APPLIED MACHINE LEARNING WITH LATENT SPACE REPRESENTATION AND MANIPULATION

by

Xunsheng Du

A dissertation submitted to the Electrical and Computer Engineering,
Cullen College of Engineering
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY
in Electrical and Computer Engineering

Chair of Committee: Dr. Zhu Han
Committee Member: Dr. Hien Nguyen
Committee Member: Dr. Jiefu Chen
Committee Member: Dr. Xuqing Wu
Committee Member: Dr. Shuxing Cheng

University of Houston
August 2020

Acknowledgements

I am so grateful for the fulfilling Ph.D. study experience which I have just gone through. For those, who helped or accompanied me during this memorable period, words are not enough to express my appreciation.

Firstly, I would like to thank my supervisor, Dr. Zhu Han, for his professional guidance and considerate support throughout my Ph.D. study. I will never forget the moments he had insightful conversations with me when I had trouble staying on the right track of my research and life. He has always been the role model for me, which I believe will last for the rest of my life.

Secondly, I would like to express my sincere appreciation to Dr. Jiefu Chen, Dr. Xuqing Wu, Dr. Hien Nguyen, and Dr. Shuxing Cheng for being my committee member who provided professional advice and insightful comments to help my research approach perfection.

My thankfulness also goes to my dear friends and colleagues in Wireless Networking, Signal Processing, and Security Lab. Dr. Kevin Tsai, Saeed Ahmadian, Mohamed Elmossalamy, Dr. Reginald Banez, Dr. Neetu Raveendran, Jing Li, Kai-Chu Tsai, Eyad Shtaiwi, Hao Gao, Wen Xie, Dawei Chen, Yuhan Kang, Dr. Hongliang Zhang, Dr. Yunan Gu, Dr. Huaqing Zhang, Dr. Yanru Zhang, Dr. Hung Nguyen, Dr. Fahira Sangare, Dr. Qiuyang Shen, Dr. Jingyi Wang, Dr. Mounika Sai, Xinyue Zhang, Jiaohao Ding, Dian Shi, Debing Wei, Dr. Benedetta Picano, Ye Yu, Zhikun Wu, and many others for accompanying me throughout this journey.

Last but not least, I would like to express my deepest appreciation to my wonderful parents, who always offer me unconditional support and love. I will be in nowhere without you.

Abstract

Machine learning is one of the most promising fields of study nowadays. It is applied to various types of industry including image classification, object detection, and time-series signals prediction, etc. Latent space is a concept that is hidden but significant to machine learning, which helps extract features of data from different dimensions. In this dissertation, we try to apply machine learning with latent space representation and manipulation in real industrial applications both with two studies, respectively.

We first apply machine learning with latent space representation to two works. First, we study the vehicle-to-vehicle relay networks with latent space to represent the decision of resource allocation in the reinforcement learning context. We propose a deep reinforcement learning model to decide the vehicles to be the relay. With the proposed model, the optimal decision is made and the largest overall data allocation is achieved. Then, we conduct a quantitative analysis of the cutting volume in real-time. This analysis is traditionally accomplished by workers on the rig, which cannot guarantee real-time and consistent reports of the cutting volume. With the proposed method, we are able to monitor the cutting volume in a real-time manner while relieving human labor.

We then apply machine learning with latent space manipulation to another two works. First, we monitor the distribution of buffelgrass, a type of invasive grass based on the remote sensing images taken by unnamed aerial vehicles. By applying deep learning along with the discrete latent space-assisted data augmentation, the buffelgrass patterns are accurately located. Second, we solve a seismic inversion problem which is a workflow for deriving the subsurface model from seismic measurements. We propose to utilize autoencoder deep networks with latent space-aligned domain adaptation to migrate the trained model to unexploited data. With the proposed method, we prototype an inversion model with generalization capability quickly in a similar scenario.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Machine learning is one of the most promising fields of study nowadays. It is applied to various types of industry including image classification, object detection, and time-series signals prediction, etc. Latent space is a concept that is hidden but significant to machine learning, which helps extract features of data from different dimensions. In this dissertation, we try to apply machine learning with latent space representation and manipulation in real industrial applications both with two studies, respectively.

1.1 Machine Learning Basics

Machine Learning is a study of computer algorithms that can help the machine learn automatically from prior knowledge without being explicitly programmed. In the past 50 years, there has been an explosion of data. Finding the patterns hidden within this mass of data becomes significant for people to understand the world. Machine Learning techniques are used to automatically find valuable underlying patterns within complex data that are hard to discover with human labor. The hidden knowledge in latent space can be further studied and utilized to predict future events and perform complicated decision making. Machine Learning has already merged into our daily life. Whenever we watch an online video, listen to a song, or do grocery shopping, the background machine learning algorithm works consistently to "guess" your preference and recommends new things appealing to you. The algorithm is even evolving after every interaction. Similar techniques include cancer detection, autonomous car driving, and new drug creation, etc.

From the perspective of the application, Machine Learning-based models can be divided into two categories: discriminative models and generative models. Taking a quiz as an analog, the discriminative models solve a multi-choice problem. Every choice corresponds

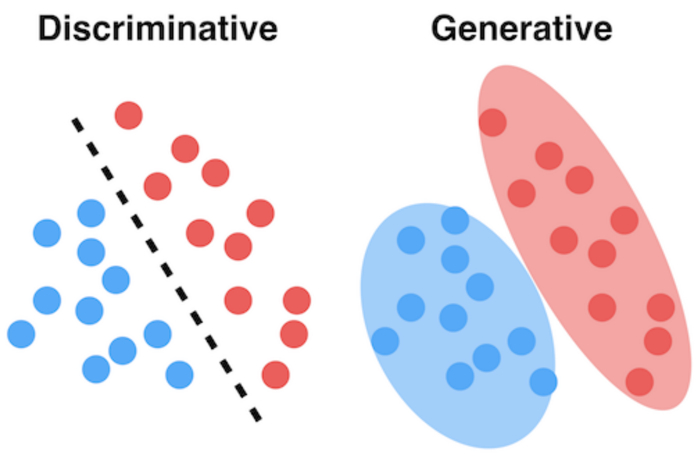


Figure 1.1: Discriminative Model vs Generative Model

to every class in the data. On the other hand, the generative models solve a completion problem. For a certain topic, students have to fill in the blank with their own answers. In the context of math, the discriminative models model the decision boundary between different classes. The training of the discriminative models adjusts the position of the boundary so as to meet a certain objective. Generative models model the actual distribution of each class. The training of the generative models changes the distribution in order to reduce the distance between the generation and the true label.

1.1.1 Discriminative Models

Like learning a language, the discriminative approach is the first step: listening. Discriminative models are used to determine the linguistic differences without speaking it first. The discriminative method directly estimate posterior probabilities $p(y|x)$ for which a functional form is assumed. It has no attempt to model underlying probability distributions but learn from the training data x directly. The goal of the discriminative models is to achieve better performance by focusing on computational resources on the given task.

Popular discriminative models include logistic regression, support vector machine (SVM) [1], traditional neural networks, nearest neighbor, and conditional random fields

(CRF) [2] [3], etc. All the above methods are trying to find a boundary that can separate two classes or multiple classes. Take SVM as an example, SVM finds a hyperplane with maximum distance from the nearest training patterns. The hyperplane here serves as a boundary to separate two groups of data samples. Those samples on the hyperplane are called support vectors. By re-defining the kernel, SVM can also generate a non-linear boundary. The example of SVM shows how discriminative models build boundary and finally discriminate different classes.

1.1.2 Generative Models

Taking the same analog, the generative approach is to learn the standard language as much as it is able to. Generative models can learn multiple languages, but each time it speaks one. The generative models model class-conditional pdfs and prior probabilities. The working process of generative models is as follows. Firstly, we assume some functional form for $p(y)$ and $p(x|y)$. Then, the parameters for $p(y)$ and $p(x|y)$ are estimated directly from training data. Finally, the Bayes rule is used to calculate $p(y|x)$. Generative models are called "Generative" since sampling from $p(y|x)$ can generate synthetic data points.

Popular generative models include Mixtures of Gaussians, Hidden Markov Models (HMM), Bayesian Networks, Markov Random Fields, Autoencoder, and Generative Adversarial Networks (GAN) [4], etc. All generative models try to learn feature representation give a certain label. GAN is a neural networks-based generative model that employs the idea of both discrimination and generation. Take the GAN as an example, we break down the generative process step by step:

1. We create a random distribution and feed it into a generator G to produce the fake x and get fake $(x, y = 0)$ input-label pair.

2. We feed the fake pair $(x, y = 0)$ and the real pair $(x, y = 1)$ into the discriminator D, alternatively.

3. The discriminator D implements binary classification so it calculates the loss for both fake x and real x.

4. The generator G also calculate the loss from the noise distribution.

5. Both losses are learned under two conditions. G generates fake x that is so close to real x that D cannot tell the difference. D gains stronger discriminative capability that fake x will be easily identified if not improved.

From the example of GAN, we know how generative models generate data samples from the perspective of math.

1.2 Latent Space Basics

Latent space is a fundamental, yet “hidden” concept of machine learning, especially for deep learning. The emergence of this concept comes from the manner people discover patterns inside the data. Generally, human beings are able to find the rules of a group of 2D or 3D data. When we plot those data on a 2D space or 3D space, the distribution of the data is easy to observe. However, in this era of data explosion, not only the volume of the data grows but so does the dimension. We have no explicit conception of the dimension higher than 3, but we can project or convert a high-dimension into 3. In this example, 3D space is the latent space of the original data space, which helps us understand the distribution of the raw data. The more detailed concept of latent space, latent representation, and latent space manipulation are introduced in the following subsections.

1.2.1 The concept of latent space

In short, latent space is the representation of compressed data. We can take hand-written digits classification as an example. The digits with the same number have more similarities than those with different numbers. The learning model tries to recognize the similarity in order to classify different digits into different classes. By further investigation, we find the model learns the structural difference between digits precisely. The structural difference in this task (and in similar image classification tasks) is the latent feature. Space where latent features lie in is named latent space. For deep learning, the utilization of latent space is crucial since it leads to learning the features of data and simplifying data representations for the purpose of finding patterns. The reason for compressing data is obtaining the most important features and getting rid of extraneous information. For instance, autoencoder [5] uses an encoder to compress data which can be restored by a decoder. The compressed data contains the most important information for reconstruction.

1.2.2 Latent Space Representation

Latent space representation contains all important information needed to represent the original data point. To represent original data, the features are represented. The learning model learns the data features and simplifies its representation to make it easier to analyze. From the latent space representation idea, the concept of representation learning is defined. Representation learning is a set of techniques that allow the system to discover the representations needed for feature extraction or classification from raw data. By utilizing representation learning, a complex form of raw data can be transformed into a simpler representation that is easier to process. The hidden layers for the deep networks aiming at image classification/recognition can be regarded as representation learning since it transforms the original data into another space for processing. Similarly, for deep reinforcement learning (DRL) [6] [7], the deep networks model can also be categorized into representation learning since the hidden layers contain the information needed for transferring the information of the environment state into actions.

1.2.3 Latent Space Manipulation

Besides data compression, the raw input can also be utilized, transformed, or incorporated into another component of the learning model, which is named latent space manipulation. An innovation based on vanilla autoencoder is called variational autoencoder [8]. Unlike the traditional autoencoder that reconstructs the input, variational autoencoder is a generative model that creates synthetic patterns that are in the same space as the training input. The variance comes from the sampling of the embedding vector based on a certain distribution (e.g., Gaussian distribution). The code in the latent space works as a loss measurement for sampling an embedding vector which is close to the latent representation of the input. Moreover, the code in the latent space can help generate a certain pattern, which is directive. Assume two variational autoencoders are trained on two similar facial image sets. One with the faces wearing glasses, while another without. In order to wear glasses for images in the latter set, an operation named arithmetic calculation can be implemented to append the pattern discrepancy onto a synthetic image. Apart from autoencoder, there is an increasing number of studies focusing on latent space manipulation.

1.3 Dissertation Contributions and Organization

In the first work, we represent the latent space and map it into the action space in the DRL scenario. In this work, we solve a resource allocation problem by combining Deep Learning (DL) [9] and Reinforcement Learning (RL) [10]. The contributions of this work can be summarized as follows:

- We propose a vehicular network topology with the highlighting features of LTE-V, such as high bit rate, long transmission range, and high capacity. This topology contains the two topologies: traffic topology and wireless communication topology.
- We extract features from the network topology to get a latent vector and map this

vector into the action space for decision making.

- Q-Learning as one branch of RL is utilized for measuring the value of a certain action in order to perform the optimal resource allocation.

We further investigate the advantage of latent space representation by constructing a space without extraneous information with proper pre-processing before feature extraction. In the second work, we implement multiple steps of pre-processing with video decoding/encoding in a real-time manner for accurately classifying the cutting volume in the borehole cleaning scenario. We summarize the main contributions of this work as follows:

- A multi-thread video encoder/decoder is implemented to process the video stream in real-time acquired by an uncelebrated camera.
- An object detection learning model is built to automatically detect the region of interests (ROI, i.e., the region covered by cuttings).
- A few pre-processing techniques are implemented for better latent space representation, which includes subsampling, whitening transformation, and instance normalization.
- A deep learning model adapted from the VGG networks is constructed for the classification work.

The latent space is not only able to be represented for the classification, but it can also be manipulated for better performance in the downstream task. In the third work, we convert the continuous latent space representation into a discrete one in the context of data augmentation for a better detection performance in pinpointing the invasive grass.

- For detecting the distribution of buffelgrass on the far-flung pasture, we adopt the deep learning-based object detection technology to pinpoint the buffelgrass.

- In order to increase the generalization capability of the detection model, we increase the data volume by proposed data augmentation with another generative model.
- Inside the generative model, we convert the continuous latent space representation to discrete representation in order to solve the posterior collapse problem and generate high-fidelity synthetic grass pattern.
- To further improve the training of the detection model, we adopt the transfer learning technique by learning from the checkpoint of the pre-trained model on other image data sets.

Latent space manipulation can not only help improve the current task but can also be utilized for domain adaptation. In the fourth work, we incorporate the latent vectors into the final loss for learning the adaptation gap in order to adapt the learned model to an unexploited but similar data set in the seismic inversion scenario. The contributions of this work are summarized as follows:

- A deep network following an autoencoder structure is constructed for seismic data inversion which is under the category of data-driven inversion method.
- We introduce optimal transport as a tool for the domain adaptation in the context of measuring the geometric distance between the distributions in the source and target domain.
- To avoid two-steps adaptation, the technique of joint distribution optimal transport is utilized for jointly considering the feature space and the label space.
- Specifically for the deep learning problem, a deepJDOT [11] method is formulated for learning the adaptation gap in mini-batch.

The rest of this dissertation is organized as follows. In Chapter 2, we solve the resource allocation problem in V2V networks by DRL with latent space representation. In

Chapter 3, we implement real-time cutting volume classification by proper pre-processing for boosting the representation of the latent space. To manipulate the latent space for better performance, we convert continuous latent space to the discrete representation for data augmentation in order to detect the invasive grass more accurately, which is elaborated in Chapter 4. In Chapter 5, we incorporate the latent space to align the adaptation gap to solve an unexploited seismic data inversion problem. Finally, in Chapter 6, we conclude our dissertation and propose some potential future works.

Chapter 2

Virtual Relay Selection in LTE-V: A Deep Reinforcement Learning Approach to Heterogeneous Data

Nowadays, vehicular network communication is a compelling technology to provide wireless connectivity among vehicles, roadsides' drivers, passengers, and pedestrians [12]. There exists a potentially promising market for vehicle-to-everything (V2X) [13], yet it has not been put into large-scale inference [14]. A lot of applications are on their way to be implemented in vehicle-to-vehicle (V2V) communication scenarios, such as road safety, traffic efficiency, and infotainment types with different performance requirements [12]. Most of those applications require low delay, high reliability, and high quality of service (QoS). In order to meet the above service requirements, several wireless access technologies have been conceived to provide radio interface including cellular systems, infrared communications, traditional Wi-Fi, and IEEE 802.11p [15].

Vehicular ad hoc network (VANET) is a part of the novel approach for intelligent transportation system (ITS) with the aim to enhance driver's safety, regulating traffic and improving the whole driving experience [16]. In real-time traffic scenarios, VANET enables inter-vehicle, vehicle-to-roadside, as well as inter-roadside communication. However, IEEE 802.11p suffers from the unbounded delay problem, scalability issue, and lack of high QoS guarantees [17]. Moreover, the transmission range is limited when it comes to huge urban traffic scenarios. Without pervasive roadside equipment, it is difficult for IEEE 802.11p to provide consecutive and long-lived vehicle-to-infrastructure (V2I) connectivity. The disadvantages of IEEE 802.11p demonstrated above activate the emerging development of LTE-V which is a potential solution supporting vehicular communications [18].

LTE has a wide coverage area, high penetration rate and also supports service of high speed. Moreover, LTE enables high data rates and low latency, which can be beneficial to vehicular safety. As a matter of fact, the safety problem on the road is more and more severe nowadays. According to the National Highway Traffic Safety Administration (NHTSA), there were 47,420 fatal crashed in 2016 [19]. These exiting features of LTE-V give birth to a lot of applications to resolve the safety problem on the road. For example, cooperative awareness message (CAM), which includes basic vehicle data such as speed, position and accelerate speed, can be exchanged among vehicles. There are some user cases for CAM including emergency vehicle warning, slow vehicle indication, intersection collision warning and so on. The other type of messages is called decentralized environmental notification message (DENM), which includes information of emergency electronic brake light and road-work zone warning, wrong-way driving warning, stationary vehicle accident, traffic condition and signal violation warning, etc. Besides those emergency-related messages, LTE-V also delivers delay-tolerant information, such as news, entertaining shows, weather forecast, etc. Apparently, there will be more and more demands for new applications emerging with the development of LTE-V technology. In the literature, there are many works focusing on solving problems in the applications presented above. From the crash warning point of view, in [20], an algorithm for the pre-crash control system was proposed to provide all-round prewarning of a potential accident. For the use case for emergency vehicles, comprehensive design of emergency vehicle warning system was proposed [21], in which vehicles not only receive warning of approaching emergency vehicle, but also are warned of detailed route information. Adaptive cruise control (ACC) is another interesting topic. [22] presented design, development, implementation, and testing of a cooperative adaptive cruise control (CACC) system, which consists of two controllers for managing approaching maneuver to leading vehicles and regulating car-following, respectively. For CACC, another work [23] proposed a RL approach for developing controllers for the secure longitudinal following of a front vehicle.

Those aforementioned works considered the common use cases in the V2X scenario and solve control problems in a relatively small scale. However, due to the development of traffic monitoring technique and internet-of-thing (IoT), more data about the traffic are available. For analyzing such a big amount of data, DL [9] is one of the promising options. DL enables the analysis of global information to obtain the overall picture. Nowadays, the development of DL makes the technique of pattern recognition more powerful. It's natural to use the power of the DL to analyze traffic and wireless communication patterns or topologies in order to extract features inside traffic data. Although, DL provides insights in data, it cannot offer decision making strategy in control-related tasks. RL [10], on the other hand, helps agent interact with environment and make decision based on the supervision of the feedback (measured as rewards) with only partial information known. DRL that combines DL and RL can build a decision making framework, which interacts with ever-changing and random environment. From the aspect of urban wireless data transmission control, a big picture is rather important since the coverage of base station will not be limited to just a few street blocks. Also, sometimes only partial information related to decision making is available, which is impossible for pure optimization. In our work, we consider a relay scenario based on V2V communication, in which a huge urban area is taken into consideration.

The main contributions of our work can be summarized as follows:

- Based on the highlighted features of LTE-V, such as high bit rate, long range, high capacity and ubiquitous coverage, a vehicular network topology is proposed, which considers both traffic topology and wireless communication topology in the V2X scenario.

- A bridge is built between traffic topology and delay-tolerant data allocation in V2X communications. We design a scheme which takes traffic topology and wireless communication topology as inputs and produces a data allocation strategy. This scheme use DL as a function approximator to formulate mapping between inputs and outputs.

We use Python Streets4MPI toolbox [24] to simulate and generate traffic heat maps, and use Tensorflow as platform to implement deep Q-learning framework.

- There is no explicit relationship between traffic topology and data allocation in wireless communications. Thus, a novel double deep Q-learning approach [25] is utilized to tackle the formulated problem. The input to our deep Q-learning is heterogeneous data containing traffic and wireless communication information. In order to achieve a better training result, actor and critic networks as well as experience replay are utilized to enhance the performance of traditional DRL. We use cumulative rewards to measure the performance of the deep Q-learning framework and compare it with the performance of other baselines. We also visualize the behavior of the deep Q-learning agent to obtain insight of its intelligence.

The rest of this chapter is organized as follows. Section II will introduce more related works concerning the control and optimization in the vehicle relay scenario. Then, our data relay model based on LTE-V is proposed in Section III, and the problem is formulated in Section IV. Based on the formulated problem, a novel DRL approach is introduced to tackle the problem in Section V. In Section VI, we will show the effectiveness of our algorithm by simulation results. Finally, some concluding remarks are given in Section VII.

2.1 Related Work

There exists some works focusing on the vehicular relay in V2X networks (either vehicle-to-vehicle or vehicle-to-infrastructure) [26] [27] [28] [29] [30] [31] [32] [33]. In [27], a vehicular relaying technique for enhanced connectivity in densely populated urban areas was designed to investigate the performance of a transmission scheme over a Long-Term Evolution-Advanced (LTE-A) network where vehicles act as relaying cooperating terminals session between a base station and an end-user. Also in urban areas, [26] investigated the impact of the greedy and selfish individual nodes on the cooperation dynamics in vehicular ad-hoc networks. A decentralized self-organized relay selection algorithm based on game theory was proposed in vehicular ad-hoc networks [26]. In cooperative networks scenario, the cooperative secure transmissions in multiple-input signal-out (MISO) vehicular relay networks is studied [29]. Seyfi *et al.* [30] investigates cooperative diversity with relay selection over cascaded Rayleigh fading channels. In [30], the authors conducted a specific analysis on the performance of a relay selection scheme for cooperative vehicular networks with the decode-and-forward (DF) protocol.

From the data forwarding or packet delivering point of view, the assistance of relay for packet delivering dramatically enlarges the coverage area and saves a huge amount of power and energy in vehicular networks. Song *et al.* [28] proposed an analytical approach based on stochastic geometry to analyze the location-aware opportunistic V2V relay scheme in terms of the transmission success probability for a target destination vehicle and the connectivity probability when the scheme is applied to inter-connect adjacent RSUs. In [34], the optimal relay station (RS) selection strategy for the vehicular subscriber station (SS) was studied. By using a highway mobility model in IEEE 802.16j MR network, Ge *et al.* formulated a nonlinear optimization problem and figured out the optimal locations of RSs. However, LTE enables a much larger coverage area, which enables a source station to communicate directly with some vehicular relays. In [35], performance evaluation of relay vehicles was addressed. Chai *et al.* formulated an optimal matching problem in a bipartite graph and solved it using the Kuhn-Munkres (K-M) algorithm. Similarly, [36] proposed a game theory approach to tackle the relay vehicle selection problem by jointly considering all the relay vehicles and source vehicles.

As we can tell from the review of previous works on vehicle relay in vehicular networks, most of the works are formulated as a joint optimization problem, for which the non-linear programming or game theory related algorithm can be implemented. However, in most cases in real traffic scenario, only partial information might be known. Meanwhile, data we obtain may only contain implicit information, which is waiting for excavation. Deep neural networks [9] is now a popular and powerful tool to extract features inside data. By traffic monitoring, 2D or even higher dimension images can be generated. Using convolutional neural networks (CNNs) [37] is one of the best choices to analyze the spatial relationship between different data samples. Since the traffic situation changes from time to time, time-correlative data also can be obtained. In this case, recurrent neural networks (RNNs) can show its prowess. In [38], a special type of RNNs, long short-term memory (LSTM) neural networks are utilized to predict long term traffic.

Although DL is powerful in pattern recognition and prediction, it cannot directly help decision making based on those patterns it observes. That is why RL [10] comes onto the stage. RL is a type of machine learning that creates agents which are capable of taking actions in an environment in order to maximize overall rewards. From the aspect of the learning method, DL can be classified into supervised, unsupervised, or semi-supervised learning. On the other hand, RL presents a form of supervision through reward without explicitly tell the agent how to perform the task. DRL [6] [7], which combines DL and RL, is now widely considered a technique that is close to artificial intelligence most. In the traffic scenario, there exist some works utilizing DRL. In [39], a safety-based control of vehicle driving is implemented by using DRL. The continuous control is achieved to let vehicles implement self-driving. In [40], a traffic light control system is achieved by using DRL in traffic simulator SUMO [41]. However, different from the solving control problems in the pure traffic scenario, our work tries to combine wireless communication with traffic in a DRL approach.

2.2 System Model

In this section, we introduce the system model in two subsections. In the first subsection, we introduce traffic and wireless communication topology. The representation of the

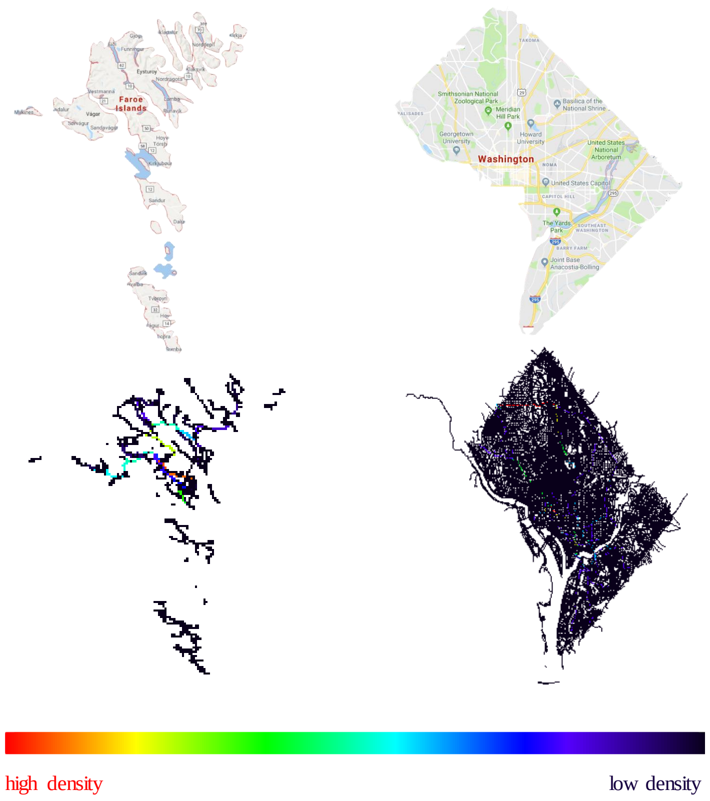


Figure 2.1: Sample traffic heat maps generated by the transportation traffic simulations.

traffic situation and the virtual vehicle relay scheme is introduced. The combination of the traffic and wireless communication topology information is fed into the deep Q-learning framework, which makes the input data heterogeneous. In the second subsection, we go to the details of the communication model we use in our virtual vehicle relay scenario.

2.2.1 Traffic and V2V Communication Topology

We consider both traffic topology and V2V communication topology. We analyze the traffic topology by using the traffic density map, which also known as the traffic heat map. In our model, we simulate a series of traffic heat maps of a certain district in a city. The transportation traffic simulator we used in our work is named Streets4MPI [24], a python based software that can simulate continuous transportation traffic heat maps. The workflow of the transportation traffic simulation can be briefly summarized as follows:

- A street network of a real city in the world is imported into the canvas. The file representing the structure of the street network is an XML file named OSM XML [42].
- A number of drivers are initialized (this number can be customized). For each driver, there is a corresponding pre-defined [origin, goal] pair, which indicates the location of the departure and arrival.
- Each driver will drive his/her car along the shortest path between the departure and arrival. While driving, each vehicle has a speed range. Between two cars, there is a minimum distance to avoid the collision.
- All calculated shortest paths are then traversed and the traffic load is recorded for each street.
- Those recorded traffic load can be visualized and represented as pixel values on the traffic heat maps.
- The simulation can be conducted circularly. The traffic heat maps vary from time to time. If one driver has finished its trip. It will start again from the departure. Thus, the total number of vehicles on the road won't change after it is set up before the simulation begins.

The simulation generates one heat map at a time. We can set the simulation interval so that a series of maps will be generated in sequence with a fixed time interval, which demonstrates the variation of the traffic flow over time. Those sample maps show the traffic in a small city. However, in the communication simulation, we will crop the map and only use a small portion.

We assume that the cache data required by the vehicular end users come from the roadside units, such as eNB, located at the suburb region of the city. The distance between

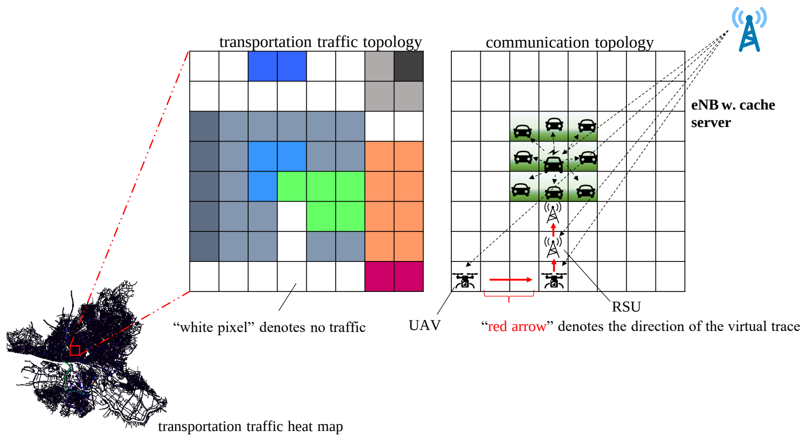

transportation traffic topology
communication topology
eNB w. cache server
"white pixel" denotes no traffic
UAV
RSU
"red arrow" denotes the direction of the virtual trace
transportation traffic heat map

Figure 2.2: Transportation topology and communication topology.

the eNB and the center of our selected urban area is around 1.5 to 2 kilometers. A cache data relay scheme is considered in our model. The eNB first distributes the cache data to a vehicle on road, which is a mobile relay node. Then this vehicle, also known as the helper, disseminates the cache data to the surrounding vehicles within its capability. This kind of transmission schemes with vehicle relay has the advantage of saving unnecessary power consumption and solving traffic scalability issue, while the scheme of distributing cache data to each vehicle individually using cellular broadband access [43] is more energy-consuming. This vehicle relay scheme will be widely made used of in the future LTE-V network. The selection of the vehicle relay node relies on the topology of traffic, which is changing all the time. Since we cannot control the trace of the vehicles, we come up with a virtual relay node selection scheme. This scheme is under the assumption that every vehicle on road can be a potential helper by installing both transmitter and receiver on board, which is also going to be achieved in the near future. Under this assumption, every vehicle on the road is ready to act as a helper. If it happens to pass by the spot where the virtual relay node is placed, the real connection between it and the eNB at the suburb can be built up, and the transmission begins. When this vehicle drives off the virtual relay node, the transmission will be handed over to the next vehicle that drives in this node. Due to the handover problem, we gradually move the virtual relay node from one spot to an adjacent one instead of jumping from one spot to another faraway spot. In this paper, we have no further consideration on the impact of handover on the quality of transmission. After receiving the data from eNB, the helper will disseminate the data to other vehicular users within its opportunistic range [44], which is a term that describes a contact opportunity between the helper and the end user. The virtual relay node will be gradually moved to the spot where the overall system transmission throughput can be achieved.

Along the movement of the virtual relay node, we can draw a trace, named virtual relay trace. This trace is not explicitly related to the transportation traffic density since we cannot control the movement of vehicles. However, as transportation traffic density is one of the important factors which impact the transmission throughput (i.e., how many potential helpers and end users around), it will implicitly influence and lead the virtual relay node to a better spot, and for better transmission performance. It is not necessary for the virtual relay trace to follow the trace of roads. If the virtual relay node is moved off the roads, where there are no vehicles, we can hand over the transmission to roadside units (RSUs) or unmanned aerial vehicles (UAVs), and let them act as the helpers. If the virtual relay trace traverses a consecutive off-road area, the trace of virtual relay node works as an intended path for the UAVs.

2.2.2 Communication Model

For LTE-V, there are two types of messages that are sent among vehicles, which require low latency and high reliability [45]. CAM contains some basic vehicle-related data such as speed, position and so on. On the other hand, DENM only works for some emergency situation, e.g., emergency electronic brake light and road-work zone warning.

In our model, we consider all vehicles in traffic can work as a message relay, which is capable of disseminating received data from eNB to nearby vehicles. We consider a contact model that vehicles can communicate with each other only when they move to within the transmission range [44], which is also called communication contact. The Poisson distributed contact rate has been observed in the real vehicular trace and has been implemented in a lot of works, such as [46] [47], which enables analysis for resource allocation and control problems.

Here, we consider the problem in another way that the traffic density and transmission capacity of helpers are two key factors affecting the communication range. Since our observation of the system is a static traffic map of a large region, the demonstration of traffic status is in a more macro way compared to a dynamic vehicle-oriented system using some traffic simulation software such as SUMO [41]. Since the density of every corner in the map is known in an omnipotent view, we can denote the opportunistic contact as $O_{v_e,u} = \alpha \cdot \frac{K_{v_e}}{D_{x_{v_e},y_{v_e}}}$, where K_{v_e} is the transmission capacity of helper v_e, and $D_{x_{v_e},y_{v_e}}$ denotes the traffic density at coordinate of the helper in map. α is a constant parameter satisfying $\alpha > 0$. Due to the limitation of the capacity of transmitter, we intuitively define the communication range $O_{v_e,u}$ in a way that $O_{v_e,u} \propto K_{v_e}$ and $O_{v_e,u} \propto \frac{1}{D_{x_{v_e},y_{v_e}}}$. This definition can be comprehended in this way: Under a certain power constraint, the more vehicles around the helper, the communication range will be shorter, since the helper will first serve the vehicle end users which are closer to him. The distance between helper v_e and subscriber u which connected to the helper is denoted as $d_{v_e,u}$. Then we have the constraint as $d_{v_e,u} \leq O_{v_e,u}$.

We consider eNBs e located in the suburb area, transmitting in circular coverage of star topology with radius r_e. The vehicle which receives the data transmission (also known as the helper) from the eNB is denoted as v_e. The distance between e and v_e is denoted as d_{e,v_e}. Since our work considers the location of the eNB and the helper in a pixel level, we assign the location of the eNB and the helper as coordinate (x_e, y_e) and (x_{v_e}, y_{v_e}), respectively. We assume the transmission between the eNB and the helper has line of sight without any block from building, trees or flying objects. And we also ignore the height difference between the eNB and the helper, which can be extended. Then we can denote the distance between the eNB and helper as $d_{e,v_e} = \sqrt{(x_e - x_{v_e})^2 + (y_e - y_{v_e})^2}$. Since the eNB can only cover the helpers within its communication range, we have the constraint $d_{e,v_e} \le r_e$.

According to the Shannon theorem and in the quasi-static scenario, the achievable rate of helper v_e when associating with transmitter e can be expressed as

$$r_{v_e}^{e} = B_{v_e} \log_2 \left(1 + \frac{p_{e,v_e} |h_{v_e}^{e}|^2}{N_o B_{v_e}} \right), \tag{2.1}$$

where B_{v_e} denotes the bandwidth allocated to helper v_e, which we set to 10 MHz. p_{e,v_e} is the transmitted power from eNB e to the receiver on v_e, which is set to 30 dBm. $h_{v_e}^{e}$ is the channel gain between helper v_e and transmitter eNB e, which is Rayleigh distributed with the mean 1. N_o is the noise spectral density. The relationship between the transmitted power p_{e,v_e} and received power at vehicular relay p_{e,v_e}^{r} is denoted as $\frac{p_{e,v_e}^{r}}{p_{e,v_e}} = \left[\frac{\sqrt{G_l}\lambda}{4\pi d_{e,v_e}} \right]^2$. $\sqrt{G_l}$ is the product of the transmit and receive antenna field radiation patterns in the line-of-sight direction, which is set to 15 dB. λ is the wavelength of transmitted signal. For the transmission rate between the helper and the subscriber $r_u^{v_e}$, we don't consider the path loss effect due to two reasons. First, the position of the virtual relay node changes gradually. Thus, the distance $d_{v_e,u}$ changes little from time to time. Second, the distance $d_{v_e,u}$ is within several meters. For different subscribers u, the path loss effect makes little difference. In the experiment, we set the value of signal-to-noise ratio (SNR) from the relay v_e to the subscribers u as 40 dB.

