BEIIMNET: SEMI-SUPERVISED CONTEXTUALLY GUIDED CONVOLUTIONAL NEURAL NETWORKS

by

Beiimbet Sarsekeyev

A thesis presented to the Department of Computer Science and the Graduate School of the University of Central Arkansas in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Science

Conway, Arkansas
May 2021

TO THE OFFICE OF GRADUATE STUDIES:

The members of the Committee approve the thesis of
Beiimbet Sarsekeyev presented on
04/09/2021

OLCAY KURSUN Digitally signed by OLCAY KURSUN
Date: 2021.04.16 03:24:50 -05'00'

Committee Chairperson

Sinan Kockara Digitally signed by Sinan Kockara
Date: 2021.04.16 08:22:07 -05'00'

Committee Member

Yu Sun Digitally signed by Yu Sun
DN: cn=Yu Sun, o=University of Central Arkansas, ou=Computer Science Dept., email=yusun@uca.edu, c=US
Date: 2021.04.16 09:00:22 -05'00'

Committee Mem

PERMISSSION

Title BeiimNet: Semi-Supervised Contextually Guided Convolutional Neural Networks

Department Computer Science

Degree Master of Science

In presenting this thesis/dissertation in partial fulfillment of the requirements for a graduate degree from the University of Central Arkansas, I agree that the Library of this University shall make it freely available for inspections. I further agree that permission for extensive copying for scholarly purposes may be granted by the professor who supervised my thesis/dissertation work, or, in the professor's absence, by the Chair of the Department or the Dean of the Graduate School. It is understood that due recognition shall be given to me and to the University of Central Arkansas in any scholarly use which may be made of any material in my thesis/dissertation.

Beiimbet Sarsekeyev

04/22/2021

ACKNOWLEDGEMENT

I thank my thesis advisor Dr. Olcay Kursun for his guidance throughout the project. He dedicated a significant amount of time to mentoring, teaching deep learning, and helping with my research. I want to thank Dr. Kursun for involving me in the NSF-supported DART project and sharing his CG-CNN figures and codes to get the thesis off to a good start.

I would like to express my gratitude to committee members Dr. Sinan Kockara and Dr. Yu Sun for their time and insights. Also, special thanks to Dr. Kockara for involving me in his project that first introduced me to deep learning. I would like to thank Dr. Ahmad Patooghy (cofounder of Intelligent Embedded Systems Lab) for providing me with the photos of the textures used in their vibrotactile signals texture classification dataset.

Finally, I want to express my deepest gratitude to my family and my wife, who have fully supported me during this research and encouraged me throughout my years of study.

This thesis is supported in part by the National Science Foundation under Award No. OIA-1946391 and in part by DoD CDMRP under Award No. W81XWH-17-2-0046. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

ABSTRACT

Deep Learning has been considered as one of the most significant milestones for Artificial Intelligence. Deep convolutional neural networks (CNNs) have been proven to be very successful in many pattern recognition tasks. Used as a pre-trained base network trained on powerful computers on large datasets, CNNs offer remarkable transfer learning capabilities. The CNN features learned in a local-to-global pyramidal architecture extracts gradually more sophisticated features in the higher layers based on, the lower ones' features. The hidden layers' connection weights provide broad-purpose (pluripotent) features that can be transferred to other networks for new target tasks, such as recognizing new object classes with possibly smaller datasets. Supervised deep CNNs are trained top-down and minimizing the classification error on large manually labelled datasets with thousands of classes, thus achieving their learning of pluripotent features. Recently proposed Contextually Guided Convolutional Neural Network (CG-CNN) architecture learns to extract such pluripotent features of a single convolutional layer in an unsupervised setting. Although CG-CNN has an advantage over deep CNNs, it can tune to these pluripotent features bottom-up without requiring massive, labelled datasets; its semi-supervised and multi-layered extensions require further research. This thesis proposed an extension named BeiimNet and demonstrated its effectiveness in applying the CG-CNN principles to semi-supervised complex pattern recognition tasks.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1 INTRODUCTION

With the recent technological advances in computing and data systems, Artificial Intelligence (AI), more specifically Deep Learning (DL), has gained substantial popularity in research and utilization. DL is efficient because it does not rely on manual feature engineering; it is based on data engineering that means the features develop to perform the complex machine learning tasks thanks to the availability of large datasets. In recent years, DL methods have influenced every central area of life, such as biometrics, health care, finance, transportation systems, and social media. Deep Convolutional Neural Networks (CNNs) are among the most promising deep learning models. CNNs learned their features (convolutional weights) and how to map the raw input to output predictions (e.g., class labels). Once they are learned, CNN features are readily transferable to new tasks where labelled data could be scarce. CNNs make excellent use of several advances in a neural network design such as deep multi-layered pyramidal architectures, weight sharing, and transfer learning to learn these powerful (pluripotent as Kursun, Dinc, and Favorov (2021) call them) features. In a nutshell, multi-layered architectures help CNNs learn their complex input-output mapping. Weight-sharing (using convolutions) helps to reduce the number of parameters needed to learn in the training process, and transfer learning allows CNNs to be re-trained (transferring some features previously learned on another more complex/general task) for few-shot learning (Malik et al. 2020; Wang et al. 2020) or on tasks where labeled data is scarce (Zhang, Zhu, and Fu 2019).

As acquiring manual labels for data can be challenging in real-world due to the volume/flux of data or due to the lack of resources and experts in many real-world applications, transfer learning makes CNNs (and deep learning in general) very appealing (Baur, Albarqouni, and Navab 2017; Zheng et al. 2020). For instance, Convolutional

Neural Networks that utilized transfer learning consistently performed health disease diagnoses from images nearly identically to health experts (Gao et al. 2020). Although the image domain is the primary field of application for CNNs, they have recently proven to be very successful for predictive analysis of signal data.

Loosely inspired by the neural architecture of the brain's cerebral cortex (Zorins and Grabusts 2015), CNN progressively extracts higher and higher-level features in a pyramidal architecture. CNNs are typically trained using large, manually labelled datasets with a large number of classes to discriminate. Once these features are developed, as they can distinguish many classes, they can be transferred and used in other domain-related tasks, such as other image classification tasks with classes that are not used in their training. This feature transfer (transfer learning) process generally uses/transfers features at the higher-layers as the higher-layer features have larger receptive fields (they see a larger field of pixels), and they are more descriptive (more nonlinear). However, in the success stories of CNNs, besides the quantitative usefulness of higher-layers' features in transfer learning, the qualitative evaluation of the early-layer features is also discussed (Yosinski et al. 2014). Early-layer features are praised for being realistic features due to their resemblance to the features tuned by the primary visual cortex (V1) (Bengio 2012; Yosinski et al. 2014; Goodfellow, Bengio, and Courville 2016; Kursun and Favorov 2019; Shrestha and Mahmood 2019; Kursun, Dinc, and Favorov 2021). These features resemble Gabor-like edge filters, gratings, and color blobs (Kursun, Dinc, and Favorov 2021). Discovering such low-level features is expected to the degree that obtaining anything else with a deep learning algorithm generally implies poorly chosen hyperparameters or a software bug (Yosinski et al. 2014). While deep CNNs generally rely on error backpropagation (top-

down learning back-propagating the error from the top classifier layer to the bottom/early layers) to learn what features should be extracted by their neurons, the emergence of V1-like features in the early layers is an exciting phenomenon. CG-CNN (Contextually Guided Convolutional Neural Networks) was proposed (Kursun, Dinc, and Favorov 2021) to demonstrate the existence of a principled approach to learning such descriptive/pluripotent features in a bottom-up fashion without relying on supervised training (i.e., without a deep network backpropagating misclassification-related error). Unlike deep CNNs, CG-CNNs (and cortical areas that inspired CG-CNN) do not rely on supervised backpropagation. Instead, they rely on some local contextual information for feature tuning (Becker and Hinton 1992; Phillips and Singer 1997; Körding and König 2000; Favorov and Ryder 2004; Hawkins and Blakeslee 2004; Kursun, Alpaydin, and Favorov 2011; Hawkins, Ahmad, and Cui 2017; Kursun and Favorov 2019).

CG-CNN addresses some of the essential shortcomings of deep learning algorithms and architectures. For instance, their reliance on large and manually labelled datasets requires optimizing their massive number of parameters, mode collapse, and vanishing gradients due to their deep multi-layered design (Abiodun et al. 2019). Using its bottom-up learning of gradually more complex/pluripotent features via a shallow CNN-based architecture, CG-CNN can be trained even in small unlabelled datasets. Taking advantage of abundantly available unlabelled data and transfer learning techniques, improved CG-CNN methods have the potential to alleviate many of these shortcomings.

Semi-supervised learning (using both labelled and unlabelled data) is a common and realistic real-world scenario. Insufficiency of labelled data can originate from the labelling process's time-consuming nature and reliance on experts to generate

labels/ground truths. Semi-supervised Deep networks learn and optimize their learned features, based on which they make their predictions, and these predictions tend to be more robust (Enguehard, O'Halloran, and Gholipour 2019; Zhang, Zhu, and Fu 2019).

The recent technological advances make semi-supervised deep learning more feasible because the acquisition/collection of unlabelled data has become more affordable. The abundant availability of unlabelled data on the internet through social media, the growing demand for automatic recommendation systems (Hoffer and Ailon 2018; Enguehard, O'Halloran, and Gholipour 2019), the development of Edge systems that bring together data collection with various sensors (e.g., camera, microphones, vibrotactile sensors) and deep learning on the same hardware platform, and the development of novel systems that do not initially have sufficient domain expertise or human resources for manual and consistent labeling of data are some of the examples of semi-supervised learning feasibility.

In this thesis, a vibrotactile signals texture classification dataset is analysed using CNN, Autoencoder, CG-CNN, and the proposed BeiimNet architectures in addition to the standard benchmark machine learning methods such as random forests, nearest neighbors, and support vector machines. The reasons for selecting these algorithms were the dataset's promising nature to demonstrate the CG-CNN approach's power. While the experimental vibrotactile dataset is simpler than an image classification dataset, in the absence of sufficient volume of labelled data and pre-trained networks to use via transfer learning, deep learning algorithms such as CNNs do not perform at their full potential (Kursun and Favorov 2019; Kursun and Patooghy 2020). Tactile and vibration sensing systems have potential in various applications in robotics and neuroscience (Schopfer, Ritter, and

Heidemann 2007; Gwilliam et al. 2010; Kursun and Patooghy 2020; Zhou et al. 2020). While humans can effortlessly perform tactile sensing tasks (e.g., texture, hardness, roughness), DL systems need improvement for achieving such levels of success in tactile information processing. Moreover, the application of the semi-supervised extension of the CG-CNN algorithm is simple on this dataset, which does require a very deep CNN architecture for achieving the computational complexity sufficient for mapping the raw input of sensor readings to the class labels of the observed textures. This thesis presents the first application of the CNNs and contextually guided neural networks (CG-CNN and BeiimNet) on the vibrotactile signals texture classification dataset collected by (Kursun and Patooghy 2020).

This thesis's structure is as follows: Chapter 2 reviews both foundational and cutting-edge materials on which this thesis is based, most notably the CG-CNN (Contextually Guided Convolutional Neural Network) method. Chapter 3 presents the proposed BeiimNet method as the semi-supervised multi-layered extension of CG-CNN. Chapter 4 presents the experimental results on the vibrotactile signals texture classification dataset with transferrable feature extraction methods used for texture classification and putting these features in use to classify new classes of textures. Conclusions and future work are discussed in Chapter 5.

CHAPTER 2 RELATED WORK

For the last several decades, the potential of Machine Learning applicability to almost any field of our lives has been widely acknowledged and increasingly utilized (LeCun, Bengio, and Hinton 2015; Alom et al. 2018; Abiodun et al. 2019; Gao et al. 2020; Sanodiya and Yao 2020; Zhou et al. 2020). Machine learning enables incorporating data into the analysis in various scientific areas and creates opportunities for solving complex problems. Machine learning consists of several types of learning paradigms. However, supervised, unsupervised, semi-supervised, and reinforcement learning are the most common among those paradigms. Supervised learning permits and takes full advantage of labelled examples or targets, where training algorithms use feedback from targets and optimizes the model parameters accordingly. Unsupervised learning focuses on identifying patterns in data without relying on labels; clustering algorithms and unsupervised dimensionality reduction algorithms (Alpaydin 2014) are good examples of this type of learning. Semi-supervised learning is advantageous when unlabelled data are abundant. Reinforcement learning is quite different in its approach to learning compared to previous learning types. RL uses agents placed in a particular environment with specific criteria for receiving a reward when they solve problems, reinforcing particular actions, and learning from those experiences (Marsland 2014).

Deep Learning has been considered as the new AI (Kursun and Favorov 2019). Among deep learning algorithms, deep convolutional neural networks (CNNs) have been particularly successful in many computer vision and pattern recognition tasks and their remarkable transfer learning capabilities. In this Chapter 2, the subsections are organized to review these topics in the following order. First, the essential components of artificial neural networks are summarized in Section 2.1; then, in Section 2.2, convolutional neural

networks are discussed. In Section 2.3, transfer learning is reviewed to explain the usefulness of learned features. Section 2.4 summarizes deep representation learning methods, and Section 2.5 focuses on generative adversarial networks. Finally, Section 2.6 introduces Contextually Guided Convolutional Neural Networks, which is the foundation of the proposed method, BeiimNet.

The most notable difference between traditional ML algorithms and DL is their feature extraction approach. While traditional ML algorithms hand-crafted the features by using feature extraction algorithms such as Scale Invariant Feature Transform (SIFT), Speeded Up Robust Features (SURF), Histogram Oriented Gradient (HOG), and Local Binary Pattern (LBP), Deep Learning features are trained automatically with hierarchical representation, where powerful, low-level, features help to optimize each successive layer to fit a specific problem (O'Mahony et al. 2020). This hierarchical approach enables DL algorithms to build remarkable features that, unlike traditional ML algorithms, have unlimited potential for improvement.

The first introduction to deep learning started with LeNet in 1998, where LeCun incorporated backpropagation with a convolutional neural network (Alom et al. 2018). The computational restrictions of that time made the application of LeNet quite challenging and impractical. Still, after a little more than a decade, modern computers' computational possibilities revitalized the deep learning field. While LeNet is the architecture that started deep learning, AlexNet is perhaps the most prominent architecture by which other models evaluate their architecture's validity. It was introduced in 2012, with a win in a complex ImageNet challenge for visual object recognition called the ImageNet Large Scale Visual

Recognition Challenge (ILSVRC) (Alom et al. 2018). It is considered a breakthrough in AI, and it brought a big wave of interest in deep learning research and applications.

