ANKARA YILDIRIM BEYAZIT UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES



CLASSIFICATION OF EEG SIGNALS USING TRANSFER LEARNING ON CONVOLUTIONAL NEURAL NETWORKS VIA SPECTROGRAM

M.Sc. Thesis by

Ahmet Esad TOP

Department of Computer Engineering

December, 2018

ANKARA

CLASSIFICATION OF EEG SIGNALS USING TRANSFER LEARNING ON CONVOLUTIONAL NEURAL NETWORKS VIA SPECTROGRAM

A Thesis Submitted to

The Graduate School of Natural and Applied Sciences of

Ankara Yıldırım Beyazıt University

In Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Engineering, Department of Computer Engineering

by

Ahmet Esad TOP

December, 2018

ANKARA

M.Sc. THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"CLASSIFICATION OF EEG SIGNALS USING TRANSFER LEARNING ON CONVOLUTIONAL NEURAL NETWORKS VIA SPECTROGRAM"** completed by **AHMET ESAD TOP** under supervision of **ASSIST. PROF. DR. HİLAL KAYA** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

.....

Assist. Prof. Dr. Hilal KAYA

Supervisor

.....

Assist. Prof. Dr. Özkan KILIÇ

Jury member

.....

Assoc. Prof. Dr. Mehmet Serdar GÜZEL

Jury Member

.....

Prof. Dr. Ergün ERASLAN

Director

Graduate School of Natural and Applied Sciences

ETHICAL DECLARATION

I hereby declare that, in this thesis which has been prepared in accordance with the Thesis Writing Manual of Graduate School of Natural and Applied Sciences,

- All data, information and documents are obtained in the framework of academic and ethical rules,
- All information, documents and assessments are presented in accordance with scientific ethics and morals,
- All the materials that have been utilized are fully cited and referenced,
- No change has been made on the utilized materials,
- All the works presented are original,

and in any contrary case of above statements, I accept to renounce all my legal rights.

Date: **Signature: ……**

Name & Surname: Ahmet Esad TOP

ACKNOWLEDGMENTS

First of all, I want to thank my supervisor Assist. Prof. Dr. Hilal KAYA for supporting me, for her constructive recommendations, knowledge and highly motivation.

I would also like to thank Assist. Prof. Dr. Veysel ALCAN for his valuable contribution.

Finally, thanks to my parents for their support.

2018, 18 December **Ahmet Esad TOP**

CLASSIFICATION OF EEG SIGNALS USING TRANSFER LEARNING ON CONVOLUTIONAL NEURAL NETWORKS VIA SPECTROGRAM

ABSTRACT

Previous studies on classifying Electroencephalography (EEG) sleep data generally use the signal itself. Many of these studies need series of pre-processing operations, manual feature extraction, complex and hard application processes. There is no need for lots of pre-processing stages in Convolutional Neural Networks (CNN) and features can be learned automatically instead of using manually extracted features. Also CNNs have better performance over most of other methods in visual classification.

The study presented in this thesis is based on applying transfer learning with CNNs via spectrogram images, that were obtained by using Short-Time Fourier Transform (STFT), on ISRUC-Sleep dataset (ISRUC) for classifying EEG signals. AlexNet trained with 100 patient subjects from ISRUC was used as pre-trained CNN. For classifying sleep stages, single-channel EEG data, that was taken from 10 healthy people, was used as target domain. To reduce overfitting, an image translation operation was employed and images were augmented horizontally. The main purpose of using transfer learning method in this study is achieving better training duration and accuracy. Applying transfer learning increased the accuracy of the classification by 3.11 percent, when compared to the result of using non-pretrained AlexNet, which was trained from scratch.

When the source domain of transfer learning was ImageNet, the accuracy was decreased by 2.73 percent compared to the result of training from scratch. Results showed that transfer learning increases the accuracy when target and source domains are similar, but it may decrease the accuracy when used on different domains (i.e., ImageNet is an image dataset and ISRUC consists of signals). In addition to the study, scalogram images from Continuous Wavelet Transform (CWT) were also tested.

Keywords: Electroencephalography (EEG), deep learning, convolutional neural network (CNN), transfer learning, spectrogram, scalogram, short-time fourier transform (STFT), continuous wavelet transform (CWT).

EEG SİNYALLERİNİN KONVOLÜSYONEL SİNİR AĞLARINDA SPEKTROGRAM YOLUYLA TRANSFER ÖĞRENİMİ KULLANILARAK SINIFLANDIRILMASI

ÖZ

Elektroensefalografi (EEG) uyku verisini sınıflandırmayla ilgili önceki çalışmalar genellikle sinyalin kendisini kullanmaktadır. Bu çalışmalardan birçoğu bir dizi ön işleme operasyonlarına, manuel özellik çıkarımına, karmaşık ve zor uygulama süreçlerine ihtiyaç duymaktadır. Konvolüsyonel Sinir Ağlarında (CNN) çok sayıda ön işleme aşamasına ihtiyaç yoktur ve el ile ayıklanan özellikleri kullanmak yerine özellikler otomatik olarak öğrenilebilir. Ayrıca CNN'ler görsel sınıflandırmada çoğu diğer yönteme göre daha iyi performansa sahiptirler.

Bu tezde sunulan çalışma, EEG sinyallerinin sınıflandırılması için ISRUC-Sleep veri kümesinde (ISRUC) Kısa-Zamanlı Fourier Dönüşümü (STFT) kullanılarak elde edilen spektrogram görüntüleri aracılığıyla CNN'ler ile transfer öğreniminin uygulanmasına dayanmaktadır. ISRUC'tan 100 hasta denek ile eğitilmiş AlexNet, ön eğitimli CNN olarak kullanılmıştır. Uyku evrelerini sınıflandırmak için, 10 sağlıklı kişiden alınmış tek kanallı EEG verileri, hedef veri alanı olarak kullanılmıştır. Aşırı uyumu azaltmak için bir görüntü kaydırma işlemi kullanılmış ve görüntüler yatayda çoğaltılmıştır. Bu çalışmada transfer öğrenimi yöntemini kullanmanın temel amacı, daha iyi eğitim süresi ve doğruluk elde etmektir. Transfer öğrenimini uygulamak, sıfırdan eğitilmiş olan ön eğitim yapılmamış AlexNet kullanarak elde edilen sonuçla karşılaştırıldığında, sınıflandırmanın doğruluğunu yüzde 3,11 oranında artırdığı görülmüştür.

Transfer öğreniminin kaynak veri alanı ImageNet olduğunda, doğruluk, sıfırdan yapılan eğitimin sonucuna göre yüzde 2,73 oranında azalmıştır. Sonuçlar, transfer öğreniminin hedef ve kaynak veri alanları benzer olduğunda doğruluğu artırdığını, ancak farklı alanlarda kullanıldığında doğruluğu azaltabileceğini göstermiştir (Şöyle ki, ImageNet bir fotoğraf veri kümesidir, ISRUC ise sinyallerden oluşur). Çalışmada ek olarak, Sürekli Dalgacık Dönüşümünün (CWT) skalogram görüntüleri denenmiştir.

Anahtar Kelimeler: Elektroensefalografi (EEG), derin öğrenme, konvolüsyonel sinir ağları (CNN), transfer öğrenimi, spektrogram, skalogram, kısa zamanlı fourier dönüşümü (STFT), sürekli dalgacık dönüşümü (CWT).

CONTENTS

M.Sc. THESIS EXAMINATION RESULT FORM ii
ETHICAL DECLARATION iii
ACKNOWLEDGMENTS iii
ABSTRACT v
ÖZ vi
CONTENTS vii
NOMENCLATURE ix
LIST OF TABLES x
LIST OF FIGURES xi

CHAPTER 1 - INTRODUCTION 1
1.1 Literature Review 2
1.2 Aim of The Study 4
1.3 Thesis Organization 5

CHAPTER 2 - MATERIALS AND METHODS 6
2.1 Machine Learning 6
2.1.1 Unsupervised Learning 8
2.1.2 Supervised Learning 8
2.2 Artificial Neural Networks 10
2.2.1 Backpropagation 14
2.3 Deep Learning 17
2.3.1 Convolutional Neural Network 20
2.3.1.1 Convolutional Layer 22
2.3.1.2 ReLU Layer 27
2.3.1.3 Pooling Layer 27
2.3.1.4 Dropout Layer 29
2.3.1.5 Fully Connected Layer 30
2.3.1.6 Softmax Layer 31
2.4 Transfer Learning 32
2.5 Spectrogram 34
2.6 Scalogram 36
2.7 Datasets 38
2.7.1 ImageNet 38
2.7.2 ISRUC-Sleep 40
2.7.2.1 Electroencephalography 41

CHAPTER 3 - IMPLEMENTATION 44
3.1 Dataset 44

3.2 Network Architecture 48
3.3 Pre-Training 49
3.4 Training 51

CHAPTER 4 - RESULTS 55
4.1 Conclusion 58

REFERENCES 59

CURRICULUM VITAE 71

NOMENCLATURE

Acronyms

AASM	American Academy of Sleep Medicine
AI	Artificial Intelligence
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
DL	Deep Learning
DNN	Deep Neural Network
EEG	Electroencephalography
EMG	Electromyography
FFT	Fast Fourier Transform
GMM	Gaussian Mixture Model
GPU	Graphics Processing Unit
HMM	Hidden Markov Model
ILSRVC	ImageNet Large Scale Visual Recognition Challenge
ISRUC	ISRUC-Sleep dataset
KNN	K-Nearest Neighbors
ML	Machine Learning
MLP	Multilayer Perceptron
NN	Neural Network
PCA	Principal Component Analysis
PSG	Polysomnographic
ReLU	Rectified Linear Units
REM	Rapid eye movement
RNN	Recurrent Neural Network
RoI	Region of Interest
STFT	Short-time Fourier Transform
SVM	Support Vector Machine

LIST OF TABLES

Table 3.1 List of layers 49
Table 4.1 Results of each fold for Spectrograms 56
Table 4.2 Results of each fold for Scalograms 57
Table 4.3 Comparison of the proposed study with previous studies 58

LIST OF FIGURES

Figure 2.1 An example flow of how ML works 6
Figure 2.2 Machine learning methods 7
Figure 2.3 Illustration of an unsupervised learning 8
Figure 2.4 Illustration of a supervised learning 9
Figure 2.5 Ann structure 10
Figure 2.6 Rosenblatt's perceptron 11
Figure 2.7 AND, OR and XOR gates on Cartesian coordinate system that are separated with single line 12
Figure 2.8 Sigmoid function and hyperbolic tangent function 13
Figure 2.9 ANN learning steps 13
Figure 2.10 An illustration of backpropagation 14
Figure 2.11 Comparison of DL and traditional ML 18
Figure 2.12 Deep learning methods 19
Figure 2.13 Comparison of DNN and traditional NN 19
Figure 2.14 Local receptive field that is 5*5 pixels, connected to the next layer's neuron 21
Figure 2.15 Convolutional Neural Network 22
Figure 2.16 5*5 Local Receptive Field of 28*28 image and its corresponding hidden neuron 23
Figure 2.17 5*5 Local Receptive Field of 28*28 image, slid by 1 unit, and its corresponding hidden neuron 23
Figure 2.18 Zero padding of 2 is applied to 28*28 input neurons 24
Figure 2.19 Visualization of the first Convolutional Layer 25
Figure 2.20 Visualization of the second Convolutional Layer 26
Figure 2.21 Visualization of the fifth Convolutional Layer 26
Figure 2.22 ReLU activation function 27
Figure 2.23 Max Pooling of 24*24 convolutional layer 28
Figure 2.24 Max Pooling and Average Pooling 29
Figure 2.25 (a) A neural network without dropout and (b) a neural network after dropout 30
Figure 2.26 Fully connected layer 31

Figure 2.27 Example softmax output 32
Figure 2.28 Transfer learning steps 33
Figure 2.29 A signal and its representative Spectrogram 34
Figure 2.30 Short-time fourier transform steps 35
Figure 2.31 Hamming window for N+1 samples 36
Figure 2.32 A signal and its representation as a scalogram 37
Figure 2.33 Actual images of URLs that present in ImageNet and their labels 38
Figure 2.34 ImageNet Large Scale Visual Recognition Challenge Results as top-5 error rate 39
Figure 2.35 Details of ISRUC-Sleep dataset 40
Figure 2.36 Details of recordings in ISRUC-Sleep dataset 41
Figure 2.37 Example EEG signals of 5 different sleep stages 42
Figure 2.38 (a) An illustration of the human brain & (b) a diagram of the cerebral 42
Figure 2.39 The International 10–20 system seen from (A) left and (B) above the head. A: ear lobe, C: central, Pg: nasopharyngeal, P: parietal, F: frontal, Fp: frontal polar, O: occipital 43
Figure 3.1 Hamming window with the width of 22 samples 45
Figure 3.2 Flowchart of generating a spectrogram 46
Figure 3.3 Generated spectrogram samples for each sleep stage 47
Figure 3.4 Generated scalogram samples for each sleep stage 48
Figure 3.5 Architecture of the CNN trained with 1e-3 learning rate 50
Figure 3.6 Flowchart of training from scratch and saving the network 51
Figure 3.7 Architecture of the CNN trained with 1e-4 learning rate for transferred layers 52
Figure 3.8 Flowchart of training that used transfer learning 53
Figure 3.9 Outline of trainings performed on the study as flowchart 54
Figure 4.1 Accuracy results of 3 experiments 55

CHAPTER 1

INTRODUCTION

Machine Learning (ML) applies some techniques on data to make a computer learnable object without being explicitly programmed [1]. One of the most used techniques in ML is Neural Networks (NN). Neural Networks enable ability to learn from data in computer technology. It is inspired by biological neural networks from human brain.

Convolutional Neural Networks (CNN) have been achieving state of the art results in many areas such as hand-written digit recognition and face detection since its beginning [2] in early 90's [3]. As Graphics Processing Units' (GPU) prices fall and their performances increase by time, Deep Learning (DL) has become extremely popular in recent years. In last few years, a record breaking performance came by AlexNet [4] and then many papers have shown excellent results in visual classification [3,5-9]. After AlexNet was introduced, Artificial Intelligence (AI) society that study on visual classification have focused much more on CNNs and they improved AlexNet's performance by changing some parameters or adding more layers, in other words, making it deeper [3,6,7].

When the data size is low, CNN trained from scratch gives terrible results. At [3], they trained a CNN from scratch with 3840 images of Caltech-256 [10] dataset and got 9.0% accuracy. But when they used same CNN architecture, AlexNet's architecture, but pre-trained with ImageNet [11], they got 65.7% accuracy and beat state-of-the-art results at that time. The transfer learning approach is beneficial for smaller scale datasets. It increased the accuracy because target and source domains were similar (i.e. Caltech-256 is a similar dataset to ImageNet) in this study. When features are different, transfer learning is not much profitable.

Sleep stage has been classified from Electroencephalography (EEG) signal with various methods by using the signal itself in so many papers [12-16]. These methods come with some outcomes and requirements like need for pre-processing and feature extraction processes. On the other hand, there is no need for feature extraction in CNNs. CNNs are more successful over so many techniques in visual classification. There are few examples of using visual representation of signals to classify sleep

stages. Thus, the approach of this thesis, which is converting EEG signals into images that represent features of signal then constructing a CNN and making those images as the inputs of CNN, emerges.

In this study, feature representation of signals in visual format is determined as spectrograms. Spectrograms are generated from signals by using Short-time Fourier Transform (STFT). CNN architecture is AlexNet's architecture and transfer learning approach is applied. A pre-trained AlexNet, that is trained with ImageNet, is used. The dataset, that is used for classification task of the study, contains EEG signals, which were pre-processed. Pre-processing is a need for the dataset, because there are lots of noise in EEG signals due to their nature. There is only translating operation done on spectrogram images while training to make augmentation which helps reducing overfitting.

1.1 Literature Review

This study covers image classification, sleep stage classification, deep learning, convolutional neural networks, and transfer learning topics.

CNNs are so popular and they are getting state of the art results in the field of image classification. LeNet-5 [17] was proposed in 1998 by founding father of CNNs, Yann LeCun, and that had 7-level convolutional network. Since that time, CNNs have similar structure that consists of stacked convolutional layers, pooling layers sprinkled between convolutional layers, and one or more fully connected layers at the end of the stacked convolutional layers.

AlexNet [4] is a deep CNN that won ImageNet Large Scale Visual Recognition Challenge (ILSRVC) in 2012 with 63.3% accuracy [9]. This was a milestone for large scale object recognition. AlexNet has very similar structure to LeNet-5 and contains 5 convolutional layers initially and followed by three fully connected layers. It has simpler design when compared to more complex designs that define state of the art results in image classification [18]. Recent improvements on performance is about increasing the number of layers [5-8], which makes the network deeper, changing the layer size [19] so the filter size such as Zeiler and Fergus decreased the filter size to 7*7 on ZF Net [3] or Simonyan and Zisserman decreased it to 3*3 on VGGNet [6] instead of 11*11 in [4] for large datasets like ImageNet.

For smaller datasets, deep learning is relatively harder to get good results such as Barnes et al. classified Tiny ImageNet and could only get 25.4% accuracy with AlexNet architecture. Then they improved by applying data augmentation, weight decay, and snapshot ensembling on different CNN architectures to get reliable results [18].

Transfer learning is a reasonable choice for small scale datasets if there is somehow similar source domain available. It transfers previously acquired knowledge to a new domain which helps solving new problems faster, better and more efficiently [20]. In [21] and [22], they used other categories as source domain to improve the performance on target categories and to reduce the need for number of training images. Their models were based on Support Vector Machines (SVM).

In [23], they studied to learn visual world weights, that are learned from various datasets excluding their biases and have good generalization ability, then they applied the learned visual world weights to new dataset with its bias that results in dataset specific model. Saenko et al. presented a study about domain adaptation, which they mapped the data from source domain to target domain, for object recognition [24]. In [23] and [24] both papers also used SVMs for classification.

Similar to this study, following papers used AlexNet trained with ImageNet and transfer learning technique to their specific works. In [25], they trained AlexNet with ILSVRC 2012, then extracted features from the trained network and trained SVM and Logistic Regression classifiers. They outperformed state-of-the-art approaches that are using feature engineering. In [26], they trained AlexNet and model of [27] on ImageNet then used these pre-trained networks for emotion recognition. Shin et al. explored 3 different CNNs that are trained with ImageNet [28]. CifarNet [29], GoogLeNet [7], and AlexNet are used as pre-trained networks and performed on 2 specific computer-aided detection problems on that study. Wimmer et al. trained 4 different CNNs including AlexNet on ImageNet dataset. Then classified Celiac Disease by using them [30].

There are several studies on sleep stage classification. In [31] and [32] Deep Belief Network (DBN) based feature extractors are used. Then a classifier algorithm such as Gaussian Mixture Model (GMM) is used by [31], Hidden Markov Model (HMM), K-Nearest Neighbors (KNN), and Support Vector Machine are used all together by

[32] and they got 91.31% accuracy. In [16], they used convolutional layers to extract features and softmax layer to classify multi-channel PSG recordings from MASS dataset [33]. Cui et al. used CNN combined with fine-grained segment [15]. They used multi-channel sleep signals from ISRUC-Sleep dataset (ISRUC) [34] and got 92.2% accuracy. All studies mentioned above used multi-channel signals.

Tsinalis et al. [14] used CNN based classification on single-channel EEG from Sleep-EDF database [35] and got 74% accuracy. Supratak et al. implemented 2 step learning algorithm such as 2 different deep CNNs are utilized to extract features and a bidirectional recurrent neural network (RNN) is used to learn transition rules of 30s of the single-channel EEG signal [36]. They got state of the art results for both MASS (86.2% accuracy) and Sleep-EDF (82% accuracy) datasets.

Vilamala et al. applied transfer learning to Sleep-EDF [37]. Spectral images of each window of EEG was obtained and then trained on a pre-trained VGGNet and they got 86% accuracy. Gharbali et al. used pre-trained AlexNet with ImageNet on scalograms that are obtained from single-channel EEG signals by using Continuous Wavelet Transform (CWT) [38]. They used 10 healthy subjects from ISRUC on the study and got 84%.

1.2 Aim of The Study

DL approaches with so many hidden layer units need powerful hardware resources. They are relatively slow at training, even though powerful resources are in use.

Also they need millions of data to get high accuracies for a number of problems. AlexNet, turning point for DL, was trained with ILSRVC-2012 which has 1.2 million training images. Classified dataset of this study has only 8589 samples totally. Most of the CNN approaches, a method of DL that is used in this study, are successful in computer vision. Dataset that is used in this study consists of signals.

In this study, aim is to classify a smaller EEG signal dataset, collected from healthy subjects during sleep, by using CNN that classifies visually with good accuracy rates. CNNs pre-trained with larger datasets are applied to get higher accuracies or at least seeing faster training time. Another aim is to see the difference between presented

model over the representative CNN model without pre-training in terms of resulting accuracy and number of epochs needed.

1.3 Thesis Organization

This thesis consists of 4 main chapters. Chapter one contains the purpose and the scope of the research. It also has overview of the related works. The second chapter has information about used methods and datasets in this study. The third chapter includes implementation details of the experimental study presented. In the final chapter, results obtained from experiments, comparison and discussion of the results are given.

CHAPTER 2

MATERIALS AND METHODS

In this chapter, subjects related to the thesis are explained in detail. Also datasets that are used in this study are presented.

2.1 Machine Learning

Learning from experiences is a natural thing for humans or animals. Humans captures information from events or observations of facts. When there is a need for that information, e.g. a new event is occurred, gathered knowledge is used for this new event.

Machine learning (ML) is a technique that makes computer learn from experiences. ML is not like a rule based technique. ML methods learn information directly from data by parsing it. Those methods don't have any pre-determined equations or explicitly programmed decisions that finds the solution. These methods basically create the way or learn the equation to the solution by using only data.

ML algorithms find patterns or regularities in data. By using those patterns, it makes decisions, predictions, or determinations. As the number of samples increase, they improve their performance and give better results. Data size is crucial for ML. See Figure 2.1 for the illustration of how ML works.