The data transmitted from eNB e to helper v_e is denoted as m_{e,v_e}, which is counted in bit. Then the time consumed during the transmission between e and v_e can be denoted as $l_{e,v_e} = \frac{m_{e,v_e}}{r_{v_e}^{e}}$. Similar to the latency between helper v_e and subscriber u, l_{e,v_e} also can be seen as the latency of the transmission between e and v_e, which is a factor affecting the quality of service.

2.3 Problem Formulation

Nowadays, the technology of monitoring real-time traffic is really mature. The traffic can be monitored based on moving vehicles under different weather and illumination conditions [48]. In the industry application, a lot of software provide us with convenient monitoring of real-time traffic every day. For example, when we drive on the highway, we can check Google Map on our phone to get the latest traffic status miles ahead of our scheduled route. Such information not only gives us the expectation of driving experience beforehand but also offer the potential decision for drivers to alter their route if the current one is not satisfying. The example above is a simple decision-making problem which we may encounter in daily life.

The example above gives us the inspiration to build a connection between the observation of traffic flow and control action in wireless communication. In our problem, there are several eNBs continuously transmitting data to a certain location. The location is represented by a certain pixel on our traffic topology (referring to Fig. 2.2). The value of the pixel stands for the traffic density at that location. This value can be zero which means no vehicle appears in that spot at that specific timestamp. If there is no vehicle on that spot, the relay task cannot be accomplished, and no rewards received. Under the assumption of volunteerism, if there exists at least one vehicle at that spot, any vehicle there can be chosen as the helper (relay node) by receiving the signal from the eNB and disseminating it to nearby vehicles. The goal of our work is to select the vehicle relay node in order to achieve the best overall system throughput without controlling the movement of vehicles on road. The core idea is that we generate a trace of the virtual relay on which potential real vehicular relay can be selected. This trace crosses through the whole map and might not follow the direction of streets.

At a certain timestamp t, we regard the current traffic heat map is one of our observations, which is denoted by the matrix $S_{traffic}$ with n rows and n columns, and each entry represents the RGB value. Thus, $S_{traffic}$ has the shape $(n, n, 3)$. After converted to greyscale, $S_{traffic}$ has a shape of $(n, n, 1)$. The other observation at timestamp t is the topology of the wireless communication, which is denoted by the matrix S_{comm} with n rows and n columns. For those spots containing the vehicular relay and connected end users, we assign the corresponding entries with 1, and otherwise with 0. S_{comm} has the shape of $(n, n, 1)$. This design helps us highlight the current location of the vehicle relay and the end-user vehicles which connect to it. We denote the observation at timestamp t as s_t, where $s_t = [S_{traffic}, S_{comm}]$. The state s_t has the shape of $(n, n, 2)$. With the observations (or states), we assume that an agent is employed by us to accomplish the virtual relay node selection task. In order to train the agent, we assume that all information regarding the wireless communication between eNBs and connected helpers, as well as helpers and connected subscribers (e.g., distance, transmission rate, the power consumed, the volume of delivered data, etc.) are known to an operator. The operator acts as a supervisor to give the feedback to the agent after any move it accomplishes. This supervision exists only during the training phase. After the agent is well-trained, we can let the agent make decisions without the instruction from the operator.

The decision of the agent relies on the observation we describe above. By feeding the observation and location of the virtual relay at last timestamp, the agent can decide an action a_t to take at time t. The action space is denoted as $\{0, 1, 2, ..., 8\}$, where 0 represents "staying put", and other numbers represent 8 possible directions in which the spot of vehicle relay can move. The traffic heat maps are fed continuously and current packets allocation topology is only related to the last action (Current action will result in another packets allocation topology at next timestamp). We can easily find out that s_t follows a first-order Markov chain as

$$P(s_t | s_1, a_1, s_2, a_2, ..., s_{t-1}, a_{t-1}) = P(s_t | s_{t-1}, a_{t-1}). \tag{2.2}$$

Under state s_t, the execution of action a_t will invite an instant reward denoted as r_t, which can be expressed as

$$
\begin{aligned}
r_t =& \beta_{r(e,v_e)} \cdot r_{v_e}^{e}(t) + \beta_{r(v_e,u)} \cdot \sum_u r_u^{v_e}(t) - \beta_{l(e,v_e)} \cdot l_{e,v_e}(t) \\
&- \beta_{l(v_e,u)} \cdot l_{v_e,u}(t) - \beta_{f(v_e)} \cdot f_{v_e} - \beta_{P(e)} \cdot P_e,
\end{aligned} \tag{2.3}
$$

where we define the instant reward at time t by taking the transmission rate $r_{v_e}^{e}(t)$ and $r_u^{v_e}(t)$ as revenue. The latency $l_{e,v_e}(t)$ and $l_{v_e,u}(t)$ during transmission, the fee f_{v_e} charged by the helper for providing service, and the energy P_e consumed by eNB e are counted as the cost. The parameter β represents the weight assigned to each component contributing to the reward. The larger the value of β, the more contribution it will make to the overall reward. We set a constraint on β that $\beta_{r(e,v_e)} + \beta_{r(v_e,u)} - \beta_{l(e,v_e)} - \beta_{l(e,v_e)} - \beta_{f(v_e)} - \beta_{P(e)} = 1$, where $\beta_{r(e,v_e)}, \beta_{r(v_e,u)} \geq 0$ and $\beta_{l(e,v_e)}, \beta_{l(v_e,u)}, \beta_{f(v_e)}, \beta_{P(e)} \leq 0$. Although, the reward we defined above includes heterogenous components, it makes sense if we treat the revenue or cost that every unit of component evokes the same. This reward can be regarded as the income of the service provider. The weighted components in the equation consider all factors that probably affect the reward, and at the same time enable the adaptation to different forms of rewards according to different demands. For example, if we are only concerned about the transmission rate, we can set $\beta_{l(e,v_e)}, \beta_{l(v_e,u)}, \beta_{f(v_e)}, \beta_{P(e)} = 0$. If there is no traffic at the spot where we place the virtual relay node, it will incur a zero reward, since $r_{v_e}^{e}(t)$ and $r_u^{v_e}(t)$ will be zero in (2.3). We won't choose a specific vehicle as the helper, and control its trajectory, since it would be much more expensive.

To summarize, the definition of the state, action and instant reward at timestamp t can be denoted as follows:

- State: $s_t = [S_{traffic}, S_{comm}]$, where $S_{traffic}$ and S_{comm} represent the matrices for traffic topology and wireless communication topology respectively.
- Action: $a_t = (a_{t,0}, a_{t,1}, ..., a_{t,8})$, where $a_{t,0}$ represents that the virtual relay node stays

put. Other actions represent the virtual relay node to be moved towards 8 directions. Every movement crosses one pixel in transportation traffic map.

- Reward: $r_t = \beta_{r(e,v_e)} \cdot r_{v_e}^{e}(t) + \beta_{r(v_e,u)} \cdot \sum\limits_{u} r_u^{v_e}(t) - \beta_{l(e,v_e)} \cdot l_{e,v_e}(t) - \beta_{l(v_e,u)} \cdot l_{v_e,u}(t) - \beta_{f(v_e)} \cdot f_{v_e} - \beta_{P(e)} \cdot P_e$. The instant reward r_t here is a sum of weighted value of the transmission rate, latency, service fee, and power consumption.

Since the action in our model is to select a spot in the map as a relay node and virtually move the node (gradually move the spot to select another vehicle at another spot), the current action taken under the current state will impact on the future reward. This inspires us to consider both the current reward and future reward. Thus, we define the long-term system reward as

$$w_t = \sum_{\tau=t}^{\infty} \chi^{\tau-t} \cdot r_\tau, \tag{2.4}$$

where $\chi \in (0,1)$ is a discounting rate, which yields a bounded objective for optimization. It reflects the fact that a current action has a weaker impact on the future reward compared with the current reward.

Notice that both current and future reward should be taken into consideration in our problem. The goal of our work is converted to designing a policy π (a mapping from state to action), denoted as $\pi: \ s_t \rightarrow a_t = (a_{t,0}, a_{t,1}, ..., a_{t,8})$, to maximize the long-term reward,

$$V^\pi(s_t) = r_t + \chi \cdot V^\pi(s_{t+1}) = r_t + w_{t+1}, \tag{2.5}$$

where $V^\pi(s_t)$ is the long-term reward using policy π under the state s_t.

Then, the long-term system reward maximization problem can be formulated as,

$$\max_\pi V^\pi(s_t) \quad \forall s_t, \tag{2.6}$$

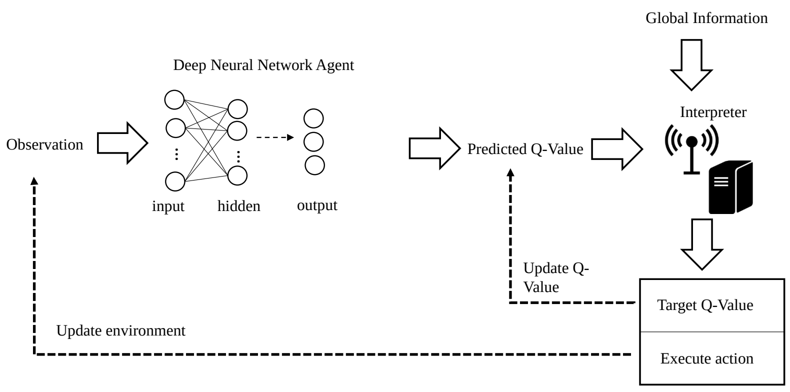


Figure 2.3: The solution pipeline for the proposed virtual relay selection problem

with the recursion relation:

$$V^{\pi}(s_t) \leftarrow r_t + \chi \cdot V^{\pi}(s_{t+1}). \tag{2.7}$$

By taking the observation s_t as input at time t, the agent will output a policy π that indicates which action a_t should be taken. The operator will execute action a_t. After the action is taken, value $V^{\pi}(s_t)$ for action a_t under current state a_t is evaluated by the operator according to (2.3). The reward will be fed back to the agent, in order to let it "remember" the experience and react better next time when facing the same state. The traffic and wireless communication topology change from previous time t to current time $t+1$ after the action executed. Then the agent should take a new observation s_{t+1} and output a new policy, and so on so forth. The final goal is to achieve the objective in (2.6).

There exist several challenges for this task listed as follows:

- Since the goal is to optimize the overall reward by the step-by-step control, the observation is discrete and consecutive, i.e., the agent only knows the state of the environment at timestamp t. The action taken at timestamp t will affect the state of the environment at time $t+1$. The state of the environment cannot be obtained beforehand. Therefore, a joint optimization approach considering current and future states is not feasible.

- Only after the operator executes the action suggested by the agent that the reward of this action is known. As we explained, the operator knows all parameters about the connection between the eNBs and connected helpers, e.g., power consumption, transmission rate, latency, and channel quality indicator. A new action taken will change the connected helpers and also the value of those parameters contributing to the reward. In other words, since location information of eNB is invisible to our agent, the agent cannot know which action has the maximal reward by simply trying every action via the brute-force approach. Once the action has been taken, the operator who is responsible for collecting every information of transmitters will know all information including the volume of transmitted data, latency, the power consumption of the eNB, etc. By using the information from the operator, the expected reward can be calculated. In short, a method of dynamic trial-and-error should be adopted.

- In our work, the virtual relay node moves gradually. Firstly, the transmission between the eNB and vehicle has a duration which continues for a few minutes or more. Secondly, the handover problem exists that it consumes some time for one vehicle switching link to another eNB or helper. Thus, in our model, the virtual relay spot should move gradually, instead of jumping from one spot to another far-off spot. In our model, the virtual relay node is controlled like a robot moving around the city. Under this assumption, the action will have a long-term effect on the reward in the future. Thus, the impact of current action on the future rewards should be considered.

To overcome the obstacles we mentioned above, we are going to propose a novel DRL method in Section 2.4 to maximize the long-term reward with implicit and partial information is provided.

2.4 Deep Q-Learning

RL is known as a sequential decision-making technique, which solves control problems in many fields. However, traditional RL is rather unstable during training. In addition, the observation of the environment is complex for some traditional agents, such as shallow-layer neural networks. With the development of novel algorithms, computation ability of a computer, availability of big data, DL can provide a much better comprehension of the environment. Thus, DRL combines DL and RL, which makes good use of this outstanding property of DL and achieves good performance in robotics, game playing, and spoken dialogue system. Deep Q-Learning is one of the most popular types of DRL. The details of how it works will be introduced in the following several subsections. In Section 2.4.1, we introduce how we obtain our training data and the method for data preprocessing. In Section 2.4.2, some concepts of Q-learning is introduced. Combining DL and Q-learning, the technique of deep Q-networks is introduced in Section 2.4.3, and actor-critic Q-networks scheme is described in Section 2.4.4. Finally, in Section 2.4.5, we explain the exploration policy of the deep Q-networks (DQNs) agent.

2.4.1 Data Preparation and Preprocessing

The traffic simulator used in our work is named Streets4MPI [24], which is a software that can simulate simple street traffic patterns based on street maps imported from OpenStreetMap [42]. It is written in Python and supports parallel computation. OpenStreetMap provides street maps from the countries all around world. We can set some parameters in the simulator to obtain different simulation results. We can load different street map by changing the parameter "osm_file". The parameter "number_of_residents" can be set to assign the total number of trips calculated in one simulation round. We also can set "max_simulation_steps" to change how many times the simulation execute. One simulation step will output one static traffic heat map for the current time. Thus, we can attain a series of traffic heat maps representing traffic topologies from time to time. After we get the traffic map, we first downsize it to make it only contain the content we need. Then, we convert the RGB traffic map to greyscale and improve the contrast of the image, which reduces the amount of computation and also improves the training speed.

We use a matrix to represent the wireless communication topology. The size of the matrix is the same as the traffic map. The spots of the helper and its surrounding connected vehicles will be annotated with 1. Other spots are annotated with 0. We first normalize the pixel value of traffic maps from 0 to 1. Then the normalized traffic topology (traffic heat maps) will be stacked with wireless communication topology (sparse matrix), which serves as the input to the first convolutional layer (shown in Fig. 2.4). Note that the traffic topology comes from the real traffic simulation brought by Streets4MPI. The communication topology is the result of the virtual relay selection at the previous time stamp.

2.4.2 Q-Learning

We utilize a category of model-free reinforcement learning algorithms named Q-Learning. Compared with the value iteration in Section 2.3, Q-Learning will evaluate Q-Values, which is a state-action values that explicitly tell agent how to react to the observation. $Q^*(s, a)$ is denoted as the expected sum of discounted long-term rewards when the agent takes action a under state s. The iteration function of Q-Learning can be denoted as

$$Q_{k+1}(s, a) \leftarrow (1 - \eta)Q_k(s, a) + \eta(r + \chi \cdot \max_{a'} Q_k(s', a')), \tag{2.8}$$

where η is the learning rate. r is the instant reward agent can obtain when it leaves state s with action a. After s and a, the agent would continuously act optimally with discount rate χ.

There are a few techniques for solving the Q-Learning problem. If the states space and action space are limited and discrete, we can utilize a tabular method, which generates a matrix called the Q-Table. The entry of the Q-Table at row i and column j represents the Q-Value $Q(s_i, a_j)$ for taking the j^{th} action a_j under the i^{th} state s_i. At each iteration, the value for one entry of the Q-Table will be updated. After enough iterations, every Q-Value in the Q-Table will become the optimal value.

However, the tabular approach can only solve the Q-Learning problem with finite state and action space. The state space in the real world is almost infinite and table cannot represent all the states. Thus, neural networks are introduced to be used as a function approximator, which maps the states to the Q-Values corresponding to certain actions.

2.4.3 Deep Q-Networks

As we have introduced in Section 2.4.2, the traditional Q-learning uses the table based method to update the long-term reward for action under a certain state. However, the table method cannot scale, which means it cannot handle the situation where the number of states is nearly infinitely large. To tackle this problem, deep Q-Networks (DQN) replaces the table as a new approach to obtain the approximation of Q-Values. Deep Q-Networks can take high-dimension features as input. In our problem, the input is a concatenated 2D image, which contains traffic and V2V communication topology. Both the traffic topology (traffic density map) and wireless communication topology (data allocation map) evolve over time. The traffic topology evolves independently, which obeys the rule of the physical model of the traffic. The data allocation map at the current time step is determined by the action agent takes at the previous timestamp and the traffic density around the current location of the virtual relay node. After the input layer, we stack several convolutional layers followed by several fully connected layers (totally 4 layers) as the structure of our deep Q-networks, which is demonstrated in Fig. 2.4. For every convolutional layer, the computation can be denoted as

$$z_{i,j,k} = b_k + \sum_{u=1}^{f_h} \sum_{v=1}^{f_w} \sum_{k'=1}^{f_{n'}} x_{i',j',k'} \cdot w_{u,v,k',k}, \tag{2.9}$$

where $i' = u \cdot s_h + f_h - 1$ and $j' = v \cdot s_w + f_w - 1$. $z_{i,j,k}$ is the output of the neuron located in row i, column j in feature map k of the convolutional layer (layer l). s_h and s_w are the vertical and horizontal strides, respectively, f_h and f_w are the height and width of the field, respectively, and $f_{n'}$ is the number of feature maps in the previous layer. $x_{i',j',k'}$ is the output of the neuron located in layer $l-1$, row i', column j', feature map k' (or channel k' if the previous layer is the input layer). b_k is the bias term for feature map k (in layer l). You can think of it as a knob that tweaks the overall brightness of the feature map k. $w_{u,v,k',k}$ is the connection weight between any neuron in feature map k of the layer l and its input located at row u, column v (relative to the neuron's receptive field), and feature map k'.

From one layer to another layer, including both convolutional layer and fully-connected in our DQN model, we utilize Rectified Linear Units [49] as the activation function. In short, we denote z as the output, and x' as the input of the next layer. Then we have the relationship as $x' = max(z, 0)$.

Generally, convolutional layers will help extract features from images and reduce the dimension of the data. The output of the last convolutional layers, which is a 2D array, should be converted to a vector and serve as the input of stacked fully-connected layers. The fully-connected layers will select the features from high dimension data. The output of the last fully-connected layer is a vector, which demonstrates the predicted Q-Value for each corresponding action.

Algorithm 1: Deep Double Q-Learning algorithm.

```
1: Initialize:
       • experience replay memory,
       • training interval and copy interval, $T_{int}$ and $T_{cop}$,
       • boolean variable $flag$ to indicate whether the game ends or not,
       • the actor network with parameter $\theta_{actor}$,
       • the critic network with parameter $\theta_{critic} = \theta_{actor}$.
2: for training steps $n = 1, 2, ..., N$ do
3:     Preprocess the initial observation $s_0$ to get beginning input $x_0$
       for neural network.
4:     for game episode $t = 1, 2, ..., T$ do
5:         Generate a random probability $p$.
6:         if $p \leq \varepsilon$ then
7:             randomly select an action $a_t$,
8:         else
9:             $a_t = \arg\max_a Q(x, a, \theta_{actor})$.
10:        end if
11:        Execute action $a_t$, get reward $r_t$ , next observation $s_{t+1}$ and $flag$
           as feedback.
12:        Preprocess $s_{t+1}$ to get next state $x_{t+1}$.
13:        Store $(x_t, a_t, r_t, x_{t+1}, flag)$ into experience replay memory.
14:        if $n \bmod T_{\text{int}} \neq 0$ then
15:            continue.
16:        end if
17:        Get a batch size of $M_{batch}$ samples $(x^{(i)}, a^{(i)}, r^{(i)}, x^{(i+1)}, flag)$
           from experience replay memory.
18:        Calculate target Q-Value, $y^{(i)} = r^{(i)} + \chi \cdot \max_{a'} Q(x'^{(i)}, a', \theta_{actor})$.
19:        Update the critic network by minimizing the cost function $J(\theta_{critic})$,
           $J(\theta_{critic})=\frac{1}{M_{batch}} \sum_{i=1}^{M_{batch}} (y^{(i)}-Q(x^{(i)},a^{(i)},\theta_{critic}))^2$,
           and perform the gradient descent method on $J(\theta_{critic})$
           with respect to $\theta_{critic}$.
20:        if $n \bmod T_{cop} = 0$ then
21:            $\theta_{actor} = \theta_{critic}$.
22:        end if
23:    end for
24: end for
```

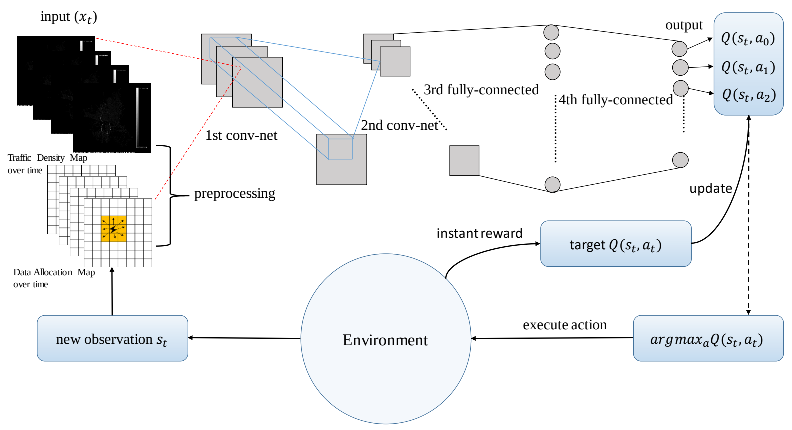


Figure 2.4: Deep Q-learning procedure for the proposed virtual relay selection problem

2.4.4 Double DQNs and Cost Function

We use two DQNs with the same architecture, but different trainable parameters, i.e., θ and θ', respectively. One DQN (the actor) will be used to drive the movement of the relay node, and the other DQN (the critic) will watch the actor's trials and learn from its mistakes. The critic, which is also known as the target Q-Network, will compute the loss for every action actor takes during training. The reason for using a separate DQN to generate the target Q-value is that the value of Q-Network will shift at every training step [25]. Using only one network usually incurs uncontrolled estimation of value. Based on the feedback loop structure of reinforcement learning, the network will become more and more unstable.

At regular and frequent intervals [50], the trainable variables in the critic network will be completely copied to the actor network. The updated actor network will start to take actions again, and critic will watch and learn from a new cognition level. This double Q-network structure is shown in Fig. 2.5.

In order to let critic learn, all experience the actor obtains during playing will be stored into a container called experience replay memory. Every unit in the experience replay memory can be described as list of the current state s_t at time t, the action a_t taken under the current observation s_t, the next observation s_{t+1} and reward r_t for taking a_t under

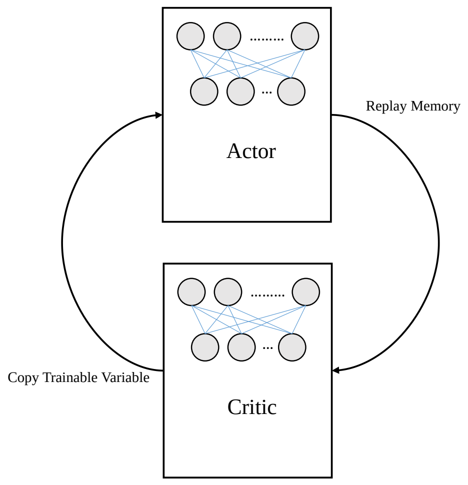


Figure 2.5: Double Q-network structure

s_t. Suppose the critic will fetch several units at once from replay memory for training. The number of units fetched at one time is called the batch size, which is denoted as M_{batch}. The cost function used for training is

$$\begin{aligned} J(\theta_{critic}) &= \frac{1}{M_{batch}} \sum_{i=1}^{M_{batch}} \left(y^{(i)} - Q(s^{(i)}, a^{(i)}, \theta_{critic})\right)^2, \\ with\ y^{(i)} &= r^{(i)} + \chi \cdot \max_{a'} Q(s'^{(i)}, a', \theta_{actor}), \end{aligned} \tag{2.10}$$

where θ_{critic} and θ_{actor} are the parameters for the critic network and actor network. $s^{(i)}$, $a^{(i)}$, $r^{(i)}$ and $s'^{(i)}$ are the current state, action, reward and next state of the i^{th} memory sampled from the experience replay memory, respectively. $Q(s'^{(i)}, a', \theta_{actor})$ is the prediction of Q-Value actor expects from the next state $s'^{(i)}$ if actor chooses action a'. $Q(s^{(i)}, a^{(i)}, \theta_{critic})$ is the prediction of Q-Value of the i^{th} memory from the expectation of critic network. y^i is the target Q-Value for the i^{th} memory. $J(\theta_{critic})$ is a cost function, which is based on mean squared error. By using mean squared error, we can easily derive the gradient, which saves the computational resources. On the other hand, we don't have outliers in the data set. The RGB value of the traffic heat map is normalized (the value falls in (0,1)), and the value function is a continuous function of the input data. So there are no outliers in the data set, which is the main problem that we should consider when we use the mean squared error. As we can see in (2.10), how the critic is trained depends on the experience which actor learns.

The training procedure for deep double Q-learning is shown in Algorithm 1. Before the training start, we first initialize the trainable parameters for the actor and critic networks, and the intervals for copying the critic's parameters to the actor's parameters. Also, experience replay memory is initialized to store the experience from the playing of the actor. In Algorithm 1, we set two loops of iterations. The outer loop is for training iterations. The index for counting training steps will increase by one for every time training occurs. The inner loop is for game iterations. For example, if we calculate the cumulative rewards every time after the virtual relay node moves 100 steps, the 100 steps of the virtual relay node movement are one game iteration. By doing so, we can periodically record the performance of the DRL agent. In every training step, DRL agent will move the virtual vehicle relay node T times to finish one complete game. In every game iteration, the deep Q-networks will update its parameters in the way that cost function is minimized.

2.4.5 Exploration Policies

When the actor is making movement in the environment, it faces a dilemma of balancing exploration and exploitation of the environment. Exploitation means that the actor should choose the action based on the best policy it currently knows. On the other hand, exploration means that the actor should explore more about the environment in order to gather more information. In other words, the best long-term strategy may involve short-term sacrifices. In order to gather enough information to make the best overall decisions, we intuitively expect the actor to explore the environment as thoroughly as possible. From the MDP point of view, if every state and every transition can be visited enough times, we can expect a better performance from the training, since we remember enough situations in replay memory. However, to achieve that, it will be time-consuming if we let the actor select actions by its own judgment every time since it might have preferences. It's the same as people who visit one restaurant many times, they know what are their favorite dishes and they tend to order their preferences more than dishes they are not familiar with.

It might take a long time for the actor to go through every state and transition. For simplicity, we utilize a technique called ϵ-greedy to artificially add interference. At each step, the actor will acts randomly with probability ϵ, or greedily (choose the optimal action) with probability $1 - \epsilon$. During training, we will gradually reduce the value of ϵ to let the actor explore the environment more at the beginning and less when it already knows plenty of good policies.

2.5 Simulation Results and Discussions

In this section, we evaluate the performance of our scheme by using Google TensorFlow platform. We implement the deep double Q-learning framework for our proposed model and implement other two schemes as baselines. One is the random action scheme and the other is the greedy action scheme. In the random action scheme, the action at every step is chosen randomly. In the greedy action scheme, the agent decides to move the virtual relay node to the one of the adjacent pixels which has the maximum traffic density (pixel value is the largest). In Section 2.5.1, we set up parameters in the simulation. In Section 2.5.2, we demonstrate some simulation results and analyze the performance of our proposed model.

2.5.1 Simulation Settings

In this simulation, we consider the traffic density map, whose size is 100×100. The original coordinate $(0, 0)$ locates at upper left of the traffic map. We set x axis along

Table 2.1: Hyperparameters of the DQNs

Layer	Name	Parameters	Dimensions	Parameter Scale
Input	–	–	(2,100,100)	–
Layer 1	Convolution	Filter (5,5,5,5)	(5,100,100)	125
Layer 2	Convolution	Filter (10,3,3,2)	(10,20,20)	90
Layer 3	Convolution	Filter (10,2,2,1)	(10,10,10)	40
Layer 4	Data flatten	–	(1000,)	0
Layer 4	Fully-connected	–	(512,)	512,000
Output	–	–	(9,)	–

the vertical direction and y axis along the horizontal direction. The location of eNB is at coordinate $(-10, -10)$. The eNB transmits data to the helper, whose coordinate can be at any of those $10,000$ coordinates. Each pixel of the map represents a squared area with the size 10 meters $\times$ 10 meters. The whole squared training field has the size 1000 meters $\times$ 1000 meters. In the simulation, we select an initial coordinate for the helper, and let it move gradually. We set the bandwidth of the transmission as 10 MHz for eNB to the helper and the helper to end users. The transmission power for the eNB to the helper and the helper to the end users are set to 30 dBm. For the transmission from the helper to the subscribers, we set the SNR to a fixed value as 40 dB. For every unit of data received by end user, we give 100 units of revenue. For every unit of power consumed for both eNB and the helper, we give it 1 unit of cost. For every unit of latency of the service, we give it 1 unit of cost. In this way, we unify the units of different components, and let the reward function (2.3) make sense. Every component in the reward function shares the same value for weight β in our experiments. The final reward is calculated by the revenue minus the cost.

For the structure of the proposed DQNs, we stack 3 convolutional layers followed by 1 fully-connected layer. The detailed configuration of the DQNs structure are shown in Table 2.1. The actor and critic Q-network have the same structure. The training occurs after actor takes action every time, which means total training steps are equal to total action steps. The size of experience replay memory is set to 5,000. The size of the minibatch is 10, and thus 10 samples will be randomly fetched from experience replay memory for every

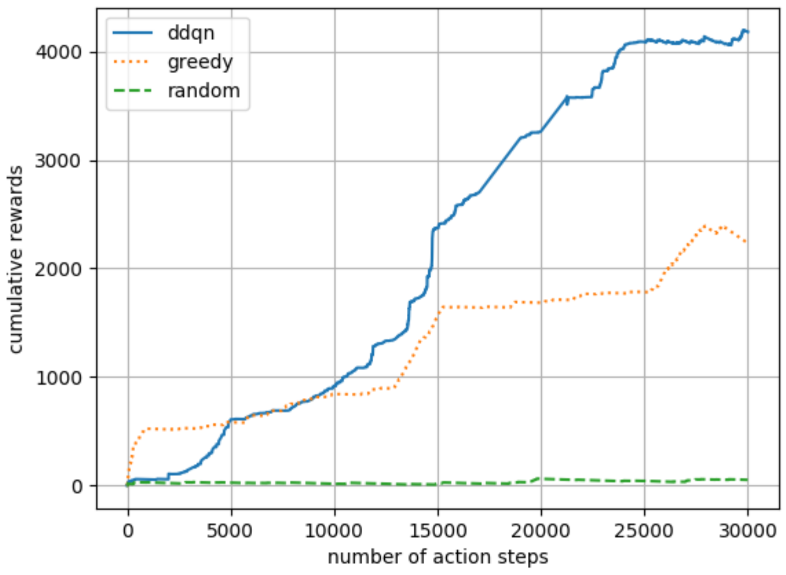


Figure 2.6: The number of action steps versus cumulative rewards

training step. All the trainable variables in critic networks will be copied to actor networks every 10 training steps. The discount rate χ is set to 0.95. The value ϵ is set to 0.95 at the beginning of training. With the iterations of training increasing, ϵ will decay until it reaches minimum value 0.05. 100 different traffic density maps will be fed into our model iteratively. We define every 100 steps as one game.

2.5.2 Simulation Results

In Fig. 2.6, we compare the cumulative rewards gained by our deep double Q-network (DDQN) with the greedy and random action scheme. As we can find out from the result, during the beginning 2,000 steps, the cumulative rewards gained by DDQN are almost the same with the random action approach. After that, the rewards gained by DDQN increase exponentially until it reaches 5,000 steps. From step 5,000 to step 15,000, the rewards gained by the agent take another exponential growth. After step 15,000, the rewards grow linearly until step 25,000. On the contrary, the random action approach gains little overall rewards compared to DDQN during the whole training process. For the greedy ac-

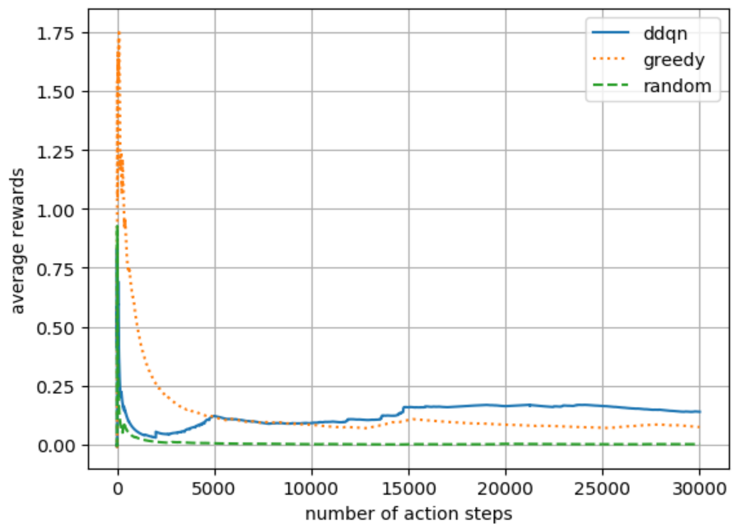


Figure 2.7: The number of action steps versus average rewards

tion scheme, it performs better than DDQN during the first 5,000 steps at beginning, and achieves an early advantage. From step 5,000 to step 10,000, the greedy action scheme accumulates approximately the same amount of rewards as the DDQN. After this period, DDQN has a larger growth rate than the greedy action scheme, and the greedy action scheme cannot catch up with the DDQN anymore. From this result, we demonstrate that the rewards in this scenario not only depend on the traffic density at a single pixel on the traffic density map, but also on the surrounding traffic patterns. The convolution operation in the neural networks works due to the consideration of the relationship between the adjacent pixels (potential location for virtual relay).

In Fig. 2.7, we evaluate the average rewards over iterations. The average rewards are calculated as the cumulative rewards divided by the current number of steps. By doing so, we burnish the cumulative rewards curves by removing variance of rewards from step to step. The average rewards curve of the random action approach is mostly flat, which indicates no growth in rewards harvesting from step to step. On the other hand, we observe that at the first 2,000 steps, the average reward of the greedy action scheme is big, and after that, it drops fast and converge at around 5,000 step. The curve for DDQN is flat and has a

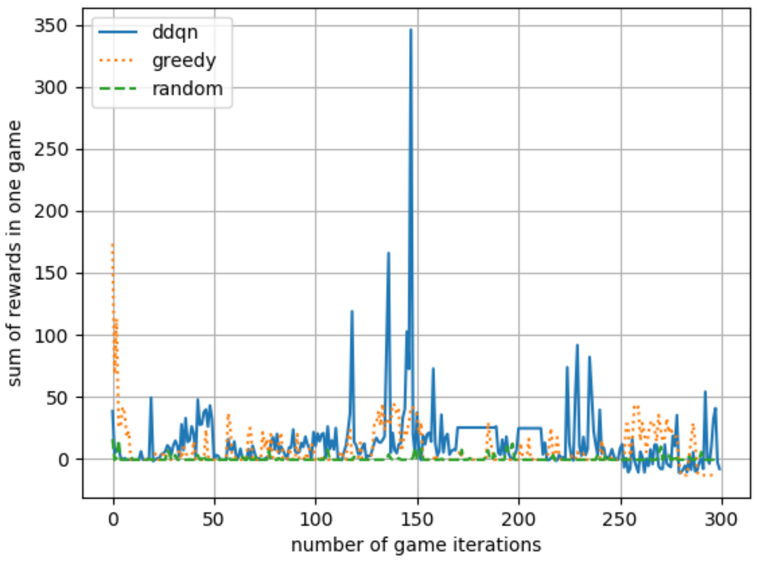


Figure 2.8: The number of game iterations versus rewards collected in one game

slight growth afterward. After around 15,000 steps, all curves converge and DDQN has the biggest average reward.