The features that are represented under the architecture are the essence of AlexNet's importance. Obtaining the first convolutional features is used to make sure that the network learns well. Getting anything else but Figure 1 is considered as a sign of a bug in the program or a poor setting for some hyperparameter (Yosinski et al. 2014).

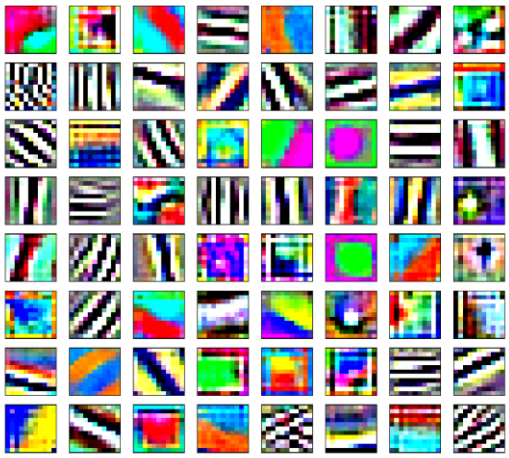

Figure 1. Representation of AlexNet's first convolutional layer features

The AlexNet's first layer features (Figure 1) have been a benchmark for low-level features. With more complex and sophisticated architectures, architectures such as GoogleNet (Figure 2) and ResNet101 (Figure 3).

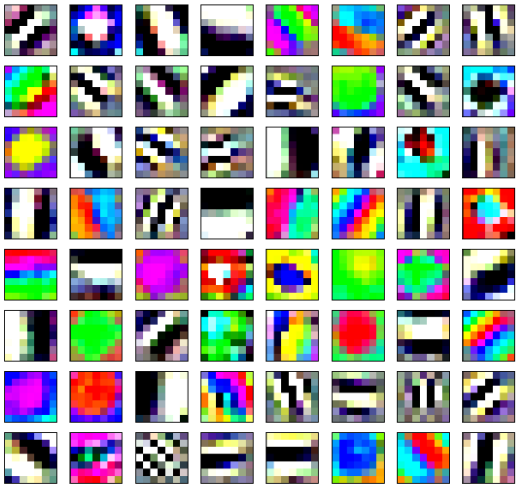

Figure 2. Representation of GoogleNet's first convolutional layer features

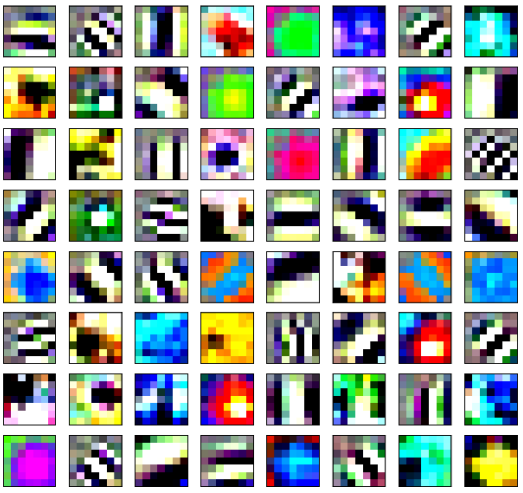

Figure 3. Representation of ResNet101's first convolutional layer features

Figure 4 referred to AlexNet architecture (LeNail 2019) and visual representation of developed features (Zeiler and Fergus 2014) from ImageNet dataset (Deng et al. 2009).

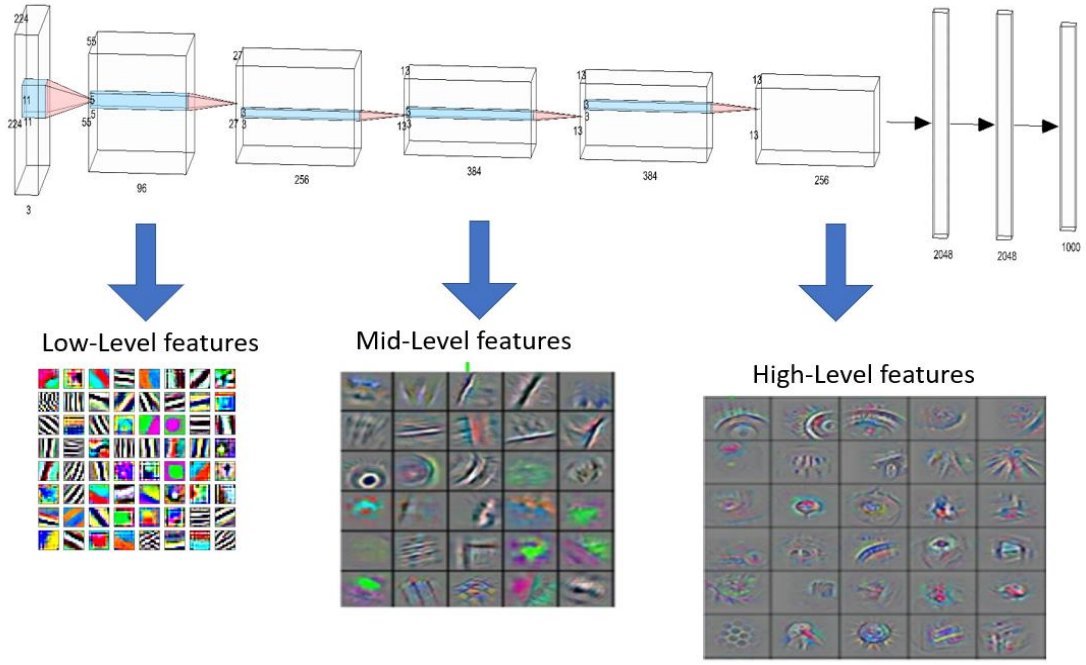

Figure 4. The representation of AlexNet Architecture with ImageNet features.

Deep Learning (DL) has become a very successful type of artificial neural network studied under Artificial Intelligence (AI) and has found practical uses in many fields such as robotics, automation, medical, and finance. The general idea is that with a deeper network, a task can achieve better accuracy. However, this can be costly in terms of computation and time. The majority of ML algorithms cannot improve after a certain point in training, but deep neural networks have the potential to improve, which can be one of the reasons for their popularity. With every increase in layers, DNN can learn from general features to more specific attributes.

2.1 Essential Components of Artificial Neural Networks

ANNs are composed of interconnected neurons organized into layers. Neurons are loosely inspired by biological neurons (Zorins and Grabusts 2015). Data is the most crucial component of any machine learning algorithm, and while data is abundant, generating high-quality datasets is quite challenging. The next component is a scoring function. It allocates what the algorithm learns to class labels.

The loss function is a component of a neural network that quantitatively assesses the learning algorithm's performance regarding training labels or ground truths. Generally, a lower loss is wanted while there is no overfitting which is an inability to generalize appropriately. Also, the enormous loss indicates the need to optimize further the algorithm's parameters (Alpaydin 2014; Marsland 2014).

Optimization methods are among the most critical components of neural networks. Optimization methods are the driving force in neural network architecture to learn their weights (which patterns to detect) from their training data. The most popular optimization methods in deep learning are Stochastic Gradient Descent (SGD) and Adam.

Neural networks use activation functions to transform their inputs nonlinearly. The most popular activation function in deep learning is the Rectified Linear Unit (ReLU) and its variations; it is also known as the ramp function. It gives zero for negative inputs and increases linearity for positive inputs while being very efficient computationally (Hayou, Doucet, and Rousseau 2019).

Backpropagation is perhaps the most significant breakthrough for neural networks. After the forward pass of the activations through the network to the last layer of the scoring function, the backpropagation algorithm passes the error backward for the computation of the gradient of the loss function to update the weights (Alpaydin 2014; Marsland 2014).

2.2 Convolutional Neural Networks

Convolutions organize the interconnections among neurons into local groups. A fully connected layer, as the name suggests, connects every input neuron to the output neuron. Generally, convolutional neural networks (CNNs) utilize fully connected layers at the end of the architecture. Deep CNNs apply various sets of considerable numbers of filters at every convolutional layer to feed the following layers. CNN layers start learning with general edges, then with the next layer, they apply filters to detect shapes (Goodfellow, Bengio, and Courville 2016). With every successive convolutional layer, CNN learns to distinguish and learn more specific features to the problem. The network uses these high-level features to make predictions. Convolutional layers usually consist of several CNN building blocks that are part of this section's next part. There are two significant advantages of CNN, local invariance and compositionality (Goodfellow, Bengio, and Courville 2016). For instance, local invariance enables the CNN classification of objects in the image without considering the object's exact location. CNNs can achieve identification of the region with the usage of pooling layers. Compositionality is the idea of creating high-level features from low-level features. This concept enables people to utilize another significant contributor to deep learning's success - transfer learning.

CNN's building blocks are convolutional layer, activation, pooling layer, fully-connected/linear layer, batch normalization, and dropout. At the core of CNN is the convolutional layer, which for obvious reasons, is the most crucial block of CNN. Convolutional layers consist of filters/kernels that are set to specific widths and heights. Convolutional layer shifts or extends these kernels throughout a specified input space and applies convolutions based on specified stride (sliding the kernel window from left to right and top to bottom) and padding (uses zero along the border to avoid mismatch)

(Goodfellow, Bengio, and Courville 2016). These kernels generate activation maps where activation indicates the presence of filter properties such as shapes. Activation such as ReLU is applied after every convolutional layer and generally outputs a reduced, original input size. The next layer is the pooling layer that reduces parameters further and helps to prevent overfitting (Goodfellow, Bengio, and Courville 2016). Pooling kernel can reduce the input by either getting the maximum value or getting an average. While max pooling is applied in the middle of the network, average pooling is part of the last layers and sometimes can substitute the fully connected layer (ex. ResNet). As mentioned above, fully connected layer(s) are the last building in CNN architecture (Goodfellow, Bengio, and Courville 2016). Batch normalization is used to normalize the convolutional layer's activations before feeding it to the next layer. Overall, batch normalization enables an efficient training process. Dropout is the form of regularization that has the primary purpose of reducing the chances of overfitting a network by dropping the connection with inputs from the previous layer at random (Goodfellow, Bengio, and Courville 2016).

2.3 Transfer Learning

Traditional machine learning algorithms train and test data with the same input feature space and distribution (Weiss, Khoshgoftaar, and Wang 2016). Transfer learning is motivated by the need for high-performance learners for domains where data acquisition is problematic, and transferring trained learners from similar domains makes transfer learning feasible and, as a tool, indispensable to current deep learning applications and best practices. There are many reasons for the limited availability of data; for instance, the high cost of generating ground truths due to the lack of domain experts can contribute to the shortage of valuable data (Malik et al. 2020). Therefore, transfer learning applications have

become highly appealing solutions to limited data problems. It is utilized along with various data types such as images, videos, text, and signal data.

The definition of transfer learning comprises two main parts: domain and task ("Transfer Learning" 2021). A domain consists of feature space and a marginal probability distribution with a learning task. Transfer learning is divided into source domain and target domain. Transfer learning aims to improve the target's learning by using the source ("Transfer Learning" 2021). There are three transfer learning types: homogeneous transfer learning, heterogeneous transfer learning, and negative transfer.

Homogeneous transfer learning represents the situation where the input feature space and label space of the target domain are equal to the input feature space and label of the source domain, making it homogeneous. Homogeneous transfer learning tries to correct marginal distribution, conditional distribution, or both (Weiss, Khoshgoftaar, and Wang 2016). Within homogeneous transfer learning, there are several well-established methodologies such as instance-based, feature-based, parameter-based, relation-based, and hybrid-based (instance and parameter) transfer learning. Instance-based transfer learning methods focus on applying weighting approaches on the source domain samples to correct marginal distribution; then retrains the target domain (Weiss, Khoshgoftaar, and Wang 2016; Tan et al. 2018). Feature-based transfer learning can be symmetric and asymmetric. While asymmetric type reweights the features to coordinate closer to the target domain, symmetric focus more on common latent space between the domains (Weiss, Khoshgoftaar, and Wang 2016). Parameter-based transfer learning aims at transferring meaning learning through common parameters between domain learners and creating multiple learners. Relation-based transfer learning uses certain relationships between

domains to transfer knowledge, and this methodology tends to be utilized the least (Weiss, Khoshgoftaar, and Wang 2016).

In heterogeneous transfer learning, opposite to homogeneous transfer learning type, the input feature space of a source domain and the target domain's input feature space are not equal. These heterogeneous transfer learning problems are highly applicable to environments abundant with data and differ in their input features space, often the case with various domains (Weiss, Khoshgoftaar, and Wang 2016). Due to being a relatively new area of research, the solutions to heterogeneous transfer learning are few. The main option of dealing with heterogeneous transfer learning problems is to employ symmetric and asymmetric feature-based methodologies that aim to equate the latent input feature space between the domains (Zhou et al., n.d.). With input feature spaces being equal, previously mentioned categories from homogeneous transfer learning can easily be applied, transforming heterogeneous transfer learning to homogenous. As the name suggests, negative transfer yields worse results with the transfer of inputs from the source domain to the target domain learner. The negative transfer could result from the poor relationship between the source and target domains (Weiss, Khoshgoftaar, and Wang 2016).

2.4 Deep Representation Learning

Representative learning is a set of algorithms and methods that aim to learn and extract representative features/ information. It could be represented with and without feature engineering methods, but representation learning is often associated with deep learning or feature learning, which avoids direct feature engineering. Since 2006, in the initial research works of deep representation learning, the central idea was a greedy layer-

wise unsupervised pre-training. The unsupervised learning approach learns a hierarchy of features one layer simultaneously; also, each layer applies new transformations to learn representative features (Bengio 2012). This idea translates well to autoencoder implementation. Initially, autoencoders were used to represent dimensionality reduction, where they served a function of a bottleneck.

Autoencoder is a type of neural network that learns to represent data in its hidden layers. AE can utilize fully connected and convolutional layers as neural building blocks. However, it is part of unsupervised learning where the network encodes input for meaningful representation and decodes those features to reconstruct the original input (Sanodiya and Yao 2020).

Autoencoders extend the idea of principal component analysis (PCA). While PCA transforms the data to linear representation, AE can produce nonlinear representations (Shrestha and Mahmood 2019). Although AE is part of unsupervised learning, encoder's features can be applied to classification networks. Recent implementations of autoencoders generally consist of two parts encoder and decoder. Encoder extracts representative features from the input, and decoder aims to reconstruct the feature map back to input space and minimize reconstruction error.

