extended fingers were calculated by covariance matrix analysis, via principal component analysis. The number of active pixels after completion of the segmentation procedure was an indication of the palm shape. The convex hull and concave depth of the palm, calculated at every frame,describes the overall hand shape in the gesture sequence. Motion descriptor features include velocities of the foremost finger and hand centroid in all three dimensions. Thus, a seventeen dimensional vector was used to describe the palm shape and motion at each frame.

4.6 Physical Model Constraints

In this section a set of experiments with the constraints of the physical model of the data introduced to the neural network are explained. A neural network is designed and trained; it is compared against a modified network with reduced connection but including information about the degree of freedom of the hand shape. This section describes various experiments conducted on the same hand-shape dataset and the same network with the three pre-processing methods. The hand shape classifier is a neural network trained on the Hand Shape dataset described earlier.

The hand-pose classification network is a sequential neural network. The selected architecture has four convolution layers followed by four fully connected layers which perform inner product. Each layer is connected to a Rectified Linear Unit (ReLu) which adds non-linearity to the network. A convolution layer is connected to the input data. The output of the top three convolution layers is pooled by max-pooling strategy. The second and third pooling layers also sub-sample the output

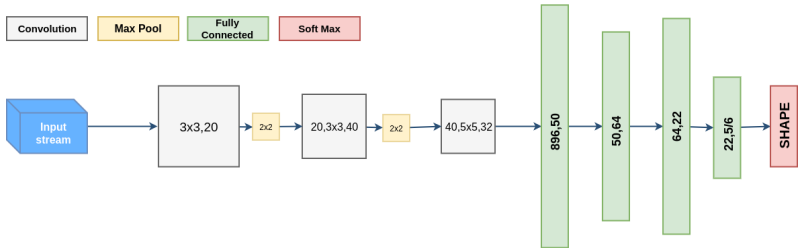


Figure 4.6: The CNN model with hand model constraints

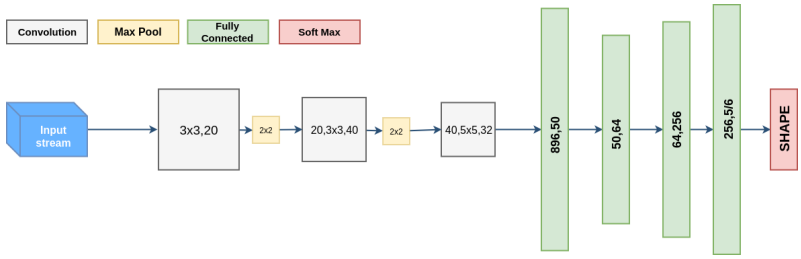


Figure 4.7: The larger CNN model without hand model constraints.

Figure 4.8: The neural network models trained for classification of the hand shapes.

image of convolution layer by a factor of two. The output layer for the network is a fully connected layer followed by a softmax function. The strategy for creating a neural networks model that solves a task on the hand model is designed such that a part of the neural networks unit is used for learning some properties the hand model neural networks $f^{h}_{\mathbf{W}}(\cdot)$. Further, additional layers are appended to the model unit this creates a new neural networks that attempts to solve task with imposed soft constraints of hand model. This new model is represented by $f^{ht}_{\mathbf{W}}(\cdot)$.

The hand models used for hand-wrist replacements have often used a twenty-two degree of freedom [149] model based on joint locations. In the model trained for the classification of the hand-shapes the model unit used is based on these fixed number joint model of hand. The combined state of the joint locations of the hand

provides information of the hand shape. The hand-model information is included in the network by adding a twenty-two node layer as one of the outer layers of the network. This creates a funnel shape in the network and forces the network to learn from a small dimensional representation of the input images.
In the first set of experiments a model enforcing layer is trained simultaneously with the classifier. Later, in the chapter, the experiments showing model information transfer will detail a solution on transfer learning using model information.

4.6.1 Training the Hand Model Network.

As described in the Section 4.5.1 six shapes are recorded to understand the hand-gesture, one of the classes is a transition shape which is inconsistent over various people. Though, eventually the model is trained to classify five classes, it is trained with all six classes to initialize the weights using a relatively larger dataset. The gradient descent optimization is completed by using the NLL loss.
The described training procedure uses the concept of transfer learning on the proposed model.

- The model $f^{h}_{\mathbf{W}}(\cdot)$ is trained on the dataset $\mathscr{D}$ with six hand shape classes, including the poorly segmented and incorrectly labelled data.
- A subset of the dataset $\mathscr{D}_s$ is selected. $\mathscr{D}_s$ includes five distinct hand shape classes, and carefully segmented and labelled data is used.
- The weights from $f^{h}_{\mathbf{W}}(\cdot)$ are used to initialize $f^{ht}_{\mathbf{W}}(\cdot)$ which is re-trained on $\mathscr{D}_s$.

In the first phase of training which uses all six classes of data the network wights are initialized by xavier initialization [150]. Of the six classes, "Point", "Join", "Open", "Fist", "Flat", and "Transition" the resulting classifier $f^{ht}_{\mathbf{W}}(\cdot)$ is trained and

tested on the first five. All the channels of the input image are normalized over the dataset to [0,1]. This normalization is done by recording the maximum amplitude value for valid pixels in the dataset. The maximum value for the depth defined during recording is used for depth channel normalization. Normalization is done while masking background pixels.

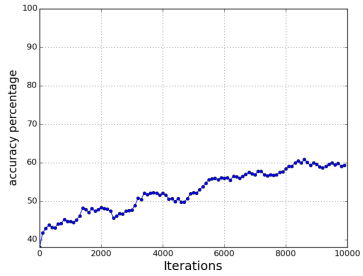
Experiment on the raw data

The first set of experiments are conducted on a two-channel input image of the dataset $\mathscr{D}$ and $\mathscr{D}_s$. The second channel provides the depth information which assists the network in assimilating the scale variations. Both channels are mean subtracted independently.

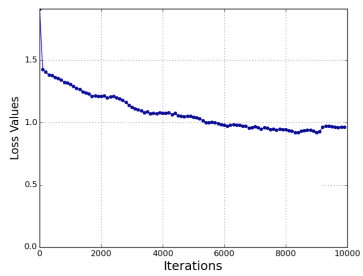
As described earlier in Section 4.6.1 the training is performed in the two. The training progression on the test data for the first phase with six class and poorly selected inputs is shown in the Figure 4.9(a), the second phase of training is shown in the Figure 4.9(c). The resulting trained network $f^{hr}_{\mathbf{W}}(\cdot)$ provides an average accuracy of 82%.

Experiment on amplitude images

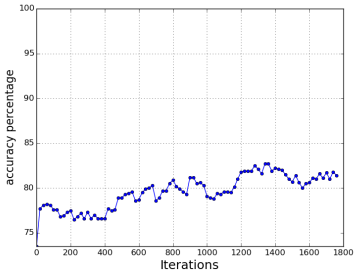
The same procedure as described in Section 4.6.1 is conducted on a single channel input of the amplitude values to train the neural network $f^{ha}_{\mathbf{W}}(\cdot)$. Using the depth channel of the image the amplitude channel is projected on a plane at a fixed distance from the camera. This, makes the input images independent of the scale.



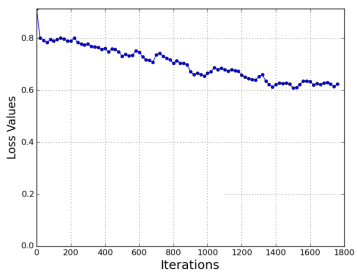
(a) Accuracy progression for training stage 1.



(b) Loss progression for training stage 1.

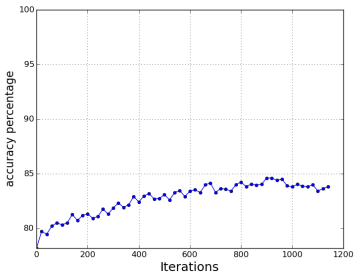


(c) Accuracy progression for training stage 2.

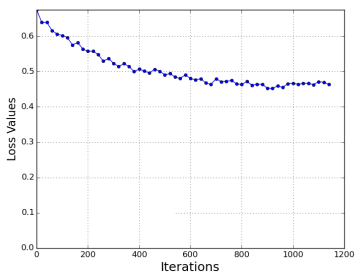


(d) Loss progression for training stage 2.

Figure 4.9: Training progression for training with two-channel images

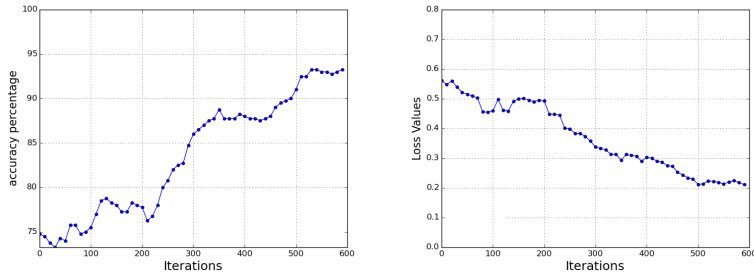


(a)



(b)

Figure 4.10: Training progression for training with amplitude images. 4.10(a): Accuracy stage 2. 4.10(b): Loss stage 2.



(a) Accuracy propogation training stage 2.

(b) Loss propagation training stage 2.

Figure 4.11: Training progression for training with binary images.

Experiment on binary images

The experiments are also carried out on the binary images extracted from the scaled amplitude images. The neural network $f^{hb}_{\mathbf{W}}(\cdot)$ is tested and trained. The best classification performance on the model was 90%. The better performance of the binary images can be attributed to the binary nature of the data. The intensity values of the pixels of a ToF camera depend on the reflectance of the surfaces and the incidence angle of the active light these factors contribute ambiguous information to the amplitude channel.

4.6.2 The Gains from the Model Constraint Imposition

To validate the advantage of the model constraints, the network is compared against another network with the model enforcing layer replaced by a larger layer Figure 4.7. The Datasets $\mathscr{D}$ and $\mathscr{D}_s$ are used in the scheme as described earlier. The performance of the raw, amplitude and binary image dataset is compared on both the original and model-enforcing networks. The Table 4.3 shows the comparison between the model with the funnel shape created by hand model and the standard model. The average performance of the hand model network exceeds the perfor-

Table 4.3: Comparison of the Hand Model Network with the larger CNN on the three kinds of datasets

Data Shape	Raw		Amplitude		Binary	
	HandCNN	CNN	HandCNN	CNN	HandCNN	CNN
Fist	84	84	87	95	87	86
Flat	83	80	87	95	89	87
Open	82	82	82	95	91	90
Point	86	85	88	95	91	91
Join	79	82	81	95	93	91
Dataset	**83**	**83**	**85**	**83**	**90**	**89**

mance of the network with a larger output layer described in Section 4.6.1.

Feature transfer on distinctly preprocessed dataset

The two stage learning uses the ability of a CNN to transfer learnt features over separate problems. Experiments were conducted to test if this similar to step training procedure can also be used with networks which were trained on the same data with different pre-processing procedures. The experiments were conducted on the corrected, five class hand data set $\mathscr{D}_s$ the background was isolated. Effectively, The two stage training was replaced such that the first network was trained on the binary images to classify the five classes. The trained network was used to transfer weights to the a network initialized to classify the amplitude images.

Two neural networks $f^{ha}_{\mathbf{W}}(\cdot)$ and $f^{hb}_{\mathbf{W}}(\cdot)$ were trained simultaneously with the amplitude and binary images $\mathscr{D}^a_s$ and $\mathscr{D}^b_s$. The networks were initialized with the same random weights, the images were presented to the both neural networks in exactly the same sequence and the range of values of the image pixels in both the datasets was $[0,1]$. Once the two networks were separately optimized, an amplitude normalized image was fed into the neural networks $f^{ha}_{\mathbf{W}}(\cdot)$ and $f^{hb}_{\mathbf{W}}(\cdot)$ respectively. The mean channel output at each layer were calculated and compared. The comparison

showed that the mean difference calculated in the euclidean sense for the first pooling layer output was of the order of 10^{-6}, which increases to 10^{-3} for the second pooling layer and 10^{-2} for the output convolution layer.

These experiments indicated the applicability of the feature transfer in the given settings. The difference was considerably larger for the fully connected layers which were closer to the output. This indicates that the filter weights learnt by the network for the first cases are general and can be reused for the training. The property of feature transfer was tested by employing the model trained with binary images obtained in last section to directly test the amplitude normalized dataset. In this experiment a test accuracy of 75.4% was achieved. When the same model was used as an initialization for the second phase training of the amplitude normalized images the accuracy of the amplitude images crossed 80% within 500 batch-iterations, Figure 4.12.

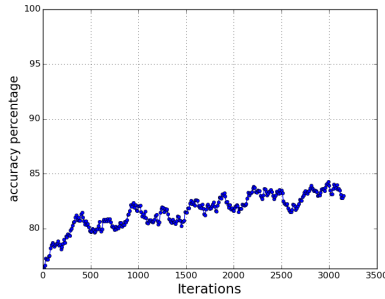


Figure 4.12: Training Accuracy: Binary Image Dataset Fine Tuned with Amplitude Image Dataset.

(a) MNIST Sample (b) SVHN Sample

Figure 4.13: Image sample from the two digit datasets.

4.7 Constituent Learning for Weight Priors

To validate the scheme proposed in the Section 4.4 and to demonstrate the usefulness of the constituent learning method. In this section, three sets of experiments are performed. These experiment also show the learning of weight priors.

4.7.1 Digit Low Dimension Representations to Digit Classifier

The first set of experiments were completed on the MNIST and the SVHN dataset. MNIST has hand written numerals and SVHN dataset has house numbers. The MNIST dataset is centered, scaled and anti-aliased version of binary images. The anti-aliasing changes the binary images to grey-scale. In our experiments we convert the MNIST data samples to binary images by thresholding. Similarly, the RGB SVHN dataset is converted to grey-scale images. One sample of each dataset is shown in the Figure 4.13. The MNIST data is already centered while the SVHN dataset provides boxes of size 32×32 around the target digit.

(a) The original sample (b) Hand depth image (c) Hand Binary image

Figure 4.14: Image sample from the Hand dataset.

In this experiment, the binary MNIST data was used to train a digit-initializer network to solve SVHN classification. The architecture in [151] was used for completing the experiments. Two such networks were trained, one with PCA, the other with LLE, as the transformation function. These initializer networks were then used to initialize the parameters for training the solver network by the gray-scale SVHN dataset. The outer 64 node layer is replaced by a 10 node classifier layer. The SVHN data is now used to train the network data to solve the digit classification problem on the SVHN dataset. Similar experiments are conducted with LeNet [152] initialized by a subsample of gray-scale SVHN used to train the binary MNIST.

4.7.2 2-D Hand Pose Locations to Hand Shape Classifier

The dynamic hand gesture dataset of [6] provides depth images and skeletal points of a hand. The one and two finger hand shape of the gestures pinch, rotate, swipe up and left, are used to create a ten class shape-classification problem to validate the labeled-constituent learning explained in Section 4.4.1.

A hand binary dataset (Figure 4.14) is generated from the depth images. A neural network which resembled the single-stream of [153] was trained with those images as input and the x, y coordinates of the skeletal points as labels. Thus, the output

Table 4.4: Digit Classification Accuracy :Two Initializations.

Init	MNIST to **SVHN**		SVHN to **MNIST**	
Type	Accu %	Iteration	Accu %	Iteration
Random	92.77	100,000	98.09	100,000
PCA	90.76	60,000	96.57	60,000
LLE	92.65	10,000	97.03	60,000

Table 4.5: Hand Classification Accuracy: Regression Initializer.

Iteration	60,000	70,000	80,000	90,000
Accuracy	79.96	79.76	81.64	81.88
Accuracy for Randomly initialized Classification				81.32

layer of the network had 44 nodes. The coordinates of the skeletal points are relative to the center of the hand. Noise is also added to the skeletal points. The variation of the hand location in the images of the dataset is not large, to better learn skeletal position relations with respect to the hand, the hand location in the training set is often randomly inside the image area. The resulting hand-pose network trained on the binary hand images is then modified to behave like a shape classifier for the depth images. The fourty-four node output is changed to ten node softmax layer and the network is re-trained with the depth dataset.

4.7.3 Results of Learning with Constituent Transfer

In order to measure the quality of transfer learning, the neural network parameters are frozen and the outer MLP layer weights are initialized with norm-random variables.

The initializer neural network is trained for a fixed number of iterations with unlabelled data to learn the target transformation, then the solver network is initialized from the initializer network. The accuracies are reported after at various iterations.

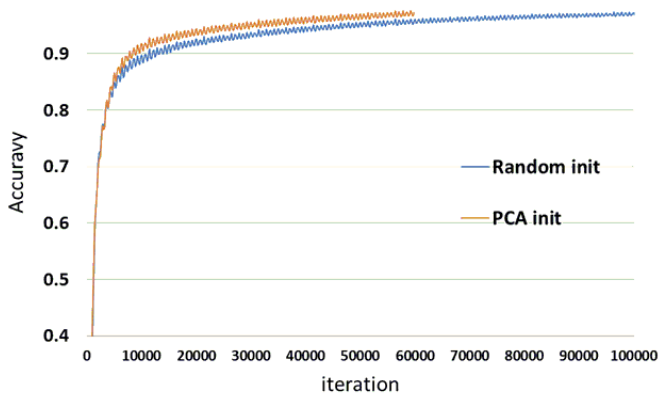


Figure 4.15: Progress of the MNIST classification with and without initialization.

For comparison a randomly initialized solver network is trained for a predetermined number of iterations to solve the task and its accuracy is recorded.

Digit classification experiment

The randomly initialized SVHN neural network was trained for hundred thousand iterations until the test data achieves an accuracy of 92.65%. It is important to note that a batch of sixty-four is used for training and the reported iterations are on batches and not epochs. SVHN neural network was initialized with PCA and LLE on MNIST and then the solver neural network was re-trained on SVHN dataset. The Table 4.4 shows the iterations required and the accuracies reached for this experiment. The same experiments were conducted when the SVHN was used for initialization of classification on MNIST database. The observations from those experiments are also noted in Table 4.4. In both cases the initialization with a target representation helped. The reported accuracies on sixty-thousand iterations are comparable to accuracies at hundred-thousand iteration in case of non-initialized network. The accuracy progress shown in Figure 4.15 shows the comparatively faster learning in the initialized model for the first sixty-thousand iterations.