In Fig. 2.8, we collect rewards gained in every game, which contains consecutive 100 iterations. Since we implement the algorithm for 30,000 iterations, we have the rewards of 300 games counted. We find out some peaks and valleys between 100 game iterations and 250 game iterations, which corresponds to action steps (or training steps) from step 10,000 to 25,000. Compared with Fig. 2.6, we find out that these spikes correspond to the sudden rewards growth after a small period of saturation. We observe that the values of rewards in valleys during this period are almost the same with the rewards of the random action approach. After those valleys, some peaks can be observed, which indicates that DDQN learns how to deal with those circumstances and recover from those adversities. Compared with DDQN, the greedy action and random action scheme have smaller variance from game to game, which indicates that there is no such learning moments happen compared with DDQN as we have discussed.

In Fig. 2.9, we evaluate the mean squared error (MSE) of our DDQN model, which measures the squared value of the difference between the predicted Q-value and target Q-

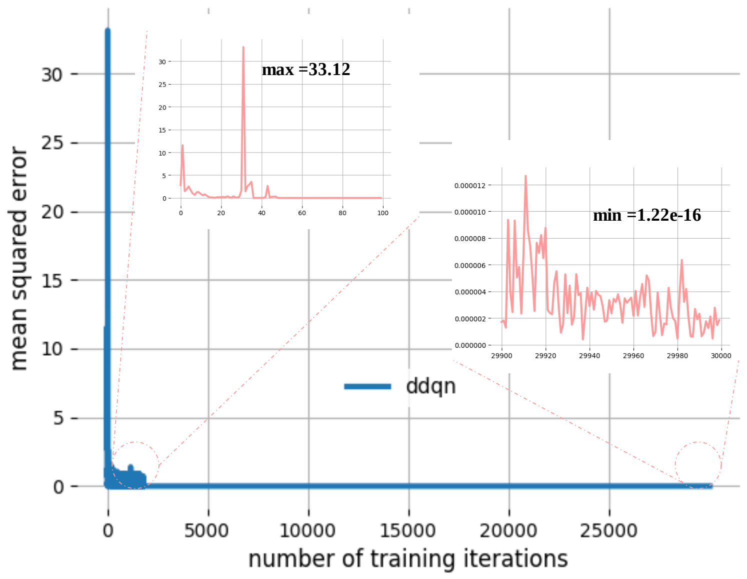


Figure 2.9: The number of training iterations versus mean square error

value. At the very beginning, the MSE is high, but drops dramatically and remains at a low level after the step 2,000. This curve demonstrates that the agent went through a studying period during the first 2,000 steps. Afterward, it handles the problem well and produces an accurate prediction of the Q-value. In order to show the result clearly, we zoom in to the first 100 steps and last 100 steps of the training, which is shown by the pink curves. The maximum and minimum mean squared error of the whole training process are 33.12 and 1.22e-16, respectively.

According to the demonstration of the figures we have discussed, there is a period at beginning of the training (the first 2,000 steps). After this period, the agent reduces errors and collects positive rewards. The behavior of the agent is represented in Fig. 2.10. We sample 6 traffic maps of the time stamps during that period. We can clearly observe in those traffic maps that there are some eye-catching patterns in the maps. At each pixels, the brighter the color is, the higher traffic density is. At step 5,000, the coordinate of the helper (relay point, which is a red dot in the map) is at $(38, 25)$ when the agent stays in an area where there exists heavy traffic. From step 12,000 to 14,000, we discover that the virtual

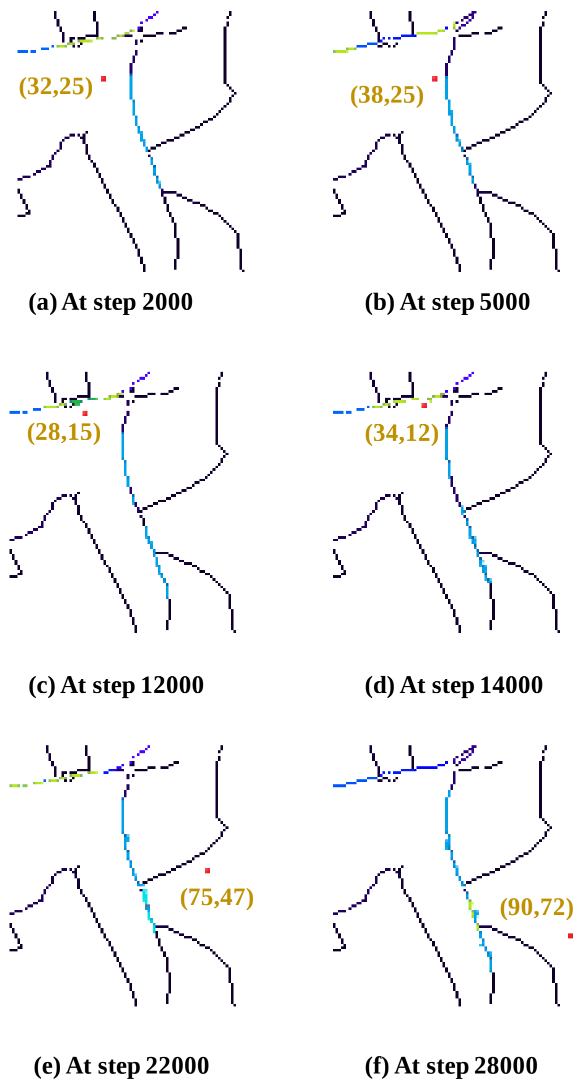
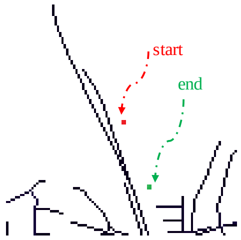


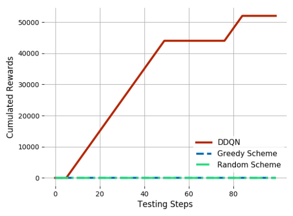
Figure 2.10: The training behavior of the virtual relay node

relay node hangs around the left corner of the map, which let it continuously and linearly collect rewards, which we can refer to the Fig.2.7. From step 22,000 to step 28,000, the virtual relay node gradually moves to the right bottom area, where the traffic density is low. Referring to Fig. 2.8, we also observe some negative rewards collected during this period accordingly. From Fig. 2.10, we demonstrate the behavior of the virtual relay node, and the relationship between the behavior and the numeric training results we have discussed and shown in previous figures.

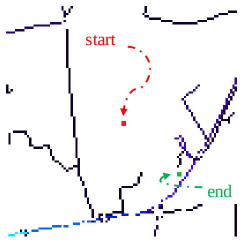
The above results show the training performance of the DDQN agent in a 100×100



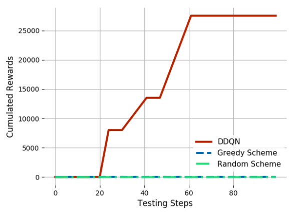
(a) 1st testing field



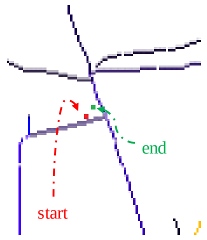
(b) Testing results in 1st testing field



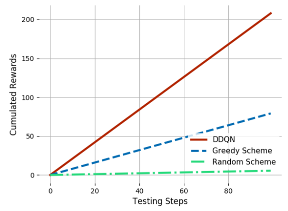
(c) 2nd testing field



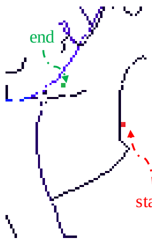
(d) Testing results in 2nd testing field



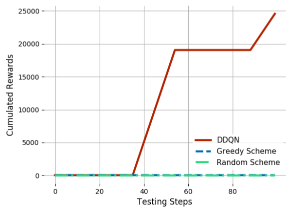
(e) 3rd testing field



(f) Testing results in 3rd testing field



(g) 4th testing field



(h) Testing results in 4th testing field

Figure 2.11: The visualization of the testing fields and corresponding cumulated rewards

sized field. For validation, we also let the agent select virtual relay node in a squared field with side length 100, however with 4 different traffic maps (thus different states or inputs). Considering the starting point of the virtual relay might affect the performance of the agent, we generate 4 traffic maps which are different with the training field. The testing process goes through 100 steps for each testing traffic topologies.

In Fig. 2.11, we evaluate the cumulated reward through testing steps. We select 4 different traffic topologies in order to diminish the performance difference caused by the

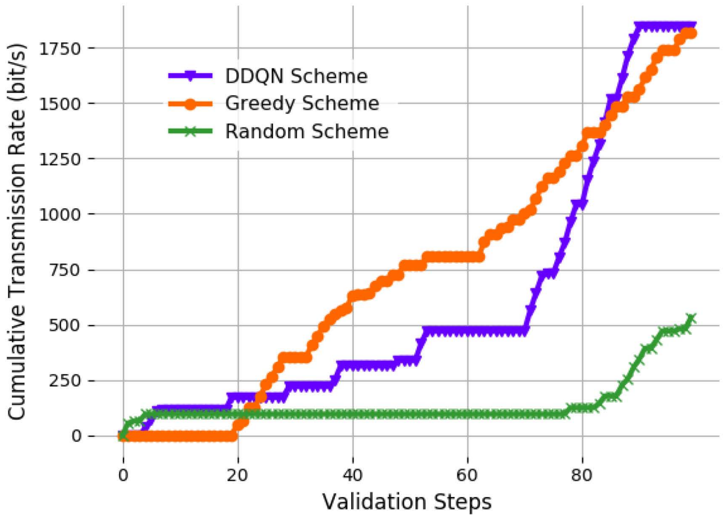


Figure 2.12: The number of validation iterations versus cumulative transmission rate

variation of traffic condition. We should consider the agent scheme to be robust if the agent is going to make wise decision no matter what traffic topology it deals with. In the left column of the figure, we can see the 4 subgraphs showing the 4 different traffic topologies we used for testing. On each traffic graph, we annotate the starting and ending spot of the virtual relay node. In the right column of the figure, 4 subgraphs shows the corresponding cumulated rewards collected through testing steps. We can observe from the results that DDQN scheme has much better performance than the greedy and random scheme. In Fig. 2.11(a), (c) and (g), we can see that the position of the virtual relay node goes through a significant change. Along these movements, the cumulated rewards gain some huge leaps, which demonstrate the intelligence the agent gained. In Fig. 2.11(e), we can see that the position of the virtual relay node barely changes. From Fig. 2.11(f), we observe that the agent considers that the sweet spot is around the starting point. A relatively constant growth of cumulated rewards is obtained by all of the three schemes. However, the DDQN scheme has larger gradient than the other two schemes.

In Fig. 2.12, we compare the cumulative transmission rate through 100 testing steps by three different schemes. There are a few saturation found in the curves, which means no

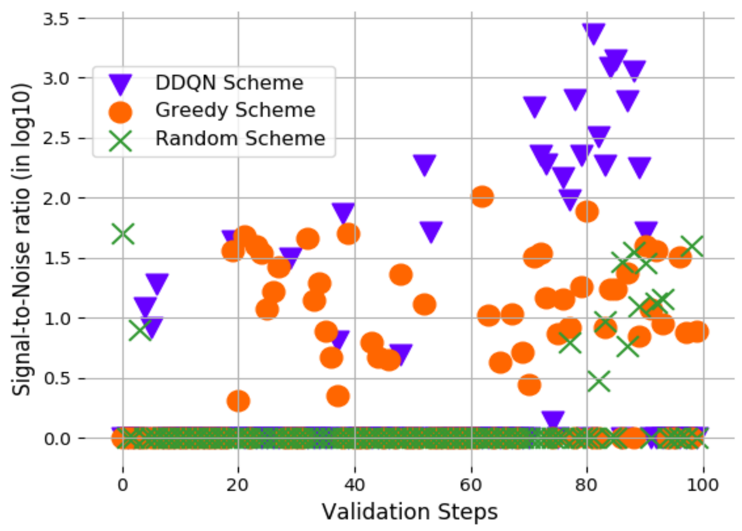


Figure 2.13: The number of validation iterations versus signal-to-noise ratio

transportation traffic found at that step, or in other words, no transmission connection built. From the result, we observe that the greedy scheme has an edge over the other two schemes at early steps. With the validation going on, the DDQN scheme catches up and surpasses the greedy scheme, which shows that the proposed DDQN scheme is more powerful harvesting the opportunity of transmission connection in the long term.

In Fig. 2.13, we use the signal-to-noise ratio (SNR) in $log10$ at the vehicle relay as the channel quality indicator. There are a few zero values on the x-axis representing no signal received at the vehicle relay instead of $SNR = 1$. We compare the result of the three schemes. The random scheme has an unsteady and discontinuous transmission. It loses the connection for a long period in the middle. The greedy scheme has a steady and continuous connection with a concentrated distribution of the SNR. The DDQN scheme is able to find better spots in order to achieve better SNR with the validation going on. The average value of the SNR from DDQN scheme is also higher than the greedy and random scheme.

In Fig. 2.14, we use bar plot to demonstrate the relative number of the connected end users which are served by the helper. This relative value is calculated based on the opportunistic contact $O_{v_e,u}$ defined in Section 2.2.2, which shows the transmission coverage

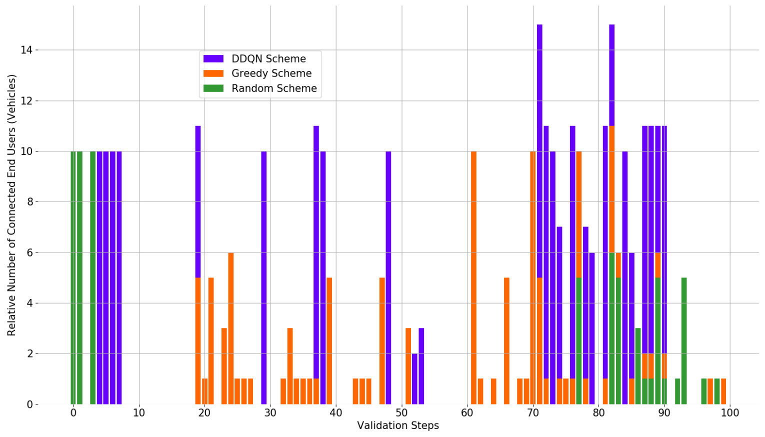


Figure 2.14: The validation iterations versus the relative number of the connected end users

of the helper. We have three findings from this result: 1) Our proposed DDQN scheme and the greedy scheme have more steady transmission coverage than the random scheme. 2) By using the greedy scheme, we can ensure the transmission connection to some extent. However, the average coverage of the end users is smaller than the DDQN scheme. 3) Generally, once the DDQN scheme builds a connection, it covers more vehicles than the greedy and random scheme.

The analysis above demonstrates that the agent in our model has achieved intriguing intelligence. DRL makes it possible to let agent make wise decisions and overcome obstacles in the environment. From the aspect of vehicle relay, our proposed model helps select location for the virtual vehicle relay with only traffic information. Moreover, by utilizing our scheme, the optimized action can be chosen for setting the position of vehicle relay at the current time without knowing the traffic condition in the future. The long-term maximum utilities can be obtained and corresponding trajectory of the virtual vehicle relay node can be produced by the agent in the proposed model. In this work, we only consider one vehicle relay scenario. From MDP point of view, if we consider n relays, every relay has 9 actions, then the size of the action space becomes 9^n. The state space in our problem is generally infinite since the distribution of the traffic changes continuously. The more action taken basically means more state encountered.

2.6 Conclusion

In this chapter, we proposed a deep reinforcement learning framework in the LTE-V scenario for virtually selecting vehicle relay node. In the framework, we use deep Q-networks as our agent, which contains two types of networks. Firstly, we use deep convolutional neural networks to extract traffic patterns in the input. Then, fully-connected networks are utilized to flatten the high dimensional data and map to the Q-values as output. Q-learning is used to obtain the target Q-values by interacting with the environment. The target Q-values can be used as the labels for training deep Q-networks. Deep Q-networks agent will move the virtual vehicle relay node gradually on the traffic map. Every movement will arise rewards, the amount of which depends on the next observation. We train our agent by letting it interact with environment iteratively and learn from trials and errors. Simulation results show that agent has an excellent performance in rewards collection and error correction. Compared with the greedy and random decision-making schemes, our agent is able to improve utility performance dramatically.

Chapter 3

Classifying Cutting Volume at Shale Shakers in Real-Time Via Video Streaming Using Deep Learning Techniques

Oil and gas well drilling is a challenging task due to time-consuming, complex and expensive operation. Proper management of the drilling process and risks will save cost for the drilling operators both in the aspect of time and economy. Being able to conduct real-time operational optimization ensures a low-cost drilling execution, and at the same time safety is guaranteed. Hole cleaning is one of the major challenges and must-do operation. A complete workflow has already been built to guide the maintenance and cleaning of the borehole for many oil and gas companies. A well-formulated workflow helps support the well integrity and reduce drilling risk and cost. One of the traditional methods needs human observation of cuttings at the shale shaker and a hydraulic and torque- and-drag model, whose operation includes a number of clean-up cycles. This continuous manual monitoring of cutting volume at shale shaker becomes the bottleneck of the current workflow and is unable to provide a consistent evaluation of the hole cleaning condition since the human labor cannot be available consistently, and the torque- and-drag operation is discrete containing break between two cycles. As a result, the current workflow cannot fully address the risk of stuck pipe or casing due to the poor controllability of the hole-cleaning.

Some researches have already addressed the automated cuttings analysis, which is employed by industry for monitoring the drilling process. Graves et al. [51] proposed an automatic image processing system to quantify the shape, size distribution, or the volume of the cuttings, which is a downhole cutting analysis patent application. The system consists of some equipment, which includes multiple high definition CCD cameras, lighting equipment for illuminating the surface of the shake, a series of energy modification devices, coaxial or fiber optic cables for data transmission, and the local or remote data processing software package. Parmeshwar et al. [52] proposed another shale shaker imaging system which contains two cameras for capturing visual and infrared data, respectively. Apart from the main camera module, the assembly of the system also includes a motion detector to coordinate the control of the lighting source. Torrione et al. [53] proposed a cutting volumes analysis system using high- definition and calibrated camera. Stereo information is obtained by using multiple cameras and devices. Multiple light sources are arranged to illuminate the shaker table and normalize the brightness of the background. Besides camera-based detection, a multitude of sensors has also been used to monitoring the cuttings in some researches. For example, Dahl et al. [54] proposed to use acoustic sensors to produce information responsive to a reflection of an emitted wave from downhole cuttings in the borehole, wherein the information is indicative of a parameter of interest relating to the downhole cuttings. After studying all the previous works and existing systems, we found one common shortcoming that the high standard is required for setting up the data acquisition system. However, this high standard can hardly be achieved due to the tough environment at the offshore rig. Besides, strict lighting and optical filters requirements raise the cost and at the same time brings more safety concerns. From the data processing point of view, only low-resolution video can be obtained due to the limitation of the bandwidth of the data transmission for real-time video streaming. The low-resolution video needs more experienced engineers for analysis.

Most of the previous work used image analysis techniques to perform the quantitative analysis on cutting volume. Guilherme et al., [55] applied a support vector regression to perform the cutting image analysis. A graph theory based optimal path forest approach was proposed to analyze the cutting flow in real-time [56]. Particle image velocimetry is another quantification method that has been used to estimate the flow rate (Mc- nutt et al., 2012). Beyond the traditional machine learning techniques used in image processing, such as K-means clustering [57], Support Vector Machine [58], and Random Forest [59], the deep learning framework [60] dramatically changes the landscape of artificial intelligence in the field of image processing and computer vision. The traditional image processing approach requires significant work on feature engineering. Since the raw data is usually noisy with missing components, preprocessing and augmenting the data play an important role in making the learning model more efficient and productive. Routray et al. [61] summarized the feature engineering in traditional learning frameworks. Scale- invariant feature transformation, the histogram of oriented gradients, and speeded up robust features are usually used as the artificially synthesized features. The deep learning framework, on the other hand, discovers the rep- resentations needed for feature detection or classification from raw data automatically. By using deep learning framework, it can help to overcome the difficulties in setup monitoring devices in a harsh environment and the data acquisition requirement for a cutting volume monitoring system at the offshore rig might be relaxed.

The objective of this study to verify the feasibility of building a real-time automatic cutting volumes monitoring system on a remote site with limited data transmission bandwidth. The minimum data acquisition hardware requirement includes:

- Single uncalibrated CCD camera
- Inconsistent lighting sources
- Low bit rate transmission (e.g., 137Kbps)
- Image processing unit without GPU support (e.g., a laptop)

A Deep Neural Network (DNN) [62] is adopted to perform the image processing and classification on cutting volumes from a shale shaker at a remote rig site. Specifically, the CNNs [63], which is successfully deployed in image classification, segmentation, and object detection etc., is implemented as the feature extractor and classifier in our model. In our study, we adopt the VGG- Net [64] as the base model. The main contributions of this study are summarized as follows:

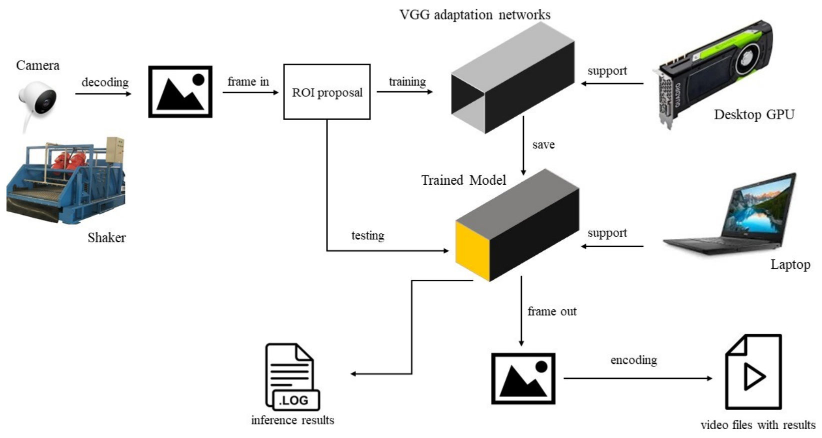


Figure 3.1: The overview of the real-time cuttings volume monitoring system

- A deep learning framework that can classify the volume of cuttings in real-time.
- A real-time video analysis system that requires minimum hardware setup efforts. The system is capable of processing low-resolution images acquired by uncelebrated cameras.
- An object detection workflow is built to automatically detect the region covered by cuttings.
- A multi-thread video encoder/decoder is implemented to improve the real-time video streaming processing.

This chapter is organized as follows. In Section 3.1, an overview of our system model is provided. Section 3.2 outlines detailed techniques for implementing the system, which includes ROI detection, data pre-processing, and the adaptation of the VGG networks. In Section 3.3, we introduce the experiment results and some performance discussions. Finally, we conclude the paper with future works in Section 3.4.

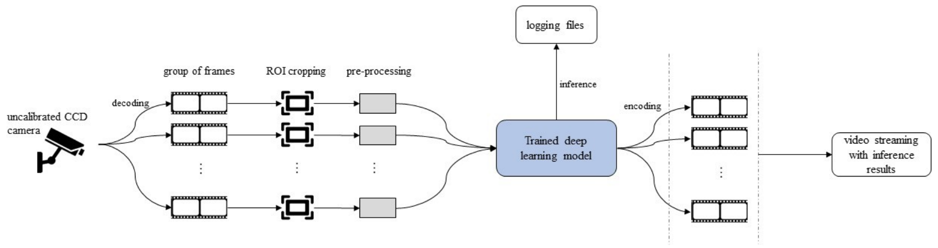


Figure 3.2: The workflow of the real-time video analysis system

3.1 Overview of the Real-Time Cutting Volume Monitoring System

The overall workflow of our real-time cutting volume monitoring system is introduced in this section (referred to Fig. 3.2). The workflow mainly consists of the following child processes: 1) The real-time video processing (decoding and encoding); 2) region of interest proposal (the region covered by cuttings); 3) data pre-processing and deep learning classification module. During the drilling process, cuttings with mud are transported through the vibrating shale shaker. We develop an intelligent video processing engine to analyze videos captured when the cuttings are transported to the shaker. The analysis results will be transported and presented on a monitor in the office in real-time, which is convenient for the drilling engineer to get the information of cutting volume timely. The continuous and real-time inference (classification) results can be used as the histogram for further analysis in the future.

The real-time video processing module is designed for adapting to the dynamic drilling environment. Monitoring cutting volume at shale shaker in real time is an important approach of overall drilling risk management for oil and gas well drilling. The advantages of our proposed system over other previous machine learning-driven systems can be summarized as follows: 1) Inference results can be delivered in real-time by interpreting live streaming-video; 2) Generate reasonably good classification results; 3) The workflow can be implemented with limited computing resources (e.g., a laptop GPU)

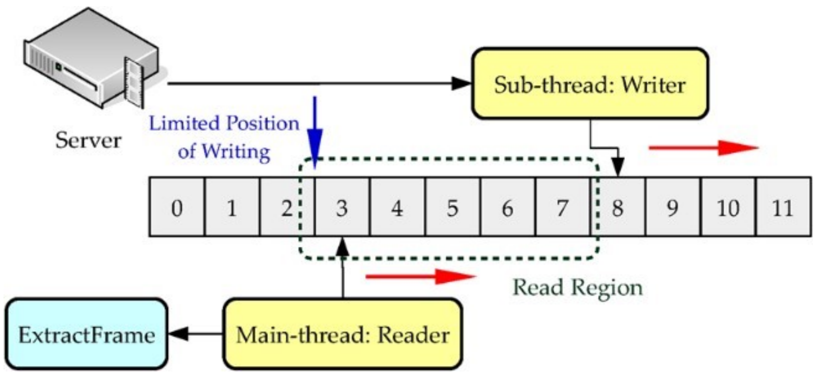
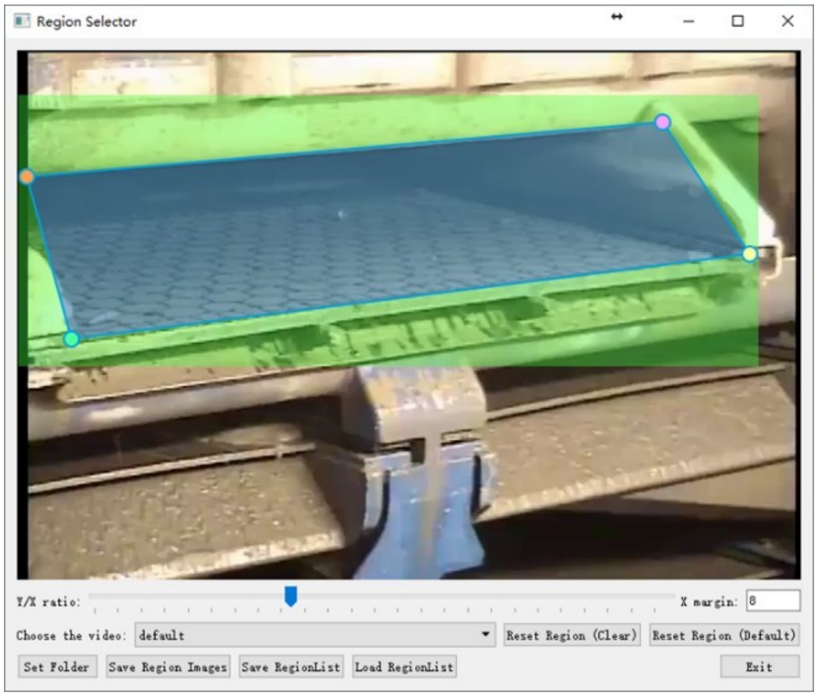


Figure 3.3: The two-threads with buffer mechanism of writing and reading video frames

There are several uncalibrated CCD cameras recording the shaker tables of several shakers at the rig. The video captured by the cameras will be transmitted to a remote workstation off the rig via a streaming service named Rig- Site Virtual Presence (RSVP). The resolution of the video is comprised due to the limited network bandwidth. The raw frames captured by the camera contains some useless information such as the main body of the shale shaker which contributes little to the final inference results. In order to remove the useless information, an automatic region proposal mechanism is proposed to successfully get the attention of the model to the region which is covered by the flow of cuttings. This mechanism is based on the Faster-RCNN [65], a highly-accurate object detection model. By finding the region of interest, a fine-grained input with much lower dimension is generated.

After finding the region of interest and cropping the original input to one with a much lower dimension, we feed the fine-grained data into the data pre-processing and deep classification model. In the phase of data pre-processing, subsampling and normalization are implemented on every frame. Subsampling helps further removal of irrelevant information on every frame, which is introduced in detail in Section 3.2.3.

Normalization reduces the noise and puts the features in the input into the same scale. After preprocessing, every frame will be trained and tested through an adaptation of VGG-16 networks. The networks are built upon a VGG-16 [64] with a customized number of layers, normalization and regularization terms. For training the deep neural network, we



Y/X ratio:

X margin: 8

Choose the video: default

Figure 3.4: GUI of selecting region of interest according to different camera angles

collected about 10 hours of video and added labels to each frame of the video manually. The trained model is then evaluated on the separate videos. The classification inference will run with each frame, which produces the label of the frame, confidence factor of the frame, and delivers the encoded results in real-time. The evaluation result is verified by our developers as well as the experienced engineers.

3.2 Methodologies

In Fig. 3.2, we demonstrate the workflow of the real-time video analysis system for classifying the cutting volume. There are several processes involved. At the beginning and the end of the pipeline, decoding and encoding operations are implemented. The decoding operation decodes the raw video stream to several groups of frames. The encoding operation collects the inference results and combines them with corresponding frames. In this way, engineers can get real-time feedback on monitoring results. In the middle of the pipeline, we implement ROI proposal and preprocessing of data in order to guarantee the

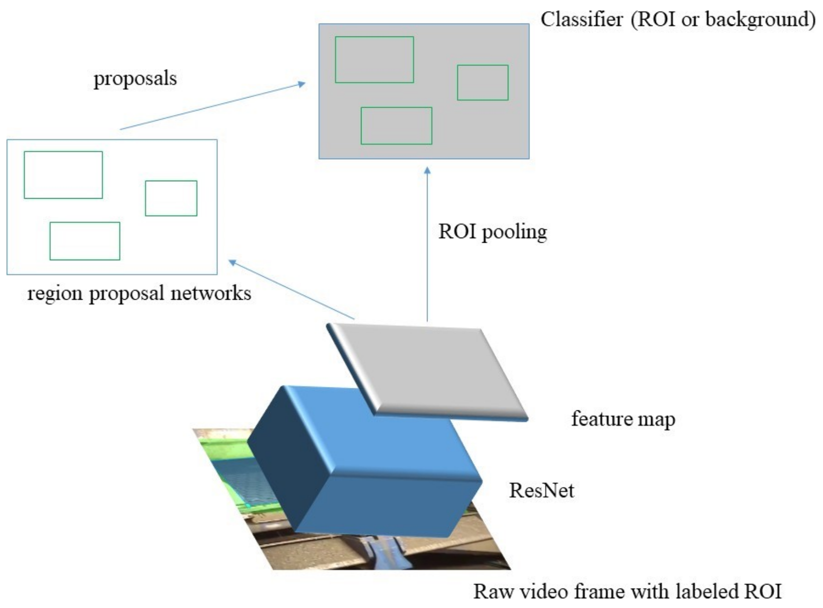

Classifier (ROI or background)
proposals
ROI pooling
region proposal networks
feature map
ResNet
Raw video frame with labeled ROI

Figure 3.5: The structure of Faster-RCNN based cutting flow detection

conciseness and simplicity of the data. Before feeding data into deep classification model, we implement all operations in a multi-threads way. Generally, one GOP will be processed in one thread. The deep learning classification iteratively takes these GOPs as training batch data. The inference results of the deep learning classification model can be saved either into log files or video streaming if required by the engineers.

3.2.1 Video Frames Extraction

In Fig. 3.3, we demonstrate the two-threads mechanism for reading/writing the source stream in real-time. The decoding process should be on the fly since the server is pushing the video stream continuously. If the decoding process fails to catch up with the speed of the video stream, there is a chance that we lose the synchronization and drop frames. This problem fits the producer-consumer pattern. In order to overcome this obstacle, a fast thread-safe circular buffer is implemented to avoid any race conditions.

The writer thread (or we call it producer thread here) continuously decodes the stream without any interruption. It is called producer since it would write any newly-decoded frames into a circular buffer shown in Fig. 3.3. The circular buffer is a common and fixed-size buffer used as a queue and shared by both the producer and the consumer. The reader thread (or we call it consumer thread here) moves its pointer to the beginning of the circular buffer (In the example shown in Fig. 3.3, the buffer starts from the frame No. 3 and ends at frame No. 7) and starts extracting frames. Parallelly, the writer thread (the producer), continuously to insert newly decoded frames into the buffer, though it is a few frames (the size of the buffer) ahead of the reader thread (the consumer). Since the size of the buffer is fixed and limited, the reader thread (the consumer) will move its pointer back to the beginning frame of the buffer after it reaches the end. The buffer size is set to an appropriate scale so that the consumer will not block the producer, or occupy too much space in the memory.

3.2.2 Region of Interest Proposal

In order to guarantee steady inference results, the users (engineers or developers) need to provide the ROI to indicate the area where cuttings are flowing on the shaker. Our learning model will pay attention to this ROI and get the input data with much less variety. We assume that the camera will not change its position and angle after we settle the ROI. The ROI will filter out many noises interfering with the classifier. We can either use a manual approach or an automatic approach to facilitate ROI selection. Before the decoding of the video stream starts, the interactive GUI (shown in Fig. 3.4) will present one frame to the user indicating the position of the shaker. The user can highlight the ROI by simply selecting four corner-points from the very first frame demonstrated by the GUI. There are options for users to change the scale and length-to-width ratio of the ROI to fit the input dimension of the classifier.

However, the manual region selection requires repeated labor. For a certain shaker, the camera angle might be changed slightly during the drilling operation purposely or ac-

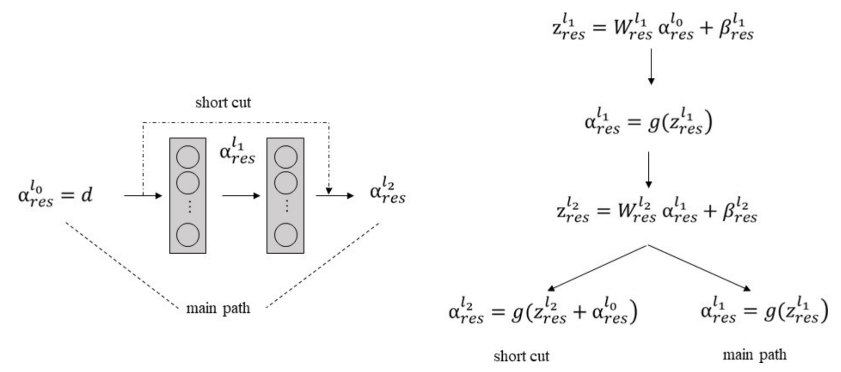

short cut
$\alpha_{res}^{l_0} = d$
$\alpha_{res}^{l_1}$
$\alpha_{res}^{l_2}$
main path
$z_{res}^{l_1} = W_{res}^{l_1}\,\alpha_{res}^{l_0} + \beta_{res}^{l_1}$
$\alpha_{res}^{l_1} = g(z_{res}^{l_1})$
$z_{res}^{l_2} = W_{res}^{l_2}\,\alpha_{res}^{l_1} + \beta_{res}^{l_2}$
$\alpha_{res}^{l_2} = g(z_{res}^{l_2} + \alpha_{res}^{l_0})$
$\alpha_{res}^{l_1} = g(z_{res}^{l_1})$
short cut
main path

Figure 3.6: The structure and operation workflow of basic residual block

cidently by the workers. For different shakers, the preset camera angle might be different. If the manual region selection is going to be adopted, the same process, which is aforementioned, should be implemented several times which is time-consuming. In order to automate this procedure, we propose a Faster-RCNN [65] based ROI detection method to detect the region that contains the cutting flow. The procedure can be described in the Fig. 3.5.