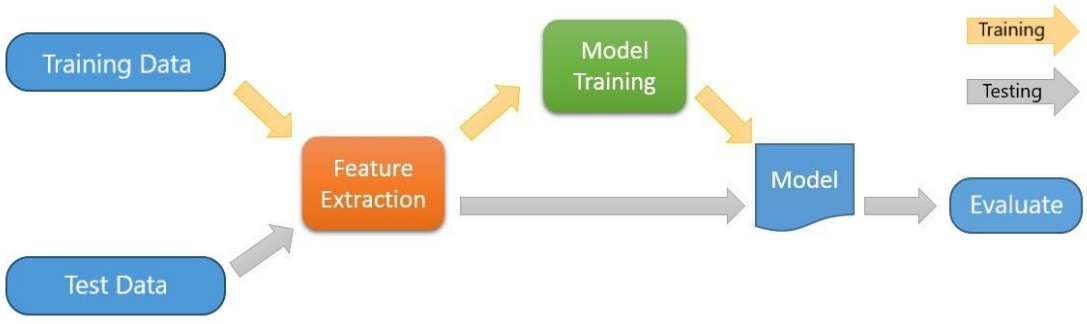


Figure 2.1 An example flow of how ML works [39]

More formal and well posed definition of ML is *"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E."* expressed by Tom M. Mitchell [40].

ML is used in a series of computing tasks when programming explicit algorithms is hard or those algorithms perform poorly. Example applications of ML can be given as network intrusion detection, e-mail filtering, and classification of computer vision.

Among different types of ML tasks, 2 broad categories are the main ones. They are supervised and unsupervised learning (see Figure 2.2). Due to no use of unsupervised learning in this thesis, it will not be provided in detail.

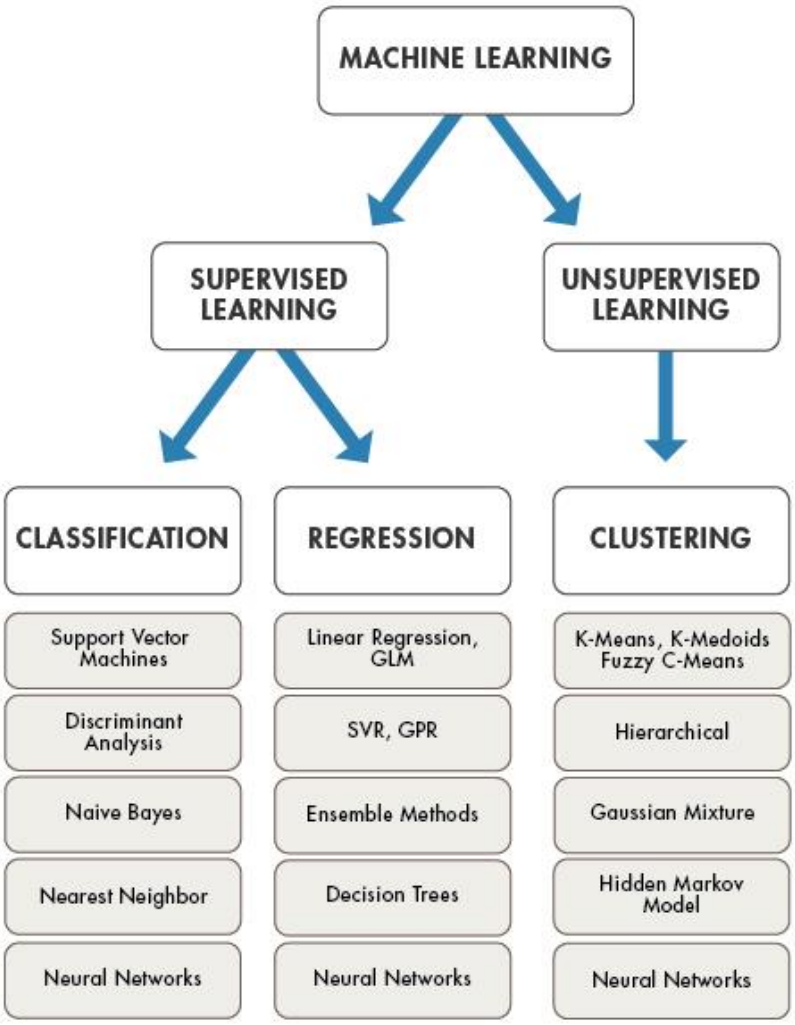


Figure 2.2 Machine learning methods [41]

2.1.1 Unsupervised Learning

Unsupervised learning is a learning type that can find patterns and relations in unlabeled data. There is only input data that doesn't have its corresponding output. It learns underlying structure and distribution in the data to model them. No correct answer or teacher available in this learning type.

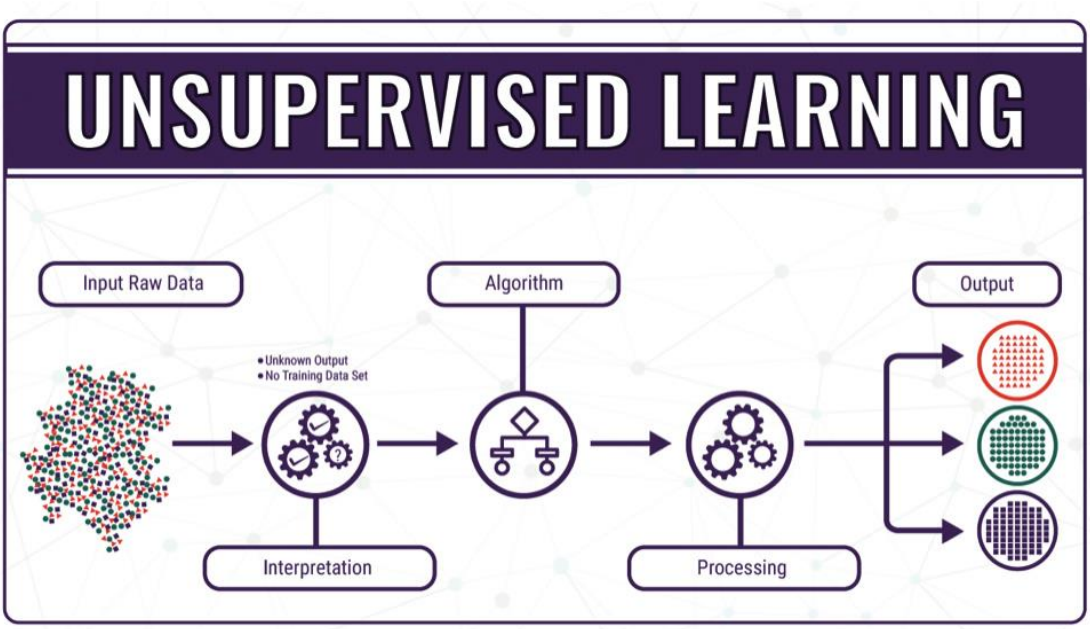


Figure 2.3 Illustration of an unsupervised learning [42]

When similarities, differences, and underlying structure in the data are discovered, they can be grouped up. If the data is grouped according to some rules, that is an association solution. Association detects sets of items that frequently occur together in dataset. If they are grouped according to inherent groupings in the data, that is a clustering solution (see Figure 2.3). Clustering splits the dataset into groups according to similarities. Common unsupervised learning algorithms are K-Means Clustering, Principal Component Analysis (PCA), Hidden Markov Model, and Apriori algorithm.

2.1.2 Supervised Learning

Supervised learning is a ML task that learns from labeled data. It learns a generalized rule of mapping sample inputs to their desired outputs from data. After training with samples, it produces a function that can map new inputs to their unknown outputs. Desired result is that it will map new inputs correctly and find their unseen labels. This is generalization ability of a learning algorithm.

$$Y=f(X) \quad (2.1)$$

where, Y is output, X is input, and f() is the learned mapping function (2.1a)

Data should be in the form of that is paired with input subject and its desired output for supervised learning.

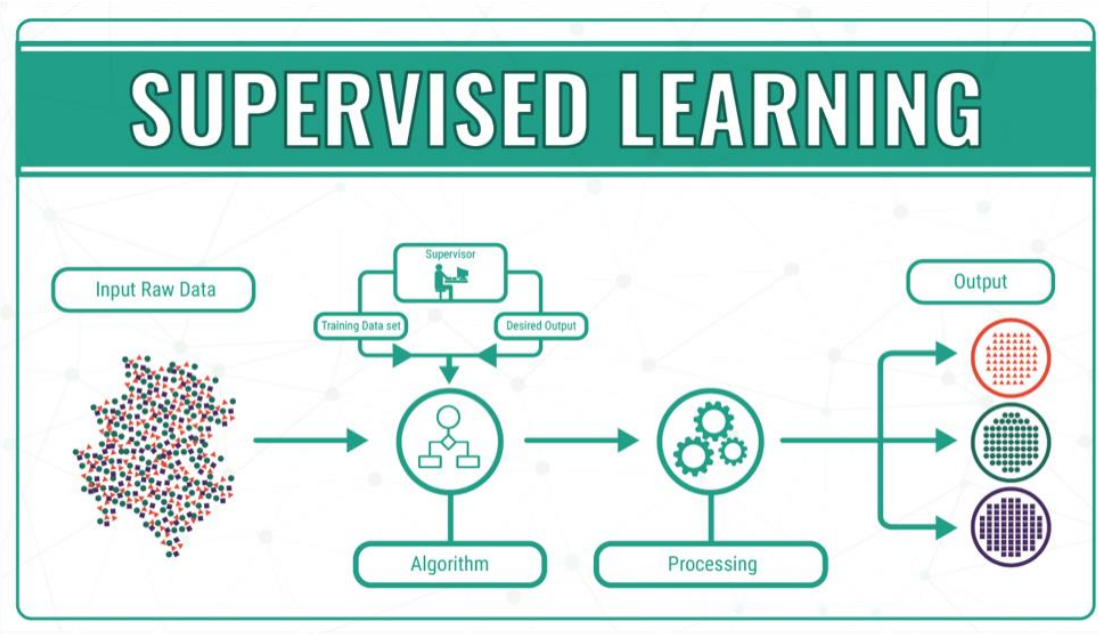


Figure 2.4 Illustration of a supervised learning [42]

Supervised learning uses two techniques: classification and regression. These 2 techniques are used to develop predictive models. An illustration of supervised learning can be found in Figure 2.4.

Regression techniques predict continuous responses. Their outputs are real numbers such as ‘temperature’.

Classification techniques predicts discrete responses. Their outputs are categorical such as ‘black’ or ‘white’. Common classification algorithms are Naïve Bayes, Support Vector Machine, Artificial Neural Networks (ANN), decision trees, logistic regression and KNN.

In this study, Supervised learning technique is used. Labels of this study’s data are categorical as ‘N1’, ‘N2’, ‘N3’, ‘REM’, and ‘WAKE’, which are sleep stages.

2.2 Artificial Neural Networks

Artificial neural networks are one of the most used techniques in ML in both supervised or unsupervised manner. It is sometimes called as Neural Networks. These networks are inspired from biological neural networks. ANN is an information processing system.

ANNs learns how to perform specific tasks from examples. They don't have any prior knowledge, don't have any programmed rules for the task, that is expected to perform by ANN, before training. ANN just gets the sample data, parse it, learn patterns in it and then the network gets the ability to perform the task. For example, the task is to categorize pens from images like 'a pen' or 'not a pen'. ANNs can learn performing that task by using sample images. ANN gets sample image pairs as its training data. Labels of sample data should be arranged by hand and they should be in form of pair. It uses the image as its input, then it gets its output result. Then it calculates the error and see how far it is to the desired output. Then it adjusts itself to get closer to the desired output. After many iterations are done with different samples, its weights are tuned to get the desired output correctly.

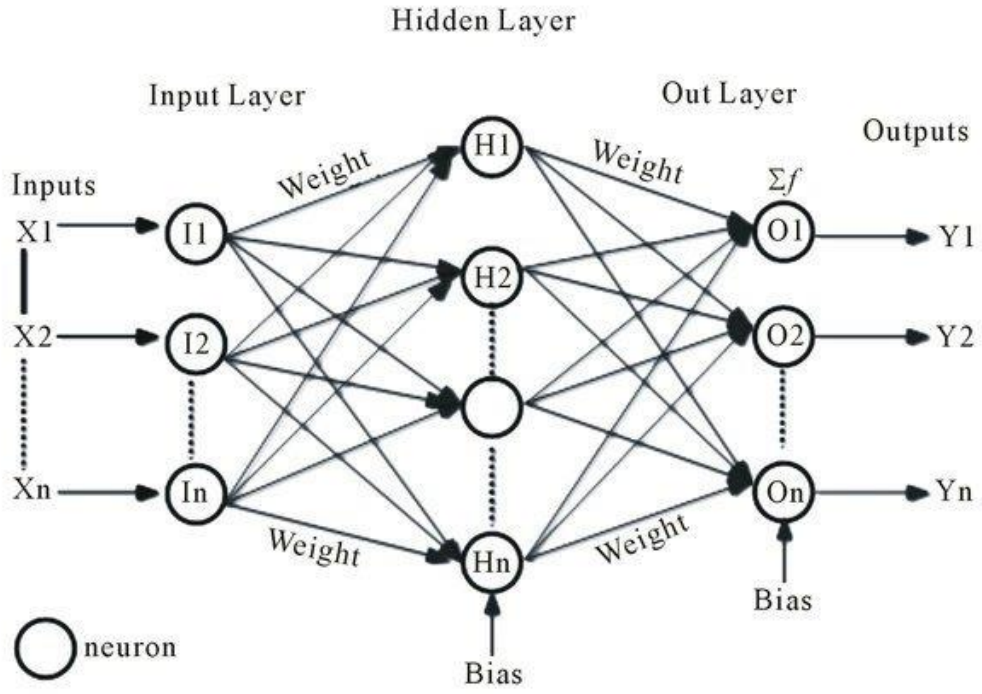


Figure 2.5 Ann structure [43]

ANN is a network that is a collection of nodes. Each node sends information to other nodes that are connected to them. The node, that received the information, can process it and then it sends the new information to another node that is connected to it. These networks consist of either fully connected nodes or not fully connected nodes. An illustration of a NN can be found in Figure 2.5.

Connections between nodes are called ‘edges’ and they can be thought as synapses. Nodes are called ‘artificial neurons’ and they can be thought as biological neurons. Each edge has its weight. Each neuron gives an output of a non-linear function. Input of the function is sum of inputs of that neuron. ANN has layers that are collections of neurons. Each layer may perform different calculations. Every layer waits to start calculations until its previous layer finished all of its calculation.

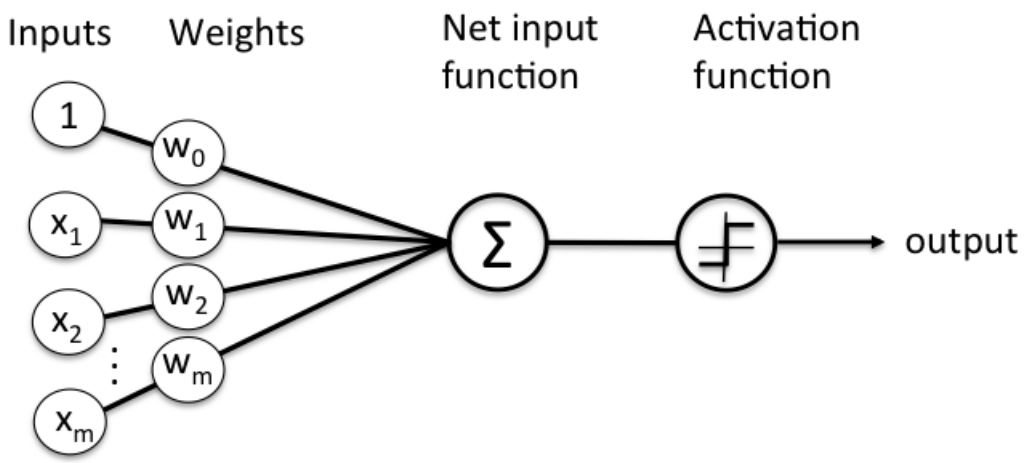


Figure 2.6 Rosenblatt’s perceptron [44]

The first artificial neuron was created in 1943 [45]. In that time, technology was not enough for doing much. Perceptron (see Figure 2.6) was created by Rosenblatt in 1958 [46]. XOR problem couldn’t be solved by Rosenblatt’s single-layer perceptron at that time. In 1975, backpropagation algorithm came and it handled the XOR problem efficiently [47].

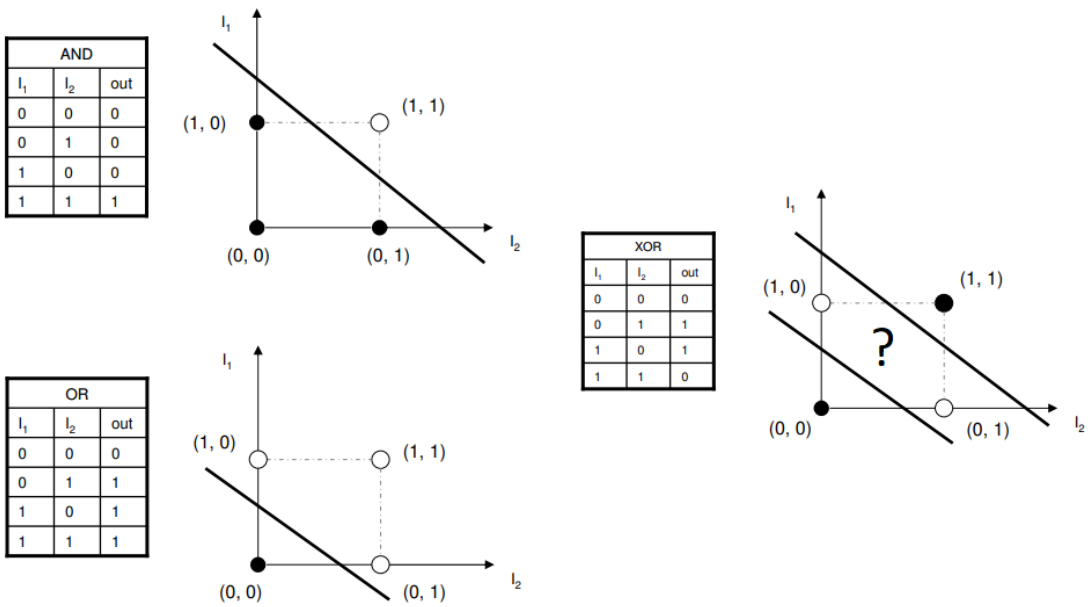


Figure 2.7 AND, OR and XOR gates on Cartesian coordinate system that are separated with single line [48]

'AND' and 'OR' gates can be solved with 1-layer perceptron. But 'XOR' gate can't be made with 1-layer perceptron (excluding the input layer). On Cartesian coordinate system, it is not possible to separate 'XOR' with a single line. 2 lines are needed to separate 'XOR' (see Figure 2.7). Hence, at least 2-layer perceptron (with the input layer 3-layer perceptron) network is needed. When there is one or more layer between input and output, these are called hidden layers and their units are called hidden units. This system is called as 'Multilayer Perceptron (MLP)'. Except the input layer, each node in other layers have a nonlinear activation function, e.g., sigmoid function or hyperbolic tangent as illustrated in Figure 2.8.

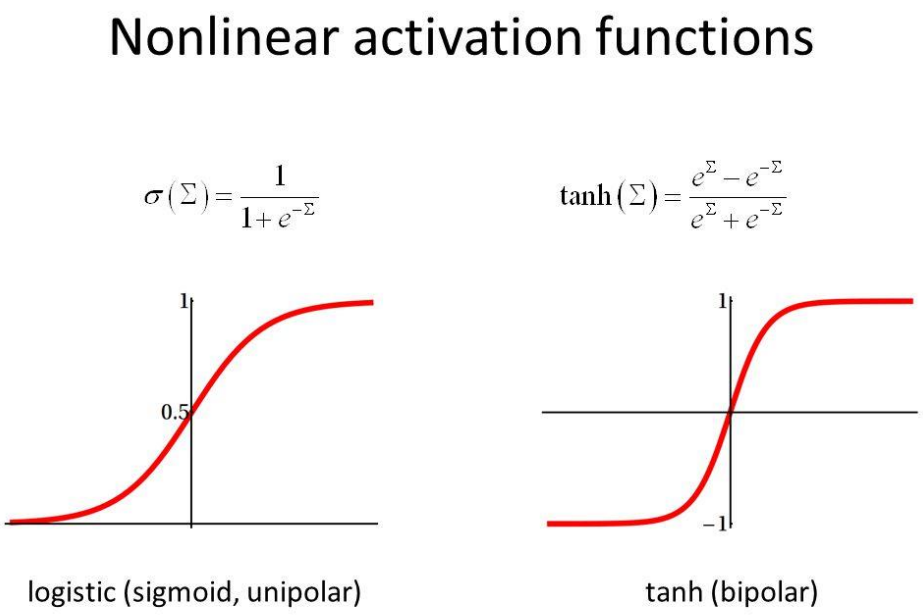


Figure 2.8 Sigmoid function and hyperbolic tangent function [49]

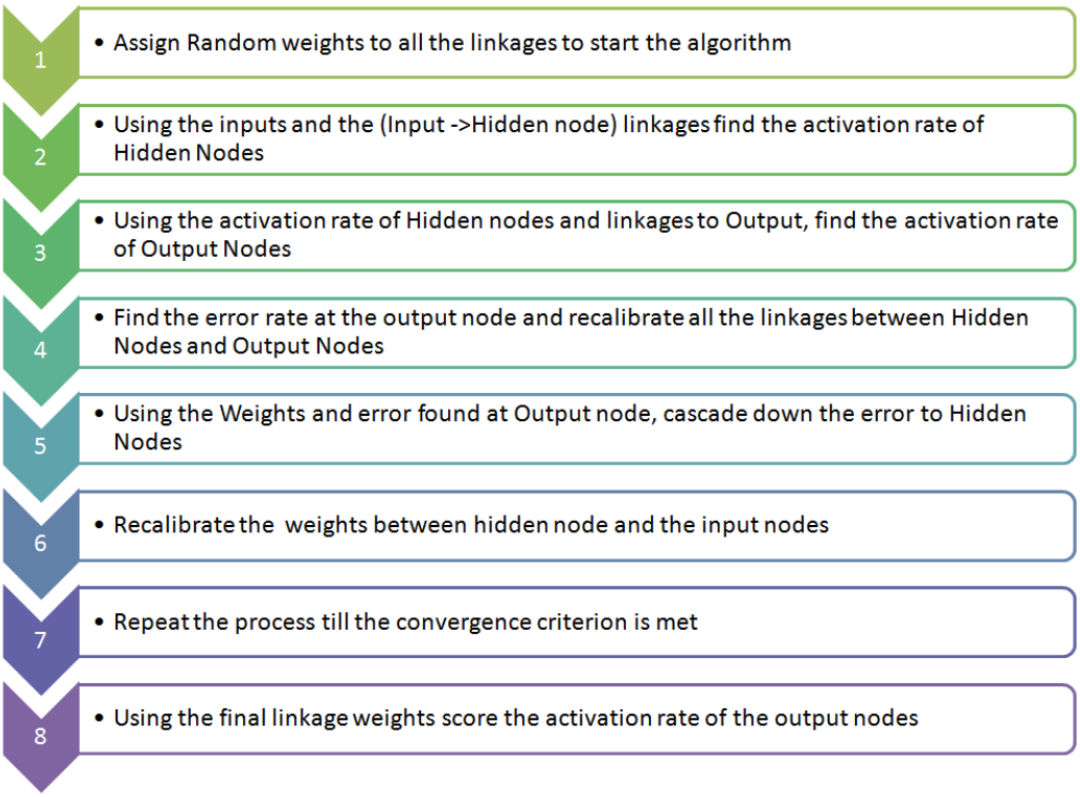


Figure 2.9 ANN learning steps [50]

MLPs use backpropagation technique for its training process. A flow of how an ANN trains is shown in Figure 2.9. ANNs have been used in wide variety of tasks such as speech recognition, computer vision and medical diagnosis. Backpropagation has the key role in that success.

