IDENTIFICATION OF UNKNOWN LANDSCAPE TYPES USING CNN TRANSFER LEARNING

by

Ashish Sharma

A thesis

submitted in partial fulfillment

of the requirements for the degree of

Master of Science in Computer Science

Boise State University

August 2017

BOISE STATE UNIVERSITY GRADUATE COLLEGE

DEFENSE COMMITTEE AND FINAL READING APPROVALS

of the thesis submitted by

Ashish Sharma

Thesis Title: Identification of Unknown Landscape Types using CNN Transfer Learning

Date of Final Oral Examination: 19th June 2017

The following individuals read and discussed the thesis submitted by student Ashish Sharma, and they evaluated the presentation and response to questions during the final oral examination. They found that the student passed the final oral examination.

Edoardo Serra, Ph.D.	Chair, Supervisory Committee
Tim Andersen, Ph.D.	Member, Supervisory Committee
Francesca Spezzano, Ph.D.	Member, Supervisory Committee

The final reading approval of the thesis was granted by Edoardo Serra, Ph.D., Chair of the Supervisory Committee. The thesis was approved by the Graduate College.

Dedicated to my parents and my siblings, Ashok and Anisha.

ACKNOWLEDGMENTS

First of all, I would like to thank Dr. Edoardo Serra who has continuously bolstered my learning experience since the beginning of my first semester. Through continuous guidance, and uncountable productive meetings in the past two years, he has helped me identify my weaknesses and push my abilities beyond the boundaries. Dr. Serra's passion and zeal for solving difficult problems are very influencing. Settling down with one solution has never been his choice. His critical thinking process while tackling research problems has enormously helped me understand the essence of the grueling process of scientific research, rather than just solving them. I'm very grateful to Dr. Serra not only for teaching me the application of data science and machine learning in different field of research domains, but also for helping me understand my own capabilities in tackling the problems in a most scientific way possible.

I would also like to extend my gratitude towards a bunch of great professors at Boise State University. This list includes Dr. Tim Andersen, Dr. Francesca Spezzano, Dr. Amit Jain, Dr. Sole Pera and Dr. Gaby Dagher who have influenced my learning experience through their amazing teaching abilities, respectful demeanor and continuous encouragement towards excellence. I would also like to thank my colleagues Mikel Joaristi, Axel Magnuson, Oxana Korzh and Haritha Akella for being a great support throughout my academic journey to a masters degree. All of their continuous encouragement, mentor-ship and positive attitude towards learning have

helped me grow as a better individual, professionally and in person.

Next, I would like to thank my parents for believing in me and letting me chase my dreams. I thank them for teaching me the essence of continuous hardship, mutual respect, and courage to be patient during difficult times. I would also like to thank all my friends from Nepal for being amazing emotional support throughout this journey.

There are many people who have directly and indirectly helped me learn, grow and become capable of completing my masters degree with this thesis during the past two years in and outside the Boise State University. Finally, I would like to thank everyone of them for being kind and for providing helping hands in need.

Thank you all!

ABSTRACT

Unknown image type identification is the problem of identifying unknown types of images from the set of already provided images that are considered to be known, where the known and unknown sets represent different content types. Solving this problem has a lot of security applications such as suspicious object detection during baggage scanning at airport customs, border protection via remote sensing, cancer detection, weather and disaster monitoring, etc. In this thesis, we focus on identification of unknown landscape images. This application has a huge relevance to the context of a smart nation where it can be applied to major national security tasks such as monitoring the borders or the detection of unknown and potentially dangerous landscapes in critical locations.

We propose effective semi-supervised novelty detection approaches for the unknown image type identification problem using Convolutional Neural Network (CNN) Transfer Learning. Recently, the CNN Transfer Learning approach has been very successful in various visual recognition tasks especially in cases where large training data is not available. Our main idea is to use pre-trained CNNs (i.e. already trained on large datasets like ImageNet [10]) that are then used to train new models specifically applicable to the landscape image dataset. Features extracted from these domain-specific trained CNN are then used with standard semi-supervised novelty detection algorithms like Gaussian Mixture Model, Isolation Forest, One-class Support Vector Machines (SVM) and Bayesian Gaussian Mixture Models to identify the unknown landscape images.

We provide two fine-tuning approaches: supervised and unsupervised. Supervised fine-tuning approach simply uses the the class categories (landscape classes, e.g. airport, stadium, etc.) of the known images dataset. The unsupervised fine tuning approach on the other hand learns the class categories from the known images using the unsupervised clustering-based algorithm. We conducted extensive experiments that prove the effectiveness of our approaches. Our best values of AUROC and average precision scores for the identification problem are 0.96 and 0.94, respectively. In particular, we statistically prove that both fine-tuning methods significantly increase the performance of the identification with respect to the non fine-tuned CNN, and unsupervised and supervised fine tuning approaches are comparable.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

CNN – Convolutional Neural Networks

DNN – Deep Neural Networks

ReLU – Rectified Linearu Unit

SGD – Stochastic Gradient Descent

BIC – Bayesian Information Criterion

i.i.d – independent and identically distributed

MSE – Mean Squared Error

GPU – Graphical Processing Units

PreT – Pre-trained CNN models

SFT – Supervised Fine Tuning Approach

UFT – Unsupervised Fine Tuning Approach

GMM – Gaussian Mixture Models

BGMM – Bayesian Gaussian Mixture Models

OCS – One-class Support Vector Machines

ISOF – Isolation Forests

AUROC – Area under Receiver Operating Characteristics

AVG PREC – Average Precision

LIST OF SYMBOLS

ϵ	Learning Rate
∇	Gradient
δ	Small change (small difference)
α	Level of Significance
λ	Regularization parameter

CHAPTER 1

INTRODUCTION

1.1 Overview

Deep Neural Networks (DNNs), esp. Convolutional Neural Networks (CNNs) have recently gained great success in lots of large-scale visual recognition tasks [25, 52, 56]. This has become possible because of publicly available large-scale image databases such as ImageNet (more than 14 million natural images from 20000+ different categories) [10] and powerful computing infrastructures such as GPUs and multi-node distributed clusters. In the past, ImageNet Large Scale Visual Recognition Competition (ILSVRC) has been challenging researchers around the world to develop effective and efficient visual recognition algorithms which led to the invention of deep CNNs like Alexnet [25], VGGNet [52] and GoogLeNet [56]. These CNNs achieved high and increasingly better accuracy in lots of visual recognition tasks such as image classification [36], object detection[44] and image segmentation [30].

Deep CNNs take weeks to train on large datasets like ImageNet even with powerful GPUs. In real world scenarios, finding sufficiently large amount of application-specific datasets for training CNN models and conducting such experiments with multiple hyper-parameter sets in as short period of time is ideal. However, it is very common to use pre-trained CNN models like Alexnet, VGGNet and GoogLeNet already trained on large datasets such as ImageNet and use these trained weights for training these

CNNs on new and relatively smaller datasets. This particular way of transferring general feature representations from one domain (or dataset) to another is known as transfer learning. In the transfer learning context, these pre-trained models can be used either as weight initialization or as feature extractors for new datasets. Using pre-trained models as weight initialization to train the CNNs in another domain or with other datasets is generally termed as a *fine-tuning approach*.