As shown in Fig. 3.5, we take the raw video frame as input which is manually labeled with the region of interest by using bounding box. Every raw frame is fed into a feature extractor, which produces a feature map. The feature map is then fed into a much smaller convolutional neural network that takes feature map as input and outputs the region proposals. Those proposals are fed into a classifier which classifies those proposals to the background class or ROI class. If a region proposal is classified into the ROI class, its coordinate, width and height will be further adjusted by a region regressor.

Since this framework is neural networks based supervised learning, backpropagation is used to train the model. The first-stage feature extractor is the residual networks. The residual networks consists of a chain of residual blocks, which is shown in Fig. 3.6. In huge amount of deep learning model experiments, increasing the number layers not necessarily results in performance improvement, but sometimes opposite consequence, which is known as the degradation in training deeper networks. The proposal of residual networks solves the degradation problem in training. We take the first residual block as example, which is shown in Fig. 3.6. If we use a single image d as input (assume feeding one image at a time), $\alpha_{res}^{l_0} = d$. The output of the first layer $\alpha_{res}^{l_1} = g(z_{res}^{l_1})$, where $g(\cdot)$ is the activation function and $z_{res}^{l_1} = W_{res}^{l_1}\alpha_{res}^{l_0} + \beta_{res}^{l_1}$ represents the linear transformation of the input $\alpha_{res}^{l_1}$. Similarly, we have the output of the second layer $\alpha_{res}^{l_2} = g(z_{res}^{l_2})$, where $z_{res}^{l_1} = W_{res}^{l_1}\alpha_{res}^{l_0} + \beta_{res}^{l_1}$. The above in the mathematical representation of 2 layers transformation in the main path of residual block. In addition, we add a short cut which transports the input $\alpha_{res}^{l_0} = d$ directly to the point right before the activation function of the second layer. Thus, we have the short cut output to be expressed as $\alpha_{res}^{l_2} = g(z_{res}^{l_2} + \alpha_{res}^{l_0})$. During training, the information will automatically select to go through main path or the short cut, which helps the model to deeper without the accuracy saturation and degradation.

The object function (loss function) considers class classifier and bounding box regressor. The total loss is as follow:

$$J(\{q_i\}, \{b_i\}) = \frac{1}{N_{cls}} \sum\nolimits_i J_{cls}(q_i, q_i^*) + \lambda \frac{1}{N_{reg}} \sum\nolimits_i q_i^* J_{reg}(c_i, c_i^*). \tag{3.1}$$

In (3.1), q_i represents the predicted confidence of instance i to be an object, and q_i^* is the corresponding binary ground-truth value. If $q_i^* = 1$, the corresponding region proposal is the area containing cutting flow, otherwise it is the background. The first term in (3.1) is the total classification loss normalized over the number of classes, N_{cls}. In our problem, $N_{cls} = 1$, since we only consider one type of area that contains cuttings. Loss function J_{cls} is the log loss over two classes (region of cuttings, region of background). In the second term of (3.1), loss function J_{reg} is the smooth L_1 function. Multiplied by q_i^*, the second term represents the regression loss over the number of proposals only when the positive instance is activated. λ is the balancing parameter. Since $N_{cls} = 1$ as aforementioned, $\frac{\lambda}{N_reg}$ has to roughly equal to 1, if the classification loss and bounding box regression loss are equally treated. However, in experiment, the detection result is stable when the value of varies over a wide range [65]. For the bounding box regression, the parameterizations of 4

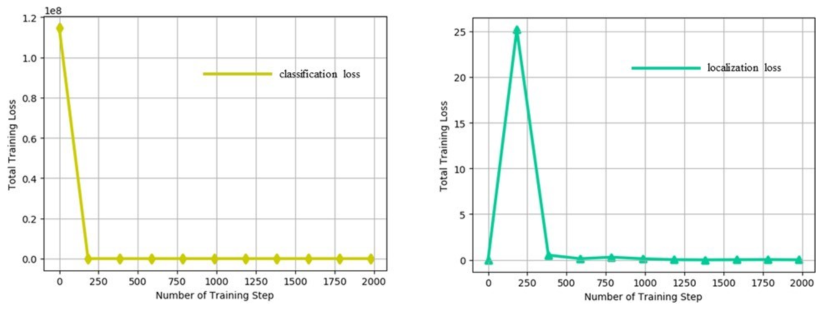


Figure 3.7: The loss convergence in training the Faster-RCNN

coordinates are adopted, which is shown as follows:

$$\begin{cases} c_a = \frac{a-a_{anc}}{w_{anc}}, c_b = \frac{b-b_{anc}}{h_{anc}}, \\ c_w = \log(\frac{w}{w_{anc}}), c_w = \log(\frac{h}{h_{anc}}), \\ c_a^* = \frac{a^*-a_{anc}}{w_{anc}}, c_b^* = \frac{b^*-b_{anc}}{h_{anc}}, \\ c_w^* = \log(\frac{w^*}{w_{anc}}), c_h^* = \log(\frac{h^*}{h_{anc}}), \end{cases} \tag{3.2}$$

where a, b, w, and h denote the bounding box's center coordinate and its width and height. Variables a, a_anc, and a^* are for predicted box, anchor box, and ground-truth box respectively (likewise for b, w, and h). The basic idea of this formulation is the regression of an anchor box to a nearby ground-truth box.

We used 50 vidoe frames for training and 4 images for testing. In Fig. 3.7, we show the results of training of cutting area detection. The loss of classification decreases and converges after around 1800 training steps. The loss of localization undergoes growth at very beginning of the training, but gradually decreases and converges at around 2000 training steps.

In Fig. 3.8, we show the results of region of interest detection. The bounding boxes contain the predicted region that covers the flow of cuttings. From the results, we observe that the machine predict the correct region with high confidence. The success of implementing region of interest detection brings the following benefits to the project: 1) the automation of the attention mechanism. For manual region selection, we have to develop an interactive GUI for the engineers to select the region of interest. The development of GUI is time-consuming, and engineers have to repeat the same operation every time they

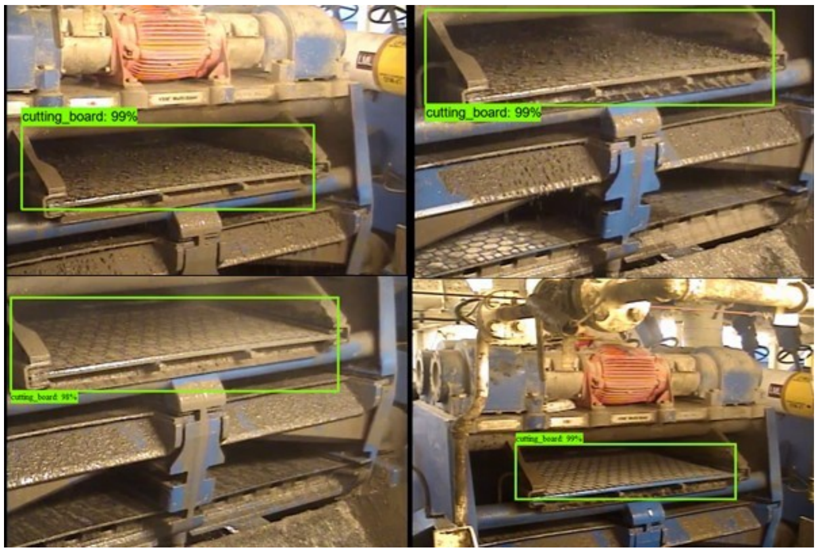


Figure 3.8: The visualization of inference result of cutting flow area detection

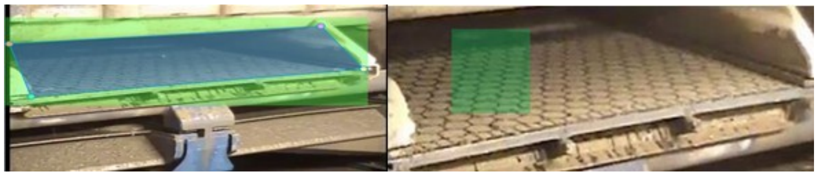


Figure 3.9: Randomized subsampling inside the ROI (the area covers the cutting flow)

start to record a new video stream; 2) the adaptation to different camera angle and distance. The environment of real operation on the rig can be rough and complex. The engineers might need to move the camera occasionally for the convenience of other operations with higher priority. The capability of adaptation to different camera angle and distance guarantees the stability of the model in constant attention to the region of interest and stable result inference.

3.2.3 Randomized Subsampling Inside Region of Interest

As mentioned in previous section, a ROI is selected either manually by the user at the beginning of the video stream as shown in the left panel of Fig. 3.9, or automatically by the cutting region detector based on Faster-RCNN framework. However, the vibration or the wind may nudge the camera's position and angle, which will compromise the classification performance if the system is trained without proper motion compensations. In this study, we propose a randomized subsampling strategy by using a stack of small image patches to overcome this problem. Image patches are densely sampled from the ROI. Instead of using the whole ROI as the input to the DNN, a stack of image patches is fed to the DNN. Either too much or too little training data can be troublesome for learning problem since meaningful model can only be learnt from sufficient large dataset yet these models tend to be intractable when the size grows. Subsampling strategy can be divided into two main paradigms: 1) active learning; 2) randomized subsampling [66]. Active learning can be regarded as a strategy that data collection process is actively guided. The objective of active learning is to bring a subset of the training dataset which is highly useful for the training task with low cost and an automatic experimental design. The experimental design consists of biological, physical, and social experiments, etc. Active learning can be used for either creating a dataset or compressing a large dataset. However, active learning is not always feasible in all scenarios. In our problem, data acquisition is through video streaming. The adjustment of the quality of video, angle and distance of the camera, and lightening quality in the environment is hard to be implemented timely, which makes the data collection necessarily passive. In this scenario, randomized subsampling is adopted as a degenerate form of active learning. On the contrary of active learning, randomized subsampling samples the original data in a randomized and unsupervised way, however, conserves the adequate properties of the dataset. In our implementation shown in Fig. 3.9, we used a squared window with much smaller scale to randomly slide over the original image and collect a fraction of the original dimensions. Intuitively, this approach increases the percentage of useful information in the data as well as the robustness to the variation of region selection either manually or automatically.

3.2.4 PCA Whitening Transformation

PCA whitening transformation is applied to video frames right before they are fed into the DNN. The goal of whitening is to make the input less redundant. We denote each video frame data as d. Since we obtain multiple frames in one second, we use D_t to denote a series of frames at time t, where $D_t = \{d_1, d_2, ..., d_I\}$. I is the total number of frames in one second. Every frame of the video is an image and can be represented as $d \in R^{W \times H}$, where W is the width, and H is the height. This whitening process consists of the following steps:

- We transfer the raw data to zero mean as follows:

$$d \leftarrow d - \frac{1}{WH} \cdot \sum d. \tag{3.3}$$

- We calculate the covariance matrix as follows:

$$\Sigma = Cov(d) = E\left[dd^T\right]. \tag{3.4}$$

  Then apply Eigen Decomposition to the covariance matrix as follows:

$$\Sigma = U \Lambda U^{-T}. \tag{3.5}$$

  Matrix U^{-T} defines as rotation needed to de-correlate the data d. By multiplying matrix U^{-T}, we can map the original principal components.

- Altogether, the whitening process can be formulated as follows:

$$W(d) = \Lambda^{-\frac{1}{2}} U^T (d - \frac{1}{WH} \cdot \sum d). \tag{3.6}$$

  The PCA whitening transformation removes the underlying correlations among adjacent frames. It potentially improves the convergence of the model.

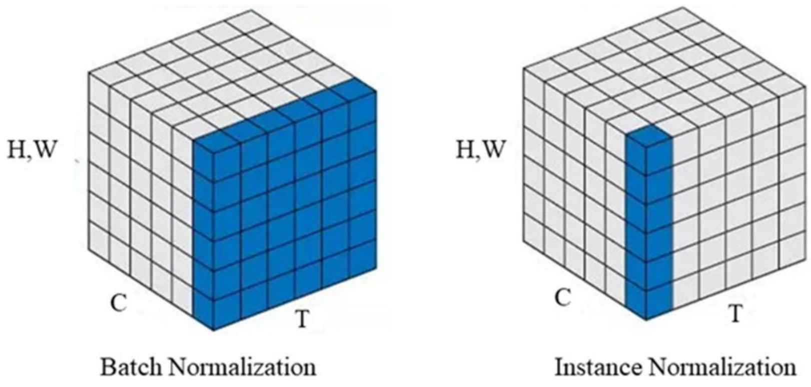


Figure 3.10: Batch Normalization vs. Instance Normalization

3.2.5 Instance Normalization

We implemented instance normalization [67] instead of batch normalization in this study. Normalization is a technique that will accelerate the training process and reduce covariate shift [68]. The mathematic representation of batch normalization is as follows:

$$\begin{cases} BN_l(d_{xy}) = \frac{d_{xy}-\mu_l}{\sqrt{\sigma_l^2+\varepsilon}}, \\ \mu_l = \frac{1}{HWT}\sum_{t=1}^{T}\sum_{x=1}^{W}\sum_{y=1}^{H} d_{txyl}, \\ \sigma_l^2 == \frac{1}{HWT}\sum_{t=1}^{T}\sum_{x=1}^{W}\sum_{y=1}^{H}(d_{txyl}-\mu_l)^2. \end{cases} \tag{3.7}$$

In (3.7), d_{xy} is the pixel value on the coordinate (x, y) of the frame d. l represents a specific channel dimension, where $l \in C$. T is the number of images in a batch. The batch normalization transformation $BN_l(d_{xy})$ normalizes samples across multiple images as well as the spatial locations. In most image classification tasks, the variance across the batch is high, e.g., a training batch may contain both the images of flowers and birds, which have different patterns. Without the batch normalization, the gradient from the small activations will be suppressed by the high activations. However, in our case, we collect the same type of training images (i.e., the cuttings) in the batch. The variance across the batch is low. By using the batch normalization, we might spread more unexpected noises across the batch. Instead, we adopt the instance normalization approach. The mathematic representation of instance normalization is as follows:

$$\begin{cases} IN_l(d_{xy}) = \frac{d_{xy}-\mu_l}{\sqrt{\sigma_l^2+\varepsilon}}, \\ \mu_l = \frac{1}{HW}\sum_{y=1}^{H}\sum_{x=1}^{W} d_{xyl}, \\ \sigma_l^2 == \frac{1}{HW}\sum_{y=1}^{H}\sum_{x=1}^{W} \left(d_{xyl}-\mu_l\right)^2. \end{cases} \tag{3.8}$$

Different from $BN_l(d_{xy})$ in (3.7), the instance normalization $IN_l(d_{xy})$ in (3.8) normalizes pixel values in each image independently. We add instance normalization to each layer before the activation function.

3.2.6 Adaptation of VGG-16 Networks

Combined with the techniques introduced in Section 3.2.1, 3.2.2, and 3.2.3, the DNN structure is shown in Fig. 3.4. The network takes whitened images $W(d)$ as the input. The input will go through two consecutive convolutional operations. Take the first convolutional computation as an example, the computation is formulated as

$$Conv_{x,y,l}^{(1)}\left(W\left(d\right)\right) = b_l + \sum_{i=1}^{f_h}\sum_{j=1}^{f_w}\sum_{l'=1}^{f_{C'}} W_{x',y',l'}(d)\cdot\theta_{i,j,l',l} \tag{3.9}$$

where $x^{'} = i\cdot s_h + f_h - 1$ and $y^{'} = j\cdot s_w + f_w - 1$. $Conv_{x,y,l}^{(1)}(W(d))$ is the output of the neuron located at row x, column y in feature map l. s_h and s_w are the vertical and horizontal strides, respectively. f_h and f_w are the height and width of the field, respectively. $f_{C'}$ is the number of feature maps in the previous layer. $W_{x',y',l'}(d)$ is the output of the neuron (at previous layer) located at row $x^{'}$, column $y^{'}$, and feature map $l^{'}$. b_l is the bias term for feature map l. $\theta_{i,j,l',l}$ is the connection weight between any neuron in feature map l of current layer and its input located at row i, column j and feature map $l^{'}$.

An instance normalization is applied after the convolutional operation, which is represented as $IN(Conv_{x,y,l}^{(1)}(W(d)))$. Then, the results goes through a nonlinear activation func-

Table 3.1: Definition of Symbols and Abbreviation.

Symbol or abbreviation	Definition
d	single frame of video fed into the neural networks
W	the width of the video frame d
H	the height of the video frame d
$g(\cdot)$	activation function in the neural networks
α^l_{res}	the output of the l layer after the activation function
z^l_{res}	the linear transformation of α^l_{res}
q_i	the predicted confidence of instance i to be an object
c_i	the coordinate related parameters for instance i
N_{cls}	number of classes to be detected
N_{reg}	number of regression boxes
$BN(\cdot)$	batch normalization
$IN(\cdot)$	instance normalization
$Conv(\cdot)$	convolutional operation in neural networks
$pool(\cdot)$	pooling operation in neural networks
CCD	charge coupled device
RSVP	rig-site virtual presence, the name for a video streaming service
GPU	graphics processing unit
DNNs	deep neural networks
CNNs	convolutional neural networks
VGG	a networks family developed by visual geometry group in University of Oxford
GOPs	giga operations per second
ROI	region of interest

tion. We utilize Rectified Linear Units (ReLu) [69] as the activation function. The output of the convolution layer is then denoted as $z^{(1)}(W(d)) = \max\left(IN\left(Conv^{(1)}_{x,y,l}(W(d))\right), 0\right)$.

Following the example of the VGG-16, in the proposed DNN, two consecutive convolutional layers are stacked together before a pooling layer. This structure helps the model to extract a deeper feature before the dimension of the tensor shrinks too fast (the pooling operation will exponentially reduce the dimension of tensors). We adopt max pooling after the two consecutive convolutional operations. The pooling operation is represented as follows:

$$pool^{(1)}(W(d)) = \max_{x,y \in K^{(1)}} z^{(2)}\left(z^{(1)}\left(W(d)\right)\right). \tag{3.10}$$

K represents the kernel. $z^{(2)}$ is the second convolutional operation, which is the same process as $z^{(1)}$.

Table 3.2: The inference time for the testing videos.

Video ID	Frame rate (fps)	Video length (s)	Time for label results (s)	Time for video inference (s)
***431-UTC_0	5.95	3607	226.6	272.5
***903-UTC_2	5.94	3606	227.3	269.1
***036-UTC_6	5.95	3609	227.5	279.5
***903-UTC_9	5.93	3603	221.6	273.4
***887-CST_1	29.67	3604	440.0	728.5
***746-CST_0	29.52	3601	436.2	724.2

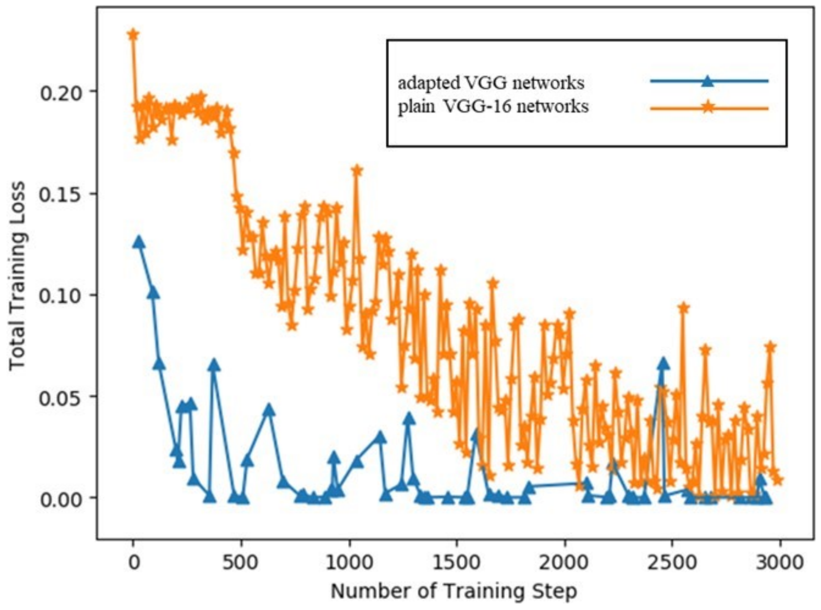


Figure 3.11: The training loss comparison of VGG netowrks and its adaptation

3.3 Experiment and Performance Evaluation

In this section, we evaluate the performance of our model according to the inference time for the classification workflow and accuracy of the cutting flow classification. Confusion matrix is used to quantitatively evaluate the classification accuracy for applying the adapted VGG network to different testing videos. All classification performance metrics are derived by comparing to manual annotations of domain experts.

In Table. 3.2, we list a portion of the testing videos, whose properties and time for inference are also demon- strated. The table mainly contains two kinds of videos, high resolution (frame rate around 30 fps) and low reso- lution (frame rate around 6 fps). The

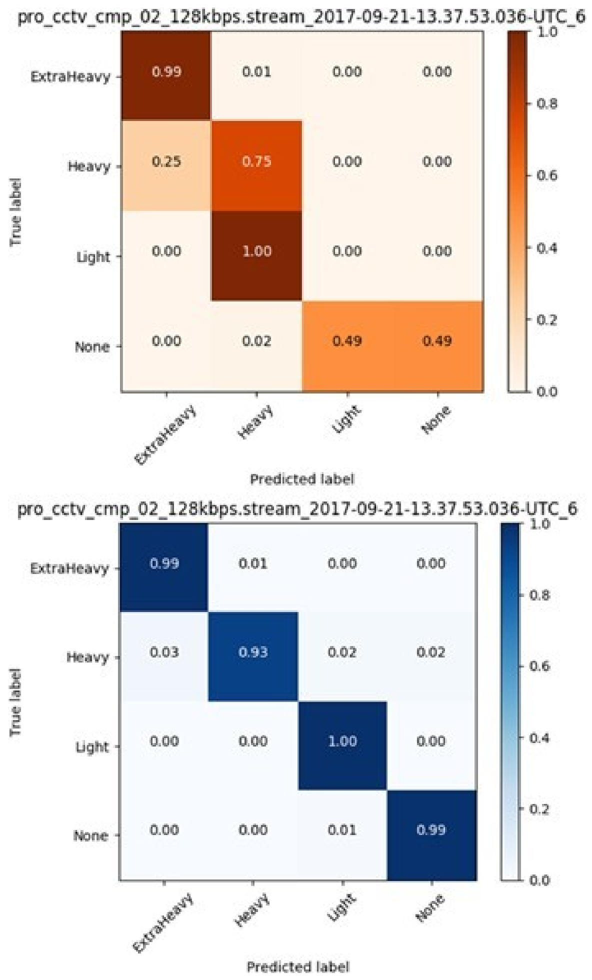


Figure 3.12: The evaluation comparison of VGG netowrks and its adaptation

length of each video is around 1 hour (3600 seconds). For one-hour high resolution videos, the time for obtaining classification result of all the frames is around 7 minutes (420 seconds). In other words, if we implement parallel inference with our model, we can process more than 8 video streams from 8 different cameras. The time for generating video files with labeled results is around 700 seconds for an one-hour stream. Likewise, the inference time for low resolution video is even less, since the frame rate is much lower compared to high resolution one. The time for label result is around 220 seconds, and 270 seconds for generating video inference.

In Fig. 3.11, we compare the training loss convergence between the plain VGG-16 networks and our proposed networks. The initial loss of the proposed networks is smaller than the plain VGG-16. At around step 500, the loss of proposed DNN already converges to a local minimum which is faster than the plain VGG-16.

To evaluate the performance, we test the proposed framework on a live video stream and compare the real- time classification results to the manual annotation. Following the criteria used by rig engineers monitoring the return cutting flow in real time, the cutting volume is classified into four discrete levels: ”ExtraHeavy”, ”Heavy”, ”Light”, and ”None”. Each video was labeled by four experts. And the ground truth labeling is the consensus among all experts. The testing results show the system can handle the live stream video without dropping frames. In order to visualize the quantitative results, we plot the confusion matrices as shown in Fig. 3.12. The left panel is the confusion matrix obtained by the proposed DNN. The right panel is the confusion matrix obtained by using the plain VGG-16. Each row in the matrix represents the ground truth label and each column represents the predicted label. Compared to the proposed DNN, the plain VGG-16 can recognize the case of “Extra Heavy”. However, it fails to differentiate “Light” and “None”. As the comparison, we observe that the proposed DNN successfully classify all classes. The testing result shows that our proposed model achieves a significant performance boost compared to the traditional VGG networks.

In Fig. 3.13, we demonstrate inference results from another three testing videos (each with around one hour). Each video contains different lighting condition, shooting angles and distances compared to others. Specifically, the first testing case shows a steady angle, constant shooting (with little interruption), and evenly distributed inference classes (labels). The second testing case shows that some videos might not contain a complete set of all the classes. In this case, ”light” and ”none” volume don’t appear in the video. In the third testing case, there is an uneven set of classes. The ”heavy” and ”light” classes appear very shortly so that in the transition of the states, our model made some mistakes, e.g., the state of ”none” lasts for several minutes while the state of ”light” appears several seconds in between. Our model has not achieved that high sensitivity to handle this case. Even there are some flaws in our model resulting some inaccuracy occasionally, the overall observation from the evaluation results is that the proposed workflow can adapt to different varieties of the input video, and achieve significantly good results.

3.4 Conclusions

In this chapter, a deep learning based workflow is proposed for classifying cutting volume on the shale shaker in real-time. With the help of automatic ROI selection and proper preprocessing, the system is able to analyze video streams captured from various angles and distances by an uncalibrated CCD camera. Additionally, the system is trained to accommodate different lighting sources as well as low-resolution video streams. The evaluation results validate the capability of the system in meeting the objective and delivering promising results in a real-time manner. The current work can be extended to quantify continuous cuttings volume after other auxiliary flow sensors are installed. For the future work, we will focus on improving the domain adaptation of the system by introducing transfer learning into the training stage so the system can be easily migrated to other video stream analysis tasks.

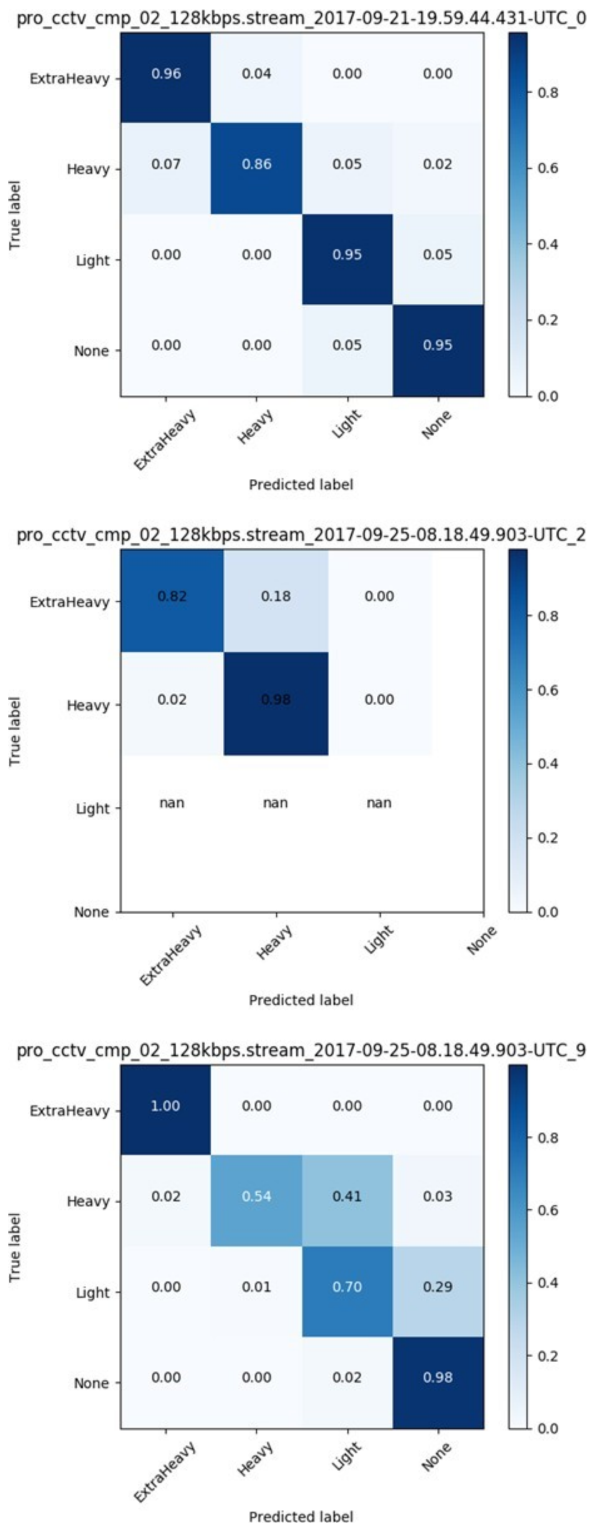


Figure 3.13: The confusion matrices for three cutting volume classification testing cases

Chapter 4

Buffelgrass Detection by Unmanned Aerial Vehicle Monitoring with High-Fidelity Data Augmentation by Vector Quantised Generative Model

Protecting the environment and minimizing the footprint of business have become increasingly important for all the global corporations who strives to become a responsible corporate citizen across the world. This becomes critical to the health of business not only from the economic perspective but also from the viewpoint of safeguarding the sustainable growth [70]. Among the diverse types of tasks in the field of environmental protection, monitoring the invasive plants, e.g., trees or grasses, contributes to an integrated approach for managing the Invasive Alien Species (IAS) for the natural environment where we live and conduct our daily business. In practice, surveying the land or water for IAS heavily relies on the field trip and manual labeling effort. This tends to be very costly and inefficient, sometimes even expose surveying people to dangerous environments. The advancement of computer vision and machine learning technologies, along with the availability of low-cost image collection methods, i.e., unmanned aerial vehicles (UAVs) enable the construction of an automatic invasive vegetation monitoring pipeline [71]. Specifically, as an invasive type of vegetation, the community of buffelgrass is detected using UAV in this work. Buffelgrass is an introduced, perennial pasture that is found across much of the Australian continent, including arid and semi-arid regions. It has been widely planted for livestock production and land rehabilitation. Meanwhile, it serves as an excellent pasture for the cattle industry for the property of fast growth. However, the growth of buffelgrass is out of control and poses a major threat to biodiversity in arid Australian areas such as Uluru-Kata Tjuta National Park, Cape Range National Park (WA) and Barrow Island (WA). Besides the weakened biodiversity, the invasion of buffelgrass also raises the economic costs through the potential demand for controlling fire risks. Due to all the potential threats, an effective approach of surveillance should be investigated. The high spatial and temporal variability of invasive grass infestations make a comprehensive ground sampling of large-range lands impractical. The utilization of unmanned aerial system (UAS) for weed and pest monitoring appears on the scene.

In literature, a drone-based estimation of vegetation fraction cover is developed in [72]. An efficient method of ground truth fractional vegetation cover (FVC) measurement using drone image is developed, which is further used for the development of a sigmoid model to measure FVC using Sentinel-2 data for a larger area. From the aspect of picture quality, the drone typically takes high-resolution images. The resolution can be up to 1cm/pixel. In [73], the effects of shadow correction on the classification of vegetation and land cover are investigated based on high-resolution aerial images. Authors prove that shadow correction can significantly improve classification results in vegetation mapping. Among different types of vegetation detection, weeds or grass detection is the most trivial and challenging task. A certain community of weeds can cover a huge area and grow rapidly, which leaves manual surveillance intractable. Apart from the target weeds, the background involves other types of grass as a distraction, which requires a more robust detection method. L. C. G. David et al. [74] propose a method to automatically map the land use in a vegetable farm with a very high-resolution aerial image (5cm/pixel) taken at an altitude of 100m. The soil is delineated from vegetation using the color index of vegetation extraction and Otsu's thresholding. Then Support Vector Machine (SVM) is implemented on various vegetation indices results for classification. More specifically, invasive weeds detection is studied in [75]. An XGBoost based image segmentation method is implemented to distinguish invasive weeds from other types of vegetation on high-resolution images captured by UAV.

The above literature demonstrates that machine learning and computer vision can be properly applied in the study of vegetation detection by UAV. In particular, the research in Geographic Object-Based Image Analysis (GEOBIA) has been sharply increasing [76]. Numerous experiments prove that both supervised and unsupervised detection algorithms are greatly influenced by the image quality, the spectral bands, the spatial resolution, the complexity of the scene, as well as the the number of labeled/marked images specially in the context of supervised learning. In [77], a task of localization of urban trees in multi-source (optical, infrared, DSM) aerial images is solved by managing data from multiple sources using CNNs. In [78], a large-scale and multi-class sample of oil palm tree data is collected. The detection is implemented via AlexNet-based DCNN training and optimization, sliding window-based label prediction, and post-processing. In order to detect small objects, deep features are needed to be extracted using novel neural networks structure. In [79], the use of deep features for the detection of small objects (cars and individual trees) in high-resolution Pleiades imagery is investigated.

Comparing to the study mentioned in [71], current work focuses more on the adoption of deep learning, one of the most promising recent developments in the field of machine learning. Deep learning not only distinguishes itself in terms of the modeling capability but also relieves the burden of extensive feature engineering required in the traditional machine learning workflow, which usually requires a lot of domain knowledge and sometimes ad-hoc analysis [80] [81]. This latter characteristic is especially useful when we try to build an automatic IAS monitoring solution, which needs to process diverse types of vegetations. Without using deep learning-based approaches, we may have to perform feature engineering for each single vegetation type, which significantly limits the generalization capability of the solution.

In the context of the studied problem, the objective is to recognize and localize alien vegetation plant from the vegetations which are natural to the monitored environment. When the studied information medium is in the format of images, collected from UAVs, manned vehicles (MAV) or satellite, the problem is referred to as object detection in the

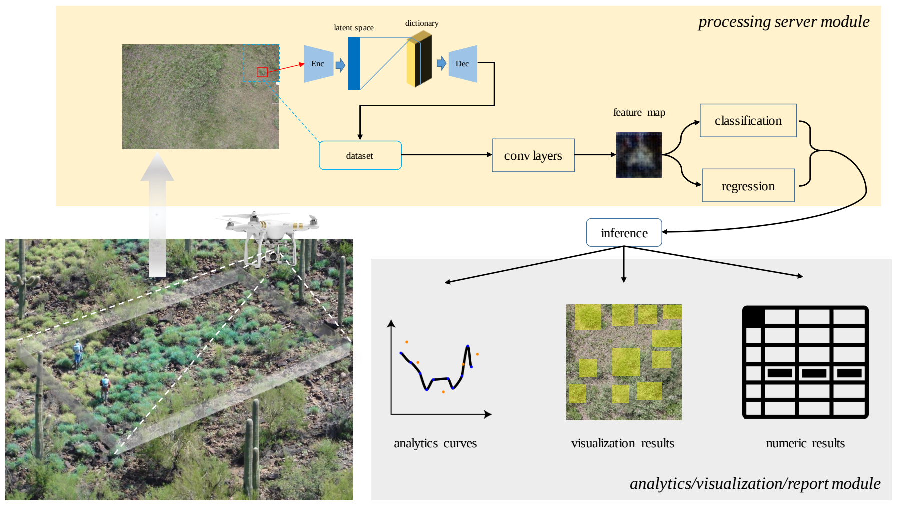


Figure 4.1: The system model of the UAV-based invasive vegetation monitoring.

field of computer vision, which has experienced very active research efforts, partially due to the resurgence of neural networks [82]. In an end-to-end object recognition pipeline, there are two sequential steps: 1) localization, i.e., identify locations that contain the studied object; 2) classification, i.e., categorize the localized object into appropriate class assuming that there are multiple classes of objects in the studied images. The deep learning-based object recognition has successfully combined these two steps into a single and trainable framework [83] [84] [85]. Due to the satisfactory performance and wide applications in multiple published studies [86] [87], we chose to use CNN-based detection framework [84] as the building block for our study. We adopt deep residual networks [88] to achieve insightful feature representation. Meanwhile, due to the limitation of the UAV payload, the dataset we obtain is too small to train with state-of-the-art deep detection models. Thus, the great necessity is that buffelgrass images should be well preprocessed and augmented. In order to achieve a better detection performance, we propose a generative model based data augmentation framework along with the inductive transfer learning. GAN [4] is regarded as the first deep model being capable of generating data instead of only discrimination such as classification and regression. The emerging of GAN prompts the development of data augmentation by using GAN to generate synthetic data [89] [90].