Hands experiment uses the hand shape classification network which is initialized

by training for skeletal position for one hundred fifty thousand iterations. This network is then used for the classification of ten classes. The Table 4.5 describes the accuracy performance over iterations of training. The initialized model achieved the accuracy achieved by the non-initialized network ten thousand iterations earlier.

The two experiments demonstrate that we could employ the model-based constituent information for transfer learning. Effectively, for similar accuracies the model pre-trained with SVHN sees 2,560,000 less repetition of MNIST data points, the model pre-trained with MNIST sees 3,000,000 less repetitions of SVHN data points and the model for hand shape classification trained on the skeletal data sees 640,000 less repetitions of hand data points and obtains a better classification accuracy.

An unlabelled and constituent property based transfer learning approach is useful in improving the learning performance on a hand shape classification problem on depth data using a network initialized by learning skeletal points on binary hand images. This finding allows transfer learning over datasets which have similar physical properties but are recorded from sensors of different properties. It is possible that weight collapsing can be better handled by designing various representations for the data and using them randomly to generate targets while training the initializer network

4.7.4 Weight Priors with Physical Model Constraints

In this section a combination of the weight prior learning and the physical model constraints is attempted. The earlier described hand model neural network is retrained using the concept from the constituent transfer network. The convolutional network is now trained to learn a twenty-two dimension low dimensional representation of the hand pose images.

The frames from hand gesture sequences are used to learn a SVD based low dimensional representation extracted from the PCA. Ten thousand frames are used to learn this PCA decomposition. Further, the hand model network is trained as a regressor with sixty-five thousand image frames. The low dimensional representations of the hand images are the mapping output to be learnt by the algorithm. The PCA-embedding network is trained using a batch learning scheme with a batch size of sixty-four. The PCA embedding is tested on six thousand frames from the same gesture dataset.

Finally, a five node soft-max layer is added to the PCA-embedding network and it

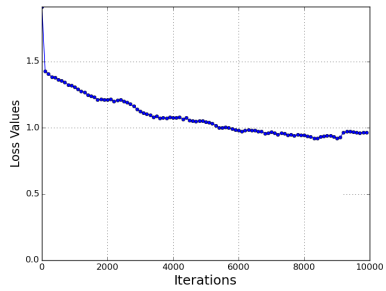


Figure 4.16: Test loss progression on PCA embedding network.

is fine-tuned to learn a classifier. The classifier learnt initialized by this procedure learns faster. The accuracy performance improved when the CNN for pose classification is trained with the proposed initialization based on first training the network with the PCA embedding. The initialized network was trained as compared to the Xavier initialized network trained for two times longer. The modified training procedure improves the classification accuracy for all classes of hand poses. This demonstrated a suitable solution for better classification when the labelled data is scarce or expensive. The new training procedure improved the average classification accuracy of the full test dataset from 90.5% reported in the Section 4.6.1 to 91.5%. The class-wise accuracy for both initializations are shown in Table 4.6. Fi-

Table 4.6: Accuracy for classification of five poses with hand model CNN the two initialisations.

Accu %	Fist	Flat	Open	Point	Join
Xavier Init	87	89	91	91	93
PCA Init	**89**	**90**	**94**	**91**	**93**

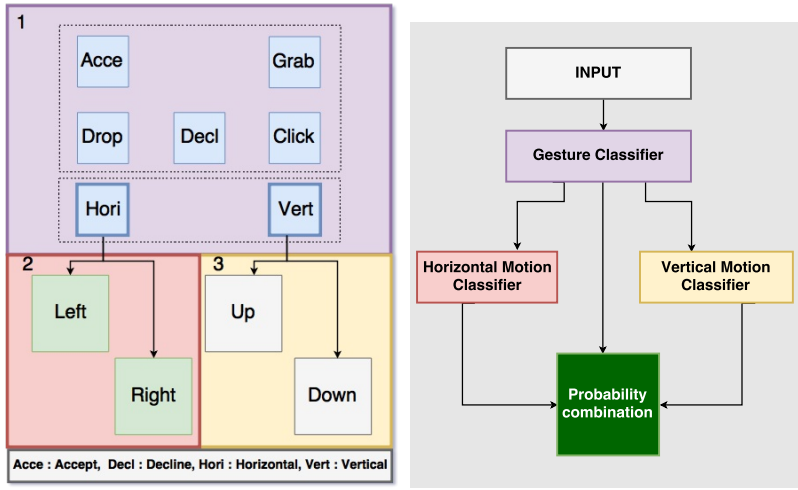
Table 4.7: Accuracy for classification of five poses with Hand model CNN the two initialisations.

Accu %	type	Fist	Flat	Open	Point	Join
Xavier Init	CNN model	86	87	90	91	91
	Hand CNN	87	89	91	91	93
PCA Init		**89**	**90**	**94**	**91**	**93**

nally, the classification accuracy comparison of the model performance for equal number of batch iterations for the three experiments, namely, the simple model, the hand model with Xavier initialization, and hand model with PCA initialization is shown in Table 4.7

4.8 Task Separation

The last section discussed a model based improvement to the CNN performance on a specified task. A similar approach of enforcing the known-information about the task to improve the performance on a time series task is presented in this section. This section takes an example of the gesture recognition and imposes the existing knowledge about the direction of motion and the hand shape change on the network. It was observed that separation of sequential learning tasks and the combination of the network output thereafter created a system that performs better than a single network.



(a) Definition of Classes by Motion and Shape. (b) Definition of Classes by Motion and Shape.

Figure 4.17: The Gesture Definition and the Block System Architecture.

4.8.1 System Architecture for Model Combination

The following experiments are based on the assertion that a hand-gesture is a sequence of frames of moving palm. It is a combination of palm motion with or without changes in the hand-shape. A gesture can thus be defined as a sequence of hand-shapes and locations where the sequence and location of various hand-shapes in one particular gesture class have a predictable, predetermined order.

The hand gesture classes $\mathscr{G}$ are divided into two sets $\mathscr{G}_s$ and $\mathscr{G}_m$ such that $\mathscr{G}_m \cap \mathscr{G}_m = \emptyset$ and . The first set has gestures which are predominantly defined by the sequence of hand-shape modifications, the second set of gestures are those defined by the direction of motion and the velocity of the hand.

To simultaneously classify these gestures, one large neural network is replaced by three networks which are trained independently, with distinct sets of feature vectors. Each neural is such that it solves a separate classification problem. The soft-max

projection of the output vector is treated as the probability vector **P**. The recurrent neural network $(f^r_{\mathbf{W}})^s$ is the classifier that identifies the primary shape and motion of the hand. The directional recurrent network $(f^r_{\mathbf{W}})^v$ and $(f^r_{\mathbf{W}})^h$ are added in sequence to make decisions on the classes.

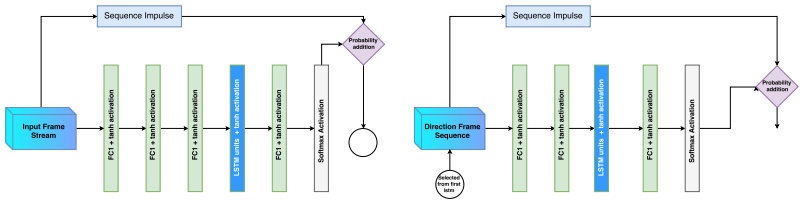
The first neural network $(f^r_{\mathbf{W}})^s$ is trained to classify seven classes as shown in Figure 4.17(a). The other two networks are trained to classify the direction of the motion, Up vs. Down and Left vs. Right. These networks are used in series with the first network. Various neural network architectures were trained and tested. One network architecture was chosen for $(f^r_{\mathbf{W}})^s$, and a smaller identical network architecture for $(f^r_{\mathbf{W}})^v$ and $(f^r_{\mathbf{W}})^h$ was used. The network architectures which provided the best cross-validation results separately were used for the classification system.

The LSTM Networks

Each network has an *LSTM layer* and several *fully connected dot product layers*. The input layer is connected to a dot product layer. Non-linearity is added to the network by using a *tanh activation function* with each fully-connected layers.

The network for the primary classifier has five layers apart from the input layer and the output *soft-max layer*. The LSTM layer is placed as the fourth layer from the input. The output layer has seven output nodes, each node represents one gesture, see Figure 4.18(a).

The binary classifier identifies the intended direction of the motion when the palm moves in horizontal or vertical direction. Since the swiping motion may be repeated more than once while completing the gesture the identification of the intended gesture is more sophisticated problem than merely identifying the direction of motion. The binary classifier LSTM network has *three hidden layers* along with the one LSTM layer Figure 4.18(b). The output layers have two nodes and a soft-max ac-



(a) LSTM network with the probability addition unit

(b) Architecture of the LSTM used for direction classification.

Figure 4.18: The LSTM architectures used in sequential prediction scheme.

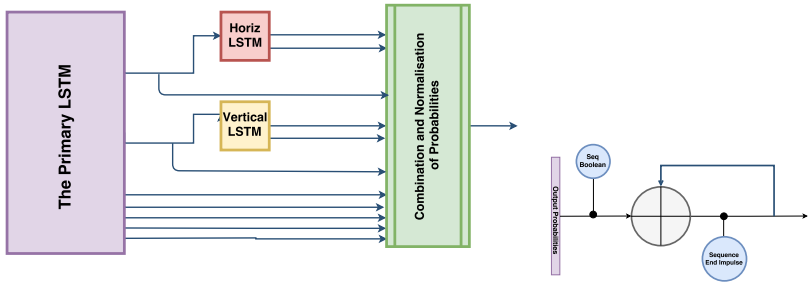
tivation function. The connection weights and bias are independent of each other in all networks . The three networks are trained independently using the samples belonging to the corresponding classes from the same training dataset. The training uses the RPROP Algorithm for the optimization process [154].

4.8.2 Prediction Combination Model

The system is shown in Figure 4.19, it can be broadly separated into a frame classification part Figure 4.19(a) which produces a nine dimensional probability vector $\mathbf{P}_\mathbf{t}^{'}$ at time t, and an output probability combination part Figure 4.19(b) which results in another nine dimensional probability vector $\mathbf{P_t}$.

In the classification part of the system the primary classifier is connected with the two motion-direction classifiers, see Figure 4.18. It provides a seven dimension probability vector, $\mathbf{P}_\mathbf{t}^\mathbf{I}$. $\mathbf{P}_\mathbf{t}^\mathbf{I}$ has five gesture probabilities $\mathbf{P}_\mathbf{t}^\mathbf{Ig}$ and probability for horizontal and vertical direction of motion $\mathbf{P}_\mathbf{t}^\mathbf{I}[\mathbf{h}]$ and $\mathbf{P}_\mathbf{t}^\mathbf{I}[\mathbf{v}]$. On identifying vertical or horizontal swiping of the hand the vertical or horizontal motion classifier is activated with binary activation signals A_v and A_h .

The activated binary classifiers then detect the intended direction of the swiping gestures resulting in the two dimension probability vectors $\mathbf{P}_\mathbf{t}^\mathbf{v}$ and $\mathbf{P}_\mathbf{t}^\mathbf{h}$ for vertical and horizontal direction respectively.



(a) 2-phase gesture recognition setup.

(b) Prediction output for gesture detection.

Figure 4.19: The system architecture and the cumulative probability addition scheme.

The gesture probabilities $\mathbf{P_t^{Ig}}$ from the primary classifier are renormalized. The vertical and horizontal probabilities $\mathbf{P_t^I[h]}$ and $\mathbf{P_t^I[v]}$ from the primary vector are used to weigh the probability vector $\mathbf{P_t^v}$ and $\mathbf{P_t^h}$. The output probabilities from LSTM units are combined (4.11) to form a nine dimensional vector $\mathbf{P_t^{'}}$ and are weighted by the there values in the primary probability vector, the resulting output vector $\mathbf{P_t^c}$ is re-normalized to form a probability vector $\mathbf{P_t^{'}}$, (4.12).

$$\mathbf{P_t^c} = \left[\frac{\sum_{j=1}^{5} \mathbf{P_t^{Ig}[j]}}{5}.\mathbf{P_t^{Ig}};\ \mathbf{P_t^v}.\mathbf{P_t^I[v]};\ \mathbf{P_t^h}.\mathbf{P_t^I[h]}\right]. \tag{4.11}$$

$$\mathbf{P^{'}} = \mathbf{P^c}/|\mathbf{P^c}|. \tag{4.12}$$

The early predictions of the LSTM based system are stabilized by using a cumulative probability addition scheme Figure 4.19(b). The cumulative addition of the probability regularizes the estimates while making an early prediction. This adds robustness towards jerks, stops and change in hand direction, during the comple-

tion of the hand-gesture sequence. Also a strategy based on maximum-probability or majority decision approach predicts the gesture at the end of the sequence. The described method makes a probability estimation for the gesture at every frame. The system output probability is given as $\mathbf{P_t}$, (4.13). The sum is reset to zero whenever an end of sequence impulse is seen. I_n is the impulse corresponding to the n^{th} sequence. The impulse has a value 1 and the impulse time is given by t_{I_n}. Effectively, the updated gesture probability vector $\mathbf{P_t^f}$ is,

$$\mathbf{P_t^f} = \mathbf{P_t^{'}} + (\mathbf{1} - \mathbf{I})(\mathbf{P_{t-1}}). \tag{4.13}$$

The past probability is added to the estimated probability until an impulse is observed. This addition is renormalised such that the sum of the vector resulting from the addition is one. $\mathbf{P_t^f}$ is the cumulative probability sum at time t and $\mathbf{P_t}$ is the normalised probability sum at t.

$$\mathbf{P_t} = \mathbf{P_t^f}/|\mathbf{P_t^f}|. \tag{4.14}$$

$$G_n = arg\max(\mathbf{P_t}) : \mathbf{t} - \mathbf{t_{I_{n-1}}} > \mathbf{t_d} \tag{4.15}$$

The probability addition is initiated again with a sequence-begin impulse. The n^{th} prediction G_n, corresponds to the index of the maximum value in the probability vector $\mathbf{P}$ (4.15). Since the initial frames of the sequence have little or no temporal context the predictions made during these first t_d frames of the input stream are not reliable. This scheme allows continuous predictions unlike majority-vote like decisions where prediction is made after viewing the entire sequence.

Results

This section describes the training progression of the three models and presents the performance of the entire system. As mentioned in the last sections the first few frames of the prediction made by the system are not considered for output, we also

Table 4.8: Confusion matrix proposed system.

%	U	D	L	R	C	A	De	G	Dr
U	**84**	4	2	0	8	2	0	0	0
D	4	**85**	0	0	0	0	8	0	3
L	0	0	**92**	1	0	0	3	3	1
R	0	0	0	**93**	0	0	3	4	0
C	0	0	0	0	**96**	0	4	0	0
A	8	4	0	0	1	**82**	4	1	0
De	3	7	0	0	0	4	**84**	2	0
G	4	0	0	0	0	5	0	**89**	2
Dr	1	4	0	0	0	0	4	0	**91**

Table 4.9: Confusion matrix single all-class LSTM

%	U	D	L	R	C	A	De	G	Dr
U	**77**	8	0	0	0	5	3	3	4
D	7	**78**	0	0	0	0	9	2	4
L	0	0	**88**	6	0	2	4	0	0
R	0	0	4	**89**	0	4	3	0	0
C	0	0	0	0	**96**	0	2	2	0
A	8	5	0	0	1	**78**	5	1	3
De	2	9	0	0	0	7	**80**	0	2
G	2	3	0	0	0	2	0	**91**	2
Dr	3	1	0	0	0	0	5	0	**91**

U:Up, D:Down, L:Left, R:Right, C:Click, A:Accept, De: Decline, G:Grab, Dr:Dropping.

skipped these frames for the evaluation analysis. The output probabilities for sequences beyond the eighth frame of the gesture, which corresponds to a time-period of 0.3 seconds are considered for the analysis.

The train-test error progression by learning epochs for the three LSTMs that form the system during a sample cross validation are shown in the primary phase of classification and the binary classifiers are depicted in Figure 4.21. The network was trained for six hundred epochs and evaluation was conducted for every second epoch. The average misclassification rate for the given training was 5%. The misclassification rate for the test data at the end of the training was 7%, see Fig-

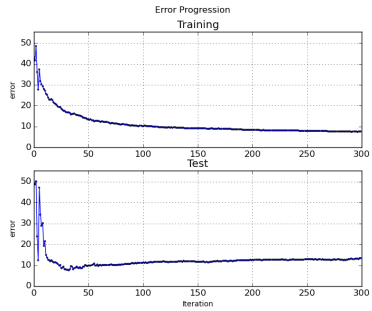
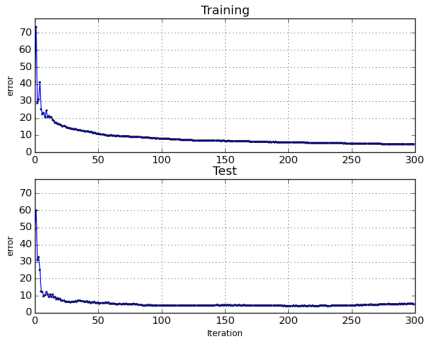


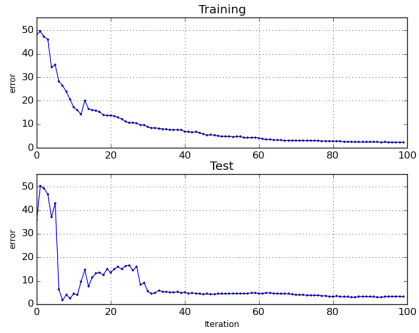
Figure 4.20: Test-Train error progression for one all-class LSTM.

ure 4.21(a). Both up-down and left-right classifiers were trained as binary classifiers. These networks were trained for four hundred epochs. It can be observed that the misclassification rate on training data after the completion of the training for the up-down motion classifier is 6%, and 1.5% for the left-right classifier. The misclassification rate for the test data is 8% and 3%, respectively, see Figure 4.21(c) and Figure 4.21(b). On combining the three networks as the described system, the observed misclassification rate for the full system is 9.25%. The Table 4.8 shows the confusion matrix for the classification of the nine gesture classes in case of the architecture following the two level classification strategy shown in Figure 4.21.

