UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

DEEP LEARNING APPROACHES IN PROBLEMS IN VARIOUS-DIMENSIONAL DATA

A DISSERTATION

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

DOCTOR OF PHILOSOPHY

By

Lei Yang
Norman, Oklahoma
2017

DEEP LEARNING APPROACHES IN PROBLEMS IN
VARIOUS-DIMENSIONAL DATA

A DISSERTATION APPROVED FOR THE
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

BY

Dr. Samuel Cheng, Chair

Dr. Pramode Verma

Dr. William Ray

Dr. Cliff Chan

Dr. Bin Zheng

Acknowledgements

First of all, I wish to express my greatest gratitude to my advisor, Prof. Samuel Cheng, for his guidance, unconditional support, and for being an excellent mentor during my Ph.D study. I am grateful for having had the opportunity to work with him.

I am forever grateful to everyone who has made my time in graduate school at the University of Oklahoma. I appreciate my internship days at MD Anderson Cancer Center, Department of Biomedical Informatics at University of California San Diego, and SAIC-USA research department. My thanks go to Prof. Jingfei Ma, Prof. Shuang Wang, Prof. Xiaoqian Jiang, Dr. Rakesh Gupta, Dr. Jerry Yu and Mike Xie, for offering the great opportunities for me to enlarge my vision.

I would like to thank my wife Yang and my parents, for all the love, patience and understanding. This thesis is dedicated to them with my sincere gratitude.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Machine learning lies at the heart of a broad range of application fields, it help us solve many specific problems of practical and commercial interests, such as spam filter [1], gene selection [2], credit risk assessment [3], face detection and recognition [4] and even enable autonomous driving vehicle [5]. It is often observed that the raw inputs of these applications are characterized with recorded facts, such as pixel intensity value in images and videos. However, much of the raw data is unstructured and need to be transformed to a set of new features. One hypothesis is that different representations can entangle and hide more or less the different explanatory factors of variation behind the data [6], an appropriate feature will help uncover the associated degree of information form that data. As a result, a good feature should be able to disentangle the factors of variation that inherently explain the structure of the distribution [7].

Due to the above mentioned reasons, feature engineering, which focuses on transforming raw data into features that better represent the underlying problem to the model [8], has become an vital part of machine learning for a long time. In general, there are two major types of feature engineering: the automatic extrac-

tion and handcraft designing approaches. The former one are typically used in feature selection. For example, to address the problem of curse of dimensionality which tends to overfit the training data in high dimensional spaces, PCA and its variants [9, 10, 11] were proposed to used as dimensionality reduction for feature extraction. PCA learns a linear transformation $f(x) = W^T x + b$ of input x, which leads to a group of orthogonal decorrelated features. On the other hand, handcrafting feature engineering turn our inputs into something the algorithm can understand produces substantial performance gains. For instance, the Histogram of Oriented Gradients (HOG) [8] and Scale Invariant Feature Transform (SIFT) [12] are popular feature engineering algorithms developed specifically for the computer vision domain. However, most feature engineering require professional data science experience, specific domain knowledge, trial-and-error with substantial effort, and generally is time-consuming. Moreover, feature crafting is not a technique we can apply universally and expect to get good result. Most critically, given a real-world problem, it is often not clear what the optimal feature representation should be manually designed.

1.1 Learning Representation

In some cases, hidden information inside the data (or instance) is more useful. Hence, a great deal of research has focused on learning good feature representations from the data in supervised or unsupervised manner. For instance, K-means clustering was used as an form of unsupervised representation learning: it begins with k arbitrary centroids $c^{(k)}$ and adjust these centroids from input data. It is extremely fast and has no hyper-parameters to tune except the model structure itself. These two properties make it appealing in practice. Sparse coding [13] is

another widely used technique, it find succinct representations of data in which each of the components of the representation is only rarely significantly active. In general, feature learning is often a very attractive alternative to manually crafted feature.

Traditionally, the majority of existing methods have been applied to learn relatively shallow (one or two layers) representations, which results in a limited expressive power. From the theoretical standpoint, universal approximation theorem [14] suggests that a model with deeper layer can be exponentially more efficient than shallow model. Therefore, there is a desire to investigate deeper representations.

1.2 Data and Deep Learning

Recent researches have shift the attention of feature learning to the deep layers and non-linearity of data representations. As we discussed earlier, the hidden information in data is more useful than raw feature most time. Deep learning is exactly such a hierarchical feature learning module, and the hierarchical representations is one of the major advantages over classical shallow feature engineering. Deep learning exploring the hidden multiple layers of highly non-linear representations, each layer in a multi-layer neural network can be seen as a representation of the input through a learned weights, the higher layer parameters formed by the composition of lower ones.

Although we have discussed the importance of data representation individually, we do not suggest that they are the sole source of improving the performance of model. Rather, it is likely that they are highly interconnected with the classifier we employ in the model. From this standpoint, deep learning is indeed

advanced in which we can jointly learn the features along with the classifier. More specifically, by adding a classification layer, we jointly combine the learning and prediction tasks in a single forward-passing network. On the other hand, non-linearity is another appealing property. For the past decades, a large number of studies have demonstrated that data representations obtained from stacking up non-linear feature extractors often yield better learning results.

Besides the theoretical justifications, one of the important reasons for the popularity of deep learning is the exponential growth of both available data and computational power. When it comes to making progress on the accuracy result, most researchers are mainly concerned with design more elegant and sophisticated model. In modern times, however, we are overwhelmed with data, the alternative path may be just collect more data. For example, ImageNet [15] containing the 1000 categories and 1.2 million high-resolution images and has become one of the most important benchmark in computer vision. As a result, the vast majority of image recognition works [16, 17, 18] directly benefit from this enriched datasets and usually train their models with a pre-trained parameters based on ImageNet. At the same time, the emergence of graphics processing units (GPU) enable us design much deeper layers network. The gradient calculation on a GPU runs much faster compare to the CPU. In summary, the limitation due to the memory resources and computation power has been significantly improved.

1.3 Overview of Chapters

This thesis develop efficient deep learning based methods for a series of tasks. I believe these problems to be inherently worth developing because of their connections to problems of practical importance. In particular, we predict the state

of pain data, address the problem of visually locating two-dimensional text objects in natural scenes, and deal with LIDAR-based point cloud car detection in autonomous driving respectively. We now provide an overview of the methods and results which are considered by subsequent thesis chapters. The introductory paragraphs of each chapter provide more detailed outlines.

Chapter 2: Deep Neural Network

We begin by briefly reviewing the background of general Forward Neural Network and the concepts of gradient based learning and back-propagation. This chapter end with the limitations which traditional neural network suffers from.

Chapter 3: Pain Data Management, a Case Study

In this chapter, We first introduce the terminology and concepts of Restricted Boltzmann Machines, and provides detailed training method used extensively in later chapters. Turning to pain management issue, we using discriminant RBM to jointly predict the pain state and compare the performances with linear dimensionality reduction approaches. We conclude by validating RBM's performance in 4 customized patients data.

Chapter 4: Deep Neural Network for Text Detection

The fourth chapter applies the variant of Convolutional Neural Network to visually detect text object in wild scene. We begin with a detailed explain of ConvNet. The sliding windows based method is arguably the most popular method in object detection, thus, we discussed the ConvNet applied in sliding-windows fashion, underlying the improved detection accuracy due to the better learned

representation. In order to speed up localization, we take a look at popular saliency models and incorporate this concept into the ConvNet.

Chapter 5: 3D Object Detection in LIDAR-based Point Clouds

This chapter focuses on methods for robustly learn 3D object appearance models. In particular, we devote to LIDAR based point cloud object detection used in autonomous driving system. Considering the challenges of directly using 3D data, we first project 3D point clouds into 2D image plane: top-down bird view and front view cylinder plane. Then we propose two potential 3D models to addressing 3D objects detection for future investigations.

Chapter 2

Deep Neural Network

> What magical trick makes us intelligent? The trick is that there is no trick.
>
> *Marvin Minsky*

Deep learning based methods play an important role in various computational fields and also enjoy tense commercial attention. For example, AlphaGo's victory over Lee Sedol [19] is a very impressive achievement among a great deal of other successful applications. But there is no magic in deep learning, no fancy novel conceptions with respect to the "old" neural network. Shallow and deep are only distinguished by the depth of network.

Traditionally, to solve a classification problem, one first designs an appropriately efficient feature and classifier which can learns a mapping between features and a predefined set of classes. If we cannot explicitly specify the rules how we design the feature, then we cannot solve the problem. Neural network extends what we can do with the training data even when we are unable to state precisely what the feature should be.

Our aim in this chapter is to describe the fundamental structure of neural network. As the building blocks of neural network, we first introduce two basic neurons: perceptron and sigmoid neuron. Once a network is constructed with these neurons, we'll equip it with back-propagation or gradient-based learning rules. Vanishing gradient [20] arise when using neural network on real world problems. We review this problem from the viewpoint of activation functions and discussed about the saturation of activation functions. To grasp a basic understanding and appreciation the strength and limitation of neural network, we'll begin with a brief review of history of neural network.

2.1 Brief History of Neural Networks and Applications

In recent years, deep neural networks have won almost all contests in machine learning. The key paradigm of artificial neural network is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working together to solve specific problems.

The first step toward neural network trace back to the 1940's when McCulloch and Pitts introduced their first computing model of a neuron [21]. The model had two inputs and one output, the weight for each input was equal, and the output was binary. In the 1950's, Rosenblatt's work [22] first introduced a two-layer network, the perceptron, which used trial-and-error to randomly change the weights in order to achieve learning. It was built in hardware and the oldest NN still in use today. In the 1960's, it seemed that perceptrons could solve any

problem. However, although the perceptron was successful in classifying certain patterns, it had a number of disadvantages. For example, one of the limitations is that the perceptron is a single node, it could only be able to solve the linearly separable problems (linear separable problem). Consequently, such limitation leads to the decline of the field of neural networks.

In the following decades, neural network earned the popularity in the 1980s due to the introduction of back-propagation algorithm [23]. Back-propagation using stochastic gradient descent for learning functions mapping low-dimensional dense vectors to other low-dimensional dense vectors. Unfortunately, it lost favour in machine learning community after 1990's. There is no clear understanding of why deeper models were not preferred by the computer research community. One direct cause might be the deeper models failed proving as capable as high expectations. Another potential reason may due to the universal approximation theorem [14] which states shallow layer is able to model any functions. Furthermore, the training difficulties and hurdles due to small training data, limited computation power and vanishing gradient problem together forced computer practitioners focus on more elegant shallow models such as kernel methods [24].

In 2006, an impressive layer-by-layer pretrained method proposed by famous cognitive psychologist and computer scientist Geoffrey Hinton [25]. His work renewed interests in deeper network by providing a new way to initialize their weights before the training process and demonstrated exceptional results. Along with the improvement on training mechanisms [26] and better choice of ReLU non-linearity [27] and new model regularization strategies such as dropout layer [28], deep learning has made a tremendous impact in a wide range of computer fields. For instance, in computer vision community, the researchers using deep learning techniques achieved previously unattainable performance on many tasks

such as image classification, objects detection, object localization, object tracking, pose estimation, image segmentation and image captioning [29, 17, 30, 31].

2.2 Biological Inspiration

Our brain extracts multiple levels of representations from sensory inputs [32]. Typically, animal are able to solve extremely difficult learning problems. Our brains are made up of billions of neurons, that send electrical signals to one another. The rate of firing tells us how "activated" a neuron is. A single neuron might have multiple incoming neurons, these incoming neurons are firing at different rates (i.e., have different activations). Based on how much these incoming neurons are firing, and how strong the neural connections are, our main neuron will decide how strongly it wants to fire. Learning in the brain happens by neurons becoming connected to other neurons, and the strengths of connections adapting over time.

To a certain extent, some artificial neural networks are models of biological neural networks and some are not, but historically, much of the inspiration for the artificial NNs came from biology and human cognition. It appears that artificial neural network are mimic biological network, for example, the primitive features learned by visual neural network are similar to those found in early visual processing stages of human visual system, such as edges or corners. However, it is important to note that the real biological world is much more complicated than this and our goal isn't to build a brain. From this point of view, deep learning is only loosely based on how the brain works. At the same time, it is observed that artificial neural network can help to better understand biological neural networks [33].

2.3 Perceptron

The perceptron sometimes called single-layer neural network, it is typically used for classification, Figure 2.1 is a graphical illustration of a perceptron with n inputs.

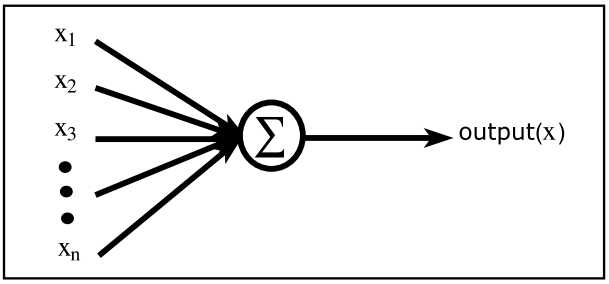


Figure 2.1: A perceptron with n inputs $x_1, x_2, ...x_n$

Perceptron learns a binary classifier by defining a function that maps its input $\mathbf{x}(x_1, x_2, ..., x_n)$ to a binary output value. Let's $\mathbf{w}(w_1, w_2, ..., w_j)$ be a vector of real-valued parameters or weights which denote the importance of the respective inputs to the output. The neuron's output, 0 or 1, is determined by whether the weighted sum $w_j x_j$ is less than or greater than a threshold value. The threshold is a real number and another parameter of neuron. More precisely, it can be represented as follows:

$$output(x) = \begin{cases} 0 & \text{if } \sum_{j=1}^{n} w_j x_j \leq threshold \\ 1 & \text{else} \end{cases} \tag{2.1}$$

We can also show the decision boundary graphically with respect to 2 dimensional parameter spaces (Figure 2.2). The decision boundary defined as: $\mathbf{w} \cdot \mathbf{x} + \mathbf{b} = 0$, where $\mathbf{b}$ is bias and $\mathbf{b} = -threshold$. The network output will be 1 for the region above and to the right of the decision boundary. The graph shows the perceptron

is simply linearly separate the inputs into 2 categories.

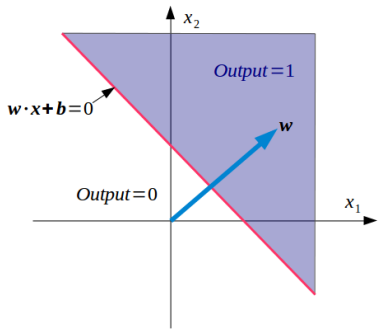


Figure 2.2: Decision boundary in 2-dimensional parameters space

Notice that a small change in the parameters of any single perceptron in the network can sometimes cause the output of that perceptron to completely flip from 0 to 1 or from 1 to 0. Furthermore, the decision boundary between the categories is linear. These properties make the perceptron network could solve only a limited category of problems. Fortunately, this issue can be solved by introducing a new type of artificial neuron called sigmoid neuron.

2.3.1 Sigmoid Neuron

Sigmoid neurons are similar to perceptrons, but modified so that small changes in their weights and biases cause only a small change in their output. Similar with Figure 2.1, the sigmoid neuron has the same inputs, but instead of being just 0 or 1, these inputs can also take any values between 0 and 1. Similarly, the output is $\sigma(\mathbf{w} \cdot \mathbf{x} + \mathbf{b})$ instead of 0 or 1, where σ is an activation function or transfer function and is defined by:

$$\sigma(z) \equiv \frac{1}{1 + e^{-z}} \tag{2.2}$$

More precisely,

$$\sigma(x, w, b) \equiv \frac{1}{1 + e^{-\sum_j w_j x_j - b}} \tag{2.3}$$

The exact form of σ isn't so important - what really matters is the shape of the function when plotted in Figure 2.3. If we substitute σ function with a step

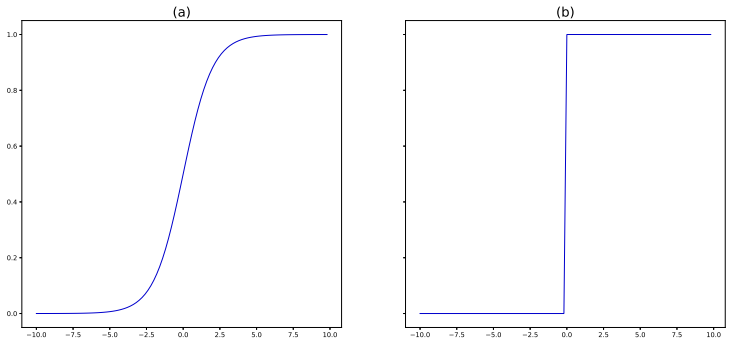


Figure 2.3: The shapes of Sigmoid and Step function: (a) Sigmoid function, (b) Step function

function, then the sigmoid neuron would be a perceptron. From this perspective, Sigmoid function is a smoothed out perceptron. The smoothness of σ means that small changes Δw_j in the weights and Δb in the bias will produce a small change $\Delta output$ in the output from the neuron. We get:

$$\Delta output \approx \sum_j \frac{\partial output}{\partial w_j} \Delta w_j + \frac{\partial output}{\partial b} \Delta b \tag{2.4}$$

$\Delta output$ is linear function of the changes Δw_j and Δb in the weights and biases.

Non-linear Transfer Functions

Sigmoid function is the standard way to model a neuron's output f as a function of its input x. In addition to the historical reason, there are a number of nice

characteristics: it's smooth, continuous, and monotonically increasing. More importantly, the exponential function has nice properties when it is differentiated. More precisely:

$$\sigma'(x) = \sigma(x)(1 - \sigma(x)) \tag{2.5}$$

This property leads to faster computation. Note that sigmoid function squash the input to the range of $(0, 1)$, thus, it can be interpreted as probability in certain extent. Another common choice for activation $f(x)$ is the hyperbolic tangent function, which has the form: $tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$. The $tanh$ squashes a real-valued number to the range $[-1, 1]$. In practice, the $tanh$ non-linearity is slightly superior to the sigmoid nonlinearity due to its output is "zero-centered". In fact, almost any non-linear real-valued differentiable (almost everywhere) function can be used as an activation function in neural nets.

2.4 Architecture of Feedforward Neural Network

With many of these simple, connected sigmoid neuron, we can construct a standard feedforward neural network (FNN). FNN is a network where connections between the units only pass forward and do not form a directed cycle. There may be some shortcut connections between distant layers. In FNN, the output from one layer is used as input to the next layer, the information moves in only forward. Figure 2.4 is a four-layer network with two hidden layers, one input layer and one output layer.

In principle, FNN has the power of a universal approximation, i.e. it can realize an arbitrary mapping of one vector space onto another vector space [34].

As an alternative to standard FNN architecture, cyclic neural networks (Fig-

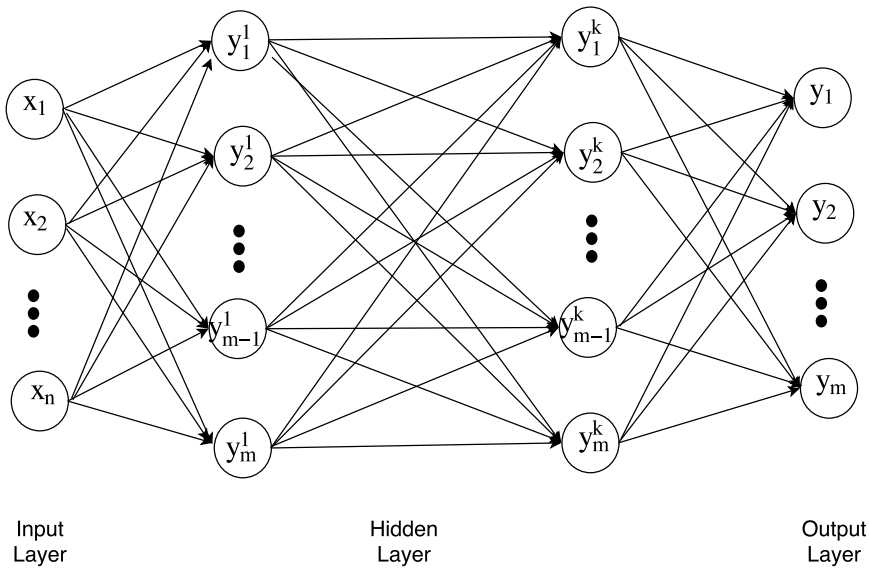


Figure 2.4: The architecture of a standard feedforward neuron network

ure 2.5) such as Recurrent neural networks (RNNs) and its variant Long Short-Term Memory (LSTM) are widely-used in a number of fields [35, 30, 36, 37] and have been demonstrated great success. This network contain cyclic connections that make them a more powerful tool to model sequence data (e.g., video) than FNNS. To a certain extent, they are the deepest of all networks. We restrict our attention to the FNNs throughout this thesis.

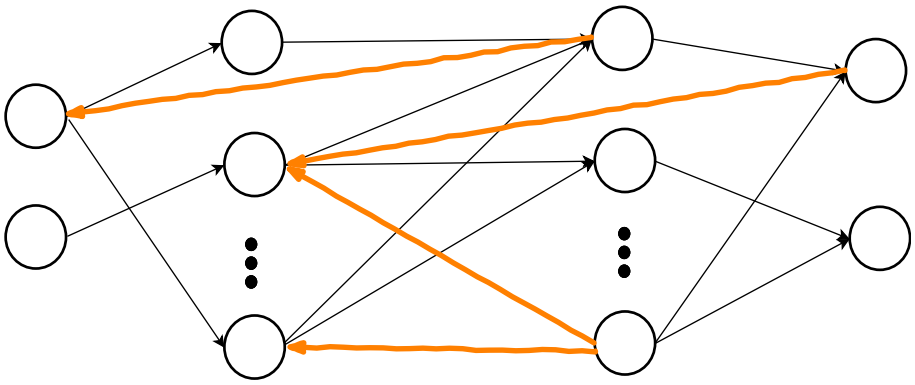


Figure 2.5: The architecture of a cyclic neuron network

2.4.1 Training Single Layer with Gradient Descent

Computational effort needed for finding the correct combination of parameters. In order to adjusting the weight and bias coefficients in a network, the vast majority of strategies use supervised learning. We will use error-correction learning which compare the network output to the ground truth output value, and using that error to direct the training. Consider a FNN with n training set $\{(x_1, a_1), (x_2, a_2), \cdots, (x_n, a_n)\}$, Let's define the mean squared error cost function:

$$C(x, a) = \frac{1}{2n} \sum_{x} \|y(x) - a\|^2 \tag{2.6}$$

Here, $y(x, w, b) = wx + b$ is the output vectors from the network which corresponding the input vectors x, and the sum is over all training inputs. The optimization of cost function will be addressed by gradient descent method.