Deep CNNs tend to learn general features (eg. edges and color blobs in images) on the lower level layers and more abstract and dataset specific features on the higher level layers of CNNs [63]. So CNNs can be efficiently trained on new datasets by using pre-trained models as weight initialization, freezing the learning rates of lower level layers and updating weights only for higher level layers (generally the last one or two fully connected layers). This allows the newly trained CNNs to adopt feature representations already learned from a source dataset and apply that knowledge to new datasets. This approach keeps researchers from having to train these complex CNNs from scratch thus saving a lot of computation time. CNN transfer learning approaches applied for many classification tasks have obtained accuracy scores greater than 90% for smaller training dataset and training time [36, 63].

In this thesis, we use CNN transfer learning to address the problem of detecting novel landscape images from the known image dataset representing several known types of landscape images. Detecting unknown images that are different and non-conforming to the distribution of the known set of images has many security-based applications. For instance, detecting undeclared locations related to nuclear power plants and military installations, and suspicious border activities by analyzing remote sensing images can strongly assist security authorities to enforce robust national security solutions. Similarly, identifying hazardous or illegal crop fields and suspicious

objects in passengers' baggage at airports are other applications that are the equally crucial security concerns that pertains to building secure and smart cities and nations.

We formulate our problem as a novelty detection task to identify unknown landscape images. We use a CNN transfer learning approach to train CNNs on two publicly available benchmark landscape datasets. We use these trained models to extract features for all known and unknown images. Features extracted from known images are used to train standard novelty detection algorithms which are finally evaluated with test sets containing both known and unknown images using the AUROC and Average Precision metric scores.

1.2 Contributions and Outline

The main contributions of this thesis are as follows:

1. We propose the new problem of identifying unknown types of landscapes when large training sets are not available in the context of transfer learning.

2. We present an effective and efficient solution to the problem of identifying unknown type of landscape by taking in input images representing several types of known landscape. Our best AUROC and Average Precision are 0.96 and 0.94, respectively.

3. We propose a baseline procedure for transfer learning without the use of any fine-tuning that does not need powerful computational resources but still achieves good performance.

4. We present a supervised fine-tuning procedure which takes the categories of the known types of landscape images as input and provides outstanding performance in comparison to the baseline procedure.

5. We also present a new unsupervised fine-tuning procedure that works without any knowledge about categories of the known image examples and produces comparable results with respect to the supervised fine-tuning procedure.

6. We compare and present statistical analysis of the difference in performance of standard novelty detection algorithms trained with features extracted from CNNs pre-trained(non fine tune tuned) and trained in supervised and unsupervised fine-tuning setups.

7. We collected two new categories of satellite image dataset: 'Military settlements' and 'Nuclear power plants'. We use these two image classes as the novel (unknown) examples in our experiments.

In **Chapter 2**, we provide the relevant background details of machine learning in supervised and unsupervised context, neural networks and discuss commonly used standard novelty detection algorithms.

In **Chapter 3**, we present the literature review of previous research works related to our thesis. We discuss in detail about the related works in three different topics: novelty detection, deep convolutional neural networks and transfer learning approaches.

In **Chapter 4**, we discuss about the methodology and implementation of convolutional neural networks, transfer learning and novelty detection algorithms in our problem domain, i.e. identifying unknown landscape image content in both supervised and unsupervised learning contexts.

In **Chapter 5**, we extend our discussion on the experimental setup, variants of experiment instances and finish up the chapter with discussion and statistical relevance about the obtained results.

Finally, we discuss the future research direction and conclude our research work presented in this thesis in **Chapter 6**.

CHAPTER 2

BACKGROUND

This chapter provides the relevant background on machine learning, neural networks and novelty detection algorithms. For more details on machine learning and neural networks, we recommend readers to a book by Goodfellow et al. [16]. Similarly for novelty detection algorithms, a survey paper by Pimentel et al. [42] is recommended.

Machine Learning. Machine learning is a form of applied statistics with emphasis on the use of computers to learn complex mathematical functions. Machine learning algorithms can learn from data (or observed facts). More formally, a machine learning process is defined as follows: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E [34]." While solving machine learning problems, it's very important to formulate what type of task (eg. classify images into multiple categories, machine translation and so on) we want our machine to learn while experiencing the available training data with improved performance (eg. accurately predict an image of "cat" as "cat"). Machine learning algorithms can be broadly categorized into supervised and unsupervised learning algorithms depending upon the nature of dataset they are learning from.

2.1 Supervised Learning

Supervised machine learning is a task of searching a function from a hypothesis space given a **training data** with the corresponding labels. Each training example consists of independent variable(s) defining the input domain of data and dependent variable(s) defining the target. Many supervised learning can be formulated as enabling a computer to learn a function $f : X \rightarrow y$, where X represents the space of independent variables (input space), and y the space of dependent variables (output space). To tackle any machine learning algorithms, we can almost always start by making some general design choices beforehand. Note that machine learning algorithms can be as difficult as learning very complex mathematical functions, and most of the time the most scientific way to tackle such problems is through a process: experimentation, evaluation and decision making. Here are the general design choices for building machine learning algorithms:

1. Choose the form of the model i.e. hypothesis space for candidate models
2. Choose loss function.
3. Select optimization procedure.

We introduced a couple of important machine learning terminologies above. We discuss about these design choices in the context of a simple machine learning algorithm, Linear Regression.

Linear Regression. Linear regression is an algorithm that models the relationship between one or more independent variables X and a scalar and real-valued dependent variable y. In other words, linear regression learns a function mapping that takes each observation in X as input and tries to approximate as accurate y

as possible. We call this process "making a prediction" over a range of continuous values. Along the same line, classification tasks have a goal to learn a function that can achieve a prediction over a discrete class of objects. For the sake of simplicity, lets consider a case where we want to model the function mapping from two independent variables (bi-variate) to one dependent variable. Consider m observations of training data with $X \in \Re^2$ and $y \in \Re$. We start by choosing the form of our model. The simplest model in this case can be a set of linear functions (F) from X to y. A general linear model to be used as a candidate function can be formulated as:

$$\hat{y} = w^T X + b \tag{2.1}$$

where, $\hat{y} = f(X) \in F$ is a predicted value for the corresponding input example in X, and $w \in \Re^2$ $(= [w_1, w_2])$ and $b \in \Re$ are the parameters that define hypothesis space. We will use the terms "parameter" and "weight" interchangeably in the rest of the chapter.