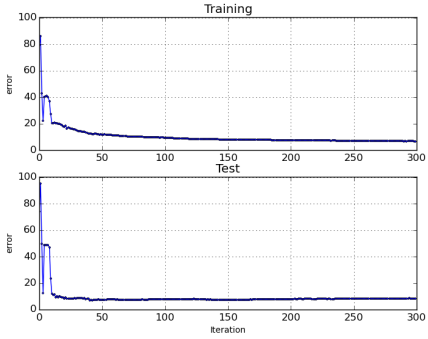
In comparison with a larger all-class single LSTM, chosen after experiments on multiple LSTM models, the performance was considerably better. The improvement in the gestures where direction is important is large. In other gestures the performance improves in all classes apart from 'Drop' where accuracy remains the same and 'Grab' which has a small decrement. The performance of the compared LSTM model is shown in Table 4.9 and the the training progression example for this is shown in Figure 4.20. Confusion matrices are calculated at each step of the 9-fold cross validation and the mean confusion matrix are reported.



(a) Primary LSTM classifier.



(b) Left-Right classifier.



(c) Up-Down classifier.

Figure 4.21: The training and test error progression for the three components of the geture classification system.

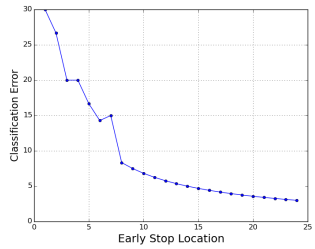


Figure 4.22: Test error at the end of training plotted against the decision latency in frames.

On the Latency of the Problem

The performance of the system improves when decisions were taken after a longer delay from the beginning of the sequence. Figure 4.22 shows the accuracy performance when the latency period for the frame-wise prediction was changed. The decision after a longer latency gained from larger temporal context and is usually more accurate. Some gestures with similar shape and short motion were misclassified, which was reflected in the occasional misclassification of "Accept" and "Decline" as "Up", "Down", respectively. This explains the lower accuracy of the up-down gestures in the combined system even though the binary classification accuracy is high. The accumulated regularization of the system output also resulted in missing of fast-very short gestures. In the next section, the idea of modelling the problem is further explored. The task of hand gesture recognition is modelled as a function of hand shape transitions. Both, the hand-gesture recognition networks and the hand-shape classifier networks are now available to us and the gesture estimates are corrected using a probabilistic combination of the task model and the shape estimates.

4.9 Task Priors

So far in this chapter, the advantages of the information addition have been demonstrated. It has been shown that a network performs better when it has physical or statistical information about the data or the task. The best shape and gesture classification networks are now combined to improve the gesture classification performance. This combination is created using a weak statistical model of the gesture classification task.

The CNN shape classifier described in Section 4.6 and the sequential LSTM combination system used for the hand gesture classification in Section 4.8 return soft-max outputs and normalized softmax values of the class estimates. The hand shape classification used a hand degree of freedom constraint and an initialization based on the dataset PCA, as explained in the Section 4.7.4, are now used in combination, and the improvement of performance is demonstrated by this combination. The earlier sections of this chapter introduced model constraints, task based prior for initialization of the network and task distribution for improved performance of the classification task. This section introduces a weak combination of the sequential gesture classifier and the instantaneous shape classifier. Apart from modelling the neural networks on the priors of the data or task, the prior derived from the gesture to shape relation are used to improve the classification performance. This section will demonstrate the use of constraints, biased-weights along with using task information at the neural network output.

4.9.1 The Probability Combination System Architecture

As mentioned earlier the output shape and gesture probabilities from the neural network and neural network system are given by $\mathbf{p}$ and $\mathbf{P}$ respectively. These prob-

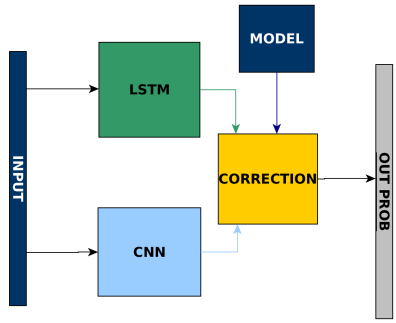
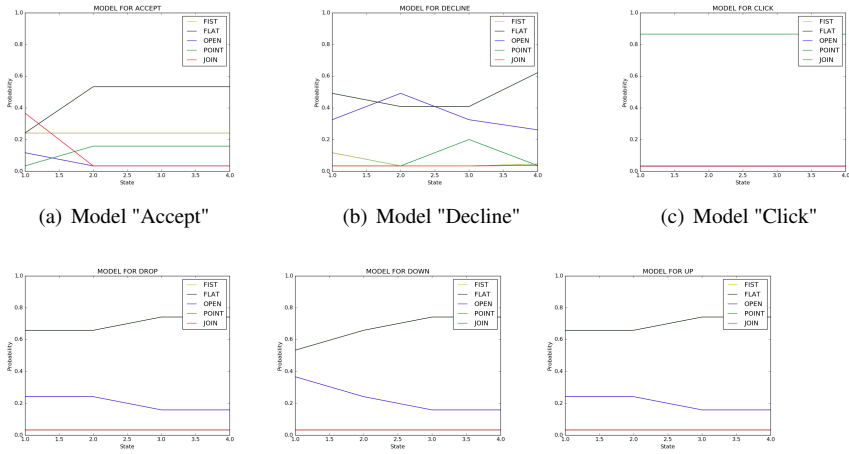


Figure 4.23: The gesture and shape probability combination system.

abilities for the hand shape and hand gestures are received from the classification systems described in the Section 4.7.4 and Section 4.8 respectively. As defined earlier the hand gesture is a combination of the transition of hand shapes and locations. At a frame t of the hand gesture g in gesture set $\mathscr{G}$ the shape probability is $\mathbf{p_t}$ and by the definition of gesture probability $\mathbf{p_t}|\mathbf{g}$ can be modelled.
The conditional probability $\mathbf{p_t}|\mathbf{g_t}$ from the gesture model along with the $\mathbf{p}$ from the CNN shape network are used to make an estimate for the $\mathbf{P_t}$, this system is shown in the Figure 4.23. The LSTM and the CNN blocks are the earlier described systems and the model block has the hand gesture models with respect to hand shapes. The hand gesture model is now described.

Designing the gesture-shape model

The hand gesture model based on the shape is designed as a discreet function of the $\mathbf{p}$ vector and time zone T, $g_T = \mathscr{F}_m(\mathbf{p}, \mathbf{T})$. It can be represented as a matrix of time and individual shape probability at that instant. These gesture models were constructed as week statistical measures. A small dataset with twenty gestures from each class was created to generate these models. Every frame of the gesture se-



(a) Model "Accept" (b) Model "Decline" (c) Model "Click"

(d) Model "Drop" (e) Model "Down" (f) Model "Up"

Figure 4.24: The models for the Hand-gestures based on Hand-Shape transition.

quence in dataset was marked with the hand gesture that these frames compose. Apart from the gesture label, each frame of the normalized gesture sequence was labelled with the hand shape closest to the hand shape in S. This dataset was used to create a probability model $\mathbf{p}_{\mathbf{s}|(\mathbf{G}=\mathbf{g})_\mathbf{t}}$, to be read as probability of a shape s occurring during the t^{th} part of the sequence given gesture g in $\mathscr{G}$. Each gesture was modelled as a sequence of five shape probabilities over a time frames of six frame each.

The shape appearing over each time frame in the samples of each dataset are counted and a probability estimate is made. To avoid zero-probability situations a value ε is added to each probability value in the models. The probabilities are renormalized after this addition. The probability model calculated for a subset of gestures are shown in are shown in Figure 4.24.

4.9.2 Final probability estimates and results

An estimate for $\mathbf{p}^{'}$ for the shape probability at the time instant t is made using the total probability, such that,

$$\mathbf{p}_{\mathbf{t}}^{'} = \sum_{\mathbf{k}} \mathbf{p}|(\mathbf{G} = \mathbf{k})_{\mathbf{t}}. \tag{4.16}$$

The probabilities $\mathbf{p}$ and $\mathbf{p}^{'}$ are used to calculate a measure of divergence between the instantaneous estimates of the two systems. The divergence calculated from shape probability is used for an online re-weighting of the iterative weighting factor. The weighted gesture probabilities at each frame are then calculated as,

$$\mathbf{P}(\mathbf{G}_{\mathbf{comb}})_{\mathbf{f}} = \mathbf{P} + (\psi(\delta(\mathbf{p}, \mathbf{p}^{'})\mathbf{P}^{'}, \tag{4.17}$$

where $\psi(x) = 1 - x$ and the divergence δ is calculated as,

$$\delta = max(\mathbf{p}) - \mathbf{p}^{'}[\mathbf{arg}_{\mathbf{max}}\mathbf{p}]. \tag{4.18}$$

This is done to give high weighting to the LSTM estimates if the strong hand shape classifier, based on CNN, is more certain about a class as compared to the weak model based on the $\mathbf{p}_{\mathbf{s}|(\mathbf{G}=\mathbf{g})_{\mathbf{t}}}$ modelling. This corrects for unintended, short change of shape while completing a gesture. The weighted gesture probability is renormalized.

Comparisons with earlier described methods

The LSTM system for gesture recognition (Section 4.8) and the CNN model initialized from the PCA (Section 4.7.4) were explained in detail earlier. These networks were trained and tested separately. Further, in the forward phase the probability output of these models on frames of the gesture sequence were combined to create a better estimate of the sequence. The corrected estimation for the gesture after

Table 4.10: Accuracy for nine gesture classification based on three methods.
U:Up, D:Down, L:Left, R:Right, C:Click, A:Accept, De: Decline, G:Grab, Dr:Drop.

% Accu	U	D	L	R	C	A	De	G	Dr
	Static gestures					Dynamic Getsures			
LSTM	77	78	88	89	96	78	80	91	91
2 Phase LSTM	**84**	85	**92**	93	**96**	82	84	89	91
Proposed	81	**86**	89	**93**	95	**85**	**87**	**92**	**92**

combining the probabilities from the two models are now reported in this section. The results obtained from this method were compared against the task separation LSTM system that was used for making a gesture estimate in the (Section 4.8 The gesture classification accuracies were also compared with a single all class LSTM which was trained and tested on the same dataset.)

The classification decisions are made only after the first ten frames (eight frames for one-third gesture and two frames wait for entire hand to be in the observation area) of the gesture. The accuracy percentages are reported as percentage of correct predictions after the first ten frames. It is observed that the short or rapid change of gestures were missed because of the LSTM learning procedure. This problem was solved with the correction scheme. The results from a single phase LSTM were compared with the LSTM combination explained earlier and finally with the probability correction paradigm that has been proposed in this work.

The proposed solution outperformed a large single LSTM consistently. On comparison with the 2-phase LSTM the solutions provided better accuracy in six of the nine classes, see Table 4.10. The proposed solutions consistently performed better on the dynamic gestures. The overall accuracy on the test dataset increased from 88.50% on the 2-phase system to 89.50% on the proposed solution.

4.10 Conclusion

Three methods of including model information or a priori information to the neural network were discussed. It was shown that the model constraints and biases introduced to the weights help in training of the neural network. These experiments were shown on hand shape and hand gesture classification problems. The property of a priori bias was also used for developing a intra-domain transfer learning methodology and it was shown that both physical property and the statistical properties of the dataset can be used for transfer learning.

Eventually, it was also shown that the neural networks trained to learn separate problems can support each other if a weak function relating these tasks can be formalized. Treating the neural networks as a function on which constraints and biases can be imposed allows introducing existing experiences into the learning process, this can be used in various ways. Such methods are particularly useful for reducing the data requirement, developing strategies for augmenting data with unlabelled datasets and using synthetic data for learning initialization. This also provides a scheme for pre-including mathematical models into the learning process.

Chapter 5

Mutli-Modal Networks and Independent Information

Combining multiple models together in certain ways improves the performance of the classifiers or regressors. Various methods for the combination include committees, boosting and majority voting.[155, 156]. The notion of independence of information and ensemble of these information to make better decision has been used in various standard learning algorithms. The Random Forests are possibly the most popular ensemble algorithms, growing a Random Forest involves combination of various decision trees such that the each tree depends on a random subset of features sampled independently.

The general procedure in creating an ensemble predictor involves creating multiple $\mathbf{f_1}(\mathbf{x}), \mathbf{f_2}(\mathbf{x}), \mathbf{f_3}(\mathbf{x}), ..., \mathbf{f_I}(\mathbf{x})$ prediction function. These functions are built such that multiple functions are trained with randomly selected independent features from the feature vector. The combination of the decision trees are made such that multiple simple models f_i are combined to make complex decisions. This strategy is based on the assertion that, two weak models with classification performance $\mathscr{E} > \varepsilon$, where ε is the performance of random decision,when combined produce

better than either model individually[157]. The ensemble models are also useful in avoiding over-fitting, theoretically a combination of models trained on mutually independent information does not over-fit however large the combinational model grows [157].

This chapter discusses methods of combination for models. The neural network based solutions are developed in a way that they carry separate information but are not necessarily mutually exclusive. The three experiments discussed in this chapter include,

- Neural Network combination of optical and inertial measurement units for human pose classification.
- A sequential Neural Network combined with block classifier for low latency classification.
- Fusion scheme using the LSTM based prediction and Kalman Filter correction using a visual tracker measurement.

The last chapter discussed introduction of models to identify *a-priori* nature of the input data and to use this knowledge towards improving the classification performance. This chapter will focus on identifying separable features in the input feature set $\mathbb{F}$ and using them such that the networks can learn better representation for certain tasks. The methods for the eventual combination of the independent information will also be discussed in the chapter.

The Section 5.1.2 presents solutions for the combination of inertial and visual information to show that the separate subsets of input information combine to improve the classification task. The experiments are made with combinational networks in Section 5.1.2 and it is demonstrated that the combination of separate information improves classification of human body gestures.

Next, in the Section 5.2.5 the combination of features learnt from the same dataset

but using distinct learning policies and thus resulting in dissimilar patterns are investigated.

5.1 Multi-Modal Models

The UTD Multimodal Human Action Dataset(UMHAD) [158] is a dataset for human action recognition. It provides a depth and inertial sensor data for the task of classifying twenty-seven human action categories recorded on eight individuals. The experiments were conducted on this dataset to validate the improvement of performance while using a multi-modal neural network. The UMHAD dataset includes the output from a Kinect imager which provides a depth image of 320×420 and an RGB image of 640×480. The RGB images are down-sampled by a factor of two to keep consistent size input. The dataset also includes a six dimension output from one inertial sensor worn on wrist or thigh depending on the action category. The start and end of an action sequence are synchronized using the start and end frame inputs. The experiments described in this section are made to confirm that the combination of information from separate sources can be combined by neural network to learn better representations for the tasks at hand.

5.1.1 Information Combination with Body Skeleton and Inertial Measures

The set of experiments conducted included building a benchmark with single source information. The Kinect frames are threaded together and an LSTM version of neural network is trained. The skeleton point locations and the inertial values are used as one dimensional vector. As these vectors describe the location and motion of the skeleton the vectors are concatenated and the sequence of these values are

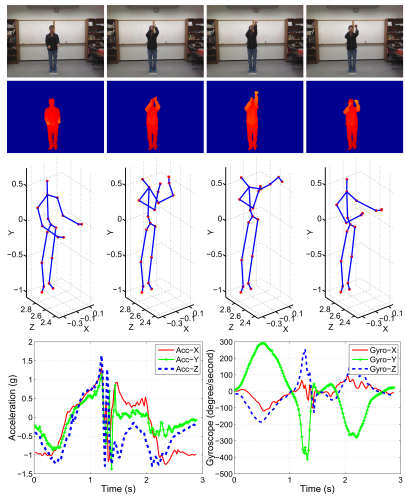


Figure 5.1: Samples of the RGB images, depth channel, skeletal points and intertial measurements.

used to train another LSTM.

The Inertial Data

Inertial data and the skeleton data are concatenated to produce a sixty-six dimension vector. It includes sixty elements that define the $x-y-z$ coordinates of twenty skeletal points and the inertial values representing the three axis of acceleration and three axis of rotation. A sample sequence of the inputs are shown in the Figure 5.1. In this experiment the 3D coordinates of the skeletal system and the inertial measures with three axes of accelerometer and three axes of magnetometer measurements are combined in the input vector. The frame rate of the Kinect camera and the inertial measurement unit are different but the dataset provides a simultaneous start and end point for the sensor output.

To combine the two inputs a constant frame rate is required. In these experiments

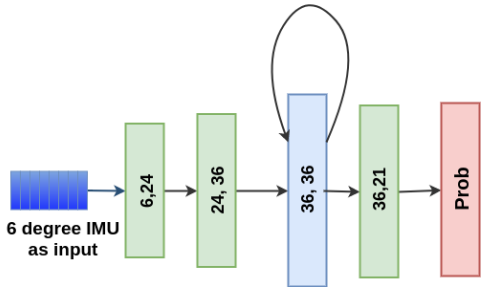


Figure 5.2: The neural network used for training the classifier with only the IMU signals.

the higher frequency inertial measurement sequence are sub-sampled to the size of the Kinect body action sequence. To reduce the loss of information, the sub-sampling is applied after doing a moving average on the concatenated inertial and skeletal signal vector with a window of kernel size three.