The gradient descent algorithm has a variety of uses in various fields [38, 39]. It works by taking the gradient of the parameter space to find the path of steepest descent. The essential step is computing the partial derivative. Mathematically, we iteratively applying the update rule in $k - th$ step:

$$w_{k+1} = w_k - \eta \frac{\partial C}{\partial w_k} \tag{2.7}$$

$$b_{k+1} = b_k - \eta \frac{\partial C}{\partial b_k} \tag{2.8}$$

Where η is a real-valued hyperparameter called learning rate. It is a common parameter and determines how much the weights can change in response to an observed error on the training set. η has a significant effects on the network's generalization accuracy as well as the training speed. Much effort has been devoted to automatically tuning the η value [40].

The gradient of the cost function always shows in the direction of the steepest ascent of the function (Figure 2.6). Thus, theoretically we can start with a random parameter initialization. It is important to note that there is no guarantee the gradient descent will find a global minimum for problems with non-convex loss function. We can use method such as simulated annealing [41] to help find the true global minimum although that may take a very long time to finish the process. There is another problem from the perspective of efficiency, from the form of Equation 2.6, the cost function is an average over cost of individual training examples. Therefore, when the number of training inputs is large the process can be very time-consuming, and this learning strategy thus inefficient.

2.4.2 Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is an iterative optimization algorithm and can be used to accelerate learning process. The basic idea is quite simple: we re-write the Equation 2.6 as:

$$C(w,b) = \frac{1}{2n}\sum_{x} C_x \tag{2.9}$$

Where C_x is the individual cost for an sample x. SGD estimates the gradient ∇C by computing ∇C_x for a small number of randomly chosen training samples. By averaging over this small sample it turns out that we can quickly get a good estimate of the true gradient ∇C, and this helps speed up learning. More precisely, suppose we randomly picking out a small number m out of N training inputs. We'll label them $x_1, x_2, ..., x_m$, and refer them as a mini-batch. The stochastic

gradient descent rule is as follows:

$$w_{k+1} = w_k - \frac{\eta}{m} \sum_{j=0}^{m-1} \frac{\partial C_{x_j}}{\partial w_k} \tag{2.10}$$

$$b_{k+1} = b_k - \frac{\eta}{m} \sum_{j=0}^{m-1} \frac{\partial C_{x_j}}{\partial b_k} \tag{2.11}$$

Where the sums $\sum_{j=0}^{m-1} \frac{\partial C_{x_j}}{\partial w_k}$ and $\sum_{j=0}^{m-1} \frac{\partial C_{x_j}}{\partial w_k}$ are over all the training examples x_j in one mini-batch. Then we pick out another randomly chosen mini-batch and train with them, until we've exhausted updating all the N training inputs to complete one epoch of training. In practice, SGD has been shown to be effective for a variety of problems.

2.4.3 Back-propagation

Deep architecture by default contains multiple layers. The back-propagation was the first computationally efficient model of how neural networks could learn multiple layers of representation. It was developed and redeveloped multiple times in a various fields [23, 42].

Ultimately, back-propagation computing the partial derivatives $\frac{\partial C}{\partial w^l_{jk}}$ and $\frac{\partial C}{\partial b^l_j}$, where w^l_{jk} means the weight from k-th neuron in $(l-1)$ layer to j-th neuron in l layer, b^l_j denotes the bias for jth neuron in l layer. Intuitively, we will encounter a difficulty when we keep calculating errors in the hidden layers. It is impossible to compute the error signal for hidden neurons directly, because output values of these neurons are unknown. The idea is to propagate error back to all neurons. For the hidden layer i, the next layer k[1], each unit computes weighted sum of its

[1]For the simplicity, we do not consider the biases in calculating the partial derivative with respect to the weights.

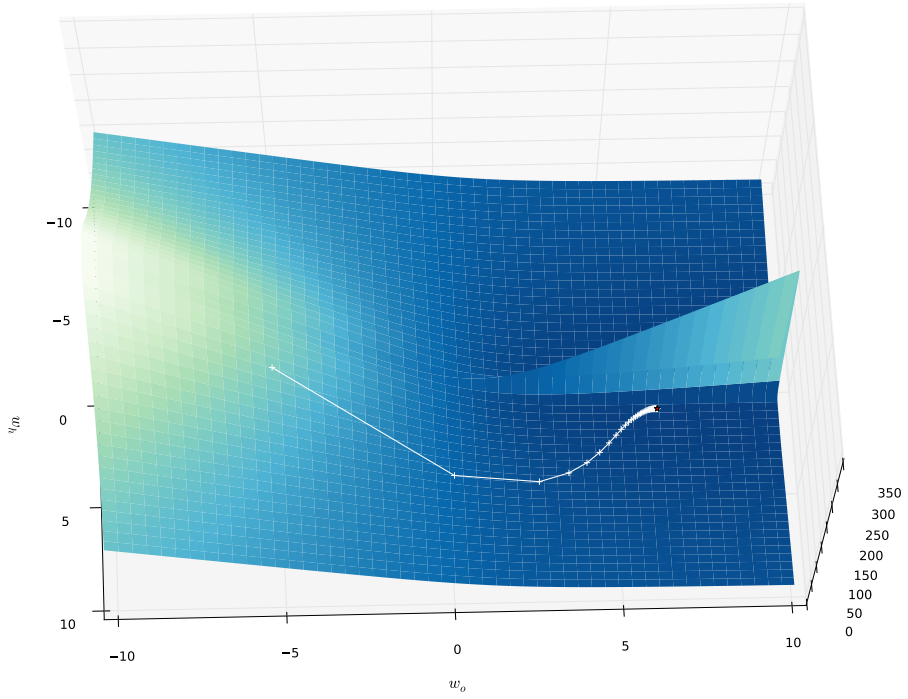


Figure 2.6: Gradient descent optimization on the surface of cost function

inputs:

$$a_j = \sum_i w_{ji} z_i \tag{2.12}$$

where z_i is activation of a unit that sends a connection to unit j and w_{ji} is the weight associated with the connection. Transformed by nonlinear activation function

$$z_j = f(a_j) \tag{2.13}$$

Using chain rule for partial derivatives, we have:

$$\frac{\partial C}{\partial w_{ji}} = \frac{\partial C}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}} \tag{2.14}$$

We define $\delta_j = \frac{\partial C}{\partial a_j}$, and Equation 2.12 suggests $\frac{\partial a_j}{\partial w_{ji}} = z_i$. Assuming the $k-th$ layer is the output layer and the network output is y_k and corresponding ground truth is t_k, the term δ_k is straightforward:

$$C = \frac{1}{2} \sum_k (y_k - t_k)^2 \tag{2.15}$$

Due to $y_k = a_k$, using the definition of δ, we have:

$$\delta_k = y_k - t_k \tag{2.16}$$

For hidden unit j, by chain rule,

$$\delta_j = \frac{\partial C}{\partial a_j} = \sum_k \frac{\partial C}{\partial a_k} \frac{\partial a_k}{\partial a_j} \tag{2.17}$$

From $a_k = \sum_j w_{kj} f(a_j)$, we obtain $\frac{\partial a_k}{\partial a_j} = \sum_k w_{kj} f'(a_j)$. Using the definition of

δ_k, we get the back-propagation formula for error derivatives at j layer:

$$\delta_j = f'(a_j) \sum_k w_{kj} \delta_k \tag{2.18}$$

From the above formulas, the value of δ for a particular hidden unit can be obtained by propagating the δ backward from units with higher layer in the network. The partial derivative with respect to bias is comparatively easy. The output for every neuron is that: $a_j = \sum_i w_{ji} z_i + b_j$, using chain rule, $\frac{\partial C}{\partial b_j} = \frac{\partial C}{\partial a_j} = \delta_j$.

For each training example, the standard back-propagation algorithm can be performed by two steps:

Step 1, **Forward propagation**

- Initialize the network with random weights and biases.
- Feed in an example x to the network and forward propagate through network using: $a_j = \sum_j w_{ji} z_i$ and $z_j = f(a_j)$.

Step 2, **Back-propagation of error**

- Evaluate the error δ_k for all output neurons using: $\delta_k = y_k - t_k$.
- Back-propagate the δ using $\delta_j = f'(a_j) \sum_k w_{kj} \delta_k$ to obtain δ_j for each hidden neuron.
- Update the weights by: $\frac{\partial C}{\partial w_{ji}} = \delta_j z_i$.
- Repeat with other examples until the network converges on the target output.

2.5 Limitations of Training Deep Network using Back-Propagation

Many researchers reported positive experimental results with typically one or two hidden layers, but training deeper networks consistently yielded poor results[2]. Training a deep network is difficult in general because the problem is highly non-linear and non-convex. In particular, there are two most well-known problems that arise in back-propagation learning process: local minimum and vanishing gradient.

2.5.1 Local Minimum

Gradient descent can get trapped in local minimum due to the cost functions have many local minimum (Figure 2.7). Hence, training tends to converge to a local minimum. To overcome the local minimum problems, many methods have been proposed. A widely used one is to train a neural network more than once, starting with a random set of weights [43]. Consequently, this strategy leads to more expensive computational cost by nature.

The learning rate is another factor to effect local minimum. Momentum [44] is a technique that can help the network alleviate this problem. Again, there is no guarantee we can converge to the global minimum in neural network.

2.5.2 Vanishing Gradient

Another issue the learning process suffered from when training a neural network is vanishing gradient problem. In particular, features in earlier layers could not

[2]convolutional network is one notable exception

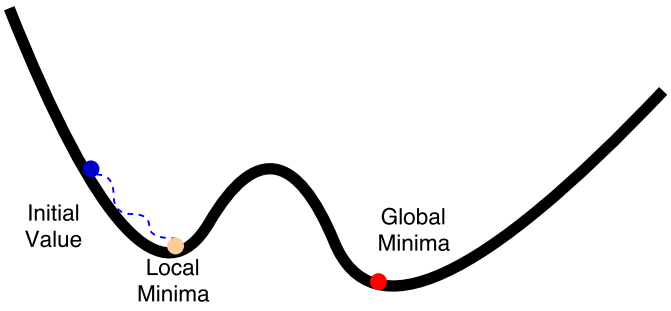


Figure 2.7: An example on local minimum on 1-d data. The back-propagation will find the local minimum instead of global minimum with the certain initialized value.

be learned because no learning signal reached these layers. Furthermore, this problem becomes worse as the architecture of network goes deeper. Notice that almost all the gradient-based learning methods suffer from this problem.

The direct cause is intuitive, back-propagation learn a parameter's value by understanding how a small change in the parameter's value will affect the network's output. If a change in the parameter's value causes very small change in the network's output - the network just can't learn the parameter effectively, which is a problem. The gradients of the network's output with respect to the parameters in the early layers become extremely small. Activation functions make great effects on this problem.

Discussion: Saturation Property of Activation Functions

Recent researches [45] have been demonstrated that the logistic sigmoid activation (or tanh function) is unsuited for training deep networks with random initialization because of the issue of saturation. Consider an activation function $f(x)$ and its derivative. If $f(x) \to \infty$ When $x \to \infty$ ($x \leftarrow -\infty$), then f is defined to be a right (left) non-saturation. An activation function is said to non-saturate

if it both left and right non-saturates [46]. Otherwise, the function is considered as a saturate function.

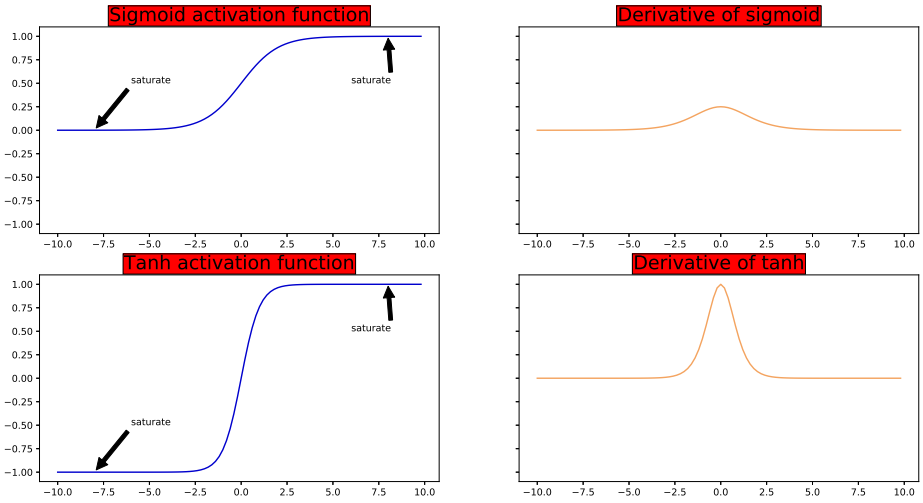


Figure 2.8: Illustration of the derivatives of sigmoid and tanh functions, both of them are saturate on the left and right side.

Sigmoid activation function is a saturate function, and it maps the real number line onto a "small" range of [0, 1] (Figure 2.8). As a result, there are large regions of the input space which are mapped to an extremely small range. In these regions of the input space, even a large change in the input will produce a small change in the output - hence the gradient is small.

There are other benefits from non-saturating nonlinearity in addition to application in the vanishing gradient problem. When it comes to training time with gradient descent, these saturating nonlinearities are much slower than the non-saturating nonlinearity such as Rectified Linear Units (ReLU). ReLU (Figure 2.9) is defined as $f(x) = max(0, x)$ and it's fast because the derivation simply involve thresholding at 0. Many works demonstrated model with ReLU units converge

faster than saturate functions. For example, deep convolutional neural networks with ReLUs train much longer than their equivalents with tanh units [27]. How-

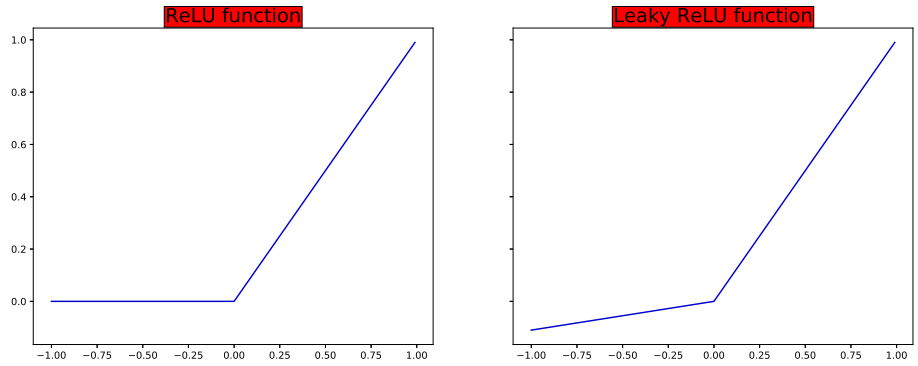


Figure 2.9: Illustration of ReLU and leaky ReLU activation functions.

ever, ReLU can 'dying'[3] during optimization because the derivative is 0 whenever the unit is not active. This could lead to cases where a unit never activates as a gradient-based optimization algorithm will not adjust the weights of a unit that never activates initially. Leaky ReLU (Figure 2.9) was an alternative to fix the 'dying' issue by allows for a small, non-zero gradient when the unit is saturated and not active [47]. Instead of the function being zero when $x < 0$, a leaky ReLU will instead have a small negative slope, more precisely:

$$LeakyReLU(x) = \begin{cases} ax & \text{if } x < 0 \\ x & \text{else} \end{cases} \tag{2.19}$$

where a is a small constant and a default value would be 0.01.

[3]Neurons are not activated regardless of inputs.

2.6 Summary

We briefly introduce the most fundamental building blocks for constructing a neural network. We will touch the first and most widely studied deep model: Deep Belief Net (DBN). DBN is proposed by Hinton [25], obtained by training and stacking several layers of Restricted Boltzmann Machines (RBM) in a greedy manner. Once this stack of RBMs is trained, it can be used to initialize a multi-layer NN for classification. RBM computing the conditional posterior over the stochastic hidden layers is intractable. Learning and inference is more tractable in single layer RBM but it suffers from lack of representational power. Therefore, energy based models are used to give RBM a representation. We will describe energy-based model, RBM, together with DBN in detail in the following chapter.

Chapter 3

Pain Data Management, a Case Study

At the end of last chapter, we discussed the hurdles the traditional back-propagation faced when train a deep FNN. Much researches have been devoted to overcome these limitations [48]. The breakthrough was achieved in 2006 when paper [25] proposed an effective layer-wised unsupervised strategy to train a Deep Belief Network (DBN). The whole network is created by stacking several layers of Restricted Boltzmann Machines (RBM) [49] in a greedy manner.

RBM is a generative approach to model an input distribution, and regained the popularity since the breakthrough in DBN. Since 2006, RBMs have been particularly successful in classification problems such as model human motion [50], phone recognition [51]. Our emphasis in this chapter will be the inference of pain state using discriminant RBM.

Pain management is inherently worth investigating. It is common and is reported by more than half of hospitalized patients [52]. Accurately assessing pain for those who cannot make self-report of pain, such as minimally responsive

or severely brain-injured patients, is challenging. In this chapter, we attempted to address this challenge by answering the following questions: (1) if the pain has dependency structures in electronic signals and if so, (2) how to apply this pattern in predicting the state of pain. To this end, we have been investigating and comparing the performance of several machine learning techniques. In particular, we first adopted different strategies, in which the collected original -dimensional numerical data was converted into binary data. Pain states are represented in binary format and bound with above binary features to construct -dimensional data. We then modeled the joint distribution over all variables in this data using the RBM. The experimental results show that RBM is able to model the distribution of our binary pain data. In addition, we show that discriminant RBM can be used in classification task, and the initial result is competitive with respect to other classifier such as support vector machine (SVM) using PCA representation and LDA discriminant method.

3.1 The Pain State Prediction Problem

Pain management is very important on patient care, and more than half of hospitalized patients have reported pain. In America, chronic pain affects about 100 million people [53]. Pain and its associated problems are among the leading public health problems in the US [54]. Although pain assessment guidelines are available, pain management are still deemed insufficient as reported by many patients and health professionals [55]. Pain management relies on the ability to accurately assess when, how and to what extent a patient is experiencing a pain. As a subjective phenomenon, the severity of the perceived pain may vary significantly among different patients. Thus a patient's self-report is usually treated

as the most reliable pain measurement [56]. Various non-physiological factors such as emotional state, environmental and socioeconomic contexts [57, 58, 59] may also have an impact on pain assessment, which makes it a non-trivial task to accurately assess pain.

The multidimensional pain theory [60], proposed by Melzack, categorizes pain based on non-observable (i.e., subjective), and observable (i.e., objective) indicators. For example, patient's self-reports of pain that include sensory, emotional, and cognitive components of pain are subjective information, which can serve as non-observable indicators. Nurses often assess and document this information. Observable indicators include the physiological and behavioural components of pain, which are usually captured and documented in critical care settings through continuous monitoring. The behavioural components are actively applied in behavioural observational pain scales. The physiological signals should also help healthcare professionals better perform the pain assessment. Although many studies have attempted to associate physiologic signals and pain, few practical and reliable methods of using physiological components in pain assessment is available. In this study, we attempted to assess the probability of pain presence based on other physiological data. This approach can particularly be useful for caring minimally responsive patients who cannot make self-report of pain.

A great deal of effort has been made to analyze pain. Recently, there has been increasing interest in exploring the task of predicting pain state using machine learning techniques. Generally, the task of prediction requires discovering (learning) patterns in training data. A good data model to represent the distribution of training data is critical in this process. [61] has successfully done using the electronic flow-sheet data of ICU patients collected for a limited time interval. These time series data were projected into a lower-dimensional subspace, and a num-

ber of data vectors (within a time window size) were represented (reconstructed) with some linear combinations of principal components (PCs). The magnitude of residual between original and reconstructed data can be used to measure the level of pain. In this previous study, we did not utilize any label (i.e., documented pain presence) from the data. We could interpolate the missing data by considering the time information when the recorded labels are incomplete. On the other hand, ignoring all the labels may result in loss of significant information. The overall approach of the study we report here is different from that of the previous one in that: (1) we focus on learning from the labeled data, (2) we treat our data as non-temporal rather than temporal data, and (3) we focus on investigating the relationships between activation of pain and the normal/abnormal state of various electronic signals. Consequently, the problem we need to solve is transformed into a supervised learning and classification problem.

In machine learning, linear methods [62] are fast and robust which successfully avoid the over-fitting problem with its simple model. In addition, they are guaranteed to produce a global optimum. However, their performance are often limited in addressing the real problem data which generally no linear boundary exist. In this study, we employ an alternative non-linear model restricted Boltzmann machine, to represent the original data v with lower-dimensional h. We trained RBM with the labeled data and feature vectors in a supervised manner. Both the feature and the labels are visible units in the model. Moreover, using the nature of this model, discriminant RBM can be used in the classification. The probability of unknown label class can be calculated through free energy when a new input was fed into RBM. In the rest of this chapter, we will present the proposed framework of our new pain prediction algorithm called PATTERN: Pain Assessment for paTients who can't TEll using Restricted Boltzmann machiNe.

3.2 Data Representation and Restricted Boltzmann Machine

The success of a classification algorithm highly depends on the choice of representation for data. One hypothesis is that different representations can more or less entangle and hide the different explanatory factors of variation behind the data [63, 6]. An attractive alternative is to estimate the parametric distribution, which explains the data best, for example, Gaussian Model and Gaussian Mixture Models. Another concern is the high data dimensionality. Higher dimensional data can provide richer and more detailed information than lower dimensional one simply because they provide more dimensional candidates; at the same time, learning from the high dimensional data often suffer from over-fitting problem. In other words, with an insufficient number of data points in the training set, we tend to memorize each data point rather than learn from it. To avoid over-fitting, some classical approaches typically project the data into lower-dimensional space, such as principal components analysis (PCA). Recently, using RBMs to model the data have been reported in a large variety of learning problems [64, 65, 66]. Theoretically, the capacity of RBMs has been demonstrated that it can provide a powerful means to represent data [67].

Boltzmann machines are important bridges between neural networks and probabilistic undirected graphical models and could be one way of implementing feedback in neural networks. In particular, it is also known as Markov random fields. Thus, we first give a brief introduction of Markov Random Field.

3.2.1 Markov Random Field

Probabilistic graphical models are an intuitive way of representing and visualizing the conditional dependence relationships between many variables in a joint distribution. There are two kinds of graphical models: those based on directed graphs and those based on undirected graphs. Our focus will be the latter one.