2.4.1 Discriminative Unsupervised Feature Learning with Exemplar Convolutional Neural Networks

This paper only utilized unlabelled data. The network discriminates between surrogate classes. Each surrogate class is generated from randomly generated image patches, called a seed. Each seed is generated with the help of various transformations. The significant difference between the surrogate tasks (Dosovitskiy et al. 2014; Kursun, Dinc, and Favorov 2021) and their seeds with classic and typical variations/augmentations

(Shorten and Khoshgoftaar 2019) in data is the descriptive and generic robustness of features. The study is primarily based on the unsupervised learning of invariant features. There are several instances of invariant feature generation/utilization in both unsupervised and supervised learning. In unsupervised learning, linear autoencoders learn invariant features by enforcing a temporal slowness constraint on the feature representation (Dosovitskiy et al. 2014). Still, they fail to utilize multiple CNN layers due to heavy reliance on direct modelling the input distribution. Some supervised learning research on learning invariant features directly penalizes the output's derivative concerning the transformation's magnitude. This research does not regularize the derivative explicitly (Dosovitskiy et al. 2014).

This method achieves vitality/robustness of transformations that is not present in the classic supervised approach. This method starts with creating surrogate training data from unlabelled images, where a random sample of N patches that contain various objects is chosen (Dosovitskiy et al. 2014). After selecting patches, a family of transformations is applied. Based on those patches' seed, labels, surrogate classes are declared that translate to sets of transformed image patches (Dosovitskiy et al. 2014). The CNN network is then trained to discriminate between those surrogate classes. The features generated from this method outperform the classification results of traditional unsupervised feature generation methods (Dosovitskiy et al. 2014). While this method beat unsupervised method competition, it could not do the same with classic supervised learning counterparts.

2.4.2 Learning Deep Representations by Mutual Information Estimation and Maximization

This paper explores maximizing mutual information between output and input of Deep encoders for representation in unsupervised learning (Hjelm et al. 2019). With the

complexity and difficulty of computing the mutual information between input and output, the proposed Deep Infomax method incorporates the input's locality to the objective, emphasizing the structure's importance (Hjelm et al. 2019). By matching prior distributions in an adversarial manner, it controls the representation characteristics. The core idea is to maximize the mutual information between the input and the output (Hjelm et al. 2019). Deep Infomax utilizes an adversarial model/learning with an encoder and a decoder. While this method outperforms many unsupervised learning tasks, it only nears the same results expected from supervised learning (Hjelm et al. 2019).

2.5 Generative Adversarial Networks

According to Yan LeCun, a founding father of convolutional neural networks, "Generative Adversarial Network is the most interesting idea in the last ten years in Machine Learning." (Miller 2019, 87), GAN framework was proposed in 2014 to estimate generative models by incorporating adversarial procedure, where two models are trained concurrently (Goodfellow et al. 2014). The first model of the framework, Generator G, captures the data distribution. The second model, Discriminator D, estimates the probability whether the input source belongs to data or G (Goodfellow et al. 2014). Figure 5 illustrates an example of GANs adversarial principle. As the original CG-CNN, the proposed semi-supervised multi-layered extension, BeiimNet, is based on such a GAN-like optimization, with a more supportive focus rather than an adversarial one.

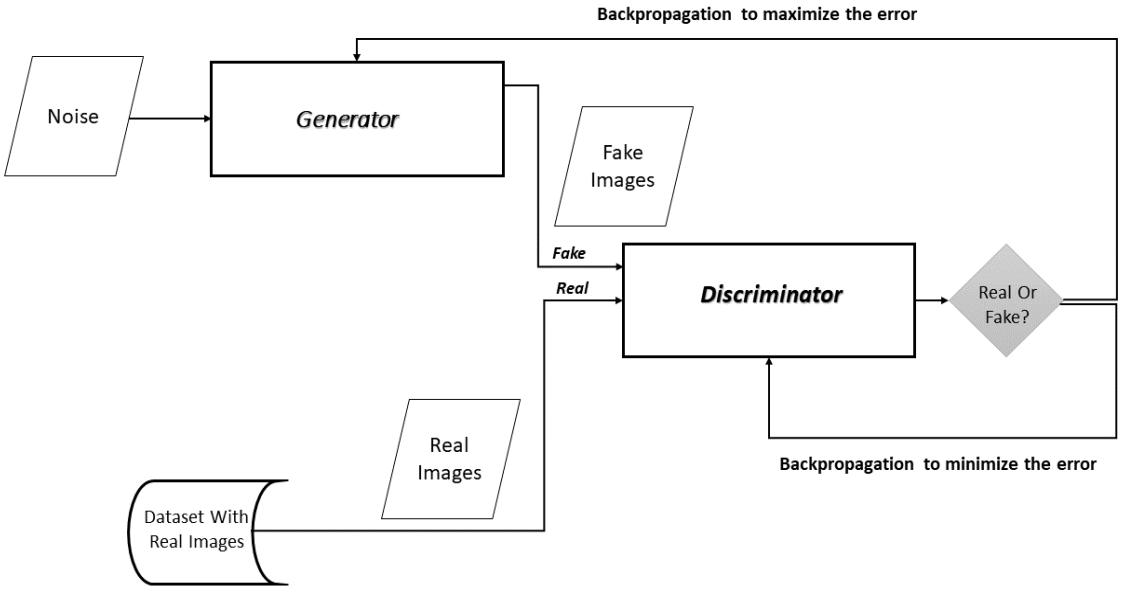

Figure 5. During its iterative training, a GAN network learns to produce increasingly realistic images that Discriminator eventually fails to discriminate the fake ones from the real ones.

2.6 Contextually Guided Convolutional Neural Networks (CG-CNN)

CG-CNN is an unsupervised (self-supervised) method that offers a means for extraction of highly discriminative and transferable features of a single convolutional area. The entire system is composed of a single convolutional layer, *Feature Generator*, connected to a linear classifier. CG-CNN training uses transfer learning to learn what to transfer by creating different classification problems for self-supervision. In other words, Feature Generator gradually learns more discriminative features that Discriminator can adapt to its ever-changing classification problems, which in turn provide feedback to Feature Generator. Although CG-CNN networks can be stacked similar to how deep autoencoders are built, multiple layers within a CG-CNN network, Figure 6, can also be a

more robust feature extractor. These multi-layer extensions of CG-CNN have not been tested so far.

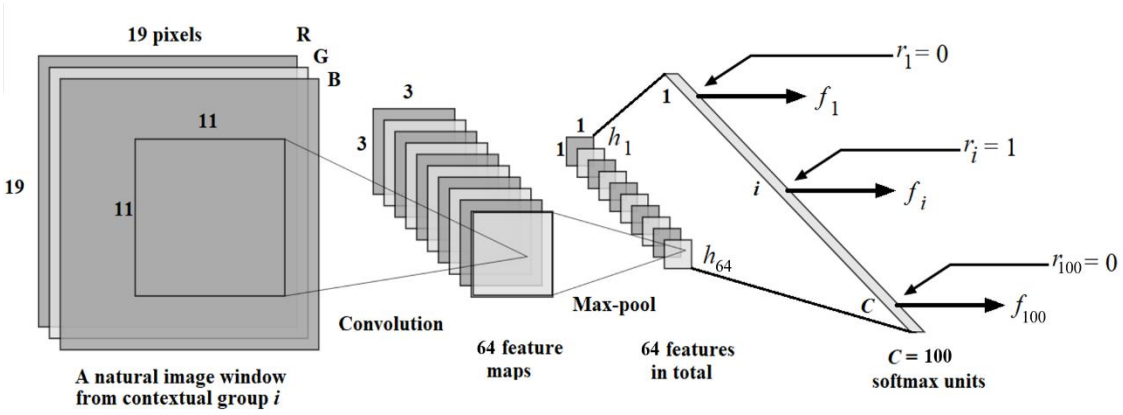

Figure 6. Representative CG-CNN architecture learns 64 general-purpose features (with 11×11convolutions) using C= 100 contextual groups/classes. Once the convolution layer converges, the feature maps can be extracted by applying the convolutions to larger images.

The CG-CNN paper (Kursun, Dinc, and Favorov 2021) used the Caltech dataset (Li Fei-Fei, Fergus, and Perona 2004). CG-CNN used only the face class to strengthen the claim that the class labels are not necessary for learning low-level features of the deep networks. The algorithm works as follows: Upon a presentation of an input pattern (a small image window chosen from one of the internally generated contextual group), the CNN layer computes the feature values that it has learned to extract so far and outputs them to the classifier, which has been trained to distinguish all contextual groups of the current task from each other. The contextual group's input patterns' prediction error is backpropagated to the classifier and the convolutional layer. The backpropagation algorithm adjusts the

connection weights in both the classifier's SoftMax layer and the CNN's convolutional layer alternatively. By minimizing the prediction error of these internally generated and ever-changing classification tasks, CG-CNN features gradually become more inferential than its inputs (Finn, Abbeel, and Levine 2017). The uniqueness of the CG-CNN method is how these classes (contextual groups) are generated internally and how error backpropagation training is carried out.

The system's training is performed over multiple iterations, with each iteration using a different set of contextual groups (training classes) (Finn, Abbeel, and Levine 2017). In each iteration, a new small unsupervised set of training examples (e.g., 50 contextual groups and with each group containing many nearby image windows) is drawn from the database, and the system is trained to discriminate against them. Note that these classes have nothing to do with any external supervision or any supervised class labels (Dosovitskiy et al. 2014; Ghaderi and Athitsos 2016), (see Figure 7 for a demonstration). Once this training is finished, another small set of classes is drawn. Training continues in the next iteration on this new set without resetting the already developed CNN connection weights. The collection of input patterns used in a given training iteration is selected by randomly picking in the database photos C=50 image patches and then applying some transformations (e.g., spatial translations, color conversions, etc.) to these *seed* image patches (Dosovitskiy et al. 2014). All the transformations of a given seed image patch are contextually related and are treated as examples of a single class. This design makes use of the aforementioned neuroscientific principles of pluripotency (Favorov and Kursun 2011; Kursun, Dinc, and Favorov 2021), and contextual guidance principles (Kursun and Favorov 2019; Kursun, Dinc, and Favorov 2021). The Feature Generator seeks pluripotent

features that can be used to discriminate any image patches from each other maximally. At the same time, the Discriminator forces those features, Figure 8, to reflect similarities of contextually related image patches.

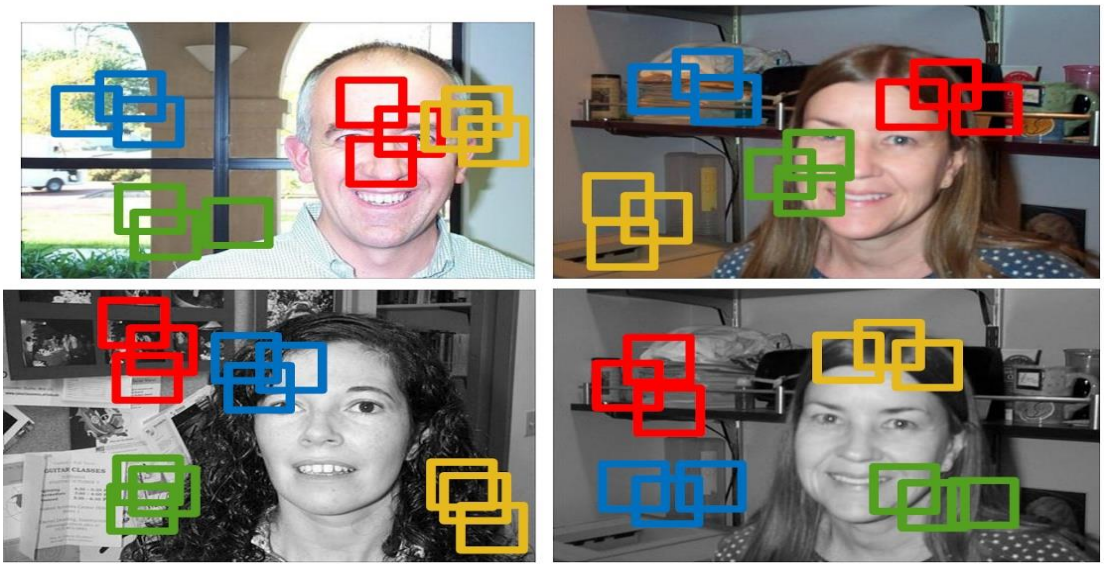

Figure 7. Exemplary images used for snipping small, e.g., 19×19, image patches for training CG-CNN. For this illustration, each task contains one image only, and within that image there are four contextual groups created (each group is shown with a different color and with three snipped image patches).

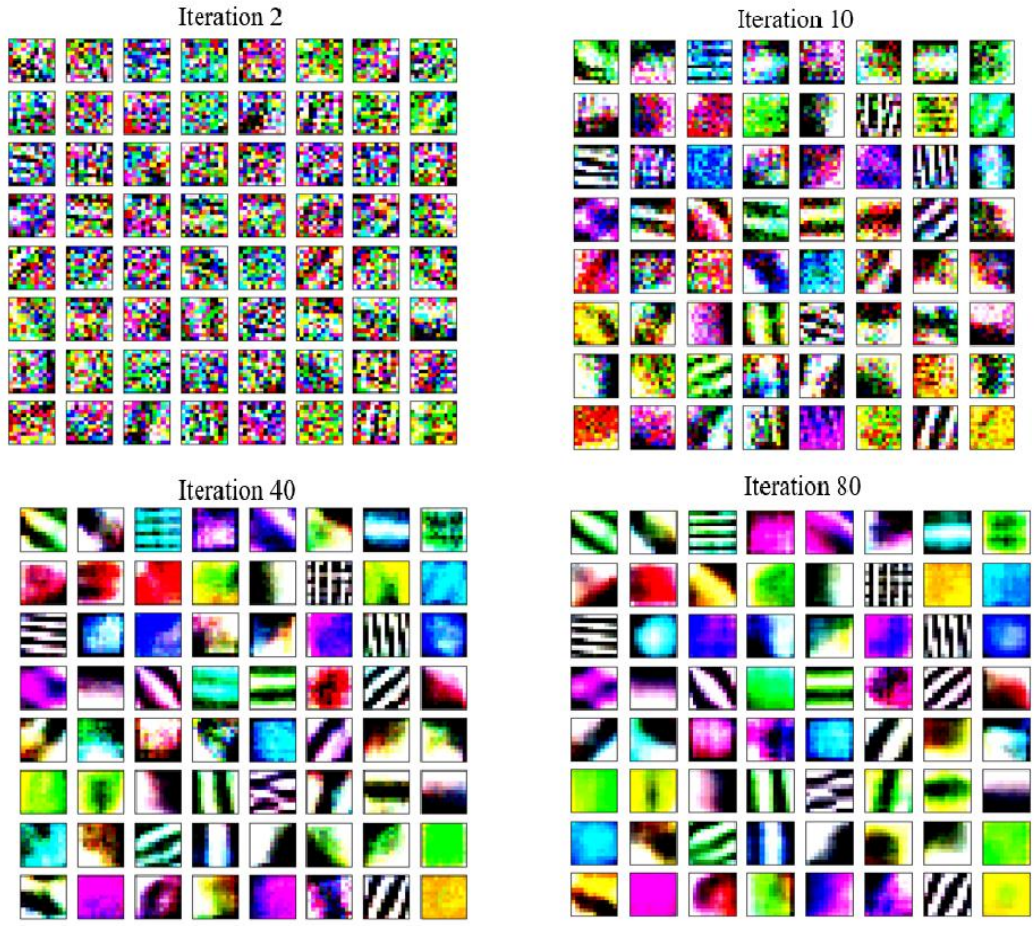

Figure 8. CG-CNN training iterations gradually improve the features of its convolutional layer, making them more and more transferable.