2.2.1 Backpropagation

Backpropagation is used for calculating the gradient to update the weights in ANNs [51]. Backpropagation is commonly used by gradient descent, which calculates gradient of the loss function. The error is distributed to previous layers and neurons that are directly or indirectly connected to the output neuron that the error is calculated on (see Figure 2.10).

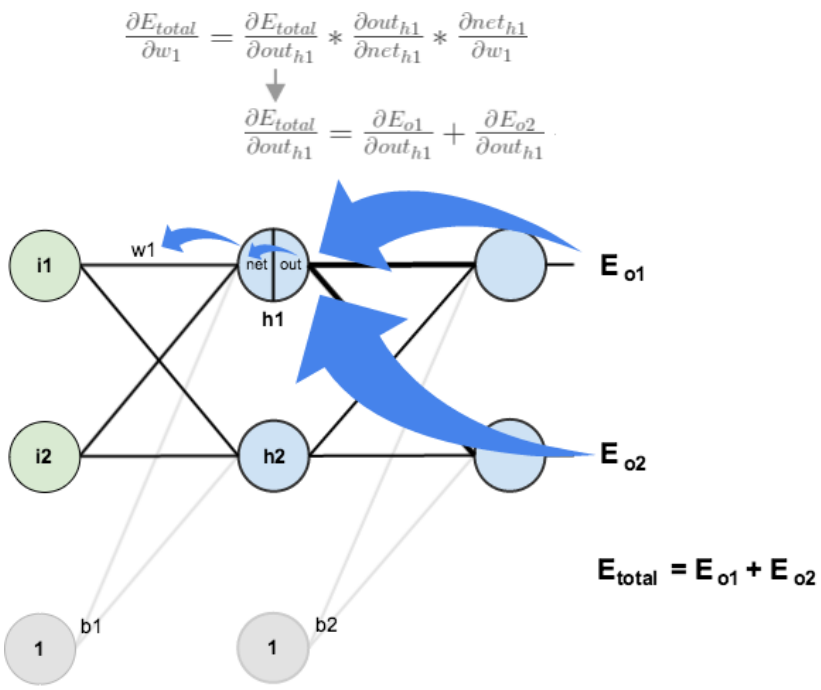


Figure 2.10 An illustration of backpropagation [52]

Through the error calculation, each neuron's net and out values are calculated. Then the Mean squared error of outputs are calculated. These can be expressed as following:

$$net_{h_i} = \sum w_j * input_k + b \quad (2.2)$$

$$out_{h_i} = \frac{1}{1+e^{-net_{h_i}}} \quad (2.2a)$$

Where w_j is each weight connected to h_i, $input_k$ is the input of each w_j

$$net_{o_i} = \sum w_j * out_{h_k} + b \quad (2.2b)$$

$$out_{o_i} = \frac{1}{1+e^{-net_{o_i}}} \quad (2.2c)$$

Where $w_j$is each weight connected to o_i, h_k is the hidden neuron of each w_j

$$E_{o_i} = \frac{1}{2}(target_i - out_{o_i})^2 \quad (2.2d)$$

$$E_{total} = \sum E_{o_i} \quad (2.2e)$$

Gradient Descent is used to minimize the Mean Squared Error cost function. Weights are updated at each iteration until the cost reaches to the minimum. It is as following [40]:

Until the termination condition, Do: (2.3)

-Initialize each Δw_i to zero, (2.3a)

-For each training examples, Do: (2.3b)

-Give input to the unit and compute the output out, (2.3c)

-For each weight w_i, Do: (2.3d)

$$\Delta w_i \leftarrow \Delta w_i + \eta(target - out)input_i \quad (2.3e)$$

-For each weight w_i, Do: (2.3f)

$$w_i \leftarrow w_i + \Delta w_i \quad (2.3g)$$

Where η is the learning rate. (2.3h)

Gradient descent is guaranteed to converge to a hypothesis with minimum squared error if the given learning rate is sufficiently small. If the learning rate is too large, gradient descent has the risk that it can over-step the minimum in the error surface. If

there are multiple local optima in the error surface, then gradient descent may not find the global optimum. Also converging to a local optimum is sometimes slow. There can be found a variation to gradient descent, called stochastic gradient descent, to reduce these difficulties [40]. Stochastic gradient descent and its difference can be seen at the below [40]:

Batch Mode Gradient Descent: (2.4)

Do until satisfied: (2.4a)

1. Compute the gradient $\nabla E_D[\vec{w}]$ (2.4b)

2. $\vec{w} \leftarrow \vec{w} - \eta \nabla E_D[\vec{w}]$ (2.4c)

$$E_D[\vec{w}] \equiv \frac{1}{2}\sum_{d \in D}(t_d - o_d)^2 \quad (2.4d)$$

Stochastic Mode Gradient Descent: (2.4e)

Do until satisfied: (2.4f)

-For each training example *d* in *D* (2.4g)

1. Compute the gradient $\nabla E_d[\vec{w}]$ (2.4h)

2. $\vec{w} \leftarrow \vec{w} - \eta \nabla E_d[\vec{w}]$ (2.4i)

$$E_d[\vec{w}] \equiv \frac{1}{2}(t_d - o_d)^2 \quad (2.4j)$$

Backpropagation algorithm for a feed-forward 2-layer network of sigmoid units (see Figure 2.10) is as following [53]:

Initialize all weights to small random numbers. Until satisfied. (2.5)

1. input the training example to the network, and compute the network outputs (2.5a)

2. for each output unit *k*: (2.5b)

$$\delta_k \leftarrow o_k(1 - o_k)(t_k - o_k) \quad (2.5c)$$

3. for each hidden unit *h*: (2.5d)

$$\delta_h \leftarrow o_h(1 - o_h)\sum_{k \in outputs} w_{kh}\delta_k \quad (2.5e)$$

4. update each network weight w_{ji}: (2.5f)

$$w_{ji} \leftarrow w_{ji} + \Delta w_{ji} \quad (2.5g)$$

where $\Delta w_{ji} = \eta \delta_j x_{ji}$, and x_{ji} is the i^{th} input to unit j. (2.5h)

2.3 Deep Learning

Deep learning (DL) is a specialized form of ML. DL is just another way to describe large NNs. As the name 'deep' implies, its learning process is happening in deeper structures. While saying deeper structures, having so many number of hidden layers is meant. Traditional NNs have only 1 or 2 hidden layers, while deep networks might have dozens or hundreds of hidden layers [7,8]. DL can be in either supervised or unsupervised manner.

DL methods use so many layers that have nonlinear processing units. These layers basically extract features and make transformations. Similar to NN, every layer uses outputs of its predecessor layer as its inputs. Every layer has its representation of abstraction.

Each layer transforms its input and make a more abstract representation. Changing the number of layers and their sizes produce different degrees of abstraction.

DL in supervised manner, don't need manual feature extraction by translating the data into intermediate feature representations. DL models are trained with large sets of data and methods learn features directly from the data.

Other than DL, ML approaches need feature engineering by hand or need some techniques act as feature extractors (see Figure 2.11). In DL, feature extraction and modeling steps are done automatically. Unlike other ML methods, DL approaches don't need any pre-processing or need so little pre-processing.

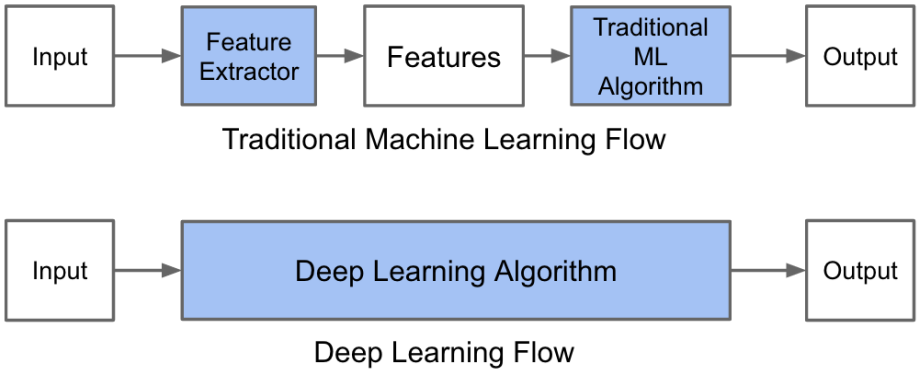


Figure 2.11 Comparison of DL and traditional ML [54]

There is a notable advantage of DL, that it keeps getting better performance as the size of the data increases. With the improvement of technology, enormous amount of data and powerful GPUs are available recently. This situation has made DL so popular recently. When having small size of data or powerless computing environment, choosing one of the other ML approaches rather than DL makes more sense.

Recent studies in DL have brought huge success to society, where DL approaches outperform humans in some tasks like classifying objects or playing board games [4,55,56].

'Deep Learning' term came out to AI community in 1986 by Rina Dechter [57]. In 1989, Yann LeCun et al. used backpropagation algorithm to a DNN, which recognizes handwritten ZIP codes on mail [2]. ANNs weren't a popular choice against SVMs or GMM-HMM due to computational cost of ANN. DL drew attention again after advances in hardware. In 2009, Nvidia used its own GPUs to train Deep-Learning NNs. That was called 'Big Bang of A.I.' [58].

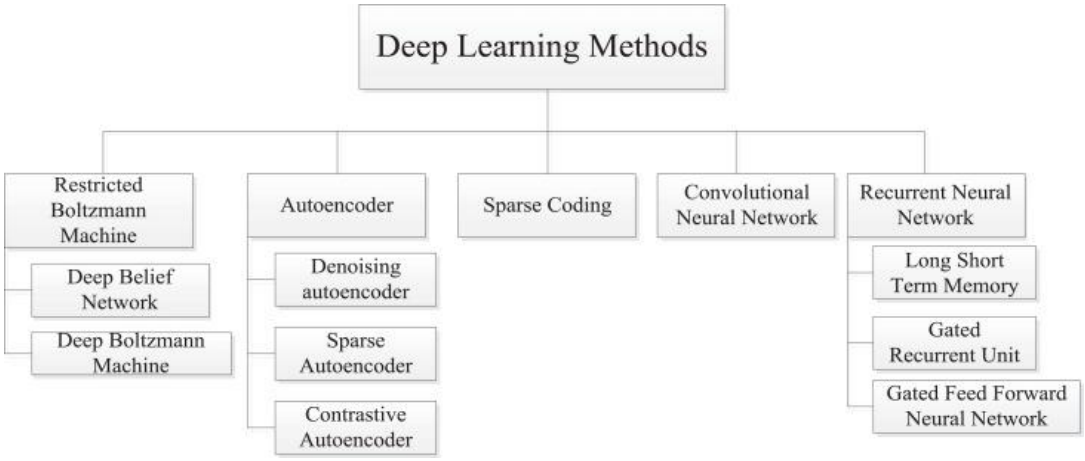


Figure 2.12 Deep learning methods [59]

Most of the modern DL approaches are based on ANNs such as Convolutional Neural Network, Recurrent Neural Networks (RNN), and Deep Belief Networks (DBN). Almost all of the methods in Figure 2.12 are based on NNs. This is why DL models are often referred to as Deep Neural Networks (DNN). These methods have been applied to wide variety of areas such as computer vision, audio recognition, speech recognition, medical diagnosis, military, fraud detection and many other fields.

DNN is an ANN with multiple hidden layers between the input and output layers (see Figure 2.13) [60]. DNNs can model linear or non-linear relationships even the relations are complex. Having extra layers in hidden layers, make possible composition of features from lower layers. This is potentially reducing the required units in hidden layers for modeling complex data [61]. DNNs are feedforward networks that data streams from the input layer to the output layer.

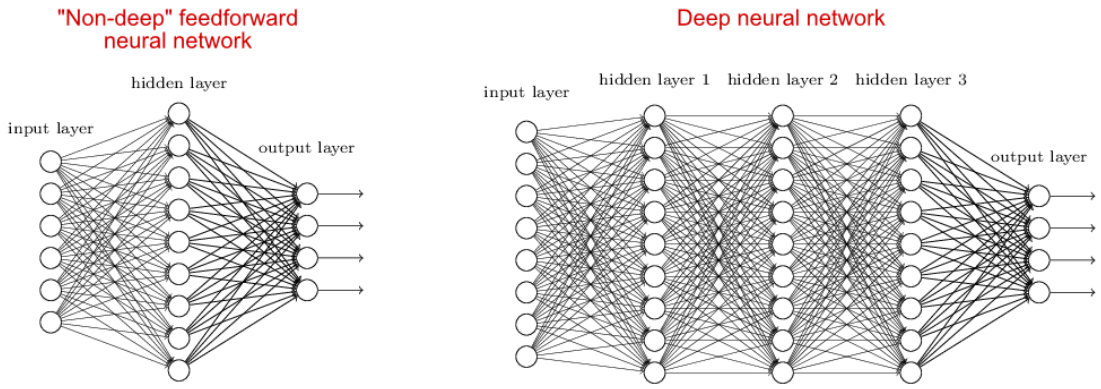


Figure 2.13 Comparison of DNN and traditional NN [62]

Training deep neural networks in supervised manner in a decent way is generally too difficult. There is one remarkable exception to this situation which is Convolutional Neural Networks [61].

2.3.1 Convolutional Neural Network

One of the most influential and popular type of DNNs is convolutional neural network. It is generally applied to computer vision such as image/video recognition and clustering them.

In 1980, neocognitron was introduced as a hierarchical, multilayered ANN by Kunihiko Fukushima [63]. It has been used for various pattern recognition tasks. It is considered as the inspiration of convolutional neural networks. Neocognitron was inspired by the model of Hubel & Wiesel that was proposed in 1959 [64].

In 1989, LeCun et al. applied backpropagation to learn convolution kernel coefficients from images [2]. Images were handwritten numbers of ZIP codes; task was recognition of them. The system learned coefficients directly from images automatically and performed better than manual coefficient design.

Yann LeCun, founding father of convolutional networks, et al. proposed 7-level convolutional network, LeNet-5, in 1998 [17]. That study was classifying digits on 32x32 images. When there is a need to process larger sized images, network should have more and larger layers. This was the limitation at that time due to lack of computation power.

After having powerful computation resources, CNN has become extremely popular. Several papers applied and described efficient ways to train CNNs using GPUs [4,55,66-69].

A CNN consists of input, output and multiple hidden layers. It is a feedforward DNN. CNNs are designed to minimize the need for pre-processing. They need less pre-processing when compared to other image classification algorithms. Especially manual feature engineering is not needed in a CNN like other DL algorithms. What make CNNs so popular is due to no need for manual feature extraction, state of the art results and re-trainability.

When there is a picture of a dog, humans can perceive what makes a dog a dog from its identifiable features such as paws or 4 legs. In similar way, CNN is able to classify it by looking the low level features such as curves and edges, and then building up to more abstract concepts through a series of convolutional layers [70]. While CNN does that, features are not supplied to the CNN, on the contrary, it automatically captures the needed information of features.

Successful image recognition models, that are based on traditional MLP, suffer from not scaling well to higher resolution images, which is called 'curse of dimensionality'. The reason behind this is full connectivity between nodes. For example, a fully connected layer of a 28*28 input image needs 28*28=784 weights and when the resolution increases, it needs 227*227=51529 weights. Number of needed weights increases exponentially and almost for every case MLPs need much higher number of weights than a CNN does. However, for a convolutional layer in a CNN, there is much less need for weights. For instance, a 7*7 filter that convolves on a 28*28 or a 227*227 image regardless of image size, the requirement is 49 learnable parameters for the filter. Also, shared weights are in use for those filters.

CNNs have 3 dimensional neurons in their layers. Every neuron is connected to a small region of previous layer, called receptive field (see Figure 2.14). Hence, there isn't that much weight needed as in traditional MLP system.

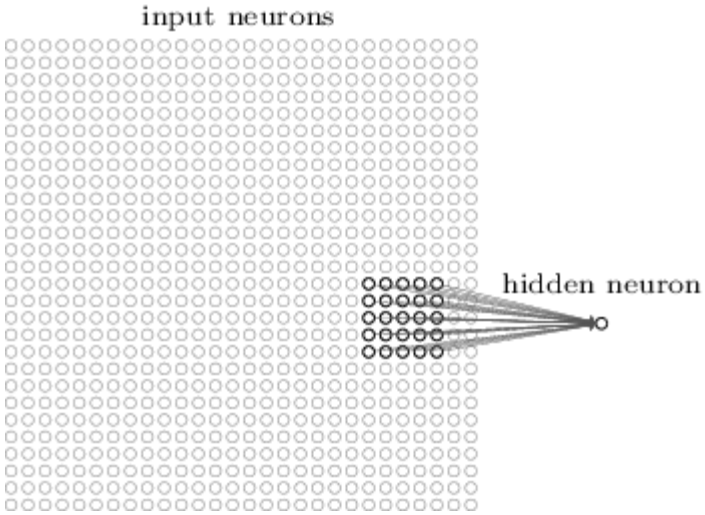


Figure 2.14 Local receptive field that is 5*5 pixels, connected to the next layer's neuron [62]

Unlike traditional ANNs or DNNs, hidden layers of CNNs have some different operations. These hidden layers generally consist of convolutional layers, rectified

linear units (ReLU), pooling layers, fully connected layers, dropout layers and softmax layer. While the convolutional layers are the main layers, others are sprinkled between these key convolutional layers except fully connected layers at the final part. These layers provide nonlinearity and preservation of dimensions. An example CNN that classifies the input image as it finds which type of tired vehicle is the input, is shown in the Figure 2.15.

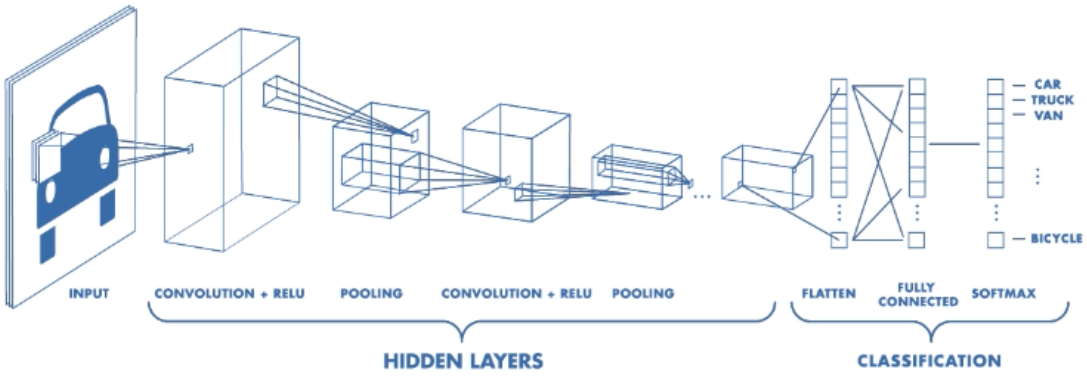


Figure 2.15 Convolutional Neural Network [71]

2.3.1.1 Convolutional Layer

Convolutional layer is the heart of a CNN. The first layer of a CNN is always a convolutional layer. They are located many times from the beginning to the ending of the network. A bunch of learnable filters are the parameters of this layer. These filters match with their corresponding receptive fields. (i.e. small region of previous layer) Array of numbers in a rectangle shape is the filter. They are working as feature identifiers.

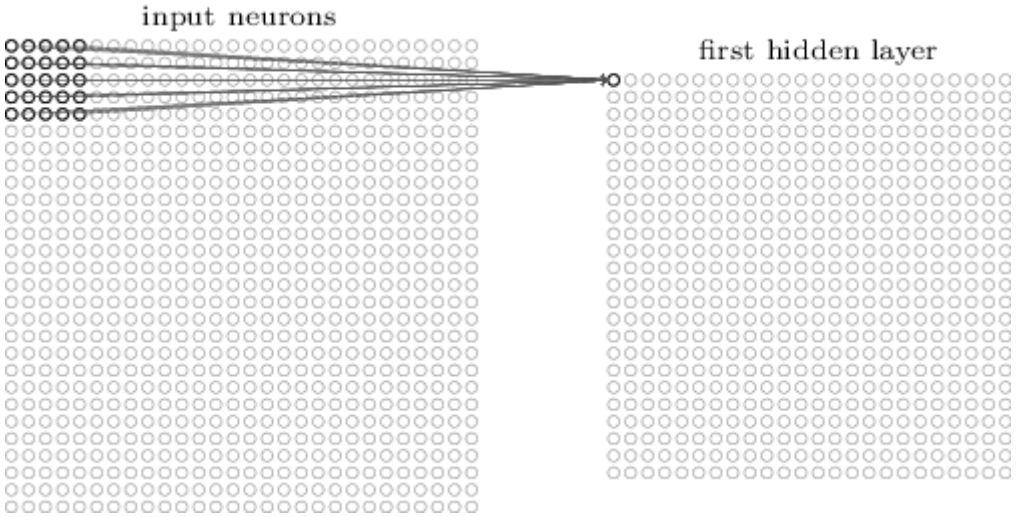


Figure 2.16 5*5 Local Receptive Field of 28*28 image and its corresponding hidden neuron [62]

In the Figure 2.16 and Figure 2.17, there is a 5*5 filter that is convolving 1 by 1 across the width and height of the input. First, it is slid across width. When there is no pixel left in width, filter goes to one pixel down vertically and the first pixel of horizontal line, then again starts sliding across the width. It is repeated for every location. While it is convolving 1 by 1, at least 24*24 (i.e. (28-5)/1+1=24) hidden neurons are needed that they are connected to. In this case, for each hidden neuron, there is a dot product computed between the entries in its corresponding 5*5 filter and the input. The result is an activation map of the filter.