Markov Random Field (MRF) [68] is undirected graphical model. A graph is $\mathcal{G} = (\nu, \varepsilon)$ is a collection of vertices, $\nu = \{x_1, x_2, \cdots x_N\}$ and ε is a set of undirected edges. Two nodes x_i, x_j is neighbor if $(x_i, x_j) \in \varepsilon$. A clique is a set of random variables c, in which all nodes are neighbors of each other. A clique is maximal if it cannot be extended any larger in that graph. For example, Figure 3.1 illustrate an example of MRF, there are 6 random variables on the graph, the corresponding maximal cliques are grouped in the right circles. MRF is this undirected graph with each edge denotes certain conditional dependency between variables and satisfy Markov property, where Markov property suggests that given its neighborhood, a node is independent on the rest of the nodes in the graph. In other words, the joint distribution of X is determined entirely by the local conditional distribution, and the local distribution of node can be denote as:

$$p(x_i|x_{\nu\setminus i}) = p(x_i|neighbor(i)) \tag{3.1}$$

3.2.2 Learning in MRF

Unsupervised learning means learning an unknown distribution q based on sample data. Training corresponds to finding the parameter θ that maximize the likelihood L given the training data $\mathbf{x} = x_1, x_2, \cdots, x_n$. More precisely, $L(\theta|\mathbf{x}) =$

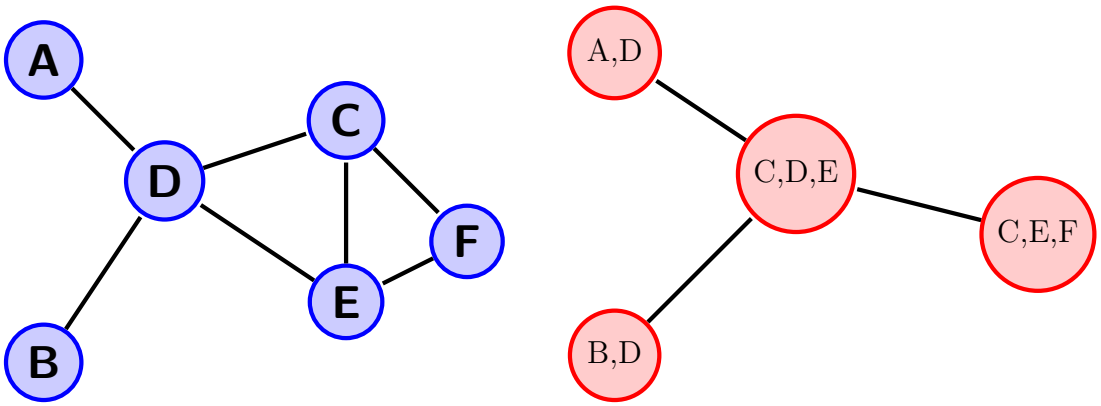


Figure 3.1: An example of MRF and clique, left (with blue circles) shows MRF graph, with nodes $\nu = \{A, B, C, D, E, F\}$ and $\varepsilon = \{(A, D), (B, D), (C, D), (C, E), (C, F), (D, E), (E, F)\}$. Right (with red circles) identify the maximal cliques $\{(A, D), (B, D), (C, D, E), (C, E, F)\}$.

$\prod_{i=1}^{n} p(x_i|\theta)$. Maximizing the likelihood is the same as maximizing the log-likelihood given by:

$$\log L(\theta|\mathbf{x}) = \log \prod_{i=1}^{n} p(x_i|\theta) = \sum_{i=1}^{n} \log p(x_i|\theta) \tag{3.2}$$

The local distribution function $p(x)$ can be constructed by Gibbs distribution.

$$p(x) = \frac{1}{Z} \prod \phi_c(x_c) \tag{3.3}$$

where x_c is each maximal clique in the MRF. The normalizing factor Z is called partition function by analogy with physical systems.

$$Z = \sum_{x} \prod \phi_c(x_c) \tag{3.4}$$

The $\phi(x_x)$ is usually written as:

$$\phi_c(x_c) = e^{\frac{1}{T} F(x_c)} \tag{3.5}$$

We define the potential of a clique as a function that associates a real number with each configuration of the clique.

This is also referred as Energy Based Models (EBMs) [69]. EBM tend to create an analogy with the thermodynamic systems and the concept of energy is defined as the objective to be minimized. Learning corresponds to finding an energy function so that associates lower energies (higher probabilities) to correct values of the remaining variables, and higher energies (lower probabilities) to incorrect values. Energy based probabilistic models define a probability distribution through an energy function, as follows:

$$p(x) = \frac{e^{-E(x)}}{Z} \tag{3.6}$$

To learn a EBM, we are given a set of training samples. In order to find the best energy function, we need to define a loss (cost) function $L(\theta, D)$, where θ is parameter, and D is dataset. The learning problem is simply to find the θ that minimize the loss. An EBM can be learnt by performing (stochastic) gradient descent on the empirical negative log-likelihood of the training data. For example, if we are talking about logistic regression. It is very common to use the negative log-likelihood as the loss function. More precisely, the loss function can be defined as:

$$L(\theta, D) = \frac{1}{N} \sum_{x^{(i)} \in D} \log p(x^{(i)}) \tag{3.7}$$

We will iteratively compute the parameter:

$$\theta' = \theta - \frac{\partial \log p(x)}{\partial \theta} \tag{3.8}$$

In many situations, we do not observe the sample x fully, or we want to intro-

duce some non-observed factors or variables to enrich the representation power of model. We consider an observed (visible) part x and a hidden part h. We can then write:

$$P(x) = \sum_h P(x,h) = \sum_h \frac{e^{-E(x,h)}}{Z} \tag{3.9}$$

In such cases, to map this formulation to one similar to Equation 3.6, we introduce the notation of free energy, defined as follows:

$$F(x) = -\log \sum_h e^{-E(x,h)} \tag{3.10}$$

which allows us to write:

$$P(x) = \frac{e^{-F(x)}}{Z} \tag{3.11}$$

The data negative log-likelihood gradient then has a particularly interesting form:

$$-\frac{\partial \log p(x)}{\partial \theta} = \frac{\partial F(x)}{\partial \theta} - \sum_{\hat{x}} p(\hat{x}) \frac{\partial F(\hat{x})}{\partial \theta} \tag{3.12}$$

It is usually difficult to determine this gradient analytically, as it involves the computation of $E_p[\frac{\partial F(x)}{\partial \theta}]$. We are computing an expectation over all possible configurations of the input x (under the distribution P formed by the model). The calculation grows exponentially as function of length of input.

The first step in making this computation tractable is to estimate the expectation using a fixed number of model samples. Recall that statistical sampling can be applied to any expectation, if $x^{(s)} \sim P(x)$, we have:

$$\int f(x)P(x)\,\mathrm{d}x \approx \frac{1}{S}\sum_{s=1}^{S} f(x^{(s)}) \tag{3.13}$$

Using statistical sampling, the gradient can then be written as:

$$-\frac{\partial \log p(x^{(i)})}{\partial \theta} \approx \frac{\partial F(x)}{\partial \theta} - \frac{1}{N}\sum_{\hat{x}\in N}\frac{\partial F(\hat{x})}{\partial \theta} \tag{3.14}$$

With the above formula, we almost have a practical, stochastic algorithm for learning EBM. The problem here is we want to generate random draws $\hat{x}$ elements of N from a target distribution P.

3.3 Restricted Boltzmann Machines (RBM)

Boltzmann Machines (BM) is a special case of MRF, it has been proposed in the 1980s [70]. Everything of BM is defined in terms of energies of joint configurations of the visible and hidden units. RBM (Figure 3.2) further restrict BMs to those without visible-visible and hidden-hidden connections. Connections between neurons are bidirectional and symmetric. This means that information flows in both directions during the training and during the usage of the network. RBM is the most popular building block for deep architecture and has been used as generative models of many different types of data [25, 71, 64, 51]. A graphical depiction of an RBM is shown below. A joint configuration, (v, h) of the visible and hidden units has an energy $E(v, h)$ given by [72]:

$$E(v,h) = -b'v - c'h - h'Wv = -\sum_{j=1}^{m} b_j v_j - \sum_{i=1}^{n} c_i h_i - \sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij} h_i v_j \tag{3.15}$$

The definition is based on product of expert model proposed by Hinton [73] and motivated by Hopfield network [72]. where W represents the weights connecting hidden and visible units and b, c are offsets (biases) of the visible and hidden layers

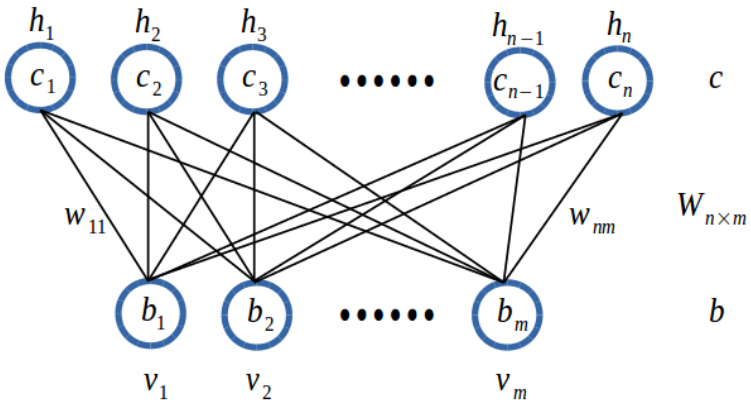


Figure 3.2: Graphical model of a RBM with m visible units and n hidden units

respectively. In other words, the network assigns a potential function $E(v, h)$ to every possible pair of a visible and a hidden vector. The free energy turns into:

$$F(v) = -b'v - \sum_i \log \sum_{h_i} e^{h_i(c_i + W_i v)} \tag{3.16}$$

Because of no direct connections between the same layer in RBMs, visible and hidden units are conditionally independent given one-another. Using this property, we can write:

$$\begin{aligned} p(h|v) &= \prod_{i=1}^{n} p(h_i|v) \\ p(v|h) &= \prod_{j=1}^{m} p(v_i|h) \end{aligned} \tag{3.17}$$

Where the marginal distribution over v is:

$$
\begin{aligned}
p(v) = \sum_h p(v,h) = \frac{1}{Z}\sum_h e^{-E(v,h)} &= \frac{1}{Z}\sum_{h_1}\sum_{h_2}\cdots\sum_{h_n} e^{\sum_{j=1}^{m} b_j v_j} \prod_{i=1}^{n} e^{h_i(c_i+\sum_{j=1}^{m} w_{ij}v_j)} \\
&= \frac{1}{Z} e^{\sum_{j=1}^{m} b_j v_j} \sum_{h_1} e^{h_1(c_1+\sum_{j=1}^{m} w_{1j}v_j)} \sum_{h_2} e^{h_2(c_2+\sum_{j=2}^{m} w_{1j}v_j)} \sum_{h_3} e^{h_3(c_3+\sum_{j=3}^{m} w_{1j}v_j)} \\
&= \frac{1}{Z} e^{\sum_{j=1}^{m} b_j v_j} \prod_{i=1}^{n} \sum_{h_i} e^{h_i(c_i+\sum_{j=1}^{m} w_{ij}v_j)} \\
&= \frac{1}{Z} \prod_{j=1}^{m} e^{b_j v_j} \prod_{i=1}^{n} \left(1 + e^{h_i(c_i+\sum_{j=1}^{m} w_{ij}v_j)}\right)
\end{aligned}
\tag{3.18}
$$

Parameter Learning in RBM

When applying RBM in classification, typically the hidden and visible units are binary, where v_j and $h_i \in \{0,1\}$. The probabilistic of unit state are defined as:

$$
\begin{aligned}
p(h_i = 1|v) &= \sigma(c_i + \sum_{j=1}^{m} w_{ij} v_j) \\
p(v_i = 1|h) &= \sigma(b_j + \sum_{i=1}^{n} w_{ij} h_i)
\end{aligned}
\tag{3.19}
$$

The model is defined by its conditional probabilities. The task is to find the weight matrix that best explains the visible data for a given number of hidden units. The free energy of an RBM with binary units further simplifies to:

$$
F(v|W,b) = -b'v - \sum_i \log\left(1 + e^{1+W_i v}\right) \tag{3.20}
$$

RBMs are usually trained using a much faster method called k-step contrastive divergence (CD-k) [73]. This requires a certain amount of practical experience

to decide how to set the values of numerical meta-parameters such as learning rate, the weight-cost, the initial values of the weights, the number of hidden units and the size of each mini-batch. RBMs can be stacked and trained in a greedy manner to form DBN. DBNs are graphical models which learn to extract a deep hierarchical representation of the training data. They model the joint distribution between observed vector x and the l hidden layers h^k as follows:

$$P(x, h^{(1)}, h^{(2)}, .., h^{(l)}) = (\prod_{k=0}^{l-2} P(h^k|h^{k+1}))P(h^{(l-1)}, h^l) \tag{3.21}$$

where $x = h^0$, $P(h^k|h^{k+1})$ is a conditional distribution for the visible units conditioned on the hidden units of the RBM at level k, and $P(h^{(l-1)}, h^l)$ is the visible-hidden joint distribution in the top-level RBM.

3.4 Comparison between RBM and Linear Models

To demonstrate the representation performance of RBM in our experiments, it is worthwhile to compare our method to linear models. We now briefly turn to the discussion of two commonly used techniques, namely, PCA and linear discriminant analysis (LDA). Both of them are linear transformation methods and attempt to represent the data with lower dimensions. We refer the reader to references for more details [9, 74], PCA method finds a subspace, where basis vectors correspond to the maximum-variance directions in the original space. The principle behind is that a large variance usually has an important structure to consider. In practice, we keep only the largest components to reduce the dimensions. When data was projected into this lower-dimensional space, we then

fed them into some classifiers (we use supported vector machine (SVM) in our experiment). In theory, however, PCA is not optimal for classification under some conditions, because it ignores the class discrimination. The discriminant dimensions could be simply discarded. A theoretically better method to find discriminant direction is LDA. It provides a linear boundary, which is generated by fitting class condition densities to the data. In a two-dimensional example shown in Figure 3.3, PCA prefer the direction which shown in red dotted line (the PCA boundary shown in red solid line) because it has the largest variance in the components directions. While LDA (in purple color) find the direction, which corresponds to the class discriminant direction. In this case, LDA should outperform PCA.

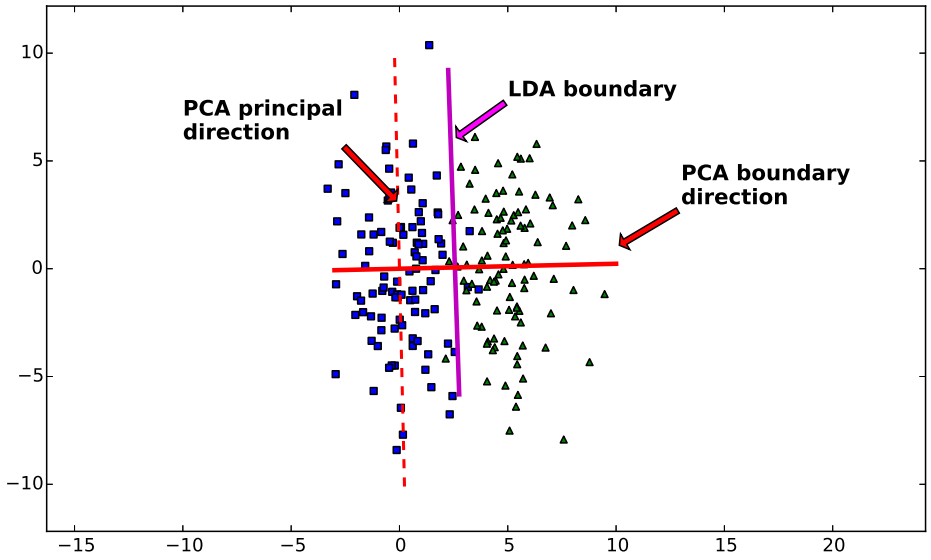


Figure 3.3: An comparison of LDA and PCA in 2-dimensional data.

3.4.1 Methods

In our study, we utilized the data from 7,384 patients, which include 937,461 flow-sheet entries in total. Based on our previous study [61] we selected data items that are documented in continuous numerical values with sufficient frequencies. We synchronized the time series data by using MATLAB.

Classification using Discriminant RBM

We addressed the task of classification using RBM with two approaches. The first one is straightforward: we directly fed the hidden vectors into another classifier. Note that RBMs provide no guarantee that the generated hidden variables will ultimately be useful for the supervised task. In other words, if we are handling the task of 2-class classification and set the number of hidden units as 1, this hidden unit usually has no connection with our labels. The second more interesting approach is discriminant RBM, which utilizes the nature of model to compute the probability. We assume that a test set $D_{test} = \{v_i\} = \{(x_i, c_i)\}$ consists of

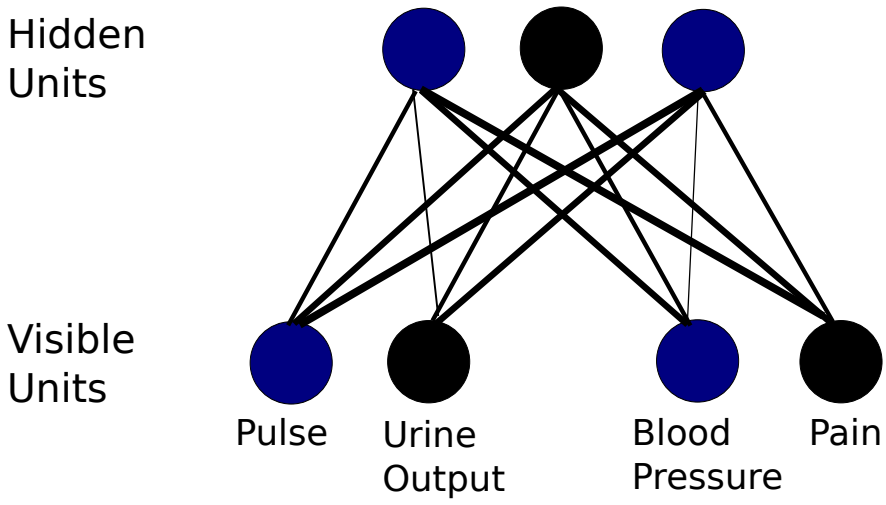


Figure 3.4: Pain label c is included in the visible layer. The black color denotes the state "on", while the blue color denotes the state "off".

Table 3.1: Number of Data Items for Selected Individuals.

Patient ID	**Number of Data Items**
7137	22
4822	28
1245	28
6563	24

an input features vector x_i and a target class $c_i \in \{0, 1\}$ (see Figure 3.4). The probabilities of two visible vector $v_1^0 = (x_1, c_1 = 0)$ and $v_1^1 = (x_1, c_1 = 1)$ can be directly computed from their free energies $F(v_1^0)$ and $F(v_1^1)$. As shown in the equation 3.22, we can further obtain the probabilities $p(v_1^0)$ and $p(v_1^1)$ using the chain rule to cancel the unknown constant Z.

$$\frac{p(c_1 = 0|x_1)}{p(c_1 = 1|x_1)} = \frac{p(c_1 = 0, x_1)}{p(c_1 = 1, x_1)} = \frac{p(v_1^0)}{p(v_1^1)} \tag{3.22}$$

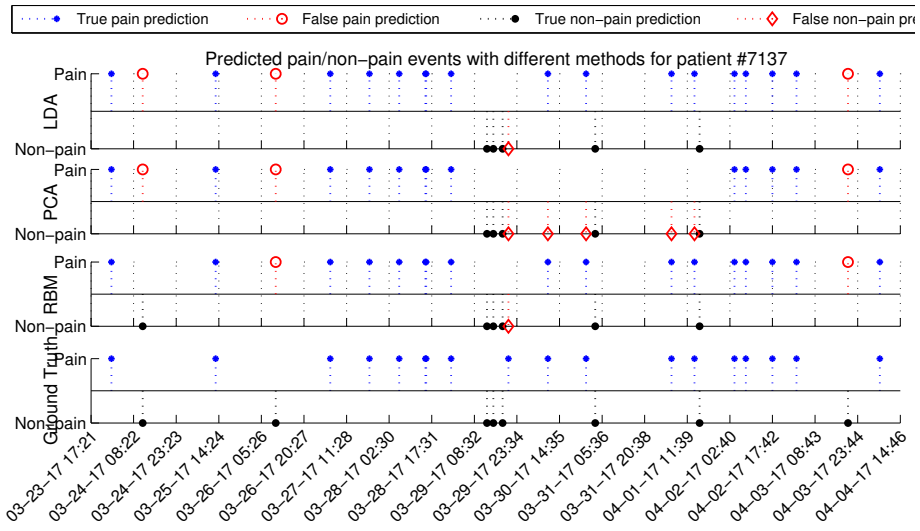
Predicting pain state

In a supervised experiment, we would expect the number of labeled data points (i.e., training data) to be large. To keep the training data as large as possible, we only selected the features whose time interval was larger than the time interval of pain labels (i.e., documented pain). Four individuals (Table 3.1) with the number of recorded labels larger than 1000 were used in the experiment. None of the 4 patients had all 78 assessment items available.

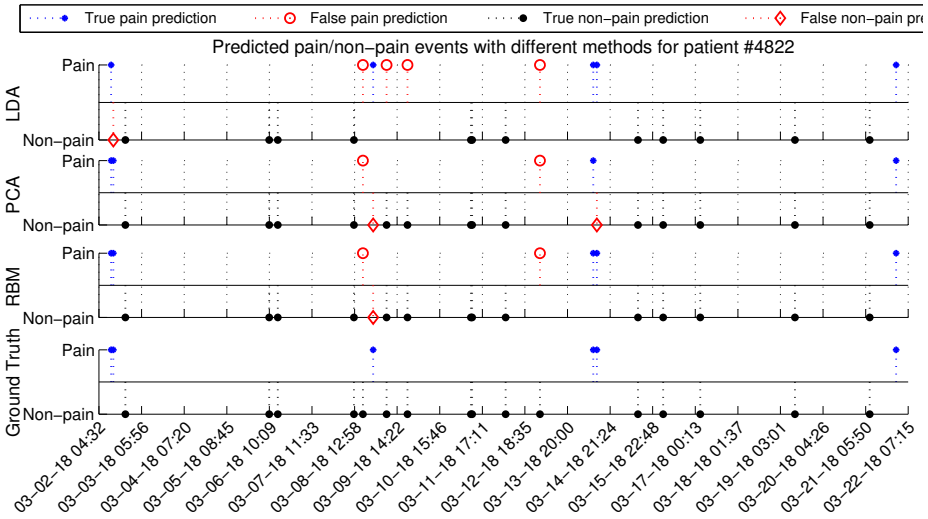
Number of available data items for the 4 patients varied from 22 to 28 (Table 3.1).