5.1.2 Neural Networks for Separate Body Skeleton and Inertial Measures

In the first set of experiments, for the body action recognition two neural networks are trained for creating single information classifiers. A neural network each is trained for classification with inertial measurements and skeletal data to classify the actions. The experiments in this chapter are made with the twenty-one classes which use the sensor on the wrist. The six classes collected with the sensor worn on the thigh are not used. The experiments are carried out such that the recordings of the six recorded individuals "one" to "six" are used to train a neural network, and the recordings from "seven" and "eight" are used for testing the trained network.

In the first experiment, a neural network is trained on the IMU sequence only. The

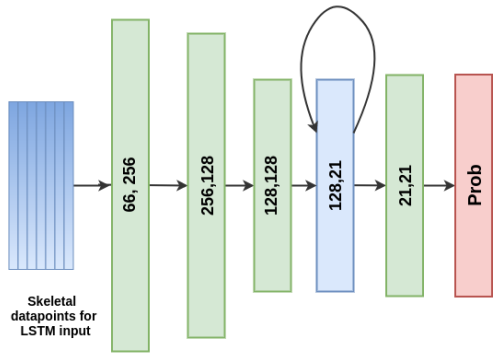
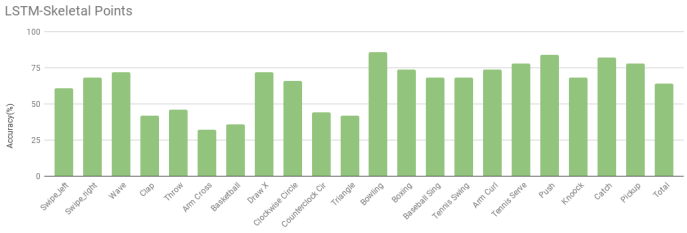


Figure 5.3: The neural network used for training the classifier with only the skeletal point inputs.

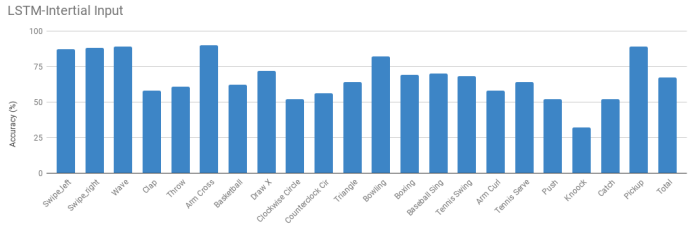
network shown in Figure 5.2 is chosen to train the IMU sequence. This network has one LSTM layer with a fully connected equal-output layer, and three fully connected layers. The output layer, like in earlier experiments on classification maps the output layer values to their softmax values. The probability addition scheme introduced in the Chapter 4 is used for the selection of the most likely action.

The second experiment follows the same scheme, but uses the Skeletal coordinates as the input to the network. The input vector thus is a sixty dimension input and is trained on the network shown in the Figure 5.3. The vector is formed such that the respective x, y, z coordinates of the skeletal points are placed together. The network has three fully connected layer and one LSTM layer with a twenty-one node input-output fully connected layer. The LSTM networks are trained for one thousand six hundred epochs and the top-one accuracy is calculated for the body action classification tasks.

The average accuracy achieved for by training with the inertial input on the inertial network shown in Figure 5.2 was 67.4% and the average accuracy for the network trained by the skeletal input was 63.9%. The Figure 5.4(a) shows the class-wise



(a) Skeletal input accuracy values



(b) IMU input accuracy values

performance for the skeleton inputs and the Figure 5.4(b) shows the class-wise performance of the inertial network.

Comparisons with combination of information in networks

Two experiments are made to evaluate the influence combination of the information. **The first experiment** for evaluating combination of information, includes separate use of the LSTM networks for IMU and skeletal points described above in a multi-column policy. The network constitute parts of the IMU and the Skeletal networks discussed earlier. These two networks receive the input information independently and the vectors are fed into the LSTM layer independent of each other Figure 5.4. The sequential information learnt in the LSTM layer then progresses in the network and is combined in the outer layers of the network. In such combination, the increase in input vectors space is handled by increased parameters in

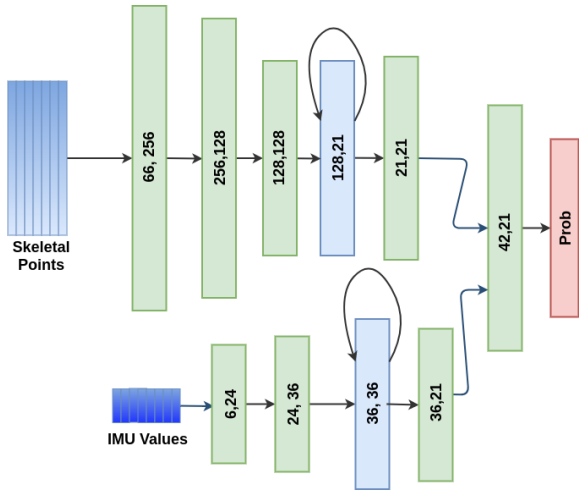


Figure 5.4: The neural network used for training the skeletal and IMU inputs with separate input.

the network. The overall weight-parameters in such combinations are less than the sum of the parameters in two independent network.

A second experiment with combined input is completed to validate the argument that the independent informations add to the performance of a task. To do so comparisons are made against a network which learns the time context simultaneously. The neural network shown in the Figure 5.5 is used for this experiment. The two sets of data are concatenated and the experiments are conducted on a combined dataset. As mentioned earlier the start and end frame from both sensors are known and are used for frequency calibration. This second neural network that is trained on a concatenated vector is designed such that the number of weights connecting the fully connected layers are approximately equal to the number of weights in the two-column network used to make the action predictions.

These experiments are conducted as described in the earlier in Section 5.1.2, but both the neural networks were trained for two thousand four hundred epochs as opposed to one thousand six hundred epochs in these experiments. According to the

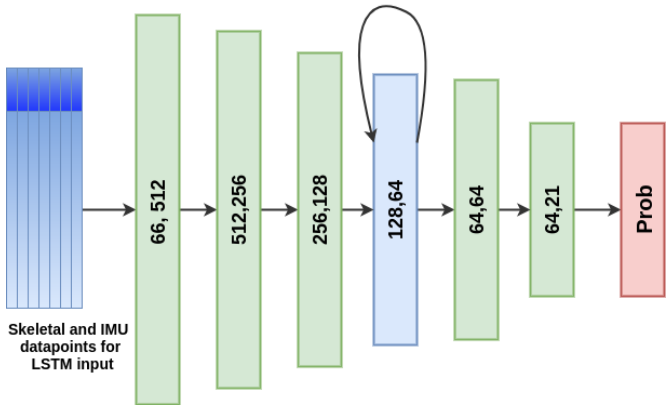


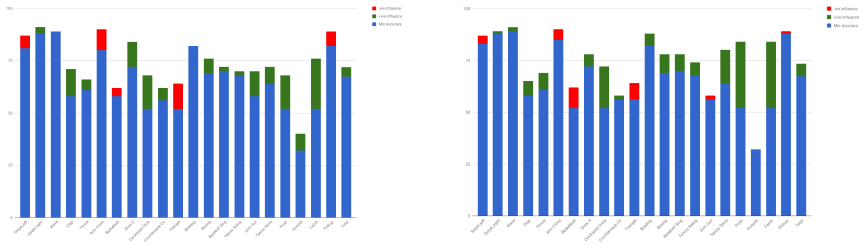
Figure 5.5: neural network used for training the combined skeletal and IMU inputs.

information. The combination of the two networks performs better than the individual networks. This confirms that the two sensor information can be combined to improve task performance in the neural network.

The influence of the combination on the average accuracy for each class is shown the charts in the Figure 5.6. The average accuracy over the dataset improves when the combination of the source are used for training the network. An average classification accuracy improvement of 6% is observed in case of the separate input decision combination scheme. The concatenated vector scheme also improves the classification performance but by a smaller margin of 4.5%. The green component of the bar-graphs in the Figure 5.6 shows the gain because of multiple information sources. The red portion of the accuracy bars shows the magnitude of the reduced accuracy.

The comparison between the separate input and concatenated vector input is shown in the Figure 5.7. The average performance of separate learning is marginally better than the input tied together. The plot shows the influence of the input kind on the classification performance.

Of the three trained networks those trained with both inertial measurement and



(a) Average accuracy for the network trained on concatenated vector

(b) Average accuracy for the network trained with separate input vectors

Figure 5.6: Class-wise average performance of the two different input networks in classifying the actions.

Table 5.1: Average Accuracy for the trained networks.

% Accu	Inertial	Skeleltal	Concatenated Input	Separate Input
	67.4	63.8	71.90	**73.3**

skeletal coordinates perform better than the network trained only with the moving average inertial data, which in turn performed better than the network trained only on the skeletal data. Further, it was observed that the neural networks that learns on separate input performs the best. The results for these experiment are shown on Table 5.1.

5.2 Multi-Information Models

The last section showed the separate combination of two data sources. The network in which the learning of temporal features is relatively more independent performs better than other networks. Further, the last section demonstrated that correct strategies for combination of distinct but not mutually exclusive information improves the learning performance more than a simple combination of information. It is of

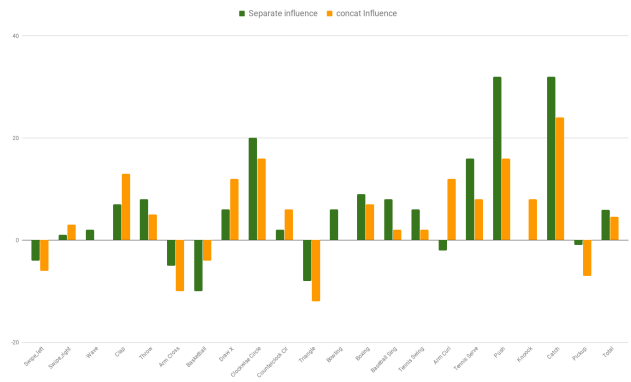


Figure 5.7: The variation of the accuracy from the IMU classifer when combined input scheme is used.

interest to investigate if the combination of features learnt from the same dataset but using distinct learning policies and thus resulting in dissimilar patterns can contribute to improving the learning performance.

The same strategy of using multi-column Neural Networks as earlier is employed. In this section the distinct inputs scheme of the last section is replaced with learning by temporal layers with different properties. These experiments are made to test if a combination of LSTM and C3D can help the latency performance of a time series classification problem. The tests have been made on an in car hand gesture dataset and the comparisons were made against an earlier reported ([18]) C3D model which resulted in best classification performance on the classification of the had dataset.

It will be demonstrated that classification properties can be improved by using combination of dis-similar concepts. Multi-column networks with columns having distinct temporal layers are created. A hand gesture recognition system that uses a combination of C3D and LSTM architectures of neural network is introduced and shown to perform well on early-recognition task. This is a combination of ideas from [18] with those of [19] for early and incomplete gesture recognition.

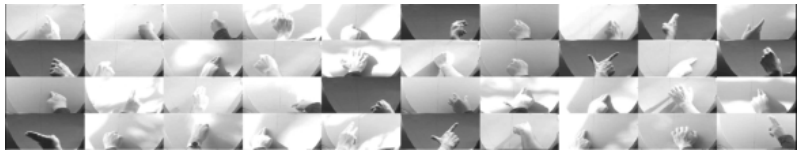


Figure 5.8: The example of the hand gestures in the Viva dataset.

5.2.1 Dataset

The VIVA challenge dataset was used for these experiments. The gestures are moving hands with changing or constant hand poses. The dataset has variations in illumination and depth. The VIVA challenge dataset has video sequences of fifteen hand gestures performed by eight subjects under varying lighting conditions. The dataset includes 885 intensity and depth video sequences [7, 114]. The datatset is recorded with the Microsoft Kinect device of resolution of 115×250 pixels. The durations of the hand gesture in the VIVA dataset are distinct. To create an equal length gesture, the normalization of the dataset sequence length is required. To compare with [18] the gestures were re-sampled to sequences of thirty-two frames. Sampling was done by repeating each frame after a fixed interval when the gesture sequence is less than thirty-two. When the sequence length is more than thirty-two the gesture sequences are sub-sampled by dropping frame after a fixed interval.

The normalized gesture sequences contain depth and intensity values. Intensity gradient values were calculated. The gradient and the depth values were normalized over the dataset and a two channel input from the gradient and normalized depth was created for each frame. The labels corresponding to the gesture type mark each frame. The gestures sub-sampling was done such that the the frame sequences with most variation in hand shape and motion are dropped with smaller probabilities. This is done by sampling based on magnitude of per pixel change over time within a gesture. The dense optical flow between two frames separated by time $\delta T = 2$

was calculated and the absolute change per pixel over the entire gesture was used for sampling distribution. This strategy allows improving the probability of conserving the fast changing frames during sub-sampling and increasing such frames when up-sampling the sequence.
Three classes which have "Swiping" hand which changes direction in later parts of the gesture and the class "Tap three times" were removed to analyse the performance. This was done because these gestures which are similar in the initial frames will be characteristically misidentified when tested for incomplete lengths. Effectively the experiments were conducted on fifteen hand gesture classes.

Short length sequences

The section focuses on the improvement in latency performance for the time series so performance of classification on shorter gesture sequences is tested. To this effect, the dataset with incomplete gesture length is created. Half length and quarter length incomplete gestures were created by only using the the first sixteen and eight frames from the start of the hand motion. To assure that the some hand motion indeed exists, the first two frames of the gesture sequence were always removed.

5.2.2 The Components of the Multi-Column Networks

As mentioned in the introduction, this experiment set will demonstrate the usefulness of property combination. To do so, a combination of the C3D and the LSTM layer based neural network architecture is used. The C3D layer uses volumetric convolutions. The input to the volumetric convolution layer is a block of frames. The network learns various volumetric filters. As the convoluted blocks are propagated forward through the max-pooling layers the learned filters reside on higher

scales space. Effectively, the minimum time frame of learning in C3D is thus the time length of the spatio-temporal filter on the layers closest to the input. This time-spread increases as we move towards the output layer in the network. The LSTM on the other hand accepts sequential input. The LSTM learns to use forget gates and identifies the length of the learned structure in the training phase.
All the networks are trained on the negative log likelihood cost function and each uses a soft-max projection on the output layer. The networks with single column are trained for three hundred epochs and two-column network trained simultaneously are trained for five hundred epochs. A Negative log likelihood cost function is used. In case of the single phase network the training is completed in 300 epochs. Owing to their larger size the two-column networks are trained for 500 epochs. Before explaining the combination, the architecture of the two networks is first described,

C3D network

The two column C3D network of [18] which has shown to work well on the same dataset, uses two networks with high and low resolution input. These networks provide two sets of properties, which are multiplied, normalized and used for predictions. The C3D network used in this work, shown in Figure 5.13, is the high resolution branch of the network proposed in [18] . The C3D consists of four volumetric-convolution layers, each of these layers have associated volumetric pooling layers. The tanh layers are used as the activation functions after the volumetric convolution. The fourth volumetric convolution feeds into a softmax layer which provides a probability vector as output. The C3D provides one output for the entire block of the K stitched inputs, the output prediction in this case is the index with highest probability.
For designing a network that learns to classify a gesture of length K, the input to the C3D is a $K \times 2 \times 57 \times 125$ vector. The experiments were conducted such that each

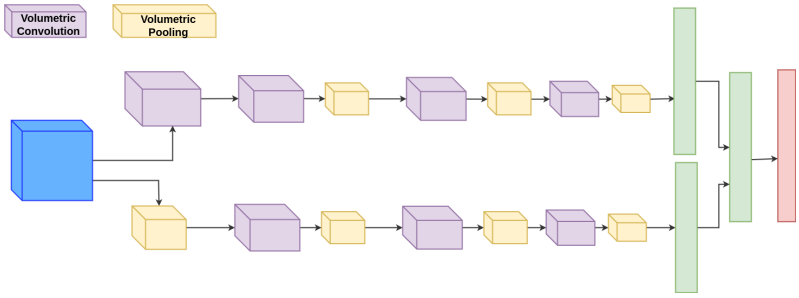


Figure 5.9: The C3D network (*"C+c"*) : Two column neural network with the input passing through a volumetric pooling layer in lower column; The input travels through multiple layers and the output layer is a softmax probability layer.

frame of the input block belonged to the same gesture type. An output probability vector of fifteen gestures is produced at the output of the C3D.

The LSTM network

The LSTM network in the second column of the network has two convolutional layers followed with the usual pooling and ReLu layers. An LSTM layer and a fully connected layer follows the convolutional layers, see Figure 5.14. The same $K \times 2 \times 57 \times 125$ input for the C3D is feed into the LSTM. The output layer is a soft-max projection. Each frame of the gesture sequence is marked by a label such that the LSTM produces a probability output at every frame of the gesture. The LSTM predictions is made by cumulative probability addition over the gesture sequence and then the index with highest probability sum at the end of the sequence is identified as the gesture.

5.2.3 Experiments with Two-Column Neural Networks

As mentioned earlier a multi-column C3D has been reported as a benchmark on the hand gesture dataset. The experiments are made to test the performance of this network on the restricted dataset of fifteen classes. Further, this network is compared against a two-column LSTM and a two-column combination network of LSTM and C3D. The neural networks are trained with data from fourteen recordings from seven persons and tested on two sets of recording from the eighth person. The final accuracy results are averaged over eight experiments where all the test persons are used once. All networks are trained for full sequence(thirty-two frames) and half and quarter sequences(sixteen and eight frames)
One column of the both the networks are as described earlier in the Section 5.2.2. In both cases n average pooling layer is used in the second layer column. This is done to provide varying scales as input to the first convolutional layers of the two columns of each network. The first volumetric pooling layer in the C3D network scales only in the spatial dimensions.

The network naming convention

The following sections use multiple neural network designs using C3D and LSTM network. To keep track of them a naming convention is proposed. For, the C3D network the high resolution input network is identified by *C* and the low resolution network by *"c"*. Similarly, the LSTM layer based networks are identified by *"L"* and *"l"* and *"R"* for simple recurrent neural network. A two-column combination is represented by +, thus a C3D network with two columns of high and low resolution will be represented by *"C+c"*.