CHAPTER 3 BEIIMNET SEMI-SUPERVISED CG-CNN

BeiimNet extends the unsupervised CG-CNN method (Kursun and Favorov 2019; Kursun, Dinc, and Favorov 2021) to semi-supervised learning by incorporating class-labelled examples (external supervision) into the training. The original CG-CNN method trains a convolutional layer so that the feature set gradually becomes more pluripotent for discriminating any set of contextually related input patterns from any other. For CG-CNN, as initially proposed, the data source contained only unlabelled examples. Especially at the first layer, neurons have such a small receptive field that the supervised class labels cannot be of much help anyway: The network snipped the image patches from the large unlabelled images, and all examples within close proximity in that image (i.e., contextually-related image patches) were given a unique group label that was to be discriminated maximally from other such groups (hence the name pluripotent). A stack of CG-CNN layers (i.e., training the next CG-layer on the previous layer's outputs) can tune to higher-order features with more global (more expansive) receptive fields at the higher CG-layers. As these features get more sophisticated, the feature tuning in higher convolutional layers should gradually benefit from supervised examples to exhibit more utility for supervised classification tasks. This step will help develop better features instead of preserving/transforming the data while maintaining all the contextual regularities. For example, the learned features' may never be interested in contrast or brightness-related frequencies.

Although the CG-CNN was originally proposed using an image dataset, other forms of data that exhibit contextual regularities, such as signals and sensor networks, can also

be used. In this thesis, BeiimNet is used to explore the contextual guidance on a signal dataset. There are several contributions of BeiimNet:

1. Demonstrating the feasibility of the CG-CNN method on a different modality. Although the CG-CNN is described in Section 2.3.4 using an image dataset, (Kursun, Dinc, and Favorov 2021) states that other forms of data exhibit contextual regularities, such as signals and sensor networks can also be used. In this thesis, BeiimNet is used to explore the contextual guidance on a signal dataset.
2. The signal dataset used in this thesis is the tactile perception dataset (Kursun and Patooghy 2020), and this thesis is the first study applying deep learning on this dataset. A better understanding of tactile information processing using deep learning is essential both in robotics and neuroscience fields (Schmitz et al. 2014; Kursun and Favorov 2019; Kursun and Patooghy 2020).
3. Although practical image-domain CNNs are deep, using the tactile dataset that does not require a very deep network is more manageable for this first semi-supervised and multi-layered extension of the CG-CNN idea. BeiimNet makes it clear that CG-CNN provides a general-purpose framework that can be easily extended to semi-supervised learning.

As shown in Figure 9, the BeiimNet algorithm chooses a new small task (either supervised or unsupervised) from the immense pool of data. This time the data source contains both labelled and unlabelled examples. The training tasks used for each Expectation-Maximization (EM) iteration is formed using either a supervised or an unsupervised task. The algorithm uses C classes for the unsupervised case and D classes for the supervised case. That is, in an unsupervised task, for each one of the C classes, a

seed image patch is selected, and a batch of input patches are snipped around it (with some additional data augmentations). As in the original CG-CNN, the Classifier layer of the contextually guided network is trained to discriminate all the C contextual groups from each other using the existing features (this is called the E-step of the EM algorithm). On the other hand, for a supervised task, BeiimNet picks labelled training examples from D classes where D is not necessarily equal to all possible classes. New classes can be added at any time as a form of continual learning (Parisi et al. 2019). Both the unsupervised and supervised cases could use data augmentation methods to improve convergence (Shorten and Khoshgoftaar 2019; Zhao et al. 2019).

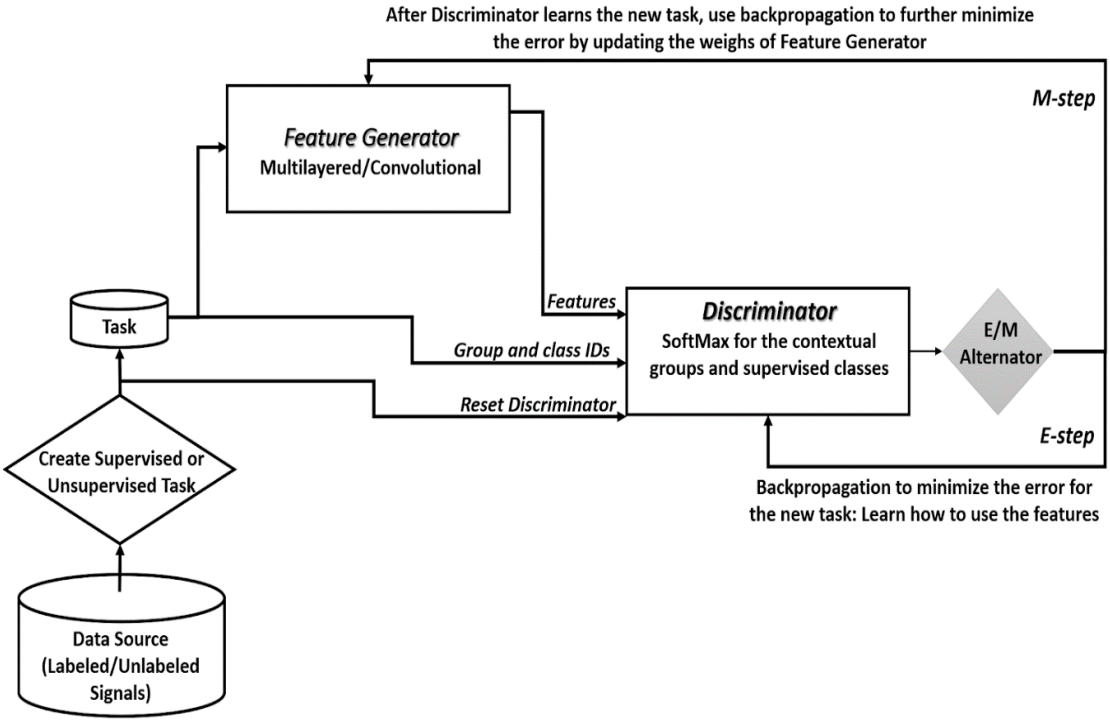

Figure 9. BeiimNet-Layer diagram. Feature Generator gradually learns more discriminative features that Discriminator can adapt to its ever-changing classification problems, providing feedback to Feature Generator.

Then the task resets the discriminator (this time with D output units corresponding to the D classes) and initiates the E-step of the E/M optimization. In the E-step, using the class-labeled examples in the task, BeiimNet trains the Discriminator (SoftMax) while freezing the feature generator/convolutional layers. The M-step starts: It freezes the discriminator and enables the feature generator to learn with weights backpropagated from the E method in the previous step. Furthermore, the M method/step uses backpropagation to minimize the error of the feature generator further. M-step is where the learned features evolve to be more beneficial for the supervised classification.

At the end of the M-step, BeiimNet switches mode and creates/forms another task (while alternating between employing supervised or unsupervised learning), and another E-M iteration starts. These iterations can be repeated a specified number of times or until convergence. The algorithm keeps track of transfer utility separately for convergence: Group accuracy for the unsupervised task and Class accuracy for the supervised task. These accuracies jitter from task to task as the tasks are created randomly; however, with iterations in time they are expected to converge.

CHAPTER 4 EXPERIMETNAL RESULTS

4.1 Experimental Dataset

The dataset used to evaluate the proposed method is called the vibrotactile signals texture classification dataset collected by Kursun and Patooghy (2020), which has 12 texture classes. Figure 10 shows the photographs of segments of the texture materials used in the data collection process. A 3D accelerometer sensor was connected to a probe rubbing against a rotating drum covered by these textured materials. For each texture, 20 seconds of sensor recordings were collected. The 3D accelerometer data consists of X, Y, Z recordings resampled at 200 Hz; therefore, the dataset is 3-by-4000. Kursun and Patooghy (2020) made the dataset publicly available and applied signal processing algorithms to demonstrate that the textures can be highly discriminated against using simple machine learning algorithms when sophisticated/manual signal feature extraction is performed on the data collected by vibrotactile sensors. In this thesis, CNNs will be applied to this dataset to demonstrate that deep learning methods can take advantage of transfer learning and avoid manual feature extraction.

Figure 10. Images of the 12 texture classes used in the dataset.

4.2 Experimental Setup

In Kursun and Patooghy (2020), there was an overlap between the training and test set signals because the sample windows were selected randomly from the 20-second recordings. This overlap made the classification task (on the test set) easier. However, in this thesis, the sample windows from the first 10 seconds of the recordings are used as the training set examples. The sample windows from the remaining 10 seconds of the recordings were used as the test set examples.

Moreover, to involve transfer learning experiments in this thesis, texture classes 1-10 were used as the source domain to learn transferrable features (i.e., train and validate the deep convolutional neural networks); and, classes 11-12 are used as the target domain, in which the transferred features are tested for their discriminative capabilities. Moreover, for retraining classifiers in the target domain, only 50 examples are used for each one of the two classes (from their first 10 seconds of recordings as mentioned above). For testing the retrained classifier, 100 examples are used from each class from the remaining 10-second recordings.

Unlike the feature extractors such as FFT used in the original study, this thesis's proposed transfer learning method can take advantage of retraining classifiers to improve the already learned features. Although FFT features could not discriminate classes 11 and 12 as accurately as the other pairs of classes, the proposed transfer learning approach learned features from the source domain (classes 1-10) that could be transferred to the target domain (classes 11 and 12) for achieving higher classification accuracies.

The BeiimNet architecture adhered to the best practices of CNN construction. Convolutions started small, with 32 feature maps to collect general features as effectively as possible, and the size of convolutions increased gradually to capture higher-level

features. In this experimental setup, all deep learning algorithms consisted of several layers, each layer containing a convolutional layer, ReLU-activation, and max-pooling layer. To compare with the proposed BeiimNet method, CNNs and Convolutional Autoencoders were implemented with comparable architectures (with the same number of layers). To compare with the single-layered CG-CNN, each method/architecture was implemented with one, two, and three layers. At the last layer, the network used an average pooling layer to make all architectures extract exactly 512 features in total. Figure 11 shows an example of the three-layered Autoencoder. Figures 12-14 show the one, two, and three-layered CNN architectures.

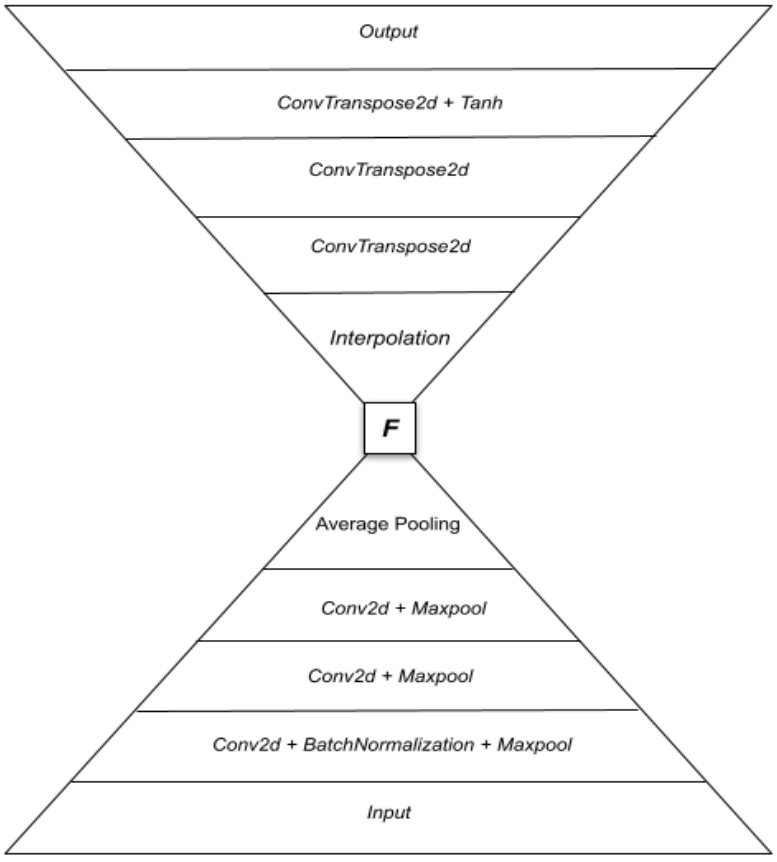

Figure 11. Autoencoder architecture with three-layered encoder (bottom pyramid) and three-layered decoder (top pyramid)

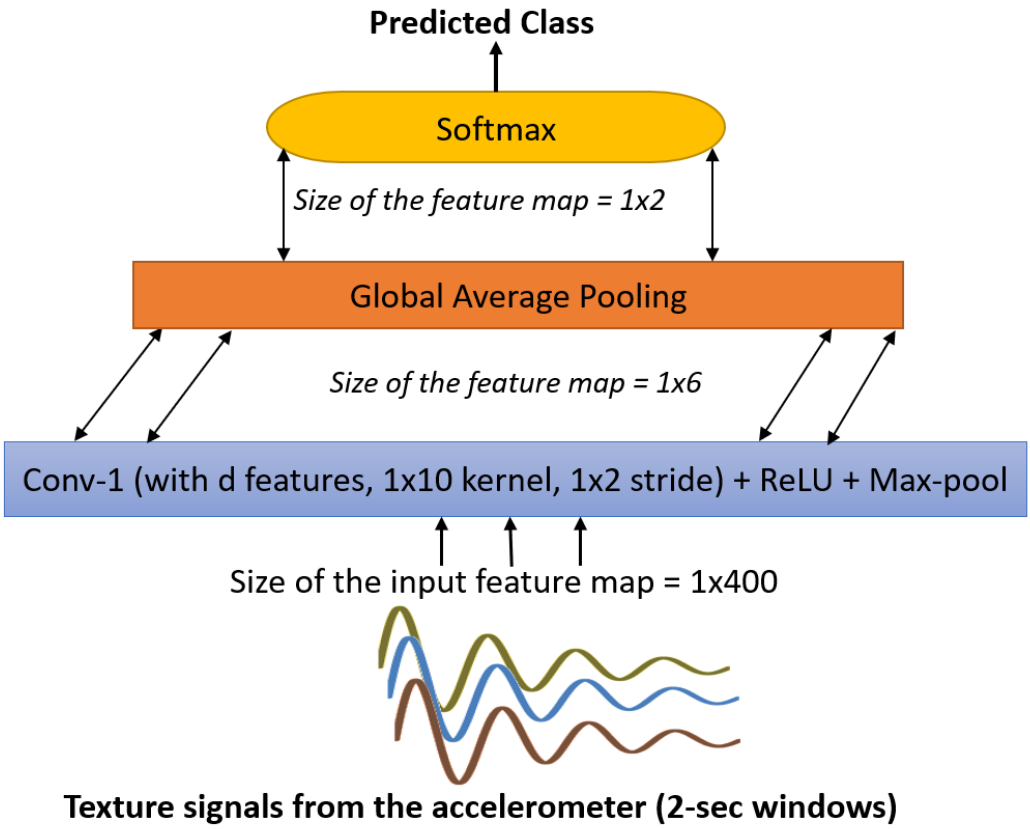

Figure 12. 1-layer CNN architecture used in the experiments.