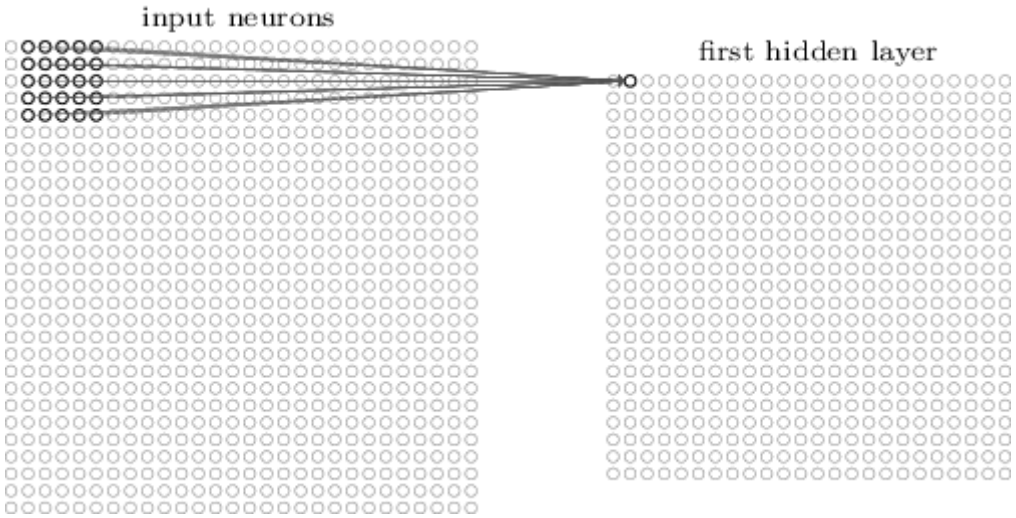


Figure 2.17 5*5 Local Receptive Field of 28*28 image, slid by 1 unit, and its corresponding hidden neuron [62]

These figures show the case that the stride length is 1 and the filter size is 5*5. Stride defines how the filter slides on the input. If stride length is different, then the filter moves not 1 by 1 and there will be a different number of hidden neurons. For example, if the stride length is 3 and the filter size is 7*7, then there will be 8*8 hidden neurons for 28*28 input image.

Keep in mind, if an RGB image is used as the input, the input layer size will be 28*28*3, the filter size will be 7*7*3, and the first hidden layer size will be 1*8*8. If there are 2 filters instead of 1, then the first hidden layer's volume will be 2*8*8.

There is another parameter in convolutional layer which is padding. Padding increases the size of input volume while filters are convolving. The case that zero padding is used, it pads the input volume with zeros around the border. E.g., if zero padding of size '2' is applied to the layer, which has 28*28 input neurons and stride length of 1 with 5*5 filters are applied on the input, then its input size will increase to 32*32, as illustrated in Figure 2.18, and it gives the output as 28*28 (i.e. that would be 24*24, if zero padding weren't applied). As it can be seen, the actual input size is 28*28 and first convolutional layer's size is remained same as the input size, 28*28.

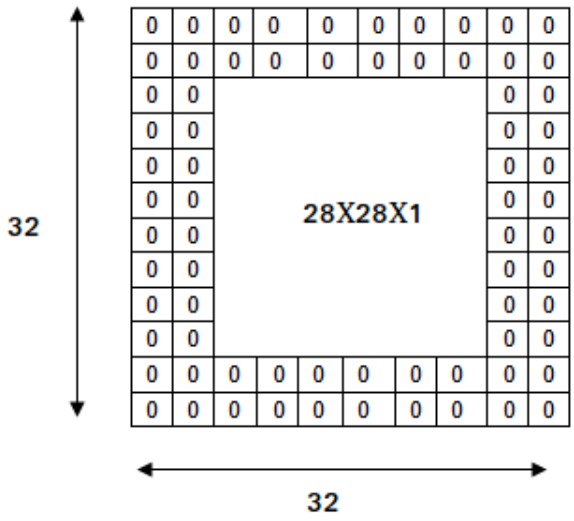


Figure 2.18 Zero padding of 2 is applied to 28*28 input neurons [72]

When filters detect some specific type of feature in input, they are activated. These operations lead the network learns activation maps of each filter. Hence, the network learns how to detect features. Initial convolutional layers tend to learn detecting low

level features (e.g. features like curves, edges, corners, colors, etc.), while later convolutional layers tend to learn detecting high level features. The visualization of the first, the second and the fifth convolutional layer of a CNN shown in Figure 2.19, Figure 2.20 and Figure 2.21 respectively. As it can be seen, the first convolutional layer consists of low level features like colors and edges which represent filters at the layer. As going deeper and deeper, convolutional layers has more detailed filters. (i.e. after the first layer through the middle layers, they start to detect features like curves, corners, and combinations of edges present on the object, in the last layers, they might start to detect big part of the whole object)

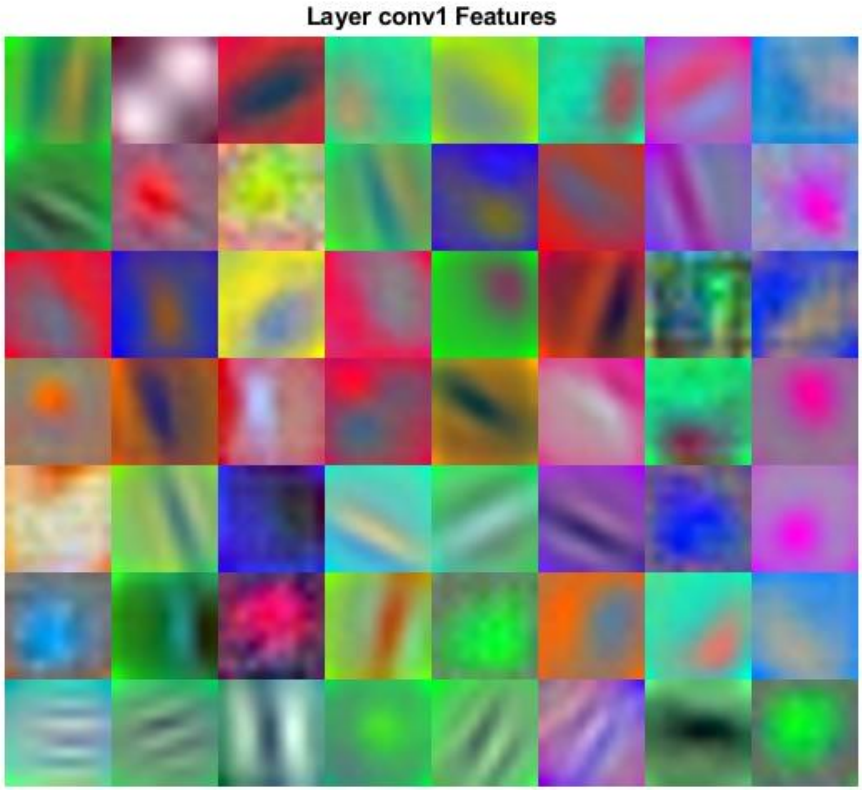


Figure 2.19 Visualization of the first Convolutional Layer [73]

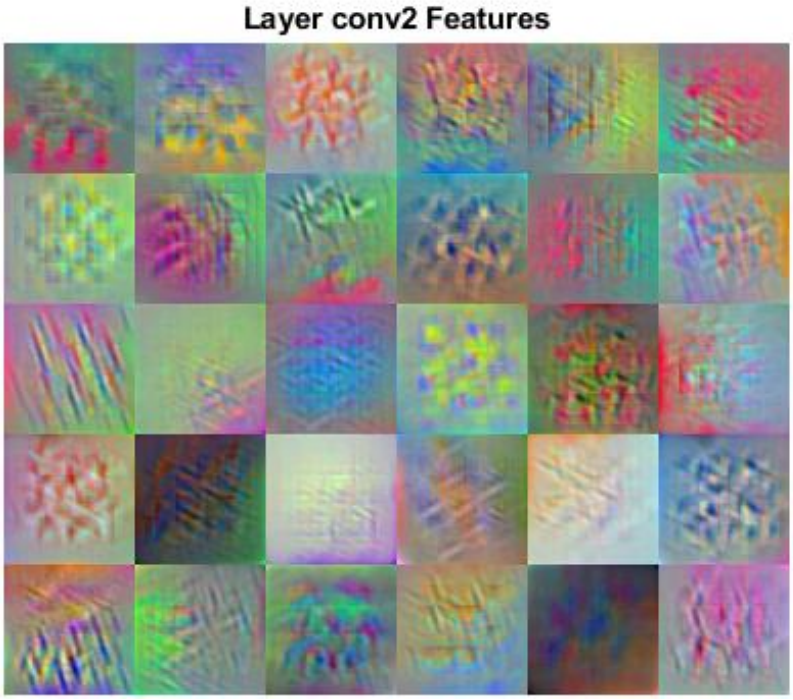


Figure 2.20 Visualization of the second Convolutional Layer [73]

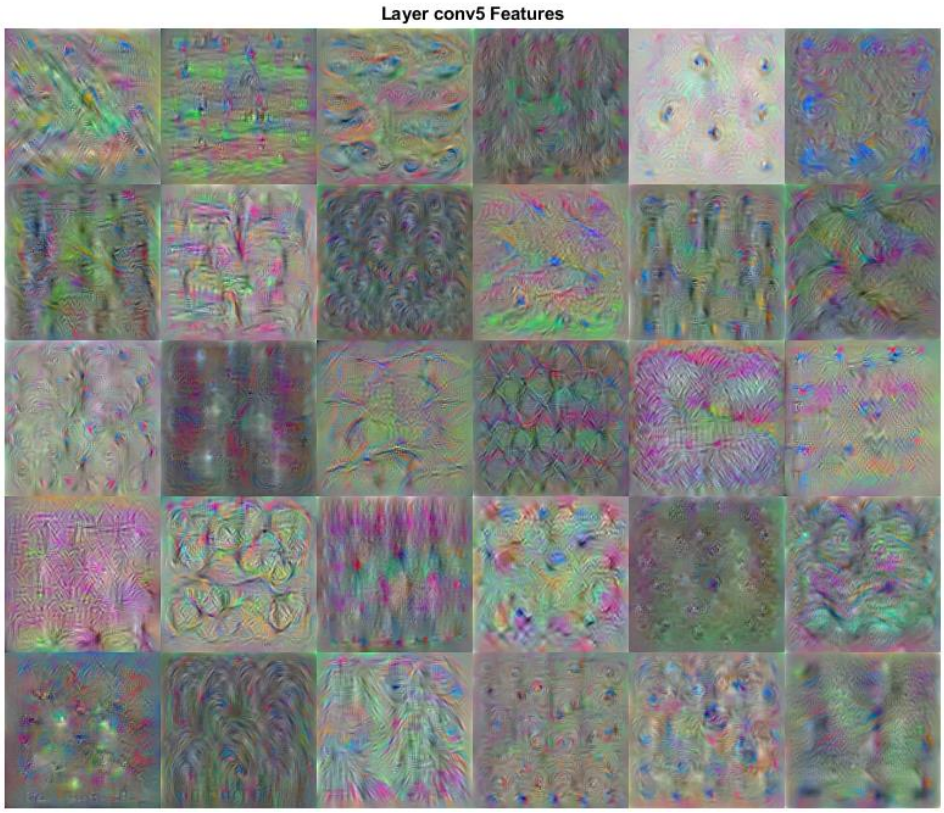


Figure 2.21 Visualization of the fifth Convolutional Layer [73]

2.3.1.2 ReLU Layer

After each convolutional layer, it is so common that there is a ReLU layer used immediately. ReLU stands for Rectified Linear Units. It applies non-linear activation function similar to sigmoid or hyperbolic tangent.

Convolutional layers just compute linear operations like multiplication and summation. ReLU brings non-linearity to the network. Sigmoid function or hyperbolic tangent can also be used to get non-linearity. ReLU's advantage is, it is computationally efficient. It trains much faster than other activation functions [4]. ReLU increases non-linearity without affecting the receptive fields and without making a major difference to the accuracy. It also helps to prevent vanishing gradient problem, where earlier layers train slowly. ReLU function turns all the negative activations to zero as illustrated in Figure 2.22.

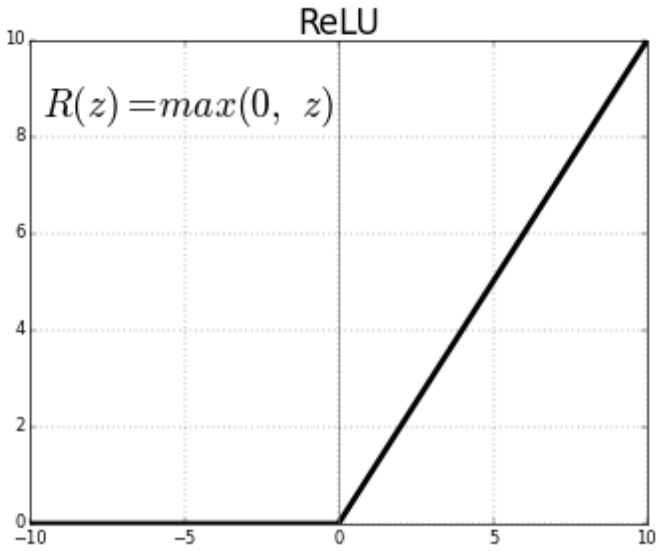


Figure 2.22 ReLU activation function [74]

2.3.1.3 Pooling Layer

In traditional CNN architecture, there are pooling layers used after most of the convolutional layers. (i.e. generally after Convolutional layer + ReLU layer) They are important concept for CNNs.

CNNs use pooling layers to reduce the spatial size of the representation, i.e., down sampling the data, by combining the output of convolutional layers into a single neuron

using non-linear functions. It simplifies the information in the output from the convolutional layer. It reduces the number of parameters that results reducing computational cost. It also helps to prevent overfitting problem. The term overfitting happens when a model becomes so familiar to the training samples and perform perfectly on the training examples but, the model can't be generalized well for samples other than training set and perform poorly on new samples.

Most common non-linear function in pooling operation is max pooling. It divides the input into partitions according to the filter size and stride length of it, then in each partition, it outputs the maximum pixel value of that partition. There are also other pooling layer options such as L2-norm pooling and average pooling. Average pooling calculates the average in each partition and outputs the average value.

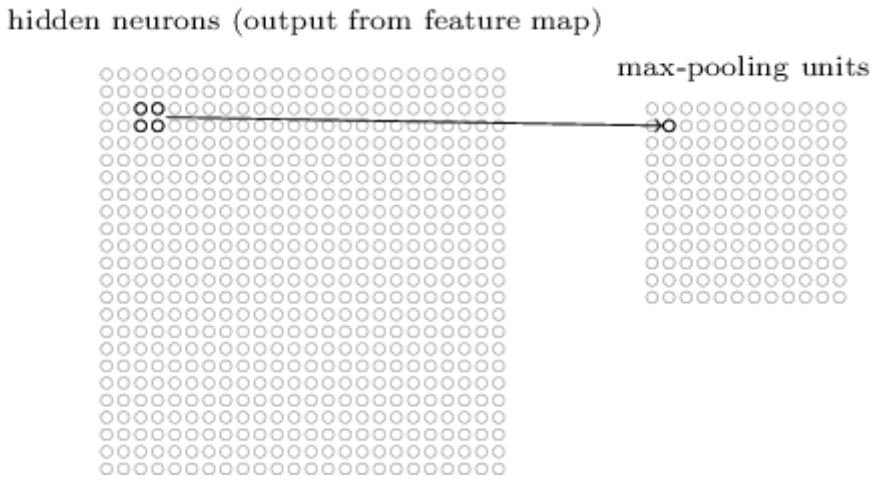


Figure 2.23 Max Pooling of 24*24 convolutional layer [62]

In Figure 2.23, the filter size is 2*2 and the stride length is 2. This is the most common form of pooling layers. Due to the filter size and the stride length, there will be 12*12 pooling layer. (i.e. (24-2)/2+1=12)

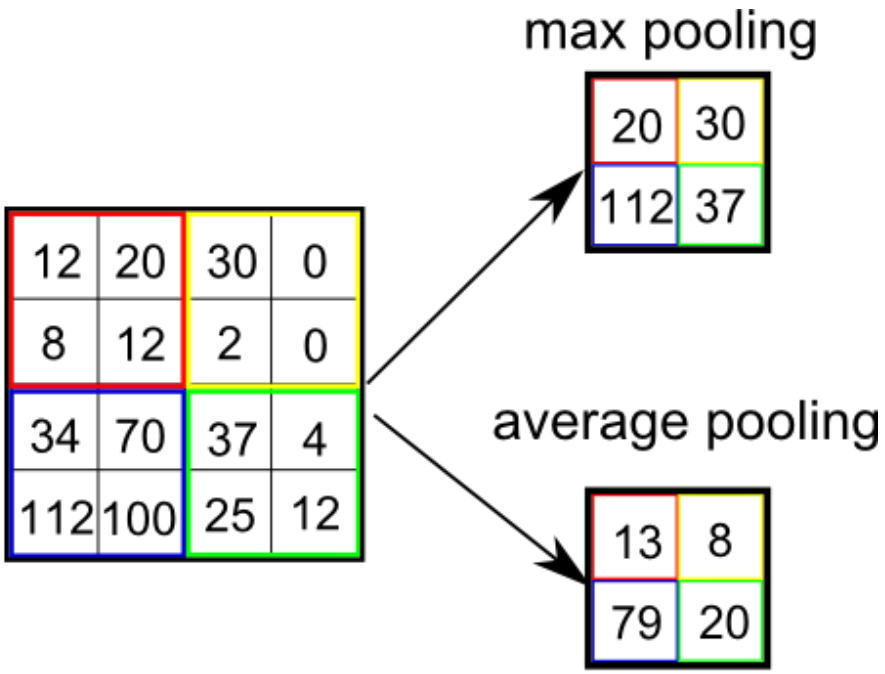


Figure 2.24 Max Pooling and Average Pooling [75]

Max pooling and average pooling operations are illustrated in Figure 2.24. There is an output of convolutional layer whose size is 4*4. The filter size is 2*2 and the stride length is 2. So size of the pooling layer is 2*2. (i.e. (4-2)/2+1=2) Max pooling for the first rectangle colored as red at the top left, '20' is the greatest number in that partition. So, '20' is the resulting value of that filter. For average pooling on the same rectangle, average of the 4 number is 13. (i.e. (12+20+8+12)/4=13)

For object detection tasks, a variant of max pooling, region of interest (RoI) pooling is widely used. It has a region proposal and fixed output size (e.g. 2*2). It divides the region into equal sized sections according to output size and finds the maximum values of each section. Then outputs these maximum values.

In both Figure 2.23 and Figure 2.24, 75% of the activations are discarded i.e., down-sampled by 2 in both height and width.

2.3.1.4 Dropout Layer

Dropout layer is simply drops out some of the activations in the layer according to a probability. It sets all of the randomly chosen activations to zero. Generally, the probability of a neuron is dropped out, ***p***, is chosen as 0.5, so hidden neurons are

dropped out randomly for that ***p*** value. In and out edges of the dropped neuron are all removed. When training process is finished, all dropped neurons are connected back with their weights. An illustration of dropout for standard NN is illustrated on Figure 2.25.

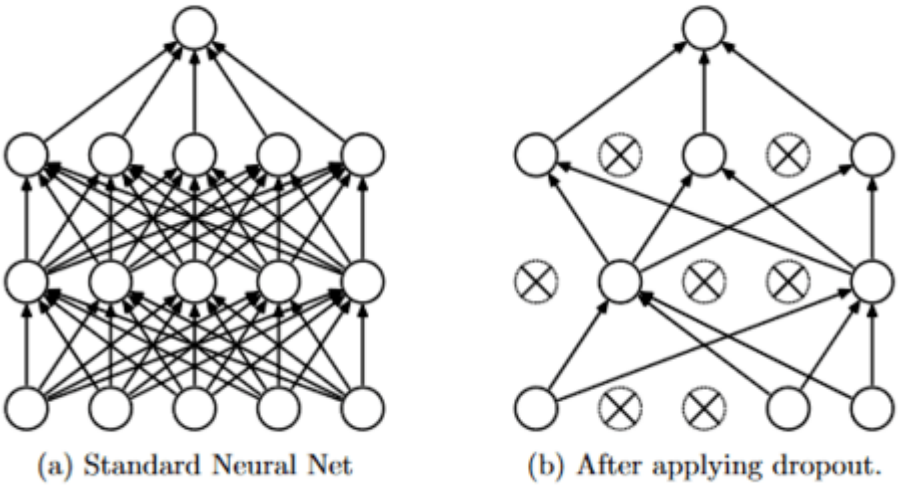


Figure 2.25 (a) A neural network without dropout and (b) a neural network after dropout [76]

After the dropout operation is done, the reduced network should be able to perform well on the task. It makes the network more generalized for the new samples other than training samples, so it makes sure the network is not so tuned to training set. In short, it helps reducing overfitting.

2.3.1.5 Fully Connected Layer

Fully connected layers are located at the final part of CNNs. After a series of convolutional and pooling layers, high level feature activation maps are occurred. The judgement of them is done by fully connected layers. Fully connected layer learns non-linear combinations of these high level features.