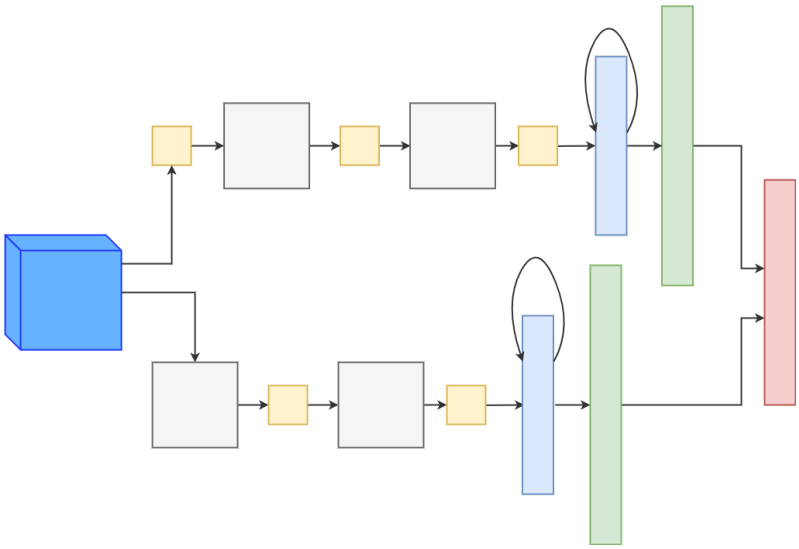


Figure 5.10: The LSTM network (*"L+l"*) : Two column neural network with the input passing through a pooling layer in the lower column; The input travels through multiple layers and the output layer is a softmax probability layer.

Two Column Neural Network

The two column C3D neural network is compared. An end-to-end training with two-column neural networks based on the components tested above is performed. First, the two column C3D is compared against a similar size LSTM network described in Section 5.2.2 . Thus the following networks are trained and tested,

- A two-column neural network with 3D convolutional layer joined at head with a fully-connected layer (*"C+c"*), see Figure 5.9,
- A the two-column neural network with convolutional layers followed by LSTM layer and joined at head with a fully-connected layer (*"L+l"*), Figure 5.10.

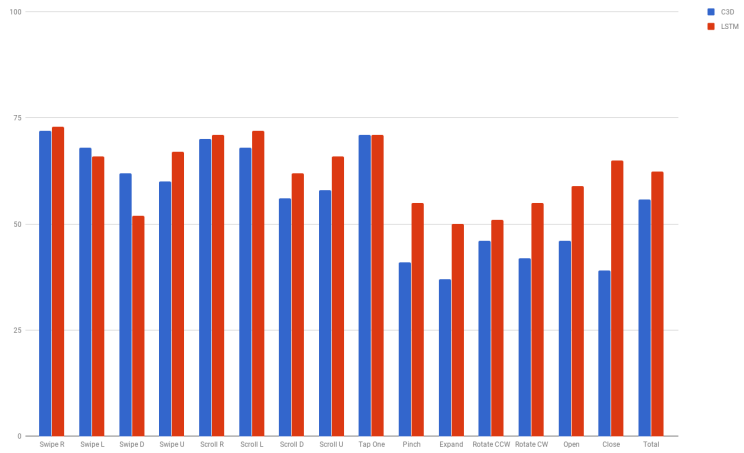


Figure 5.11: The accuracies of 16 frame sequence on *"C+c"* (Red) and *"L+l"* (Blue).

The *"C+c"* and *"L+l"* networks of Figure 5.9 and Figure 5.10 respectively are trained and tested for full sequence gestures of length $K = 32$ and half length gestures of frame length $K = 16$.

The test accuracy in percentage for the two networks for different frame lengths is reported in the Table 5.2. The *"L+l"* network performs worse than the *"C+c"* network on the full sequence gestures. Though, the *"L+l"* network performs better than the *"C+c"* network for shorter sequences. The results from these experiments are listed in Table 5.2. The class-wise performance of the performance of these networks on the full sequence and half length gestures is shown in the Figure 5.12 and Figure 5.11 respectively.

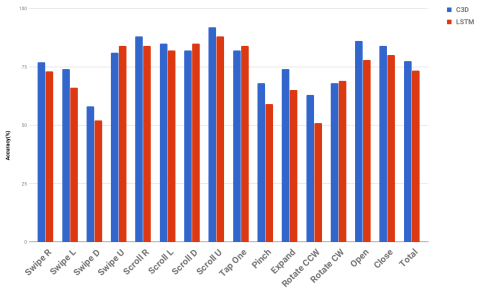


Figure 5.12: The accuracies of 32 *"C+c"* (Red) and *"L+l"* (Blue).

Gesture length	Network: *"C+c"* Accu: (%)	*"L+l"* (%)
32	**77.4**	73.4
16	55.7	**62.3**
8	31.6	**37.3**

Table 5.2: Classification Accuracy with the two-column LSTM and C3D.

5.2.4 Single Column Neural Network

The results from the last section motivates training single column C3D networks and LSTM networks to identify if the behavior of volumetric convolutions and LSTM layers remain consistent. These networks are trained with the same set of inputs and labels and the initialization procedures, cost function remain the same as earlier. Apart from the two networks a network with classical recurrent layer is also tested. The model architectures are exactly like the larger column of the neural network models described in Section 5.2.2. The three neural networks trained were,

- A neural network architecture from the large column of the convolutional LSTM (*"L"*) used in two-column experiments,
- A similar C3D network (*"C"*) taken from the two-column network gesture classification network,
- A neural network architecture from the large column of the convolutional LSTM used in two-column experiments with LSTM layer replaced by a recurrent network (*"R"*).

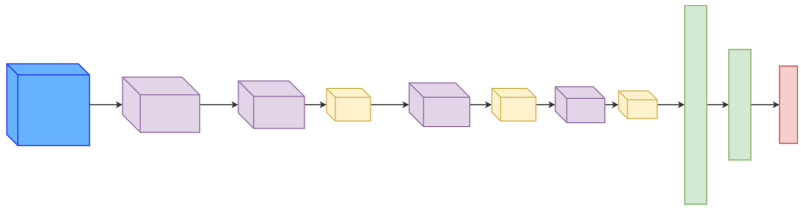


Figure 5.13: The C3D network (*"C"*) : A two channel image is input to the network and the output layer is a softmax layer resulting an output probability of 15 dimensions

Table 5.3: Accuracy with single phase models

Gesture length	*Network: "C"* Accuracy (%)	*"L"* (%)	*"R"* (%)
32	**73.6**	64.6	35.6
16	47.6.7	**51.3**	35.6

The results of Table 5.3 demonstrate that the performance of classical recurrent neural network for the classification is poorer compared with the performance of the neural network architectures that use the LSTM layers or volumetric convolutional layers. This is expected because an RNN is not capable of learning long time contexts.

Looking at the classification performance of Table 5.3, it is also apparent that the performance of the C3D reduces considerably when an early detection of gesture is made using a C3D network. The performance also deteriorates for networks with convolutional layers and an LSTM layer. An important observation is the considerably smoother decay of performance in the network with LSTM layer as compared to the C3D which has volumetric convolution layers. The performance of the LSTM and C3D network is consistent with the observations from the two-column networks tested earlier. The C3D network performs better on the full length sequence but its performance worsens more rapidly than the LSTM network on testing on incom-

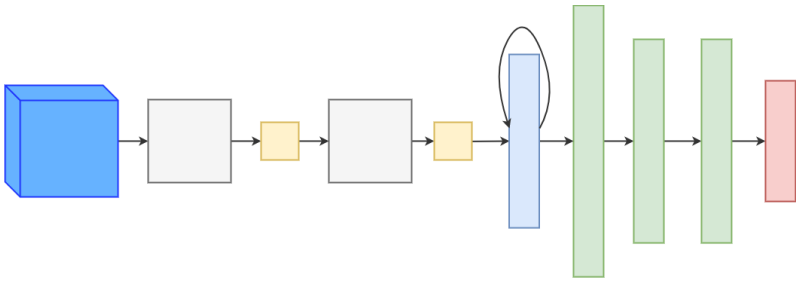


Figure 5.14: The LSTM network (*"L"*) : A two channel image is input to the network and the output layer is a softmax layer resulting an output probability of 15 dimensions

plete gesture sequences. Thus, both single column and the two column networks behave such that the C3D performs better on full sequence gestures and the LSTM network performs better on the shorter sequences.

5.2.5 LSTM and C3D Combination

The observations that C3D consistently perform better on long sequence gestures, while the LSTM network always works better than C3D on shorter sequences encourages the experiments with combinations of the C3D with LSTM. The single phase LSTM and C3D networks trained in Section 5.2.4 were used to create the combination network *"C+L"*. The output probabilities of these trained networks were combined with a separately trained MLP. The MLP learns to combine the output of the probability predictions made by the two separate networks.

The cumulative sum of the LSTM was normalized and a larger thirty eight dimensional vector was created by merging this resulting vector with the C3D output. The MLP is trained with an input of a thirty-eight vector input; the output is the probability vector. The entire system is shown in Figure 5.15.

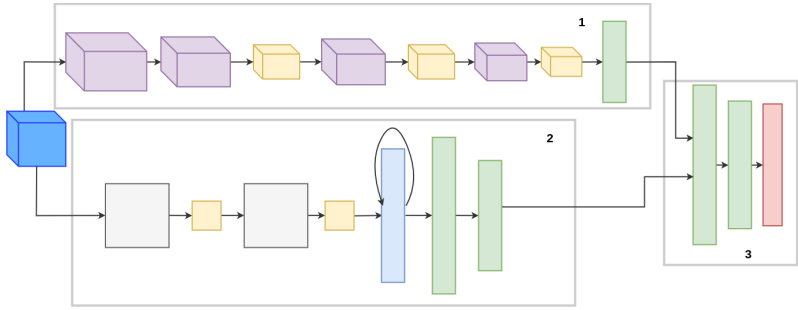


Figure 5.15: The network *"C+L"*: Part 1 and 2 are *"C"* and *"L"* with fiteen dimension softmax output; Part 3 is a simple MLP which learns to combine the outputs.

Training the MLP

The fifteen dimensional probability vector from the C3D and LSTM are combined together to form a thirty vector input to the MLP. The MLP has a hidden layer with sixty four nodes and an output layer of fifteen. The labels of the C3D are used to train the MLP.

The learning rate is initialized to 0.005 and reduced by a factor of two after each fifty steps. The MLP combines the classification probability from the two networks and uses a learning rate of 0.01, it is trained for fifty epochs.

To validate this network, various training and test iterations were made. The networks were trained with reducing latency time. The MLP is separately trained for full length gesture of thirty-two frames, half length sequence of sixteen and for quarter length of eight frame latency. These results are compared with the best results received from either the *"L+l"* or the *"C+c"* network. The class-wise accuracy results are plotted in the Figure 5.16(a) and Figure 5.16(b) for the thirty-two and sixteen frame gestures respectively. The average accuracies are tabulated and compared in the Table 5.4.

The results of this combinatorial networks tabulated in Table 5.4 demonstrate that the network performs slightly worse than the two column C3D network in case of

Table 5.4: Classification Acccuracies of *"C+L"*, *"C+c"* and *"L+l"*; the accuracy of best network is bold.

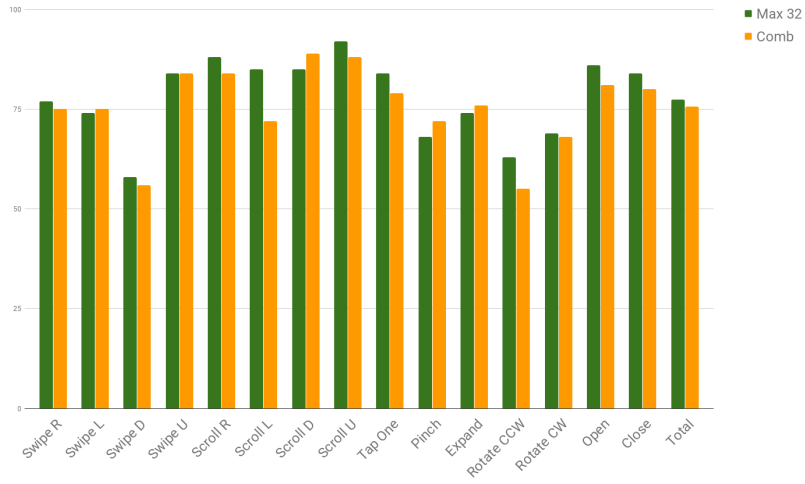
Gesture length	Network: *"C+L"* Accuracy(%)	*"L+l"* (%)	*"C+c"* (%)
32	75.6	73.3	**77.4**
16	**65.7**	62.3	55.7
8	**39**	31.7	28.7

long gestures.

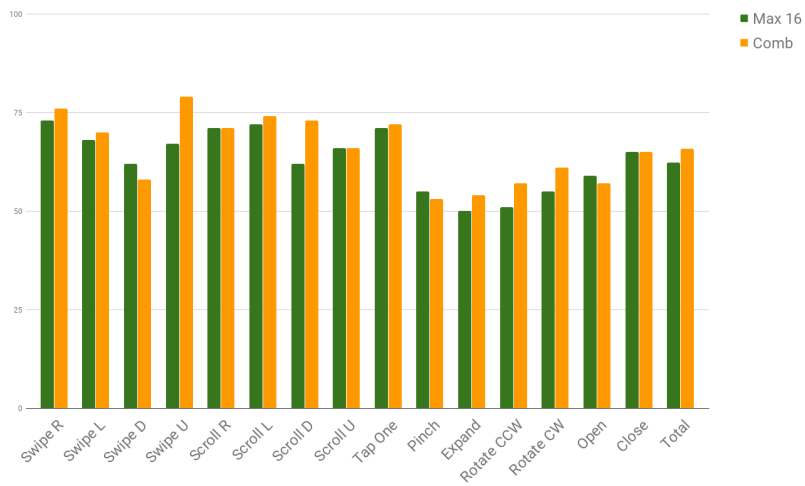
However, the combination network outperforms the two-column LSTM based gesture classifier in every scenario. When classification accuracies are calculated at shorter latency period it is observed that the combinational network starts performing better than the two-column C3D network. For a half length gesture sequence the accuracy of the combinatorial network is 10% higher than the C3D network (reported in Table 5.4), it is also marginally better than the LSTM network by 3%. The combination of the block learning property of the C3D with the contextual learning of the LSTM network may explain the improved performance of the network on shorter incomplete sequences. It is shown that a C3D network continues to perform well on full sequence gestures. The accuracy results for the experiments conducted on the one-fourth length sequences demonstrate similar results. It also shows the difficulty of early identification of the gestures. It is clear that the accuracy rates falls dramatically as the sequence length is reduced.

5.3 Sensor Fusion with Neural Networks

The combinational models discussed earlier suggest that neural network can be a part of various fusion algorithms that combine Inertial Measurement Unit (IMU) and optical data. To evaluate if the information combination discussed in this chapter also works in a sensor fusion sense an architecture for using Kalman filter for



(a) Class-wise average accuracy compared for 32 frame hand gestures



(b) Class-wise average accuracy compared for 16 frame hand gestures

Figure 5.16: Class-wise average performance of the *"C+L"* network(Gold) compared against the best of *"C+c"* and *"L+l"*(Green).

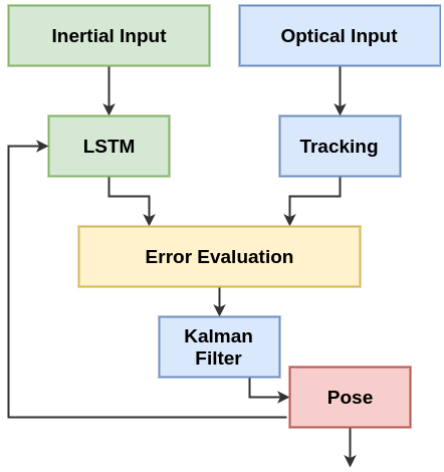


Figure 5.17: Architecture of the proposed fusion system.

fusing the inertial data processing. The classical problem of $6DoF$ pose tracking using accelerometer and a gyroscope inertial values with optical tracking estimate is now proposed to be solved using neural network components. The inertial measurements are used to predict camera pose using an LSTM, which acts as a subfunction $\overline{\mathscr{F}}$ of the fusion problem. The LSTM estimates are fused with optical tracker estimates using the Kalman Filter. The proposed combination is done under the assumption that the camera and inertial sensors are synchronized in the sense that recording is initiated at the same time for both and the inertial sensors frequency of measurements f_s is a multiple of the camera frame capturing frequency f_c so that $C = f_s/f_c$.

5.3.1 Fusion System Architecture

For a frame I_k of the camera the visual tracker provides a a translation vector $\mathbf{t}$ and orientation quaternion $\mathbf{q}$, based on detection and matching of visual features from the image. Apart from this, the inertial input to the LSTM consist of a sixty-seven

dimension input correspond to the last ten inertial measurements($C = f_s/f_c = 10$ in this case) and seven elements of the final pose estimate of the system $[\mathbf{t}|\mathbf{q}]_{k-1}$ corresponding to the previous frame $k-1$. Based on a C previous system outputs and inertial measurements the LSTM is trained to provide an estimation of the current camera pose $[\mathbf{t}|\mathbf{q}]_k^{\text{LSTM}}$.

A comparison module is used to detect failures of the visual tracking system by comparing it to the output of the LSTM. The inertial tracking done by the LSTM can slowly drift away from the correct pose. However, it does not abruptly produce highly erroneous outputs. On the other hand the visual tracking system can output estimates with high error for some frames. In order not to allow these errors to contaminate the fusion system output the comparison module calculates two distance metrics d_t, d_q between the LSTM output and the visual output, where d_t is the euclidean distance between the estimated positions and d_q is the distance between the corresponding angles of the estimated quaternions. If one of d_t,d_q are found to exceed a threshold it is taken as an indication of failure of the visual system, and only the output of the LSTM is passed to the next module. This comparison module is only used as a safeguard against heavily erroneous poses from the visual tracker and is activated very rarely. The system is thus able to retain a correct pose for some video frames even if visual tracking fails. However, because of the drift of inertial tracking, if no correct pose is given from the visual system for a large number of consecutive the system would have to be reinitialized.