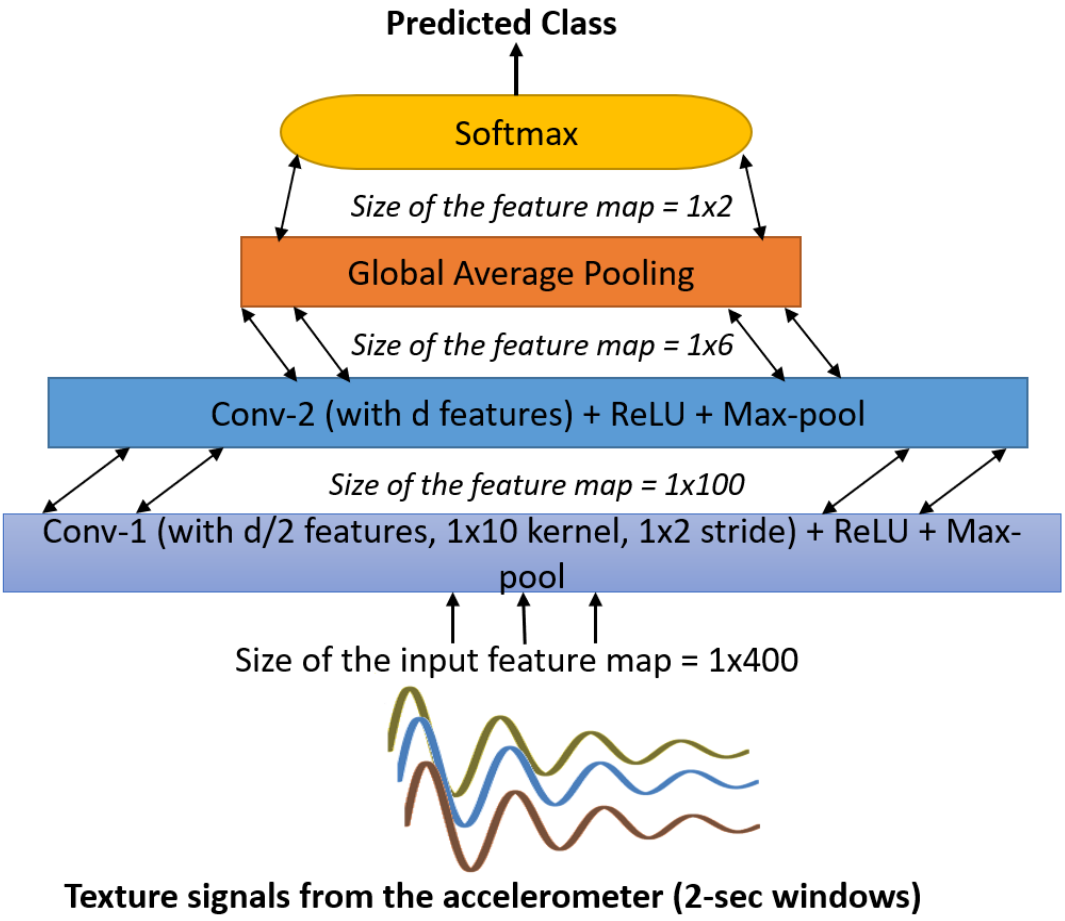

Figure 13. 2-layer CNN architecture used in the experiments.

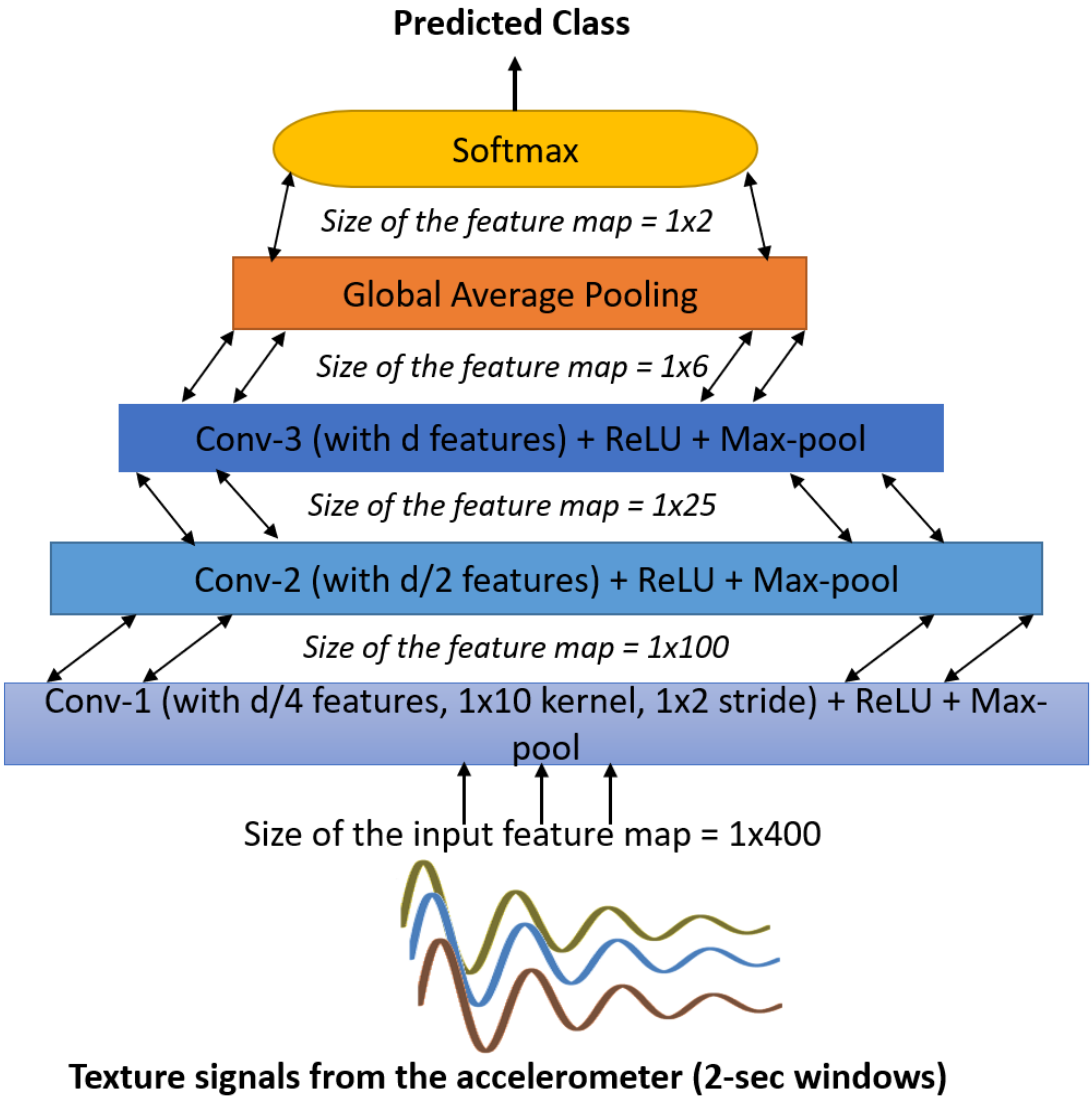

Figure 14. 3-layered CNN architecture used in the experiments. While the size of the receptive fields of the neurons double from one convolutional area to the next, the number of feature maps is doubled as well (e.g., for the 3-layer CNN, 25, 50, and 100 neurons were used in the first, second, and third layers, respectively).

As autoencoders are unsupervised, the full 20-second recordings were used for training, and the learned features were then transferred. As CNNs are supervised learners, only the first 10 seconds of the recordings in the source domain were used to train their

features. As the BeiimNet architecture utilizes semi-supervised learning, as mentioned in Chapter 3, its training alternated between supervised and unsupervised learning for each algorithm's task. For supervised tasks, the algorithm used the first 10 seconds of the recordings utilizing the known class labels. For the unsupervised tasks, it used the remaining 10-second recordings utilizing the contextual guidance principle. For the tactile dataset, using 30 contextual groups was found to be near optimal for training BeiimNet. Increasing the number of contextual groups did not affect the accuracy significantly, whereas using fewer contextual groups resulted in less transferrable features leading to poor accuracy.

In addition to the retraining of the learned deep networks in the target domain for measuring the quality of the learned features, the convolutional layers of these learned deep networks were frozen to use them solely as feature extractors (without retraining). In this feature-extractor setting, the learned features were fed to standard machine learning classifiers such as Support Vector Machines, Logistic Regression, and K-Nearest Neighbor (Alpaydin 2014; Fernández-Delgado et al. 2014), to classify texture classes 11 and 12 using X, Y, and Z sensors.

4.3 Results of Tactile Data

Table 1 compared FFT (Fast Fourier Transform) feature extraction with various classification methods and CNN, with and without transfer learning. As mentioned in Section 4.2, all the results were based on the target domain's binary classification problem, textures 11 and 12. Support Vector Machine, Naive Bayes, K-Nearest Neighbour, Multilayer Perceptron, and Random Forest classification methods were used to evaluate FFT features. For the Naive approach (as opposed to Transfer Learning), randomly initialized 3-layered deep CNN architecture was trained from scratch on the target domain. Also, the same architecture was trained on the source domain, textures 1-10, and the features were transferred and retrained to perform classification in the target domain. Except for X signal data, CNNs with and without transferred features outperformed classifiers with FFT extracted features (Table 1). The remaining tables represented the experimental results on deep learning, specifically the semi-supervised BeiimNet, compared with supervised CNNs and unsupervised Autoencoders.

Table 1. Comparison between CNN and Fourier Transform feature extraction methods on target domain classification for X, Y, Z and XYZ sensor-signal data. The results are represented as an average of 10 runs and their standard deviation.

Feature Extractors	***Sensor Signal Data for the Target Domain***			
	X	**Y**	**Z**	**XYZ**
Naive CNN	73.3 ± 6.0	66.2 ± 4.4	80.0 ± 3.8	79.6 ± 2.2
CNN with source domain features	75.1 ± 5.2	**68.3 ± 4.8**	**85.6 ± 2.6**	**82.8 ± 3.5**
FFT+SVM	57.0 ± 6.8	57.8 ± 5.4	68.9 ± 3.3	77.2 ± 7.4
FFT+NB	**77.2 ± 5.5**	47.1 ± 3.6	53.6 ± 2.8	56.6 ± 3.1
FFT+KNN	68.3 ± 3.8	50.6 ± 6.2	73.9 ± 4.3	69.1 ± 5.6
FFT+MLP	65.7 ± 15.2	50.0 ± 7.9	55.5 ± 9.9	61.6 ± 12.9
FFT+RF	70.6 ± 6.6	51.6 ± 6.8	68.1 ± 6.1	61.7 ± 9.7

For the application on the X-sensor data in Table 2, BeiimNet-1 performed better than other CNN retraining methods. While transferring CNN with 3-layered features gave SVM the best results, LR benefited the most from Autoencoder with a 2-layered encoder. Autoencoder with a 1-layered encoder delivered the best features to KNN and outperformed every other classifier for the target domain's X signal data with an accuracy of 91.8 percent.

Table 2. Target domain's X-Sensor signal classification results with source domain features generated by three methods - Autoencoders, CNNs, and BeiimNet. The results include an average of 10 runs and standard deviation for CNN (re-training), SVM, LR, and KNN classifications.

Source Domain Feature Extractor	*Target Domain and X Sensor Signal Data*			
	CNN (Re-training)	SVM	LR	KNN
BeiimNet-1	**79.9 ± 5.7**	75.2 ± 2.9	74.1 ± 3.7	79.2 ± 5.1
CNN 1-Block	75.2 ± 3.4	73.6 ± 4.5	74.6 ± 5.0	86.2 ± 4.3
Autoencoder 1-Block	71.7 ± 5.5	68.8 ± 5.3	68.7 ± 4.3	**91.8 ± 3.3**
BeiimNet-2	79.8 ± 4.2	73.7 ± 5.0	76.6 ± 5.8	83.7 ± 5.4
CNN 2-Block	75.0 ± 2.4	69.3 ± 5.5	75.1 ± 4.0	86.0 ± 2.3
Autoencoder 2-Block	75.2 ± 3.7	74.9 ± 4.3	**81.2 ± 3.5**	64.4 ± 7.5
BeiimNet-3	78.2 ± 4.0	70.2 ± 8.9	72.4 ± 7.2	76.4 ± 8.3
CNN 3-Block	75.1 ± 5.2	**75.7 ± 4.8**	76.7 ± 3.2	76.5 ± 4.9
Autoencoder 3-Block	77.8 ± 4.0	71.7 ± 3.3	76.4 ± 2.3	54.8 ± 2.3

In Table 3, CNN with 2-block features gave the highest accuracy result among all re-trained CNNs. However, BeiimNet-1 features transferred to SVM and Logistic regression outperformed all CNNs, with logistic regression displaying the best results for Y-signals - 82.8 percent.

Table 3. Target domain's Y-Sensor signal classification results with source domain features generated by three methods - Autoencoders, CNNs, and BeiimNet. The results include an average of 10 runs and standard deviation for CNN (re-training), SVM, LR, and KNN classifications.