In classification tasks it is necessary to perform pre-processing of the data before applying the algorithm. In our experiment, we converted the numerical data t into binary data x . It is worth mentioning that the binary representation may be inappropriate in many real problems, although their interpretation

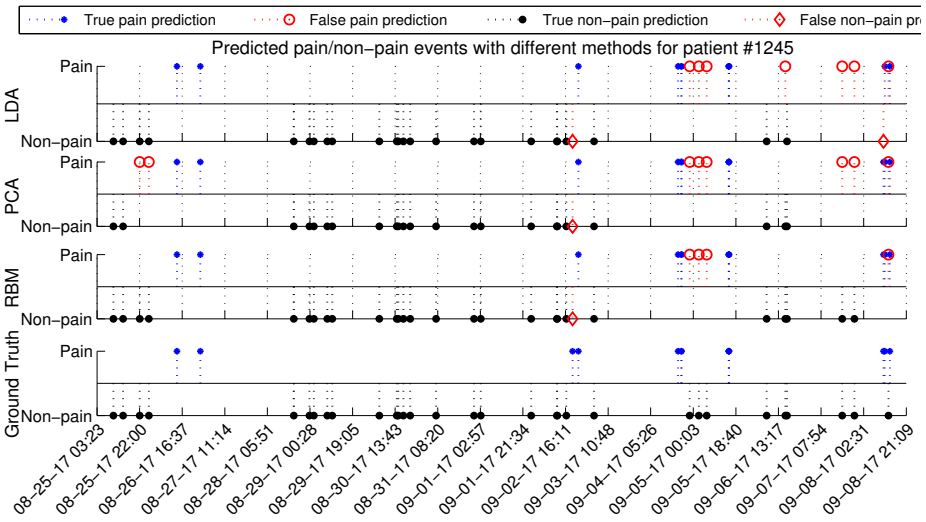
("normal" and "abnormal") makes sense in our medical electronic data. Another important step is personalization, since the indication of a "normal" state varies among different individuals. We assumed the probability distribution of status measurement to be a Gaussian. We fit the numerical feature into this Gaussian probability distribution function, $(x_i = 1) \sim \mathcal{N}(t_i|\mu, \sigma^2)$, where $p(x_i = 1)$ represents the probability of current feature is abnormal. The mean and variance can be directly calculated from the samples. With this procedure, the resulting feature vectors are well-suited to the standard binary RBM. In this experiment, we convert non-binary probability into binary value by setting threshold = 0.5. Since LDA allows only the $number of classes - 1$ dimension to be used, there is no parameter to be set in our 2-class task. For PCA, the size of the reduced dimension was selected as $k = 4$, which can cover most energy of the original data. In our experiment, we considered RBM as a non-parametric model and allowed the number of hidden variables to vary by the data [75]. As we have a trade-off to make: while a larger number of hidden units usually give a more powerful representation of distribution it also exaggerates the over-fitting problem. Therefore, the number of hidden units was adjusted between 15 and 30.



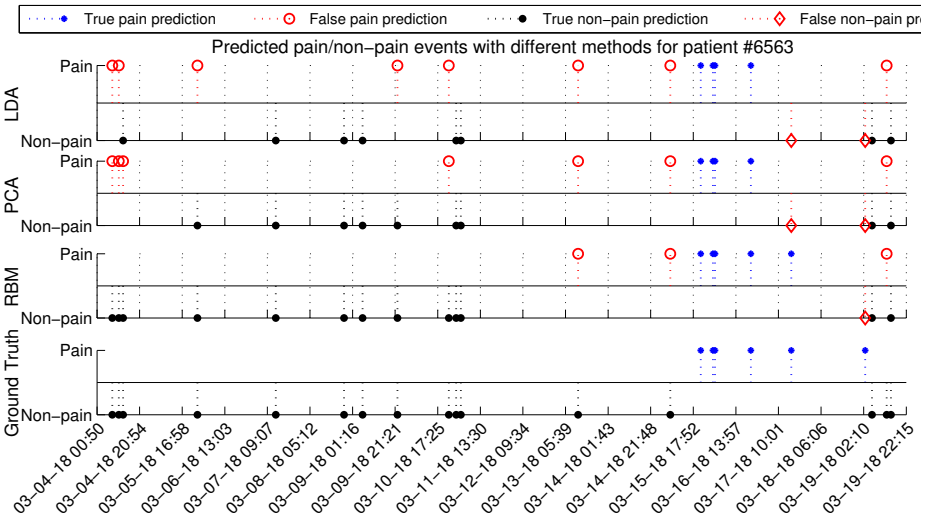
(a) Patient ID 7137



(b) Patient ID 4822

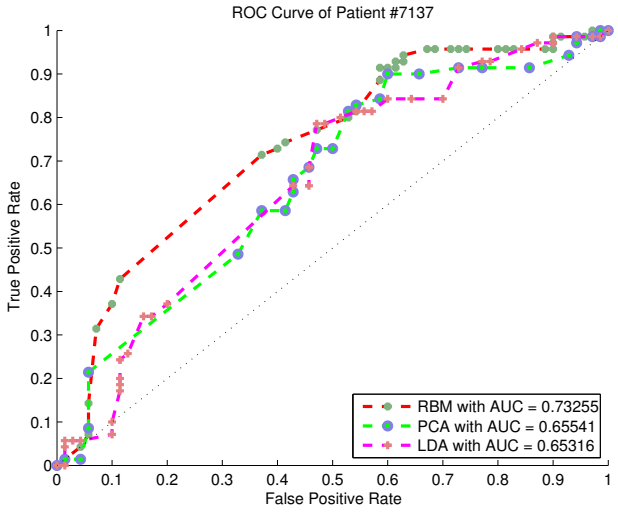


(c) Patient ID 1245

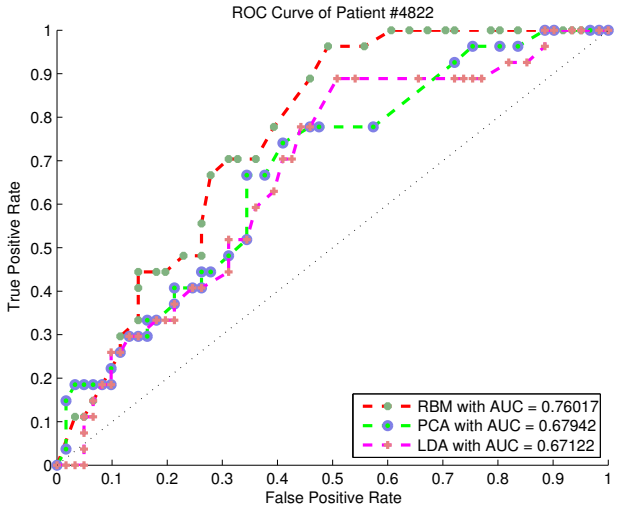


(d) Patient ID 6563

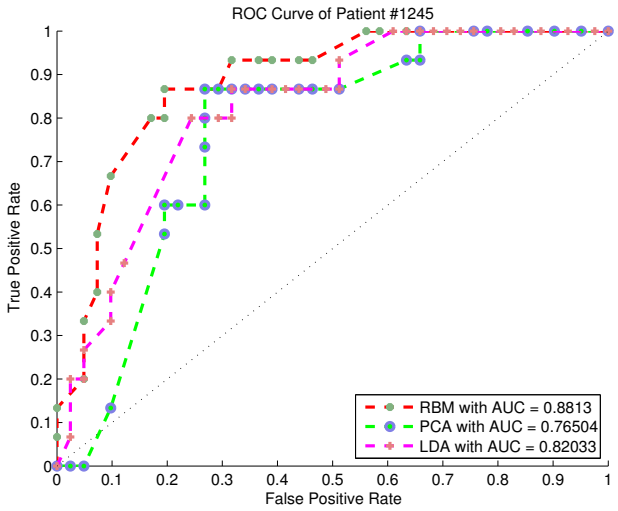
Figure 3.5: Comparison of RBM, LDA, and PCA in pain state prediction. (a), (b), (c) and (d) are the predicted pain labels using their optimum threshold from patients (with IDs 7137, 4822, 1245, and 6563). Artificial timestamps are used for patient privacy.



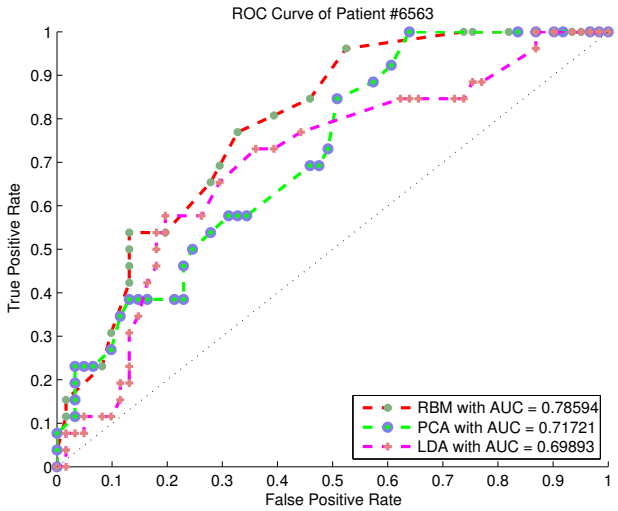
(a) Patient ID 7137



(b) Patient ID 4822



(c) Patient ID 1245



(d) Patient ID 6563

Figure 3.6: Comparison of RBM, LDA, and PCA in classification. ((a), (b), (c) and (d) are the ROC curves (with IDs 7137, 4822, 1245, and 6563).

3.4.2 Experimental Results

With regard to the classification performance, we first examined the Receiver Operating Characteristic (ROC) curves for each model and calculated the area

Table 3.2: Classification results using discriminant RBM, PCA with SVM, and LDA.

Patient ID	Model	AUC	Sensitivity	Specificity	Accuracy
7137	RBM	**0.73255**	0.7143	**0.6286**	**0.6714**
	PCA + SVM	0.65541	**0.8143**	0.4714	0.6429
	LDA	0.65316	0.8	0.4858	0.6429
4822	RBM	**0.76017**	**0.7037**	**0.6885**	**0.6932**
	PCA + SVM	0.67729	0.6667	0.6557	0.625
	LDA	0.67122	**0.7037**	0.5902	0.6591
1245	RBM	**0.8813**	**0.8667**	**0.8049**	**0.8214**
	PCA + SVM	0.76504	**0.8667**	0.7317	0.7679
	LDA	0.82033	0.0.8	0.7561	0.7679
6563	RBM	**0.78594**	**0.7692**	0.6721	**0.7011**
	PCA + SVM	0.71721	0.5769	**0.6885**	0.6552
	LDA	0.69893	0.7308	0.6393	0.6667

under the curve (AUC) by varying the classification threshold (Figure 3.6). The ROC curve is a plot of true positive rate on the vertical axis against false positive rate on the horizontal axis. Sensitivity, specificity and accuracy were used as the criterion of classification performance with the optimum boundary. True positive (TP), true negative (TN), false positive (FP) and false negative (FN) are defined as the true pain prediction, true non-pain prediction, false pain prediction and false non-pain prediction. Then we have the following: $Sensitivity = \frac{TP}{(TP+FN)}$, $Specificity = \frac{TN}{(TN+FP)}$ and $Accuracy = \frac{RN+TP}{(TP+TN+FN+FP)}$.

Table 3.2 suggests that the performance of PCA and LDA is quite similar. Even though intuitive choice will prefer LDA than PCA, there is no guarantee that LDA will outperform PCA, especially when the size of training data is not sufficiently large. This observation is also reported in [76]. With carefully setting parameters, discriminant RBM can achieve better accuracy in all 4 experiments.

3.4.3 Summary

Our experiments show that the RBM classification is competitive to methods of LDA and SVM using PCA. The AUC was improved and the predicted pain labels using RBM outperforms the LDA and PCA using the optimum threshold, respectively. However, we notice that most results of AUC are still in a fair level (where the value of AUC is smaller than 0.8). Some insights into this limitation can be concluded as follows: (1) we have not given any consideration to the temporal data. Ignoring time will result in significant loss of information. We carried out the experiments by using the training data in the first half of time series, and made pain prediction with the test data, which corresponds to the second half of the time series, the detection rate of all methods turned out to be very poor. (2) Currently, we adopted the fundamental binary RBM. An obvious observation is that binary representation may be insufficient to represent states very well. For example, the value of blood pressure higher or lower than mean value may give totally different contributions to the pain response. In order to enrich the representation power, we will incorporate the Softmax visible units to the model in our future work. (3) Most importantly, training of RBM expects the number of training data to be large. However, our experiment was done with a relatively small dataset. To this end, we will try to learn a model using a bigger dataset, and fine-tune the parameters only using the specific individual data in future work to minimize data loss.

Chapter 4

Deep Neural Networks for Text Detection

4.1 Convolutional Neural Network

Before 2006, the vast majority of network had less than 3 hidden layers. One remarkable exception were Convolutional neural networks (ConvNets) [77] which typically are able to train multiple layers of hidden units with back-propagation. A good example is the classic LeNet [78] which proposed by Lecun in 1990s and has been successfully applied to hand-written digit recognition for several decades.

Similarly, ConvNet was inspired from biological visual cortex. A variety of ConvNets have enjoyed a series of successes in many problems related to object classification and recognition [79]. Coupled with the rapid advancements in GPU (graphics processing units) computation, it is now a standard way to train much larger and more powerful ConvNets that achieve state-of-the-art performance on standard benchmarks.

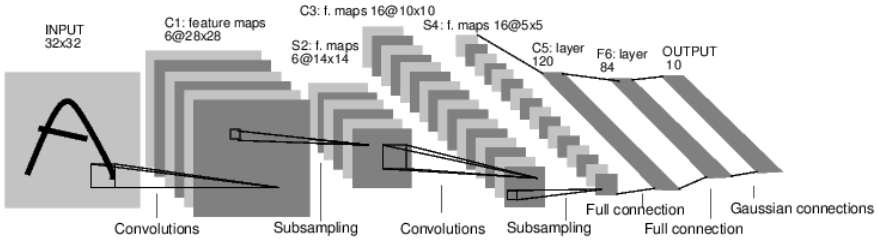


Figure 4.1: LeNet5, a 5-layer ConvNet which was first proposed by [78] in 1998.

Concretely, a ConvNet is just a multilayer, hierarchical neural network. A basic ConvNet, such as the one in Figure 4.1, consists of applying multiple layers of learned convolution kernels together with element-wise nonlinearities, often followed by a spatial pooling (subsampling). Regular FNNs are both computationally intensive and memory intensive, making them extremely hard to deploy and train on the system. ConvNet address this limitations by involving Convolution layer and dramatically decrease the number of parameters.

4.1.1 Convolutional Layer

Convolutional (Conv) layer is the essential part of ConvNet. It aims to using spatial information between the pixels of an image, in other words, it take advantage of the property of heavy correlation between nearby pixels in the image. The concept of receptive field was proposed to capture this correlation. Mathematically[1], given a filter size (f, f), the value at position (x, y) in layer l on the feature map v after the Conv layer is given by:

$$v_{xy}^{l} = \sum_{p=0}^{f-1} \sum_{q=0}^{f-1} w_{pq} v_{(x+p)(y+q)}^{l-1} \tag{4.1}$$

[1]Notice that the equation is correlation rather than convolution. Correlation is almost the same with convolution except that convolution is associative. From the point view of learning parameter, there is no difference between the two operations.

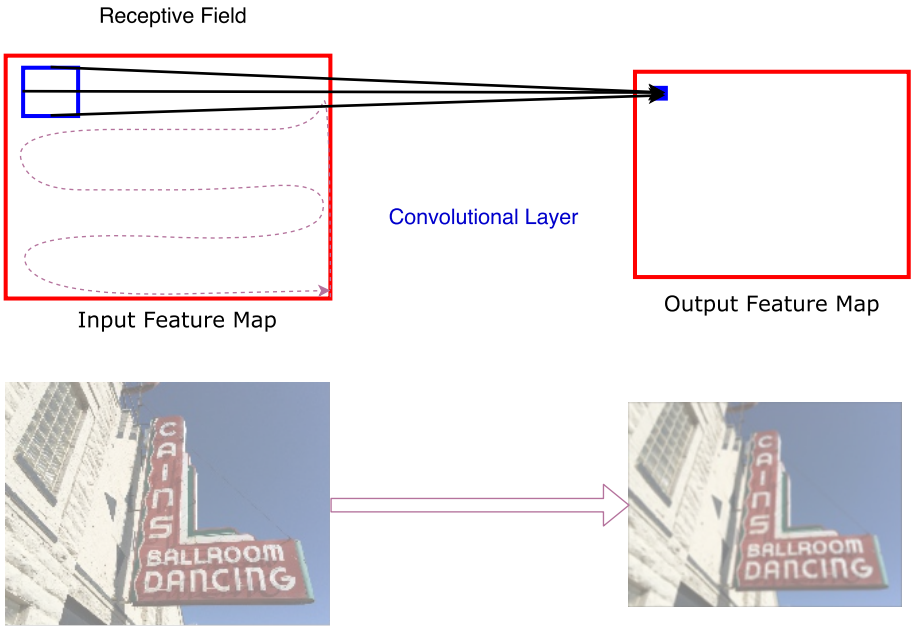


Figure 4.2: An example of Convolutional layer.

Conv layer offers a number of advantages over regular fully connected layer. First, the forward propagation is easier because the regular FNN is over-parametrized and Conv layer requires much less number of parameters, thus avoiding waste of computation. For example, for an input feature map 100×100 and the same size output feature map, there are 10^8 weight parameters. Conversely, it we using 64 filters with each size 3×3 [2], there are only $3 \times 3 \times 64 = 576$ parameters. As a result, Conv layer reduce the number of parameters by an order of magnitude.

4.2 The Text Localization Problem

A large number of research has been done on Optical Character Recognition (OCR) [80]. Many commercial OCR systems are able to achieve the high detec-

[2] small filter sizes are sufficiently good for image application

tion accuracy. However, OCR generally focused more on specific scanned documents with clean background rather than arbitrary image formats with random background and surrounding. This disadvantage limit the practical application of OCR.

Localization and recognition of text from natural scene has been one of the important focuses in the field of computer vision. Understanding text usually conveys valuable information and has a wide variety of applications. For example, the self driving car can detect and understand the traffic signs (e.g. "Stop", "One way") to navigating the street; the online search engine can automatically read the annotation of video and thus allow fast retrieval by query terms [81]. (Figure 4.3).

(a) (b)

Figure 4.3: Two examples illustration of the texts in wild scene. (a) Two stop signs in parking lot. (b) Breaking news subtitle in video.

4.2.1 Challenges

Text localization in scene image is particularly difficult and still an active area of research for a number of reasons. First and most importantly, the basic problem of knowing what is text or character is difficult. Much effort was devoted to the development of representing text feature. Various assumptions and empirical rules (features) [82, 83] were proposed to represent text. However, most hand-

engineered features are actually very common in natural scenes. For example, considerable non-text objects (such as leafs, bars, walls) are composed of similar small shapes as well as strokes with texts. Secondly, some approaches detect characters and group them together. The problem is that there could be a very large number of characters. For example, there are over 50,000 Chinese characters. Thirdly, compare to regular object detection such as car detection, the aspect ratio could not be fixed in advance due to the text-line varies. Moreover, scene image often suffering from inconsistent lighting, occlusions, orientations and noise. Hence, a robust system should be able to tolerate these extreme situations.

In general, the common text recognition strategy involves text localization which determine whether or not the input image contains any text, if does then localize candidate regions; the recognition step attempts to recognize the text depicted within the regions, or potentially reject the bounding box as a false positive detection. In this thesis, we focus on the first step. In particular, we tackle the task of the text localization in real world scene images, as opposed to extracting text from scanned documents.

4.2.2 Overview of methods

The design of feature which be able to represent text pattern is critical in this task. ConvNets [84] have been successfully used to achieved remarkable results in a variety of image recognition problems [29, 85]. Various ConvNets have been employed in the text localization by [86, 87, 88]. Their commonly-used pipelines (Figure 4.4) mainly involve running sliding (generally rectangular) window detector over input image with a number of scales to generate a text/non-text confidence map, and integrating the responses with the candidate spacings by

post-processing like non-maximal suppression (NMS) and beam search. The reported results show [86, 88, 81] that ConvNets performs well even in the extreme situations. Unfortunately, another critical challenge arises on side of practical applications. Localizing text in an image is potentially a computationally infeasible task as generally any of the 2^N subsets can correspond to text (where N is the number of pixels for a square image).

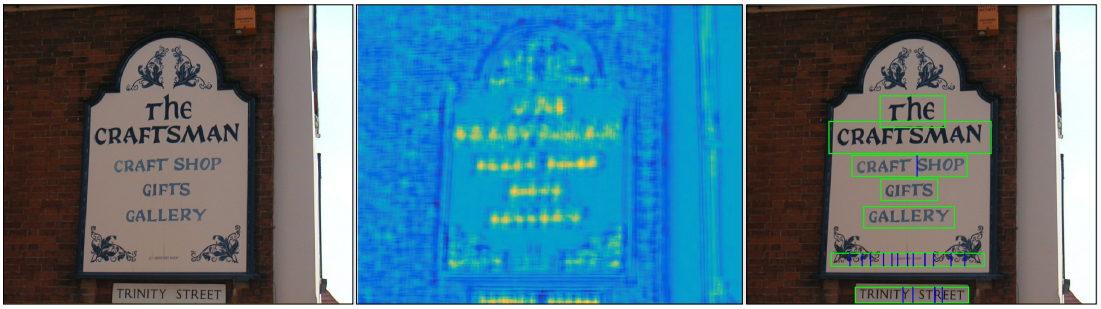


Figure 4.4: Illustration of conventional sliding windows localization of text using the ConvNet based detector. From left to right: input image, saliency map generated using text/non-text classifier, bounding box results after post processing.

Recently, region proposal was combined with ConvNet method in text localization [89] to avoid exhaustive search through images. The work achieved exceptionally impressive results, and the accuracy is closely approaching human-level performance. While the computation time was decreased dramatically, it still cost about 6 seconds with the existence of single GPU. The reason is that they first over-segment the image and as a result they generate a large number of candidate regions in order to maintain a high recall. In other words, region proposals take equal computation effort through the image; most computation efforts are devoted to eliminate the non-text candidates. Motivated by the observation that the human visual system can collect enough information with a single glance of an image, we employed one-pass ConvNet to assist localizing the

text candidate regions.

Contributions: In contrast to previous works that typically concentrate on improving the text detection accuracy, and driven by the demand for real-time, our work aims at efficiently and effectively localize the text under extreme environment. To achieve these, our main contributions are presented as follows: 1) We directly track text as an object rather than detect characters and group them together. In the processing of grouping, the majority of existing methods require a fixed-sized lexicon dictionary which reduce the generality of their models. Our approach does not need the lexicon information. 2) Our pre-trained ConvNet automatically output the coordinates of text bounding boxes associated scores. The detections are performed in the manner of single-shot forward passing. We significantly improved the computation efficiency by a large margin. 3) We utilize the ConvNet to represent whole scene image rather than text-only features, and encode the contextual spatial information into localization. To the best of our knowledge, this is the first work reported in the literature of text spotting using one-shot ConvNet.

4.2.3 Related work

Overall, text localization methods usually fall into two primary paradigms: region based and sliding windows detector based. Region based approaches [82, 83, 90] extract character candidates from images, and then group characters into word regions. The intuition behind this is the assumption of characters from the same language have similar properties (color, size, intensity, stroke-width, etc.). These approaches are attractive because it's a general language model and invariant to the scale. Note however that there is no universally rules can differentiate

text in all scene images. While most of these existing approaches are able to detect text well with clean background (e.g. only text in the image), they usually failed in the presence of blurring, significant variations and complex background [82]. Another dominant approach is sliding windows detector based methods [91, 92]. The majority of existing methods train a character/non-character classifier based on manually designed features, such as SIFT [93], and HOG [94]. Then these methods scan through the image at a number of scales. Recently, the ConvNet has caught significant attention in both academic and industrial communities. Through introducing more intermediate neural layers, a deep network model can have higher representing power and can be trained to automatically extract features of an object. [86] first trained ConvNet character/non-character classifiers. However, in doing so, the exhaustive search based detector was far too slow for real-time applications. Nevertheless, ConvNet detector achieved higher recognition rates than most region based methods.