Finally, the Kalman Filter module is used to combine the estimated pose from the LSTM with the estimated pose from the visual system. A simple linear Kalman filter model is used. The unitary quaternions $\mathbf{q}$ representing orientation are always renormalized at the output of the LSTM and the Kalman Filter. The entire system is shown in Figure 5.17

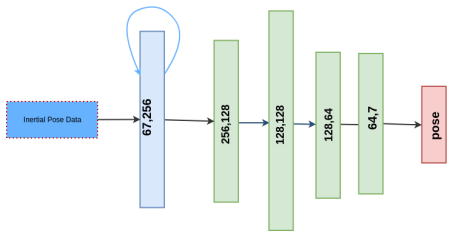


Figure 5.18: Architecture of the proposed LSTM for camera pose estimation from inertial measurements.

Visual tracker

Features are extracted from each received frame image I_k and matched to the registered features based on descriptor similarity. Subsequently, a ratio test between first and second match similarity is performed in order to exclude ambiguous matches. Using the selected matched features $2D$ points in the current image and their known $3D$ correspondences from registration, refined pose estimation is achieved within a RANSAC scheme. The visual tracker outputs a camera pose estimate $[\mathbf{t}|\mathbf{q}]_k^{\text{vision}}$ for every frame consisting of a rotation quaternion $\mathbf{q}$ and translation vector $\mathbf{t}$ with respect to the registered targets' coordinate system, or an error message indicating failure of the visual tracker.

LSTM

A neural network architecture with an LSTM layer is trained as a regressor. The neural network is such that the input vector includes all the sensor outputs over the time period in which the optical system makes a camera-pose estimation. A ground truth with inertial measurements and accurate camera pose is recorded for training. At time instance k the LSTM receives the estimated camera pose $[\mathbf{t}|\mathbf{q}]_{k-1}$ system output through the feedback channel as seen in Figure 5.17. Given this input, the neural network is trained to produce an estimate of the current camera

pose $[\mathbf{t}|\mathbf{q}]_k^{\text{LSTM}}$. The regression estimates of the neural network are made by minimizing the error function used in [159]. This is practically a mean square error function with a scaling parameter for the quaternions error in order to bring position and quaternion error values to approximately the same level. Various neural network architectures were trained and the one providing the best results is further described in detail. The training uses the RPROP Algorithm for the optimization process [154].

The neural network architecture is such that the input layer is followed by an LSTM layer which captures the temporal relationships in the input sequence. The LSTM layer has 256 nodes. Three more fully connected layers are further added to the network. The output layer is a seven node layer, representing the spatial location $\mathbf{t}$ and the orientation quaternion $\mathbf{q}$ of the camera. Non-linearity is added to the network by using a *tanh* activation function with each layer apart from the output layer Figure 5.18. The LSTM network is unwrapped such that the input is fed as a sequence of vectors generated from the sensors and the camera pose. Decision predictions are made in a moving window style, such that the window slides over the incoming sequences. The network is trained on a training dataset with ground truth camera poses obtained from observation of fiducials and corresponding inertial measurements. The trained LSTM can be treated as a subfunction $\overline{\mathscr{F}}$ of the sensor fusion problem function $\mathscr{F}$, the LSTM function $\overline{\mathscr{F}}$ returns a pose estimate in frame k using the previous pose estimates from frames $k-1$ to $k-i$ where i is the depth of the regression and inertial sensor measurements.

Kalman filter

The sensor fusion system uses a linear Kalman Filter to fuse the estimates of the LSTM and the optical tracker. The filter state is also a vector consisting of a position $\mathbf{t}$ and orientation quaternion $\mathbf{q}$ together represented by $\mathbf{v}$. Thus, the sensor fusion

system uses a linear Kalman Filter to fuse the estimated poses $\mathbf{v}_k^{\text{LSTM}}$ and $\mathbf{v}_k^{\text{vision}}$ from the LSTM and the visual tracking respectively, into a final system output pose estimate $\mathbf{v}_k$. During the prediction step of the filter, instead of using the previous state of the filter the pose estimate from the LSTM is used. Thus, the resulting prediction equations are

$$\hat{\mathbf{v}}_k = \mathbf{F}_k \mathbf{v}_k^{\text{LSTM}} \tag{5.1a}$$

$$\Sigma_{k|k-1} = \mathbf{F}_k \Sigma_{k-1|k-1} \mathbf{F}_k^T + \mathbf{Q}_k, \tag{5.1b}$$

where $\mathbf{F}_k$ is the state transition matrix which we set to an identity matrix, Σ_k is the state covariance matrix, and $\mathbf{Q}_k$ is the process noise covariance matrix which we set to a diagonal matrix containing the variance of the noise estimated at the outputs of the LSTM. The noise estimate is obtained based on the error measurement made on the training data. The update of the filter is performed by using the output of the visual tracking system $\mathbf{v}_k^{\text{vision}}$ as the measurement.

Evaluation of the Proposed Architecture

The experiments were completed on a printed poster images surrounded by number of fiducials. These fiducials are described in detail in [160]. A poster is used as the target of the visual tracking system described and the fiducials are exclusively used for the purpose of obtaining ground truth for camera pose and the final evaluations do not use the information from the fiducials. The camera frames were used in the experiment without any preprocessing. The camera and inertial sensor respective coordinate frames are not aligned and no hand-eye calibration between them was used, the LSTM is expected to learn these properties. The same holds for gravity removal from acceleration measurements.

During the experiments three image sequences and corresponding synchronized inertial measurements datasets were captured. Out of these datasets, the first one with a length of 9800 video frames and 98000 inertial sensor measurements was

Table 5.5: Tracking accuracy comparison between a pure visual approach and the LSTM based physical model fusion system.

	Overlap %	**Failed Pose estimate(%)**
sequence 1(slow) visual	86.3%	7.3%
sequence 1(slow) fusion	90.8%	0%
sequence 2(fast) visual	77.1%	15.7%
sequence 2(fast) fusion	85.6%	0%

used for training the LSTM, and the other datasets of shorter duration (3389 and 3106 video frames) were used for evaluation of the LSTM and of the system as a whole. All datasets were recorded with the capturing device hand-held and contain fast translational motion in all directions, as well as fast rotational motion along all three axes, and combinations of translational and rotational motion.

The evaluations are made by measuring the overlap between the re-projection rectangle estimated by the system output camera pose, and the expected rectangle. The average overlap percentage between the visual tracker and the inertial-visual tracker is compared for two sequences of images. One sequence with smoother or slow motion and the other with more abrupt or fast motions. The results of Table 5.5 show that using the fusion approach an increase can be achieved in the overlap percentage in both the slow and the fast sequence.

This experiment was to demonstrate that while in the earlier sections the model information and independent information separation improved the classification performance of the networks. A trained neural network can be used as a replacement for the prediction models in a fusion system, where the neural network contributes to the dynamics of the system, while the measurement modifies the noise performance.

5.4 Conclusion

That multi-modal features improve the performance of a machine learning algorithm has been known, it was demonstrated with IMU-visual data on the body gesture recognition problem. This is further extended to show that the multi-property learning network perform better than the single information networks. This hypothesis was tested with the combination of such sequential learning networks and the time filtering networks used simultaneously. This network combination has been demonstrated on the hand gesture sequences and it was shown that the combination of the LSTM and C3D networks provide better early gesture classification properties. Finally, in contrast to the models improving the LSTM estimates in Chapter 3, in this chapter it was demonstrated the learnt LSTM model is used as a prediction model whose estimates can be corrected by measurements from another sensor in the Kalman filtering setup.

Chapter 6

Application

The previous chapters introduced various algorithms for the hand shape and gesture classification, these methods tried to solve the classification problems with small neural network models and minimized data requirements. This chapter describes the outlines of an actual in-car hand shape and gesture classification system. The data collection strategy, the architecture of such a system, and the validation of the results is described in detail.

In the Section 6.2 the set-up of the proposed application is defined. The sensor choice, the choice of the position of the sensor, the location of the gesture and shape detection area is described in this section. The reasons for these choices are also provided. The same section also describes the dataset captured in the car which the application is trained and tested against. In the next Section 6.2.2, the flow of the data is described. The input, preprocessing, tracking, and pre-processing blocks are explained in some details. The decision block which use the neural network is described in the Section 6.3.1, in the same section the eventual neural network model used for the shape and gesture classification procedure is presented. The results of tests made on the data and the computation machine used for the tests are

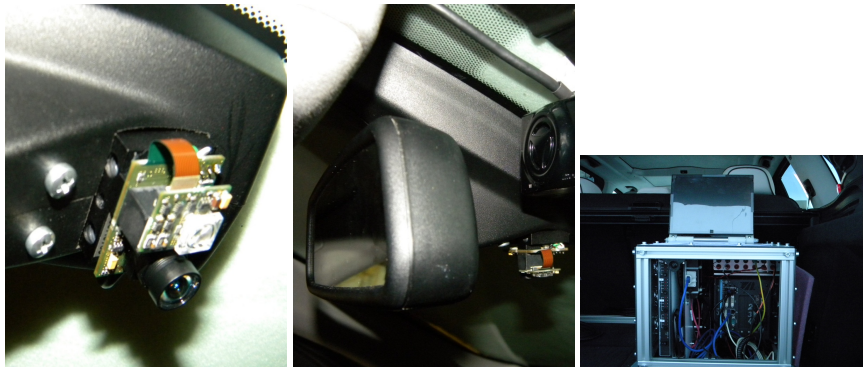
also detailed in the section Section 6.3.1.

6.1 Application Motivation

Driver distraction is a critical issue for driving safety. The Driver Information System (DIS), music players, navigation system are some of the components often shipped along with the automobiles. These are important assistance for the driver and co-passengers but also contribute to the driver distractions.

One of the early and still popular empirical evaluation of distraction in car [161] recommended fifteen seconds as the maximum time for completing any interaction with DIS. Since, various studies have been conducted to analyze the influence on driver concentration when they interact with these objects in the car [1, 162, 2]. These studies found that distraction disrupts natural driving and causes errors. The study by [2] lists dialing or answering a mobile phone, adjusting MP3 players among the causes for distraction, it has been experimented that such distraction contributed to road accidents. It was observed that the time taken to complete simple interactions with the electronic elements within the car ranged from 8.5 seconds to 25 seconds [1]. It was also evaluated that the complexity of the interaction increases the distraction time and the length of interaction. In this context, one of the component of interaction that can be reduced is the searching for controls and pressing, rotating the controls. The work by [107] preferred interactions and using gesture based interaction over other interaction because this can remove the visual distraction. This study was further validated by the experiments conducted by [163] who reported reduced visual distraction by employing on-steering hand gestures.

The development of a hand-gesture based in-car HCI system is motivated from these studies. The proposed application will develop a high accuracy hand shape and gesture classifier, with the constraints of a preferred low latency identification, without high computation required.



(a) The ToF sensor for the application. (b) Placement of the ToF sensor. (c) The application setup in the car.

Figure 6.1: The 3D ToF camera location in the car and the recording setup

6.2 The Sensor and Application Pipeline

A low cost, low power, ideally 3D image sensor is required for the application. The various solutions available for recording the 3D depth images includes the stereo vision based combination of two 2D sensors or one 2D camera with pattern projection. These solutions are typically susceptible to changing lighting conditions. This happens because of the changing shadows in the moving car in case of the projections based depth calculations. In case of the stereo vision the changing lights and light angles create a varying contrast map for the image and such systems are known to fail in scene content that has no or only rare contrast.

The ToF camera is developed on the principles of reflection of modulated light source and its phase detection. Thus the depth calculation (the distance between the sensor and the object) of each pixel is calculated by accurate measures of time difference from the light source until it is received back by the sensor after reflection. This makes the sensor less susceptible to the varying lighting conditions within the camera field of view. The second amplitude channel has the important

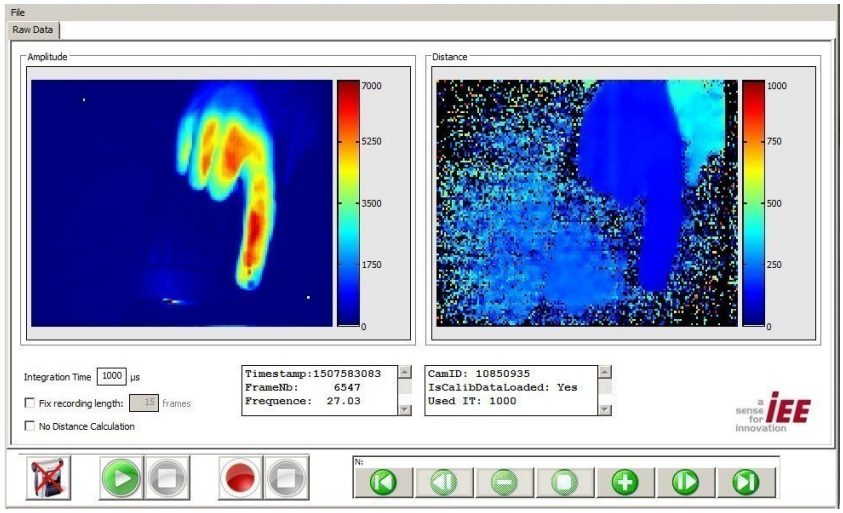


Figure 6.2: The recording tool for recording data in the car.

property that the pixel values are dependent on the reflectance of the object surface, this makes the properties of the amplitude channel similar to a grey scale channels.

6.2.1 Sensor Placement

Avoiding driver distraction, required short gestures done close to the steering wheel. When the actions are committed in a small region close to the steering, minimizing the scene clutter and occlusions are important for making a decision on the placement of the camera. It has been observed that an overhead location is best suited for such problems [164]. While, this choice causes substantial self occlusion of the hand when palm points towards the car floor it is the optimal location of the camera to remove the occlusion due to objects in the car. A ToF camera was fixed in the overhead console behind the rear view mirror of the car. The data for the proposed application is collected inside the car using a data collection tool. The recordings of the two channel images are made to test the algorithm on the desktop version

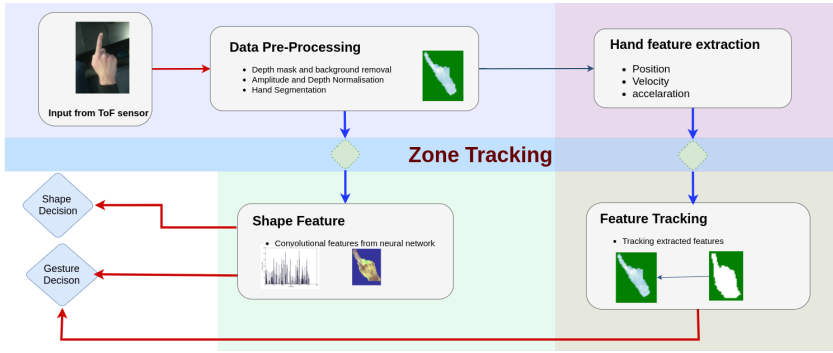


Figure 6.3: The flow for the application architecture.

and a recording tool was developed for recording hand shapes and gestures in the car Figure 6.2.

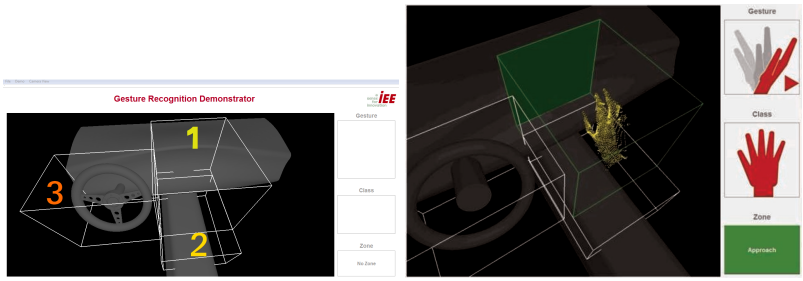
6.2.2 Application Architecture

A desktop application to integrate the gesture and shape classification application was used for including the proposed methodologies. The application proposal uses existing motion and position feature and feature tracking blocks. These features are used along with the features extracted from neural networks, these features are the same as were described in the Section 4.5.4

1. Sensor image collection and preprocessing the sensor output.

2. The hand position, velocity vectors are calculated using the hand center and the fingertip location.

3. The convolution neural network is used to make a shape prediction. The shape features of convolutional network is combined with the velocity features calculated by tracking and are input into an LSTM.



(a) The car zones used in the application (b) Gesture and shape identification in zone 1.

Figure 6.4: Screenshots from the car applications.

The sensor provides a 120×165 pixel two-channel image. In the application, the depth channel is employed for background removal. The largest contour in the amplitude image is then identified and conserved, this removes the noisy pixels. The depth values corresponding to the pixel position in the amplitude channel are used for re-scaling the hand image to equal depth. The image is centered in region of interest. The car used for the recording and development of the application is divided into three zones. A cuboidal area is identified around the car dashboard and marked as zone one, the area around the gear box is identified as zone two and area around steering area is identified as zone three, Figure 6.4(a). The application identifies hand gestures and shapes in the zone one. As the gestures are palm movement, this constraint removes various unintended gestures that are otherwise identified, it also allows better pre-processing of the data. The features calculation is done exactly as defined in Section 4.5.4. Of the seventeen feature components, fourteen values which include the position, velocity and acceleration of the hand and finger are used as the input to the LSTM network. The features that represent the shape of the hand used in the Section 3 are replaced by the shape representation vector from a convolutional neural network used for shape classification.