Clearly, we can see that there are sufficiently many values of parameters we can choose from and each set of such values defines one candidate in the hypothesis space. A simple "guess-and-check" approach would be practically unfeasible since the dimension of input space can be as large as in the order of thousands (or millions) and the search space of parameters would increase exponentially. Then how do we find the best set of parameters that would give us the best fit over our m training data? In other words, how can we obtain the optimized values of parameters? To answer this question, we opt another design choice, i.e. choose a loss function. Loss function enables the machine learning algorithm to capture how far is a candidate model from the true distribution of training data. A commonly used loss function

in linear regression setting is mean squared error (MSE) between the target and the predicted values by the candidate model.

$$MSE_{train} = L(w)_{train} = \arg\min_{w,b}[1/m \sum_{i=1}^{m}(w^T x_i + b - y_i)^2] \tag{2.2}$$

$L(w)_{train}$ is also called an **objective function** that defines a goal of a learning algorithm to achieve the lowest possible mean squared error while achieving the best performance. Parameters w and b control the behavior of the algorithm. The weights depict the importance of each independent variable while computing the output. The higher weight value for a corresponding variable means the higher influence of this variable in determining the output. The lower weight value would result in lesser effect on the prediction.

In the real world supervised learning setting, we measure the performance of machine learning algorithms by evaluating the learned function (or model) with previously unseen observations. These observations are commonly called **test data** and the process of computing output for test data is termed as *inference or prediction*. The ability of machine learning algorithms to perform well on tasks with the test data is known as algorithm's generalization power. How can a machine learning algorithm do well with test data when we set to observe only the training data? This is a central idea of how machine learning algorithms work and can be reasoned well with the help of *statistical learning*. Machine learning algorithms are developed on top of very important statistical assumption that basically says that the observations in training and test data are observed during data generating process as independent and identically distributed (**i.i.d.**) samples.

Loss function as shown in equation 2.2 is a formulation of an optimization problem

that aims to minimize the cost associated with each set of w and b values given the training data. The sub-expression $(w^T x_i + b - y_i)^2$ is the measurement of cost of this model obtained from squared difference between the true output y_i and the predicted value $w^T x_i + b$. Before we discuss how to come up with the optimized parameters that minimize the cost of the function, let's discuss common challenges pertaining to the model selection. Until now, we came to an understanding that we want to reduce the training error (prediction inaccuracy in context of regression and classification task) during the learning process. Perhaps, equally important is the ability of a learning algorithm to obtain minimum error while inferring the output for previously unseen test data. This error is commonly termed as **generalization or test error**.

Model complexity, Overfitting and Underfitting. In terms of mathematical definition, equation 2.1 is a simple model because we can describe it by using only two simple first-order coefficients ($[w_1, w_2]$) and a scalar b. Consider a context where we are trying to find the fitting function using linear regression whose true distribution is a polynomial function with degree 4. In this case the optimal complexity of the model that we would want to use to learn this distribution would ideally be a polynomial function with degree 4.

Figure 2.1 shows the behavior of three different (simple to optimal to complex) models selected to learn the training data that have a true distribution defined by a polynomial function with degree 4. In all the three sub-figures, green line is the true distribution of the training data. The left plot is the case where the true function is being fitted with a *simple linear model*. In this case, the model fails even to fit the seen training data. Here we call that the model is **underfitting** provided that it is not being able to obtain the sufficiently low training error (can be seen from the larger distance of data points from the fitted function (blue line)). In this case the selected

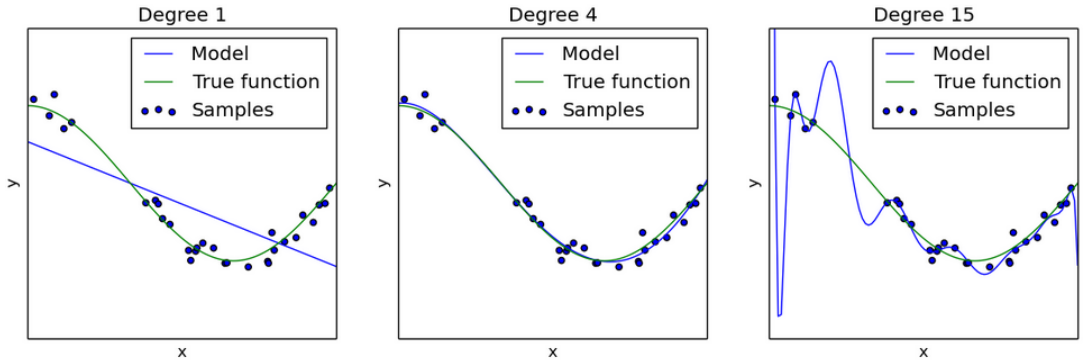


Figure 2.1: Example of underfitting and overfitting when trying to fit the true distribution defined by a polynomial function with degree 4 with linear regression. (left) Underfitting case: Linear regression trained with a linear model with degree 1. (center) Optimal case: Linear regression trained with a linear model with degree 4 (right) Overfitting case: Linear regression trained with a linear model with degree 15. Source: [39]

model is too simple to represent the underlying distribution of the training data. In the right plot, a *complex polynomial function* with an order 15 is being used to fit the true distribution of the training data. It can be clearly seen that the selected model has fitted the training examples with nearly zero training error. However, this model tend to perform very poor with unseen observations. For example, if we want to predict the output of the data near the left-most peak of the same image, the generalization error is huge and is larger than even the simplest linear model. This is the case of **overfitting** where the selected model is not able to maintain the small prediction error while being evaluated with unseen examples as in training phase. The plot in the center is the case of ideal model that every machine learning algorithm would like to learn for a given distribution of data (i.e. polynomial function with degree 4). The close alignment of the true function (green curve) and the fitted model (blue curve) depicts that this model almost perfectly fits the training data, and also performs well to the unseen data. The selected model is said to have *optimal*

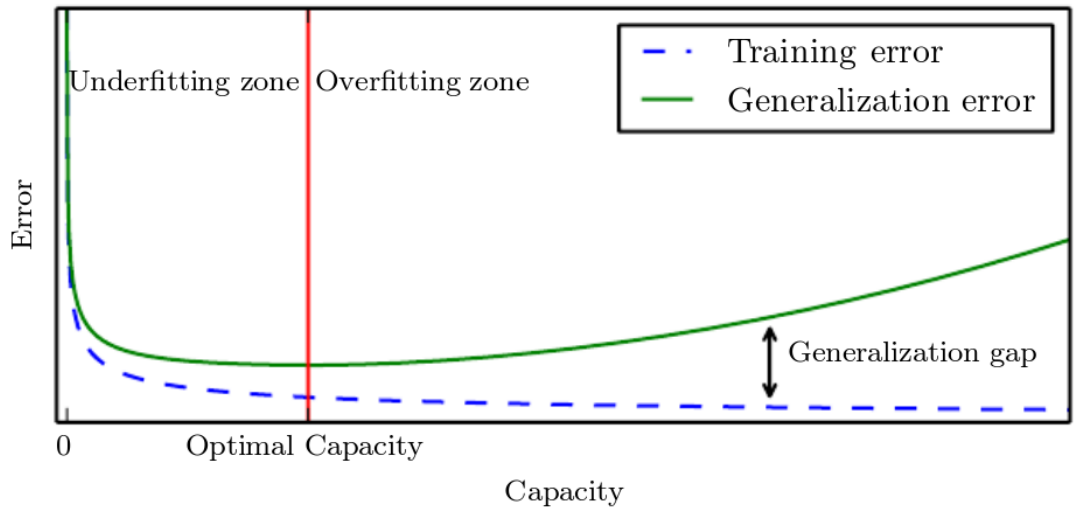


Figure 2.2: Overfitting and underfitting in terms of relationship between training and generalization error. Source: [16]

capacity or complexity to represent the underlying training data. Note: to test this mentally, imagine a point in each of the plots in the figure and compare the difference between the shortest distance of that point from the green and blue functions. Try it for all the three plots. The larger the distance, the greater the error.