Character and text (word) based approaches. It is important to note that both of the aforementioned paradigms address the text localization by employing the same mechanism: localizing characters, and subsequently grouping these into words. Both models, however, do not capture the structure of text in the real world. Intuitively, a lone character can be a very ambiguous object, and often impossible to distinguish them from the rest of the image.

In order to reduce the number of windows, and thus be able to apply more advanced classification, the method of bounding-box proposals was formed. More recent work by Jaderberg [89] employed this method in text detection for the first time. In their pipeline, more than 10^7 regions were first proposed using Edgebox [95], and then this number was decreased to 10^4 by filtering with random forest (RF). For each region, bounding box regression and text recognition were

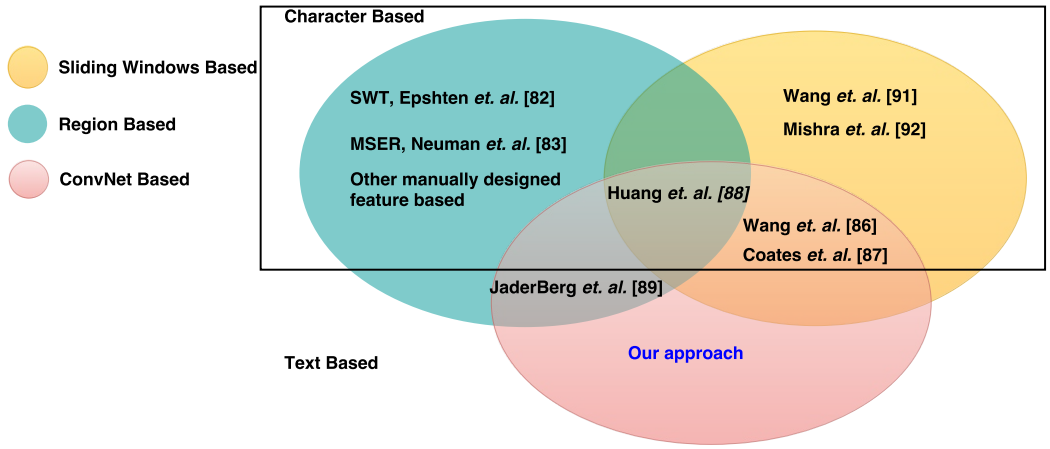


Figure 4.5: A pictorial representation of categories of various methods and our approach

performed to assist localization. This work offers a balance between efficiency and performance. However, there are two sources of inefficiency: (i) redundancy, the number of candidate regions is usually in the order of millions, which leads to computation resources spent in the following filtering step; (ii) intolerant to deteriorate image. Edgebox heavily relies on the edge detection performance, which is notably sensitive to the low image quality.

Our approach belong to text-based (Figure 4.5). We take a further step based on [89] and address both of its two inefficiencies. Our intuition lies behind the human visual system. Human beings can easily localize the text regions in terms of a glance. It would be desirable to make this process working in ConvNet. We were inspired from the work [96], [97], [98] in object detection and proposed to locating the text region through one-pass ConvNet rather than proposing a large number of candidate regions, as schematically illustrated by Figure 4.6. This conceptually straightforward approach outperforms previous methods from the viewpoint of computation efficiency by a large margin, while the accuracy is not

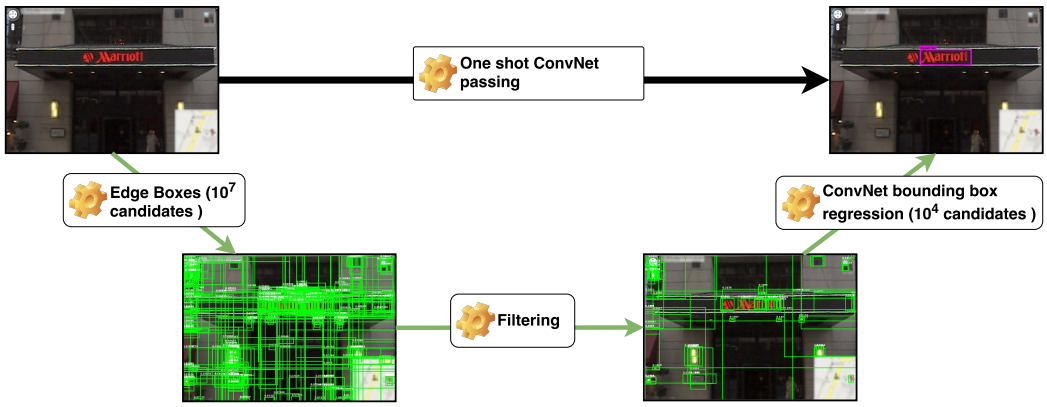


Figure 4.6: An overview of the text localization paradigms of our framework (top) and region proposals based framework [89] (bottom).

much decreased[3].

4.2.4 Region Based Methods

Text data can be analysed at different levels of representation. For example, text data can easily be treated as a group of words (SWT).

SWT

Epshtein [82] proposed the SWT (stroke width transform) for text detection. The SWT is an image operator which assigns a stroke width to each pixel of an image. The stroke width is determined by shooting rays on edges in the direction of the gradient. If the ray hits an edge with the same gradient direction modulo 180 degrees, the length of the ray determines the stroke width of the underlying pixels. Pixels which are adjacent to each other belong to the same character if their stroke width is similar. Hence, connected components are formed by segmenting the stroke width map. Adjacent connected components are grouped to words if

[3]The localization accuracy could be refined with the similar post recognition step which employed in [89].

the median stroke width ratio of two connected components does not exceed 2, the height ratio of the two components is smaller than 2 and distance and average color difference are within Thresholds learned from the training set. In addition, SWT is able to applied to many languages and fonts without training. Conversely, the performance of SWT decrease drastically under complex background. A SWT based text detection can be given by Figure 4.7.

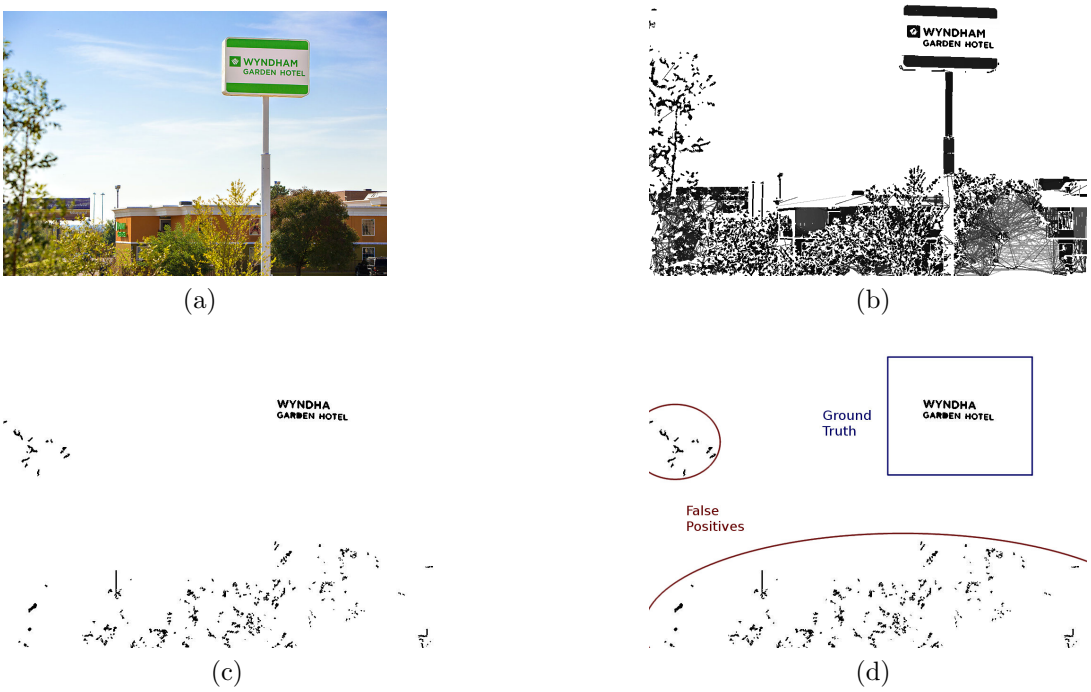


Figure 4.7: An example of the SWT used for text detection of an image (a). The SWT assigns every pixel the value of the width of the stroke it belongs to (b). Text and characters tend to be regions of uniform stroke width, so the variance of each connected component can be computed (c). Regions of low variance are likely to be characters, and can be grouped for word detection (d).

4.3 ConvNet Approaches

In this section, we rethink text localization from the perspective of computation efficiency. Given an image I, we are asked to find all the regions contain text in a efficient manner. At the end of the previous section, we talked about potential benefits from the region proposals. Below, we first provide a brief overview of the Edgebox method.

4.3.1 ConvNet in sliding windows

As an alternative strategy for text detection, many previous work uses sliding window methods to find text in an image. Sliding window based methods operate in a manner similar to generic object detection – that is taking a small window of interest within the full image, and classifying whether or not text is contained within that window. This window is then translated across the entire image, and at different scales, to detect text at all possible positions and scales. The output is a pixel wise text/no-text classifier, creating a text saliency map. The text saliency map is then generally thresholded to generate bounding box predictions for text localization.

4.3.2 Region Proposal Methods

Region proposal algorithms build on low-level features, and then create a set of hypotheses (candidate regions) based on their own defined "objectness". For all reported proposal methods, there is a trade-off between computational tractability and high detection quality. [99] evaluated ten detection proposal methods, among which SelectiveSearch [100] and EdgeBox (Fig 4.9) outperformed all other methods by a wide margin in ground truth recall and overlap ratio. We will then

give a brief introduction of these two methods:

Selective Search

Selective Search first applies superpixel segmentation [101], and then performs bottom-up hierarchical grouping of regions by combining neighbouring ones iteratively. Selective search using 4 sub-similarities which represent the similarities from the viewpoint of color histogram, gradient derivation of orientations, size fraction and the measurement of fitness. Therefore, it can produce completely general object candidates with no limitations. In addition, Selective Search requires no learning process and can even detect the text not placed in horizontal line (Figure 4.8).

(a) (b)

Figure 4.8: Example of selective search applied in the text image.

Edgebox

Edge boxes [95] utilize object boundary information which obtained from structured decision forests [102, 103]. The bounding boxes which have fewer contours

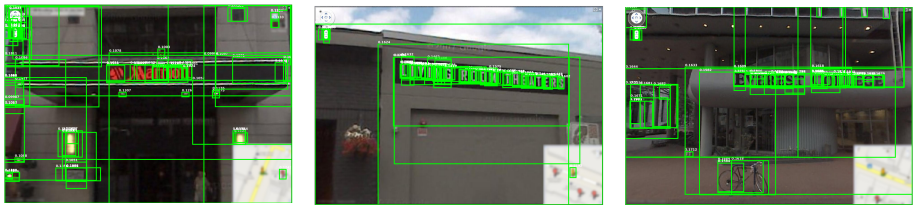


Figure 4.9: Top 100 proposed region boxes. (Note that the number of windows are restricted for illustration)

straggling the boundary of the box are considered more likely to be objects (Figure 4.9). They first compute the edge response map using Structured Edge detector, and perform NMS orthogonal to the edge responses. A candidate bounding box b is assigned a score s_b based upon the number of edges wholly contained by b, the boxes are then sorted by the score.

Since region proposals generators are primarily used to reduce the computational cost of the detector, they should be significantly faster than the detector itself. Under our hardware setting of CPU 2.4GHZ, the Selective Search takes about 2 seconds to compute proposals for a typical 500×300 image. While the Edgebox method only runs 250ms. From this perspective, Edgebox provide the best compromise in speed versus quality.

The state-of-art work first generate a number of objectness score using Edgebox. Unfortunately, because the score assigned is not related to the text, we have to handle every proposals. According to [89], more than 99% of the boxes are false-positives and have to be eliminated with further steps. Although the step of Edgebox is typically fast, the filtering stage, however, requires an expensive process of classifying due to millions of candidates.

Another serious downside which also happened on the region based methods is that the edge performance on extreme situations. The performance will drastically decline in the case of blurring and bad image resolution. For example, in

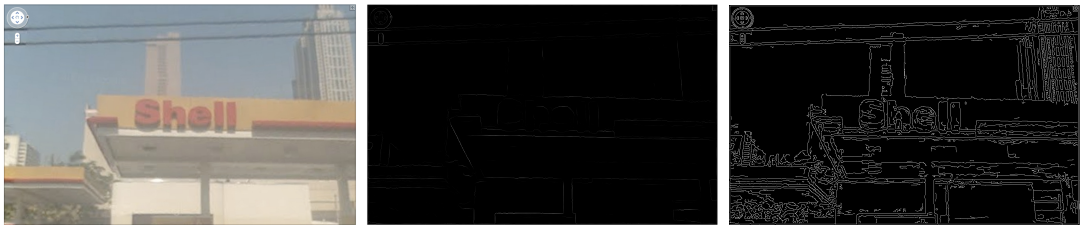


Figure 4.10: Both performances of structured edges (used in Edgebox) and canny edges(used in SWT and Extreme Region) are declined in the case of blurring)

Figure 4.10, structured edges appear to ignore the text structure, where canny detector preserve too much details. The former simply leads to the disappearing of text objects, where the latter produces too much candidate regions. Both of them result in failure of text localization.

4.3.3 Text localization: Bottom-up and Top-down cues

The region proposals improve the computation efficiency by generating millions of boxes instead of billions in conventional sliding windows manner. Our fundamental hypothesis is that the text region proposal step could ultimately be skipped. In other words, we search the (small number of) text regions instead of filtering (large number of) non-text regions. One reasonable strategy that may be used is to mimic how visual system spot texts and only focus on selected regions.

One promising potential solution relies on one much broader concept of saliency map. Saliency maps which are designed for predicting which regions are likely to attract human attention, has continued to be an active research area in computer vision for past decades.

As a general observation, texts in real world are intentionally designed to attract attention, and consequently can be considered in the scope of significant object. Traditionally, the approaches are divided into two categories: bottom-up

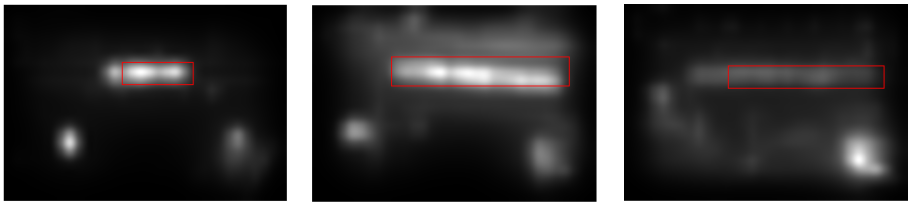


Figure 4.11: GBVS saliency maps calculated on SVT [86] dataset. The red rectangle indicates the ground truth text regions. The left appears to help text localization, the middle produces too much regions, and the right selects non-text regions

model and top-down model.

Bottom-up model works without any previous assumptions of the content of processed images, and models the unconscious visual processing in early vision and is mainly driven by hand-crafted low-level (multi-scale) cues (e.g., orientation, contrast and color). Given a feature map M, our goal is to compute an activation map A such that $M(i, j)$ is somehow unusual in its neighbourhood will correspond to high values of activation A. Various models [104, 105, 106] were proposed to solve it according to some criterion. A typical scheme defined as $A(i, j) = -log(p(i, j))$, where $p(i, j) = Pr(M(i, j)|neighborhood)$. We compute the saliency activation map (Figure 4.11) using [106] due to its reliability and efficiency. Naively, it might seem that the bottom-up saliency map will shrink the search region. However, the benefit is limited, since the computation of saliency relies on find all 'surprise' objects without taking into account the specific text appearance. More critically, with this saliency map using the "winner take all" policy, texts can simply be ignored.

Top-down ConvNet forward search

Top-down methods take advantage of data-specific information as prior knowledge, and are usually task-driven and learning-based. Saliency problem were

converted to a detection and regression problem. Fundamentally, we are using ConvNet to integrate these two tasks into one output layer. The text regions are represented as multiple bounding boxes as convention. In localization, we carry out one time forward-passing through the pre-trained network. Generally, in the training of ConvNets, we model the problem as classification. Instead, we model the prediction of text regions. More specifically, we treat the task of text localization as a regression problem.

Training objectives bounding box regression

We train a ConvNet to attempt to predict all text regions and their confidence scores. Given a training set of M pairs $\{(I_i, b_i)\}$, where each element I_i is a training scene image, and each b_i denotes the corresponding text label. The label b is composed of 4 coordinate informations $\{x, y, H, W\}$[4]. (x, y) denotes the normalized coordinates of center relative to the bounds of the sub-region, W and H indicate the width and height relative to the whole image respectively. First consider only one text region included, we need to learn the parameters that minimize the L_2 error: $\arg\min \sum_{i=1}^{M} \xi_i$, where the loss function is:

$$\xi_i = \sum_{i=1}^{M} ||g(I_i; w) - b_i||^2 = \sum_{i=1}^{M} (x_i - \hat{x_i})^2 + (y_i - \hat{y_i})^2 + (H_i - \hat{H_i})^2 + (W_i - \hat{W_i})^2 \quad (4.2)$$

where the g is ConvNet forward passing and w is the network parameter. When the problem extends to multiple regions. Intuitively, we breaks down input image into a $d \times d$ sub-regions, and extend the label into $\{l, x, y, H, W\}$. If the center of a text (word) falls into a sub-region, that cell is responsible for detecting that

[4]We use the same structure of bounding box as [107] and [96].

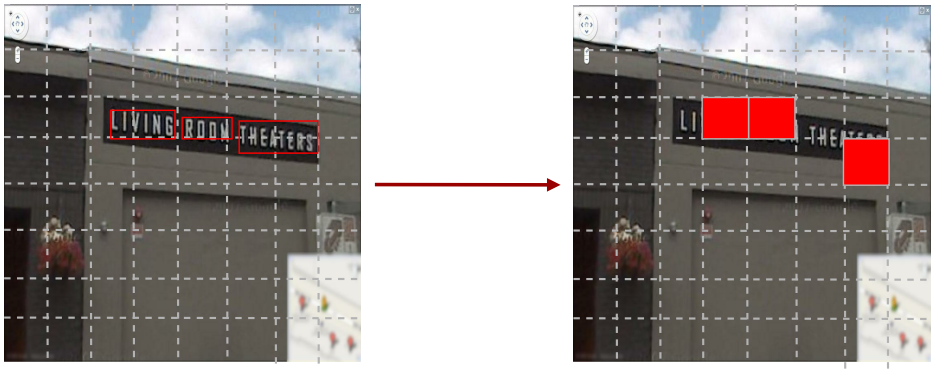


Figure 4.12: The annotation of the texts bounding boxes and corresponding responsible training sub-regions

object and we assign $l = 1$, otherwise, we assign the indicator $l = 0$. Because a particular sub-window may contain multiple texts, we set the number of bounding boxes as $N = 2$. In this case, we define the scoring function:

$$\xi_i = \sum_{i=1}^{M} \sum_{j=1}^{d^2} l_{ij}(b_i - \hat{b_i})^2 \tag{4.3}$$

ConvNet architecture

In general, the ConvNets served in the task of text localization typically contain a small number of hidden layers [89, 81, 86] due to the comparatively small size of texts. Note that our training data is entire scene image rather than text, we adopted the GoogleNet [108], which is a generic ConvNet architecture and achieved outstanding performance in image recognition tasks. We derive our scene text model from the [96]. Our network consists of stack of Conv layers followed by fully connected (fc) layers. The Conv layers are interspersed with maxpooling layers at various stages. The contextural features extracted from the Conv layers are fed into two fc layers. The output of the second fc layer is fed into

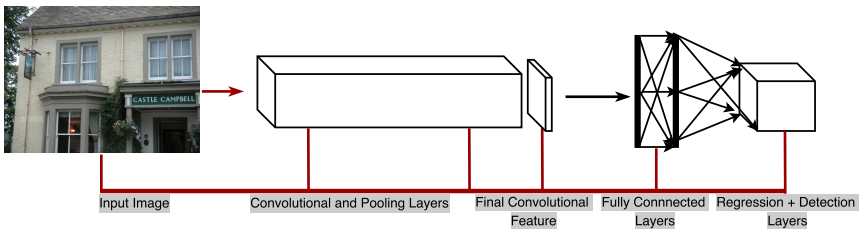


Figure 4.13: Our ConvNet model replaces traditional classifiers such as softmax. It takes as the whole input image features and outputs multi-dimentional data

the output layer, which performs text detection and regression. A description of the architecture can be found in Figure 4.13. The ConvNet weights are optimized using back-propagation using the open-source darknet framework [96].

Discussion: back-propagation text search

Another potential model is back-propagation weakly text localization [109]. Let us consider an image I and a class c, the back-propagation saliency map can be represented as the gradient of class score $S_c(I)$ with respect to the image: $g_c(I) = \frac{\partial S_c(I)}{\partial I}$. Intuitively, the pixels that are closely related to the text affect changes in S_c more, which means that nearby regions of pixels would have high values in saliency map. However, this model assumes that the most part of image is text object. This strong assumption may not hold for most of scene images. In practice this model would have to come in a pipeline that first segments the image into pieces that only help which contain individual text objects.

4.4 Experiments

In order to compare the performance of our approach against standard benchmarks and state-of-the-art, we evaluated our method on two commonly used benchmark datasets.

4.4.1 Datasets

We evaluate our methods for text localization on two public datasets: (i) **ICDAR03** robust text locating and (ii) Street View Text (**SVT**) datasets. Both of them have bounding box annotations.

International Conference of Document Analysis and Recognition (ICDAR) including multiple datasets, labelled by year. The ICDAR03 dataset consists of 432 images in total, and 251 images are used for test. The dataset has word and character bounding box annotations.

The Street View Text (SVT) dataset [91] contains 647 words and 3796 letters in 249 images taken from Google Street View. The dataset is more challenging because most of the images come from business signs and exhibit a high degree of variability in appearance and resolution. In addition, due to the limited number of scene images which contains text in the training process, we collected a new scene text training dataset which contains 1000 street view images along with the annotations.

4.4.2 Training and Implementation Details

We pre-train our ConvNet layers on the training dataset. The ConvNet consists of 22 layers – 15 Conv layers, 5 pooling layers, and 2 fully connected layers layers. Each layer contains learnable parameters and a linear transformation. The dropout procedure is used to randomly zeroing a proportion of the parameters, and we set the proportion as 0.5. The network takes an image which is resized to 448×448. We use the momentum of 0.9 and a decay of 0.0005. The outputs predicting a sub-regions size 11×11. Our ConvNet is trained on entire scene images from ICDAR03, SVT and our collected 1000 training sets. Various methods

augmented their datasets with synthetic text training examples in order to improve model performance [86, 81]. Note that, however, due to the relatively small training set, and [110] shows features can be transferred from general to specific by the last layer of the network. We copied the pretrained parameters trained on ImageNet [15] challenge as our initial weights to improving performance. Error bac-prorogation was applied to update the new parameters.