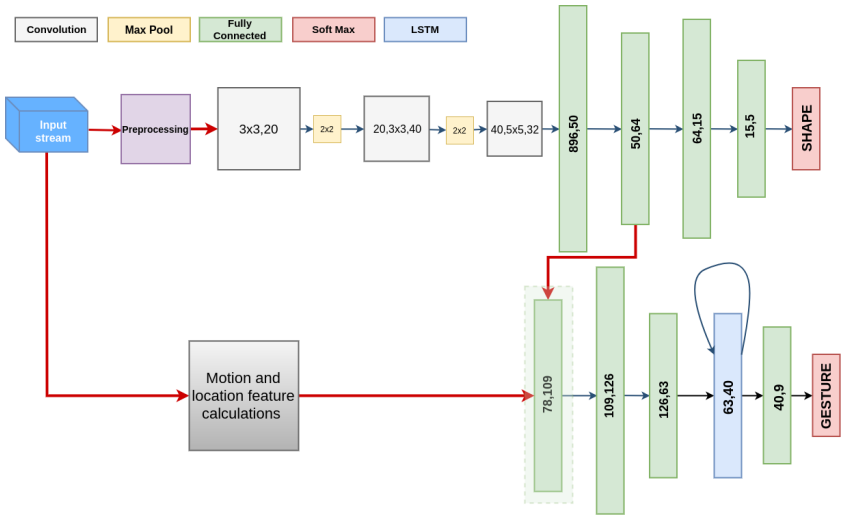


Figure 6.5: The neural network architectures.

6.3 Neural Networks and Tests

The application simultaneously identifies the hand gesture and hand shapes. Both shape and gesture are identified only when the hand is in an area of interest. The hand shape classifier is also used as a feature generator for the hand gesture classifier.

The architectures for the two neural networks used for shape and gesture classification is described in the Figure 6.5. The shape classifier is the same as the one described in the Chapter 3. The gesture classifier is a simple LSTM network with five fully connected layers. The input to the LSTM network is the position and motion features combined with sixty-four dimension hand shape descriptor resulting from the second layer of the network used for the shape classification.

The neural networks are activated when the hands are in the zone 1 shown in Figure 6.4(a), the neural network makes a softmax prediction of the shape and gesture

class.

6.3.1 Neural Network Training and Test

The earlier used hand shape dataset described in Chapter 3 is combined with a dataset recorded for gestures.

Training the shape classifier

The Neural Networks are trained using a hybrid dataset of shape and gesture recordings made in the desk and car. The recordings made for the desk hand dataset described in Section 4.5.1 are combined with a recording made using the same set-up but inside the car. The recording inside the car is made for thirteen individuals. The network is first initialized by combining the large hand shape dataset that has one hundred five thousand data samples, with the car dataset in which there are eighteen thousand samples. Nine thousand samples from car data are used in this initialization.
In the second phase of the training only the car-dataset is used and fifteen thousand of the car data points including those used in initial training are used for training the classification. The overall training is completed for classification of five hand shapes. The second phase of training tries to specify the network into the a in-car hand shape classification network.

Training the gesture classifier

The gesture classifier is trained similar to the hand-shape classifier. A desk-car hybrid hand gesture dataset is used for initial training of the network, the training is finalized with the car dataset and the final network is used for the training of the seven gesture class classifier.

Table 6.1: The application: classification accuracy of the five hand shapes.

Accu %	Fist	Flat	Open	Point	Join
	94	**91**	**96**	**93**	**96**

The dataset for the training of the gesture is created by first collecting the hand shape representation vector for each frame of the gesture and concatenating it with the motion feature classifier calculated from the hand and finger location tracking.

6.3.2 Forward Phase and Application Evaluations

The forward phase of the application has the hand shape classification and the hand gesture classification running simultaneously. The scheme used for these two classification as shown on Figure 6.5 causes a delay in the gesture prediction as the classifier waits for the feature input from the first first hand shape input. To ascertain that the hand-shape features include the full hand image, the application waits $t_d = 2$ frames after the first hand shape prediction is made to start the input into the LSTM network. The LSTM then makes the first prediction after a delay of $K = 10$, thus the overall delay of the LSTM prediction is $T_d = t_d + K + 1 = 13$ frames.
The classified hand shapes are "Fist", "Point", "Flat", "Open" and "Join". The classified hand gestures are "Up", "Down" movements, "Left" and "Right" swipe, and "Click". The performance is tested on an Intel® Xeon CPU. The time for the forward phase of the CNN is 6.7ms and the time for running the LSTM once the input from the convolution network and the tracking features are received is 4.2 ms.
These networks were tested with two thousand gestures and ten thousand hand shapes, the accuracy results for these tests are shown in the Table 6.1 and Table 6.2.

The percentage accuracies of the hand shape and hand gestures are the average correct estimate made by the respective neural networks where the class estimate for the output softmax vector $\mathbf{P}$ containing elements p_i the estimated class i is cal-

Table 6.2: The application: classification accuracy of the five hand gestures.

Accu %	Up	Down	Left	Right	Click
	91	**89**	**89**	**92**	**95**

culated as, $\max_i(\mathbf{P})$ for the shape classification problem, and $\max_i(\sum_{t=T_d}^{\tau} P_t)$ for a gesture sequence of length τ.

6.4 Conclusion

Various applications in the car can use the vision based HCI system. The application has so far been tested on a desktop PC and will be integrated into the car system soon. The use of small models and shared features allows an architecture that requires less memory and can be used in real time. The application uses known car dimensions for pre-processing of the hand images. The training of the network used data collected on the desktop setup described in Section 4.5.1 and was corrected with the data recorded inside the car.

Eventually, this algorithm should be a part of the car embedded software and the driver should be able to control various systems in the car with it. Such use cases will still require further compression of the neural network models and better integration on the car embedded system.

Chapter 7

Conclusion

The goals of the thesis broadly covered developing methods for feature evaluation, identifying methods for improving the feature learning and feature combination methods for classification. This goal was inspired from the attempt to reduce the time required for collecting the data, this includes both the process of collecting a large dataset as well as reducing the requirement for labeled data.

While the neural network based machine learning algorithms are powerful tools for learning, performance improvement with constant data and model size is difficult to achieve. Simultaneously, developing an automobile based system that needs to work on varying car interiors and constantly changing illumination the dataset variance is large. Further, labeling gesture sequences and marking the start and end of sequences in large dataset requires manual interventions. The development of a real-time system that runs on limited computing power inside a car also creates constraints on the size of the neural network model.

The methods for reduction of training data were developed by, firstly, developing schemes for better combination of the a-priori physical and statistical information, and secondly, by proposing a scheme for combination of neural network. In

Chapter 3 introduced a step by step procedure of increasing physical and statistical information about data, task and environment for improving the classification performances. In the Chapter 4 the combination of the features is shown to improve the learning performance and a method for parallel learning of features consisting of separate properties is proposed. The improvement of performance using such strategies were shown with experiments on the hand shape and hand gesture datasets.

7.1 Summary of Thesis Achievements

Various methods for hand shape and pose classification already exist. The hand shape and hand gesture classification system presented in this system uses the ideas developed for effective transfer learning and better weight initialization schemes. Emphasis was placed on developing methods of inculcating priors into the neural network. To review, the contributions of the thesis as they appeared in this document are as listed,

- A modification to the SFA extraction procedure for better exploitation of orthogonal features. This results in better performance of hand shape classification problem.
- Development of Random Forest based methods for the evaluation of the contribution of features from different layers to the classification problem.
- Proposing a funnel shape output layer for hand shape neural network classifier, and introducing statistical priors to it.
- Developing a constituent transfer scheme for intra-domain transfer learning.

- Using a weak task model for the correction of LSTM sequence estimates using CNN static estimates. The gesture estimates made by a LSTM were corrected by the CNN hand shape estimates.
- A multi-property combination of neural network features for faster sequence classification. The combination of the C3D network with LSTM network is done for better classification of incomplete gesture sequences.
- A proposed simultaneous in-car hand-shape and hand-gesture classification algorithm that runs real time on a desktop test PC.

The contributions include theoretical and practical solutions for feature extraction and feature uses problems. The experiments on orthogonality and feature separation provide a perspective for understanding feature learning. Further, methods for prior learning involves ideas borrowed from the fields of domain transfer learning and sensor fusion for use with neural networks.

7.2 Future Work

Many different adaptations and theoretical formulations have been left to be completed. The preparation of the data, searching for the optimum model and training the neural network with the dataset is a time consuming process. Thus, experiments need to be made in future to better understand parts of work presented in this thesis. Here we list the most important experiments that should be made further,

1. The evaluation on neural network features' contributions to the classification task made in Chapter 3, may be used for model pruning. A method to develop such pruning, by developing sparsely connected network and then reducing the network redundancies could be useful for developing a procedural method for model size reduction.

2. Further, the constituent learning proposed in the Section 4.4 requires formalization of the properties that can be used for information transfer. In the reported experiments, intuitive properties for model information learning were used. A better approach would be a mathematical formalization for identification of such properties. Further experiments with varying sensors and real data will strengthen the arguments for the importance of the method.

Apart from this, recently presented variations of SFA called slow and steady feature analysis [165] should be analyzed. An earlier proposed network pruning method [147] can be used in conjugation with the feature evaluation work.
Overall, various contributions of this thesis, which included designing biases, developing methods for domain transfer, and evaluating feature contributions of neural network will be of interests in developing strategies to better initialize neural network with a known mathematical formulation for the phenomenon of the dataset or task. Such methods will substantially reduce the labelled data requirement. Better transfer learning policy will also help researchers use large datasets more creatively for tasks to be completed on independent dataset received from another sensor.

Bibliography

[1] Jeff Greenberg, Louis Tijerina, Reates Curry, Bruce Artz, Larry Cathey, Dev Kochhar, Ksenia Kozak, Mike Blommer, and Peter Grant. Driver distraction: Evaluation with event detection paradigm. *Transportation Research Record: Journal of the Transportation Research Board*, (1843):1–9, 2003.

[2] Kristie L Young and Paul M Salmon. Examining the relationship between driver distraction and driving errors: A discussion of theory, studies and methods. *Safety science*, 50(2):165–174, 2012.

[3] Michael Nielsen, Moritz Störring, Thomas B Moeslund, and Erik Granum. A procedure for developing intuitive and ergonomic gesture interfaces for hci. In *International gesture workshop*, pages 409–420. Springer, 2003.

[4] Justine Cassell. A framework for gesture generation and interpretation. *Computer vision in human-machine interaction*, pages 191–215, 1998.

[5] Norbert A Streitz, Peter Tandler, Christian Müller-Tomfelde, and Shin'ichi Konomi. Roomware: Towards the next generation of human-computer interaction based on an integrated design of real and virtual worlds. *Human-Computer Interaction in the New Millenium, Addison Wesley*, pages 551–576, 2001.

[6] Quentin De Smedt, Hazem Wannous, and Jean-Philippe Vandeborre.

Skeleton-based dynamic hand gesture recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–9, 2016.

[7] Viva. http://cvrr.ucsd.edu/vivachallenge/index.php/hands/hand-gestures/.

[8] Cem Keskin, Furkan Kıraç, Yunus Emre Kara, and Lale Akarun. Real time hand pose estimation using depth sensors. In *Consumer Depth Cameras for Computer Vision*, pages 119–137. Springer, 2013.

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

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

[11] Aditya Tewari, Bertram Taetz, Frederic Grandidier, and Didier Stricker. Using mutual independence of slow features for improved information extraction and better hand-pose classification. *International Conferences in Central Europe on Computer Graphics, Visualization and Computer Vision, Journal of WSCG*, pages 35–43, 2015.

[12] Pietro Berkes. Pattern recognition with slow feature analysis, February 2005. URL `http://cogprints.org/4104/`.

[13] Aditya Tewari, Fangsheng Gu, Frederic Grandidier, and Didier Stricker. Quantitative evaluation of salient deep neural network features using random forest. IEEE IVMSP 2018, Aristi Greece 10-12 June 2018, 2018.

[14] Aditya Tewari, Frederic Grandidier, Bertram Taetz, , and Didier Stricker. Adding model constraints to cnn for top view hand pose recognition in range images. In *Proceedings of International Conference on Pattern Recognition Applications and Methods (ICPRAM, ISBN 978-989-758-173-1), Rome, Italy*, pages 170–177, 2016.

[15] Aditya Tewari, Bertram Taetz, Frederic Grandidier, and Didier Stricker. Two phase classification for early hand gesture recognition in 3d top view data. In *Lecture Notes in Computer Science, International Symposium on Visual Computing (ISVC)*, volume 10072, pages 353–363. Springer, 2016.

[16] Jonathan Tompson, Murphy Stein, Yann Lecun, and Ken Perlin. Real-time continuous pose recovery of human hands using convolutional networks. *ACM Transactions on Graphics (ToG)*, 33(5):169, 2014.

[17] Tewari Aditya, Gu Fangsheng, Grandidier Frederic, and D Stricker. A probabilistic combination of cnn and rnn estimates for hand gesture based interaction in car. In *16th IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2017.

[18] Pavlo Molchanov, Shalini Gupta, Kihwan Kim, and Jan Kautz. Hand gesture recognition with 3d convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–7, 2015.

[19] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2625–2634, 2015.

[20] Rambach JR, Tewari A, Pagani A, and D Stricker. Learning to fuse: A deep learning approach to visual-inertial camera pose estimation. In *Proceedings of 15th IEEE International Symposium on Mixed and Augmented Reality.*, 2016.

[21] Aditya Tewari, Bertram Taetz, Frederic Grandidier, and Didier Stricker. Learning to transfer constituent features in convolutional neural networks. Submitted to Journal of WSCG 2018, Pilsen,May 28 - June 1, 2018, 2018.

[22] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE, 2001.

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

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

[25] Juha Parkka, Miikka Ermes, Panu Korpipaa, Jani Mantyjarvi, Johannes Peltola, and Ilkka Korhonen. Activity classification using realistic data from wearable sensors. *IEEE Transactions on information technology in biomedicine*, 10(1):119–128, 2006.

[26] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[27] Yoshua Bengio. A connectionist approach to speech recognition. In *Advances in Pattern Recognition Systems Using Neural Network Technologies*, pages 3–23. World Scientific, 1993.

[28] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

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

[30] John R Williams and Kevin Amaratunga. Introduction to wavelets in engineering. *International journal for numerical methods in engineering*, 37 (14):2365–2388, 1994.

[31] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *IEEE transactions on pattern analysis and machine intelligence*, 27(10):1615–1630, 2005.

[32] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3): 346–359, 2008.

[33] Piotr Dollár, Zhuowen Tu, Pietro Perona, and Serge Belongie. Integral channel features. 2009.

[34] Y. Boureau, F. Bach, Y. LeCun, and J. Ponce. Learning mid-level features for recognition. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2559–2566, June 2010.

[35] Quoc V Le, Will Y Zou, Serena Y Yeung, and Andrew Y Ng. Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3361–3368. IEEE, 2011.

[36] Geoffrey E Hinton, Michael Revow, and Peter Dayan. Recognizing handwritten digits using mixtures of linear models. In *Advances in neural information processing systems*, pages 1015–1022, 1995.

[37] Du-Ming Tsai and Shia-Chih Lai. Independent component analysis-based background subtraction for indoor surveillance. *IEEE Transactions on image processing*, 18(1):158–167, 2009.

[38] Chengjun Liu and Harry Wechsler. Independent component analysis of gabor features for face recognition. *IEEE transactions on Neural Networks*, 14 (4):919–928, 2003.

[39] Ian T Jolliffe. Principal component analysis and factor analysis. In *Principal component analysis*, pages 115–128. Springer, 1986.

[40] Matthew A Turk and Alex P Pentland. Face recognition using eigenfaces. In *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR'91., IEEE Computer Society Conference on*, pages 586–591. IEEE, 1991.

[41] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.

[42] Marzia Polito and Pietro Perona. Grouping and dimensionality reduction by locally linear embedding. In *Advances in neural information processing systems*, pages 1255–1262, 2002.

[43] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500): 2319–2323, 2000.

[44] Jaron Blackburn and Eraldo Ribeiro. Human motion recognition using isomap and dynamic time warping. In *Human motion–understanding, modeling, capture and animation*, pages 285–298. Springer, 2007.

[45] Markus Turtinen, Matti Pietikainen, and Olli SilvÉn. Visual characterization of paper using isomap and local binary patterns. *IEICE transactions on information and systems*, 89(7):2076–2083, 2006.

[46] Zhenhua Guo, Lei Zhang, and David Zhang. A completed modeling of local binary pattern operator for texture classification. *IEEE Transactions on Image Processing*, 19(6):1657–1663, 2010.

[47] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4): 115–133, 1943.

[48] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[49] S. Dreyfus. The computational solution of optimal control problems with time lag. *IEEE Transactions on Automatic Control*, 18(4):383–385, August 1973. ISSN 0018-9286. doi: 10.1109/TAC.1973.1100330.

[50] Paul Werbos and Paul J. (Paul John. Beyond regression : new tools for prediction and analysis in the behavioral sciences /. 01 1974.

[51] Sankar K Pal and Sushmita Mitra. Multilayer perceptron, fuzzy sets, and classification. *IEEE Transactions on neural networks*, 3(5):683–697, 1992.

[52] PJ Werbos. Beyond regression: New tools for prediction and analysis in the behavioral sciences. ph. d. thesis, harvard university, cambridge, ma, 1974. 1974.

[53] Arthur Earl Bryson. *Applied optimal control: optimization, estimation and control*. CRC Press, 1975.

[54] Yann LeCun, D Touresky, G Hinton, and T Sejnowski. A theoretical framework for back-propagation. In *Proceedings of the 1988 connectionist models summer school*, pages 21–28. CMU, Pittsburgh, Pa: Morgan Kaufmann, 1988.

[55] Geoffrey Sampson. Parallel distributed processing: Explorations in the microstructures of cognition, 1987.

[56] S. Amari. A theory of adaptive pattern classifiers. *IEEE Transactions on Electronic Computers*, EC-16(3):299–307, June 1967. ISSN 0367-7508. doi: 10.1109/PGEC.1967.264666.

[57] Alexander Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J Lang. Phoneme recognition using time-delay neural networks. In *Readings in speech recognition*, pages 393–404. Elsevier, 1990.