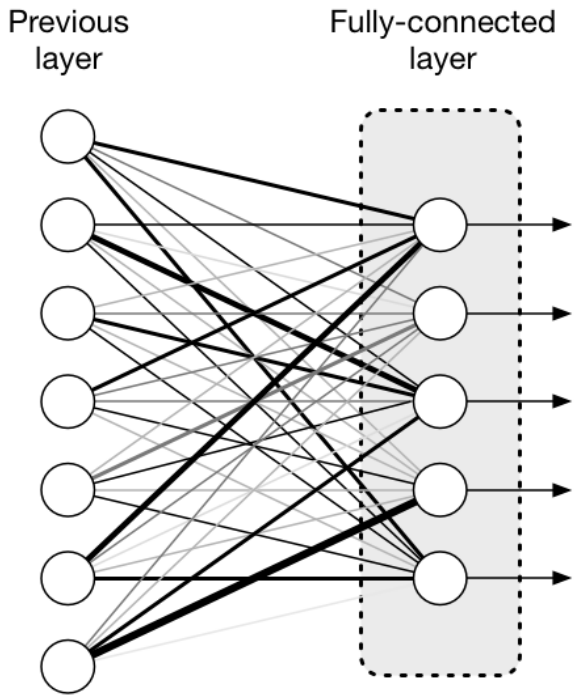


Figure 2.26 Fully connected layer [77]

In this layer, every neuron is connected to all activations of the previous layer as illustrated in Figure 2.26. This situation is similar to traditional MLP neural networks. The fully connected layer keeps combined and mixed information from all of the convolutional layers that matters the most.

2.3.1.6 Softmax Layer

Softmax layer is used at the end as the last layer of a CNN. It performs multi class classification. ‘Softmax’ name comes from softmax function, which is a probability distribution function. An illustration of softmax layer’s output, that takes output of last layer at the forward pass as its input and gave probabilities of 3 classes that they are likely belong, is in Figure 2.27.

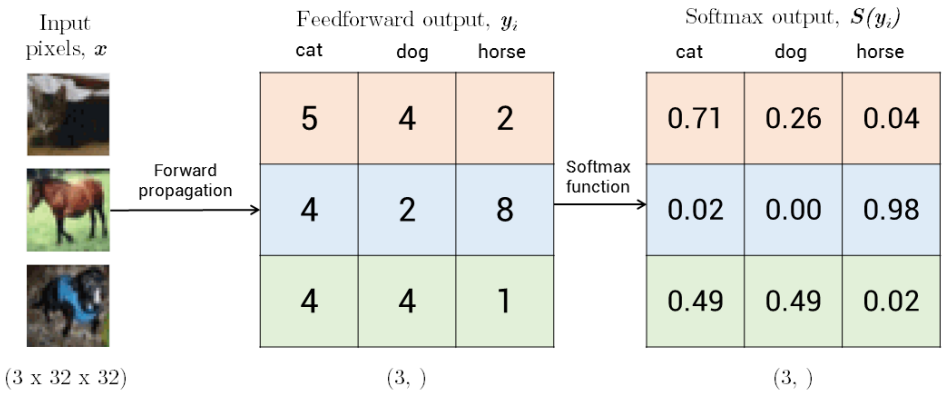


Figure 2.27 Example softmax output [78]

Softmax function takes a number of score values as input and squashes them into values in the range of 0 and 1. Sum of all values it squashed is 1.

Input of the function: 'z_j', where $j=1,\ldots, K$ (2.6)

Softmax Function: $$\sigma(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$ (2.6a)

After the softmax layer, forward pass is finished and the network has an error that is calculated as comparing predicted values and target values. Stochastic gradient descent is the most common algorithm that is used for updating weights and biases in CNNs.

2.4 Transfer Learning

Transfer learning is a machine learning technique that transfers previously acquired knowledge to a new domain. The new domain should be somehow similar to the previous one. Its motivation is solving new problems faster, better and more efficiently [20].

Transferring knowledge between tasks is in nature of humans and animals. For machine learning area, transferring knowledge between neural networks is applied in 1993 [79].

The definition of transfer learning at [20] is quite illustrative and satisfactory. It was *"Given a source domain D_s and learning task T_s, a target domain D_T and learning*

task T_T, transfer learning aims to help improve the learning of the target predictive function $f_T(\cdot)$ in D_T using the knowledge in D_S and T_S, where $D_S \neq D_T$, or $T_S \neq T_T$".

For CNNs, transfer learning can be applied easily. What to do is, transferring activation maps of the trained network. So, when they are transferred, the architecture of the network comes with weights and biases naturally. Transfer learning steps are simply illustrated in Figure 2.28.

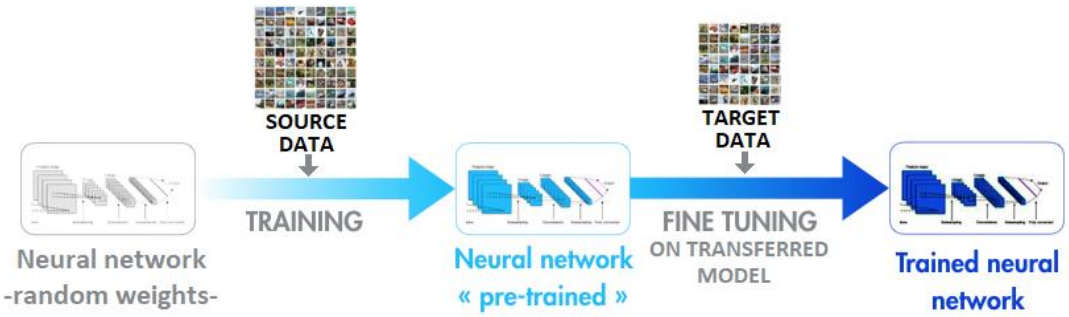


Figure 2.28 Transfer learning steps [80]

When transfer learning is applied, transferred layers' weights and biases are frozen or they have so small learning rate that is negligible. Then, fine-tuning that pre-trained model for the target task by training normally. If the learning rate of the transferred layers is same as not transferred ones, then all activation maps may be deteriorated that causes worse results.

The idea is, a trained network is able to extract features of source task. When transferred those feature activator layers, features of the target task can also be extracted without even training once. While feature extractor parts of network are transferred, high level reasoning and classification parts are not. A brand new classifier is added according to the target domain.

Even the target task is not similar, transferred layers can still extract features which are low-level features like colors, curves, or edges. However, high-level feature extractor parts of the transferred layers still try to activate same high-level features that do not exist in the target domain. Therefore, transfer learning may not be so successful. If tasks are extremely different, there may be negative effect rather than positive effect to the target task. This situation can be thought as layers' weights are initialized

randomly same as training from scratch, but the small learning rate applied to the transferred layers suppress their evolvement.

2.5 Spectrogram

Spectrogram is a visual representation of a signal's signal strength over time. It is a visual way of representing the frequency contents of a signal. They are generally used for sound signals [81-86].

On a spectrogram image, the vertical axis represents frequencies, the horizontal axis represents positive time, that increases to the right direction, and intensity of colors represent amplitude of the frequency components. While louder amplitudes are represented by light colors through red, low amplitudes are represented by dark colors through blue. An illustration of how a signal is represented as a spectrogram is in Figure 2.29.

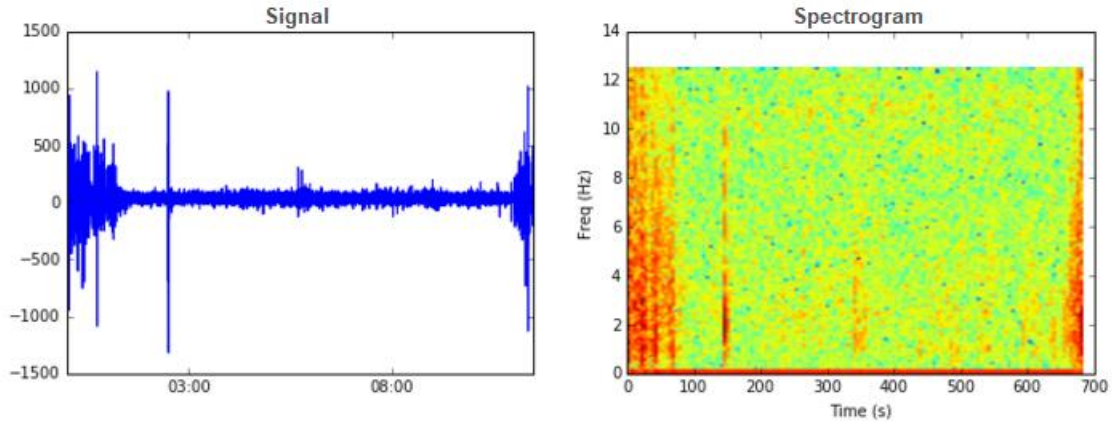


Figure 2.29 A signal and its representative Spectrogram [87]

Spectrograms can be generated by various ways. Most common one is using Fourier transform. To generate it from signal, fast fourier transform (FFT) is used. FFT produces exactly the same result as discrete fourier transform produces but in a much faster way. However, there would be no such a knowledge about time if only FFT is used. So, the actual function of spectrogram is short-time fourier transform. STFT is a sequence of FFTs of windowed data segments as illustrated in Figure 2.30.

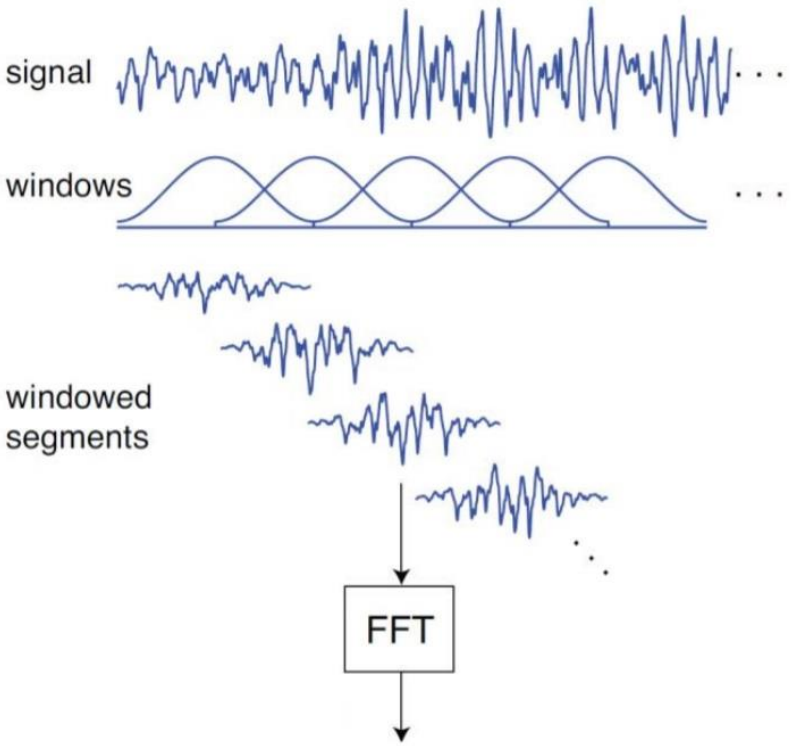


Figure 2.30 Short-time fourier transform steps [88]

STFT states frequency and phase of a signal's local partitions that changes over time [89]. The process is splitting the input signal into small chunks or frames equally along time axis then computing FFT for each chunk to calculate the magnitude of the frequency components. Hence each chunk gets its frequency distribution. Each chunk has its corresponding vertical line for the image. These chunks usually overlap to reduce artifacts at the boundary and have small information from previous chunk. (i.e., generally 50% of chunks are overlapped with the next one as in Figure 2.30) If the collection of all of these frequency components are plotted, i.e., each chunk's corresponding vertical line is located according to its order in the sequence, then this forms the spectrogram image. The process explained above corresponds to computing the squared magnitude of the STFT of the signal $x[n]$, for a window width ω that can be expressed as:

$$spectrogram\{x[n]\}(m,\omega) = |\boldsymbol{STFT}\{x[n]\}(m,\omega)|^2 \tag{2.7}$$

The formula of STFT is as following:

$$\boldsymbol{STFT}\{x[n]\}(m,\omega) = \sum_{n=-\infty}^{\infty} x[n]w[n-m]e^{-j\omega n} \tag{2.7a}$$

where $x[n]$ is the signal to be transformed, $w[n]$ is the windowing function, m is the time, ω is the frequency, n is the chunk.

Size of the window changes the representation in some aspects. If a narrow window is used, then it will provide results that are more accurate in timing compared to wider window. If a wide window is used, it will produce more precise results in frequency representation but poor time resolution compared to narrower window. An example windowing function, Hamming Window, is illustrated in Figure 2.31 and its equation is as following.

$$w(n) = 0.54 - 0.46\cos\left(2\pi\frac{n}{N}\right), \quad where\ 0 \le n \le N \tag{2.8}$$

$$Where\ the\ window\ width,\ \omega = N + 1 \tag{2.8a}$$

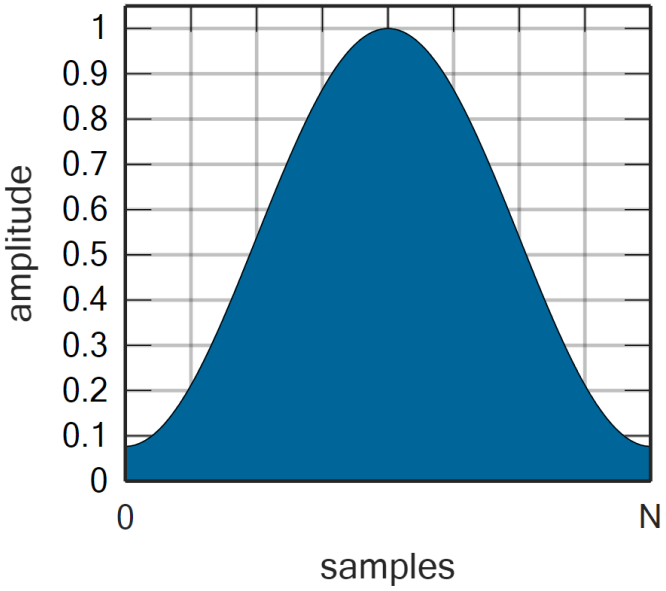


Figure 2.31 Hamming window for N+1 samples

2.6 Scalogram

Scalogram is a visual representation of wavelet transform. A scalogram is the equivalent of a spectrogram for wavelets [90].

The horizontal axis of a scalogram represents time, vertical axis represents scales, and coefficient values are represented by intensity of colors. Scalogram image is the representation of coefficients that comes from wavelet transform with respect to time

and scales. Continuous wavelet transform is used to compute the coefficients. An illustration of scalogram can be seen in Figure 2.32.

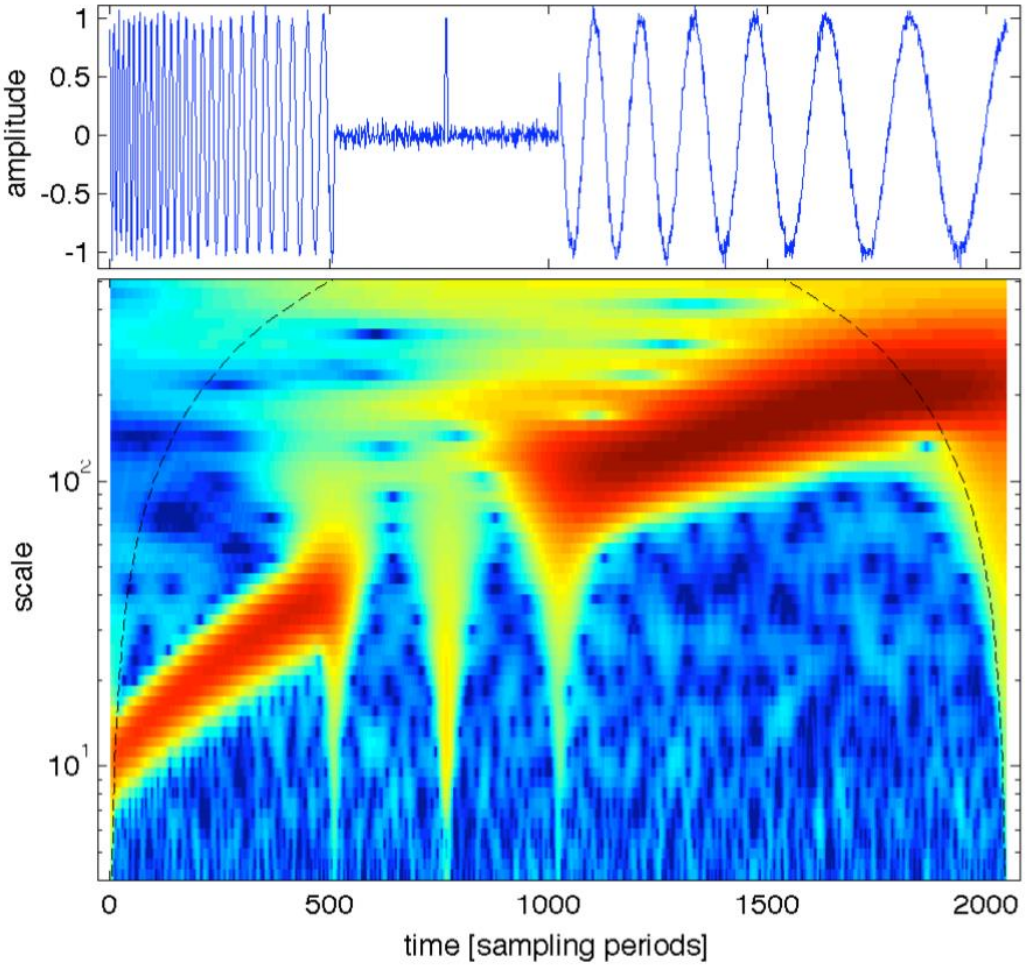


Figure 2.32 A signal and its representation as a scalogram [91]

Continuous wavelet transform is convolution of the input data sequence with a set of functions generated by the mother wavelet. FFT can be used to compute the convolution. When the mother wavelet is not complex, the output of the continuous wavelet transform is real valued. The continuous wavelet transform of a function x(t) at a scale (a>0) a is expressed as following:

$$X_w(a,b) = \frac{1}{|a|^{\frac{1}{2}}} \int_{-\infty}^{\infty} x(t)\bar{\psi}\left(\frac{t-b}{a}\right) dt \qquad (2.9)$$

Where $\psi(t)$ is a continuous function in both time and frequency domain and called as mother wavelet.

The main purpose of the mother wavelet is to provide a source function to generate the daughter wavelets which are simply the translated and scaled versions of the mother wavelet.

2.7 Datasets

Datasets are the integral part of the field of ML. Most important results can be achieved by having advances in datasets as new and larger datasets available rather than advances in algorithms.

Datasets are the key limiting factor of ML's success on the fields that don't have sufficient data available. In this study, presented datasets are used directly or indirectly.

2.7.1 ImageNet

ImageNet [11] is a large scale image dataset. ImageNet contains over 14 million images with over 22000 classes (see Figure 2.33). ImageNet only provides thumbnails and URLs of images, not the actual ones.



Figure 2.33 Actual images of URLs that present in ImageNet and their labels [9]

ImageNet project started in 2006 and published originally in 2009. After the release, it became extremely popular and quickly evolved into an annual competition that algorithms compete with each other on identifying objects in the dataset with the lowest error rate.

The competition is called ImageNet Large Scale Visual Recognition Challenge. ILSRVC dataset contains approximately 1.2 million training images, 50 thousand validation images, 100 thousand test images and 1000 object classes for classification task [9]. From 2010 to 2017 (i.e. the last time it was evaluated, when this paper was written) In just 7 years and 8 competitions, top-5 accuracy rate of classifying objects in the dataset rose from 71.8% to 97.3% (see Figure 2.34) [9,92]. The results have surpassed human abilities and proved that bigger data leads better decisions. ImageNet has a dramatic importance at current interest on DL.

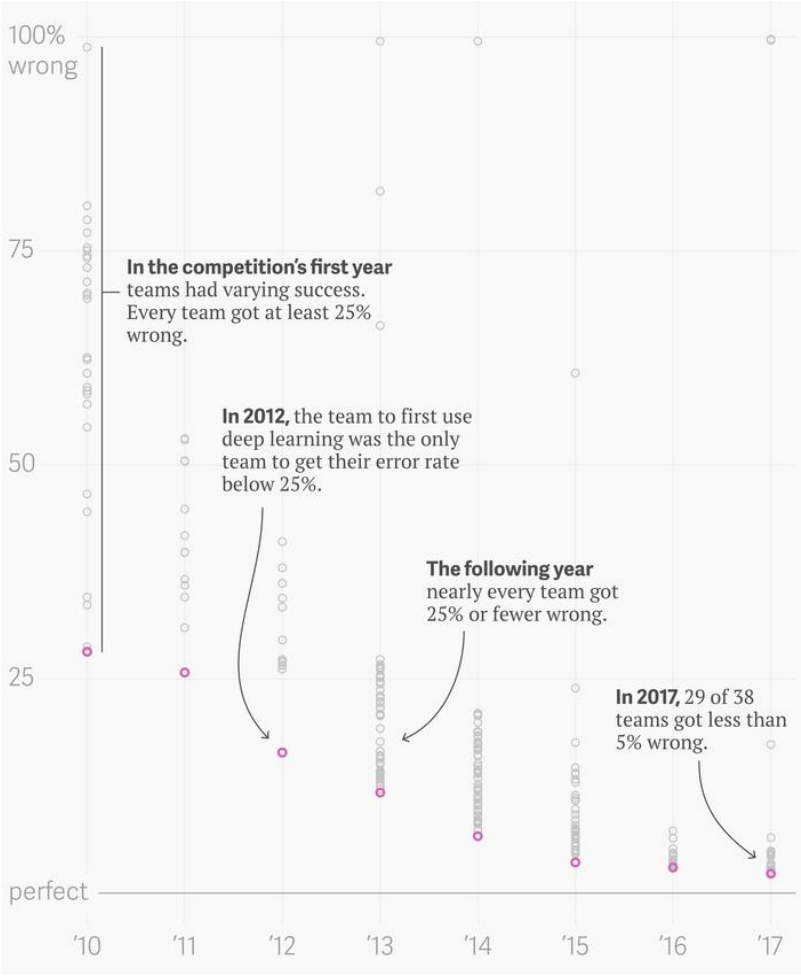


Figure 2.34 ImageNet Large Scale Visual Recognition Challenge Results as top-5 error rate [93]

2.7.2 ISRUC-Sleep

ISRUC-Sleep dataset is a wide and publicly available dataset that contains polysomnographic (PSG) data of 118 subjects in total [34]. ISRUC-Sleep dataset was created to help sleep researchers in their studies.