4.4.3 Quantitative Evaluation

To evaluate the performance, we followed the standard evaluation protocol (precision, recall and F-measure) previously used in ICDAR03 [111]. A localization result is considered to be correct (true positive) if the overlap ratio between the ground-truth and the detected text above 0.5. The overlap ratio between two rectangular bounding boxes b_1 and b_2 is defined as the ratio of intersection over union (IoU): $\frac{b_1 \cap b_2}{b_1 \cup b_2}$. The definitions of precision and recall are: $precision = \frac{|TP|}{|E|}$, $recall = \frac{|TP|}{|T|}$, where $|T|$ and $|E|$ are the number of ground-truth and estimated bounding boxes, $|TP|$ indicates the number of true positive. The harmonic F-score f combine the precision and recall and denotes as $f = \frac{2 * precision * recall}{precision + recall}$. Since the focus of this work is solely on text detection, we did not using the feedback from recognition step to assist detection accuracy.

Experimental results comparisons are shown in Table 4.1. As can be seen, our method achieved 0.92 and 0.89 precision, 0.67 and 0.53 recall on ICDAR03 and SVT respectively. The precision is significantly higher than most baselines and comparable with the region propose based methods [89]. Our recall is notably less than [89] due to the bounding box failed to catch the small texts. It is worth noting that [89] builds on an extremely large dictionary (contains 90M training

Table 4.1: Text localization accuracy performance comparison on the ICDAR03 and SVT (P denotes precision, R denotes recall and f represents the F-score)

	ICDAR03			SVT		
Algorithm	**P**	**R**	**F**	**P**	**R**	**F**
Sliding windows + ConvNet + Beam Search[86]	0.72	0.51	0.59	0.54	0.30	0.38
Region proposal + ConvNet + recognition assist [89]	**0.96**	**0.85**	**0.90**	0.85	**0.68**	**0.76**
SWT [82]	0.73	0.60	0.66	-	-	-
Bandlet SWT [112]	0.76	0.66	0.71	-	-	-
MSER [83]	0.59	0.55	0.57	-	-	-
Our approach	0.92	0.67	0.78	**0.89**	0.53	0.66

data and 90k words) to enhance the performance; in contrast, our model is only trained with thousands of training images. Furthermore, note that although a small lexicon was provided in SVT, we did not utilize this information, this feature allows our method well-suited to the practical application since specialized lexicons are not readily available.

The method involved was implemented on a machine with an Intel i7-4770 (3.4 GHZ) CPU. When run on single GPU (Tesla C2075), the cost of localization on a typical SVT image and ICDAR is approximately 0.10 second per frame. Since we first resized the image, the size of image has no significant effects on the calculation time. With removing the step of region proposal and consequently avoiding a large number of filtering processes, our approach outperforms previous methods by a wide margin, it is approximately 50 times faster than [89], and orders of magnitude faster than the conventional [86]. For region based approaches, we observed that there is a considerable efficiency difference. They spent less than 1 second when handle very clean and small image, but cost more than 60 seconds when deal with complicated SVT images. The complexity of background has

critical impact on the process of their post-processing. Correspondingly, [82] and [83] tends to introduce more false positives when address the challenging SVT, and consequently cost much more time to eliminates the no-text regions. This confirms that the manually designed features are difficult to address complex situations. Figure 4.14 shows more results from our approach.

4.5 Conclusions

In this work, we tackled the text localization problem based on ConvNet. Our model combines the text detection and region regression into one ConvNet, and offers a practical way to speed up text localization. We presented results on two text benchmarks ICDAR03 and SVT. Our results show that the efficiency in terms of computation cost were improved by a wide margin. Our approach requires only one ConvNet forward passing, in spite of the simplicity, however, it roughly 50 times faster than [89] and accelerates the [86] by orders of magnitude. We limit ourselves to modeling English texts and digital numbers, the methods should be able to be extended to a multilingual localizer. There are a number of limitations in our approach, first, the scene image I must be resized. This is problematic for text images, because the aspect ratio is an important cue in text detection. Second, we simply treat text as regular object, the ConvNet is difficult to determine the correct text boundary without lexicon information when two texts are close. Our plans for future work include investigation the role of scene resized sub-regions and the refinement of bounding boxes.

Figure 4.14: Our fast search approach results on SVT and ICDAR datasets.

Chapter 5

Object Detection in LIDAR-based Point Clouds

In recent years, 3D object detection becoming a fundamental task for various challenged application domains such as scene understanding, robot navigation and autonomous driving [113, 114, 115, 5]. For example, the autonomous driving system has to plan the path, drive to the free space and avoid other vehicles on the highway, it's critical to be able to predict accurate vehicle's 3D localization, orientation and distance.

ConvNet and other deep learning techniques have been achieved human-level performances in 2D images from previous chapters introduction. While the 2D object detection approaches are very useful in many applications such as Internet-based image retrieval and recognition, it is much less popular in 3D applications. Recent approaches have extended and employed ConvNet in video classification [116, 117, 118] to recognize human actions and detect abnormal events. In their works, video data was addressed as 3D tensor with one temporal dimension was incorporated into a sequence of spatial images. In the majority of these works,

the input to the network is a stack of consecutive video frames, so the model is expected to implicitly learn spatio-temporal motion-dependent features in the hierarchical layers. Previous ConvNets have demonstrated good performance upon video datasets, note that most of them consider the sequence as a segmented object, and thus only simple classification task was devoted to.

Although it's beneficial to a number of tasks, 3D data investigation is much less investigated and among the greatest challenges in computer vision. One of the reasons is it's hard to collect large amounts of data. In modern times, it's much easier to capture 3D data, such as Stereo Imager, Google project Tango, Microsoft Kinect, Intel RealSense, all lead to a renewed interests in 3D object recognition. In this thesis, we emphasis on a special 3D data point: LIDAR-based point clouds. More precisely, we address the problem of vehicle detection with only data from LIDAR sensors (Figure. 5.1 show a 3D environment with vehicles labeled from KITTI[1]). The ConvNet is performed on an $300 \times 300 \times 40$ occupancy grid, each voxel (volume pixel) covering a small patch of $0.1 \times 0.1 \times 0.1$. We present a simple yet efficient model to fusion the multiple layers of top-down view data. In this chapter, we present a simple yet efficient model to fusion the 2D and 3D Conv layer into a ConvNet. Experiments show that the fusion yields significantly better performance than the projection-based method.

5.1 Autonomous Driving and Sensors

On average, more than a million people are killed in road crashes each year [119], and most of the traffic accidents can be avoid by being more careful[2]. At the same

[1] A benchmark datasets gathered of urban street scenes in German.

[2] This is particularly true given that 39 percent of the crash fatalities in 2011 involved alcohol use by one of the drivers

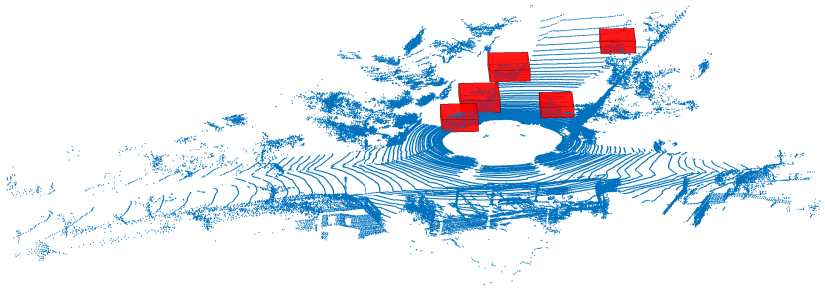


Figure 5.1: The LIDAR point cloud data from KITTI dataset, the vehicles are labeled with red bounding boxes.

time, we are getting closer and closer to allowing robotic or vehicle to interact with the real world. Autonomous driving cars have the potential to significantly enhance the traffic safety and thus change our transportation. Besides, some of the benefits of this task include energy and pollution consumption reduction.

The research on the intelligent vehicle have made progresses for several decades [120, 121, 122, 123]. In recent years, many high-tech companies leading the charge in this task. For example, Google has endeavor to promote autonomous driving for almost a decade [124]. Tesla, Audi and Toyota also unveiled their autonomous driving vehicle prototype.

To perform fully (or partially with the assistance of driver[3]) autonomous driving, a autonomous system needs to equip with a basic set of capabilities. More specificity, it must be able to avoid obstacles, detect where are the traffic lights and road signs, perceive the surroundings accurately and quickly, and most importantly find the free space to plan path for future steps. To this end, the advanced sensors to gather information about the environment is critical.

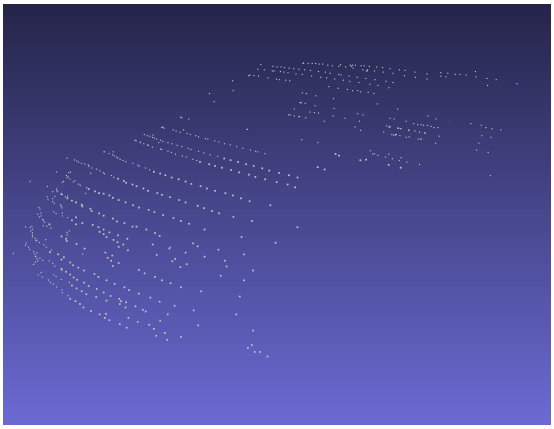
[3]for example, an emergency intervention to avoid a collision

Each sensor has its own strengths and weakness. Range sensors are heavily used in obstacle avoidance, mapping and navigate safely through environments. Stereo Imager are able to provide depth information. While image capture device is arguably the best sensor to mount in the vehicle. For example, NVIDIA cooperated with Audi released a demo on CES 2017, the vehicle count only on vision sensor to control the car without human intervention involved in the driver's seat. It has become increasingly clear, however, that a purely vision sensor-enabled vehicle cannot adequately and safely address the problem of road scene understanding. Indeed, image is a rich cue to object detections, however, from the application standpoint, unless there is a reason to believe that the obstacles in the scene, it's a necessary to incorporate multi-sensors in the autonomous driving system. Therefore, the most popular sensor modalities for object detection deployed in the robotics community are the combination of cameras and lasers.

5.1.1 Point Cloud and Velodyne LIDAR

There are two categories of sensors: passive and active sensors. Different with passive sensor, such as camera (without flash) which measure the reflected light. LIDAR is an active sensor. It measures distance using laser light. The light reflected from the measured surface is analysed and the data stored as a cloud of points. For this reason, one of the key advantages of active sensor is being able to collect imagery night and day, as well as through clouds and various weather conditions. Most importantly, LIDAR is able to produce a 3-dimensional map of surface coordinates in a scene, and can reveal underlying structural information in great detail. At the same time, the laser scan formed is a collection of points in the 3D space, and is sometimes referred to as a " point cloud ". However, compared

(a) Chair 3D data from ModelNet10



(b) Car LIDAR data from

Figure 5.2: The example of a regular CAD data and a sparse LIDAR data.

Table 5.1: Specifications of 2 popular Velodyne LIDARs.

Key Features	**HDL-64E**	**VLP-16**
number of channels	64	16
range	120m	100m
number of points per second	2.2 Million	$300,000$
Horizontal FOV[4]	360°	360°
Vertical FOV	26.9°	30°
Accuracy	< 2 cm	~ 3 cm
Horizontal Resolution	$\sim 0.08 - 0.35^\circ$	$\sim 0.1 - 0.4^\circ$
Vertical Resolution	$\sim 0.4^\circ$	2°

with vision methods, laser data is inherently highly sparse. For example, a given car object (Figure 5.2b) contain only less than 100 point clouds, but a CAD chair may contain more than $200,000$ points.

Our project use Velodyne LIDAR. The resolution of Velodyne LIDAR scanners can be very good, achieving an accuracy of a few mini-meter at 100m range. Velodyne LIDAR (Table 5.1) capable of a full 360° horizontal view which making it more informed about the environment than a human driver.

5.2 Point Cloud Projection

Since a 2D image is a projection of the 3D world and 2D image recognition is well-investigated. Intuitively, one potential approach is projecting 3D data into lower dimensional image planes. From the standpoint of view angle, we have to options: top down view and front view.

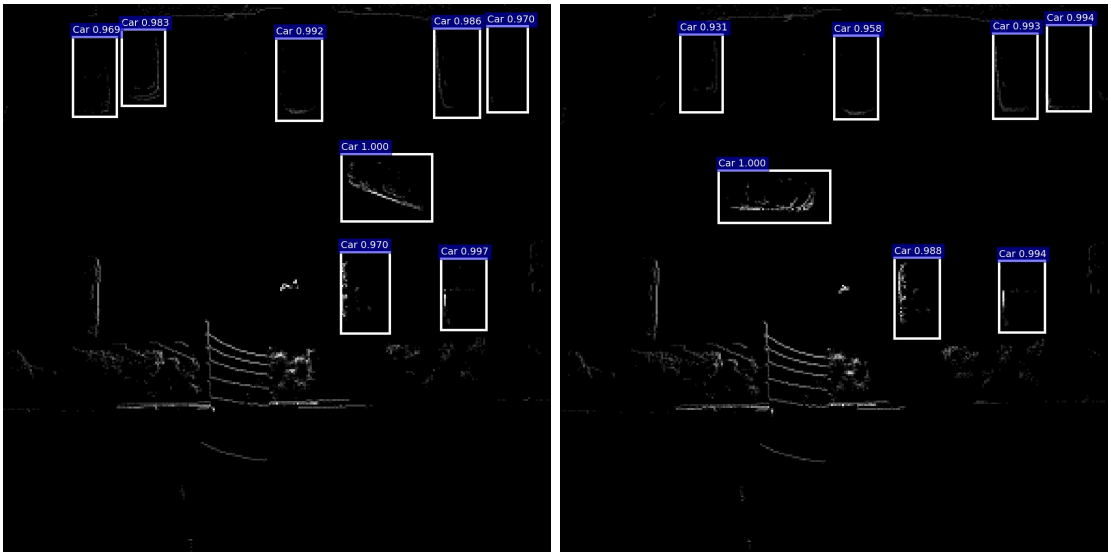
Top Down View

For any given LIDAR 3D data, we target at the effective rage of the 3D space $[-15, 15]$ meters in x-axis, $[0, 30]$ meters in y-axis, and $[-2, 2]$ meters in vertical direction (z-axis). Given these range, the points which located outside this cube bounding box are then being removed. For all the preserved data, we encode the 3D data by top-down bird's view with pixel resolution of 0.1 meter, resulting in a $300 \times 300 \times 40$ volume. The top down image can be simply generated by eliminating the z-axis.

Using region proposal network approaches such as Faster-rcnn [18], we are able to produce the 2-D vehicle bounding box, then the 3D bounding boxes can be back-projected using a pre-defined z value. Top-down view detection is shown to be sufficient for detecting objects with a limited ranges (Figure 5.3). The problem is that we have to customize the range of z, in other words, the vehicle will be occluded if there exists other objects on the top of the car. This limited the algorithm can not work on general situation.

Front View Cylinder Plane

An alternative to top-down view is front-view image plane [125]. In front view, a coordinate vector in the spherical coordinate system is defined as $m = (r, \theta, \phi)^T$,



(a) (b)

Figure 5.3: Illustration of top-down view Velodyne VLP-16 projection and detection bounding boxes.

where r is the distance range, θ is azimuth or horizontal angle, and ϕ the elevation component (Figure 5.4). The coordinates are constrained so that $r \geq 0, -\pi < \theta \leq \pi$, and $-\pi/2 < \phi \leq \pi/2$. A point in Cartesian coordinates is represented by $p = (x, y, z)^T$. The transformation from the spherical to the Cartesian coordinate system is given by: $p = rv$. where v is a unit direction vector defined as:

$$v = \begin{bmatrix} sin\theta cos\phi \\ sin\phi \\ cos\theta cos\phi \end{bmatrix} \tag{5.1}$$

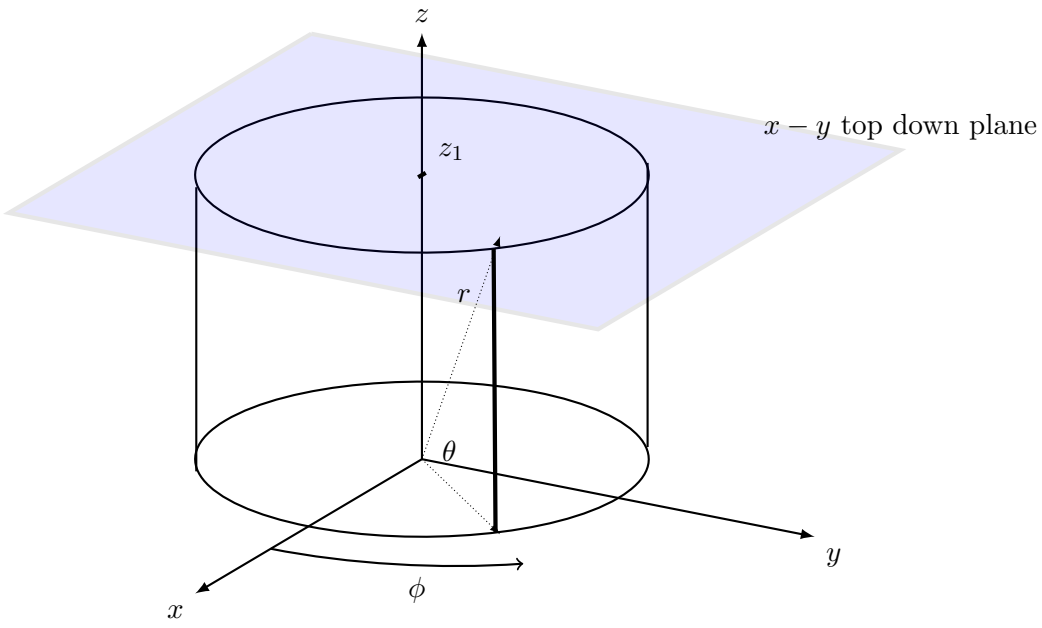


Figure 5.4: An illustration of front view cylinder coordinate and panel of top-down view.

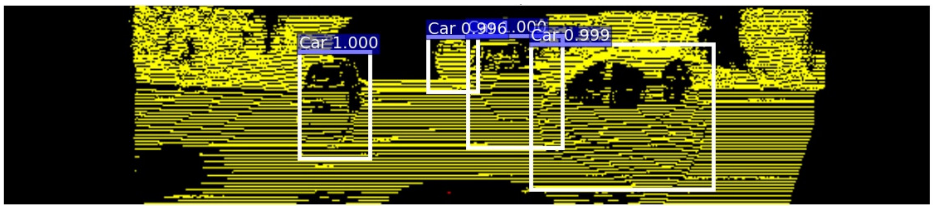
The transformation from Cartesian to spherical coordinate system is given by:

$$m = \begin{bmatrix} r \\ \theta \\ \phi \end{bmatrix} = \begin{bmatrix} \sqrt{x^2 + y^2 + z^2} \\ \arctan(y/x) \\ \arcsin(z/\sqrt{x^2 + y^2 + z^2}) \end{bmatrix} \tag{5.2}$$

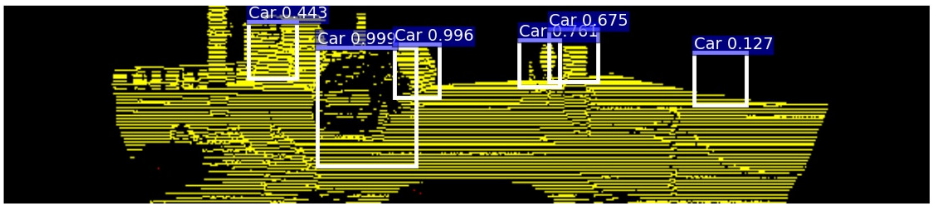
Similar with [125], we define the image location variables:

$$p = \left\lfloor \frac{\theta}{\Delta\theta} \right\rfloor \qquad \text{and} \qquad q = \left\lfloor \frac{\phi}{\Delta\phi} \right\rfloor$$

Then our front-view image can be generated by fill the element at (p, q) with 2-channel data (r, z). Elements in positions where no points are projected into are filled with $(r, z) = (0, 0)$.



(a)

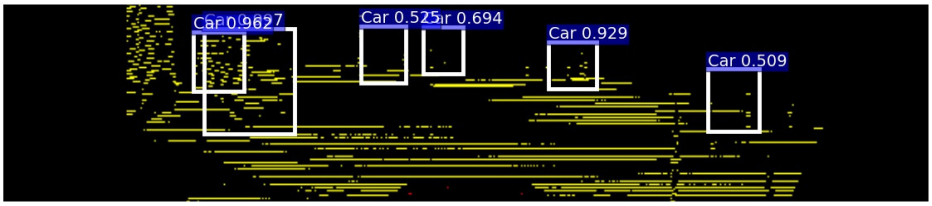


(b)

Figure 5.5: Car detection experiments with faster-rcnn approach with 64 beams front-view image.

Different with [125] which employs fully ConvNets based method [126], we apply the faster-rcnn's strategy in the front view car detection and achieved the similar performance with [125] reported on KITTI datasets (Figure 5.5). However, the problem raised when we simulate the VLP-16 data by down-sampled the 64-beams into 16-beams, there is no obvious vehicle pattern shown in the down-sampled image (Figure 5.6).

(a)



(b)

Figure 5.6: Car detection experiments with faster-rcnn approach with down-sampled 16-layer front-view image.

5.3 3D Volume Representation

The projective nature of the image forming process means information loss is inevitable. In order to properly classify a cube or 3D bounding box, sliding window approach can be intuitively used to score the cuboid objectness. Various representations have been designed manually or learned from data. For example, sliding shapes [127] was proposed as a 3D object detector that runs sliding windows in 3D to directly classify each 3D window. On the other hand, SIFT and HOG are extended into SIFT-3D [128] was proposed to slide a 3D detection window in 3D space. HOG-3D [129]. However, adding a new dimension for moving the windows significantly enlarges the search spaces, and make this method infeasible to applying on the project.

Current state-of-the-art methods [130, 131, 132] rely on ConvNets to address the task of 3D detection. For the data structure, the majority of existing methods based on two categories of representation: volumetric representation and multi-view representation. The first one discretized spatially as binary voxels, while the latter one represent the 3D object as a set of projected 2D pixel images. In either case, there is a need to lifting from 2D neural net to 3D neural net. We concentrate on the volumetric representation and using occupancy grid maps to describe the 3D spatial correlation.

5.3.1 Occupancy Grid Maps

Directly using the LIDAR range measurement has a major drawback. First of all, no free space are modeled, it can not differentiate the free space and unmapped space. Then, original point clouds store large amounts of measurement points and hence are not memory-efficient. In addition, features based on point clouds often require spatial neighbourhood queries, which can quickly become intractable with large number of points. Therefore probabilistic occupancy grid map was first proposed to represent the raw LIDAR data by [133]. Note that the appearance of voxel grid map is uniformed discretization but highly depends on the resolution of voxel grid in space. Suppose we have 5 measurements, then given different pre-defined of resolutions, we can have different occupied states (Figure 5.7).