Source Domain Feature Extractor	*Target Domain and Y Sensor Signal Data*			
	CNN (Re-training)	SVM	LR	KNN
BeiimNet-1	71.0 ± 5.7	**81.8 ± 3.3**	**82.8 ± 4.5**	70.7 ± 5.0
CNN 1-Block	68.9 ± 5.1	78.7 ± 4.1	79.5 ± 2.5	65.0 ± 4.8
Autoencoder 1-Block	60.1 ± 3.8	77.9 ± 3.8	77.3 ± 4.2	61.6 ± 4.7
BeiimNet-2	75.7 ± 3.4	75.1 ± 7.6	75.4 ± 9.1	**72.0 ± 5.2**
CNN 2-Block	**77.7 ± 3.0**	81.0 ± 3.8	82.7 ± 2.4	66.7 ± 4.2
Autoencoder 2-Block	57.0 ± 6.2	60.4 ± 6.1	59.3 ± 4.1	52.6 ± 5.7
BeiimNet-3	70.2 ± 7.5	73.3 ± 7.4	70.7 ± 7.8	67.3 ± 10.1
CNN 3-Block	68.3 ± 4.8	71.2 ± 6.2	66.1 ± 3.8	59.4 ± 5.0
Autoencoder 3-Block	57.6 ± 5.4	53.3 ± 5.0	52.8 ± 5.0	54.0 ± 5.4

From Table 4, BeiimNet-2 and BeiimNet-3 features performed the best for all Z sensor signal classification methods, with BeiimNet-3 CNN-retraining outperforming the

rest, 88.2 percent. SVM and KNN benefitted the most from BeiimNet-2, while logistic regression results performed well with BeiimNet-3 features.

Table 4. Target domain's Z-Sensor signal classification results with source domain features generated by three methods - Autoencoders, CNNs, and BeiimNet. The results include an average of 10 runs and standard deviation for CNN (re-training), SVM, LR, and KNN classifications.

Source Domain Feature Extractor	*Target Domain and Z Sensor Signal Data*			
	CNN (Re-training)	SVM	LR	KNN
BeiimNet-1	74.5 ± 6.2	75.0 ± 7.1	76.2 ± 6.1	76.3 ± 7.0
CNN 1-Block	72.7 ± 5.2	64.8 ± 5.6	74.1 ± 4.9	71.8 ± 4.7
Autoencoder 1-Block	65.6 ± 5.1	62.9 ± 14.4	67.8 ± 15.4	64.3 ± 8.1
BeiimNet-2	87.2 ± 4.7	**86.6 ± 3.3**	84.0 ± 2.6	**81.6 ± 4.4**
CNN 2-Block	84.0 ± 1.7	76.1 ± 4.9	79.1 ± 4.1	71.5 ± 3.0
Autoencoder 2-Block	77.2 ± 3.4	77.0 ± 2.2	78.3 ± 2.5	67.6 ± 6.2
BeiimNet-3	**88.2 ± 5.3**	85.7 ± 5.1	**84.6 ± 5.9**	80.6 ± 5.2
CNN 3-Block	85.6 ± 2.6	73.2 ± 3.5	79.8 ± 3.3	65.7 ± 5.5
Autoencoder 3-Block	76.85 ± 3.34	76.45 ± 2.93	77.3 ± 3.56	58.8 ± 6.9

For the application on the three sensors combined (referred to as XYZ Signals), BeiimNet-2 outperformed the other methods for CNN retraining, SVM, and KNN. At the

same time, Logistic Regression benefited slightly more from BeiimNet-3 features. Overall, CNN retraining with BeiimNet-2 features gave the highest result of 93.1 percent for the XYZ signal combination (see Table 5).

Table 5. Target domain's XYZ-sensor signal classification results with source domain features generated by three methods - Autoencoders, CNNs, and BeiimNet. The results include an average of 10 runs and standard deviation for CNN (re-training), SVM, LR, and KNN classifications

Source Domain Feature Extractor	*Target Domain and XYZ Sensor Signal Data*			
	CNN (Re-training)	SVM	LR	KNN
BeiimNet-1	87.1 ± 3.9	79.6 ± 6.8	80.6 ± 5.4	76.6 ± 6.9
CNN 1-Block	80.5 ± 8.4	74.7 ± 3.7	80.5 ± 2.2	72.8 ± 5.4
Autoencoder 1-Block	79.0 ± 6.5	75.7 ± 5.1	76.9 ± 4.9	71.6 ± 4.6
BeiimNet-2	**93.1 ± 2.1**	**82.4 ± 2.4**	82.1 ± 3.8	**78.0 ± 4.6**
CNN 2-Block	86.7 ± 1.3	76.4 ± 5.7	82.4 ± 4.5	73.0 ± 4.8
Autoencoder 2-Block	83.4 ± 4.6	78.5 ± 3.3	80.2 ± 3.7	61.4 ± 6.3
BeiimNet-3	92.1 ± 3.4	81.0 ± 3.4	**82.5 ± 2.9**	77.7 ± 5.9
CNN 3-Block	82.8 ± 3.5	67.0 ± 5.6	70.1 ± 5.6	66.8 ± 6.4
Autoencoder 3-Block	78.3 ± 3.0	76.2 ± 3.9	77.4 ± 4.0	55.9 ± 7.6

Table 6 compared CG-CNN and BeiimNet to evaluate how well their features generalize without further retraining (fine-tuning in the target domain). As BeiimNet uses a semi-supervised extension of the contextual guidance principle, it outperformed the unsupervised approach suggesting that, as expected, incorporating available class labels improves the discriminative capabilities of the learned features.

Table 6. Target domain's XYZ-sensor signal classification results with source domain features generated three architectures of CG-CNN and BeiimNet. The results include an average of 10 runs and a standard deviation of SVM, LR, and KNN classifications.

Source Domain Feature Extractor	*Target Domain and XYZ Sensor Signal Data*		
	SVM	LR	KNN
CG-CNN-1	70.3 ± 11.1	70.2 ± 10.2	64.7 ± 4.2
BeiimNet-1	79.6 ± 6.8	80.6 ± 5.4	76.6 ± 6.9
CG-CNN-2	77.3 ± 5.9	79.0 ± 5.4	72.5 ± 8.5
BeiimNet-2	82.4 ± 2.4	82.1 ± 3.8	81.6 ± 4.4
CG-CNN-3	74.6 ± 4.4	76.5 ± 3.1	76.8 ± 2.5
BeiimNet-3	82.0 ± 3.4	82.5 ± 2.9	77.7 ± 5.9

As seen in Figures 15 to 22, the supervised classification accuracy of the Unsupervised network (CG-CNN) was lower than that of the Semi-supervised (BeiimNet), which was the contribution of semi-supervised learning to the contextually guided training. On the other hand, as the Supervised network did not take advantage of the contextual guidance, it did not generalize as well as BeiimNet.

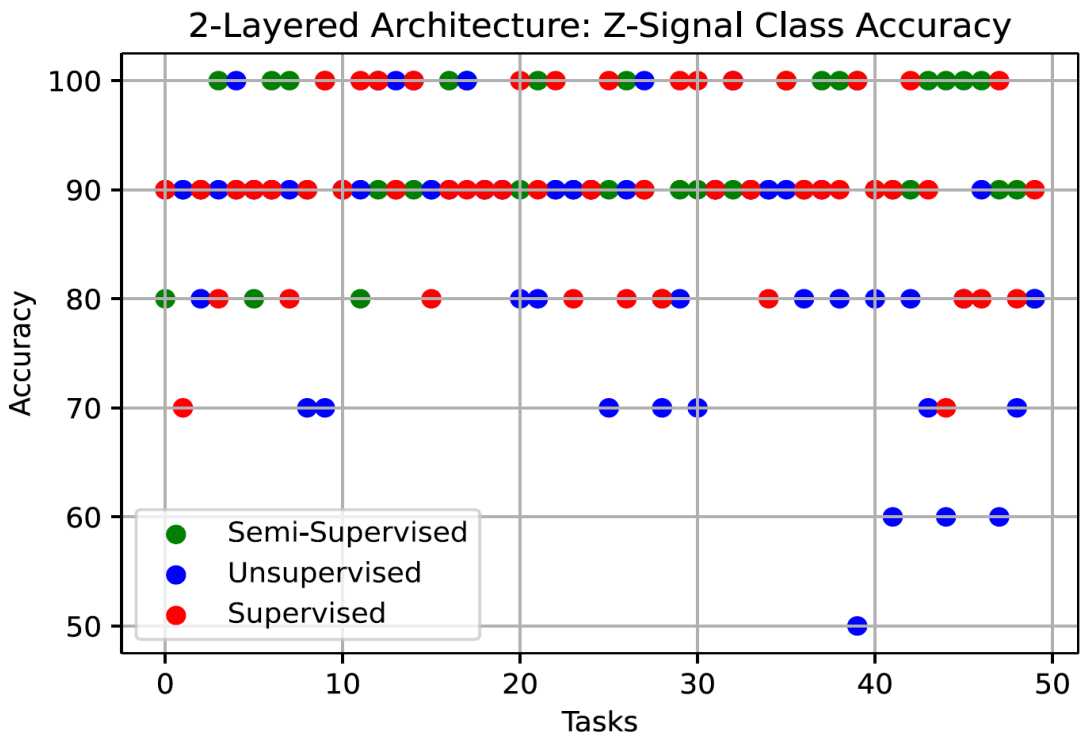

Figure 15. Illustration of Supervised, Unsupervised, and Semi-Supervised 2-layer network convergence source domain class-accuracy – tactile sensor-signal dataset (Z-sensor)

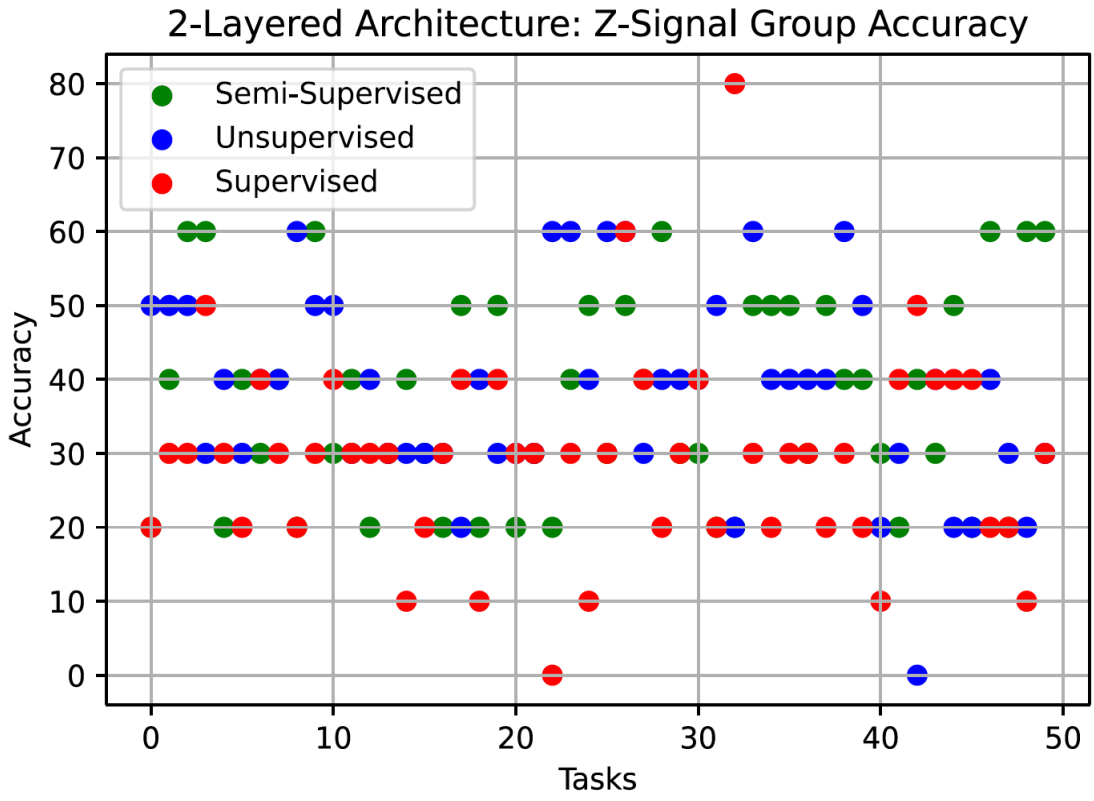

Figure 16. Illustration of Supervised, Unsupervised, and Semi-Supervised 2-layer network convergence source domain contextual group accuracy – tactile sensor-signal dataset (Z-sensor)

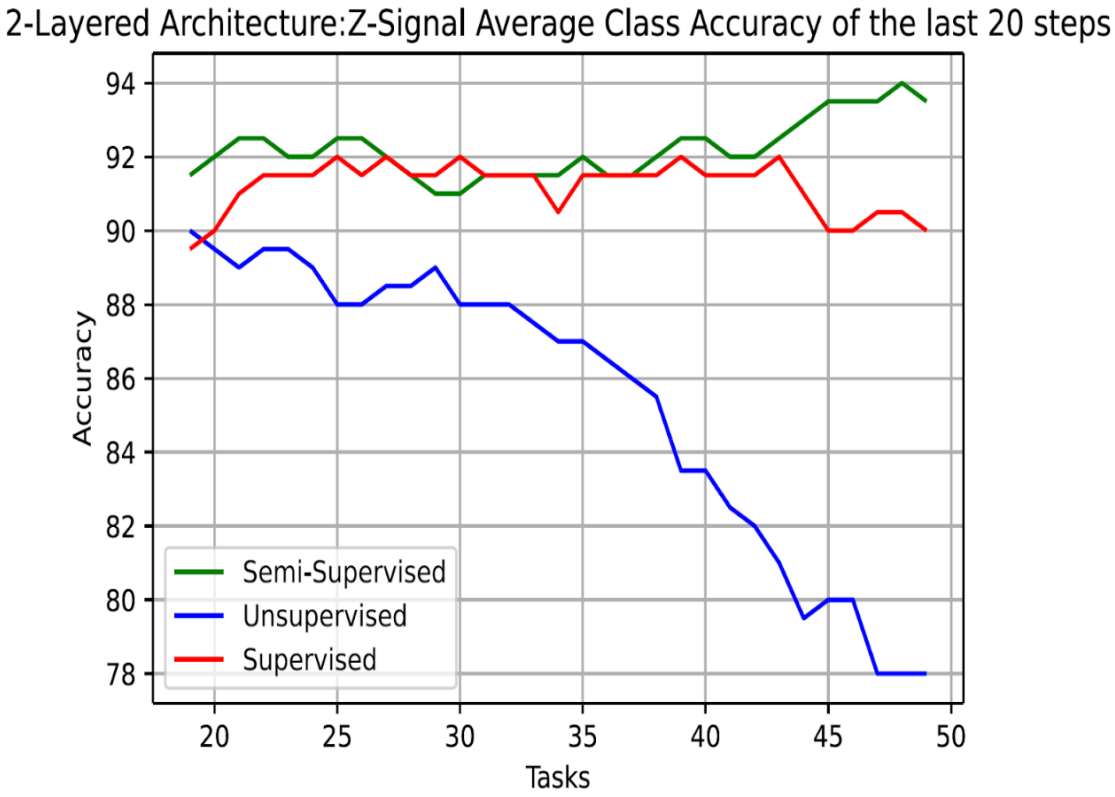

Figure 17. Illustration of Supervised, Unsupervised, and Semi-Supervised 2-layer network convergence source domain class accuracy of the last 20 steps – tactile sensor-signal dataset (Z-sensor)