In conclusion, a well performing machine learning algorithm learns from the training data with small training error during the learning phase, and also maintains the small difference between training and generalization error during the inference phase. Figure 2.2 intuitively describes all the scenario discussed above based on the choice of the model.

Regularization. So far, we discussed how we can define a machine learning algorithm as an optimization problem (equation 2.2) and challenges of selecting an appropriate model for a problem at hand. We discussed one way of modifying the learning algorithm in terms of complexity of the model in the hypothesis space. But in

real world scenarios, there are cases where there might be multiple appropriate models that can equally perform well to fit the underlying data, and we should still choose a better among those models. To aid on this, and to help prevent the overfitting of the machine learning algorithms, a commonly used technique called regularization can be employed. Regularization discourages the model to learn parameters with too large values and causing one variable to have more affect on prediction than another. We can define a loss function for a linear regression problem by extending the equation 2.2 by introducing a **weight decay** factor as:

$$L(w) = \arg\min_{w,b}[1/m \sum_{i=1}^{m}(w^T x_i + b - y_i)^2] + \lambda(w_1^2 + w_2^2) \tag{2.3}$$

where λ is a hyperparameter that penalizes the parameters and encourages the smaller values. $\lambda = 0$ means no effect of regularization and larger λ enforces smaller parameter values. In simple terms, by imposing the regularization term $f_w = w_1^2 + w_2^2$ (commonly used L2 norm regularizer), we are able to modify the learning algorithms not to select a model too far from the true distribution and perform well with both training and test data.

Model Selection Criterion. *Cross-validation* is one of the successful approaches to evaluate the performance of trained models in machine learning. Training a model with a fixed training data and evaluating the trained model with a fixed and usually very small test data can pose problems during generalization. This is because if the fixed small test data are used to validate the model and if the test data turns out not to represent the underlying true distribution of the data (regardless of the i.i.d. assumption, which is a possible case), then the trained model might not well generalize to other unseen data other than the ones used to validate the model. So

to avoid such a scenario, k-fold cross validation technique is used. Training data is divided into k equal and disjoint samples, and usually k-1 folds are used to train the model and the remaining 1 fold is used to validate. This process is repeated for k times selecting the unique validation set for each case. The final performance measure is then averaged over the k-cases hence bolstering the generalization ability of the trained models. In our work, we use 10-fold cross-validation to validate the generalizability of training of convolutional neural networks (as discussed later in this and subsequent chapters.)

Bayesian Information Criterion (BIC). When fitting models in machine learning, we discussed how we can reduce the training error (or statistically speaking increase the likelihood) by increasing or decreasing the parameters. But doing so without proper attention can also lead to a overfitting problem. Mathematically, BIC can be formulated as:

$$BIC = ln(m)k - 2ln(\hat{L}) \tag{2.4}$$

where,

$\hat{L}$ = the maximized value of the likelihood function of the selected model,

x = training data,

m = sample size,

k = # of free parameters associated to selected model. eg. w and b in linear regression

The lower the value of BIC, the better the goodness of fit of the selected model. We have used BIC to select the appropriate gaussian mixture model (to be discussed in detail later in this chapter and subsequent chapters) while applying in novelty

detection task. Similar to *BIC*, *Akaike Information Criterion (AIC)* also tries to balance the trade-off between the representational power (model's complexity) of the selected model and its ability to produce lower training error with the selected parameters. *BIC* has preference for simpler models compared to *AIC*.

2.1.1 Optimization

In the previous sections, we looked into a case where a machine learning algorithm can effectively be represented as an optimization problem. In machine learning, one of the most commonly used optimization algorithms is gradient descent. If we pose the restriction on selection of objective function to be only differentiable functions, then we can compute the gradient of the function (using numerical method or analytical method using calculus). In the neural network context, gradients are computed by using back-propagation algorithms [45] (to be discussed in detail later).

Common gradient descent operation and parameter update:

$$\nabla_w L(w) = \nabla_w [1/m \sum_{i=1}^{m} (w^T x_i + b - y_i)^2] + \lambda(w_1^2 + w_2^2) \tag{2.5}$$

$$\triangle w := -\epsilon \nabla_w L(w) \tag{2.6}$$

$$w := w + \triangle w \tag{2.7}$$

Gradient descent algorithm takes an objective function ($L(w)$) such as equation 2.3, usually a function of a model's parameters (w, b) in the context of machine learning, and iteratively updates model parameters in the direction opposite to the

gradient of $L(w)$, $\nabla L(w)$ obtained by computing the partial derivatives w.r.t. each parameter (equations 2.5 to 2.7). The gradient descend constructs the first order approximation of Taylor expansion of L. Such gradients are used as search direction. There are different variants of gradient descend algorithm. *Batch gradient descent,* also known as vanilla gradient descent, computes the gradient of the loss function w.r.t. the model parameters for the entire dataset and updates the parameters only when it is done considering the entire dataset. In practical scenarios, the training data can be very large (e.g. ImageNet has millions of training example images) and the algorithm can easily run out of resources if the entire dataset is used to compute the gradients. *Stochastic gradient descent (SGD)* in contrast uses a single example to compute the gradient and update the parameters. SGD solves the problem of resource limitation and redundant gradient computations for large datasets, but it is highly influenced by the extreme examples in the training data allowing it to perform frequent updates with high variance and lets the convergence overshoot the local minimum. However, this phenomenon can be controlled by a hyper-parameter known as learning-rate (ϵ in equation 2.6) that tells the SGD or any gradient based algorithms whether to take large or small steps while updating the parameters. In the context of neural networks, it is very common to update parameters using the small batches of training data. Also known as *mini-batch stochastic gradient descent* , this algorithm allows us to perform many approximate updates instead of fewer exact updates making it practically feasible in many problems involving very large training data which is also the case of neural network problems.