LIDAR measurements are afflicted with uncertainty: typically, the error in the range measurements is in the order of centimeters. Let z^t be a set of measurements, while $t = 1, 2, ...T$ be a sequence of range measurements that either hit $z^t = 1$ or pass $z^t = 1$ a given voxel with coordinates (i, j, k).

For the most intuitive "hit and pass through" model. We can define each voxel as:

$$l_{ijk}^t = min(l_{ijk}^{t-1} + z^t, 1) \tag{5.3}$$

To differentiate the free space and unknown space, there are two different methods to compute the grid: 1) Binary occupancy grid. In this model, each voxel is assumed to holds a probability value representing the degree of how occupied that voxel is by an obstacle. The probabilistic estimate of occupancy for each voxel is computed with log odds for numerical stability. Using the formulation

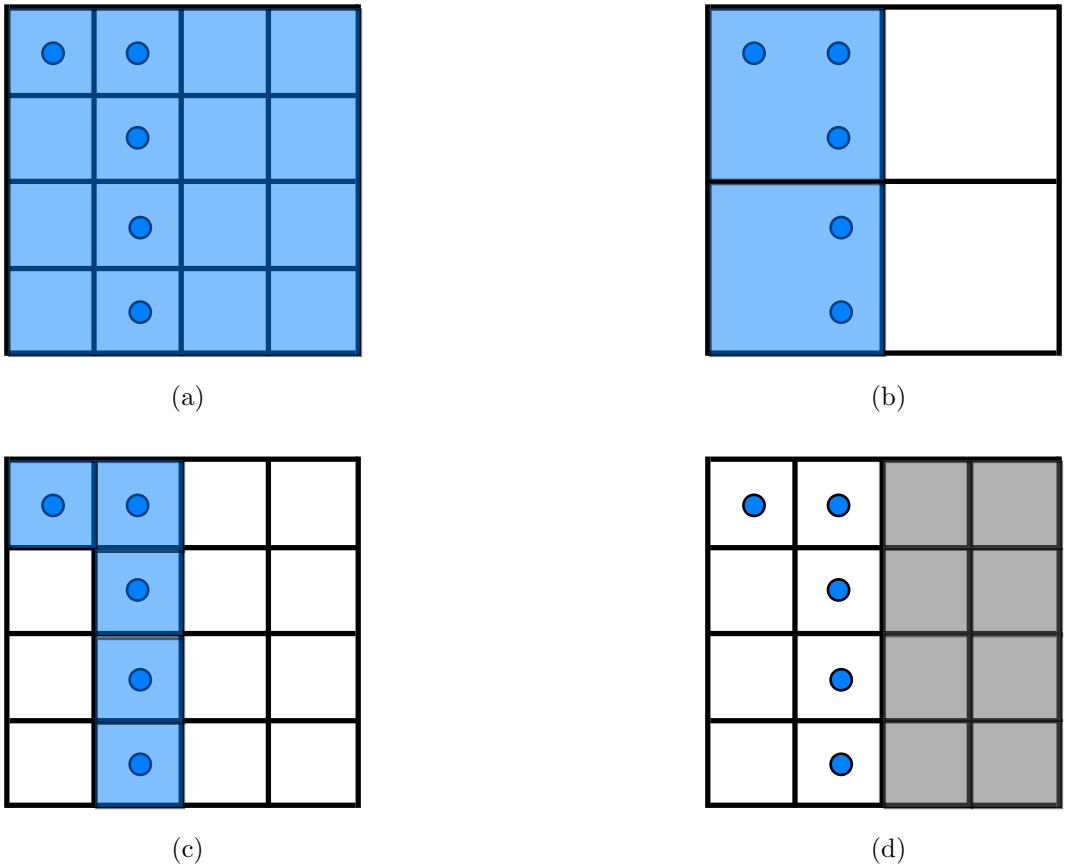


Figure 5.7: Illustration of states of grid in 4 resolutions. Blue denotes occupied, white denotes empty and grey means unknown state

from [134], we update each voxel traversed by the beam as:

$$l_{ijk}^{t} = l_{ijk}^{t-1} + z^{t} l_{occ} + (1 - z^{t}) l_{free} \tag{5.4}$$

Assuming $p(x)$ denotes the probability of current voxel's probability being occupied and has range of $[0, 1]$, then the l can be defined as:

$$l(x) = log \frac{p(x)}{1 - p(x)} \tag{5.5}$$

The initial probability of occupancy is set to $p^{0} = 0.5$, or $l^{0} = 0$.

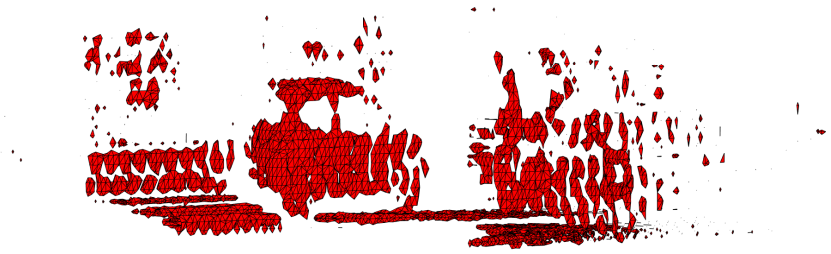


Figure 5.8: An example of KITTI data's surface of occupancy grid map.

2) Density grid. In this model each voxel stores a single value expressing the density, corresponding to the probability the voxel would block a sensor beam. We use the formulation from [135], where we track the Beta parameters α^t_{ijk} and β^t_{ijk}, with a uniform prior $\alpha^0_{ijk} = 1$ and $\beta^0_{ijk} = 1$. The update for each voxel affected by the measurement z^t:

$$\alpha^t_{ijk} = \alpha^t_{ijk} + z^t \qquad \text{and} \qquad \beta ijk^t = \beta^t_{ijk} + (1 - z^t) \tag{5.6}$$

The voxel's density can be represented as:

$$\mu^t_{ijk} = \frac{\alpha^t_{ijk}}{\alpha^t_{ijk} + \beta ijk^t} \tag{5.7}$$

It is important to note that our point cloud data is not a sequence of measurements. Thus, we do not differentiate the order of t and $t - 1$. We turn the point cloud into 3D voxels through a simple "hit and pass through" process, and an example of voxelization can be found on Figure 5.8.

5.3.2 3D Convolutional Neural Networks

Inspired by the ability of 2D ConvNet. The 3D convolution is achieved by convolving a 3D kernel to the volume formed by stacking multiple contiguous frames together. Formally, the value at position (x, y, z) with feature map size $f \times f \times f$ is given by:

$$v_{xyz}^{l} = \sum_{p=0}^{f-1} \sum_{q=0}^{f-1} \sum_{r=0}^{f-1} w_{pqr} v_{(x+p)(y+q)(z+r)}^{l-1} \tag{5.8}$$

Similarly, the 3D (max) pooling layer with a pooling kernel of $g \times g \times g$ can be represented as:

$$v_{xyz}^{l} = \max_{i,j,k \in \{0,1,\ldots,g-1\}} v_{(gx+i)(gy+j)(gz+k)}^{l} \tag{5.9}$$

A comparison of 2D and 3D convolutions is given in Figure 5.9. Note that 3D Conv layer is different with multi-channel image convolution. In RGB image classification, the channel will be combined with multiple 2D convolutional operation. In contrast, 3D Conv layer perform convolution along 3 different Axes (Fig 5.10).

Small Object Detection with 3D Models

2D models could not detect small objects such as pedestrian due to their project nature. A number of works [136, 137, 138] proposed to extract features for retrieval and classification of 3D models. However, it also suggests that 3D ConvNet is more time-consuming to searching 3D sliding windows. In order to address this

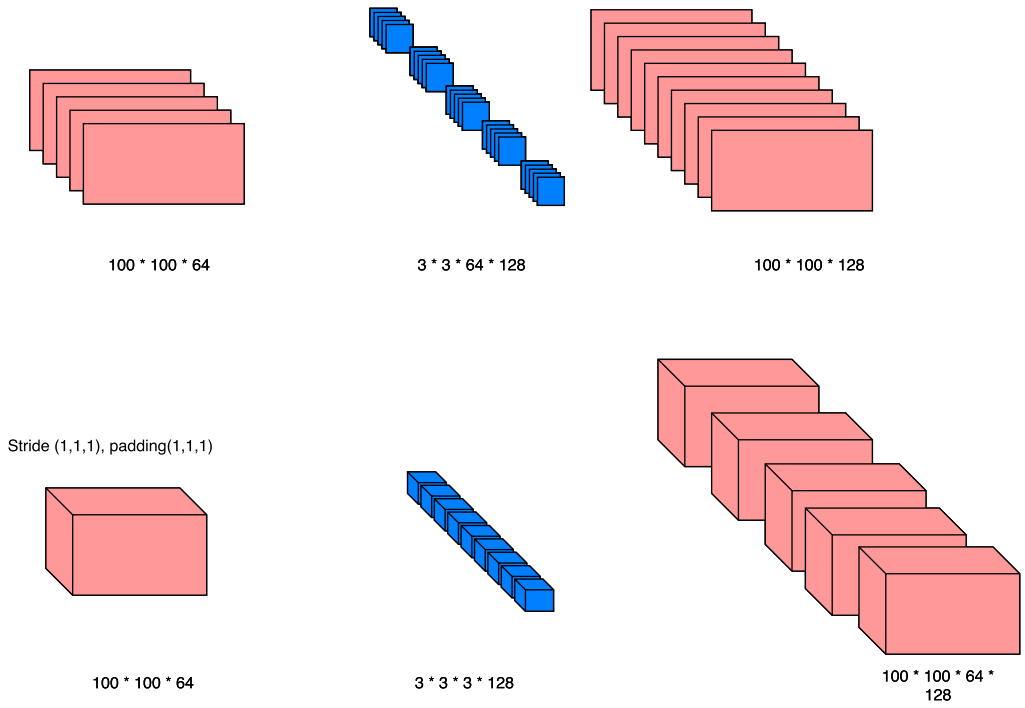


Figure 5.9: Comparison of 2D and 3D convolutions. In the second row of the figure, the size of the convolution kernel in the temporal dimension is 3, and the sets of connections are color-coded so that the shared weights are in the same color.

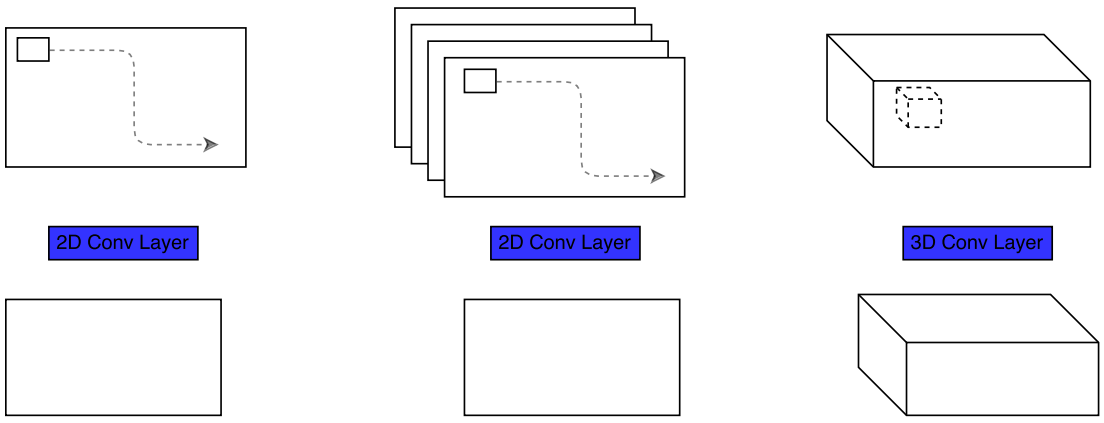
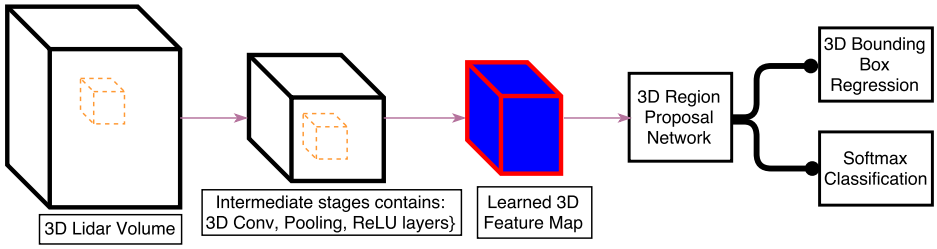
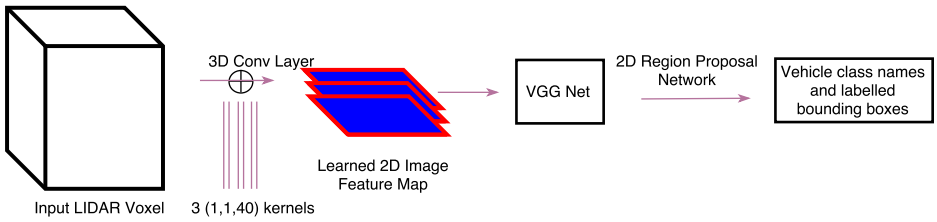


Figure 5.10: The illustration of 2D Conv layer, 2D Conv layer on multi-channel images and 3D Conv layer on 3D data.

problem, we extended the region proposal network (RPN) in [18] to 3D RPN. Our proposed network takes a 3D volume as input and output a set of 3D fixed-scale bounding boxes with objectness scores. In order to avoiding the problem of using too many 3D Conv Layers, while still preserving a 3D voxel input, we fuse multiple 2D Conv layers and single 3D layer into a network. In other words, we input the 3D occupancy grid which preserve more 3D spatial information, and using multiple 2D Conv layers to achieve the computational efficiency. We generate a 2D top-down image by means of using 3D ConvNet. The underlying idea is quite simple, we simply insert one 3D Conv layer in the first layer of faster-rcnn. Our model's architecture are illustrated in Figure 5.11.



(a)



(b)

Figure 5.11: Proposed 3D network architectures. (a): 3D RPN network. (b): Mixed 2D and 3D network

We initialized the parameter in the first layer is set as 1 instead of the Gaussian randomized value, because this is actual equivalent to the top-down 2D image. The parameters after the first layer are initialized with VGG-16 network that was pre-trained to perform classification on the ImageNet dataset [5]. The network

consists of one $1*1*40$ fully connected layer and eight 3×3 Conv layers (followed by ReLU non-linearities) and five 2×2 max-pooling layers and has shown excellent performance. For the training loss function, we use the same multi-task (log) loss proposed in [18] and [16]:

$$L(p_i, t_i) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*) \tag{5.10}$$

where i is the index of an anchor in and p_i is the predicted probability of anchor i being an object. The ground truth label p_i^* is 1 if the anchor is positive, and is 0 if the anchor is negative. t_i is a vector representing the coordinates of the predicted bounding box, t_i^* is the labeled region ground truth. L_{cls} and L_{reg} are log loss over two classes and regression loss respectively. N_{cls} is the mini-batch size and N_{reg} denotes the number of anchor locations. The interested reader is referred to the original paper [18], for detailed discussions of the faster-rcnn paradigm.

5.4 Experiments and summary

Our first goal of experiment is to show that 3D model is able to capture the 3D spatial information. We trained and tested classification experiments for first 8 categories in ShapeNet CAD data [130]. The input layer accepts a fixed-size volume grid of $30 \times 30 \times 30$ voxels. The training loss and top-1 testing accuracy are shown in Figure 5.12. The plot suggests that a 7-layer 3D ConvNet achieve the classification accuracy above 92% after only 2 epochs training. We then evaluate our approach on the collected VLP-16 datasets (Figure 5.13). The data sets consists of 4 captured package videos with of 10 fps frequency. Again, we only concentrate on the still LIDAR frame, no temporal information is involved

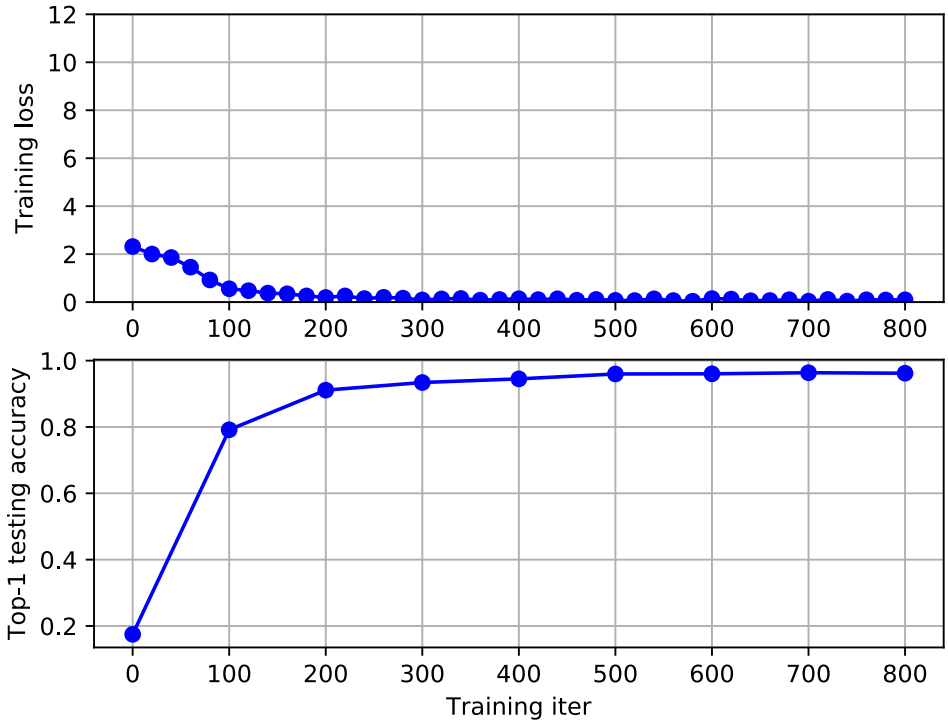


Figure 5.12: Training loss and top-1 accuracy on ShapeNet 3D data.

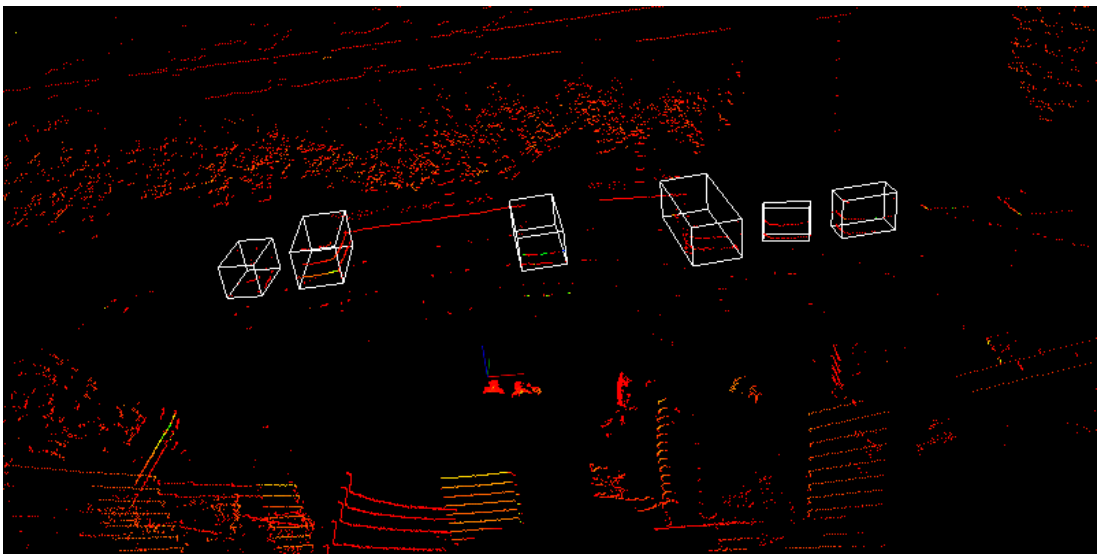


Figure 5.13: Car detection results demo using Velodyne VLP-16.

in the experiments.

To evaluate the effectiveness of the 2D models, we prepare the projected data follow the paradigm we introduced in section 5.2. We report the results of 2D image-based detection results with KITTI 64 beams data and VLP-16 data. In KITTI dataset, the 7480 training LIDAR frames was divided into 6000 for training and 1480 for testing. All methods start from the same pre-trained VGG16 network and the performance measurement metric we used is from benchmark KITTI: a true positive detection is considered when the IoU overlap between bounding box and ground truth > 0.7. We set the detection threshold score as 0.75, and experiment show our top-down view approach (52%/85%) is better than the front-view method (49%/72%) by precision in both VLP-16 and KITTI 64 data respectively. For the computation cost, we use one NVIDIA GeForce Titan X GPU in our experiments. Both top-down and front-view image-based approaches takes around 100 ms (10Hz). For the pedestrian detection, both projection approaches are failed to detect the pedestrian bounding boxes. The 3D

models with small object detection is still on the ongoing stage but the preliminary experiments showing they can generate cube candidates but with a large number of false detection. In current stage, the critical difficulty is the lack of training data with pedestrian in a short range. In this chapter, We first introduce the LIDAR sensor and the sparse point clouds. We then show that projection based methods can be used to detect vehicles. Next we describe the potentially promising network in which direct extension from 2D ConvNet to 3D ConvNet and incorporate 2D and 3D in a network.

Chapter 6

Summary and Contributions

We examined the general theme of object classification and detection in the context of three particular tasks with different number of dimensions. The first case of this thesis addressed pain prediction problem. First of all, we converted the original collected attributes into binary features, thus avoiding the hard problem of how to represent highly-diverged attributes of collected data. Generally, medical data prediction involves explicitly selecting the right set combination of features, which needs the much effort on trial and error. We employed the discriminative RBM to directly solve the problem of feature selection and prediction simultaneously. We also observed the discriminative RBM appear to yield better accuracy than to the classical PCA and LDA with SVM classifier methods. To the best of our knowledge, our work is the first one reported to incorporate discriminate RBM (one-layer DBN) in the pain management research.

The next chapter dealt with the task of 2D text object localization. Chapter 4 first directly applying off-the-shelf ConvNet classifier in sliding window manner. This, however, significantly increased the computation time. We then employed the saliency mechanism in ConvNet to quickly localize the text. Driven by the

demand for real-time application, our work aims to speed up the localization and robust detection under extreme environment such as seriously blurring and degradation. To achieve these, our main contributions are as follows: 1) We directly track text as an object rather than detect characters and group them together. In the processing of grouping, the majority of existing methods require a fixed-sized lexicon dictionary which reduce the generality of their models. Our approach does not need the lexicon information. 2) Our pre-trained ConvNet automatically output the coordinates of text bounding boxes associated scores. The detections are performed in the manner of single-shot forward passing. We significantly improved the computation efficiency by a large margin. 3) We utilize the ConvNet to represent whole scene image rather than text-only features, and encode the contextual spatial information into localization. In addition, the majority of existing text detection method's performance heavily rely on the complexity of image background. In contrast, the pipeline represented in this thesis does not have this limitation.