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

[59] Patrice Y Simard, David Steinkraus, John C Platt, et al. Best practices for convolutional neural networks applied to visual document analysis. In *ICDAR*, volume 3, pages 958–962, 2003.

[60] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. *CoRR*, abs/1612.01925, 2016. URL `http://arxiv.org/abs/1612.01925`.

[61] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: To-

wards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[62] Teuvo Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9): 1464–1480, 1990.

[63] Dominik Scherer, Andreas Müller, and Sven Behnke. Evaluation of pooling operations in convolutional architectures for object recognition. In *International conference on artificial neural networks*, pages 92–101. Springer, 2010.

[64] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038, 2014. URL `http://arxiv.org/abs/1411.4038`.

[65] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *CoRR*, abs/1412.7062, 2014. URL `http://arxiv.org/abs/1412.7062`.

[66] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1520–1528, 2015.

[67] Sepp Hochreiter. Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*, 91:1, 1991.

[68] Justin Bayer, Daan Wierstra, Julian Togelius, and Jürgen Schmidhuber. Evolving memory cell structures for sequence learning. In *International Conference on Artificial Neural Networks*, pages 755–764. Springer, 2009.

[69] Alex Graves, Abdel-rahman Mohamed, and Geoffrey E. Hinton. Speech

recognition with deep recurrent neural networks. *CoRR*, abs/1303.5778, 2013. URL http://arxiv.org/abs/1303.5778.

[70] Alexander Förster, Alex Graves, and Jürgen Schmidhuber. Rnn-based learning of compact maps for efficient robot localization. In *ESANN*, pages 537–542, 2007.

[71] Moez Baccouche, Franck Mamalet, Christian Wolf, Christophe Garcia, and Atilla Baskurt. Sequential deep learning for human action recognition. In *International Workshop on Human Behavior Understanding*, pages 29–39. Springer, 2011.

[72] Moez Baccouche, Franck Mamalet, Christian Wolf, Christophe Garcia, and Atilla Baskurt. Spatio-temporal convolutional sparse auto-encoder for sequence classification. In *BMVC*, pages 1–12, 2012.

[73] KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, abs/1409.1259, 2014. URL http://arxiv.org/abs/1409.1259.

[74] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[75] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

[76] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *CoRR*, abs/1410.3916, 2014. URL http://arxiv.org/abs/1410.3916.

[77] Jonathan Baxter. Learning to learn. chapter Theoretical Models of Learning to Learn, pages 71–94. Kluwer Academic Publishers, Norwell, MA, USA, 1998. ISBN 0-7923-8047-9. URL http://dl.acm.org/citation.cfm?id=296635.296644.

[78] Bianca Zadrozny. Learning and evaluating classifiers under sample selection bias. In *Proceedings of the twenty-first international conference on Machine learning*, page 114. ACM, 2004.

[79] John Blitzer, Mark Dredze, and Fernando Pereira. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th annual meeting of the association of computational linguistics*, pages 440–447, 2007.

[80] John Blitzer, Ryan McDonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 120–128. Association for Computational Linguistics, 2006.

[81] Zheng Wang, Yangqiu Song, and Changshui Zhang. Transferred dimensionality reduction. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 550–565. Springer, 2008.

[82] Rie Kubota Ando and Tong Zhang. A high-performance semi-supervised learning method for text chunking. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 1–9. Association for Computational Linguistics, 2005.

[83] Neil D Lawrence and John C Platt. Learning to learn with the informative vector machine. In *Proceedings of the twenty-first international conference on Machine learning*, page 65. ACM, 2004.

[84] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[85] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[86] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 513–520, 2011.

[87] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y Ng. Self-taught learning: transfer learning from unlabeled data. In *Proceedings of the 24th international conference on Machine learning*, pages 759–766. ACM, 2007.

[88] Tuan Anh Le, Atilim Gunes Baydin, Robert Zinkov, and Frank Wood. Using synthetic data to train neural networks is model-based reasoning. *CoRR*, abs/1703.00868, 2017. URL `http://arxiv.org/abs/1703.00868`.

[89] Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *European Conference on Computer Vision*, pages 443–450. Springer, 2016.

[90] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Deep transfer learning with joint adaptation networks. *arXiv preprint arXiv:1605.06636*, 2016.

[91] Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. Simultaneous deep transfer across domains and tasks. In *Computer Vision (ICCV), 2015 IEEE International Conference on*, pages 4068–4076. IEEE, 2015.

[92] Steven J Nowlan and Geoffrey E Hinton. Simplifying neural networks by soft weight-sharing. *Neural computation*, 4(4):473–493, 1992.

[93] Chester Ornes and Jack Sklansky. A neural network that explains as well as predicts financial market behavior. In *Computational Intelligence for Financial Engineering (CIFEr), 1997., Proceedings of the IEEE/IAFE 1997*, pages 43–49. IEEE, 1997.

[94] Henry A Rowley, Shumeet Baluja, and Takeo Kanade. Neural network-based face detection. *IEEE Transactions on pattern analysis and machine intelligence*, 20(1):23–38, 1998.

[95] Xingyi Zhou, Qingfu Wan, Wei Zhang, Xiangyang Xue, and Yichen Wei. Model-based deep hand pose estimation. *arXiv preprint arXiv:1606.06854*, 2016.

[96] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[97] Nitish Srivastava and Ruslan R Salakhutdinov. Multimodal learning with deep boltzmann machines. In *Advances in neural information processing systems*, pages 2222–2230, 2012.

[98] Mark J Huiskes and Michael S Lew. The mir flickr retrieval evaluation. In *Proceedings of the 1st ACM international conference on Multimedia information retrieval*, pages 39–43. ACM, 2008.

[99] Mark J Huiskes, Bart Thomee, and Michael S Lew. New trends and ideas in visual concept detection: the mir flickr retrieval evaluation initiative. In *Proceedings of the international conference on Multimedia information retrieval*, pages 527–536. ACM, 2010.

[100] Harry McGurk and John MacDonald. Hearing lips and seeing voices. *Nature*, 264(5588):746, 1976.

[101] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. Multimodal deep learning. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 689–696, 2011.

[102] Ryan Kiros, Ruslan Salakhutdinov, and Richard S Zemel. Unifying visual-semantic embeddings with multimodal neural language models. *arXiv preprint arXiv:1411.2539*, 2014.

[103] Di Wu and Ling Shao. Multimodal dynamic networks for gesture recognition. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 945–948. ACM, 2014.

[104] Edgar A Bernal, Xitong Yang, Qun Li, Jayant Kumar, Sriganesh Madhvanath, Palghat Ramesh, and Raja Bala. Deep temporal multimodal fusion for medical procedure monitoring using wearable sensors. *IEEE Transactions on Multimedia*, 20(1):107–118, 2018.

[105] Terry C Lansdown, Nicola Brook-Carter, and Tanita Kersloot. Distraction from multiple in-vehicle secondary tasks: vehicle performance and mental workload implications. *Ergonomics*, 47(1):91–104, 2004.

[106] Paul Green. Visual and task demands of driver information systems. Technical report, 1999.

[107] Mads Gregers Jæger, Mikael B Skov, Nils Gram Thomassen, et al. You can touch, but you can't look: interacting with in-vehicle systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1139–1148. ACM, 2008.

[108] William J Horrey. Assessing the effects of in-vehicle tasks on driving performance. *Ergonomics in Design: The Quarterly of Human Factors Applications*, 19(4):4–7, 2011.

[109] William T Freeman and Michal Roth. Orientation histograms for hand gesture recognition. In *International workshop on automatic face and gesture recognition*, volume 12, pages 296–301, 1995.

[110] Yun Liu, Zhijie Gan, and Yu Sun. Static hand gesture recognition and its application based on support vector machines. In *Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2008. SNPD '08. Ninth ACIS International Conference on*, pages 517–521, Aug 2008. doi: 10.1109/SNPD.2008.144.

[111] James Davis and Mubarak Shah. Recognizing hand gestures. In *European Conference on Computer Vision*, pages 331–340. Springer, 1994.

[112] Jianying Hu, Michael K Brown, and William Turin. Hmm based online handwriting recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 18(10):1039–1045, 1996.

[113] Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[114] Eshed Ohn-Bar and Mohan Manubhai Trivedi. Hand gesture recognition in real time for automotive interfaces: A multimodal vision-based approach and evaluations. *IEEE Transactions on Intelligent Transportation Systems*, 15(6):2368–2377, 2014.

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

[116] Markus Oberweger, Paul Wohlhart, and Vincent Lepetit. Hands deep in deep learning for hand pose estimation. *arXiv preprint arXiv:1502.06807*, 2015.

[117] Jonathan J Tompson, Arjun Jain, Yann LeCun, and Christoph Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. In *Advances in neural information processing systems*, pages 1799–1807, 2014.

[118] Jiachen Yang and Ryota Horie. An improved computer interface comprising a recurrent neural network and a natural user interface. *Procedia Computer Science*, 60:1386–1395, 2015.

[119] Jonathan Tompson, Murphy Stein, Yann Lecun, and Ken Perlin. Real-time continuous pose recovery of human hands using convolutional networks. *ACM Transactions on Graphics (ToG)*, 33(5):169, 2014.

[120] Danhang Tang, Hyung Jin Chang, Alykhan Tejani, and Tae-Kyun Kim. Latent regression forest: Structured estimation of 3d hand poses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(7):1374–1387, 2017.

[121] Hugh Leather, Edwin Bonilla, and Michael O'Boyle. Automatic feature generation for machine learning based optimizing compilation. In *Code Generation and Optimization, 2009. CGO 2009. International Symposium on*, pages 81–91. IEEE, 2009.

[122] Gerald Tesauro. Td-gammon: A self-teaching backgammon program. In *Applications of Neural Networks*, pages 267–285. Springer, 1995.

[123] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[124] Peter Földiák. Learning invariance from transformation sequences. *Neural Computation*, 3(2):194–200, 1991.

[125] Laurenz Wiskott and Terrence Sejnowski. Slow feature analysis: Unsupervised learning of invariances. *Neural computation*, 14(4):715–770, 2002.

[126] Laurenz Wiskott. Slow feature analysis: A theoretical analysis of optimal free responses. *Neural Computation*, 15(9):2147–2177, 2003.

[127] Guoying Li and Jian Zhang. Sphering and its properties. *Sankhyā: The Indian Journal of Statistics, Series A*, pages 119–133, 1998.

[128] Zhang Zhang and Dacheng Tao. Slow feature analysis for human action recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(3):436–450, 2012.

[129] Yann LeCun and Corinna Cortes. The mnist database of handwritten digits, 1998, 2012.

[130] Zhou Ren, Junsong Yuan, and Zhengyou Zhang. Robust hand gesture recognition based on finger-earth mover's distance with a commodity depth camera. In *Proceedings of the 19th ACM International Conference on Multimedia*, MM '11, pages 1093–1096, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0616-4. doi: 10.1145/2072298.2071946. URL `http://doi.acm.org/10.1145/2072298.2071946`.

[131] Giorgio Panin, Sebastian Klose, and Alois Knoll. Real-time articulated hand detection and pose estimation. In *Advances in Visual Computing*, pages 1131–1140. Springer, 2009.

[132] J-F Collumeau, Rémy Leconge, Bruno Emile, and Hélène Laurent. Hand gesture recognition using a dedicated geometric descriptor. In *Image Processing Theory, Tools and Applications (IPTA), 2012 3rd International Conference on*, pages 287–292. IEEE, 2012.

[133] Chan-Su Lee, Sung Yong Chun, and Shin Won Park. Articulated hand configuration and rotation estimation using extended torus manifold embedding. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 441–444. IEEE, 2012.

[134] Houtao Deng and George Runger. Gene selection with guided regularized random forest. *Pattern Recognition*, 46(12):3483–3489, 2013.

[135] Thanh-Tung Nguyen, Joshua Zhexue Huang, and Thuy Thi Nguyen. Unbiased feature selection in learning random forests for high-dimensional data. *The Scientific World Journal*, 2015, 2015.

[136] Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pages 396–404, 1990.

[137] Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics bulletin*, 1(6):80–83, 1945.

[138] Rajat Raina, Andrew Y Ng, and Daphne Koller. Constructing informative priors using transfer learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 713–720. ACM, 2006.

[139] Anna Choromanska, Mikael Henaff, Michaël Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surface of multilayer networks. *CoRR*, abs/1412.0233, 2014. URL `http://arxiv.org/abs/1412.0233`.

[140] David JC MacKay. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.

[141] David JC MacKay. The evidence framework applied to classification networks. *Neural computation*, 4(5):720–736, 1992.

[142] Stanley F Chen and Ronald Rosenfeld. A survey of smoothing techniques for me models. *IEEE transactions on Speech and Audio Processing*, 8(1): 37–50, 2000.

[143] Ciprian Chelba and Alex Acero. Adaptation of maximum entropy capitalizer: Little data can help a lot. *Computer Speech & Language*, 20(4):382–399, 2006.

[144] Ming-Yu Liu and Oncel Tuzel. Coupled generative adversarial networks. In *Advances in neural information processing systems*, pages 469–477, 2016.

[145] Judy Hoffman, Dequan Wang, Fisher Yu, and Trevor Darrell. Fcns in the wild: Pixel-level adversarial and constraint-based adaptation. *arXiv preprint arXiv:1612.02649*, 2016.

[146] Rahaf Aljundi, Rémi Emonet, Damien Muselet, and Marc Sebban. Landmarks-based kernelized subspace alignment for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 56–63, 2015.

[147] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[148] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[149] R Weir, M Mitchell, S Clark, G Puchhammer, M Haslinger, R Grausenburger, N Kumar, R Hofbauer, P Kushnigg, V Cornelius, et al. The intrinsic hand–a 22 degree-of-freedom artificial hand-wrist replacement. Myoelectric Symposium, 2008.

[150] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[151] Ian J Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, and Vinay Shet. Multi-digit number recognition from street view imagery using deep convolutional neural networks. *arXiv preprint arXiv:1312.6082*, 2013.

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

[153] Hengkai Guo, Guijin Wang, and Xinghao Chen. Two-stream convolutional neural network for accurate rgb-d fingertip detection using depth and edge information. *arXiv preprint arXiv:1612.07978*, 2016.

[154] Martin Riedmiller and Heinrich Braun. A direct adaptive method for faster backpropagation learning: The rprop algorithm. In *Neural Networks, 1993., IEEE International Conference on*, pages 586–591. IEEE, 1993.

[155] Yoav Freund, Robert Schapire, and Naoki Abe. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780): 1612, 1999.

[156] Louisa Lam and SY Suen. Application of majority voting to pattern recognition: an analysis of its behavior and performance. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 27(5):553–568, 1997.

[157] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[158] C. Chen, R. Jafari, and N. Kehtarnavaz. Utd-mhad: A multimodal dataset for human action recognition utilizing a depth camera and a wearable inertial sensor. In *2015 IEEE International Conference on Image Processing (ICIP)*, pages 168–172, Sept 2015. doi: 10.1109/ICIP.2015.7350781.

[159] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2938–2946, 2015.

[160] Alain Pagani, Johannes Koehler, and Didier Stricker. Circular markers for camera pose estimation. In *WIAMIS 2011: 12th International Workshop on Image Analysis for Multimedia Interactive Services, Delft, The Netherlands, April 13-15, 2011*. TU Delft; EWI; MM; PRB, 2011.

[161] Paul Green. The 15-second rule for driver information systems. In *Proceedings of the ITS America Ninth Annual Meeting*. Intelligent Transportation Society of America Washington, DC, 1999.

[162] Stefan Mattes. The lane-change-task as a tool for driver distraction evaluation. *Quality of Work and Products in Enterprises of the Future*, 57, 2003.

[163] Tanja Döring, Dagmar Kern, Paul Marshall, Max Pfeiffer, Johannes Schöning, Volker Gruhn, and Albrecht Schmidt. Gestural interaction on the steering wheel: Reducing the visual demand. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 483–492, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0228-9. doi: 10.1145/1978942.1979010. URL `http://doi.acm.org/10.1145/1978942.1979010`.

[164] Micah Alpern and Katie Minardo. Developing a car gesture interface for use as a secondary task. In *CHI '03 Extended Abstracts on Human Factors*

in Computing Systems, CHI EA '03, pages 932–933, New York, NY, USA, 2003. ACM. ISBN 1-58113-637-4. doi: 10.1145/765891.766078. URL http://doi.acm.org/10.1145/765891.766078.

[165] Dinesh Jayaraman and Kristen Grauman. Slow and steady feature analysis: higher order temporal coherence in video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3852–3861, 2016.

Aditya Tewari

Enschede, Netherlands

Education

2014- 2019	**TU Kaiserslautern** Germany	PhD	Hand Pose and Gesture by Feature learning (Augmented Vision Group)	
2012-2013	**University of Southampton** United Kingdom	*MSc*	Systems and Signal Processing	(First)
2006-2010	**GB Pant University, Pantnagar** India	*BTech*	Electronics and Communication Engineering	(First)

Assignments

Xsens, Enschede Netherlands,
March, 2019 – PRESENT
Machine Learning Scientist.

- Using Neural Networks for missing data estimation and sparse sensor based 3D body tracking algorithms. Contributions to optimisation procedures in multi-sensor setup.

IEE SA, Luxembourg,
May, 2018 – February, 2019
Computer Vision Scientist.

- Developing machine learning components of the in-car ADAS and passenger safety systems based on vision sensors.

IEE SA, Luxembourg,
2014-2018
Research Trainee.

- Developing machine learning algorithms for Time of Flight Camera based hand gesture and hand shape classification which can be integrated to an existing in-vehicle system.

DFKI Kaiserlautern, Germany,
Augmented Vision Group
November, 2013 – September,2018

- Internship on hand pose classification using earth movers distance.
- PhD project on prior Knowledge introduction to Neural Network learning.

University of Southampton, United Kingdom,
Controls and Signal Processing Group
January, 2013 – September,2014

- Developing a 2-D Parabolic Radon Transform to Detect Linear patterns and angular variation in Lung images.