In this chapter, we propose a variational version of the generative model for buffelgrass pattern augmentation and show its effect on the performance of detection. The main contributions of this chapter is summarized as follows:

We come up a deep learning based invasive vegetiation-montiroing workflow based on UAS-collected images. The proposed workflow enables the automaic detection capability.

- In order to detect the distribution of buffelgrass on the far-flung pasture, we adopt the novel deep learning based object detection technique to draw bounding boxes on the location where we infer the existence of the buffelgrass. In order to encourage the convergence of training detection model, we adopt inductive transfer learning.
- In practice, the amount of collected buffelgrass patches is limited, which poses significant challenge to the adoption of deep learning or other supervised machine learning technologies. In our proposed pipeline, we develop a model-driven data augmentation strategy to systematically boost the buffelgrass patterns.

The chapter structure is as follows. In Section 4.1, we describe the network architecture as the background and explain the transfer learning data augmentation approach being pursued in this research, which combines the theoretical foundation and engineering approach to address the practical issue of a limited number of labeled training data. We report the simulation process and discuss the result observed along with the experimental study in Section 4.2. We conclude this paper with discussion in Section 4.3.

4.1 Methods

In this study, we implement a CNN-based deep learning detection model to pick out the buffelgrass patterns in the natural grasslands, which we are trying to monitor and protect. This section focuses on describing the model building pipeline. In Section 4.1.1, we

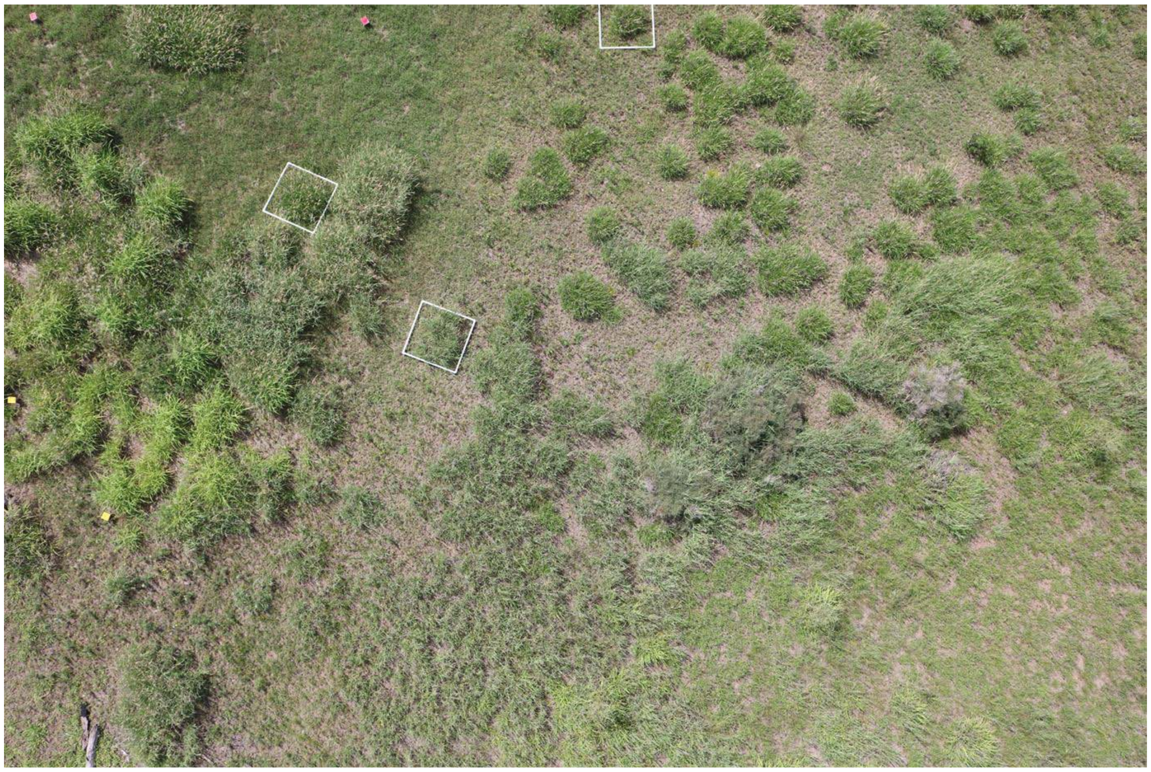


Figure 4.2: A sample of raw image taken by UAV.

discussed the step of data pre-processing and traditional manual data augmentation, used for increasing the data size and the variances among the training data set. In Section 4.1.2, we introduced a novel CNN-based generative model for high-fidelity buffelgrass pattern synthesis instead of using traditional data augmentation approaches. As another corner stone to compensate for the lack of training data and long training time, we utilized the model initialization strategy, which is based on the theory of transfer learning [91], in Section 4.1.3. Finally, in Section 4.1.4, we discuss the protocol and configuration adopted in the context of the CNN-based detection [84].

4.1.1 Data Description and Manual Augmentation

In the study, the training set is created by sampling from raw images captured by the UAVs (referring to Fig. 4.2). The raw images are of size $8,000 \times 5,000$. Due to the

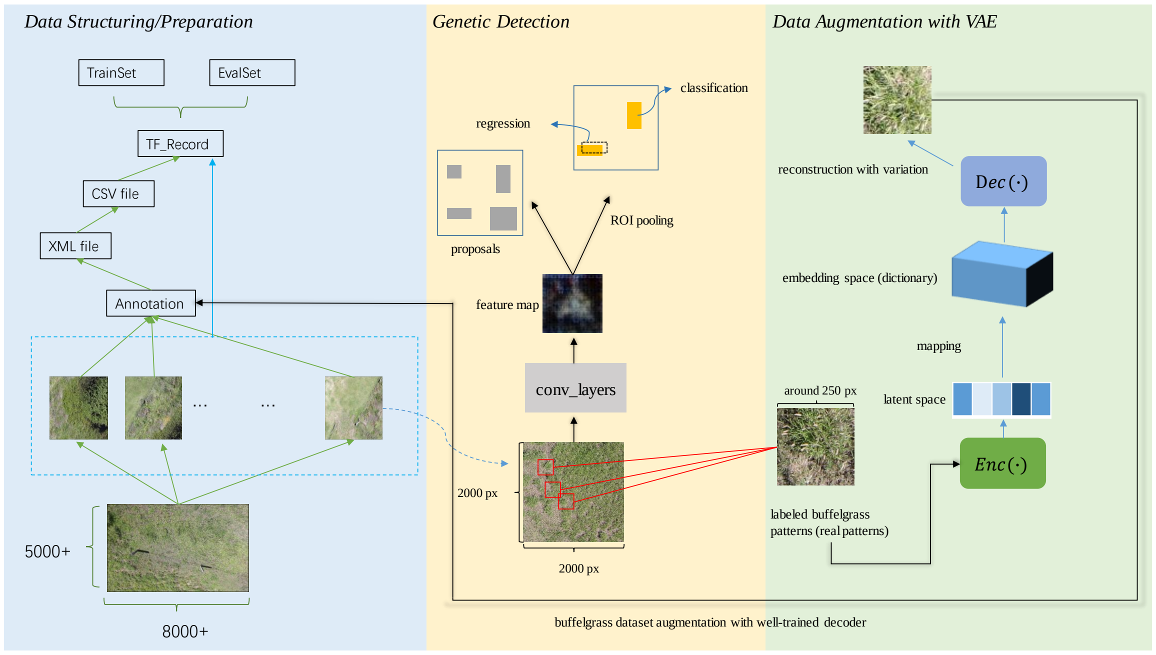

Data Structuring/Preparation
TrainSet
EvalSet
TF_Record
CSV file
XML file
Annotation
5000+
8000+
Genetic Detection
classification
regression
proposals
ROI pooling
feature map
conv_layers
2000 px
2000 px
Data Augmentation with VAE
reconstruction with variation
Dec(·)
embedding space (dictionary)
mapping
around 250 px
latent space
Enc(·)
labeled buffelgrass patterns (real patterns)
buffelgrass dataset augmentation with well-trained decoder

Figure 4.3: The overall workflow of the buffelgrass detection.

GPU-related limitation on the size of images feed into the model, we draw image patches of size $2,000 \times 2,000$ from the raw image using a randomly sampling process. In practice, the random sampling strategy can segment some existing buffelgrass patterns into different image patches, which indirectly create new types of combinations of buffelgrass patterns and the surrounding environment. This can result a more diverse training data set, and helps to increase the model robustness [92].

The buffelgrass pattern augmentation involves two steps.

- we adopt a few manual augmentation techniques in order to increase the variance of data samples before augmentation by machine.
- several generative models based on deep networks are utilized to create a more diversed training data set being able to capture more modes in the underlying distribution and creating much more trainable data from a random space.

First of all, on top of the sampled image patches, we adopt the following manual data augmentation processes to enhance the diversity of training data set. This is not only necessary for increasing the number of training samples for generative models in the next step, but also increasing the variances among the training data set.

- Randomly flipping and rotation: In reality, buffelgrass may grow towards different directions, the images taken from UAVs is not able to capture all of the possible angle variances. We randomly choose some buffelgrass patterns within the bounding boxes to flip or rotate, which implicitly simulate the UAV flight path from different angles.
- Randomly alter the contrast of the patterns: The daylight condition varies for each UAV flight. To let the model adapt to different lighting of the patterns, we randomly choose some patterns and alter its contrast.
- Manual pattern augmentation: Compared with the vast grassland, the scale of the buffelgrass flora is much smaller. This is referred to data set imbalance, where the size of one class of the data set, i.e., the area of buffelgrass flora, is much smaller than the size of background grassland. To boost the occurrence of the buffelgrass pattern in the training data set, we manually crop some patterns from the raw images and randomly sample them on the raw images which only have background.

4.1.2 Generate Synthetic Buffelgrass Pattern

As mentioned in the last subsection, in order to generate more buffelgrass patterns with higher variance, we randomly flip, rotate, then copy and paste those patterns to some existing backgrounds without buffelgrass. These methods are straight-forward and effective, but require significant human labor with only basic layout alternation. Motivated by this drawback, we develop a model-driven approach that can generate a diverse typos for grass patterns. Specifically, we leverage the generative neural network model.

Being one of the generative models, autoencoder can reconstruct data from encoded prior learned from the existing data sets [93]. It suffers from lower variance [94], which is the reason that autoencoder is rather used for compressing data into a lower dimension form than utilized as a generator for synthetic data generation. To remedy this drawback, a variational version of autoencoder (VAE) is proposed. The goal of this structure is to construct a distribution of latent space rather than a fixed feature.

On the contrary, a GAN generates synthetic data from a random latent space. By doing so, synthetic data with more variety is generated. Besides variety of the generation, GAN provides an approach to measure the difference between the real and fake data. Instead of using conceived loss function, e.g., L2 loss, which is the most common case, the critic is a neural network which works as a loss function to measure the difference between two high-dimension signals. However, GAN tends to suffer from model collapse [95], especially for generator. The visualization performance of the generator is thus limited. Intuitively, more prior information (the perfect prior is the encoded real input) is needed rather than a random space, which encourages us to combine the VAE and GAN.

The concept of variational autoencoder is first proposed by Anders et al. in [96]. Instead of a previous fixed latent space used in traditional autoencoder, variational version of autoencoder imposes a prior (typically a normal distribution) to present a variational and continuous distribution of latent code. However, this continuous type of latent code is static (a certain distribution, e.g., normal distribution), which will probably arise the problem of posterior collapse [97], where the latents are ignored when they are paired with a powerful autoregressive decoder. Referring to the discrete representation of latent space which is discussed in [98], we present a vector quantisation of latent space representation which is learnt for synthetic buffelgrass data generation.

Starting from general autoencoder formulation, we denote the latent space z generated by buffelgrass sample which is denoted as x_{grass}. The encoder and decoder are denoted as $Enc(\cdot)$ and $Dec(\cdot)$, respectively. The encoding and decoding process is denoted as

$$\begin{aligned} z &\sim Enc(x_{grass}) = q(z|x_{grass}), \\ \tilde{x}_{grass} &\sim Dec(z) = p(x_{grass}|z). \end{aligned} \tag{4.1}$$

Assume the dimensionality of the output of the encoder is D, which is denoted as $z \in \mathbb{R}^D$. As the basic idea of variational autoencoder, there is a manipulation of latent space required to obtain variance in representation. In vanilla variational autoencoder, the prior of the latent space is assumed to follow normal distribution. The encoder (represented as neural networks) will generate two internal layers representing the mean μ and variance σ^2, respectively. The variance of the latent space is achieved by sampling from a normal distribution with mean μ and variance σ^2. The challenge is that the sampling operation is not derivable, which make the gradient decent not feasible in training the networks. To overcome this challenge, [99] proposed a reparametrization method to sample from normal distribution.

However, the vanilla variational autoencoder imposes an normal distributed prior which is continuous and constant. In order to construct a discrete and learnt distribution for latent space representation, we adopt the idea of quantisation of the latent vectors, which is also a form of dictionary learning [100]. We define a latent space $\mathbf{e} \in \mathbb{R}^{K \times D}$, where K is the size of the discrete latent space and D is the dimensionality of each embedding vector e_i. The latent space (dictionary) $\mathbf{e}$ is denoted as

$$\mathbf{e} = [e_1, e_2, ..., e_K]. \tag{4.2}$$

The output of the encoder follows a continuous distribution. In order to convert the continuous variables to a discrete representation in the dictionary $\mathbf{e}$, a mapping strategy is defined to find the nearest vector in the dictionary measured by Euclidean distance:

$$z \to e_K, \; K = \arg\min_j \|z - e_j\|_2. \tag{4.3}$$

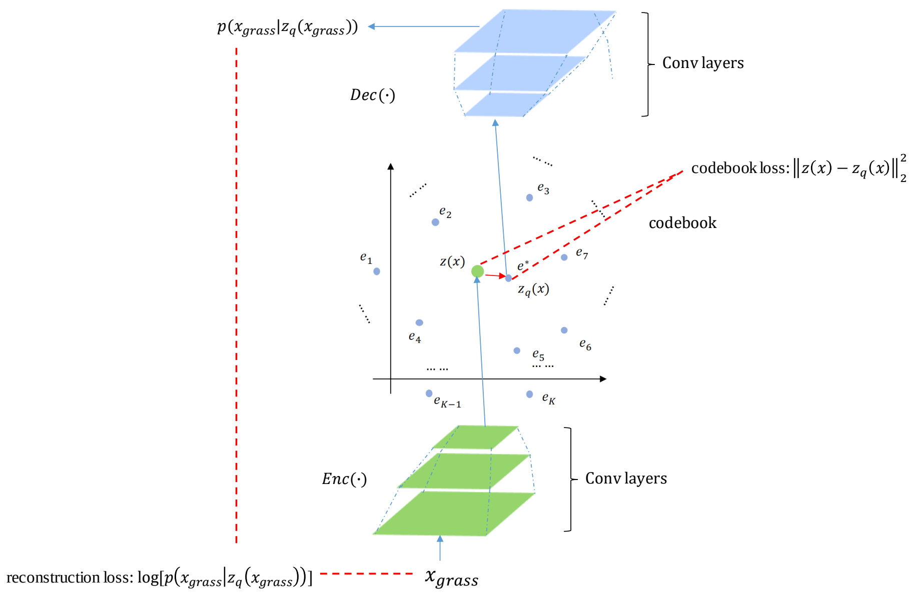


Figure 4.4: The structure of the vector quantised variational autoencoder.

The first term in (4.1) can be replaced by

$$q(z=e_K|x_{grass}) = \begin{cases} 1, & K = \arg\min_j \|z-e_j\|_2 \\ 0, & otherwise \end{cases}. \tag{4.4}$$

After mapping, we get the corresponding vector which is from the dictionary denoted as z_q. To preserve the location correlation in the image, we use convolutional neural networks to extract a feature map. Thus, z and z_q are two dimensional matrix instead of a vector, which is denoted as

$$z = \begin{pmatrix} z_{11} & z_{12} & \cdots & z_{1m} \\ z_{21} & z_{22} & \cdots & z_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ z_{m1} & z_{m2} & \cdots & z_{mm} \end{pmatrix} \rightarrow$$
$$z_q = \begin{pmatrix} z^q_{11} & z^q_{12} & \cdots & z^q_{1m} \\ z^q_{21} & z^q_{22} & \cdots & z^q_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ z^q_{m1} & z^q_{m2} & \cdots & z^q_{mm} \end{pmatrix}, \tag{4.5}$$

where $m \times m$ is the size of the latent space, and $D = m \times m$.

In the last section, we mentioned the challenge of VAE-based generative model lies in the non-differentiable property of the sampling layer. The vanilla VAE utilizes the reparametrization strategy to overcome this obstacle by making using of the property of normal distribution. In autoencoder or even vallina VAE, the loss can be simply denoted as the following equation:

$$\|x_{grass} - Dec(z)\|_2^2 , \tag{4.6}$$

which directly measures the Euclidean distance between the real input grass pattern with the reconstructed patterns (the output of the decoder). However, in vector quantisation VAE, the output of the model is reconstructed from the vector z_q in the dictionary **e**. Thus, the actual loss we consider should be

$$\|x_{grass} - Dec(z_q)\|_2^2 , \tag{4.7}$$

instead. However, in (4.7), the mapping $z \rightarrow z_q$ is non-differentiable. In other words, ∇z_q cannot be propagated to ∇z during training, and at the same time $z = Enc(x_grass)$, which means the encoder $Enc(\cdot)$ is untrainable. A quick alternative for (4.7) is its combination with (4.6), which is represented as

$$\|x_{grass} - Dec(z)\|_2^2 + \|x_{grass} - Dec(z_q)\|_2^2 . \tag{4.8}$$

In this way, the gradients can be propagated back into encoder, however an unnecessary objective is included in the loss, which is not part of our training goal. To get rid of the redundancy in the loss, and at the same time let the gradient pass through, we adopt the idea of straight-through estimator, which is proposed in [101]. This idea can be considered as a conditional propagation that gradients are counted during forward propagation but ignored during backward propagation (or be designed to other form to meet the goal). We replace the (4.8) with the following one:

$$\|x_{grass} - Dec(z + sg[z_q - z])\|_2^2 , \tag{4.9}$$

where $sg[\cdot]$ is the stop gradient function. $sg[z_q - z]$ can be denoted as

$$sg[z_q - z] = \begin{cases} z_q - z, & \text{forward pass} \\ 0, & \text{back pass} \end{cases} . \tag{4.10}$$

By applying (4.10), (4.9) can be represented as conditional equation:

$$L_{VAE} = \begin{cases} \|x_{grass} - Dec(z_q)\|_2^2, & \text{forward pass} \\ \|x_{grass} - Dec(z)\|_2^2, & \text{back pass} \end{cases} . \tag{4.11}$$

In (4.11), we only consider the reconstruction loss. However, we also want to learn the dictionary **e**, or in other words $\|z - z_q\|_2^2$ should be small. (4.11) can be upgraded to

$$\|x_{grass} - Dec(z + sg[z_q - z])\|_2^2 + \beta \|z - z_q\|_2^2 , \tag{4.12}$$

where β determines the percentage of the similarity between z and z_q counted in the total loss. More specifically, z will determine the quality of the reconstruction, and z_q is relatively more adaptable since there are plenty of vectors in **e** which is potential candidate for the discrete representation. Intuitively, we want a good generalization of the mapping in (4.5) such that for any latent matrix z (from the same latent distribution), we can obtain a discrete representation z_q which is sufficiently good to reconstruct the input. Thus, to get high-fidelity reconstruction, the latent space is relatively fixed while the embedding space (discrete representation) is more adaptable. In order words, z_q should be closer to z rather than z closer to z_q. Since the gradient of $\|z - z_q\|_2^2$ equals to the gradient of z plus the gradient of z_q, $\|z - z_q\|_2^2$ can be decomposed and transformed to the following equation

$$\|z - sg[z_q]\|_2^2 + \|sg[z] - z_q\|_2^2 . \tag{4.13}$$

We assign larger weight to $\|sg[z] - z_q\|_2^2$ instead of $\|z - sg[z_q]\|_2^2$ in (4.13). In this way, in the phase of backpropagation, the training of z_q will be put more emphasis on rather than z. By replacing the term $\beta \|z - z_q\|_2^2$ in (4.12), we get the final version of the loss equation

$$\begin{aligned} \mathcal{L}_{VAE} = & \|x_{grass} - Dec(z + sg[z_q - z])\|_2^2 + \beta \|sg[z] - z_q\|_2^2 \\ & + \gamma \|z - sg[z_q]\|_2^2 , \end{aligned} \tag{4.14}$$

where $0<\gamma<\beta$.

While training with the overall loss $\mathcal{L}_{VAE}$, the embedding space **e** is well-maintained and encoder/decoder is trained for properly generating latent space z and reconstruction $p(x_{grass}|z_q(x_{grass}))$. In order to achieve the diversity of the latent space z, an autoregressive model is used to model the entry distribution of the latent space. As aforementioned, latent space z has the size of $m \times m$, where m is much smaller than n. The latent space z can be represented by a joint probability density $p(z)$:

$$p(z) = p(z_1, z_2, ..., z_{m^2}). \tag{4.15}$$

Assume we arbitrarily start from an entry in the set $\{z_1, z_2, ..., z_{m^2}\}$ and regressively generate the next entry by using the chain model, the (4.15) can be converted to a conditional density distribution:

$$p(z) = p(z_1)p(z_2|z_1)...p(z_{m^2}|z_1, z_2, ..., z_{m^2-1}). \tag{4.16}$$

In the experiment, the above autoregressive model is implemented by PixelCNN [102]. Since the latent space is relatively small, the kernel we use can be much smaller, and the small kernel can cover all previous entries (probability density), which is shown in (4.16). Meanwhile, latent space is the output of convolutional operation by the encoder. The local correlation of different entry can be preserved. The advantage of fast training of PixelCNN, long dependency of autoregressive model, and the preservation of the relative position of entries can be achieved at the same time.

To summarize, the training and generation of the synthetic buffelgrass images can be divided into two phases. In phase I, the VAE-related components will be trained. The gradients of encoder, decoder, latent space and quantised space will be updated in each iteration of training, which is shown in Alg. 2. In phase II, the prior to the latent space will be learnt. The latent space would be depended on the input image set. The variance of the latent space can be created by fitting an autoregressive model to learn the distribution and sampling from the imitation distribution. The learnt latent space can be used to generate

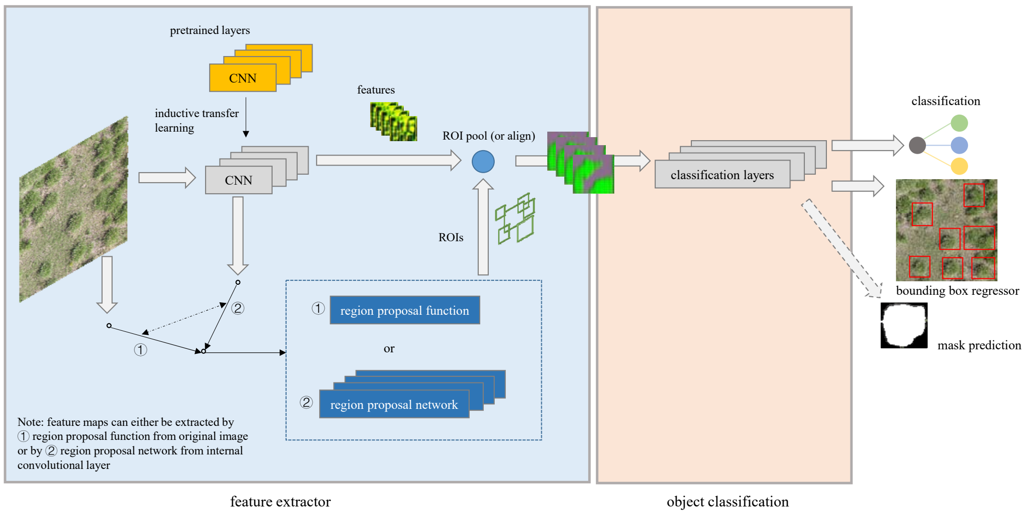

pretrained layers
CNN
inductive transfer learning
features
CNN
ROI pool (or align)
ROIs
classification layers
classification
bounding box regressor
mask prediction
① region proposal function
or
② region proposal network
Note: feature maps can either be extracted by ① region proposal function from original image or by ② region proposal network from internal convolutional layer
feature extractor
object classification

Figure 4.5: Object detection framework for detecting the buffelgrass.

synthetic buffelgrass images through the learnt codebook $\hat{\mathbf{e}}$. This phase is shown in Alg. 3.

Algorithm 2: Training of the Vector Quantised VAE.

```
1: Initialize: The encoder networks $Enc(\cdot)$, the decoder networks $Dec(\cdot)$, the input
   image $x_{grass}$
2: Obtain the latent space $z$ by encoder networks:
   $z \leftarrow Enc(x_{grass})$
3: Quantize the latent space $z$ with the codebook (dictionary) $e$:
   $z_q \leftarrow Quantize(z)$
4: Obtain the reconstructed grass $\hat{x}_{grass}$:
   $\hat{x}_{grass} \leftarrow Dec(z_q)$
5: Compute overall loss according to (4.14), update gradients:
   $\nabla z, \nabla z_q, \nabla \theta_{Enc}, \nabla \theta_{Dec}$
```

4.1.3 Inductive Transfer Learning

The deep learning-based object detection has achieved state-of-the-art performance in multiple studies reported recently [103]. However, adopting these models in real industry application have suffered from the lacking a sufficient number of labeled training data set. In this study, we follow the inductive transfer learning approach, which enables us to leverage the previous trained model as the foundation on which we fine tune with limited

Algorithm 3: Training of the prior and synthetic image inference

```
1: Initialize: The trained latent space $\hat{z}$ and trained codebook $\hat{\mathbf{e}}$
2: Quantize the latent space with the trained codebook $\hat{\mathbf{e}}$:
   $z_q = Quantize^*(\hat{z})$
3: Learn the quantized space using PixelCNN:
   $\hat{z}_q = TrainPixelCNN(z_q)$
4: Sampling from the probability density $\hat{z}_q$
5: Generate synthetic buffelgrass image by the trained decoder $Dec^*(\cdot)$:
   $x_{grass}^{fake} \leftarrow Dec^*(\hat{z}_q)$
```

data set in our study [104] [105], as shown in Fig. 4.5.

In general learning scenario, we denote domain as $\mathcal{D}$, feature space as χ and marginal probability distribution $P(\mathcal{I})$. $\mathcal{I}$ represents a data example from the domain χ and consists of n feature dimensions, where $\mathcal{I} = \{i_1, i_2, ..., i_n\} \in \chi$.

For a given specific domain $\mathcal{D} = \{\chi, P(\mathcal{I})\}$, a task $T = \{\mathcal{O}, f(\cdot)\}$ consists of two components: a label space $\mathcal{O}$ and an objective predictive function $f(\cdot)$. The function $f(\cdot)$ can be learnt by feeding the training data $\{i_j, o_j\}$, where $i_j \in \mathcal{I}$ and $o_j \in$ O. The prediction $f(i) = P(o|i)$ is the output of the model by feeding a data sample i.

Based on the above definition of the general learning problem [91], we defined our inductive transfer learning as follows: Given a source domain $\mathcal{D}_S$ (we used the COCO dataset [106] in our case study) and learning task T_S (object detection or segmentation tasks), a target domain $\mathcal{D}_G$ (buffelgrass dataset)and learning task T_G (object detection task), the inductive transfer learning aims to help the learning process of the target predictive function $f_T(\cdot)$ in $\mathcal{D}_G$ using the prior experience in $\mathcal{D}_S$ and T_S, where $\mathcal{D}_S \neq \mathcal{D}_G$ and $T_S = T_G$.

Basically, utilizing the pre-trained parameters as a model initialization strategy allows faster convergence of the model and finds a better local minimum point which has a smaller training loss.

4.1.4 Protocol for Buffelgrass Detection

A generic deep learning-based detection solution usually consists of three key links:

- CNN-based feature extraction layers;
- region proposal networks for mapping feature maps to ROIs;
- detection layers for both classifying an object into a certain class (in this problem, only one class, i.e., buffelgrass) and adjusting the bounding boxes' size and position.

The CNN structure adopted here (referring to the "CNN" box in Fig. 4.5) is ResNet 101 [88]. ResNet, as a feature extractor, can learn rich feature representations for a wide range of images.

In order to learn a richer feature representation of buffelgrass, a common practice is to stack as many layers as we can, intuitively. However, in large-scale experiments, the more layers a plain CNN has, the more likely that it will suffer the degradation problem [88] when the training curve starts converging. ResNet family solves the degradation problem by simplifying several layers of networks processing and converting it to a direct mapping of $x \rightarrow y$ with a function $H(x)$ represented by a few stacked non-linear layers. ResNet defines the residual function using $F(x) = H(x)–x$, which can be reframed into $H(x) = F(x) + x$. $F(x)$ denotes the stacked layers except for the input layer and x denotes the identity function (the input equals to the output). The intuition of this design is that: if the identity mapping is optimal, it is much easier to put the residual function $F(x) = 0$ than to fit an identity mapping x by a stack of non-linear layers. The testing of this design proves to be effective when stacking a huge number of layers [88]. By adopting the ResNet 101 structure as the feature extractor, we stack enough number of layers in order to learn a sufficient representation of buffelgrass patterns.

The extracted feature maps will be mapped to the ROIs by the region proposal networks. The ROIs mark the positions on the feature maps where we assume there is the object we want to detect. There are two key concepts involved in this proposal process. Firstly, we arbitrarily draw some proposal boxes around a centroid to assume it contains the object, which is called the anchor. We use four scales and three ratios of aspect to generate the anchor. Thus, for each centroid, we have $4 \times 3 = 12$ anchors. We set the centroid on the feature maps with the stride of 16 both vertically and horizontally. Secondly, we want to determine the ROIs by comparing the overlapping of the anchors and annotated bounding boxes. In our experiment, we set the positive samples with IoU > 0.7 (intersection-over-union [84]), while the negative samples with IoU < 0.3.

The output of the region proposal networks will be used as the input of a fully-connected neural networks, which will predict object class (classification), adjust the position and size of bounding boxes (regression) as well as generate mask for the object inside each bounding box which covers the area of the object in pixel level (instance segmentation). For the final loss function of this fully-connected networks, we set the localization loss weight larger than the objectness loss weight as we care more about the location of the potential invasive grass than the correctness of the classification.

In the experiment, we find it costly to focus on judging the positivity or negativity of potential vegetation since there are many similarities between the invasive grass and the native grass. Thus, paying more attention to the location of the potential vegetation will be more effective and meaningful to the inference of the model. Based on the property observed in our experiment, we put more emphasis on tuning the location of the potential vegetation (even some of the boxes may contain the negative object) while the impact of detection missing is minor and constant over different detection configuration.

4.2 Experimental Results and Discussion

In this section, we are going to present the experimental results in two phases. Firstly, the numeric and visualization results of synthetic buffelgrass pattern generation are pre-

Table 4.1: Networks Structure of Vanilla GAN.

Discriminator Networks			**Generator Networks**		
Layer	Filter/Stride	Output Size	Layer	Filter/Stride	Output Size
Conv1	$5 \times 5/2 \times 2$	$128 \times 128 \times 16$	fully-connected1	—	$8 \times 8 \times 256$
Conv2	$5 \times 5/2 \times 2$	$64 \times 64 \times 32$	Conv-Trans1	$3 \times 3/2 \times 2$	$16 \times 16 \times 128$
Conv3	$5 \times 5/2 \times 2$	$32 \times 32 \times 64$	Conv-Trans2	$3 \times 3/2 \times 2$	$32 \times 32 \times 64$
fully-connected1	—	500	Conv-Trans3	$3 \times 3/2 \times 2$	$64 \times 64 \times 32$
fully-connected2	—	100	Conv-Trans4	$3 \times 3/2 \times 2$	$128 \times 128 \times 16$
fully-connected3	—	1	Conv-Trans5	$3 \times 3/2 \times 2$	$256 \times 256 \times 3$

Table 4.2: Networks Structure of W-GAN.

Critic Networks			**Generator Networks**		
Layer	Filter/Stride	Output Size	Layer	Filter/Stride	Output Size
Conv1	$5 \times 5/2 \times 2$	$128 \times 128 \times 128$	fully-connected1	—	$32 \times 32 \times 512$
Conv2	$5 \times 5/2 \times 2$	$64 \times 64 \times 256$	Conv-Trans1	$5 \times 5/2 \times 2$	$64 \times 64 \times 256$
Conv3	$5 \times 5/2 \times 2$	$32 \times 32 \times 512$	Conv-Trans2	$5 \times 5/2 \times 2$	$128 \times 128 \times 128$
Conv4	$4 \times 4/1 \times 1$	$32 \times 32 \times 1$	Conv-Trans3	$5 \times 5/2 \times 2$	$256 \times 256 \times 3$

sented. In this part, we demonstrate how the discrete latent space model (VQ-VAE) is superior over both implicit model (GAN family model) and explicit but continuous latent space model (vanilla VAE). Secondly, we blend the synthetic buffelgrass patterns into the background without buffelgrass so that we automatically “grow” it on widely spread grassland, which significantly increases the volume and quality of the available training data set. Based on augmented data set, we evaluate how much it will help to enhance the detection performance.

4.2.1 Results for Synthetic Buffelgrass and Augmented Data Set Generation

The training curves of different generative models on the buffelgrass data set are shown in Fig. 4.6. Since the objectives of different models differ from each other, their respective training loss is not comparable. However, the convergence of the training curves demonstrate the progress of training. For Vanilla GAN, neither discriminator loss nor gen-

Table 4.3: Networks Structure of VQ-VAE (Encoder & Decoder).

Input Size	256×256
Latent Layer	32×32
β (commitment loss coefficient)	0.25
Batch Size	10
Hidden Units	128
Residual Units	64
Layers	2
Codebook Size	512
Codebook Dimension	64
Encoder Conv Filter Size	3
Upsampling Conv Filter Size	4

erator loss is close to convergence. The curves fluctuate strongly during the training. For W-GAN, the discriminator loss fluctuates in a reasonable range, however the curve for generator loss is smooth yet hard to converge within a satisfying range. For VQ-VAE, two curves are plotted:

- First, the training loss (reconstruction loss) has a smooth convergence and can be less than 0.1 after convergence.
- Second, the average codebook usage (perplexity) is plotted.

As aforementioned, the objective of VQ-VAE contains the reconstruction performance and the maintenance of the codebook. The more codebook usage, the higher the perplexity, which is a term to indicate how easy the prediction can be made by the model. We can observe that perplexity curve converge and reach a high level.