We then turned to object detection in LIDAR. Chapter 5 begins by developing data projection-based approaches. We first investigated front view projection and proposed our top-down bird view projection, 2D region-based ConvNet was then performed on both projections. To fully utilize the rich 3D information, volumetric approaches were received much attention. However, 3D models generally need much more parameters and thus more computational cost. In this thesis, we describe two preliminary alternative approach which efficiently extend 2D RPN to 3D RPN and integrate 2D and 3D ConvNet. Experiments on KITTI and our collected 16-layer LIDAR data demonstrates that proposed bird view projection method outperforms the front-view projection, and the 3D model are potentially able to detect small object with large number of training data.

6.0.1 Publications

- L. Yang, J. Hua, M. Bittner, S. Cheng, A Factor Graph Based Method for Cell Image Tracking and Apoptosis Analysis. *IEEE International Symposium on Biomedical Imaging (ISBI 2014), May, 2014, Beijing, China.*

- L. Yang, J. Ma, S. Cheng, J.B. Son, J. Hazle, B. W. Carter, and S. Lin. Quantitative ADC Measurement Analysis. *AAPM2015, work done while at MD Anderson) – 2015.*

- L. Yang, S. Wang, X. Jiang, S. Cheng, and H. Kim. PATTERN: Pain Assessment for paTients who can't TEll using Restricted Boltzmann machiNe. *BMC Informatics and Decision Making.*

- L. Yang, S. Cheng, S. Wang and P. K. Verma. Text Search: Towards Fast Text Localization in Scene Images. *IEEE International Symposium on Multimedia (ISM2016), San Jose, CA, USA.*

Bibliography

[1] Androutsopoulos, I., Koutsias, J., Chandrinos, K.V., Paliouras, G., Spyropoulos, C.D.: An evaluation of naive bayesian anti-spam filtering. arXiv preprint cs/0006013 (2000)

[2] Guyon, I., Weston, J., Barnhill, S., Vapnik, V.: Gene selection for cancer classification using support vector machines. Machine learning **46**(1-3) (2002) 389–422

[3] Galindo, J., Tamayo, P.: Credit risk assessment using statistical and machine learning: basic methodology and risk modeling applications. Computational Economics **15**(1-2) (2000) 107–143

[4] Viola, P., Jones, M.J.: Robust real-time face detection. International journal of computer vision **57**(2) (2004) 137–154

[5] Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, IEEE (2012) 3354–3361

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

[7] Larochelle, H., Bengio, Y., Louradour, J., Lamblin, P.: Exploring strategies for training deep neural networks. Journal of Machine Learning Research **10**(Jan) (2009) 1–40

[8] Domingos, P.: A few useful things to know about machine learning. Communications of the ACM **55**(10) (2012) 78–87

[9] Jolliffe, I.: Principal component analysis. Wiley Online Library (2002)

[10] Schölkopf, B., Smola, A., Müller, K.R.: Kernel principal component analysis. In: International Conference on Artificial Neural Networks, Springer (1997) 583–588

[11] Hyvärinen, A., Oja, E.: Independent component analysis: algorithms and applications. Neural networks **13**(4) (2000) 411–430

[12] Lowe, D.G.: Object recognition from local scale-invariant features. In: Computer vision, 1999. The proceedings of the seventh IEEE international conference on. Volume 2., Ieee (1999) 1150–1157

[13] Olshausen, B.A., Field, D.J.: Sparse coding with an overcomplete basis set: A strategy employed by v1? Vision research **37**(23) (1997) 3311–3325

[14] Cybenko, G.: Approximation by superpositions of a sigmoidal function. Mathematics of Control, Signals, and Systems (MCSS) **2**(4) (1989) 303–314

[15] Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, IEEE (2009) 248–255

[16] Girshick, R.: Fast r-cnn. In: Proceedings of the IEEE International Conference on Computer Vision. (2015) 1440–1448

[17] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2016) 770–778

[18] Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems. (2015) 91–99

[19] Moyer, C.: How google's alphago beat a go world champion. The Atlantic, March **28** (2016)

[20] Hochreiter, S.: The vanishing gradient problem during learning recurrent neural nets and problem solutions. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems **6**(02) (1998) 107–116

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

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

[23] Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Cognitive modeling **5**(3) (1988) 1

[24] Schölkopf, B., Burges, C.J.: Advances in kernel methods: support vector learning. MIT press (1999)

[25] Hinton, G.E., Osindero, S., Teh, Y.W.: A fast learning algorithm for deep belief nets. Neural computation **18**(7) (2006) 1527–1554

[26] Hinton, G.: A practical guide to training restricted boltzmann machines. Momentum **9**(1) (2010) 926

[27] Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th international conference on machine learning (ICML-10). (2010) 807–814

[28] Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research **15**(1) (2014) 1929–1958

[29] Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. (2012) 1097–1105

[30] Vinyals, O., Toshev, A., Bengio, S., Erhan, D.: Show and tell: A neural image caption generator. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2015) 3156–3164

[31] Toshev, A., Szegedy, C.: Deeppose: Human pose estimation via deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2014) 1653–1660

[32] Grafton, S.T., Hamilton, A.F.d.C.: Evidence for a distributed hierarchy of action representation in the brain. Human movement science **26**(4) (2007) 590–616

[33] Ciresan, D., Giusti, A., Gambardella, L.M., Schmidhuber, J.: Deep neural networks segment neuronal membranes in electron microscopy images. In: Advances in neural information processing systems. (2012) 2843–2851

[34] Svozil, D., Kvasnicka, V., Pospichal, J.: Introduction to multi-layer feed-forward neural networks. Chemometrics and intelligent laboratory systems **39**(1) (1997) 43–62

[35] Mikolov, T., Karafiát, M., Burget, L., Cernockỳ, J., Khudanpur, S.: Recurrent neural network based language model. In: Interspeech. Volume 2. (2010) 3

[36] Donahue, J., Anne Hendricks, L., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., Darrell, T.: Long-term recurrent convolutional networks for visual recognition and description. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2015) 2625–2634

[37] Graves, A., Schmidhuber, J.: Framewise phoneme classification with bidirectional lstm and other neural network architectures. Neural Networks **18**(5) (2005) 602–610

[38] Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., Hullender, G.: Learning to rank using gradient descent. In: Proceedings of the 22nd international conference on Machine learning, ACM (2005) 89–96

[39] Recht, B., Re, C., Wright, S., Niu, F.: Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In: Advances in Neural Information Processing Systems. (2011) 693–701

[40] Vogl, T.P., Mangis, J., Rigler, A., Zink, W., Alkon, D.: Accelerating the convergence of the back-propagation method. Biological cybernetics **59**(4) (1988) 257–263

[41] Van Laarhoven, P.J., Aarts, E.H.: Simulated annealing. In: Simulated Annealing: Theory and Applications. Springer (1987) 7–15

[42] Griewank, A.: Who invented the reverse mode of differentiation? Optimization Stories, Documenta Matematica, Extra Volume ISMP (2012) (2012) 389–400

[43] Iyer, M.S., Rhinehart, R.R.: A method to determine the required number of neural-network training repetitions. IEEE Transactions on Neural Networks **10**(2) (1999) 427–432

[44] Qian, N.: On the momentum term in gradient descent learning algorithms. Neural networks **12**(1) (1999) 145–151

[45] Mishkin, D., Matas, J.: All you need is a good init. arXiv preprint arXiv:1511.06422 (2015)

[46] Gulcehre, C., Moczulski, M., Denil, M., Bengio, Y.: Noisy activation functions. arXiv preprint arXiv:1603.00391 (2016)

[47] Maas, A.L., Hannun, A.Y., Ng, A.Y.: Rectifier nonlinearities improve neural network acoustic models. In: Proc. ICML. Volume 30. (2013)

[48] Utgoff, P.E., Stracuzzi, D.J.: Many-layered learning. Neural Computation **14**(10) (2002) 2497–2529

[49] Smolensky, P.: Information processing in dynamical systems: Foundations of harmony theory. Technical report, DTIC Document (1986)

[50] Taylor, G.W., Hinton, G.E., Roweis, S.T.: Modeling human motion using binary latent variables. Advances in neural information processing systems **19** (2007) 1345

[51] Mohamed, A.r., Hinton, G.: Phone recognition using restricted boltzmann machines. In: Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on, IEEE (2010) 4354–4357

[52] Strohbuecker, B., Mayer, H., Evers, G.C., Sabatowski, R.: Pain prevalence in hospitalized patients in a german university teaching hospital. Journal of pain and symptom management **29**(5) (2005) 498–506

[53] Simon, L.S.: Relieving pain in america: A blueprint for transforming prevention, care, education, and research. Journal of Pain & Palliative Care Pharmacotherapy **26**(2) (2012) 197–198

[54] Lippe, P.M.: The decade of pain control and research. Pain Medicine **1**(4) (2000) 286–286

[55] Stannard, C., Johnson, M.: Chronic pain management-can we do better? an interview-based survey in primary care. Current medical research and opinion **19**(8) (2003) 703–706

[56] McCaffery, M.: Patients in pain: What they say, and what they really mean. Director (Cincinnati, Ohio) **13**(2) (2005) 104–106

[57] Cook, K.F., Dunn, W., Griffith, J.W., Morrison, M.T., Tanquary, J., Sabata, D., Victorson, D., Carey, L.M., MacDermid, J.C., Dudgeon, B.J., et al.: Pain assessment using the nih toolbox. Neurology **80**(11 Supplement 3) (2013) S49–S53

[58] Godfrey, H.: Understanding pain, part 1: physiology of pain. British Journal of Nursing **14**(16) (2005)

[59] Briggs, E.: Understanding the experience and physiology of pain. Nursing Standard **25**(3) (2010) 35–39

[60] Melzack, R.: From the gate to the neuromatrix. Pain **82** (1999) S121–S126

[61] Wang, S., Jiang, X., Ji, Z., El-Kareh, R., Choi, J., Kim, H.: When you can't tell when it hurts: a preliminary algorithm to assess pain in patients who can't communicate. In: AMIA Annual Symposium Proceedings. Volume 2013., American Medical Informatics Association (2013) 1429

[62] Bishop, C.M.: Pattern recognition. Machine Learning **128** (2006) 1–58

[63] Belkin, M., Niyogi, P.: Laplacian eigenmaps for dimensionality reduction and data representation. Neural computation **15**(6) (2003) 1373–1396

[64] Salakhutdinov, R., Mnih, A., Hinton, G.: Restricted boltzmann machines for collaborative filtering. In: Proceedings of the 24th international conference on Machine learning, ACM (2007) 791–798

[65] Krizhevsky, A., Hinton, G.E., et al.: Factored 3-way restricted boltzmann machines for modeling natural images. In: International conference on artificial intelligence and statistics. (2010) 621–628

[66] Taylor, G.W., Hinton, G.E.: Factored conditional restricted boltzmann machines for modeling motion style. In: Proceedings of the 26th annual international conference on machine learning, ACM (2009) 1025–1032

[67] Fischer, A., Igel, C.: Training restricted boltzmann machines: An introduction. Pattern Recognition **47**(1) (2014) 25–39

[68] Jordan, M.I., Bishop, C.: An introduction to graphical models (2004)

[69] LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., Huang, F.: A tutorial on energy-based learning. Predicting structured data **1** (2006) 0

[70] Ackley, D.H., Hinton, G.E., Sejnowski, T.J.: A learning algorithm for boltzmann machines*. Cognitive science **9**(1) (1985) 147–169

[71] Hinton, G.E., Salakhutdinov, R.R.: Replicated softmax: an undirected topic model. In: Advances in neural information processing systems. (2009) 1607–1614

[72] Hopfield, J.J.: Neural networks and physical systems with emergent collective computational abilities. Proceedings of the national academy of sciences **79**(8) (1982) 2554–2558

[73] Hinton, G.: Training products of experts by minimizing contrastive divergence. Neural computation **14**(8) (2002) 1771–1800

[74] Mika, S., Ratsch, G., Weston, J., Scholkopf, B., Mullers, K.R.: Fisher discriminant analysis with kernels. In: Neural Networks for Signal Processing IX, 1999. Proceedings of the 1999 IEEE Signal Processing Society Workshop., IEEE (1999) 41–48

[75] Larochelle, H., Bengio, Y.: Classification using discriminative restricted boltzmann machines. In: Proceedings of the 25th international conference on Machine learning, ACM (2008) 536–543

[76] Martínez, A.M., Kak, A.C.: Pca versus lda. IEEE transactions on pattern analysis and machine intelligence **23**(2) (2001) 228–233

[77] LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Backpropagation applied to handwritten zip code recognition. Neural computation **1**(4) (1989) 541–551

[78] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11) (1998) 2278–2324

[79] LeCun, Y., Kavukcuoglu, K., Farabet, C.: Convolutional networks and applications in vision. In: Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on, IEEE (2010) 253–256

[80] Nagy, G., Nartker, T.A., Rice, S.V.: Optical character recognition: An illustrated guide to the frontier. In: Electronic Imaging, International Society for Optics and Photonics (1999) 58–69

[81] Jaderberg, M., Vedaldi, A., Zisserman, A.: Deep features for text spotting. In: Computer Vision–ECCV 2014. Springer (2014) 512–528

[82] Epshtein, B., Ofek, E., Wexler, Y.: Detecting text in natural scenes with stroke width transform. In: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, IEEE (2010) 2963–2970

[83] Neumann, L., Matas, J.: A method for text localization and recognition in real-world images. In: Computer Vision–ACCV 2010. Springer (2011) 770–783

[84] LeCun, Y., Bengio, Y.: Convolutional networks for images, speech, and time series. The handbook of brain theory and neural networks **3361**(10) (1995)

[85] Ciresan, D., Meier, U., Schmidhuber, J.: Multi-column deep neural networks for image classification. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, IEEE (2012) 3642–3649

[86] Wang, T., Wu, D.J., Coates, A., Ng, A.Y.: End-to-end text recognition with convolutional neural networks. In: Pattern Recognition (ICPR), 2012 21st International Conference on, IEEE (2012) 3304–3308

[87] Coates, A., Carpenter, B., Case, C., Satheesh, S., Suresh, B., Wang, T., Wu, D.J., Ng, A.Y.: Text detection and character recognition in scene images with unsupervised feature learning. In: Document Analysis and Recognition (ICDAR), 2011 International Conference on, IEEE (2011) 440–445

[88] Huang, W., Qiao, Y., Tang, X.: Robust scene text detection with convolution neural network induced mser trees. In: Computer Vision–ECCV 2014. Springer (2014) 497–511

[89] Jaderberg, M., Simonyan, K., Vedaldi, A., Zisserman, A.: Reading text in the wild with convolutional neural networks. International Journal of Computer Vision **116**(1) (2016) 1–20

[90] Subramanian, K., Natarajan, P., Decerbo, M., Castañòn, D.: Character-stroke detection for text-localization and extraction. In: Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on. Volume 1., IEEE (2007) 33–37

[91] Wang, K., Babenko, B., Belongie, S.: End-to-end scene text recognition. In: Computer Vision (ICCV), 2011 IEEE International Conference on, IEEE (2011) 1457–1464

[92] Mishra, A., Alahari, K., Jawahar, C.: Top-down and bottom-up cues for scene text recognition. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, IEEE (2012) 2687–2694

[93] Smith, D.L., Field, J., Learned-Miller, E.: Enforcing similarity constraints with integer programming for better scene text recognition. In: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, IEEE (2011) 73–80

[94] Yi, C., Yang, X., Tian, Y.: Feature representations for scene text character recognition: A comparative study. In: Document Analysis and Recognition (ICDAR), 2013 12th International Conference on, IEEE (2013) 907–911

[95] Zitnick, C.L., Dollár, P.: Edge boxes: Locating object proposals from edges. In: Computer Vision–ECCV 2014. Springer (2014) 391–405

[96] Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. arXiv preprint arXiv:1506.02640 (2015)

[97] Erhan, D., Szegedy, C., Toshev, A., Anguelov, D.: Scalable object detection using deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2014) 2147–2154

[98] Redmon, J., Angelova, A.: Real-time grasp detection using convolutional neural networks. In: Robotics and Automation (ICRA), 2015 IEEE International Conference on, IEEE (2015) 1316–1322

[99] Hosang, J., Benenson, R., Schiele, B.: How good are detection proposals, really? arXiv preprint arXiv:1406.6962 (2014)

[100] Uijlings, J.R., van de Sande, K.E., Gevers, T., Smeulders, A.W.: Selective search for object recognition. International journal of computer vision **104**(2) (2013) 154–171

[101] Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient graph-based image segmentation. International journal of computer vision **59**(2) (2004) 167–181

[102] Dollár, P., Zitnick, C.L.: Structured forests for fast edge detection. In: Proceedings of the IEEE International Conference on Computer Vision. (2013) 1841–1848

[103] Dollár, P., Zitnick, C.L.: Fast edge detection using structured forests. IEEE transactions on pattern analysis and machine intelligence **37**(8) (2015) 1558–1570

[104] Itti, L., Koch, C., Niebur, E.: A model of saliency-based visual attention for rapid scene analysis. IEEE Transactions on Pattern Analysis & Machine Intelligence (11) (1998) 1254–1259

[105] Zhang, L., Tong, M.H., Marks, T.K., Shan, H., Cottrell, G.W.: Sun: A bayesian framework for saliency using natural statistics. Journal of vision **8**(7) (2008) 32

[106] Harel, J., Koch, C., Perona, P.: Graph-based visual saliency. In: Advances in neural information processing systems. (2006) 545–552

[107] Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on, IEEE (2014) 580–587

[108] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2015) 1–9

[109] Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034 (2013)

[110] Yosinski, J., Clune, J., Bengio, Y., Lipson, H.: How transferable are features in deep neural networks? In: Advances in neural information processing systems. (2014) 3320–3328

[111] Lucas, S.M., Panaretos, A., Sosa, L., Tang, A., Wong, S., Young, R.: Icdar 2003 robust reading competitions. In: ICDAR. Volume 2003., Citeseer (2003) 682

[112] Mosleh, A., Bouguila, N., Hamza, A.B.: Image text detection using a bandlet-based edge detector and stroke width transform. In: BMVC. (2012) 1–12

[113] Saxena, A., Sun, M., Ng, A.Y.: Make3d: Learning 3d scene structure from a single still image. IEEE transactions on pattern analysis and machine intelligence **31**(5) (2009) 824–840

[114] Geiger, A., Lauer, M., Wojek, C., Stiller, C., Urtasun, R.: 3d traffic scene understanding from movable platforms. IEEE transactions on pattern analysis and machine intelligence **36**(5) (2014) 1012–1025

[115] Henry, P., Krainin, M., Herbst, E., Ren, X., Fox, D.: Rgb-d mapping: Using depth cameras for dense 3d modeling of indoor environments. In: the 12th International Symposium on Experimental Robotics (ISER). Volume 20., Citeseer (2010) 22–25

[116] Simonyan, K., Zisserman, A.: Two-stream convolutional networks for action recognition in videos. In: Advances in neural information processing systems. (2014) 568–576

[117] Tran, D., Bourdev, L., Fergus, R., Torresani, L., Paluri, M.: Learning spatiotemporal features with 3d convolutional networks. In: Proceedings of the IEEE International Conference on Computer Vision. (2015) 4489–4497

[118] Venugopalan, S., Rohrbach, M., Donahue, J., Mooney, R., Darrell, T., Saenko, K.: Sequence to sequence-video to text. In: Proceedings of the IEEE International Conference on Computer Vision. (2015) 4534–4542

[119] Anderson, J.M., Nidhi, K., Stanley, K.D., Sorensen, P., Samaras, C., Oluwatola, O.A.: Autonomous vehicle technology: A guide for policymakers. Rand Corporation (2014)

[120] Bishop, R.: A survey of intelligent vehicle applications worldwide. In: Intelligent Vehicles Symposium, 2000. IV 2000. Proceedings of the IEEE, IEEE (2000) 25–30

[121] Franke, U., Gavrila, D., Gorzig, S., Lindner, F., Puetzold, F., Wohler, C.: Autonomous driving goes downtown. IEEE Intelligent Systems and Their Applications **13**(6) (1998) 40–48

[122] Okuda, R., Kajiwara, Y., Terashima, K.: A survey of technical trend of adas and autonomous driving. In: VLSI Technology, Systems and Application (VLSI-TSA), Proceedings of Technical Program-2014 International Symposium on, IEEE (2014) 1–4

[123] Urmson, C., Anhalt, J., Bagnell, D., Baker, C., Bittner, R., Clark, M., Dolan, J., Duggins, D., Galatali, T., Geyer, C., et al.: Autonomous driving in urban environments: Boss and the urban challenge. Journal of Field Robotics **25**(8) (2008) 425–466

[124] Gomez, L.R.P., Fairfield, N., Szybalski, A., Nemec, P., Urmson, C.: Transitioning a mixed-mode vehicle to autonomous mode (December 13 2011) US Patent 8,078,349.

[125] Li, B., Zhang, T., Xia, T.: Vehicle detection from 3d lidar using fully convolutional network. arXiv preprint arXiv:1608.07916 (2016)

[126] Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2015) 3431–3440

[127] Song, S., Xiao, J.: Sliding shapes for 3d object detection in depth images. In: European conference on computer vision, Springer (2014) 634–651

[128] Scovanner, P., Ali, S., Shah, M.: A 3-dimensional sift descriptor and its application to action recognition. In: Proceedings of the 15th ACM international conference on Multimedia, ACM (2007) 357–360

[129] Klaser, A., Marszałek, M., Schmid, C.: A spatio-temporal descriptor based on 3d-gradients. In: BMVC 2008-19th British Machine Vision Conference, British Machine Vision Association (2008) 275–1

[130] Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al.: Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1512.03012 (2015)

[131] Brock, A., Lim, T., Ritchie, J., Weston, N.: Generative and discriminative voxel modeling with convolutional neural networks. arXiv preprint arXiv:1608.04236 (2016)

[132] Su, H., Maji, S., Kalogerakis, E., Learned-Miller, E.: Multi-view convolutional neural networks for 3d shape recognition. In: Proceedings of the IEEE international conference on computer vision. (2015) 945–953

[133] Moravec, H., Elfes, A.: High resolution maps from wide angle sonar. In: Robotics and Automation. Proceedings. 1985 IEEE International Conference on. Volume 2., IEEE (1985) 116–121

[134] Thrun, S.: Learning occupancy grid maps with forward sensor models. Autonomous robots **15**(2) (2003) 111–127

[135] Tipaldi, G.D., Arras, K.O.: Flirt-interest regions for 2d range data. In: Robotics and Automation (ICRA), 2010 IEEE International Conference on, IEEE (2010) 3616–3622

[136] Maturana, D., Scherer, S.: Voxnet: A 3d convolutional neural network for real-time object recognition. In: Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on, IEEE (2015) 922–928

[137] Xie, J., Fang, Y., Zhu, F., Wong, E.: Deepshape: Deep learned shape descriptor for 3d shape matching and retrieval. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2015) 1275–1283

[138] Fang, Y., Xie, J., Dai, G., Wang, M., Zhu, F., Xu, T., Wong, E.: 3d deep shape descriptor. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2015) 2319–2328