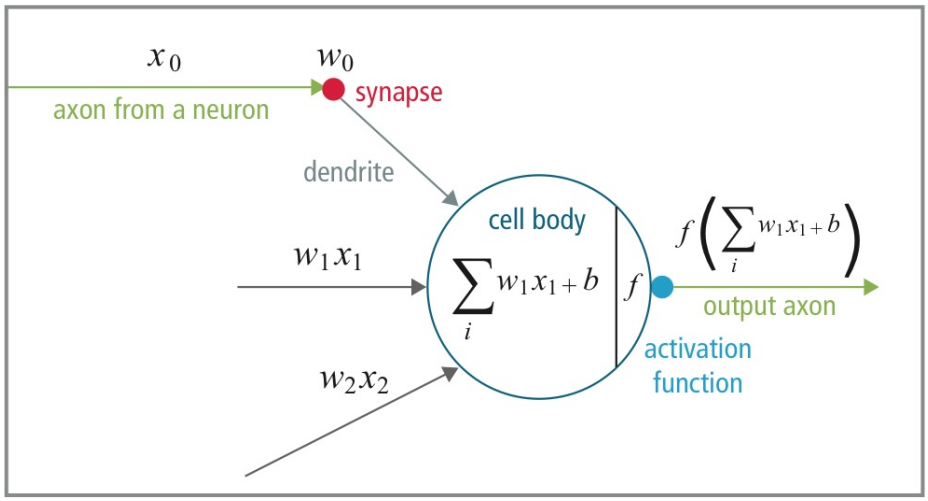


Figure 2.3: A diagram depicting a process of a single computational unit in artificial neural network analogous to the biological neuron. x_0, x_1, x_2 are the values of a single input example, w_0, w_1, w_2 are the parameters associated to the input layer. $\sum_i w_i, x_i$ is a linear transformation as seen in linear models which is passed through a activation function to apply non-linear (relu, sigmoid, tanh) transformation, and is finally propagated to the next layer. Source: [23]

2.1.2 Neural Networks

Linear models like linear regression and logistic regression can be trained efficiently and reliably but have a downside that the model representational capacity is limited to linear functions and the models cannot understand the interaction between the input variables. This can be easily illustrated by trying to classify the output of truth table of XOR function into binary (0,1) classes. We quickly realize that we cannot obtain a decision boundary that could linearly separate the input space into 0 and 1. Neural networks help to solve this problem. The primary difference between the linear models and neural networks is the non-linearity of a neural network. Even though, the non-linearity associated with neural networks makes a problem of solving convex optimization of loss function to non-convex, same gradient-based optimization algorithm like mini-batch SGD can be used to iteratively obtain the optimized model parameters

in neural networks as well. Neural networks have cascaded layered architecture. Each layer consists of computational units, generally known as neurons (from the fact that neural networks were partially inspired from biological neurons), that takes in input from the preceding layers and outputs a single valued result similar to a linear function. The computed output is then passed through a non-linear transformation (some common non-linearities used are sigmoid range(0,1), tanh range(-1,1), rectified linear unit (relu: $x \rightarrow max(0, x)$) which is then used as input to the next layer. The process of propagating the intermediate outputs in each layer of the neural network is known as *feed forward*. Figure 2.3 shows the operation of one computational unit showing analogy to the operation of biological neuron. In neural networks, in addition to the design choices required to consider like in linear models, it is imperative to decide the architecture of the network we want to train for a problem at hand. The layer where the input is fed is known as input layer, the layer from where the output is fetched is known as output layer and all the other intermediate layers that perform series of linear and non-linear transformations are known as hidden layers. For example, a simple 3-layer fully-connected network is shown in Figure 2.4 which can be easily generalized to wider (more computational units in each layer) and deeper (more layers) networks. The computational units in same layers do not interact with each other and only interact with the layers immediately preceding and succeeding, making the overall computation easier to implement as matrix and vector computations. Almost all the neural network implementations are based off of efficient matrix and vector computations. And, the term **deep neural network** comes from using a large number of layers while designing the neural network architecture.

We can clearly see that with neural networks, even with one intermediate layer (known as hidden layer), the input and output spaces do not interact directly with

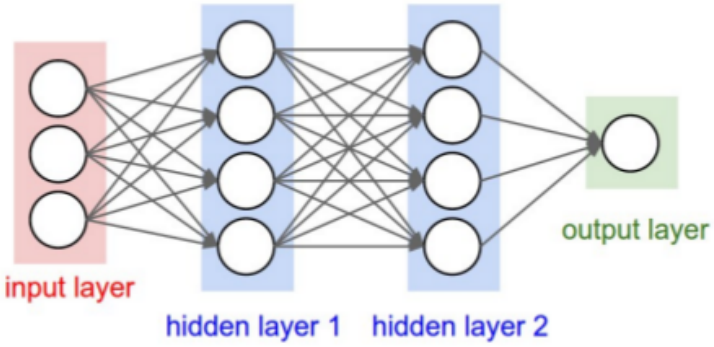


Figure 2.4: A simple 3-layer fully-connected neural network with two intermediate layers (hidden layers). This network for example can be used to learn the non-linear relationship between the input in three dimensional space that maps to a single-valued output. Source: [23]

each other. Then, the question is how do we compute the loss function and optimize the model parameters? Turns out, the similar representation of loss functions and same gradient-based algorithms can also be used to obtain the optimized model parameters, however with the help of back propagation algorithm which efficiently computes the gradients of neural networks. Backpropagation algorithm was formulated to encourage self-organizing neural networks.

2.1.3 Back Propagation Algorithm

Optimization algorithms like SGD require the computation of gradients which can pose computational overhead when a very large dataset is involved in machine learning algorithms. SGD algorithms use the information from the computed gradient in order to update the parameters towards the optimized value. In neural network applications, complex architectures might contain from few to hundreds of layers with thousands of computational units in each layer resulting in millions to billions

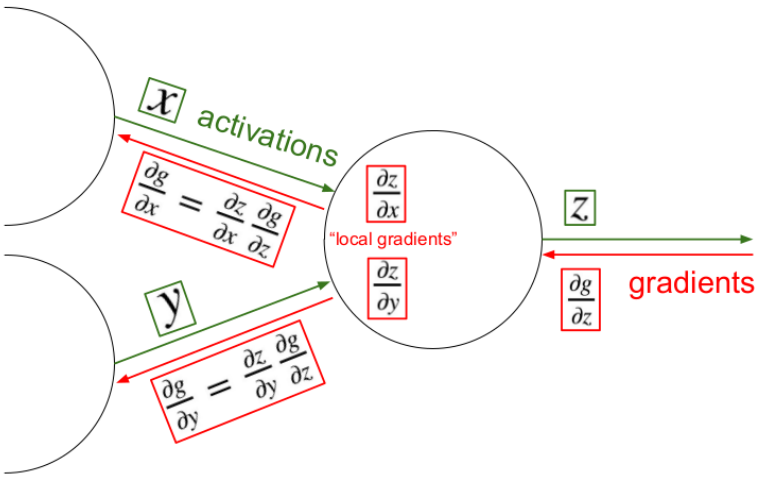


Figure 2.5: Backpropagation of gradients through each layer shown for the case of single computational unit. Note how the local gradients computed at computational unit is back propagating to previous layers by using a chain-rule of calculus. Source: [22]

of parameters to update in a single pass. Backpropagation algorithm [45] efficiently computes gradients of the scalar-valued loss function with respective to the model parameters. Not only in neural networks, backpropagation algorithm can be used with any input-output objective function with bounded derivatives. Once the forward pass is completed, with a series of linear and non-linear transformations through multiple hidden layers, the output in the output layer is compared to the expected output (eg. true classes in classification task) and the loss function is formulated. The gradients of the loss function given by the difference between the obtained and expected output are computed in the backward pass. First, the local gradients are computed for each computational unit just by considering the inputs to that single unit. Then, using the chain rule of calculus, the local gradients are propagated in the backward direction upto the input layer. Now that we have computed the gradients for each layer in the network, again gradient-based optimization like mini-batch SGD can be

used to update the network parameters associated to each layer given the direction of the update by the gradients computed using the backpropagation. Figure 2.5 shows the propagation of local gradients through a single computational unit in backward direction of the neural network.