To validate the performance of different generative models, we adopt the Fréchet inception distance [107] to measure the distance between the synthetic images with the real images. For real image batch x_{grass} and fake image batch x_{grass}^{fake}, we feed both into Inception Networks [108] and obtain the activations of an intermediate layer, which are denoted as $Incep(x_{grass})$ and $Incep(x_{grass}^{fake})$, respectively. Both activations are modeled as multivariate Gaussian distribution, where $Incept(x_{grass}) \sim \mathcal{N}(\mu_{grass}, \Sigma_{grass})$ and $Incept(x_{grass}^{fake}) \sim$

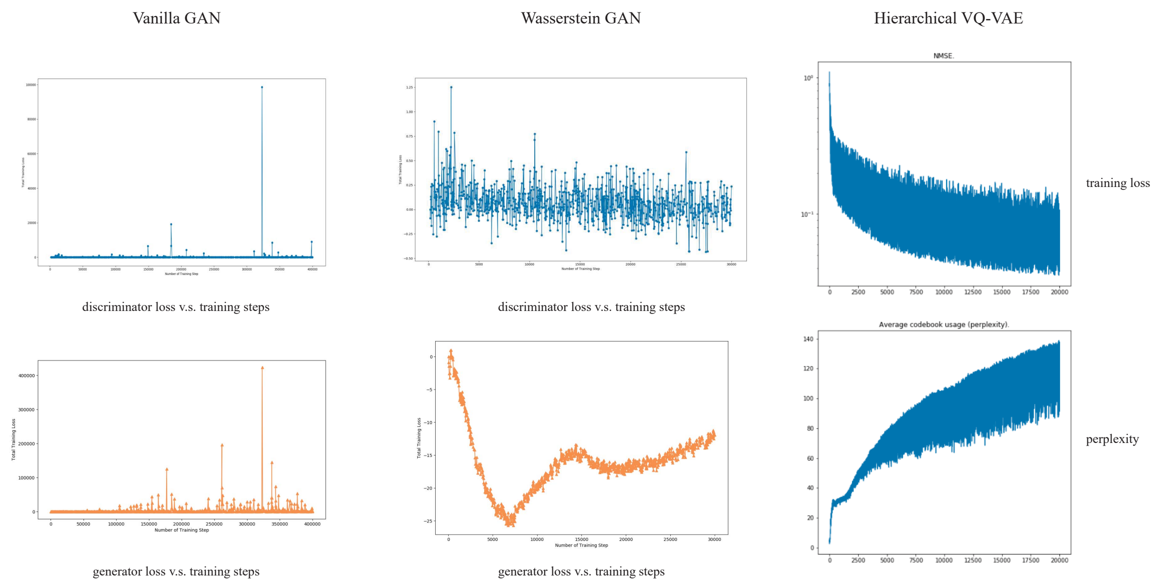


Figure 4.6: The training curves of three generative models.

$\mathcal{N}(\mu_{grass}^{fake}, \Sigma_{grass}^{fake})$. The Fréchet inception distance between the real buffelgrass set and fake buffelgrass set can be denoted as

$$\begin{aligned} FID(x_{grass}, x_{grass}^{fake}) &= \left\| \mu_{grass} - \mu_{grass}^{fake} \right\|_2^2 \\ &+ Tr(\Sigma_{grass} + \Sigma_{grass}^{fake} - 2(\Sigma_{grass}\Sigma_{grass}^{fake})^{\frac{1}{2}}), \end{aligned} \tag{4.17}$$

where $Tr(\cdot)$ sums up all the diagonal elements. The smaller the FID, the more diverse generated images are, and more certain that an image belongs to a specific class. We validate the diversity and quality of the generated buffelgrass patterns via FID, which is shown in Table. 4.4. A huge boost is observed from employing W-GAN instead of vanilla GAN. The vanilla GAN achieves FID score of 109.43 by comparing real image set and synthetic image set. The deployment of W-GAN falls dramatically from 109.43 to 32.80. From Fig. 4.6(a), we can also clearly observe that the training curve of the generator is much stable compared to the generator of vanilla GAN. The adoption of vector quantised VAE further improves the performance by 1.95, which indicates a more diverse and high-quality set of images are generated.

Apart from the numeric results indicating the difference between the real data set and

Table 4.4: The Comparison of Synthetic Buffelgrass Generation via Numeric Results

Metric	Fréchet Inception Distance
Vanilla GAN	109.43
W-GAN	32.80
VQ-VAE	**30.85**

the generated data set, a visualization presentation is also adopted to provide human sensory comparison. In Fig. 4.7, output samples from three generative models are compared. The column (a), (b), and (c) correspond to the results from vanilla GAN, W-GAN, and vector quantised VAE, respectively. Vanilla GAN generates results that can hardly be classified into a kind of natural vegetation. The marks of artificial manipulation are evident. On the contrary, W-GAN is capable of creating more realistic buffelgrass patterns, on which the leaves can be clearly distinguished. However, the shortcome of the output from W-GAN is that it lacks fidelity in the images. If we blend those images patterns back into the background, the distinction between synthetic images and background is huge, and the blurry patterns involve unnecessary noise. Vector quantised VAE solves this problem by generating high-fidelity and high-diversity buffelgrass patterns shown in column (c). The blending of the high-fidelity patterns make the synthetic training images for detection more realistic.

In Fig. 4.8, we show the results of blending generated buffelgrass into a background grassland. The "Real Data" and "Synthetic Data" are compared in the context of detection result. The purple bounding boxes localize the synthetic buffelgrass patterns which are generated by vector quantised VAE. The location for each synthetic buffelgrass is randomly selected. Truncations and occlusions are included in order to ensure the robustness. The blending results are close to the real data shown in Fig. 4.8. The close inception is needed to distinguish them from real ones which makes them high-quality synthetic samples.

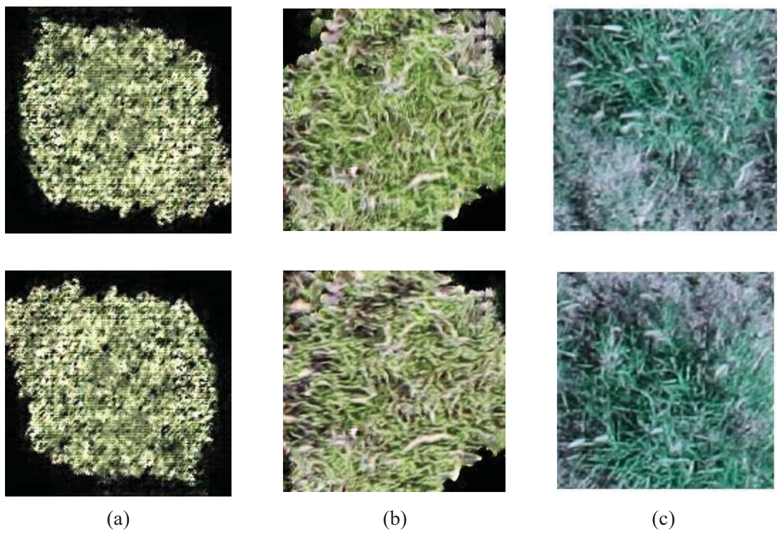


Figure 4.7: The comparison of visualization sample results of three generative models.

4.2.2 Results for Buffelgrass Detection with Different Augmentation Schemes

In this work, three different augmentation schemes (mentioned in Section 4.2.1) along with the manual augmentation are conducted on the raw data set. In the network setup stage, the usage of transfer learning is also evaluated in order to evaluate its effects on the training performance. The comparison of four augmentation schemes is presented in Fig. 4.9. The upper graph shows the training loss of four different augmentation schemes without transfer learning. All of the four schemes have shown intense fluctuation in the first half of the training. Training curves converge gradually while in the second half of the training, the fluctuation becomes smaller compared to the initial stage of the training. It is obvious that the VQ-VAE augmentation scheme (represented by the dashed purple line) has the smallest variance in the training loss when the curve is stable. The W-GAN augmentation scheme can achieve similar convergence compared with the manual augmentation. However, in the GAN augmentation scheme, training does not converge into a stable and acceptable range without transfer learning. The distinction becomes even larger when comparing the GAN augmentation scheme with other three schemes under the condition that

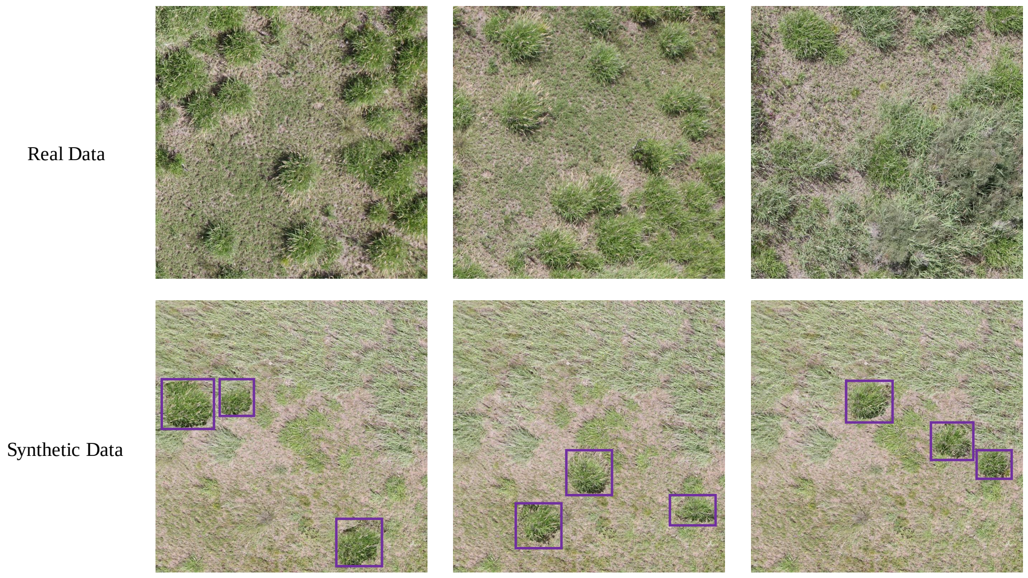
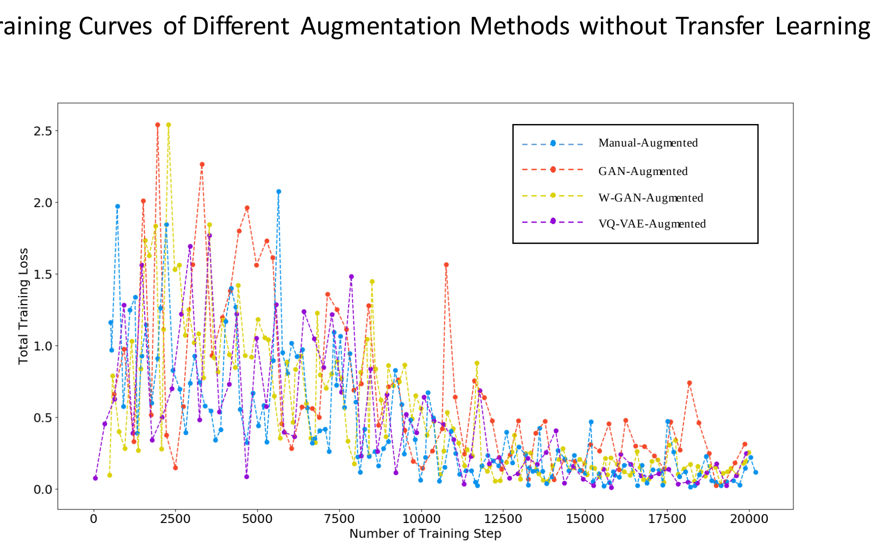


Figure 4.8: Samples of synthetic images for buffelgrass detection.

transfer learning is adopted (shown in the lower graph in Fig. 4.9). Our experiment and observation indicates that GAN training is affected and limited by the scale of the output. The larger the output, the harder to achieve a stable training and higher fidelity. On the contrary, the training curves look similar for the other three schemes when the convergence is achieved.

In Fig. 4.10, we include the training curves of different augmentation schemes independently in order to clearly demonstrate the influence of transfer learning on the training performance. It is clear that transfer learning brings better convergence performance to the training of all augmentation schemes. Moreover, when we zoom into the first a few steps, the initial loss of the training with transfer learning is significantly smaller than the ones without transfer learning. The results demonstrate that transfer learning brings at least two advantages in training a detection model: 1) The small initial loss (or better parameters intialization) and fast convergence; 2) Better convergence performance after the training is stable.

Apart from the numeric analysis of the testing performance, we show the visualiza-



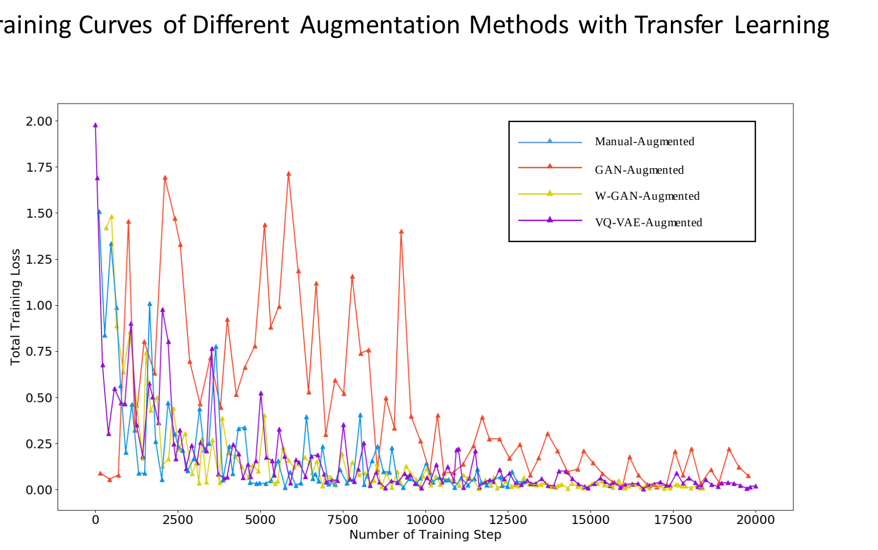


Figure 4.9: Training curve of the detection with (w.o) transfer leanring.

tion results of a group of samples from the testing set. Different augmentation schemes are validated on this group of samples shown in Fig. 4.11. Row (I), (II), (III), (IV), and (V) represents the visualization results on different samples from the target (manual labeling by expertise), manual augmentation, GAN augmentation scheme, W-GAN augmentation scheme, and VQ-VAE augmentation scheme, respectively. Column (a)-(e) represents each testing sample with different testing models. The testing samples shown in the figure include different growing conditions of the buffelgrass. Five samples (each with the size

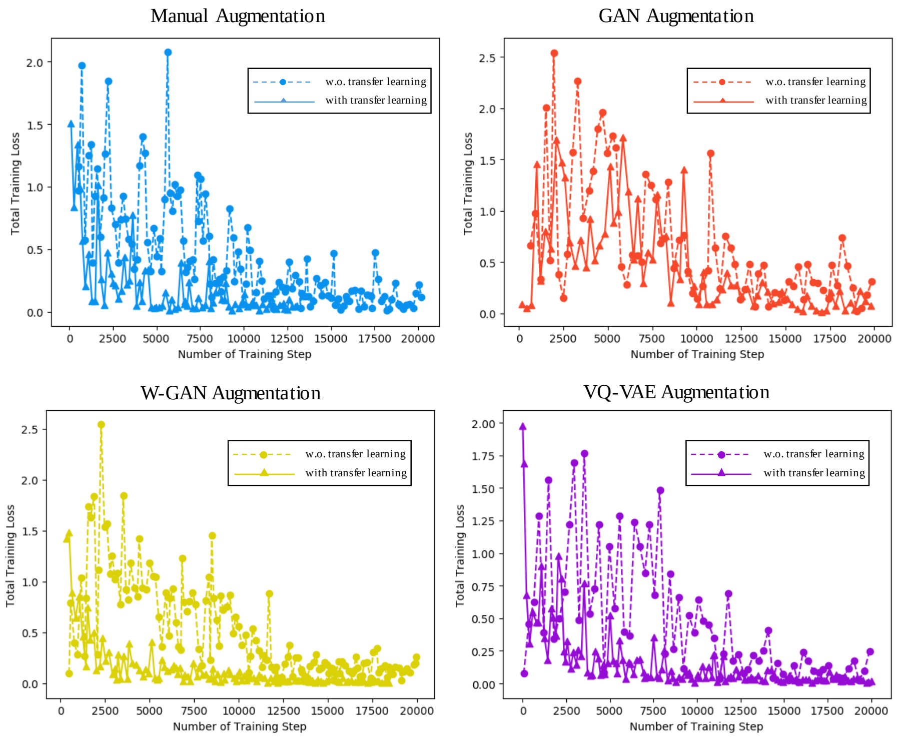


Figure 4.10: The comparison of the convergence of different augmentation strategies.

Table 4.5: Detection Results with Different Augmentation Schemes.

	w.o. transfer learning	with transfer learning
plain detection	2.522	
Manual Augmentation	8.953	37.340
GAN Augmentation	7.198	31.580
W-GAN Augmentation	12.070	46.920
VQ-VAE Augmentation	**29.120**	**54.150**

2000×2000 in pixel) involve the buffelgrass growing either sparsely or densely, horizontally or vertically. Column (a) and (b) contain two samples with sparsely distributed buffelgrass. The results show that none of the augmentation schemes helps the detection model much when buffelgrass is sparsely ditributed. Column (c)-(e) contain three samples where buffelgrass patterns are closely and densely distributed. All of the four schemes have precise prediction to some extent. However, a performance boosting can be clearly observed through row (III)-(V). It is hard to distinguish the difference between the results from row (II) and (III), but the adoption of the Wasserstein distance in GAN and the utilization of

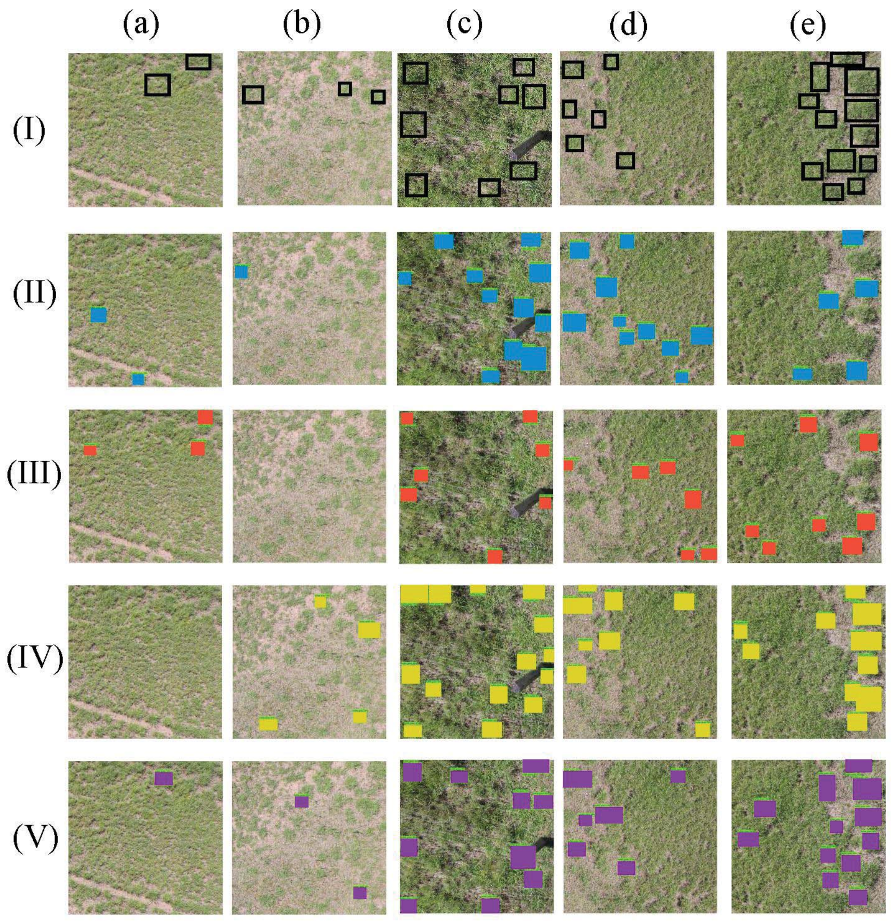


Figure 4.11: The visualization of the testing results with different augmentation schemes.

the vector quantised VAE dramatically improve the detection performance in terms of the metric of confusion matrix (true/false positive/negative). Specifically, VQ-VAE, compared to other schemes, can make more accurate prediction especially in the place where a large population of buffelgrass grows closely.

We demonstrate a visualization image with the resolution of $8,688 \times 5,792$ in Fig. 4.12. The result in the Fig. 4.12 shows the detection model with the VQ-VAE augmentation is capable of capturing all the annotated boxes with an accurate location prediction. Recall the detection total loss design in the Section 4.1.4, we find this design work well with a dramatic performance on the box regression. However, it sacrifices the accuracy of classification that some spots on the lower area of the image are misclassified into the invasive

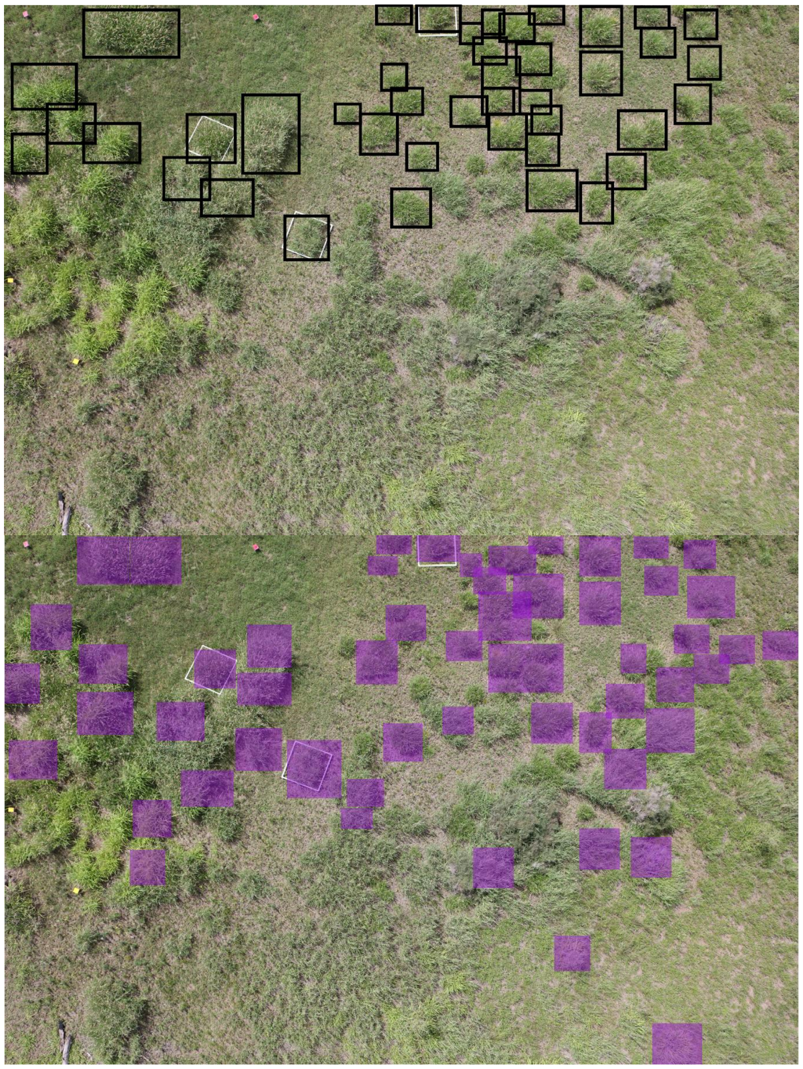


Figure 4.12: The visualization of the testing result of an original sized image taken by UAV.

grass.

In order to investigate the impact of different synthetic data volume on the detection accuracy. In Fig. 4.13, We compare the average precision (AP@IOU0.6) for different synthetic-to-real data ratios. Average precision is a metric that combines recall and precision for ranked retrieval results. For each evaluation, we fix the total volume of the training data, but blend with different ratios of synthetic data. We can observe from this figure that synthetic data helps increase the detection accuracy more when its volume is moderate. For detection model without transfer learning, we get the highest precision when the synthetic

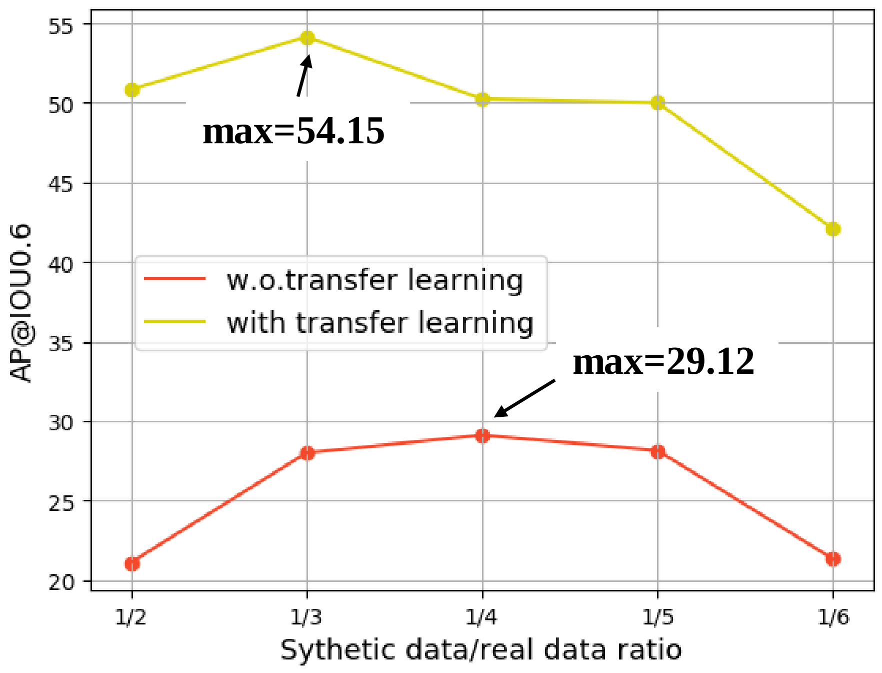


Figure 4.13: The comparison of the average precision for different synthetic data ratios.

data/real data ratio is around $1/4$. For detection model with transfer learning, the highest precision emerges when the synthetic data/real data ratio is around $1/3$. In accordance with this observation, we can derive that keeping a reasonable proportion of real data set is essential to maintain the major distraction mode, on top of which a more advance synthetic data augmentation scheme can help.

4.3 Conclusions

In this study, we aim to develop a set of technologies that supports the automatic detection of invasive vegetations. Even our study was progressed in a specific vegetation type, buffelgrass. The developed model and corresponding metholodies are well suited to other types of tasks related to invasive vegeation monitoring, or the more wide community with strong interests in automating the process of monitoring the natural environment.

In this study, we overcome a set of challenges thare are associated with real business applications:

- The raw images taken by the UAVs are not suitable for processing directly.
- In practice, it can be difficult to obtain enough training data set desired by supervised learning approaches. This poses the sharp contrast from some bench-mark studies widely adopted in academia, the data labelling effort is not considered as a bottle-neck. This universal challenge to most industry application in the domain of machine learning leads us to develop a novel approach based on probabilistic generative model, i.e., VQ-VAE to automatically generate authentic buffelgrass image patches, which can be implanted into the background image. This approach provide an automatic approach to meet the real challenge of adopting deep learning models, and a solution to the imbalanced data set, a common issue in the machine learning studies. On top of this training data boosting process, we further utilize the transfer learning technique to boost the training performance. This transfer learning scheme also provides a satisfactory performance in the evaluation. Compared with the model trained from scratch, the transfer learning scheme brings a much better catch on the buffelgrass pattern, typically on strong patterns. Meanwhile, the transfer learning scheme consumes less training time to converge with a better convergence performance.
- In the context of detection model design, we adopt deep ResNet as feature extractor. Constructing an automatic and intelligent mechanism for pattern augmentation is a promising conception to make current work more solid, typically on strong patterns. In this work, we implement the data augmentation by attaching those pattern images on the background image manually. This process boosts the detection performance well, but it is relatively time-consuming with little help on increasing the diversity of the training set. Developing an automatic blending mechanism for appending small patterns on the large background is a promising conception to make current work more solid.

Chapter 5

Unexploited Seismic Data Inversion by Joint Distribution Optimal Transport with Deep Encoder-Decoder Networks

Seismic data is used to reveal the subsurface structure based on seismic wave propagation. From the reflected seismic wave, geoscientists can estimate the physical properties of the subsurface. Seismic exploration can be applied and utilized in many fields (e.g., hydrocarbon exploration and Earth's crustal investigation) since its capability of detection from small to large scale. To implement seismic exploration, a signal transmitter (e.g., an air gun) and several receivers are required. The inversion of the seismic data generates the reconstruction of the subsurface velocity model, based on which geoscientists are able to estimate the target imaging bodies. There are several types of seismic inversion. Stochastic inversion is one widely used method for seismic wave inversion [109] [110] [111]. Besides, full-waveform inversion (FWI) is now appealing due to its high accuracy and resolution [112] [113].

Besides the subsurface imaging we have introduced and discussed in the last paragraph, FWI is also applied to other fields such as breast cancer detection in medicine [114]. FWI can be calculated either in the time domain or the frequency domain. Either approach involves non-linear computation which is computationally expensive to solve [115]. Similar to most machine learning problems, the solution to inversion problems may include many local minima making the technique less robust. To mitigate the ill-posedness of the problem, many approaches have been proposed and developed in recent years including preconditioning methods [116], prior information-based methods [117], and multiscale methods [118], etc.

In recent years, DNN reveal the significant capability of generalizing non-linear map-

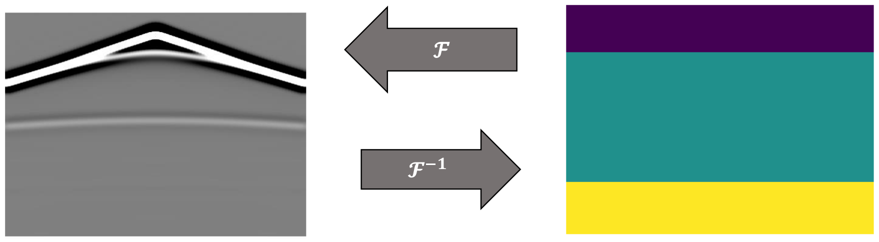


Figure 5.1: A general data-driven seismic inversion task definition.

ping in the fields of natural language processing, image classification, object detection, and even image-to-text transformation [119] [120] [121] [122] [123] [124] [125]. For the inversion problem, DNNs-based image-to-image reconstruction brings new perspectives for inversion to learn the mapping function $\mathcal{F}$ from the input image to output reconstruction. Since the image transformation requires the ability of spatial information reservation, the CNN is widely used for inversion problem processing. Jain and Seung [126] proposed a vanilla CNN to denoise an image subjected to Gaussian noise. Eigen et al. [127] proposed a CNN-based framework for denoising photographs which are covered with dirt and rain. Specifically, for the seismic inversion problem, Araya-Polo et al. [128] proposed a CNN-based model to reconstruct velocity maps from a semblance cube calculated from raw data. CNN is able to be packaged into a residual structure that learns a residual between two or more layers by skipping connection from the input of the residual block to its output. This residual network structure is well suited to image restorations when there is adequate similar content between the input and output [129] [130]. By incorporating analytical solutions, the inversion problem can also be solved. [131] proposed an idea to start with an analytical approach and an associated inference algorithm and unfold the inference iterations as layers in a deep network. Zeng et al. [132] proposed the autoencoder method to learn useful representations of low-resolution and high-resolution images. Autoencoder is one branch of CNN based structure that is specifically employed for image reconstruction. Likewise, Wu et al. [133] proposed a model named InversionNet to build the mapping from raw seismic data to velocity maps by embedding the information in the raw seismic data into an encoded space. As an improvement for [133] due to the information loss in the embedding space, Li et al. [134] proposed a built-in component named SeisInvNet to learn a feature map spatially aligned to the velocity model from each seismic trace. Afterward, velocity maps are reconstructed from the feature maps.

After reviewing existing literature, we found that few researchers work on the topic of domain adaptation in the deep networks based seismic inversion problem. However, domain adaptation is a common and consistent demand from the real industry application. Also, in [134], the authors mentioned they carried an experiment of their proposed model on the noisy data set on which they attempted domain adaptation by finetuning their model from the existing checkpoints. However, this approach suffers from a degradation problem on the original data set. This degradation problem weakens the generalization property of the model which makes the reusing of the original model difficult. In practice, practitioners have to make space to store both the source and target model, and need time for maintaining both models for different but similar data flow.

Inspired by the above demand and challenge, we propose to utilize the optimal transport approach to assist the training of the deep network in order to let the model adapt to the noisy data set while preserving the performance on the original data set. The proposed model is named encoder-decoder joint distribution optimal transport (enc-dec JDOT). In this study, we construct our model from a vanilla autoencoder. However, we investigate the encoder and decoder separately. The output of the encoder is an embedding space which contains the most significant information in the inputs. We also take this latent vector as an output of the model. Thus, the networks will produce the latent vector as well as the final reconstruction. By combining the latent vector distance and the final output distance, we get a total distance function that can measure the difference between the original data set (source domain) and the new noisy data set (target data set). This total distance function works as a total cost for transferring the source distribution to target distribution. In this way, we are able to solve an optimal transport problem by getting an optimal transport plan

γ. The dot product of γ and total distance will be backpropagated through the networks and update the weights which can couple the performance on the source and target data set.

We carry out experiments on the self-proposed three-layers source data set and its corresponding noisy target data set. From the visualization and analysis results, we demonstrate our proposed enc-dec JDOT framework can adapt to the target data set without "looking at" the target data label. Major contributions in this study are summarized as follow:

- We investigate the DNN-based seismic inversion in a specific background that the deep learning model can immigrate to an unexploited data set. The study indicates the promising potential that deep learning models can be reused on new data sets. Thus, it saves time and economic costs.
- We demonstrate our proposed model can adapt to the new data set while preserving the performance on the original data set to certain extent. This finding may lead to a learning model that can generalize the patterns in several kinds of seismic and velocity data, which shows there is a possibility that a universe model can be created to implement inversion task on multiple types of seismic data.
- We generalize the deepJDOT method proposed in [11] from classification only to the regression problem. The classification task has the advantage of certain data-label correspondence which helps to put a similar source and the target data into the same category. By the experiment, we show that the regression task can also work on the well-designed source and target data couples.

The rest of this chapter is organzied as follows: Section 5.1 introduces more related works on inversion solution using encoder-decoder structure as well as the optimal transport in domain adaptation. The proposal of the deep networks is introduced in Section 5.2. In Section 5.3, the basic knowledge of optimal transport and its extended version, joint distribution optimal transport will be elaborated. In Section 5.4, we propose enc-dec JDOT to solve the domain adaptation problem in the field of seismic inversion. The corresponding experiment results will be demonstrated in Section 5.5. Finally, a conclusion will be drawn in Section 5.6.

5.1 Related Works

This study covers several key fields of research. First of all, a deep learning based approach is utilized to solve the FWI problem. Specifically, an encoder-decoder network is used to learn the seismic inversion mapping end to end. Furthermore, the optimal transport based method is used for domain adaptation. In this section, we will introduce some related works categorized into the above topics.

5.1.1 Deep Learning-Based FWI

In [135], Lewis et al. did not implement an end-to-end generalization of FWI using deep learning. Instead, they utilize neural networks to generate some prior knowledge. This prior information will assist FWI for a reconstruction. Considering seismic measurement is time-series data, Richardson et al. [136] proposed using RNNs to solve the forward wave propagation modeling.

5.1.2 Autoencoder Network-Based Seismic Inversion

The autoencoder concept is first proposed in [5], which is originally and widely used for data compression. The autoencoder consists of two components: the encoder and the decoder. The encoder extracts the principal components and information in the input and transfers them into a 1-D embedding vector. The decoder translates the embedded information inside the embedded space into a reconstruction of the input. Normally, the input can be an image, a group of texts, or a series of well-organized signals. For the data which has specific structures or patterns, autoencoder can reconstruct them without providing low-level information and without losing much information. Due to the property of the autoencoder, it is widely applied to high-fidelity images generation [137] [138] [8]. For the autoencoder applications, the embedding vectors contain the high-level portrait of the input. If low-level features are needed due to the high complexity of the information inside the input, the low-level features can also be aligned by other networks layers [139] [140]. For the seismic inversion problem, the basic autoencoder structure can work. This structure is adopted in [133] which is named InversionNet. The InversionNet can handle the horizontal interfaces and dipping faults and be enhanced after applying conditional random field (CRF) [2] [3]. There is performance limitation in [133] when investigating the details of some seismic inputs. To address this challenge, Li et al. [134] proposed a novel component named seismic inversion network (SeisInvNet) to first learn a feature map spatially aligned to the velocity model from seismic traces.