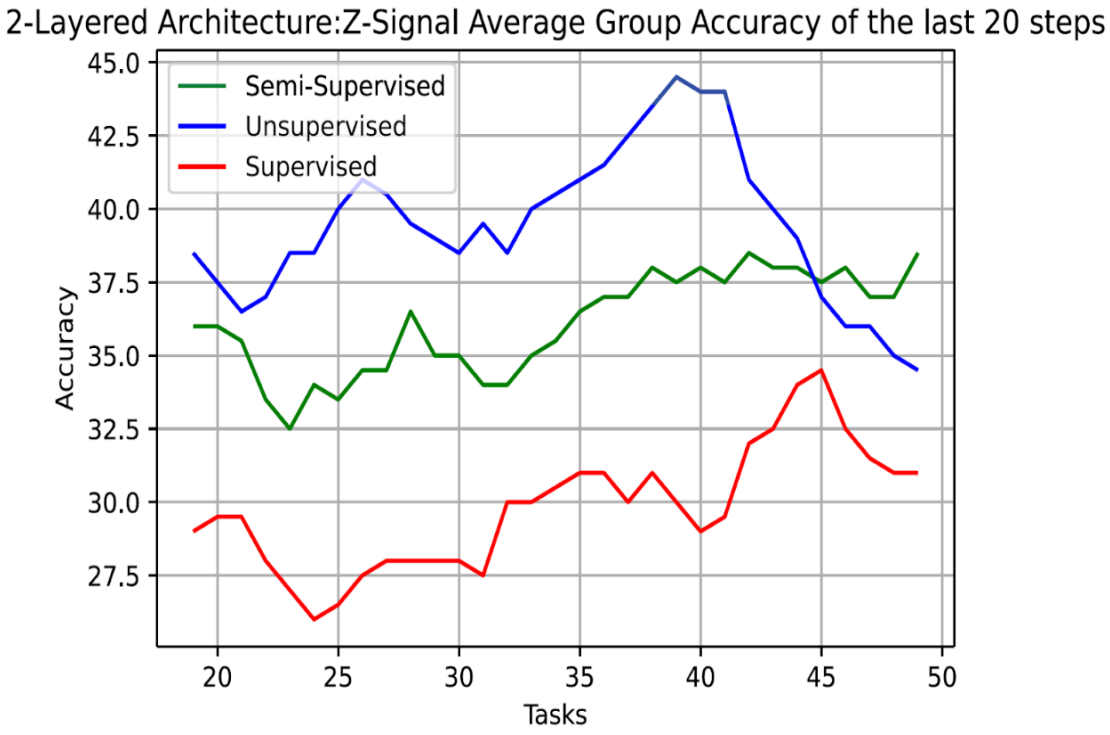

Figure 18. Illustration of Supervised, Unsupervised, and Semi-Supervised 2-layer network convergence source domain group accuracy of the last 20 steps – tactile sensor-signal dataset (Z-sensor)

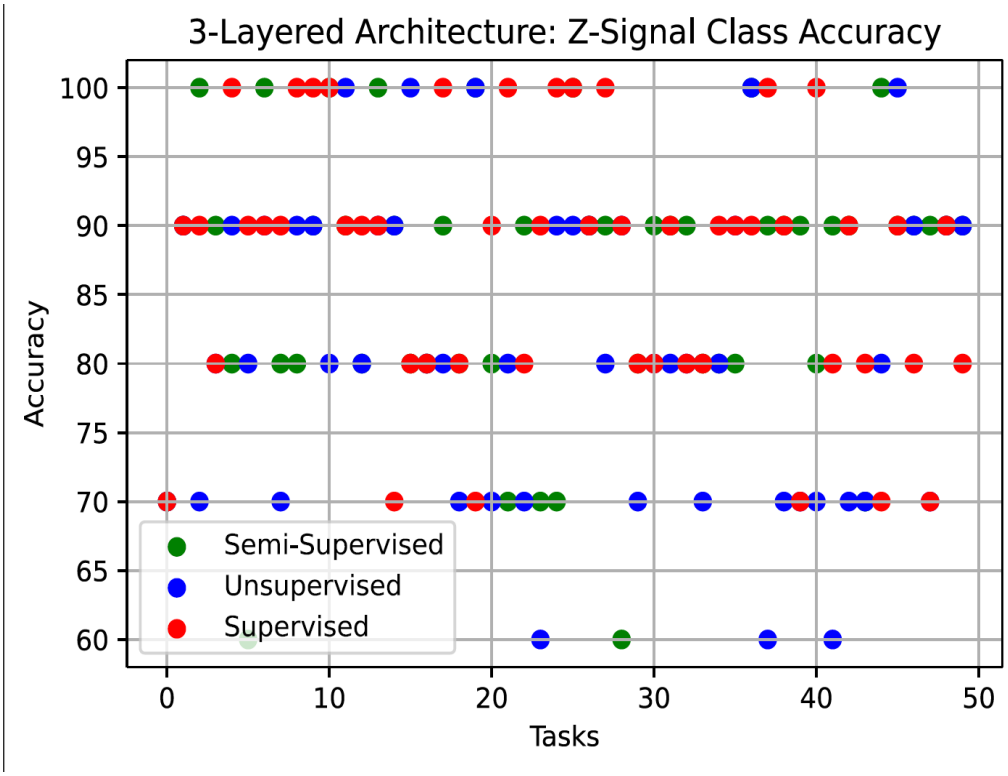

Figure 19. Illustration of Supervised, Unsupervised, and Semi-Supervised 3-layer network convergence source domain class-accuracy – tactile sensor-signal dataset (Z-sensor)

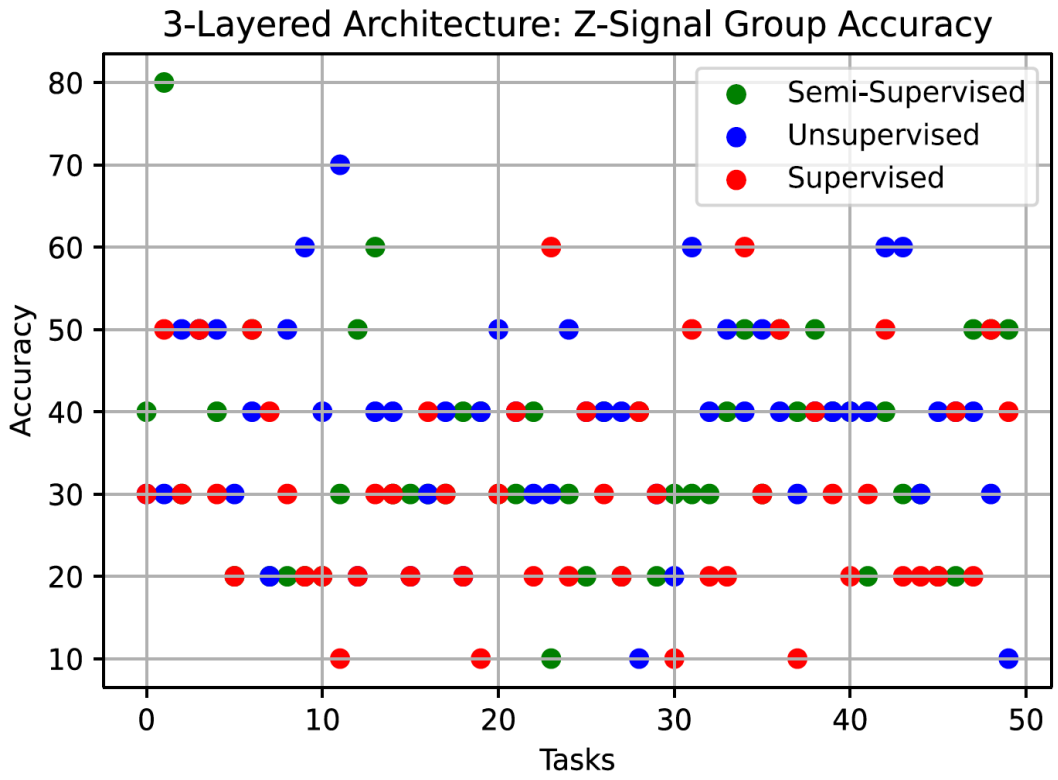

Figure 20. Illustration of Supervised, Unsupervised, and Semi-Supervised 3-layer network convergence source domain contextual group accuracy – tactile sensor-signal dataset (Z-sensor)

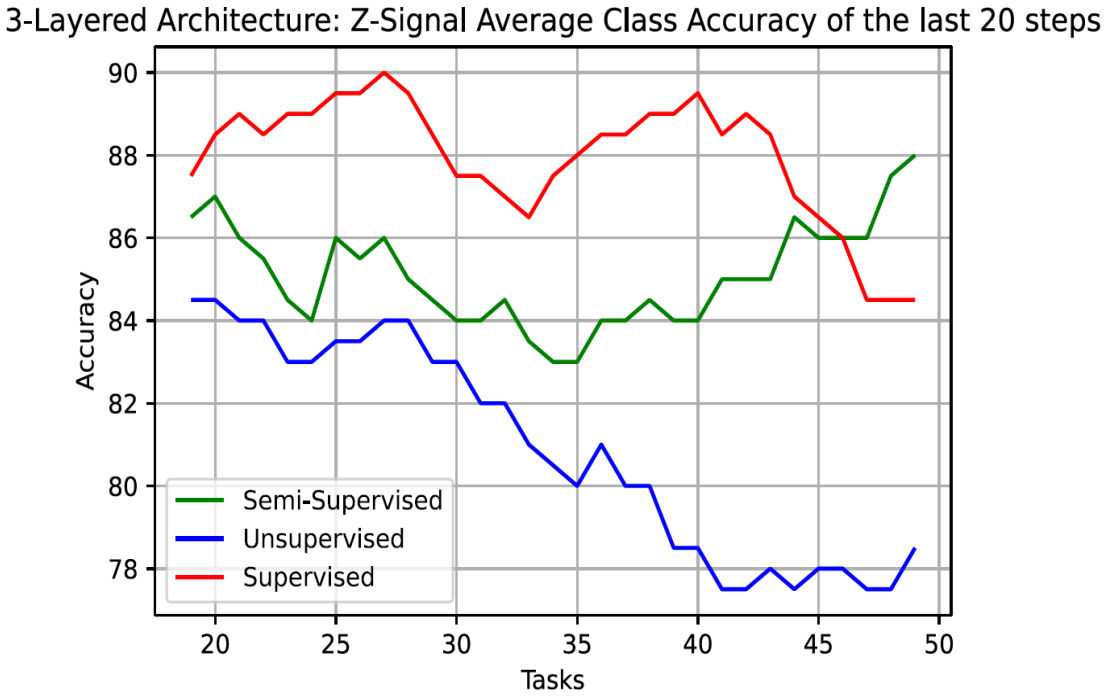

Figure 21. Illustration of Supervised, Unsupervised, and Semi-Supervised 3-layer network convergence source domain class accuracy of the last 20 steps – tactile sensor-signal dataset (Z-sensor)

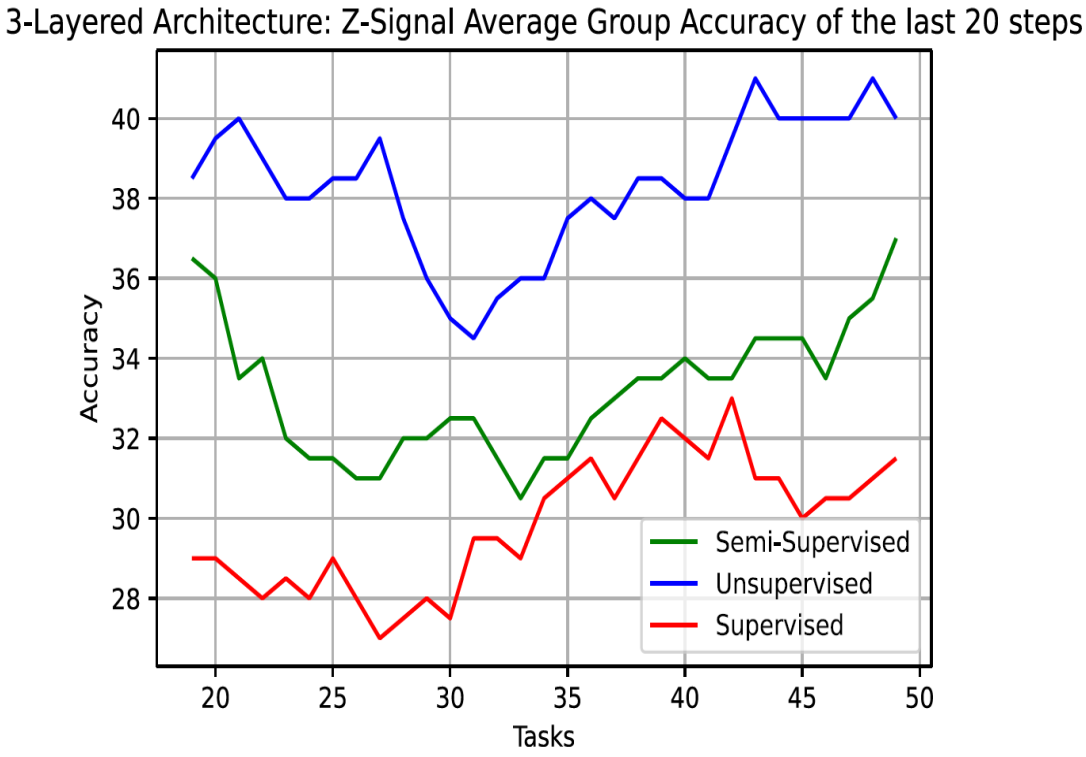

Figure 22. Illustration of Supervised, Unsupervised, and Semi-Supervised e-layer network convergence source domain group accuracy of the last 20 steps – tactile sensor-signal dataset (Z-sensor)

CHAPTER 5 CONCLUSION

Deep learning has shown great success in AI in recent years, offering practical and transferable solutions in many applications with potential improvements and innovations. The availability of large volumes of data is undoubtedly one of the key contributors to deep learning success. With the high cost of manual labelling data, the semi-supervised deep learning systems can use massive amounts of unlabelled data for regularizing/improving their extracted features. For example, rapidly advanced hardware and cloud technologies accessing raw/unlabelled data via various sensors and the internet makes semi-supervised deep learning very appealing. Recently proposed Contextually Guided Neural Networks (CG-CNN) offer an extendible/scalable approach to deep learning. In its original form, CG-CNN uses a self-supervised approach for utilizing unlabelled examples and learns to extract highly generalizable/transferable features. In this thesis, as a semi-supervised multi-layer extension of CG-CNN, the BeiimNet method is developed. BeiimNet shows great promise in developing transferable features, as demonstrated in its application to a vibrotactile signal texture classification dataset. BeiimNet is compared against CG-CNN, regular CNNs, Autoencoders, Fourier Transform, followed by standard machine learning classifiers such as Random Forests, SVMs, Nearest Neighbors, and so on.