Summary. In general understanding, many of the concepts discussed with linear models like simple linear regression problems are generalizable to the complex machine learning algorithms like SVMs and deep neural networks as well. The design choices will vary depending upon the nature and size of training data that the considered algorithms will experience and the machine learning task to be accomplished. With neural networks, there are further decisions to be made on what type of network architecture to chose. The in-depth discussion on these topics is outside the scope of this thesis.

2.1.4 Convolutional Neural Networks (CNNs)

Deep learning methods are representation learning approaches that can pick up the large dataset (images, audio, video etc.) in their raw form and can build hierarchical feature representation of dataset using the multi-layered deep network architectures [26]. Deep Neural Networks have recently been widely used in the field of computer vision in a range of visual recognition tasks, trained through layers of deep convolutional networks and back-propagation algorithm [27, 26]. In the last 5-6 years, due to the upsurge of high-performance computing devices such as GPUs and large-scale distributed systems and availability of large scale public image repositories such as ImageNet, CNNs have gained a great success range of visual recognition tasks such as image classification [25, 36], object localization and detection [14, 44, 29], image segmentation [30, 38, 64] and anomaly detection [46].

In context of visual recognition, convolutional operations are inspired from the way human's visual cortex perceive and interpret visual images before sending the visual signal to the brain, where the signal is mapped into understandable abstract object (eg. face of a friend, etc.). Rather than scanning the entire image at once, our visual cortex works by generating activations for more general features in the images. For example, the abstract image of the human face is recognized as a human face by our brain through a multi-layered feature mapping mechanism, where our brain tries to learn edges and curves, then smaller distinct objects and then more abstract features before interpreting that images as a human face. Similar to this principle, convolutional neural networks also try to learn end-to-end features from the raw image gradually learning from granular properties (edges, corners, objects, etc.) of image upto the abstract concept of the image (human face). Figure 2.6 [28] shows a simple fully convolutional neural network that starts with an input layer (the raw pixels are flattened to be represented as vectors) which is passed to a first convolutional layer. Convolutional layers contain **filters or kernels**. Kernels can be interpreted as feature analyzers. In this case, there are 6 kernels that will produce 6 feature maps after convoluting the input image. Convoluting an image involves following operations: a) scanning an image from left-top to the bottom-right, b) performing an element-wise dot-product between the kernel and that specific part of image currently being convoluted (also known as receptive fields), c) continuing convolution with a stride size that basically defines how many pixels should the kernels move to the right next. Kernels are generally smaller in size and are represented as a squared matrix, and are shared among all the input examples. Next, the sub-sampling or pooling layer is applied to the activation map resulted from each kernel to reduce the size of the operational matrix yet preserving the most information. The commonly used pooling

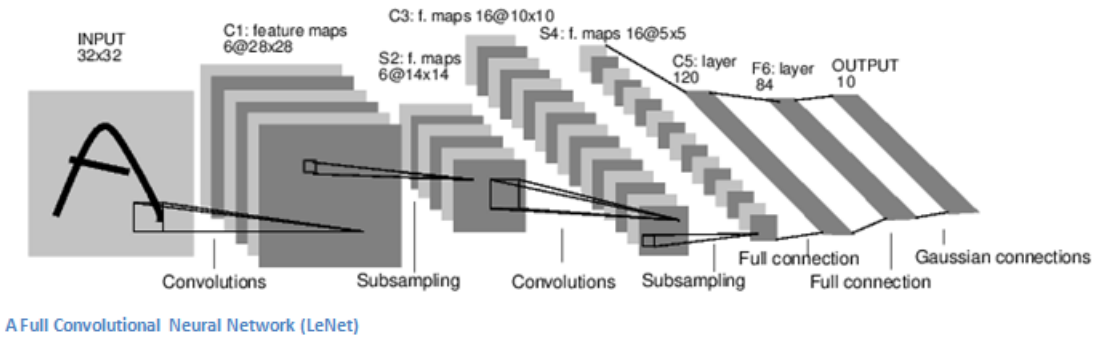


Figure 2.6: LeNet convolutional neural network architecture Source: [28]

functions are $max - pooling, average - pooling$, etc. The output of the preceding layers are fed as input to the next layer, and the output is traversed into the output layer of the network. For example in a classification problem, results of the output layer is commonly termed as logits and the operational network layers such as softmax is applied to convert logits into a probability distribution representing each number to be the probability of that input belonging to a particular category.

CNNs started gaining great attention after Krizhevsky et al. [25] won the annual ILSVRC challenge in 2012 with their breakthrough ConvNet model popularly known as AlexNet. Since then, many researchers have come up with significant improvement in the original AlexNet model (by fine-tuning the network's hyper-parameters) or have invented many other deeper and wider networks that are best suited for various image processing and recognition tasks. In the following subsections, we discuss in detail about some standard ConvNet architecture that we will be using in our implementation for training and feature extraction in both supervised and unsupervised setups in the later chapters.

AlexNet. Krizhevsky et al. [25] won the annual Imagenet Large-Scale Visual Recognition Challenge (ILSVRC) in 2012 with their breakthrough ConvNet model

popularly known as AlexNet. This architecture constitutes an eight layer network(five convolutional and three fully-connected). Each convolutional layer is scanned by various number of kernels (each specifically responsible to find specific features in the image), and the network tends to learn from general to more specific features as it goes into the deeper layers. Inspired from the working procedure of the human visual cortex, the size of the kernels signify the size of the receptive field in the input image to be considered for learning features specific to that particular area. Similar to the standard neural networks, units in the fully-connected layers are connected to all the units in the preceding layer. To add non-linearity to the network, this architecture uses Rectified Linear Units (ReLU)[35] as a activation function after each convolutional and fully-connected layer. This network adopts gradient descent to optimize weights in the network. ReLU adds non-saturating non-linearity in the network which is much faster than the other activation functions like *tanh* and sigmoid while applying gradient descent. The local response normalization used in several layers implements a form of lateral inhibition that creates competition amongst neuron outputs for important activities. This basically aids generalization on top of the non-saturating ReLUs. The pooling layers provide ways to select outputs of the neighboring neuron groups mapped by the same kernels and use a single value for a group. Alexnet uses overlapping *MAX-pooling* approach to select much represented value from neighborhood pixels.