5.1.3 Optimal Transport with Machine Learning and Seismic Inversion

Optimal Transport (OT) [141] is a theory to compare the probability distribution in terms of geometrical distance. In the traditional Monge [142] or Kantorovich problem [143] [141], optimal transport is used to solve a transport plan γ. With the rise of machine learning, the optimal transport is widely applied as a distance measurement which can re-define the traditional loss function by injecting some geological property, which is known as the Wasserstein distance [144] for aligning two distributions in a more smooth and effective way. OT also has been applied to learn the transformation between [145] [146] [147] domains in order to accomplish domain adaptation with associated theoretical guarantees [142] [143] [141]. In the domain adaptation, γ is used to transport data samples through an estimated mapping called barycentric mapping [148]. A new classification model then is learned on the transported source data representation.

There are several works focusing on solving seismic inversion using the optimal transport approach. In [149], Yang et al. proposed to use the Wasserstein distance as a replacement of the least-squares norm as a misfit function to avoid cycle-skipping issues. This issue can increase the risk of computing a local rather than the global minimum of the misfit. Similarly, [150] proposed to use optimal transport distance to measure the misfit between the seismograms. In [151], L Métivier et al. used optimal transport to implement 3D full FWI. To our best knowledge, there are few studies focusing on the optimal transport-based domain adoption for seismic inversion, which encourages us to focus on this field.

5.2 Deep Networks Structure

As aforementioned, this study intends to derive the inversion mapping $\mathcal{F}^{-1}$ so that velocity models can be produced from corresponding seismic input data. In general, we denote seismic measurement as $\mathbf{m}$, and corresponding velocity model as $\mathbf{v}$. The forward and inverse process can be illustrated as

$$\begin{cases} \mathbf{m} = \mathcal{F}(\mathbf{v}), \\ \mathbf{v} = \mathcal{F}^{-1}(\mathbf{m}). \end{cases} \tag{5.1}$$

In this study, we construct deep networks to directly obtain an approximation of $\mathcal{F}^{-1}$ mapping from seismic data $\mathbf{m}$ to velocity model $\mathbf{v}$. The autoencoder based structure is utilized as an approximator. Unlike the traditional autoencoder that learns the input itself, the autoencoder in this study first learns the embedding of the input seismic data by the encoder, and then decodes the embedding and translates into the velocity model domain. The encoder extracts the high-level features from the seismic input and obtains the embedding vector which has a much lower dimension. The decoder translates the embedding into the velocity model domain. The detailed implementation of the network structure is indicated in Fig. 5.3. In the problem we are going to solve, both source and target model adopt the

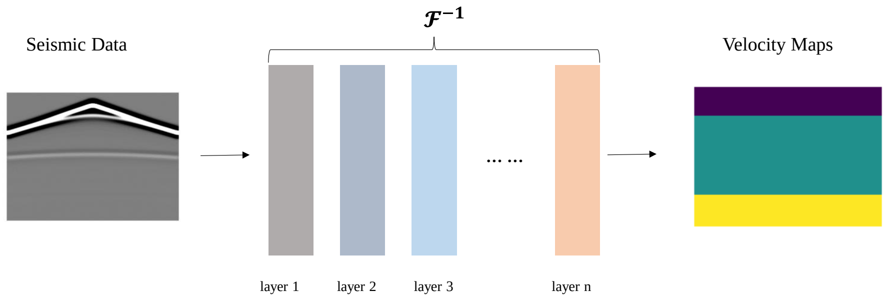


Figure 5.2: Neural networks structure for learning the inversion function $\mathcal{F}^{-1}$.

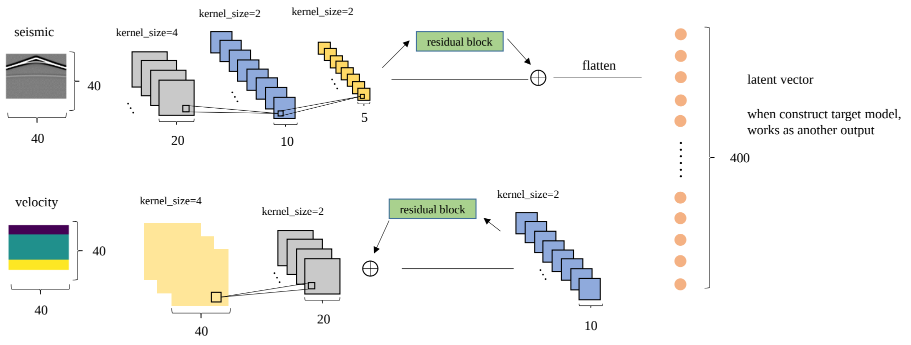


Figure 5.3: The demonstration of the encoder and decoder network on which domain adaptation is implemented.

same network structure.

5.2.1 Encoder

The overall approximation of the encoder can be denoted as an encoding function

$$\mathbf{z} = \mathbf{e}(\mathbf{m}), \tag{5.2}$$

where the output of the encoding is a latent embedding vector $\mathbf{z}$. The components of encoder include convolutional layers, a residual layer, and ReLu or Sigmoid function. The convolutional layers are crucial for feature extraction while reserving the spatial correlation among different locations. The residual layer is stacked after a few convolutional layers since we want to build a deeper network while avoiding the vanishing gradient problem. The residual layer is nothing but adding a shortcut to the next layer directly from the output of the current layer. The main path can be stacked with several convolutional layers as well.

Table 5.1: Networks Structure of Encoder and Decoder.

Encoder Networks			**Decoder Networks**		
Layer	Filter/Stride	Output Size	Layer	Filter/Stride	Output Size
Conv1	$4 \times 4/2 \times 2$	$20 \times 20 \times 8$	Conv-Trans1	$2 \times 2/2 \times 2$	$10 \times 10 \times 16$
Conv2	$2 \times 2/2 \times 2$	$10 \times 10 \times 16$	Residual Stack	$3 \times 3/1 \times 1$	$10 \times 10 \times 16$
Conv3	$2 \times 2/2 \times 2$	$5 \times 5 \times 16$		$1 \times 1/1 \times 1$	$10 \times 10 \times 16$
Residual Stack	$3 \times 3/1 \times 1$	$5 \times 5 \times 16$		$3 \times 3/1 \times 1$	$10 \times 10 \times 16$
	$1 \times 1/1 \times 1$	$5 \times 5 \times 16$		$1 \times 1/1 \times 1$	$10 \times 10 \times 16$
	$3 \times 3/1 \times 1$	$5 \times 5 \times 16$	Conv-Trans2	$2 \times 2/2 \times 2$	$20 \times 20 \times 16$
	$1 \times 1/1 \times 1$	$5 \times 5 \times 16$	Conv-Trans3	$4 \times 4/2 \times 2$	$40 \times 40 \times 3$

The output of the residual layer is the addition of the main path and the shortcut. Typically, ReLu activation is applied after every convolutional layer. However, we utilize the Sigmoid function after the last layer instead of ReLu since the output range of the Sigmoid function can be easily mapped to the normalized image.

5.2.2 Embedding Vector

The embedding vector $\mathbf{z}$ is the output of the encoder network. Generally, the embedding vector acts as a latent representation only. In this study, it is also manipulated and contributes to the final loss function. By narrowing the difference in the latent space between the source domain and target domain, the model is able to learn similar patterns from similar inputs. Moreover, comparing the latent space is more efficient than comparing it in the input space. The comparison makes sense since the embedding vector contains the principal information in the input.

5.2.3 Decoder

The embedding vector $\mathbf{z}$ is translated into the velocity model domain by the decoder networks, whose mapping can be denoted as

$$\mathbf{v} = \mathbf{g}(\mathbf{z}). \tag{5.3}$$

The decoder networks work as an inverse process of the encoder networks. The components of the decoder are almost the same with the encoder except all convolutional layers are replaced by the deconvolutional layers. Deconvolution operation (a.k.a. convolution transpose) enlarges the dimension of the current tensors, which help translate the embedding vector to the final output. In the residual layer, convolutional layer also works since we put the stride $(1, 1)$ which reserves the size. The detailed implementation can be found in Fig. 5.3 and Table. 5.1.

5.3 Optimal Transport for Domain Adaptation

The proposed method is based on optimal transport, which is introduced first in this section. After grasping the basic knowledge of the optimal transport, we move on to joint distribution optimal transport (JDOT) to explore the domain adaptation. In Section 5.3.1, we introduce the basic concept of optimal transport. In Section 5.3.2, we introduce the formulation optimal transport-based domain adaptation technique named joint distribution optimal transport.

5.3.1 Optimal Transport

Optimal transport [141] (OT) is the symmetrical measurement of probability distribution in a geometrical manner. In the standard OT formulation, it searches a probabilistic parameter $\gamma \in \prod(\mu_1, \mu_2)$ to couple two distributions μ_1 and μ_2. This coupling (or transport plan) arises a transport cost, which consists of the dot product of the distance between two data samples with the cost to bring two samples together. Mathematically, it can be formulated as

$$OT_c(\mu_1, \mu_2) = \inf_{\gamma \in \prod(\mu_1, \mu_2)} \int_{\mathcal{R}^2} c(x_1, x_2) d\gamma(x_1, x_2), \tag{5.4}$$

where $c(x_1, x_2)$ is the cost function measuring the workload to bring sample x_1 and sample x_2 together. The space of joint probability distributions is described as $\prod(\mu_1, \mu_2)$, which in the discrete setting can be formulated as

$$OT_c(\mu_1, \mu_2) = \min_{\gamma \in \prod(\mu_1, \mu_2)} \langle \gamma, \mathbf{C} \rangle_F, \tag{5.5}$$

where $\langle \cdot, \cdot \rangle_F$ is the Frobenius dot product, $\mathbf{C} \geq 0$ is a cost matrix $\in \mathbb{R}^{n_1 \times n_2}$ representing the pairwise costs $c(x_1, x_2)$, and γ is a matrix of size $n_1 \times n_2$ with prescribed marginal. The objective in (5.5) in the machine learning context typically utilized to measure the distance between two distributions. This distance is named Wasserstein distance when the distance is L1 or L2 norm. When on a small scale, solving (5.5) is a simple linear programming problem with equality constraints.

5.3.2 Joint Distribution Optimal Transport

JDOT is proposed by Courty et al. [152]. The main contribution is the proposal of learning a classifier embedded in the cost function c directly by jointly considering the source data set and target data set. The traditional solution is to adapt the representation (in the raw data space or latent space) first and learn the classifier on the adapted features. The basic idea in [152] is to jointly consider the source/target data and features/labels. μ_s (for source domain) and μ_t (for target domain) are used to measure the product space $\mathcal{X} \times \mathcal{Y}$. The cost associated with space $\mathcal{X} \times \mathcal{Y}$ is denoted as

$$d(x_i^s, y_i^s; x_j^t, y_j^t) = \alpha c(x_i^s, x_j^t) + \lambda_t L(y_i^s, y_j^t), \tag{5.6}$$

where the annotation s and t represent source domain and target domain, respectively. For the classification task, $c(\cdot, \cdot)$ can be represented as L2 norm and $L(\cdot, \cdot)$ can be represented as cross-entropy. α and λ_t are hyper-parameters serving as a coefficient to control the weight between the contribution of feature distance and label distance. Since we only collect input samples in the target domain, we design a surrogate function $f(x_j^t)$ to replace the target label y_j^t. In general, function $f(\cdot)$ works as a function generalization is a neural networks context.

For the classifier in the learning context and referring to (5.5), a minimization formulation can be denoted as

$$\min_{f,\gamma\in\Pi(\mu_s,\mu_t)} \langle\gamma, \mathbb{D}_f\rangle_F, \tag{5.7}$$

where $\mathbb{D}_f$ involves a classification loss representing all the pairwise costs $d(\cdot,\cdot)$. For better discrimination results, the samples being paired share similar patterns and the same label in the classification problem context. In [152], authors prove the minimization of (5.7) equals to the minimization of the learning bound in domain adaptation problems. There exists a limitation in [152]. γ can scale quadratically in size to the number of samples being transported. In [11], this drawback is solved by transport a mini-batch of data each iteration. However, [11] only considers the application of JDOT in the classification tasks. Thus, we proposed our enc-dec JDOT to implement the regression task by replacing the classification loss with mean squared error and carefully designing the data sample pairs (x_i^s, x_j^t) to make it work in the seismic inversion scenario.

5.4 Proposed Methods

In this section, we first introduce the formulation of the proposed enc-dec JDOT in Section 5.4.1. Then, in Section 5.4.2, we elaborate on how to solve the objective defined in enc-dec JDOT.

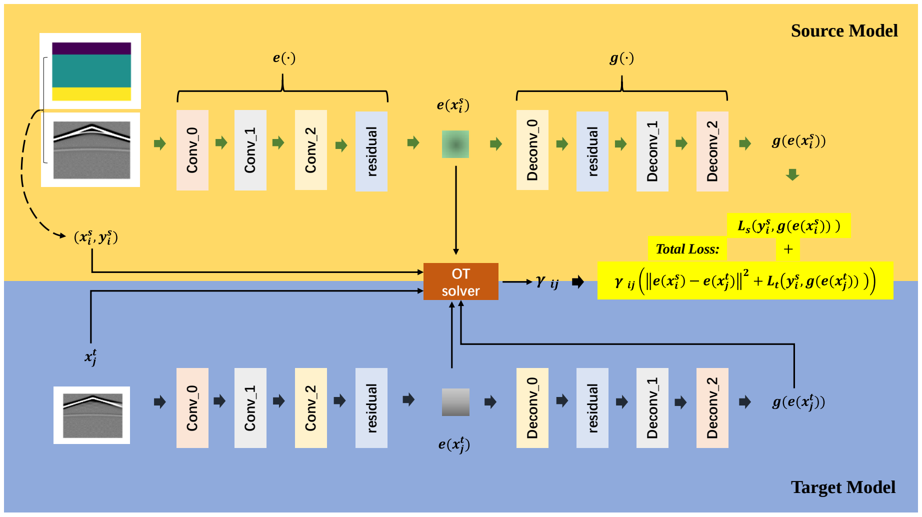


Figure 5.4: The overall framework of Enc-Dec-JDOT.

5.4.1 Encoder-Decoder Joint Distribution Optimal Transport (Enc-Dec JDOT)

Speaking of the structure, our Enc-Dec JDOT model contains two parts: i) an encoding function $e : x \to z$ (e stands for "encoder"), which takes seismic measurements as inputs and maps them into a latent space Z. ii) a decoding function or a generator $g : z \to y$, which generates corresponding velocity maps from the embedded latent space Z. The latent representation z is the output of the encoding function e and the input of the decoding function g. To make the model also works well on the unexploited dataset, we conduct the optimal transport on the latent space Z as well as the output of the generator g, which is the minimization of the 2D-Wasserstein distance between the source output and target output. The solution can be formulated as follows

$$\min_{\gamma \in \prod(\mu_1,\mu_2), f, g} \sum_i \sum_j \gamma_{ij} d(e(x_i^s), y_i^s; e(x_j^t), g(e(x_j^t))), \tag{5.8}$$

where $d(e(x_i^s), y_i^s; e(x_j^t), g(e(x_j^t)))=\alpha_s \left\| e(x_i^s) - e(x_j^t) \right\| + \lambda_t L(y_i^s, g(e(x_j^t)))$, and α_s and λ_t are the hyper-parameters serving as the trade-off coefficients. From this formulation, we can observe that we not only incorporate the dissimilarities inside the latent space, but also the discrepancy in the output space of the generator g. By taking latent space into account, the dissimilarity in the deep representation can lead the training towards model compatibility since the noise in the input space such as the signal received on the surface level of the receivers is ignored.

As aforementioned, we consider a unsupervised scenario where the generator loss in the objective defined in (5.8) only relies on the target domain. This mechanism entirely forgets the generator learnt on the source domain which will dramatically affect the robustness of the model. To avoid such problem, we manually incorporate the loss defined on the source domain, which leads to the final Enc-Dec-JDOT objective:

$$\min_{\gamma,g,e} \frac{1}{n^s} \sum_i L_s(y_i^s, g(e(x_i^s))) + \sum_{i,j} \gamma_{ij}\alpha_s \left\| e(x_i^s) - e(x_j^t) \right\| + \lambda_t L(y_i^s, g(e(x_j^t))). \tag{5.9}$$

We will solve this optimization problem in a feasible memory and time complexity.

5.4.2 Solving Enc-Dec-JDOT

Referring to [11], solving the objective in (5.9) in deep JDOT structure, the optimization can be conducted in two-steps approach, since we found there are two groups of variables to be optimized: i) γ: related to the distribution coupling and ii) e,g related to the deep networks parameters. In each iteration of mini-batch training of the Enc-Dec networks, the optimization can be summarized as

- With fixed CNN parameters $(\hat{e}, \hat{g})$ and for each mini-batch, obtain the coupling

$$\min_{\gamma \in \Pi(\mu_s,\mu_t)} \sum_{i,j=1}^{m} \gamma_{ij}(\alpha_s \left\| \hat{e}(x_i^s) - \hat{e}(x_j^t) \right\| + \lambda_t L_t(y_i^s, \hat{g}(\hat{e}(x_j^t)))) \tag{5.10}$$

  using the network simplex flow algorithm.

- With fixed coupling $\hat{\gamma}$ obtained at the previous step, update the encoder e and decoder/generator g with stochastic gradient update for the following loss on the mini-

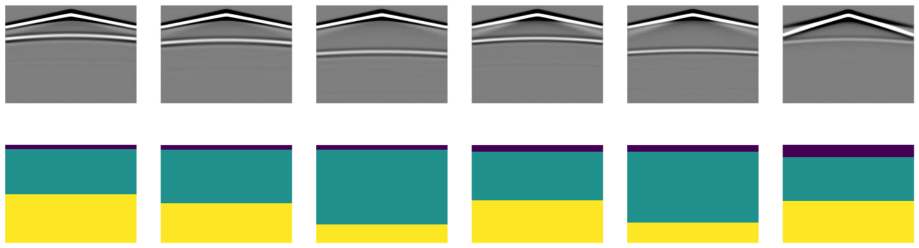


Figure 5.5: The plain data set (top) and their corresponding subsurface models (bottom)

batch:

$$\frac{1}{m}\sum_{i=1}^{m} L_s(y_i^s, g(e(x_i^s))) + \sum_{i,j=1}^{m} \hat{\gamma}_{ij}(\alpha_s \left\| e(x_i^s) - e(x_j^t) \right\| + \lambda_t L_t(y_i^s, g(e(x_j^t)))). \tag{5.11}$$

This two-steps strategy aligns the source and target data and $\hat{\gamma}$ performs label propagation between source and target domains. The generator can generate authentic velocity maps with both source latent representation and target latent representation as well.

Algorithm 4: Optimization of Enc-Dec-JDOT.

```
1: Initialize: The encoder networks $e = Enc(\cdot)$, the decoder networks $g = Dec(\cdot)$,
   the source seismic input $x^s$, the source velocity label $y^s$, and target seismic
   input $x^t$
2: for every batch of inputs ${x^s}_{batch}, {y^s}_{batch}, {x^t}_{batch}$ do
3:     with fixed networks parameters $\hat{e}$, $\hat{g}$, solve for $\gamma$ in (5.10)
4:     with fixed coupling $\hat{\gamma}$ update the network parameters $e$ and $g$ with
       ADAM [153] in (5.11)
5: end for
```

5.5 Experiments

5.5.1 Data Set

We create two data sets. First, a plain data set is created for the source model to train on. The plain data set is simulated with three-layers plain subsurface. Each velocity model contains 100×100 grid points. However, in order to increase the training speed and save

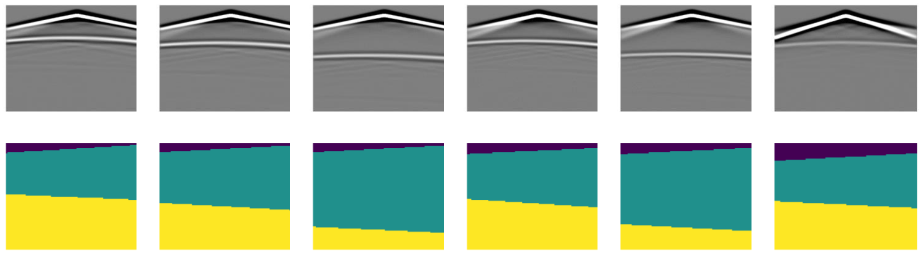


Figure 5.6: The noisy data set (top) and their corresponding subsurface models (bottom)

the storage space for the model, we resize the velocity model to 40×40 before feeding them into the deep networks. The difference among different velocity maps in the plain data set is the depth of each layer. In our simulation, the deeper in the subsurface, the higher the velocity. Then, based on the plain data set, we create a noisy data set with every boundary containing a certain slope. Generally, the noisy data set is more challenging to reconstruct since each slope has a random angle although in a certain range. Thus, the noisy data set contains more information for the learning model to infer.

By implementing forward modeling on velocity maps, we get corresponding seismic measurements (as training inputs) which are collections of synthetic seismograms. For both the plain data set and the noisy data set, we put one source in the middle along the horizontal axis. The receivers are evenly and symmetrically distributed on both sides of the source. The receiver interval is 10 m. We use a Ricker wavelet with a peak frequency of 10 Hz as the source time function. A staggered-grid finite-difference scheme with a perfectly matched layered absorbing boundary condition is utilized to generate synthetic seismic reflection data. The velocity maps and corresponding seismic data in the plain data set and noisy data set are demonstrated in Figs 5.5 and 5.6, respectively.

5.5.2 Experiment Settings

For training the source and target model, we collect $2,000$ data samples for training and 50 for testing in the plain data set and the noisy data set, respectively. As aforementioned, in order to reduce the computational complexity and computing time, we resize the original seismic data input from 100×100 to 40×40. To further reduce the input dimension, we normalize the input and cast it into grayscale images. For training the source model, we set the training batch size of 16 and only fetch data samples from the plain data set. Thus, the input size for training the source model is $(40, 40, 1, 16)$. After several convolution operations in the encoder networks, we get a flattened embedding vector with size $(400, 16)$. The decoder networks translate the embedding vector to the velocity map with the size back to $(40, 40)$. We train the source model for 15 epochs. After training the source model, we start to train the target model from the pretrained checkpoint. While training the target model, the data samples from both the plain data set and the noisy data set will be fed into the networks. The difference is that we take both input and the label from the plain data set but only input from the noisy data set. Since the target model will be trained and tested on the noisy data, this makes the task unsupervised learning. For how the input tensors are flowing inside the networks, please refer to Fig. 5.4. We use ADAM optimizer [153] to train the networks with the initial learning rate set as 1e-4. The deep networks have exactly $14,923$ trainable parameters trained end to end without other internal transformations. The networks training is accomplished with a single Nvidia GTX 1070 GPU. The development platform is TensorFlow Keras.

5.5.3 Evaluation Methods

The evaluation methods used in this study mainly consist of two components: the qualitative comparison and the quantitative comparison.

- Qualitative Comparison: Basically, qualitative comparison demonstrates the visual difference between two sets of velocity maps which are the reconstruction inference. Since velocity maps generally use several blocks of color to represent different layers (due to different wave propagation velocity inside the certain layer), the difference

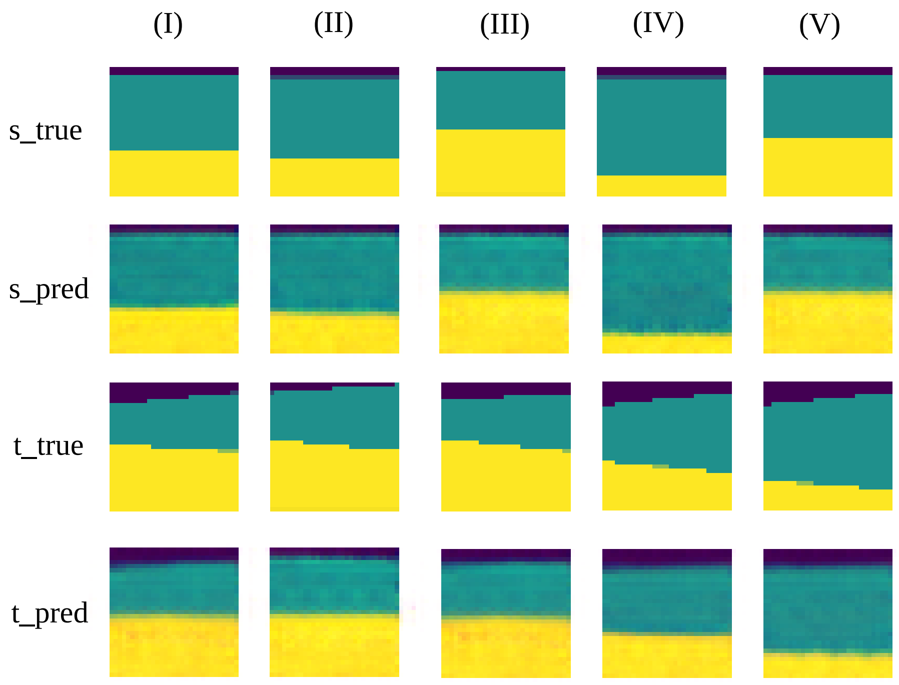


Figure 5.7: Groudtruth subsurface model and the inverted subsurface model generated by the source model.

can be easily told by observing the distribution of the colored blocks. Moreover, this comparison method can be investigated in detail to compare the misfit degree of the interface and the velocity value which is indicated by the RGB value of the heat map.

- Quantitative Comparison: The numeric approach is utilized in quantitative comparison. The reconstruction task is regarded as a regression task in the image domain. The commonly used metrics for the regression task are mean squared error (MSE) and mean absolute error (MAE). The MSE and MAE are denoted as $\frac{1}{n}\sum_i (\mathbf{v}_i - \mathbf{v}_i{}^*)^2$ and $\frac{1}{n}\sum_i |\mathbf{v}_i - \mathbf{v}_i{}^*|$, respectively.

5.5.4 Qualitative Comparison

Firstly, we demonstrate the qualitative comparison of the source model on the plain data set and the noisy data set, which is shown in Fig. 5.7. The columns (I) - (V) represent five different testing samples that are randomly drawn from the testing sets. The rows "s_true" and "t_true" represent the label samples from the plain data set and the noisy data set, respectively. For a certain column, we call it one type of data sample since the difference between the one from the plain data set and the one from the noisy data set is only the slope of the interface. However, the depth of the interface is close which makes the adaptation easier. The rows "s_pred" and "t_pred" represent the reconstruction from the source model by feeding the seismic data samples from the plain data set and the noisy data set, respectively. By looking into all the prediction results on the data samples from the plain data set, we are able to say the source model can reconstruct the velocity maps well, especially the depth of the interface. Referring to row "s_pred", we can observe some blurred pixels and some undulation, but in general, the source model works on the plain data set. Referring to row "t_pred", we have two observations. On one side, the source model has some generalization capability that it can predict some of the slopes in the noisy data set (e.g., row "t_pred" column "(I)" and row "t_pred" column "(II)"). On the other side, there is still a difficulty for the source model to predict the ladders on the interface, which is the main pattern difference between the plain data set and the noisy data set. This observation demonstrates the lack of interpretation on the data set which the source model has never "seen" before.

Secondly, we compare the target model performance with the source model using qualitative comparison. Similar to the demonstration of the source model reconstruction, we randomly draw another five data samples from the noisy testing data set. Since the noisy data set is simulated on the basis of the plain data set, the comparison should be more focused on if the model has grasped the pattern difference between the plain data set and the noisy data set. Generally speaking, the source model can also work to some extent on the noisy data set, which we already discussed in the last paragraph. However, we can investigate the details in the velocity maps shown in Fig. 5.8. Rows (I) - (V) represent five different testing data samples. Columns (a), (b), (c), and (d) represent the label, source model prediction, target model prediction, and the velocity profile comparison

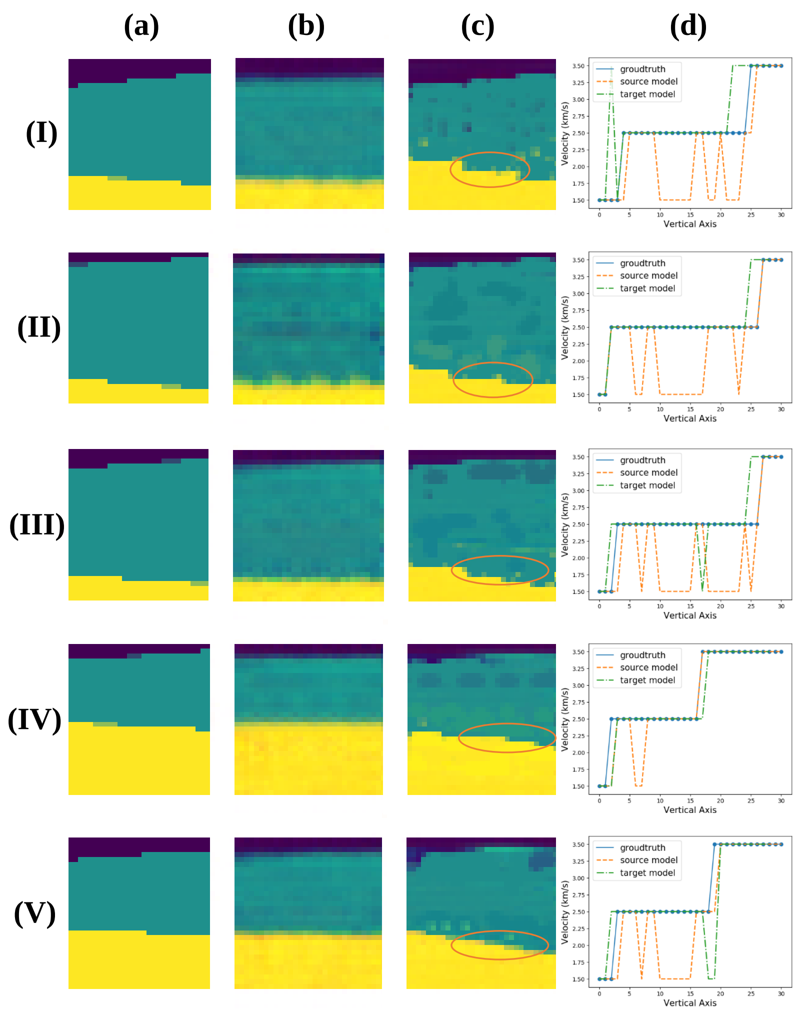


Figure 5.8: Qualitative comparison between the source model and target model on the noisy data set.

between the former three velocity maps. With a general observation, we know that both the source model and target model can successfully invert the velocity model. However, after zooming in and looking at the details on the interface of the subsurface layer, we can indicate that the target model presents a better reconstruction in terms of the subsurface structure and geological interface. For example, the velocity map label in the row (III) contains three obvious ladders on the interface. The source model prediction (in column (II)) barely presents this difference with the plain data, the reason for which is apparent that the source model never learns on the noisy data set. On the contrary, the target model

Table 5.2: Numeric Results of the Source Model and Target Model Inversion.

Data Set	Plain Test Data Set		Noisy Test Data Set	
metric / model name	MSE	MAE	MSE	MAE
Source Model	0.002137	0.010602	0.01385	0.02461
Target Model	0.002659	0.010250	0.00728	0.01726

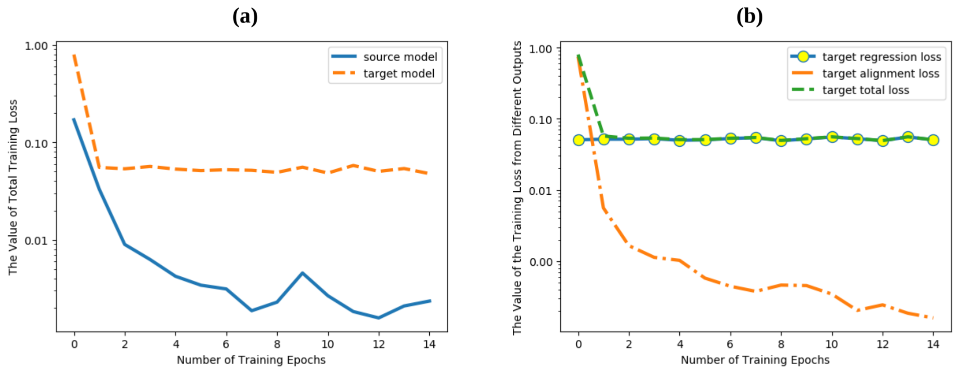


Figure 5.9: The training loss of the source and target model.

prediction (in column (III)) grasps this pattern difference and generates a somewhat blurred but obvious “ladder” pattern.

From the velocity profiles at the vertical central axis shown in column (d), we can find that the target model is better at the velocity recovery than the source model in most cases. The curve by the target model is almost identical to the groundtruth while the source model makes some minor bias more often. We can see that there is one sample that source model has better performance than the target model, which might be resulted from insufficient training.

5.5.5 Quantitative Comparison

In Table 5.2, we show the numeric results of source model inversion and target model inversion by using MSE and MAE on both plain test set and noisy data set. One general observation is that the source model works much better on the plain data set than the noisy data set with MSE is 20 times larger and MAE is twice after the model migration. Another observation is that the target model basically accomplishes in terms of 1) preserving the ability of generalization on the plain data set, and 2) achieving progress on the noisy data set while no supervised learning involved. Firstly, we can focus on the results on the plain test data set. For both MSE and MAE measurements, the target model can achieve almost the same performance without much degradation. The error increasing is totally acceptable. Secondly, we observe that the target model shows the success of adaptation to the noisy data set with an improvement in both MSE and MAE. The numeric results show this adaptation scheme for transporting source domain knowledge to the target domain.

5.5.6 Networks Mechanism Analysis

The Enc-Dec JDOT basically designs a custom loss function in a way that the knowledge of training the source model can be migrated to the target model by the OT solver recalling in Fig. 5.4. Understanding the training process is crucial for figuring out what is going on in the training procedure as well as a potential future improvement. In Fig. 5.9, we demonstrate the source model and target model loss curves as well as the loss components inside the total target model loss during training. From Fig. 5.9(a), we find that the convergence behavior for the source and target model is similar at first several epochs. When the loss curves are stabilizing, there is more undulation in the target model curve than the source model curve. After this observation, we move on to the subfigure (b). We find that there are also undulations for the alignment loss though in a relatively small range. However, the small vibration in the intermediate layer might cause a larger variance in the regression outputs. This observation might lead to a further research topic: how to choose the intermediate layer (or layers) to guarantee a more stable training.

5.6 Conclusions

In this chapter, we investigate the domain adaptation problem in deep learning-based seismic inversion. We found that few researchers have a focus on this topic which is yet of significant demand in the oil-an-gas industry. In light of the lack of research on this specific topic, we propose to utilize optimal transport as the tool to align the source domain and target domain for deep networks training, which is named Enc-Dec-JDOT. In our experiment, the alignment of the intermediate layer feature along with the output alignment assists the target model to achieve progress on a noisy data set which the deep networks have not exploited before. The key component of the proposed scheme is the design of a custom loss function which incorporates the distance between the source domain and target domain in the output layer with the distance in the intermediate layer. The distance is evaluated and optimized by the optimal transport algorithm, which is simply a linear programming problem. The parameters in optimal transport work as hyperparameters for assistance in training the deep networks. To validate the proposed framework, we create two self-proposed data sets to compare the performance of the source model and the target model. The experiment results show the target model can achieve progress on an unexploited data set. Our study paves a promising way for any pre-trained deep learning-based model to adapt to similar but unexploited data while reducing time and labor costs on obtaining training labels.