Although the unsupervised CG-CNN approach excels in discriminating contextual seed classes, without proper fine-tuning by taking advantage of the labelled examples, it may fall short in the subsequent supervised classification tasks. Therefore, the semi-supervised approach BeiimNet is proposed in this thesis for having a higher potential for classification. As future work, BeiimNet will be applied to more challenging image classification tasks in combination with higher-order contextual clues such as image segments.

REFERENCES

Abiodun, Oludare Isaac, Muhammad Ubale Kiru, Aman Jantan, Abiodun Esther Omolara, Kemi Victoria Dada, Abubakar Malah Umar, Okafor Uchenwa Linus, Humaira Arshad, Abdullahi Aminu Kazaure, and Usman Gana. 2019. "Comprehensive Review of Artificial Neural Network Applications to Pattern Recognition." *IEEE Access* 7: 158820–46. https://doi.org/10.1109/ACCESS.2019.2945545.

Alom, Md Zahangir, Tarek M. Taha, Christopher Yakopcic, Stefan Westberg, Paheding Sidike, Mst Shamima Nasrin, Brian C. Van Essen, Abdul A. S. Awwal, and Vijayan K. Asari. 2018. "The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches." *CoRR* abs/1803.01164. http://arxiv.org/abs/1803.01164.

Alpaydin, Ethem. 2014. *Introduction to Machine Learning*. 3rd ed. Adaptive Computation and Machine Learning. Cambridge, MA: MIT Press.

Baur, Christoph, Shadi Albarqouni, and Nassir Navab. 2017. "Semi-Supervised Deep Learning for Fully Convolutional Networks." In *Medical Image Computing and Computer Assisted Intervention − MICCAI 2017*, edited by Maxime Descoteaux, Lena Maier-Hein, Alfred Franz, Pierre Jannin, D. Louis Collins, and Simon Duchesne, 311–19. Cham: Springer International Publishing.

Becker, Suzanna, and Geoffrey Hinton. 1992. "Self-Organizing Neural Network That Discovers Surfaces in Random-Dot Stereograms." *Nature* 355 (February): 161–63. https://doi.org/10.1038/355161a0.

Bengio, Yoshua. 2012. "Deep Learning of Representations for Unsupervised and Transfer Learning." In *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, edited by Isabelle Guyon, Gideon Dror, Vincent Lemaire, Graham Taylor, and Daniel Silver, 27:17–36. Proceedings of Machine Learning Research. Bellevue, Washington, USA: PMLR. http://proceedings.mlr.press/v27/bengio12a.html.

Deng, Jia, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. "Imagenet: A Large-Scale Hierarchical Image Database." In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 248–55. Ieee.

Dosovitskiy, Alexey, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. 2014. “Discriminative Unsupervised Feature Learning with Convolutional Neural Networks.” In *Advances in Neural Information Processing Systems*, edited by Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger. Vol. 27. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2014/file/07563a3fe3bbe7e3ba84431ad9d055af-Paper.pdf.

Enguehard, Joseph, Peter O’Halloran, and Ali Gholipour. 2019. “Semi-Supervised Learning With Deep Embedded Clustering for Image Classification and Segmentation.” *IEEE Access* 7: 11093–104. https://doi.org/10.1109/ACCESS.2019.2891970.

Favorov, Oleg V., and Olcay Kursun. 2011. “Neocortical Layer 4 as a Pluripotent Function Linearizer.” *Journal of Neurophysiology* 105 (3): 1342–60. https://doi.org/10.1152/jn.00708.2010.

Favorov, Oleg V., and Dan Ryder. 2004. “SINBAD: A Neocortical Mechanism for Discovering Environmental Variables and Regularities Hidden in Sensory Input.” *Biological Cybernetics* 90 (3): 191–202. https://doi.org/10.1007/s00422-004-0464-8.

Fernández-Delgado, Manuel, Eva Cernadas, Senén Barro, and Dinani Amorim. 2014. “Do We Need Hundreds of Classifiers to Solve Real World Classification Problems?” *Journal of Machine Learning Research* 15 (90): 3133–81.

Finn, Chelsea, Pieter Abbeel, and Sergey Levine. 2017. “Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks.” *ArXiv:1703.03400 [Cs]*, July. http://arxiv.org/abs/1703.03400.

Gao, Fei, Hyunsoo Yoon, Teresa Wu, and Xianghua Chu. 2020. “A Feature Transfer Enabled Multi-Task Deep Learning Model on Medical Imaging.” *Expert Systems with Applications* 143 (April): 112957. https://doi.org/10.1016/j.eswa.2019.112957.

Ghaderi, Amir, and Vassilis Athitsos. 2016. “Selective Unsupervised Feature Learning with Convolutional Neural Network (S-CNN).” In *2016 23rd International Conference on Pattern Recognition (ICPR)*, 2486–90. Cancun: IEEE. https://doi.org/10.1109/ICPR.2016.7900009.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.

Goodfellow, Ian J., Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. "Generative Adversarial Networks." *ArXiv:1406.2661 [Cs, Stat]*, June. http://arxiv.org/abs/1406.2661.

Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2020. "Generative Adversarial Networks." *Communications of the ACM* 63 (11): 139–44. https://doi.org/10.1145/3422622.

Hawkins, Jeff, Subutai Ahmad, and Yuwei Cui. 2017. "A Theory of How Columns in the Neocortex Enable Learning the Structure of the World." *Frontiers in Neural Circuits* 11: 81. https://doi.org/10.3389/fncir.2017.00081.

Hawkins, Jeff, and Sandra Blakeslee. 2004. *On Intelligence*. USA: Times Books.

Hayou, Soufiane, Arnaud Doucet, and Judith Rousseau. 2019. *On the Impact of the Activation Function on Deep Neural Networks Training*.

Hjelm, R. Devon, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. 2019. "Learning Deep Representations by Mutual Information Estimation and Maximization." In .

Hoffer, Elad, and Nir Ailon. 2018. "Semi-Supervised Deep Learning by Metric Embedding." *ArXiv:1611.01449 [Cs]*, December. http://arxiv.org/abs/1611.01449.

J. C. Gwilliam, Z. Pezzementi, E. Jantho, A. M. Okamura, and S. Hsiao. 2010. "Human vs. Robotic Tactile Sensing: Detecting Lumps in Soft Tissue." In *2010 IEEE Haptics Symposium*, 21–28. https://doi.org/10.1109/HAPTIC.2010.5444685.

Körding, Konrad P, and Peter König. 2000. "Learning with Two Sites of Synaptic Integration." *Network: Computation in Neural Systems* 11 (1): 25–39. https://doi.org/10.1088/0954-898X_11_1_302.

Kursun, Olcay, Ethem Alpaydin, and Oleg V. Favorov. 2011. “Canonical Correlation Analysis Using Within-Class Coupling.” *Pattern Recogn. Lett.* 32 (2): 134–44. https://doi.org/10.1016/j.patrec.2010.09.025.

Kursun, Olcay, Semih Dinc, and Oleg V. Favorov. 2021. “Contextually Guided Convolutional Neural Networks for Learning Most Transferable Representations.” *ArXiv:2103.01566 [Cs]*, March. http://arxiv.org/abs/2103.01566.

Kursun, Olcay, and Oleg V. Favorov. 2019. “Suitability of Features of Deep Convolutional Neural Networks for Modeling Somatosensory Information Processing.” In . Vol. 10995. https://doi.org/10.1117/12.2518573.

Kursun, Olcay, and Ahmad Patooghy. 2020. “An Embedded System for Collection and Real-Time Classification of a Tactile Dataset.” *IEEE Access* 8: 97462–73. https://doi.org/10.1109/ACCESS.2020.2996576.

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. 2015. “Deep Learning.” *Nature* 521 (7553): 436–44. https://doi.org/10.1038/nature14539.

LeNail, Alexander. 2019. “NN-SVG: Publication-Ready Neural Network Architecture Schematics.” *Journal of Open Source Software* 4 (33): 747. https://doi.org/10.21105/joss.00747.

Li Fei-Fei, R. Fergus, and P. Perona. 2004. “Learning Generative Visual Models from Few Training Examples: An Incremental Bayesian Approach Tested on 101 Object Categories.” In *2004 Conference on Computer Vision and Pattern Recognition Workshop*, 178–178. https://doi.org/10.1109/CVPR.2004.383.

Malik, Hassaan, Muhammad Shoaib Farooq, Adel Khelifi, Adnan Abid, Junaid Nasir Qureshi, and Muzammil Hussain. 2020. “A Comparison of Transfer Learning Performance Versus Health Experts in Disease Diagnosis From Medical Imaging.” *IEEE Access* 8: 139367–86. https://doi.org/10.1109/ACCESS.2020.3004766.

Marsland, Stephen. 2014. *Machine Learning: An Algorithmic Perspective*. 2nd ed. New Jersey, USA: CRC Press.

Miller, Arthur I. 2019. “10 Ian Goodfellow’s Generative Adversarial Networks: AI Learns to Imagine.” In *The Artist in the Machine: The World of AI-Powered Creativity*, 87–98.

O’Mahony, Niall, Sean Campbell, Anderson Carvalho, Suman Harapanahalli, Gustavo Velasco Hernandez, Lenka Krpalkova, Daniel Riordan, and Joseph Walsh. 2020. “Deep Learning vs. Traditional Computer Vision.” In *Advances in Computer Vision*, edited by Kohei Arai and Supriya Kapoor, 128–44. Cham: Springer International Publishing.

Parisi, German I., Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. 2019. “Continual Lifelong Learning with Neural Networks: A Review.” *Neural Networks* 113 (May): 54–71. https://doi.org/10.1016/j.neunet.2019.01.012.

Phillips, William A., and Wolf Singer. 1997. “In Search of Common Foundations for Cortical Computation.” *Behavioral and Brain Sciences* 20 (4): 657–83. https://doi.org/10.1017/S0140525X9700160X.

Sanodiya, Rakesh Kumar, and Leehter Yao. 2020. “Unsupervised Transfer Learning via Relative Distance Comparisons.” *IEEE Access* 8: 110290–305. https://doi.org/10.1109/ACCESS.2020.3002666.

Schmitz, A., Y. Bansho, K. Noda, H. Iwata, T. Ogata, and S. Sugano. 2014. “Tactile Object Recognition Using Deep Learning and Dropout.” In *2014 IEEE-RAS International Conference on Humanoid Robots*, 1044–50. https://doi.org/10.1109/HUMANOIDS.2014.7041493.

Schopfer, M., H. Ritter, and G. Heidemann. 2007. “Acquisition and Application of a Tactile Database.” In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 1517–22. https://doi.org/10.1109/ROBOT.2007.363539.

Shorten, Connor, and Taghi M. Khoshgoftaar. 2019. “A Survey on Image Data Augmentation for Deep Learning.” *Journal of Big Data* 6 (1): 60. https://doi.org/10.1186/s40537-019-0197-0.

Shrestha, Ajay, and Ausif Mahmood. 2019. “Review of Deep Learning Algorithms and Architectures.” *IEEE Access* 7: 53040–65. https://doi.org/10.1109/ACCESS.2019.2912200.

Tan, Chuanqi, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. 2018. “A Survey on Deep Transfer Learning.” *ArXiv:1808.01974 [Cs, Stat]*, August. http://arxiv.org/abs/1808.01974.

“Transfer Learning.” 2021. In *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Transfer_learning&oldid=1014785266.

Wang, Yaqing, Quanming Yao, James Kwok, and Lionel M. Ni. 2020. “Generalizing from a Few Examples: A Survey on Few-Shot Learning.” *ArXiv:1904.05046 [Cs]*, March. http://arxiv.org/abs/1904.05046.

Weiss, Karl, Taghi M. Khoshgoftaar, and DingDing Wang. 2016. “A Survey of Transfer Learning.” *Journal of Big Data* 3 (1): 9. https://doi.org/10.1186/s40537-016-0043-6.

Yosinski, Jason, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. “How Transferable Are Features in Deep Neural Networks?” *ArXiv:1411.1792 [Cs]*, November. http://arxiv.org/abs/1411.1792.

Zeiler, Matthew D., and Rob Fergus. 2014. “Visualizing and Understanding Convolutional Networks.” In *Computer Vision – ECCV 2014*, edited by David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, 818–33. Cham: Springer International Publishing.

Zhang, Wei, Yongfeng Zhu, and Qiang Fu. 2019. “Semi-Supervised Deep Transfer Learning-Based on Adversarial Feature Learning for Label Limited SAR Target Recognition.” *IEEE Access* 7: 152412–20. https://doi.org/10.1109/ACCESS.2019.2948404.

Zhao, Junbo Jake, Michaël Mathieu, and Yann LeCun. 2016. “Energy-Based Generative Adversarial Network.” *CoRR* abs/1609.03126. http://arxiv.org/abs/1609.03126.

Zhao, Zhong-Qiu, Peng Zheng, Shou-tao Xu, and Xindong Wu. 2019. “Object Detection with Deep Learning: A Review.” *ArXiv:1807.05511 [Cs]*, April. http://arxiv.org/abs/1807.05511.

Zheng, Xiaoqing, Hongcheng Wang, Jie Chen, Yaguang Kong, and Song Zheng. 2020. “A Generic Semi-Supervised Deep Learning-Based Approach for Automated Surface Inspection.” *IEEE Access* 8: 114088–99. https://doi.org/10.1109/ACCESS.2020.3003588.

Zhou, J., L. Zheng, Y. Wang, and C. Gogu. 2020. “A Multistage Deep Transfer Learning Method for Machinery Fault Diagnostics Across Diverse Working Conditions and Devices.” *IEEE Access* 8: 80879–98. https://doi.org/10.1109/ACCESS.2020.2990739.

Zhou, Joey Tianyi, Sinno Jialin Pan, Ivor W Tsang, and Yan Yan. n.d. “Hybrid Heterogeneous Transfer Learning through Deep Learning,” 7.

Zorins, Aleksejs, and Peteris Grabusts. 2015. “Artificial Neural Networks and Human Brain: Survey of Improvement Possibilities of Learning.” *Environment. Technology. Resources. Proceedings of the International Scientific and Practical Conference* 3 (June): 228. https://doi.org/10.17770/etr2015vol3.165.