PSG data is collected from subjects during their sleep. Each PSG recording consists of signals including EEG, electrooculography (EOG), electrocardiography (ECG), electromyography (EMG) from 19 channels. It has 3 sub-groups as illustrated in Figure 2.35.

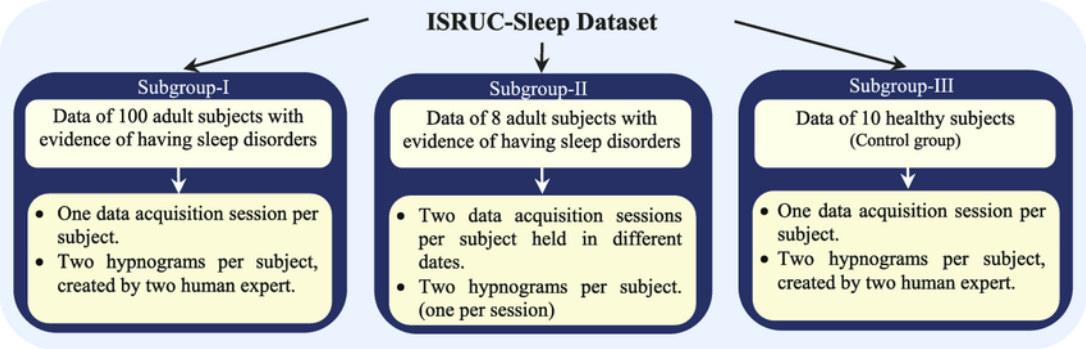


Figure 2.35 Details of ISRUC-Sleep dataset [34]

All recordings were segmented into epochs of 30s. Each segmented recording was labeled by two human experts according to American Academy of Sleep Medicine (AASM) which categorizes into 5 sleep stages. Labeled stages are N1, N2, N3, wake and rapid eye movement (REM). Data is recorded by the sensors that are placed on the body parts such as right leg, left leg and chin for EMG recordings, left and right eyes for EOG recordings, left and right ear lobes for EEG recordings, chest area for ECG recordings. There are 2 EOG channels, 6 EEG channels, 3 EMG channels, and 1 ECG channel form a subject's recording. Descriptions and details of recordings are presented in Figure 2.36.

Channel number	Type of the signal	Label	Frequency rate/Hz	Butterworth	Notch filter	Description
1	EOG	LOC-A2	200	0.3 Hz–35 Hz	50 Hz	Left eyes movements
2		ROC-A1				Right eyes movements
3	EEG	F3-A2	200	0.3 Hz–35 Hz	50 Hz	Brain channels with the references A1 and A2, which placed in the left and right ear-lobes
4		C3-A2				
5		O1-A2				
6		F4-A1				
7		C4-A1				
8		O2-A1				
9	Chin EMG	X1	200	10 Hz–70 Hz	50 Hz	Chin EMG, placed between the chin and the lower lip
10	ECG (EKG)	X2	200		50 Hz	Electrocardiographic
11	Leg-1 EMG	X3	200	10 Hz–70 Hz	50 Hz	Left leg movement
12	Leg-2 EMG	X4				Right leg movement

Figure 2.36 Details of recordings in ISRUC-Sleep dataset [34]

All recordings of EEG, EOG, ECG, EMG were sampled at 200 Hz. The signals of dataset were all pre-processed to reduce the noise.

2.7.2.1 Electroencephalography

Human brain is a complex dynamic system that consists of millions of neurons interconnected with each other. These neurons communicate with each other and carry the information that are coming from or going to the brain.

Electroencephalography (EEG) is a monitoring method to record electrical activity of the brain. It is an electrophysiological, which studies the electrical properties of body parts or functions, monitoring method. It records the brain's spontaneous electrical activity, in other words, brain waves.

EEG is typically collected by using electrodes placed around the scalp. These electrodes are generally noninvasive, but invasive electrodes are sometimes used. It measures voltage fluctuations resulting from ionic current, which is the electric current resulting from motion of charged atoms or molecules.

EEG is frequently used to diagnose epilepsy, sleep disorders, coma, depth of anesthesia and brain death. Example EEG signals of different sleep stages can be seen in Figure 2.37.

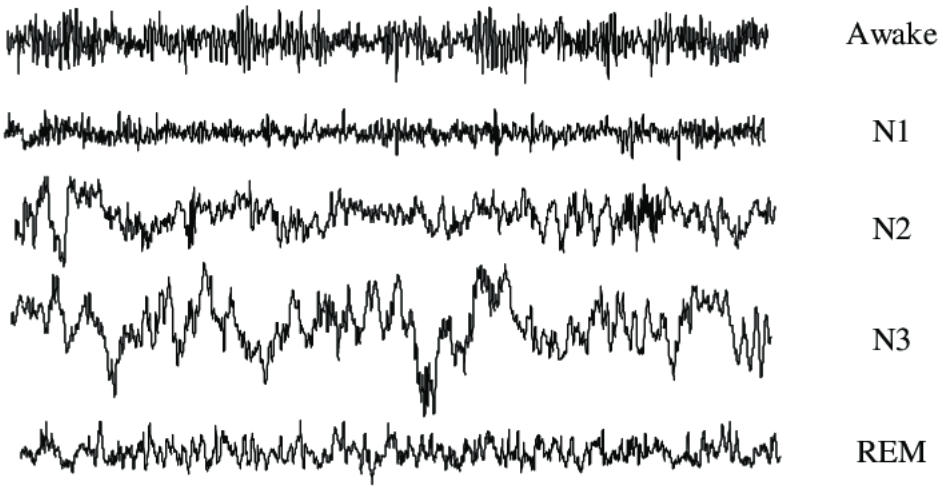


Figure 2.37 Example EEG signals of 5 different sleep stages [94]

Electrode Placement

The cerebrum, which is the largest part of the brain, is divided into two hemispheres. Its outer surface is called as 'cerebral cortex'. The cerebral cortex consists of four lobes which are called frontal, parietal, temporal, and occipital lobes (see Figure 2.38). EEG measures the electrical activity of the cerebral cortex.

ISRUC-Sleep dataset contains EEG data collected from all-night recordings with duration around eight hours. A set of sensors that collected the data in a non-invasive way, according to the international 10-20 standard electrode placement (see Figure 2.39).

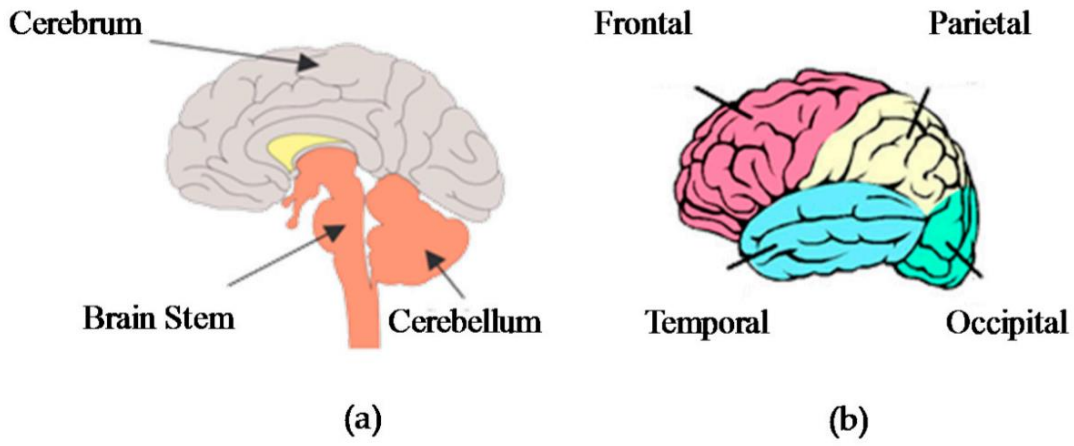


Figure 2.38 (a) An illustration of the human brain & (b) a diagram of the cerebral [95,96]

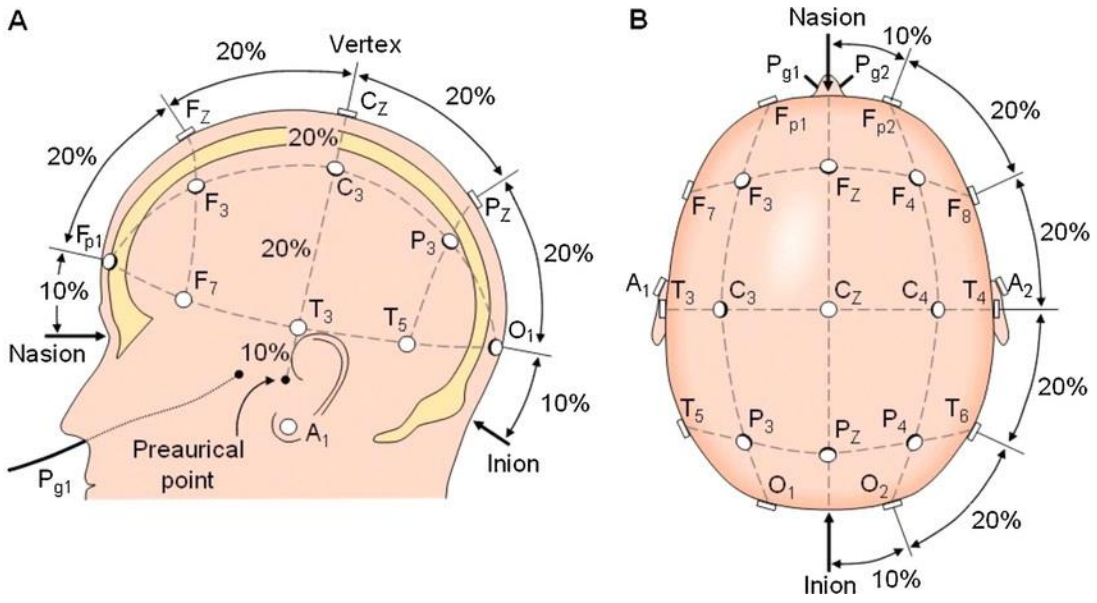


Figure 2.39 The International 10–20 system seen from (A) left and (B) above the head. A: ear lobe, C: central, Pg: nasopharyngeal, P: parietal, F: frontal, Fp: frontal polar, O: occipital [97]

The signals were recorded according to AASM manual. For sleep stage classification from EEG signals, C3-A2 channel was commonly used and also AASM recommends using C3-A2 channel. Hence, records of the recommended channel were used in this study.

CHAPTER 3

IMPLEMENTATION

DL approaches need large amount of data and high performance hardware, mostly GPU, to get reliable success. As a subfield of DL, CNN has the same requirements. As the network goes deeper and number of parameters is increased, CNNs need more data and powerful resources.

When CNN was determined to be applied due to its advantages and good results are desired but there is a small scale dataset, this is a challenging task. One of the ways to overcome this issue is using transfer learning. Even there is a large scale dataset available, transfer learning may be still beneficial such as it can reduce the training duration.

In this study, EEG signals from ISRUC-Sleep dataset was used to classify sleep stages of healthy people. There are 10 healthy subjects in the dataset and their corresponding number of epochs is 8589. Each epoch is labeled to one of 5 sleep stages. Although it may not be considered as a very small dataset, transfer learning still improved its performance.

There is also a trade-off arisen about using CNN that ISRUC is a dataset that consists of signals, but CNNs are very successful in computer vision. CNNs became extremely popular in the visual recognition and most of the interest on CNNs are computer vision based. Also, there is not much example of using visual representation of signals to classify sleep stages. Therefore, signals were converted to spectrograms which visualize both frequency and amplitude of the signal.

3.1 Dataset

The target domain was the domain that has EEG signals that were collected from C3-A2 sensor placed on 10 healthy subjects of ISRUC-Sleep dataset. It has 8589 signals totally. The source domain was the signals collected by the same sensor from 100 patient subjects of the dataset. Total number of signals is 87187.

ISRUC consists of healthy subjects and patient subjects that have sleep disorders. EEG signals, that were used in the study, was sampled at 200 Hz. EEG signals were pre-processed by using a notch filter. Also they were filtered in the interval of 0.3 to 35 Hz. In this study, no further pre-processing was applied to reduce the noise or extract features.

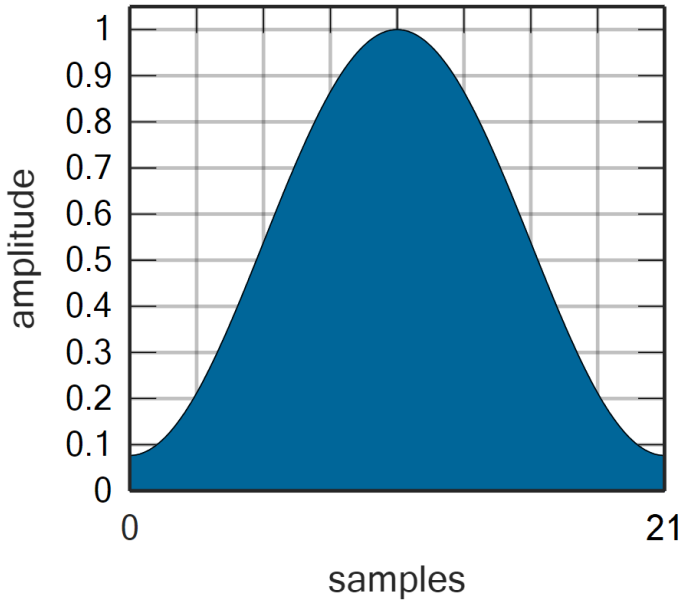


Figure 3.1 Hamming window with the width of 22 samples

Spectrograms were used to represent the signals visually. STFT, which was computed by using FFT, was applied to generate spectrograms from signals. Hamming window was selected as windowing function. Window width was specified as 22 in terms of samples as illustrated in Figure 3.1. Number of points, that chunks are overlapping, was 11. Number of FFT points computed per chunk was 44. Resolution of generated images were 678*535 pixels in ‘rgb’ format and they were stored as ‘.png’ files. The flowchart of how a spectrogram image was generated from a signal is given in Figure 3.2. Example generated spectrogram images for each sleep stage are illustrated in Figure 3.3.

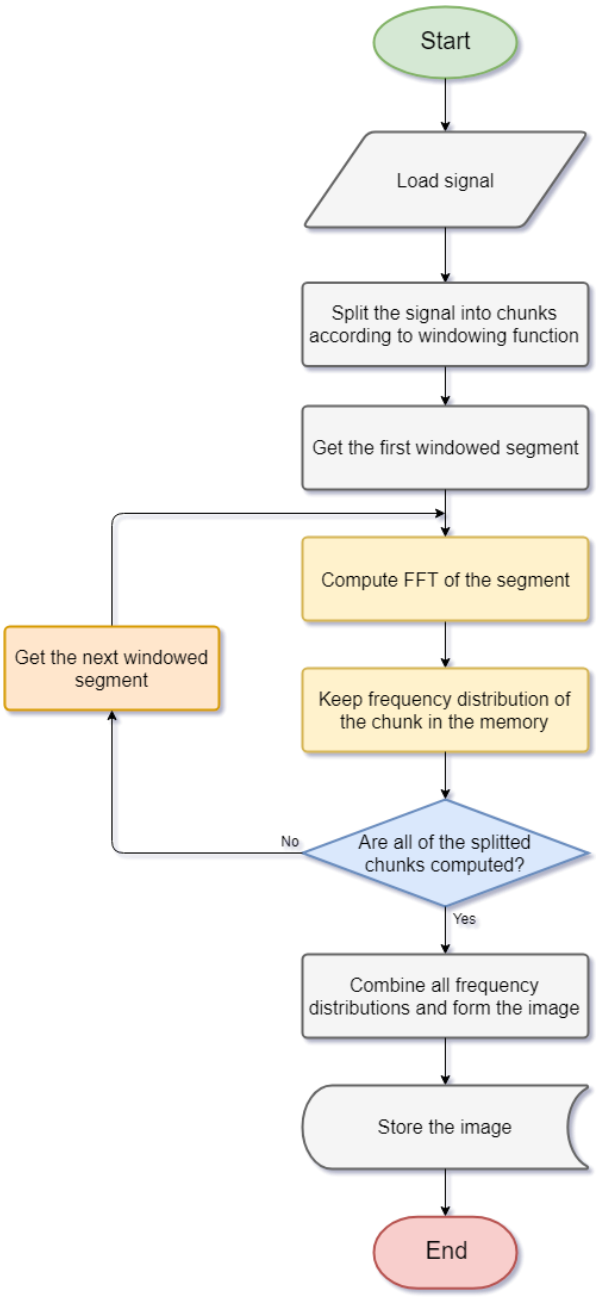


Figure 3.2 Flowchart of generating a spectrogram

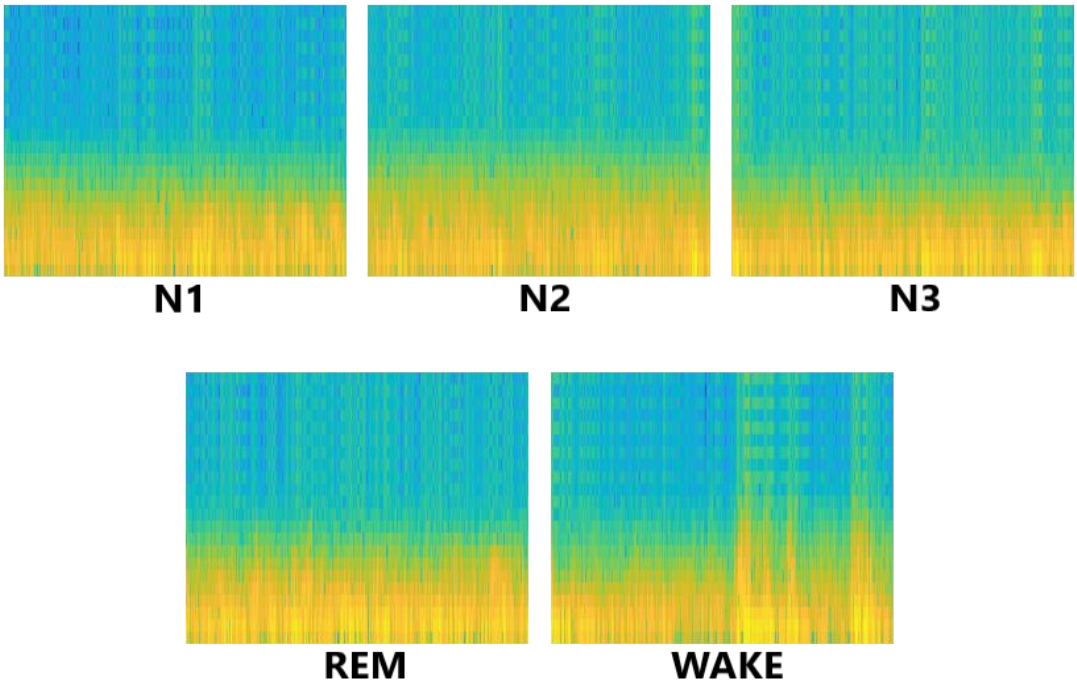


Figure 3.3 Generated spectrogram samples for each sleep stage

Scalograms were also used as representative of signals to be able to compare with spectrograms. Continuous wavelet transform (CWT) was used for that purpose to get coefficients and generate scalograms. Mother wavelet was cmor15-1 and number of scales were 1024. Resolution of generated images were 678*508 pixels in ‘rgb’ format. Example generated spectrogram images for each sleep stage are illustrated in Figure 3.4.

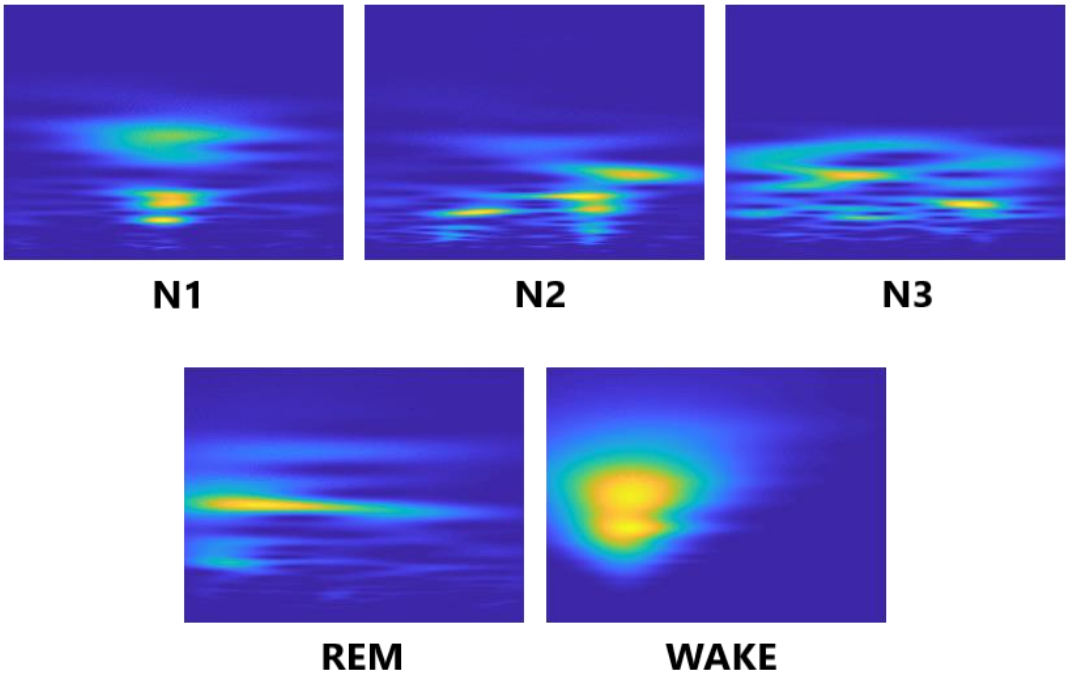


Figure 3.4 Generated scalogram samples for each sleep stage

3.2 Network Architecture

The study is based on using CNNs to make classification. Exactly the same architecture of AlexNet except the size of the last fully connected layer, which was changed to 5, was applied for all experiments.

AlexNet's structure consists of 8 layers, which the first 5 are convolutional layers and the last 3 are fully connected layers. It used ReLU layers immediately after each convolutional layer and each fully connected layer except the last one. Max pooling was also used in the network to reduce the spatial size of the representations. The input size was 227*227*3.

Size of the last fully connected layer was arranged as 5, due to sleep stage classification task as the data was labeled into 5 categories. The architecture of the used version is shown in both Figure 3.5 and Figure 3.7. All layers of the used architecture are listed in Table 3.1.