Chapter 6

Conclusion and Future Works

6.1 Conclusion Remarks

In this dissertation, we have successfully applied machine learning to various types of industrial applications with latent space representation and manipulation. Based on the results of the works in this dissertation, we can find the potential of utilizing latent space for deep analysis and insightful understanding of different application scenarios. First, we investigate a machine learning solution with latent space representation for a vehicle relay problem. We propose a deep reinforcement learning framework in the LTE-V scenario for virtually selecting the vehicle relay node. Specifically, the structure of the deep network follows the deep Q-networks scheme. We build two deep Q-networks working as the actor-critic mechanism, which helps a more stable training. For both actor networks and critic networks, a few convolution layers are stacked for a latent space representation. Then, a fully-connected layer is built to map the latent space into the Q-value space. Q values represent the value of the action under a certain state. The Deep Q-networks agent will move the virtual relay according to the Q-value. Each move arises a reward (or punishment). We train the agent with this feedbacks from the environment. With the proposed methods, we can achieve an improvement in the relay performance by comparing it with other baseline methods.

Secondly, we apply the machine learning model with latent space representation to classifying cutting volume into different levels. In this model, we import well-preprocessed video frames in a real-time manner streaming. By implementing automatic ROI selection, we are able to let the model focus on a certain location in the video frames where cuttings exist. We propose a model named VGG16-adaptation for solving the cutting volume classification task. By the above preprocessing and model adaptation, we successfully convert the raw input into the latent space which contains the information needed for the classifier. The evaluation results validate that the capability of our proposed model for delivering promising results in a real-time manner.

Thirdly, we implement latent space manipulation for generating synthetic image patterns to finally assist the invasive grass detection. In this task, the raw input is the image set obtained by UAVs, which is not suitable for processing directly by the learning model. Thus, we crop the large images into small patches and let model work on small patches first, then merge them back into large full-size images. In practice, the volume of the data set is limited. The data origins from several national parks, but still not enough to fulfill a model with the best capability. Inspired by this limitation, we propose to use generative models to generate synthetic data patterns. By experiments, we find out this data augmentation works best when the synthetic patterns are of high fidelity. Thus, we convert the original continuous latent space into a discrete latent space which contains more information about the synthetic generation. We blend these high-fidelity synthetic grass patterns back into the original background images. By this manipulation, we find out this data augmentation scheme with discrete latent space representation helps the detection model best.

Finally, we bring the latent space manipulation technique for a domain adaptation problem in deep learning-based seismic data inversion. The learning models in the seismic data inversion scenario often encounter different data set by exploring different reservoirs. The demand exists that we need the model which has been well-trained on the original data set to be migrated to the data set we never saw before. To achieve so, we need a domain adaptation technique to help reduce the gap between the source domain and the target domain. We propose a learning framework that adopts optimal transport as a tool for domain adaptation. This framework jointly learns the inputs, labels, and embedded vectors in the latent space, which are all considered as components of the final loss function. The finding in the experiment is the adapted target model can generate the desired velocity output without feeding labels in the target domain. Meanwhile, the target model can also work well on the source domain without forgetting the knowledge previously learned. On the other hand, the source model trained on the source data set works well only on the source domain, which shows the feasibility of the adaptation of the target model.

6.2 Future Works

- *Future improvement on deep domain adaptation:* Deep domain adaptation [154] is an active research area today and it will bring a significant improvement in the generalization capability of the learning models. From the experiment of our work, we still find several limitations and unknown knowledge which can be further explored:

  - Define the task relatedness: In our study, we find out that the source domain and target domain should be well-designed in order to be adapted. The domain adaptation technique will not work on ill-defined data sets. The standard or at least a relative general rule to measure the task relatedness should be further studied.
  - From one-step adaptation to multi-steps adaptation: In our study, we only implement a one-step adaptation, which works on a simple scenario where the source data is adapted to a simple noisy data set. We can further investigate a more complex scenario where the adaptation gap is larger. In a more complex scenario, a multi-steps mechanism may be needed to get the knowledge from multiple pre-trained models.
  - Achieve better performance on regression: the deep domain adaptation is mostly studied on a classification related to a deep discriminative model. When it comes to the regression problem, we have to consider the discrepancy to be obtained from a more complex manifold which makes the model update more difficult. We have to face the challenge that we might need to convert the origi-

nal large dimension (e.g., an image scale) to a smaller dimension by using some transformation such as discriminator networks in GAN.

- *Future exploration for solving unlabeled seismic data inversion by self-supervised learning:* Self-supervised learning [155] [156] is regarded as a type of unsupervised learning since it needs no label but extracts "labels" implicitly from the input itself. The basic idea of self-supervised learning contains two main procedures. First, we need a pretext model to enrich the latent space representation. Second, from the enriched representation, we train a downstream model for our final goal. This two-steps process is similar to the transfer learning mechanism. For our specific problem, we can detail our potential solution to answer the following concerns:

  - Choose a pretext task: Since the overall workflow is a two-steps procedure, the quality of the upstream process is fundamental to the performance of the final goal. Thus, it requires us to further investigate the insight physics of the seismic data. For example, the seismic data is in the time domain whose sequence can be learned first. The time-series information might help to construct the structural contents inside the subsurface model.
  - Fine-tuning for the downstream task: This phase should be considered as a transfer learning process, which requires much carefulness that the pretext model weights are not getting hurt. There are a few techniques that can be tried for a stable and effective transfer of the pretext weights to downstream, such as gradual unfreezing, discriminative learning rates, and one-cycle training, etc.
  - Consistency loss for multiple pretext representation: Same as the domain adaptation, we might incorporate multiple pretext models so that a more meaningful and generalized latent space representation can be learned for the downstream. For multiple pretext models, we hope the intermediate outputs should contain similarity in the features. We can add a consistency loss to measure the distance

between those intermediate outputs so as to guarantee those models are trained in the same direction.

References

[1] C. Cortes and V. Vapnik, “Support-Vector Networks,” *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[2] P. Krähenbühl and V. Koltun, “Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials,” in *Advances in Neural Information Processing Systems 24*, J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, Eds., Granada, Spain, Dec. 2011.

[3] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. Torr, “Conditional Random Fields as Recurrent Neural Networks,” in *International Conference on Computer Vision*, Santiago, Chile, Dec. 2015.

[4] I. J. Goodfellow, “NIPS 2016 tutorial: Generative Adversarial Networks,” *CoRR*, vol. abs/1701.00160, 2017. [Online]. Available: http://arxiv.org/abs/1701.00160

[5] D. H. Ballard, “Modular Learning in Neural Networks,” in *Proceedings of the Sixth National Conference on Artificial Intelligence*, Seattle, WA, Jul. 1987.

[6] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, “Playing Atari with Deep Reinforcement Learning,” *arXiv preprint arXiv:1312.5602*, 2013.

[7] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, “Human-Level Control Through Deep Reinforcement Learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, Jan. 2015.

[8] H. Huang, Z. Li, R. He, Z. Sun, and T. Tan, “IntroVAE: Introspective Variational Autoencoders for Photographic Image Synthesis,” in *Proceedings of the 32nd Inter-*

national Conference on Neural Information Processing Systems, Montréal, Canada, Dec. 2018.

[9] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, Feb. 2015.

[10] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998.

[11] B. B. Damodaran, B. Kellenberger, R. Flamary, D. Tuia, and N. Courty, “DeepJDOT: Deep Joint Distribution Optimal Transport for Unsupervised Domain Adaptation,” in *ECCV - 15th European Conference on Computer Vision*, Munich, Germany, Sep. 2018.

[12] S. Chen, J. Hu, Y. Shi, and L. Zhao, “LTE-V: A TD-LTE-Based V2X Solution for Future Vehicular Network,” *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 997–1005, Sep. 2016.

[13] G. Araniti, C. Campolo, M. Condoluci, A. Iera, and A. Molinaro, “LTE for Vehicular Networking: a Survey,” *IEEE Communications Magazine*, vol. 51, no. 5, pp. 148–157, May 2013.

[14] N. P. Affairs, “USDOT Releases 2016 Fatal Traffic Crash Data,” https://www.nhtsa.gov/press-releases/usdot-releases-2016-fatal-traffic-crash-data.

[15] IEEE, “IEEE standard for information technology– local and metropolitan area networks– specific requirements– part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications amendment 6: Wireless access in vehicular environments,” *IEEE Std 802.11p-2010*, pp. 1–51, Jul. 2010.

[16] C. H. Lee, K. G. Lim, B. L. Chua, R. K. Y. Chin, and K. T. K. Teo, “Progressing toward urban topology and mobility trace for vehicular ad hoc network (vanet),” in *IEEE Conference on Open Systems (ICOS)*, Langkawi, Malaysia, Oct. 2016.

[17] M. Amadeo, C. Campolo, and A. Molinaro, “Enhancing IEEE 802.11p/wave to provide infotainment applications in vanets,” *Ad Hoc Networks*, vol. 10, no. 2, pp. 253 – 269, Mar. 2012.

[18] Z. Hameed Mir and F. Filali, “LTE and IEEE 802.11p for vehicular networking: a performance evaluation,” *Journal on Wireless Communications and Networking*, vol. 2014, no. 1, pp. 1–15, May 2014.

[19] N. H. T. S. Administration., “2012 Motor Vehicle Crashes: Overview,” https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/811856.

[20] S. Zuther and K. Dietmayer, “360 ° environment sensing and signal processing for an automotive pre-crash application,” in *IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, Pune, India, Nov. 2009.

[21] A. Buchenscheit, F. Schaub, F. Kargl, and M. Weber, “A vanet-based emergency vehicle warning system,” in *IEEE Vehicular Networking Conference (VNC)*, Tokyo, Japan, Oct. 2009.

[22] V. Milanés, S. E. Shladover, J. Spring, C. Nowakowski, H. Kawazoe, and M. Nakamura, “Cooperative adaptive cruise control in real traffic situations,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 1, pp. 296–305, Feb. 2014.

[23] C. Desjardins and B. Chaib-draa, “Cooperative adaptive cruise control: A reinforcement learning approach,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 1248–1260, Dec. 2011.

[24] J. Fietkau, “A street traffic simulation example project for MPI-based parallel computing in Python,” https://github.com/jfietkau/Streets4MPI.

[25] H. van Hasselt, A. Guez, and D. Silver, “Deep Reinforcement Learning with Double Q-learning,” *CoRR*, vol. abs/1509.06461, 2015.

[26] D. Tian, J. Zhou, Z. Sheng, M. Chen, Q. Ni, and V. C. M. Leung, “Self-Organized Relay Selection for Cooperative Transmission in Vehicular Ad-hoc Networks,” *IEEE Transactions on Vehicular Technology*, vol. 66, no. 10, pp. 9534–9549, Oct. 2017.

[27] M. F. Feteiha and H. S. Hassanein, “Enabling Cooperative Relaying VANET Clouds Over LTE-A Networks,” *IEEE Transactions on Vehicular Technology*, vol. 64, no. 4, pp. 1468–1479, Apr. 2015.

[28] W. Song and X. Tao, “Analysis of a Location-Aware Probabilistic Strategy for Opportunistic Vehicle-to-Vehicle Relay,” in *2017 IEEE 86th Vehicular Technology Conference (VTC-Fall)*, Toronto, Canada, Sep. 2017.

[29] D. Wang, P. Ren, Q. Du, L. Sun, and Y. Wang, “Security Provisioning for MISO Vehicular Relay Networks via Cooperative Jamming and Signal Superposition,” *IEEE Transactions on Vehicular Technology*, vol. 66, no. 12, pp. 10 732–10 747, Dec. 2017.

[30] M. Seyfi, S. Muhaidat, J. Liang, and M. Uysal, “Relay Selection in Dual-Hop Vehicular Networks,” *IEEE Signal Processing Letters*, vol. 18, no. 2, pp. 134–137, Feb. 2011.

[31] X. Tang, P. Ren, and Z. Han, “Hierarchical Competition as Equilibrium Program With Equilibrium Constraints Towards Security-Enhanced Wireless Networks,” *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 7, pp. 1564–1578, Jul. 2018.

[32] Q. Du, H. Song, and X. Zhu, "Social-Feature Enabled Communications Among Devices Toward the Smart IoT Community," *IEEE Communications Magazine*, vol. 57, no. 1, pp. 130–137, Jan. 2019.

[33] L. Ma, X. Huo, X. Zhao, and G. D. Zong, "Observer-based adaptive neural tracking control for output-constrained switched MIMO nonstrict-feedback nonlinear systems with unknown dead zone," *Nonlinear Dynamics*, vol. 99, no. 2, pp. 1019–1036, Jan. 2020.

[34] Y. Ge, S. Wen, Y. H. Ang, and Y. C. Liang, "Optimal Relay Selection in IEEE 802.16j Multihop Relay Vehicular Networks," *IEEE Transactions on Vehicular Technology*, vol. 59, no. 5, pp. 2198–2206, Jun. 2010.

[35] R. Chai, Y. Qin, S. Peng, and Q. Chen, "Transmission Performance Evaluation and Optimal Selection of Relay Vehicles in VANETs," in *IEEE Wireless Communications and Networking Conference (WCNC)*, San Francisco, CA, Mar. 2017.

[36] B. Yang, X. Sun, R. Chai, L. Cai, and X. Yang, "Game Theory based relay vehicle selection for VANET," in *IEEE 24th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, London, UK, Sep. 2013.

[37] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, Lake Tahoe, Nevada, Dec. 2012.

[38] X. Ma, Z. Tao, Y. Wang, H. Yu, and Y. Wang, "Long short-term memory neural network for traffic speed prediction using remote microwave sensor data," *Transportation Research Part C: Emerging Technologies*, vol. 54, no. 0, pp. 187 – 197, May 2015.

[39] X. Xiong, J. Wang, F. Zhang, and K. Li, “Combining Deep Reinforcement Learning and Safety Based Control for Autonomous Driving,” *CoRR*, vol. abs/1612.00147, 2016.

[40] H. V. Hasselt, A. Guez, and D. Silver, “Deep Reinforcement Learning with Double Q-Learning,” in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. Phoenix, Arizona: AAAI Press, Feb. 2016.

[41] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, “SUMO–simulation of urban mobility: an overview,” in *Proceedings of SIMUL, The Third International Conference on Advances in System Simulation*, Barcelona, Spain, Oct. 2011.

[42] O. Foundation, “OpenStreetMap,” https://www.openstreetmap.org.

[43] M. Rondinone, J. Gozalvez, J. Leguay, and V. Conan, “Exploiting context information for V2X dissemination in vehicular networks,” in *IEEE 14th International Symposium on ”A World of Wireless, Mobile and Multimedia Networks” (WoWMoM)*, Madrid, Spain, Jun. 2013.

[44] X. Zhu, Y. Li, D. Jin, and J. Lu, “Contact-Aware Optimal Resource Allocation for Mobile Data Offloading in Opportunistic Vehicular Networks,” *IEEE Transactions on Vehicular Technology*, vol. 66, no. 8, pp. 7384–7399, Aug. 2017.

[45] C. Ye, P. Wang, C. Wang, and F. Liu, “Mobility management for LTE-based heterogeneous vehicular network in V2X scenario,” in *2nd IEEE International Conference on Computer and Communications (ICCC)*, Chengdu, China, Oct. 2016.

[46] K. Lee, Y. Yi, J. Jeong, H. Won, I. Rhee, and S. Chong, “Max-Contribution: On Optimal Resource Allocation in Delay Tolerant Networks,” in *IEEE INFOCOM*, San Diego, CA, Mar. 2010.

[47] H. Zhu, L. Fu, G. Xue, Y. Zhu, M. Li, and L. M. Ni, “Recognizing Exponential Inter-Contact Time in VANETs,” in *IEEE INFOCOM*, San Diego, CA, USA, Mar. 2010.

[48] J. Zhou, D. Gao, and D. Zhang, “Moving Vehicle Detection for Automatic Traffic Monitoring,” *IEEE Transactions on Vehicular Technology*, vol. 56, no. 1, pp. 51–59, Jan. 2007.

[49] R. Arora, A. Basu, P. Mianjy, and A. Mukherjee, “Understanding Deep Neural Networks with Rectified Linear Units,” in *International Conference on Learning Representations*, Vancouver, Canada, Apr. 2018.

[50] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” *CoRR*, vol. abs/1509.02971, 2015.

[51] W. A. Graves and M. D. Rowe, “Down hole cuttings analysis,” Nov. 2014, uS Patent App. 14/363,264.

[52] V. Parmeshwar, J. Orban, and B. B. Arefi, “Shale shaker imaging system,” Jun. 8 2017, uS Patent App. 14/982,510.

[53] P. A. Torrione, “System and method for estimating cutting volumes on shale shakers,” Mar. 2017, uS Patent App. 15/251,940.

[54] J. Dahl, C. J. Morgan, and M. E. Gillen, “In-situ downhole cuttings analysis,” Apr. 11 2017, uS Patent 9,617,851.

[55] I. R. Guilherme, A. N. Marana, J. P. Papa, G. Chiachia, L. C. Afonso, K. Miura, M. V. Ferreira, and F. Torres, “Petroleum well drilling monitoring through cutting image analysis and artificial intelligence techniques,” *Engineering Applications of Artificial Intelligence*, vol. 24, no. 1, pp. 201–207, 2011.

[56] A. N. Marana, G. Chiachia, I. R. Guilherme, J. P. Papa, K. Miura, M. V. Ferreira, and F. Torres, "An intelligent system for petroleum well drilling cutting analysis," in *2009 International Conference on Adaptive and Intelligent Systems*. IEEE, 2009, pp. 37–42.

[57] N. Dhanachandra, K. Manglem, and Y. J. Chanu, "Image segmentation using K-means clustering algorithm and subtractive clustering algorithm," *Procedia Computer Science*, vol. 54, pp. 764–771, 2015.

[58] L. Yang, S. Yang, P. Jin, and R. Zhang, "Semi-supervised hyperspectral image classification using spatio-spectral Laplacian support vector machine," *IEEE Geoscience and Remote Sensing Letters*, vol. 11, no. 3, pp. 651–655, 2013.

[59] P. Du, A. Samat, B. Waske, S. Liu, and Z. Li, "Random forest and rotation forest for fully polarized SAR image classification using polarimetric and spatial features," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 105, pp. 38–53, 2015.

[60] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[61] S. Routray, A. K. Ray, and C. Mishra, "Analysis of various image feature extraction methods against noisy image: SIFT, SURF and HOG," in *2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT)*. IEEE, 2017, pp. 1–5.

[62] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[63] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[64] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

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

[66] F. L.-F. Wauthier, "Learning from Subsampled Data: Active and Randomized Strategies," Ph.D. dissertation, UC Berkeley, 2013.

[67] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," *arXiv preprint arXiv:1607.08022*, 2016.

[68] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[69] R. Arora, A. Basu, P. Mianjy, and A. Mukherjee, "Understanding Deep Neural Networks with Rectified Linear Units," *CoRR*, vol. abs/1611.01491, 2016. [Online]. Available: http://arxiv.org/abs/1611.01491

[70] L. Dogaru, "The Importance of Environmental Protection and Sustainable Development," *Procedia - Social and Behavioral Sciences*, vol. 93, pp. 1344 – 1348, Oct. 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877042813034861

[71] J. Sandino, F. Gonzalez, K. Mengersen, and K. J. Gaston, "UAVs and Machine Learning Revolutionising Invasive Grass and Vegetation Surveys in Remote Arid Lands," *Sensors*, vol. 18, no. 2, pp. 1–13, Feb. 2018. [Online]. Available: https://www.mdpi.com/1424-8220/18/2/605

[72] A. K. Maurya, D. Singh, and K. P. Singh, “Development of Fusion Approach for Estimation of Vegetation Fraction Cover with Drone and Sentinel-2 Data,” in *IEEE International Geoscience and Remote Sensing Symposium*, Valencia, Spain, Jul. 2018.

[73] T. Kumpumäki and T. Lipping, “Effects of shadow correction on vegetation and land cover classification from high resolution aerial images,” in *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, Beijing, China, Jul. 2016.

[74] L. C. G. David and A. H. Ballado, “Vegetation Indices and Textures in Object-Based Weed Detection from UAV Imagery,” in *IEEE International Conference on Control System, Computing and Engineering (ICCSCE)*, Penang, Malaysia, Nov. 2016.

[75] J. Sandino and F. Gonzalez, “A Novel Approach for Invasive Weeds and Vegetation Surveys Using UAS and Artificial Intelligence,” in *International Conference on Methods Models in Automation Robotics (MMAR)*, Miedzyzdroje, Poland, Aug. 2018.

[76] G. J. Hay and G. Castilla, “Geographic Object-Based Image Analysis (GEOBIA): A New Name for a New Discipline,” *Lecture Notes in Geoinformation and Cartography*, pp. 75–89, Jan. 2008.

[77] L. Pibre, M. Chaumon, G. Subsol, D. Lenco, and M. Derras, “How to Deal with Multi-Source Data for Tree Detection Based on Deep Learning,” in *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, Montreal, Canada, Nov. 2017.

[78] W. Li, H. Fu, and L. Yu, “Deep Convolutional Neural Network Based Large-Scale Oil Palm Tree Detection for High-Resolution Remote Sensing Images,” in *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, Fort Worth, Texas, Jul. 2017.

[79] M. Dahmane, S. Foucher, M. Beaulieu, F. Riendeau, Y. Bouroubi, and M. Benoit, "Object Detection in Pleiades Images Using Deep Features," in *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, Beijing, China, Jul. 2016.

[80] Y. Bengio, A. Courville, and P. Vincent, "Representation Learning: A Review and New Perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.

[81] Y. LeCun, K. Kavukcuoglu, and C. Farabet, "Convolutional Networks and Applications in Vision," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, Paris, France, May 2010.

[82] Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu, "Object Detection with Deep Learning: A Review," *arXiv preprint arXiv:1807.05511*, 2018.

[83] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. Lecun, "Overfeat: Integrated Recognition, Localization and Detection Using Convolutional Networks," in *International Conference on Learning Representations (ICLR2014), CBLS*, Banff, Canada, Apr. 2014.

[84] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *Advances in Neural Information Processing Systems*, vol. 39, no. 6, pp. 1137–1149, Jun. 2016.

[85] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, Las Vegas, NV, Jun. 2016.

[86] S. Ren, K. He, R. Girshick, and J. Sun, "Applying Faster R-CNN for Object Detection on Malaria Images," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops(CVPRW)*, Honolulu, HI, Jul. 2017.

[87] D. Ribli, A. Horváth, Z. Unger, P. Pollner, and I. Csabai, “Detecting and Classifying Lesions in Mammograms with Deep Learning,” *Scientific Reports*, vol. 8, no. 1, pp. 1–7, Mar. 2018.

[88] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, Jun. 2016.

[89] K. Chen, X. Zhou, W. Xiang, and Q. Zhou, “Data Augmentation Using GAN for Multi-Domain Network-Based Human Tracking,” in *IEEE Visual Communications and Image Processing (VCIP)*, Taichung, Taiwan, Dec. 2018.

[90] G. Wang, W. Kang, Q. Wu, Z. Wang, and J. Gao, “Generative Adversarial Network (GAN) Based Data Augmentation for Palmprint Recognition,” in *Digital Image Computing: Techniques and Applications (DICTA)*, Canberra, Australia, Dec. 2018.

[91] S. J. Pan and Q. Yang, “A Survey on Transfer Learning,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[92] A. Mikołajczyk and M. Grochowski, “Data Augmentation for Improving Deep Learning in Image Classification Problem,” in *International Interdisciplinary PhD Workshop (IIPhDW)*, Sczcecin, Poland, May 2018.

[93] G. E. Hinton and R. R. Salakhutdinov, “Reducing the Dimensionality of Data with Neural Networks,” *Science*, vol. 313, no. 5786, pp. 504–507, Jul. 2006. [Online]. Available: https://science.sciencemag.org/content/313/5786/504

[94] G. E. Hinton, A. Krizhevsky, and S. Wang, “Transforming Auto-Encoders,” in *ICANN*, Heidelberg, Germany, Jun. 2011, pp. 44–51.

[95] J. Li, A. Madry, J. Peebles, and L. Schmidt, “On the Limitations of First-Order Approximation in GAN Dynamics,” in *ICML*, Stockholm, Sweden, Jul. 2018.

[96] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther, “Autoencoding Beyond Pixels Using a Learned Similarity Metric,” in *Proceedings of the 33rd International Conference on International Conference on Machine Learning*, New York, NY, Jun. 2016. [Online]. Available: http://dl.acm.org/citation.cfm?id=3045390.3045555

[97] J. Lucas, G. Tucker, R. Grosse, and M. Norouzi, “Understanding posterior collapse in generative latent variable models,” in *ICLR 2019 Workshop*, New Orleans, LA, May 2019.

[98] A. van den Oord, O. Vinyals, and K. Kavukcuoglu, “Neural Discrete Representation Learning,” in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, Long Beach, CA, Dec. 2017.

[99] D. Kingma and M. Welling, “Auto-Encoding Variational Bayes,” in *International Conference on Learning Representations (ICLR)*, Banff, Canada, Apr. 2014.

[100] J. Mairal, J. Ponce, G. Sapiro, A. Zisserman, and F. R. Bach, “Supervised Dictionary Learning,” in *NIPS*, Vancouver, Canada, Dec. 2008.

[101] Y. Bengio, N. Léonard, and A. C. Courville, “Estimating or Propagating Gradients through Stochastic Neurons for Conditional Computation,” *CoRR*, vol. abs/1308.3432, 2013. [Online]. Available: http://arxiv.org/abs/1308.3432

[102] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu, “Pixel Recurrent Neural Networks,” *CoRR*, vol. abs/1601.06759, 2016. [Online]. Available: http://arxiv.org/abs/1601.06759

[103] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask R-CNN,” in *IEEE International Conference on Computer Vision (ICCV)*, Venice, Italy, Oct. 2017.

[104] G. TensorFlow., “How to Retrain an Image Classifier for New Categories,” https://www.tensorflow.org/hub/tutorials/image_retraining.

[105] Z. Huang, Z. Pan, and B. Lei, “Transfer Learning with Deep Convolutional Neural Network for SAR Target Classification with Limited Labeled Data,” *Remote Sensing*, vol. 9, no. 9, pp. 1–21, Aug. 2017. [Online]. Available: http://www.mdpi.com/2072-4292/9/9/907

[106] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft COCO: Common Objects in Context,” in *The European Conference on Computer Vision (ECCV)*, Zurich, Switzerland, Sep. 2014.

[107] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, “GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium,” in *Conference on Neural Information Processing Systems (NIPS)*, Long Beach, CA, Dec. 2017.

[108] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going Deeper with Convolutions,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, Jun. 2015.

[109] P. Soupios, I. Akca, P. Mpogiatzis, A. T. Basokur, and C. Papazachos, “Applications of hybrid genetic algorithms in seismic tomography,” *Journal of Applied Geophysics*, vol. 75, no. 3, pp. 479 – 489, Nov. 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0926985111001790

[110] Q. Guo, H. Zhang, F. Han, and Z. Shang, “Prestack Seismic Inversion Based on Anisotropic Markov Random Field,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 2, pp. 1069–1079, Feb. 2018.

[111] Z. Gao, Z. Pan, C. Zuo, J. Gao, and Z. Xu, “An Optimized Deep Network Representation of Multimutation Differential Evolution and its Application in Seismic Inversion,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 7, pp. 4720–4734, Jul. 2019.

[112] Z. Wu and T. Alkhalifah, “Simultaneous inversion of the background velocity and the perturbation in full-waveform inversion,” *GEOPHYSICS*, vol. 80, no. 6, pp. R317–R329, Sep. 2015.

[113] D. Dagnino, V. Sallarès, and C. R. Ranero, “Waveform-Preserving Processing Flow of Multichannel Seismic Reflection Data for Adjoint-State Full-Waveform Inversion of Ocean Thermohaline Structure,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 3, pp. 1615–1625, Nov. 2018.

[114] Y. Lin, L. Huang, and Z. Zhang, “Ultrasound waveform tomography with the total-variation regularization for detection of small breast tumors,” in *Medical Imaging 2012: Ultrasonic Imaging, Tomography, and Therapy*, J. G. Bosch and M. M. Doyley, Eds., vol. 8320, International Society for Optics and Photonics. San Diego, CA: SPIE, Feb. 2012, pp. 13–21. [Online]. Available: https://doi.org/10.1117/12.910765

[115] J. Virieux and S. Operto, “An overview of full-waveform inversion in exploration geophysics,” *Geophysics*, vol. 74, no. 6, p. WCC1 (online), Dec. 2009. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00457989

[116] A. Guitton, G. Ayeni, and E. Díaz, “Constrained full-waveform inversion by model reparameterization,” *GEOPHYSICS*, vol. 77, no. 2, pp. R117–R127, Mar. 2012.

[117] Y. Ma and D. Hale, “Quasi-newton full-waveform inversion with a projected hessian matrix,” *GEOPHYSICS*, vol. 77, no. 5, pp. R207–R216, Aug. 2012.

[118] K. T. Tran, M. McVay, M. Faraone, and D. Horhota, “Sinkhole detection using 2d full seismic waveform tomography,” *GEOPHYSICS*, vol. 78, no. 5, pp. R175–R183, Aug. 2013.

[119] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[120] P. Jiang, F. Gu, Y. Wang, C. Tu, and B. Chen, “DifNet: Semantic Segmentation by Diffusion Networks,” in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, Montréal, Canada, Dec. 2018.

[121] T. Wang, M. Liu, J. Zhu, A. Tao, J. Kautz, and B. Catanzaro, “High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, Jun. 2018.

[122] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, “Show and tell: A neural image caption generator,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, Oct. 2015.

[123] T. Xu, P. Zhang, Q. Huang, H. Zhang, Z. Gan, X. Huang, and X. He, “AttnGAN: Fine-Grained Text to Image Generation with Attentional Generative Adversarial Networks,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, Jun. 2018, pp. 1316–1324.

[124] J. Zhu, T. Park, P. Isola, and A. A. Efros, “Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks,” in *2017 IEEE International Conference on Computer Vision (ICCV)*, Venice, Italy, Oct. 2017.

[125] Z. Yi, H. Zhang, P. Tan, and M. Gong, “DualGAN: Unsupervised Dual Learning for Image-to-Image Translation,” in *IEEE International Conference on Computer Vision (ICCV)*, Venice, Italy, Oct. 2017.

[126] V. Jain and S. Seung, “Natural Image Denoising with Convolutional Networks,” in *Advances in Neural Information Processing Systems 21*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds. Vancouver, Canada: Curran Associates, Inc., Dec. 2008, pp. 769–776. [Online]. Available: http://papers.nips.cc/paper/3506-natural-image-denoising-with-convolutional-networks.pdf

[127] D. Eigen, D. Krishnan, and R. Fergus, “Restoring an Image Taken through a Window Covered with Dirt or Rain,” in *IEEE International Conference on Computer Vision*, Sydney, Australia, Dec. 2013.

[128] M. Araya-Polo, J. Jennings, A. Adler, and T. Dahlke, “Deep-learning tomography,” *The Leading Edge*, vol. 37, no. 1, pp. 58–66, Dec. 2018.

[129] J. Kim, J. K. Lee, and K. M. Lee, “Accurate Image Super-Resolution Using Very Deep Convolutional Networks,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, Jun. 2016.

[130] W. Li, F. Liu, L. Jiao, and F. Hu, “Multi-scale residual reconstruction neural network with non-local constraint,” *IEEE Access*, vol. 7, pp. 70 910–70 918, May 2019.

[131] K. Gregor and Y. LeCun, “Learning Fast Approximations of Sparse Coding,” in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ser. ICML’10. Madison, WI, USA: Omnipress, Jun. 2010.

[132] K. Zeng, J. Yu, R. Wang, C. Li, and D. Tao, “Coupled Deep Autoencoder for Single Image Super-Resolution,” *IEEE Transactions on Cybernetics*, vol. 47, no. 1, pp. 27–37, Nov. 2017.

[133] Y. Wu, Y. Lin, and Z. Zhou, *Inversionet: Accurate and efficient seismic-waveform inversion with convolutional neural networks*, Anaheim, CA, Aug. 2018, pp. 2096–2100.

[134] S. Li, B. Liu, Y. Ren, Y. Chen, S. Yang, Y. Wang, and P. Jiang, "Deep-Learning Inversion of Seismic Data," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 58, no. 3, pp. 2135–2149, Dec. 2020.

[135] W. Lewis and D. Vigh, "Deep learning prior models from seismic images for full-waveform inversion," in *SEG Technical Program Expanded Abstracts 2017*, Houston, TX, Sep. 2017.

[136] Richardson, Alan, "Seismic full-waveform inversion using deep learning tools and techniques," 2018. [Online]. Available: http://arxiv.org/abs/1801.07232

[137] A. Van Den Oord, O. Vinyals *et al.*, "Neural discrete representation learning," in *Advances in Neural Information Processing Systems*, Long Beach, CA, Dec. 2017.

[138] A. Razavi, A. van den Oord, and O. Vinyals, "Generating diverse high-fidelity images with vq-vae-2," in *Advances in Neural Information Processing Systems*, Vancouver, Canada, Dec. 2019.

[139] K. H. Jin, M. T. McCann, E. Froustey, and M. Unser, "Deep Convolutional Neural Network for Inverse Problems in Imaging," *IEEE Transactions on Image Processing*, vol. 26, no. 9, pp. 4509–4522, Jun. 2017.

[140] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *Lecture Notes in Computer Science*, vol. 9351, Oct. 2015, pp. 234–241.

[141] C. Villani, *Optimal transport: old and new*. Springer Science & Business Media, 2008, vol. 338.

[142] Cayley, "On Monge's "Mémoire sur la Théorie des Déblais et des Remblais."," *Proceedings of the London Mathematical Society*, vol. s1-14, no. 1, pp. 139–143, Nov. 1882.

[143] L. Kantorovitch, "On the Translocation of Masses," *Management Science*, vol. 5, no. 1, pp. 1–4, Oct. 1958.

[144] L. Rueschendorf, "The Wasserstein Distance and Approximation Theorems," *Probability Theory and Related Fields*, vol. 70, pp. 117–129, Mar. 1985.

[145] N. Courty, R. Flamary, D. Tuia, and A. Rakotomamonjy, "Optimal Transport for Domain Adaptation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 9, pp. 1853–1865, Oct. 2017.

[146] N. Courty, R. Flamary, and D. Tuia, "Domain Adaptation with Regularized Optimal Transport," in *Proceedings of the 2014th European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part I*, Berlin, Heidelberg, Sep. 2014.

[147] M. Perrot, N. Courty, R. Flamary, and A. Habrard, "Mapping Estimation for Discrete Optimal Transport," in *Advances in Neural Information Processing Systems (NeurIPS)*, Barcelona, Spain, Dec. 2016, pp. 4197–4205.

[148] I. Redko, A. Habrard, and M. Sebban, "Theoretical Analysis of Domain Adaptation with Optimal Transport," *Machine Learning and Knowledge Discovery in Databases*, pp. 737–753, Jan. 2017.

[149] Y. Yang, B. Engquist, J. Sun, and B. F. Hamfeldt, "Application of optimal transport and the quadratic Wasserstein metric to full-waveform inversion," *GEOPHYSICS*, vol. 83, no. 1, pp. R43–R62, Jan. 2018.

[150] L. Métivier, R. Brossier, Q. Mérigot, E. Oudet, and J. Virieux, "Measuring the misfit between seismograms using an optimal transport distance: application to full waveform inversion," *Geophysical Journal International*, vol. 205, no. 1, pp. 345–377, Feb. 2016.

[151] L. Métivier, R. Brossier, Q. Mérigot, E. Oudet, and J. Virieux, "An optimal transport approach for seismic tomography: application to 3D full waveform inversion," *Inverse Problems*, vol. 32, no. 11, pp. 1–37, Sep. 2016.

[152] N. Courty, R. Flamary, A. Habrard, and A. Rakotomamonjy, "Joint Distribution Optimal Transportation for Domain Adaptation," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, Red Hook, NY, Dec. 2017.

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

[154] B. Holländer, "Deep domain adaptation in computer vision." [Online]. Available: https://towardsdatascience.com/deep-domain-adaptation-in-computer-vision-8da398d3167f

[155] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. Hinton, "Big Self-Supervised Models are Strong Semi-Supervised Learners," 2020.

[156] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum Contrast for Unsupervised Visual Representation Learning," 2019.