To avoid overfitting, AlexNet used various data augmentation techniques to feed the network with varieties of transformed input images by enlarging the dataset using label-preserving transformations. The dropout layers reduce complex co-adaptations of neurons, thus strengthening the representation of such neurons in the network by randomly dropping out some other neurons which are deprived to contribute

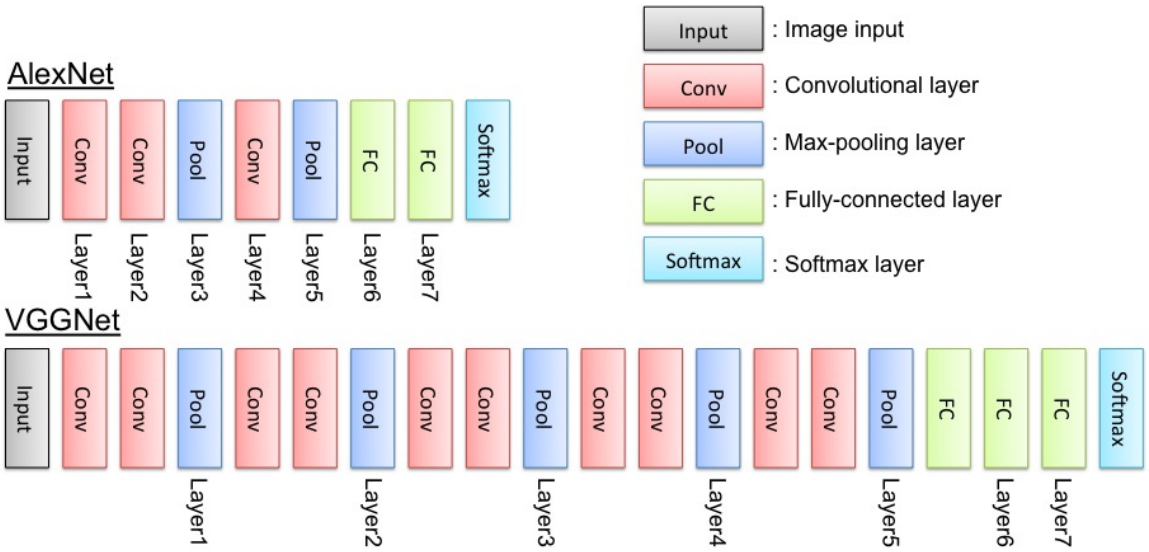


Figure 2.7: CNN architectures: Convolutional neural network architecture of Alexnet model with 7 learnable layers (top) and convolutional neural network architecture of VGGNet with 19 layers (including all convolutional and pooling layers) (bottom). Source: [24]

in the forward pass and back-propagation. Meaning that, for each sample, the network randomly selects a weight shared sub-network to boost the contribution of selected subsets of neurons thus preventing overfitting and enhancing the network's generalization power. Figure 2.7 (top) shows the architecture of 7-layered Alexnet.

VGGNet. VGGNet [52] developed by K. Simonyan et al. was the winning ConvNet architecture in image localization task in ILSVRC competition, 2014. This network highly contributed in proving that the depth of the weighted layers in networks plays a significant role in improving image localization and classification task. Input images are scanned through convolutional layers using filters with smaller (3x3) receptive fields. Most of the convolutional layers are followed by pool layer which performs 2x2 *MAX-pooling* with stride 2 and 0 padding. The stack of convolutional layers is followed by three fully-connected layers. This network also uses ReLU for non-linearity and does not use local response normalization. Figure 2.7 (bottom)

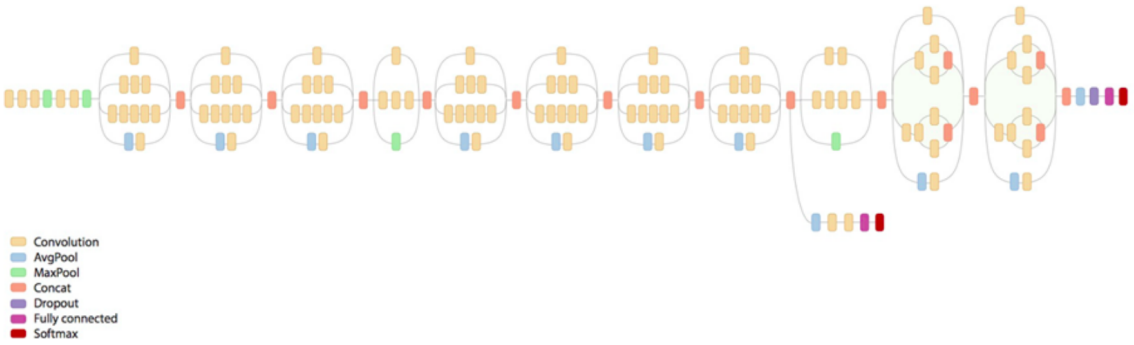


Figure 2.8: CNN architectures: Deep fully convolutional network architecture of Googlenet. Source: [56]

shows the architecture of VGGNet with 19 learnable layers.

GoogLeNet. Commonly known as inception model, Szegedy et. al [56] designed a deep convolutional neural network architecture that achieved state-of-the-art performance on image classification and detection task in the ImageNet challenge 2014. The key contribution of this architecture is the size of the learnable parameters that has 12 times fewer parameters than Alexnet, allowing the model to train in shorter time with less memory consumption. Another advantage of this model is that transfer learning using the pre-trained GoogLeNet model is much faster than Alexnet and VGGNet allowing the researchers to experiment and evaluate models in shorter amount of time. Figure 2.8 shows the architecture of fully convolutional GoogLeNet CNN.

2.2 Unsupervised Learning

Unsupervised learning algorithms are types of machine learning algorithms that learns the important properties of the underlying training data being experienced without the supervisory labeled responses. Commonly applied unsupervised machine learning algorithms are clustering, anomaly detection and novelty detection algorithms.

2.2.1 Novelty Detection

Novelty detection is a machine learning task of identifying new or unknown data during the inference or generalization phase (test data) that were not present in the training data during the learning phase. In simple terms, the machine learning algorithm is constructed to learn the characteristics of only the "normal" or "known" training data. From now on, we use *"normal" and "known"* interchangeably. Novelty detection algorithm expects mixtures of known and novel examples in test data during the generalization phase. By novel or unknown examples, we mean that these novel examples in test data come from different probability distribution or lies far in the feature space than the training data. We will also use the terms *"novel" and "unknown"* interchangeably as well. Novelty detection has important applications in domains involving large datasets generated from critical systems. Other applications include cyber-intrusion detection [43, 13, 62], terrorist activity, system breakdown, fraud detection [41, 59, 4, 12], data leakage prevention [51] and many other specialized applications [15].

Just to build an intuition around novelty detection algorithms, very similar classes of algorithms known as *outlier detection* algorithms also have near-to-similar goals of identifying outlier or abnormal data points in the test data. The only difference is that, the unknown or abnormal instances of data points are also considered during the training phase. Outliers in dataset possess highly deviated statistical characteristics which are not in agreement with the majority of observations in the training data.