Table 3.1 List of layers

Layer Number	Layer Type	Details
1	Image Input	227x227x3 images
2	Convolutional	96 11x11x3 convolutions with stride [4 4] and padding [0 0 0 0]
3	ReLU	non-linear ReLU function
4	Cross Channel Normalization	cross channel normalization with 5 channels per element
5	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
6	Convolutional	256 5x5x48 convolutions with stride [1 1] and padding [2 2 2 2]
7	ReLU	non-linear ReLU function
8	Cross Channel Normalization	cross channel normalization with 5 channels per element
9	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
10	Convolutional	384 3x3x256 convolutions with stride [1 1] and padding [1 1 1 1]
11	ReLU	non-linear ReLU function
12	Convolutional	384 3x3x192 convolutions with stride [1 1] and padding [1 1 1 1]
13	ReLU	non-linear ReLU function
14	Convolutional	256 3x3x192 convolutions with stride [1 1] and padding [1 1 1 1]
15	ReLU	non-linear ReLU function
16	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
17	Fully Connected	4096 fully connected layer
18	ReLU	non-linear ReLU function
19	Dropout	50% dropout
20	Fully Connected	4096 fully connected layer
21	ReLU	non-linear ReLU function
22	Dropout	50% dropout
23	Fully Connected	5 fully connected layer
24	Softmax	softmax
25	Classification Output	cross entropy loss

3.3 Pre-Training

Two different pre-training steps were performed to carry out transfer learning. The pre-trained CNNs were AlexNet trained with ImageNet (i.e. 1.2 million images from ImageNet, not all of the images that database contain, in fact it is ILSVRC-2012 data) and AlexNet trained with 100 patient subjects of ISRUC.

AlexNet trained with ImageNet is publicly available and thus obtained. Actual training phase was not performed for that network in this study. That was trained previously with 1.2 million images and validated with 50 thousand images.

Pre-training with ISRUC data was performed by using spectrograms generated from 100 patient subject. 95% of the signals were split out as training set, 5% of them was left out for validation. Mini-batch size was 64. Solver was chosen as stochastic gradient descent with momentum, where momentum was set to 0.9. It was used while updating weights and biases after forward pass. The learning rate was arranged as 1e-3 as can be seen in Figure 3.5.

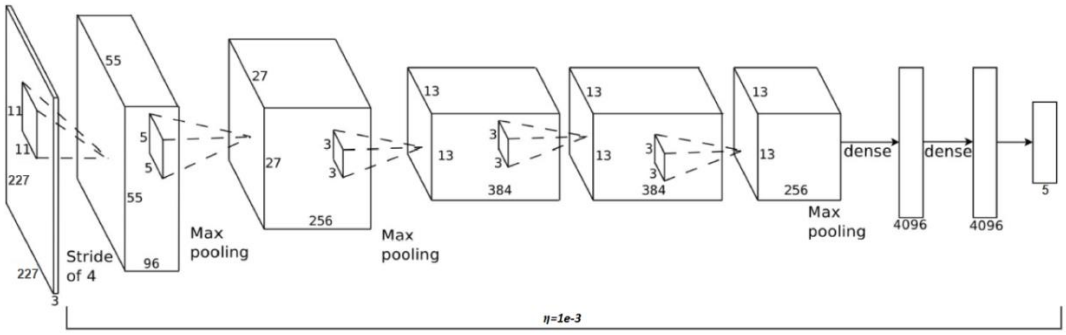


Figure 3.5 Architecture of the CNN trained with 1e-3 learning rate [98]

Image augmentation was held as only translating images from -60 pixels to 60 pixels randomly on x-axis. The reason behind only horizontal axis was used for augmentation is, spectrograms holds information about time on x-axis, while y-axis represents frequencies of the signal. Losing information from frequencies makes the image unpredictable. So, transformations that may distort the image vertically weren't used and making reflection on horizontal direction wasn't used either, even it is not affecting anything with y-axis. All images were resized from 678*535 to 227*227 pixels due to input size of the network.

Training was started with the parameters as described above. All weights and biases were initialized randomly. When the training was finished, the trained network with its final weights and biases was saved for using future purposes as in this study. A flowchart of how pre-training was performed is illustrated in Figure 3.6.

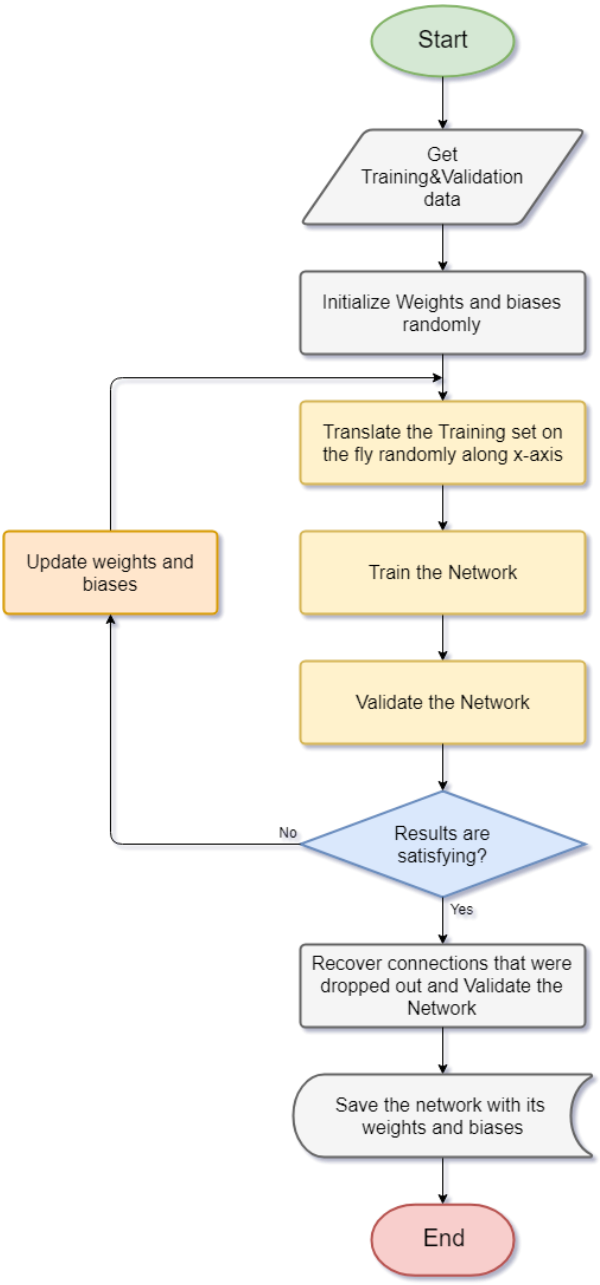


Figure 3.6 Flowchart of training from scratch and saving the network

3.4 Training

Three different trainings were performed on the same target data which was spectrogram images generated from 10 healthy subjects of ISRUC. Two of the trainings were performed for transfer learning, while the other one was performed as training from scratch.

When the network explained in Section 3.3 was trained and saved, it was ready for transferring its acquired knowledge. It was loaded to the training environment and its layers except the last fully connected layer (the first 22 layer at Table 3.1) were transferred with their weights and biases for the target task. The last 3 layers of the network were initialized freshly with random weights and biases.

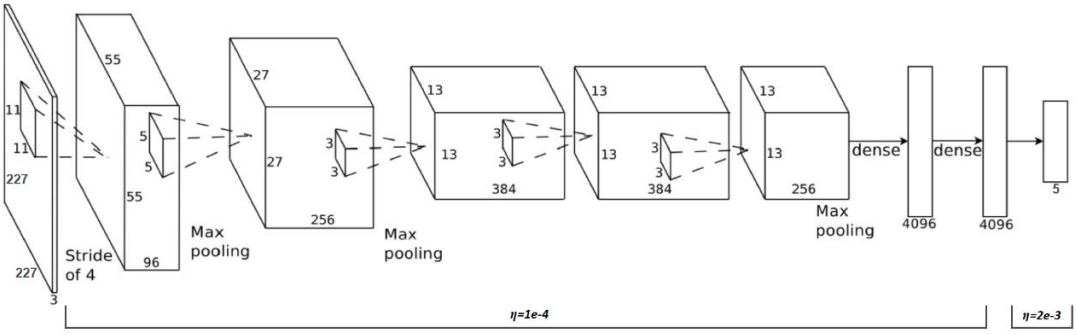


Figure 3.7 Architecture of the CNN trained with 1e-4 learning rate for transferred layers [98]

10-fold cross validation was performed, which 9 folds were separated for training and 1 fold for validation, for all 3 training processes. For pre-trained CNNs, weights and biases of the transferred layers were updated with the learning rate of 1e-4 while others were 2e-3 as illustrated in Figure 3.7 and number of epochs it was trained all along was 16. Training from scratch one was trained in the same way with the same parameters as in Section 3.3, the only difference was applying 10-fold cross validation. Image augmentation and resizing was also the same as in Section 3.3 for all.

How the transfer learning process was performed is given as a flowchart in Figure 3.8. Also, the outline of how 3 different training operations were done is illustrated in Figure 3.9.

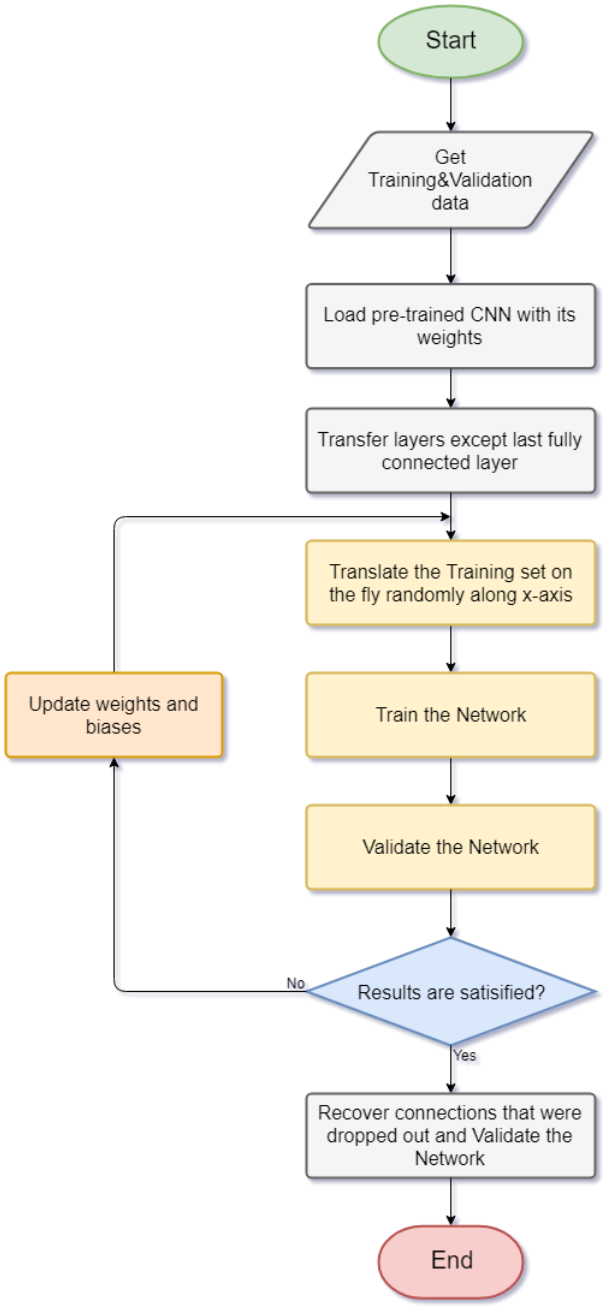


Figure 3.8 Flowchart of training that used transfer learning

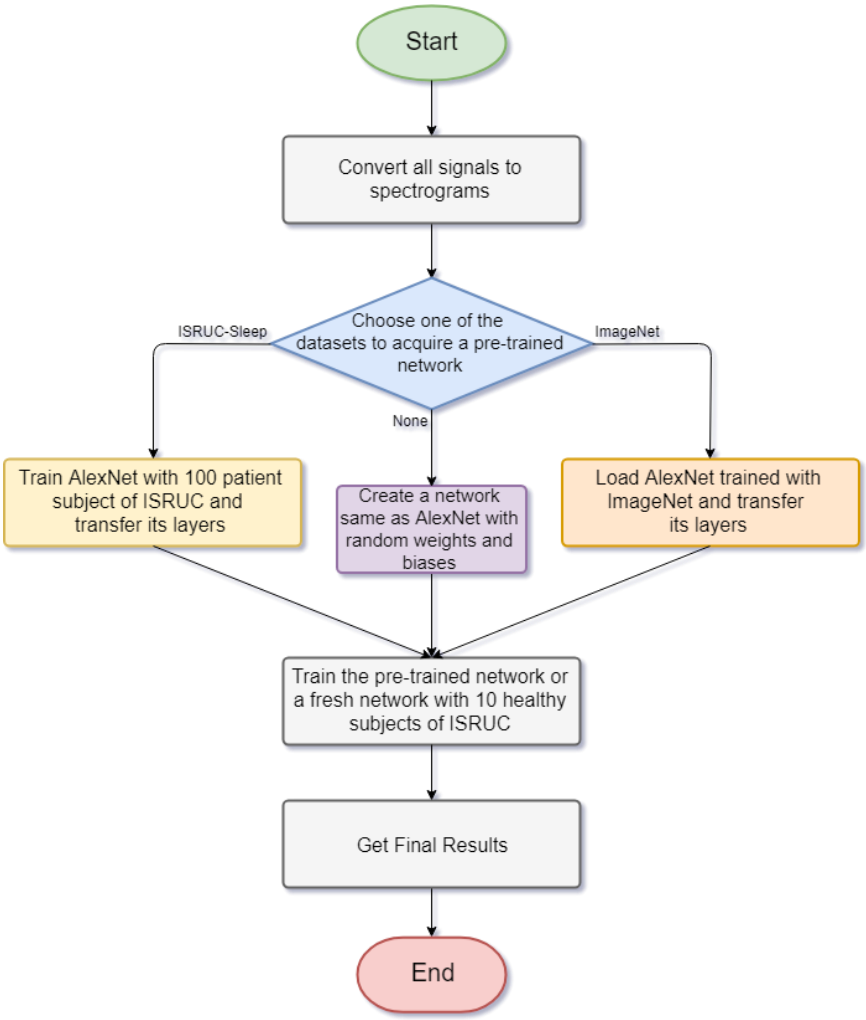


Figure 3.9 Outline of trainings performed on the study as flowchart

CHAPTER 4

RESULTS

All experiments were performed in MATLAB with the hardware resources of one Intel® Core™ i7-7820HQ CPU, 16 GB RAM, and a 4 GB NVIDIA® Quadro® M1200 GPU.

When all 10 trainings for each network were finished, average accuracy of them were calculated for evaluation (accuracy=ACC=$\frac{(TP+TN)}{(P+N)}$, where TP refers to true positive, TN refers to true negative, P and N refer to positive and negative respectively). All results that are showed in this section for the same target domain which explained in Chapter 3. The overall accuracy of the proposed system is 81.05%. The final results are illustrated in Figure 4.1. The results of each fold while applying 10-fold cross validation are listed in Table 4.1.

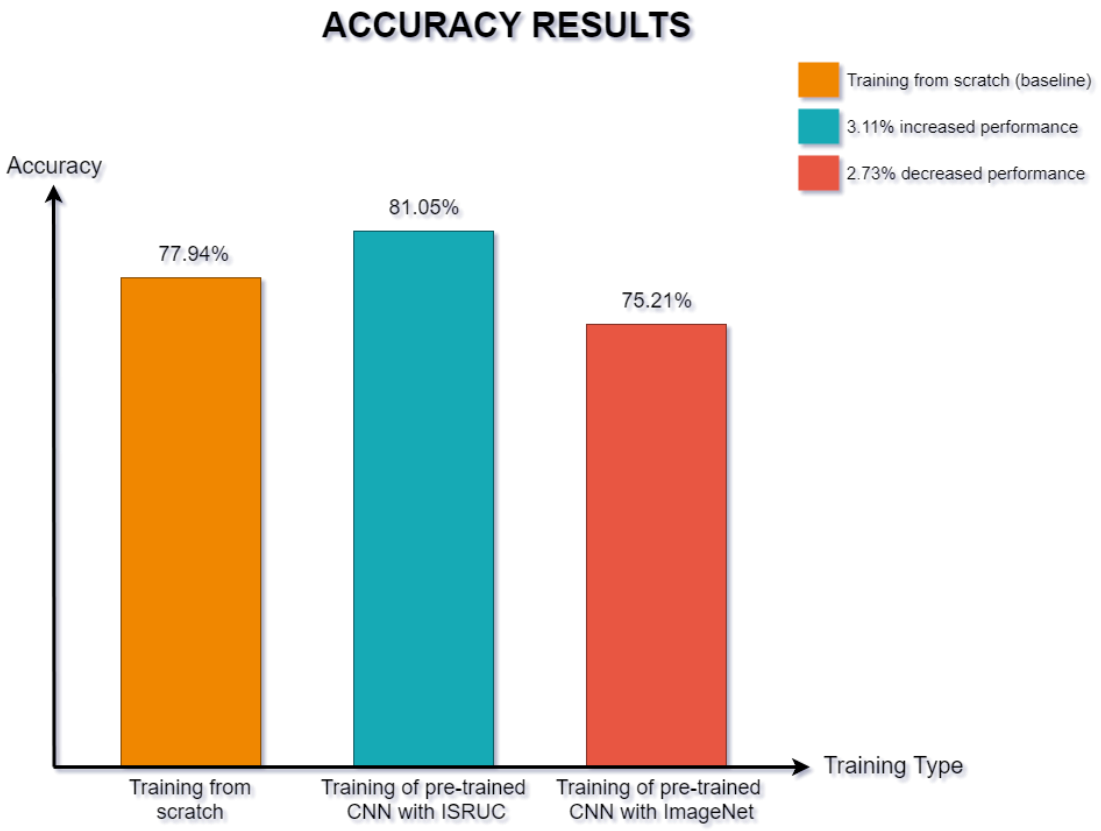


Figure 4.1 Accuracy results of 3 experiments

Table 4.1 Results of each fold for Spectrograms

Training Type / Fold Number	Training from scratch	Training of pre-trained CNN with ISRUC	Training of pre-trained CNN with ImageNet
Accuracy of 1. Fold	0.7662	0.8035	0.7430
Accuracy of 2. Fold	0.8039	0.8028	0.7593
Accuracy of 3. Fold	0.7546	0.7988	0.7416
Accuracy of 4. Fold	0.7680	0.8100	0.7470
Accuracy of 5. Fold	0.7720	0.8012	0.7511
Accuracy of 6. Fold	0.7846	0.8231	0.7402
Accuracy of 7. Fold	0.7762	0.8112	0.7400
Accuracy of 8. Fold	0.7965	0.8244	0.7732
Accuracy of 9. Fold	0.7864	0.8156	0.7641
Accuracy of 10. Fold	0.7860	0.8140	0.7614
Average Accuracy	**77.94%**	**81.05%**	**75.21%**

Training duration for both 16 epochs, that transfer learning was applied, was approximately 32 minutes, and for 36 epochs, which is training from scratch, was approximately 80 minutes.

Transfer learning was applied to achieve better performance. In the study, two different source domains were used for transferring the knowledge to the same target domain. First source domain was ImageNet, which has images of objects. It is relatively different than the target domain. Second source domain was ISRUC-Sleep data, which has signals of sleep stages. It is totally similar, even it is same dataset but different groups.

When both of the domains are EEG signal based, transfer learning increased the accuracy by 3.11% compared to the training from scratch. When one of them is image based and other one is signal based, the accuracy was decreased by 2.73%.

The results came from using scalograms instead of using spectrograms are showed in Table 4.2. As it can be seen, the effect of similar or different source domains to the target domain's result is same as it was in spectrograms.

Table 4.2 Results of each fold for Scalograms

Training Types / Fold Number	Training from scratch	Training of pre-trained CNN with ISRUC	Training of pre-trained CNN with ImageNet
Accuracy of 1. Fold	0.7162	0.7546	0.7418
Accuracy of 2. Fold	0.7561	0.7934	0.7141
Accuracy of 3. Fold	0.7604	0.7814	0.7500
Accuracy of 4. Fold	0.7514	0.7853	0.7328
Accuracy of 5. Fold	0.7360	0.7535	0.7384
Accuracy of 6. Fold	0.7721	0.7907	0.7465
Accuracy of 7. Fold	0.7444	0.7421	0.7351
Accuracy of 8. Fold	0.7418	0.7721	0.7337
Accuracy of 9. Fold	0.7573	0.7876	0.7433
Accuracy of 10. Fold	0.7279	0.7558	0.7535
Average Accuracy	**74.64%**	**77.17%**	**73.89%**

For scalograms, when both of the domains were similar, transfer learning increased the accuracy by 2.53% compared to the training from scratch. When they weren't similar, the accuracy was decreased by 0.75%.

A comparison of the proposed study with previous studies, that all use deep learning methods with single channel EEG signals and all of them are the most recent studies that define state-of-the-art in the field, can be found in Table 4.3.

Table 4.3 Comparison of the proposed study with previous studies

Study	Top-1 Overall Accuracy %	Dataset
Tsinalis et al. [14]	74%	Sleep-EDF
Supratak et al. [36]	86.2%	MASS
Supratak et al. [36]	82%	Sleep-EDF
Vilamala et al. [37]	86%	Sleep-EDF
Gharbali et al. [38]	84%	ISRUC-Sleep
Proposed Study	**81.05%**	**ISRUC-Sleep**

4.1 Conclusion

Results showed that transfer learning increased the accuracy when both of the domains are similar, but it decreased the accuracy when used on different domains. Number of epochs needed and so the training duration both were decreased. The result of the proposed study achieved pretty close overall accuracy compared to state of the art results (see Table 4.3).

Future works can be established on fine-tuning the parameters of STFT that generates spectrograms or fine-tuning scalogram parameters. Also, testing the proposed model on various signal datasets including transferring knowledge from different datasets can be done. Different image augmentation types can be adapted. Using a different CNN architecture rather than AlexNet can be done. Also, other recommended EEG channels from AASM can be used.

It is clear that after making adjustments on parameters of spectrogram or scalogram generation, overall performance may increase. Combining this study's best resulting model with adjusted parameters of scalogram or spectrogram generation may boost the best result.

REFERENCES

[1] Koza J. R., Bennett F. H., Andre D. and Keane M. A., "Automated Design of Both the Topology and Sizing of Analog Electrical Circuits Using Genetic Programming," in *Artificial Intelligence in Design '96*, Springer, 1996, pp. 151-170.

[2] LeCun Y., Boser B., Denker J. S., Henderson D., Howard R. E., Hubbard W. and Jackel L. D., "Backpropagation Applied to Handwritten Zip Code Recognition," *Neural Computation*, vol. 1, no. 4, pp. 541-551, 1989.

[3] Zeiler M. D. and Fergus R., "Visualizing and Understanding Convolutional Networks," in *Computer Vision – ECCV 2014*, vol. 8689, Zurich, Switzerland, Springer, Cham, 2014, pp. 818-833.

[4] Krizhevsky A., Sutskever I. and Hinton G. E., "ImageNet Classification with Deep Convolutional Neural Networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, Lake Tahoe, Nevada, Curran Associates Inc., 2012, pp. 1097-1105.

[5] Lin M., Chen Q. and Yan S., "Network In Network," *CoRR, Abs/1312.4400*, 2013.

[6] Simonyan K. and Zisserman A., "Very Deep Convolutional Networks for Large-Scale Image Recognition," *CoRR, Abs/1409.1556*, 2014.

[7] Szegedy C., Liu W., Jia Y., Sermanet P., Reed S., Anguelov D., Erhan D., Vanhoucke V. and Rabinovich A., "Going Deeper with Convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, IEEE, 2015, pp. 1-9.

[8] He K., Zhang X., Ren S. and Sun J., "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, IEEE, 2016, pp. 770-778.

[9] Russakovsky O., Deng J., Su H., Krause J., Satheesh S., Ma S., Huang Z., Karpathy A., Khosla A., Bernstein M., Berg A. C. and Fei-Fei L., "ImageNet

Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211-252, 2015.

[10] Griffin G., Holub A. and Perona P., "Caltech-256 object category dataset," California Institute of Technology, Los Angeles, CA, 2007.

[11] Deng J., Dong W., Socher R., Li L. J., Li K. and Fei-Fei L., "ImageNet: A Large-Scale Hierarchical Image Database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Miami, FL, IEEE, 2009.

[12] Liang S. F., Kuo C. E., Hu Y. H., Pan Y. H. and Wang Y. H., "Automatic Stage Scoring of Single-Channel Sleep EEG by Using Multiscale Entropy and Autoregressive Models," *IEEE Transactions on Instrumentation and Measurement,* vol. 61, no. 6, pp. 1649-1657, 2012.

[13] Sharma R., Pachori R. B. and Upadhyay A., "Automatic Sleep Stages Classification Based on Iterative Filtering of Electroencephalogram Signals," *Neural Computing and Applications,* vol. 28, no. 10, pp. 2959-2978, 2017.

[14] Tsinalis O., Matthews P. M., Guo Y. and Zafeiriou S., "Automatic Sleep Stage Scoring with Single-Channel EEG Using Convolutional Neural Networks," *CoRR, Abs/1610.01683,* 2016.

[15] Cui Z., Zheng X., Shao X. and Cui L., "Automatic Sleep Stage Classification Based on Convolutional Neural Network and Fine-Grained Segments," *Complexity,* vol. 2018, no. Article ID 9248410, 2018.

[16] Chambon S., Galtier M. N., Arnal P. J., Wainrib G. and Gramfort A., "A Deep Learning Architecture for Temporal Sleep Stage Classification Using Multivariate and Multimodal Time Series," *IEEE Transactions on Neural Systems and Rehabilitation Engineering,* vol. 26, no. 4, pp. 758-769, 2018.

[17] Lecun Y., Bottou L., Bengio Y. and Haffner P., "Gradient-based learning applied to document recognition," *Proceedings of the IEEE,* vol. 86, no. 11, pp. 2278-2324, 1998.

[18] Barnes Z., Cipollone F. and Romero T., "Techniques for Image Classification on Tiny-ImageNet," Stanford, CA, 2017.

[19] Sermanet P., Eigen D., Zhang X., Mathieu M., Fergus R. and Lecun Y., "OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks," in *CoRR, abs/1312.6229*, 2013.

[20] Pan S. J. and Yang Q., "A Survey on Transfer Learning," *IEEE Transactions on Knowledge and Data Engineering,* vol. 22, no. 10, pp. 1345-1359, 2010.

[21] Aytar Y. and Zisserman A., "Tabula rasa: Model transfer for object category detection," in *2011 International Conference on Computer Vision*, Barcelona, IEEE, 2011, pp. 2252-2259.

[22] Tommasi T., Orabona F. and Caputo B., "Safety in numbers: Learning categories from few examples with multi model knowledge transfer," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, San Fransisco,CA, IEEE, 2010, pp. 3081-3088.

[23] Khosla A., Zhou T., Malisiewicz T., Efros A. A. and Torralba A., "Undoing the Damage of Dataset Bias," in *European Conference on Computer Vision 2012*, Berlin, Springer, 2012, pp. 158-171.

[24] Saenko K., Kulis B., Fritz M. and Darrell T., "Adapting Visual Category Models to New Domains," in *European Conference on Computer Vision, ECCV-2010*, Berlin, Springer, 2010, pp. 213-226.

[25] Donahue J., Jia Y., Vinyals O., Hoffman Zhang J., N., Tzeng E. and Darrell T., "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition," in *International Conference on Machine Learning*, JMLR, 2014, pp. 647-655.

[26] Ng H. W., Nguyen V. D., Vonikakis V. and Winkler S., "Deep Learning for Emotion Recognition on Small Datasets Using Transfer Learning," in *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction*, New York, NY, ACM, 2015, pp. 443-449.

[27] Chatfield K., Simonyan K., Vedaldi A. and Zisserman A., "Return of the Devil in the Details: Delving Deep into Convolutional Nets," *CoRR, abs/1405.3531,* 2014.

[28] Shin H.C., Roth H. R., Gao M., Lu L., Xu Z., Nogues I., Yao J., Mollura D. and Summers R. M., "Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning," IEEE Transactions on Medical Imaging, vol. 35, no. 5, pp. 1285-1298, 2016.

[29] Krizhevsky A. and Hinton G., "Learning multiple layers of features from tiny images," University of Toronto, Toronto, 2009.

[30] Wimmer G., Vécsei A. and Uhl A., "CNN Transfer Learning for the Automated Diagnosis of Celiac Disease," in *2016 Sixth International Conference on Image Processing Theory, Tools and Applications (IPTA)*, Oulu, Finland, IEEE, 2016, pp. 1-6.

[31] Langkvist M., Karlsson L. and Loutfi A., "Sleep Stage Classification Using Unsupervised Feature Learning," *Advances in Artificial Neural Systems,* vol. 2012, no. Article No: 5 Article ID: 107046, p. 9, 2012.

[32] Zhang J., Wu Y., Bai J. and Chen F., "Automatic Sleep Stage Classification Based on Sparse Deep Belief Net and Combination of Multiple Classifiers," *Transactions of the Institute of Measurement and Control,* vol. 38, no. 4, pp. 435-451, 2016.

[33] O'Reilly C., Gosselin N., Carrier J. and Nielsen T., "Montreal Archive of Sleep Studies: an Open-Access Resource for Instrument Benchmarking and Exploratory Research," *Journal of Sleep Research,* vol. 23, no. 6, pp. 628-635, 2014.

[34] Khalighi S., Sousa T., Santos J. M. and Nunes U., "ISRUC-Sleep: A comprehensive public dataset for sleep researchers," *Computer Methods and Programs in Biomedicine,* vol. 124, pp. 180-192, 2015.

[35] Kemp B., Zwinderman A., Tuk B., Kamphuisen H. and Oberye J., "Analysis of a Sleep-Dependent Neuronal Feedback Loop: the slow-wave microcontinuity of

the EEG," *IEEE Transactions on Biomedical Engineering*, vol. 47, no. 9, pp. 1185-1194, 2000.

[36] Supratak A., Dong H., Wu C. and Guo Y., "DeepSleepNet: A Model for Automatic Sleep Stage Scoring Based on Raw Single-Channel EEG," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 25, no. 11, pp. 1998-2008, 2017.

[37] Vilamala A., Madsen K. H. and Hansen L. K., "Deep Convolutional Neural Networks for Interpretable Analysis of EEG Sleep Stage Scoring," in *2017 IEEE 27th International Workshop on Machine Learning for Signal Processing (MLSP)*, Tokyo, IEEE, 2017, pp. 1-6.

[38] Gharbali A. A., Najdi S. and Fonseca J. M., "Transfer Learning of Spectrogram Image for Automatic Sleep Stage Classification," in *International Conference Image Analysis and Recognition*, Cham, Springer, 2018, pp. 522-528.

[39] Chandra A. and Narayanan D., "Machine Learning and Text Analytics," 23 July 2014. [Online]. Available: https://blogs.technet.microsoft.com/machinelearning/2014/07/23/machine-learning-and-text-analytics/. [Accessed 10 October 2018].

[40] Mitchell T. M., Machine Learning, New York, NY: McGraw-Hill, Inc., 1997, p. 432.

[41] MathWorks, "Machine Learning in MATLAB," [Online]. Available: https://www.mathworks.com/help/stats/machine-learning-in-matlab.html. [Accessed 10 October 2018].

[42] Big Data Made Simple, "Machine Learning Explained: Understanding Supervised, Unsupervised, and Reinforcement Learning," [Online]. Available: http://bigdata-madesimple.com/machine-learning-explained-understanding-supervised-unsupervised-and-reinforcement-learning/. [Accessed 20 8 2018].

[43] Richmond A., "A Brief Introduction to Artificial Neural Networks," [Online]. Available: http://tuxar.uk/brief-introduction-artificial-neural-networks/. [Accessed 10 October 2018].

[44] Camillo F., "Neural Representation of Logic Gates," 8 July 2017. [Online]. Available: https://towardsdatascience.com/neural-representation-of-logic-gates-df044ec922bc. [Accessed 10 October 2018].

[45] McCulloch W. S. and Pitts W., "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115-133, 1943.

[46] Rosenblatt F., "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain," *Psychological Review*, vol. 65, no. 6, p. 386, 1958.

[47] Werbos P. J., Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences, Cambridge, MA: Harvard University, 1975.

[48] Araújo L., "Solving XOR with a single Perceptron," 26 March 2018. [Online]. Available: https://medium.com/@lucaspereira0612/solving-xor-with-a-single-perceptron-34539f395182. [Accessed 10 October 2018].

[49] llhthinker, "A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification," 13 December 2017. [Online]. Available: https://github.com/llhthinker/NLP-Papers/blob/master/text%20classification/2017-10/A%20Sensitivity%20Analysis%20of%20(and%20Practitioners%E2%80%99%20Guide%20to)%20Convolutional/note.md. [Accessed 10 October 2018].

[50] Srivastava T., "How does Artificial Neural Network (ANN) algorithm work? Simplified!," 20 October 2014. [Online]. Available: https://www.analyticsvidhya.com/blog/2014/10/ann-work-simplified/. [Accessed 10 October 2018].

[51] Goodfellow I., Bengio Y., Courville A. and Bengio Y., Deep Learning, Cambridge, MA: MIT press, 2016.

[52] Mazur M., "A Step by Step Backpropagation Example," 17 March 2015. [Online]. Available: https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/. [Accessed 10 October 2018].

[53] Rumelhart D. E., Hinton G. E. and Williams R. J., "Learning Representations by Back-Propagating Errors," *Nature,* vol. 323, no. 6088, p. 533, 1986.

[54] Moujahid A., "A Practical Introduction to Deep Learning with Caffe and Python," 26 June 2016. [Online]. Available: http://adilmoujahid.com/posts/2016/06/introduction-deep-learning-python-caffe/. [Accessed 10 October 2018].

[55] Cireşan D., Meier U. and Schmidhuber J., "Multi-column Deep Neural Networks for Image Classification," *CoRR,* vol. abs/1202.2745, 2012.

[56] Silver D., Huang A., Maddison C. J., Guez A., Sifre L., Van Den Driessche G., Schrittwieser J., Antonoglou I., Panneershelvam V. and Lanctot M., "Mastering the Game of Go with Deep Neural Networks and Tree Search," *Nature,* vol. 529, no. 7587, p. 484, 2016.

[57] Dechter R., "Learning While Searching in Constraint-Satisfaction Problems," in *Proceedings of the Fifth AAAI National Conference on Artificial Intelligence*, Philadelphia, PA, AAAI Press, 1986, pp. 178-183.

[58] Huang J., "The Intelligent Industrial Revolution," 24 October 2016. [Online]. Available: https://blogs.nvidia.com/blog/2016/10/24/intelligent-industrial-revolution/. [Accessed 11 October 2018].

[59] Nweke H. F., Teh Y. W., Al-garadi M. A. and Alo U. R., "Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges," *Expert Systems with Applications,* vol. 105, pp. 233-261, 2018.

[60] Schmidhuber J., "Deep learning in neural networks: An overview," *Neural Networks,* vol. 61, pp. 85-117, 2015.

[61] Bengio Y., "Learning Deep Architectures for AI," *Foundations and Trends in Machine Learning,* vol. 2, no. 1, pp. 1-127, 2009.

[62] Nielsen M. A., Neural Networks and Deep Learning, Determination Press, 2015.

[63] Fukushima K., "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, vol. 36, no. 4, pp. 193-202, 1980.

[64] Hubel D. H. and Wiesel T. N., "Receptive Fields of Single Neurones in the Cat's Striate Cortex," *The Journal of Physiology*, vol. 148, no. 3, pp. 574-591, 1959.

[65] Cireşan D. C., Meier U., Masci J., Gambardella L. M. and Schmidhuber J., "Flexible, High Performance Convolutional Neural Networks for Image Classification," in *Twenty-Second International Joint Conference on Artificial Intelligence*, vol. 22, Barcelona, AAAI Press, 2011, pp. 1237-1242.

[66] Chellapilla K., Puri S. and Simard P., "High Performance Convolutional Neural Networks for Document Processing," in *Tenth International Workshop on Frontiers in Handwriting Recognition*, La Baule, Suvisoft, 2006.

[67] Hinton G. E., Osindero S. and Teh Y.-W., "A Fast Learning Algorithm for Deep Belief Nets," *Neural Computation*, vol. 18, no. 7, pp. 1527-1554, 2006.

[68] Bengio Y., Lamblin P., Popovici D. and Larochelle H., "Greedy Layer-wise Training of Deep Networks," in *Advances in Neural Information Processing Systems*, Cambridge, MA, MIT Press, 2006, pp. 153-160.

[69] Ranzato M., Poultney C., Chopra S. and LeCun Y., "Efficient Learning of Sparse Representations with an Energy-based Model," in *Advances in Neural Information Processing Systems*, Cambridge, MA, MIT Press, 2006, pp. 1137-1144.

[70] Deshpande A., "A Beginner's Guide To Understanding Convolutional Neural Networks," 20 July 2016. [Online]. Available: https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/. [Accessed 11 October 2018].

[71] MathWorks, "Convolutional Neural Network," [Online]. Available: https://www.mathworks.com/solutions/deep-learning/convolutional-neural-network.html. [Accessed 10 October 2018].

[72] Vernwal R., "How should I start with CNN?," 29 July 2017. [Online]. Available: https://becominghuman.ai/how-should-i-start-with-cnn-c62a3a89493b. [Accessed 10 October 2018].

[73] MathWorks, "Visualize Features of a Convolutional Neural Network," [Online]. Available: https://www.mathworks.com/help/deeplearning/examples/visualize-features-of-a-convolutional-neural-network.html;jsessionid=ed1e6964f21a3cb14e7e22144285. [Accessed 10 October 2018].

[74] Sarkar K., "ReLU : Not a Differentiable Function: Why used in Gradient Based Optimization? and Other Generalizations of ReLU," 31 May 2018. [Online]. Available: https://medium.com/@kanchansarkar/relu-not-a-differentiable-function-why-used-in-gradient-based-optimization-7fef3a4cecec. [Accessed 10 October 2018].

[75] Singhal H., "Convolutional Neural Network with TensorFlow implementation," 17 June 2017. [Online]. Available: https://medium.com/data-science-group-iitr/building-a-convolutional-neural-network-in-python-with-tensorflow-d251c3ca8117. [Accessed 10 October 2018].

[76] Srivastava N., G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research,* vol. 15, no. 1, pp. 1929-1958, 2014.

[77] Hollemans M., "Matrix Multiplication with Metal Performance Shaders," 22 February 2017. [Online]. Available: http://machinethink.net/blog/mps-matrix-multiplication/. [Accessed 10 October 2018].

[78] Miranda L., "Understanding Softmax and the Negative Log-Likelihood," 13 August 2017. [Online]. Available: https://ljvmiranda921.github.io/notebook/2017/08/13/softmax-and-the-negative-log-likelihood/. [Accessed 10 October 2018].

[79] Pratt L. Y., "Discriminability-based Transfer Between Neural Networks," in *Advances in Neural Information Processing Systems*, San Francisco, CA, Morgan Kaufmann Publishers Inc., 1992, pp. 204-211.

[80] Wainrib G., "Transfer Learning and the Rise of Collaborative Artificial Intelligence," 15 February 2017. [Online]. Available: https://medium.com/owkin/transfer-learning-and-the-rise-of-collaborative-artificial-intelligence-41f9e2950657. [Accessed 10 October 2018].

[81] Saunders F. A., Hill W. A. and Franklin B., "A Wearable Tactile Sensory Aid for Profoundly Deaf Children," *Journal of Medical Systems*, vol. 5, no. 4, pp. 265-270, 1981.

[82] Geitgey A., "Machine Learning is Fun Part 6: How to do Speech Recognition with Deep Learning," 24 December 2016. [Online]. Available: https://medium.com/@ageitgey/machine-learning-is-fun-part-6-how-to-do-speech-recognition-with-deep-learning-28293c162f7a. [Accessed 11 October 2018].

[83] Dragonetti M., "Bird Songs and Calls With Spectrograms of Southern Tuscany," [Online]. Available: http://www.birdsongs.it/index.asp. [Accessed 11 October 2018].

[84] Deng L., Seltzer M. L., Yu D., Acero A., Mohamed A.-r. and Hinton G., "Binary Coding of Speech Spectrograms Using a Deep Auto-Encoder," in *INTERSPEECH-2010*, 2010, pp. 1692-1695.

[85] Badshah A. M., Ahmad J., Rahim N. and Baik S. W., "Speech Emotion Recognition from Spectrograms with Deep Convolutional Neural Network," in *2017 International Conference on Platform Technology and Service (PlatCon)*, Busan, IEEE, 2017, pp. 1-5.

[86] Deller J. R., Proakis J. G. and Hansen J. H., Discrete Time Processing of Speech Signals, Upper Saddle River, NJ: Prentice Hall PTR, 1993.

[87] Fiedler A., "How to Create an EEG Spectrogram with Python," 5 January 2016. [Online]. Available: http://andyfiedler.com/2016/01/how-to-create-an-eeg-spectrogram-with-python. [Accessed 10 October 2018].

[88] Sethares W. A., Rhythm and Transforms, London: Springer-Verlag, 2007.

[89] Sejdić E., Djurović I. and Jiang J., "Time-Frequency Feature Representation Using Energy Concentration: An Overview of Recent Advances," *Digital Signal Processing*, vol. 19, no. 1, pp. 153-183, 2009.

[90] Sejdic E., Djurovic I. and Stankovic L., "Quantitative Performance Analysis of Scalogram as Instantaneous Frequency Estimator," *IEEE Transactions on Signal Processing*, vol. 56, no. 8, pp. 3837-3845, 2008.

[91] de Wit T. D., "Multiscale Analysis," 2010. [Online]. Available: https://warwick.ac.uk/fac/cross_fac/complexity/study/msc_and_phd/co907/resources/wavelets.pdf. [Accessed 10 October 2018].

[92] ImageNet, "Large Scale Visual Recognition Challenge (ILSVRC)," Stanford Vision Lab, [Online]. Available: http://www.image-net.org/challenges/LSVRC/. [Accessed 11 October 2018].

[93] Gershgorn D., "The Data That Transformed AI Research and Possibly the World," 26 July 2017. [Online]. Available: https://qz.com/1034972/the-data-that-changed-the-direction-of-ai-research-and-possibly-the-world/. [Accessed 10 October 2018].

[94] Khalighi S., Sousa T., Pires G. and Nunes U., "Automatic Sleep Staging: A Computer Assisted Approach for Optimal Combination of Features and Polysomnographic Channels," Expert Systems with Applications, vol. 40, no. 17, pp. 7046-7059, 2013.

[95] Aboalayon K., Faezipour M., Almuhammadi W. and Moslehpour S., "Sleep Stage Classification Using EEG Signal Analysis: A Comprehensive Survey and New Investigation," Entropy, vol. 18, no. 9, p. 272, 2016.

[96] "Electroencephalography I Laboratory," [Online]. Available: https://glneurotech.com/docrepo/teaching-labs/EEG_I_Teacher.pdf. [Accessed 28 October 2018].

[97] Malmivuo J. and Plonsey R., Bioelectromagnetism: Principles and Applications of Bioelectric and Biomagnetic Fields, New York: Oxford University Press, 1995.

[98] Pedraza A., Gallego J., Lopez S., Gonzalez L., Laurinavicius A. and Bueno G., "Glomerulus Classification with Convolutional Neural Networks," in *Annual Conference on Medical Image Understanding and Analysis*, Cham, Springer, 2017, pp. 839-849.

CURRICULUM VITAE

PERSONAL INFORMATION



Name Surname : Ahmet Esad TOP

Date of Birth : 25.07.1993

Phone :

E-mail : aetop@ybu.edu.tr

EDUCATION

Master Degree : T.C. Ankara Yildirim Beyazit University - 2018 - GPA:3.88/4

Bachelor : T.C. Yildirim Beyazit University - 2016 - GPA: 3.31/4

High School : Muradiye High School of Science - 2011 - GPA: 87.20/100

WORK EXPERIENCE

Research Assistant : T.C. Ankara Yildirim Beyazit University - April 2017 - Present

TOPICS OF INTEREST

- Machine Learning
- Cloud Computing
- Computer Network
- Data Mining

December 2018
ANKARA

Department of Computer Engineering

Ahmet Esad TOP