Novelty detection has been widely applied to detect novel objects in image data and video streams. Building security applications like surveillance systems and alert systems is a very common task that falls within the application of novelty detection

algorithms. Aiding the security applications via implementation of novelty detection algorithms in image datasets is the primary goal of our thesis. Novelty detection algorithms are commonly categorized into different categories: a) probabilistic (eg. Gaussian Mixtures), b) distance-based (eg. K-means clustering), c) domain-based (eg. one-class support vector machines), d) reconstruction-based (eg. neural networks, autoencoder) and so on. We apply Gaussian mixture models, Bayesian gaussian mixture models, one-class SVM and isolation forest (based on ensemble of trees) algorithms to apply novelty detection algorithm in our methodology. We use these algorithms to identify novel landscape images in the test data that are not present during the training phase. We will discuss in more details about the application of these algorithms specific to our application in Chapter 4. Next, we discuss in detail about how each of these algorithms can be trained to learn the normal characterstics in training phase and how to evaluate each of these models given the previously unseen test data containing both known and novel instances of data.

Gaussian Mixture Models. Gaussian mixture model is a non-bayesian, parametric probability density based model and assumes that all the data points are generated from a mixture of Gaussian distributions with unknown parameters. Gaussian Mixture model implements EM-algorithm for fitting the training data. Each sample is fitted to a probability specific to each cluster and a sample is assigned to a cluster for which it has higher probability.

One-class SVM. One-class SVM is an unsupervised algorithm that learns a decision function for outlier detection which classifies new data as similar or different from the samples used in the training set. Given a set of training samples, it will develop a soft boundary defining the known samples and classifies new observations as belonging to the known samples or not.

Isolation Forest. Isolation forest is a model-based method that isolates unknown data points (in context of novelty detection) or anomalies (in context of anomaly detection) instead of learning profiles of the normal training data. Isolation Forest is an ensemble anomaly detection algorithm that computes anomaly score for each sample. This algorithm works by isolating observations by first randomly selecting a feature and then randomly selecting a split from a range of minimum and maximum value of the selected feature. The average depth of a tree required to isolate a sample over a forest gives the measure of normality or abnormality of that score in a distribution. Fundamentally, according to isolation forest, isolating unknown data points is easier as only few splits (fewer conditions to be checked and hence lower average tree depth) are required to distinguish the unknown data points from the known ones. In our application to identify unknown landscape images, we compute the average anomaly score of examples in test data computed from multiple random trees (base classifiers). The lower is the score for an image, the more is the data point unknown.

Clustering Ensemble. Alexander et al. introduces the cluster ensembles framework to tackle the problem of combining multiple clustering of a set of objects (landscape images in our application domain) into a single consolidated clustering labels without accessing the features or algorithms that determined the cluster partitions [55]. In other words, this cluster ensemble algorithm allows us to compute the consensus cluster labels from the collection of label vectors obtained by training multiple instances of well known clustering algorithms (this information actually is not required for the cluster ensemble) such as Gaussian mixture models. High-quality consensus functions are effectively and efficiently applied to induce the similarity measures between the cluster labels computed from different processes (multiple

instances of clustering algorithm in our case) finally computing the unique consensus label for each object. In our application, we use the clustering ensemble technique to the clustering based novelty detection algorithms being used, Gaussian Mixture Model, to obtain the consensus cluster labels in order to train convolutional neural networks.

Autoencoder. An autoencoder is an artificial neural network which is a commonly applied for unsupervised novelty detection based on the reconstruction error of the training examples and nonlinear dimensionality reduction [47]. A simple autoencoder can be seen very similar to a feed-forward multilayer perceptron neural network. An autoencoder has two architectural parts: encoder and decoder. The encoder part of an autoencoder aims to learn a encoded representation (embeddings) of training in different feature space data by efficiently reducing the dimensionality of the original data space. The decoder phase on the other hand try to reconstruct the original data by taking the embeddings (compressed feature vectors) as input. The output of an autoencoder has the same number of computational units as the input (original feature dimension). The intuition behind an autoencoder to be used as novelty detection algorithm comes from the ability of autoencoder to effectively reconstruct the examples that have similar statistical properties in the original feature space, thus obtaining the smaller reconstruction errors for known type of objects. An autoencoder tends to obtain higher reconstruction errors for the novel or unknown images. Similar to any novelty detection algorithm, autoencoders are also trained with only the known examples in training data. Once the optimized embeddings are learned using the training data, the reconstruction errors are computed for all the known and novel data in test set. The higher the reconstruction error, the higher is the chance of that data point to be unknown. Figure 2.9 shows an architecture of a

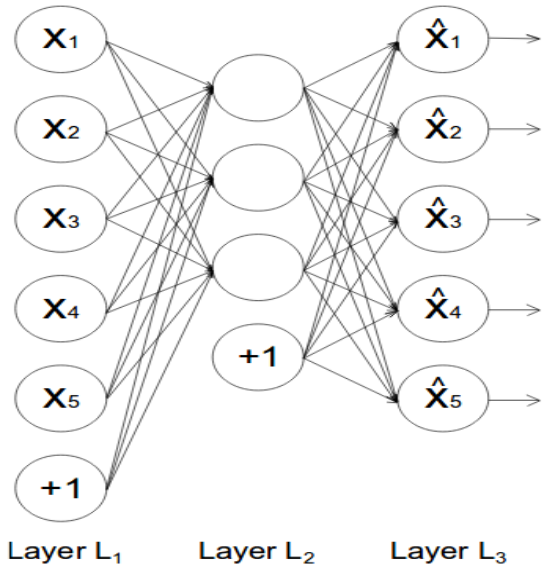


Figure 2.9: Simple architecture of an autoencoder with an input (Layer L_1), one hidden (Layer L_2) and an output layer (Layer L_3). Layer L_2 captures the embeddings of input layer into lower dimensional space. These embeddings are used by output layer to reconstruct original data. Source: [47]

simple 3-layered feed-forward autoencoder.

2.3 Performance Metrics

Similar to any other machine learning algorithms, it is important that we evaluate the performance of our convolutional neural network and novelty detection algorithms specific to our classification and unknown landscape image identification tasks respectively. One common way to validate classification tasks with CNNs is via k-fold cross validation. In our experiments, we perform ten-fold cross validation to evaluate our trained CNN models. To validate novelty detection algorithms, we compute a novelty scores for the test data after training various novelty detection algorithms. We use these scores and the ground-truth (whether the object or landscape image in our case

is known or novel) in order to compute Area under Receiver Operating Characterstics (AUROC) and Average Precision scores.

AUROC. AUROC score measures the discriminating ability of classifiers or novelty detection algorithms to correctly classify objects in different categories: known category and novel category in novelty detection. AUROC score basically tells how well the novelty detection algorithm is able to distinguish known objects as known and novel objects as novel in the test dataset. The AUROC is a plot with false positive rate of a discriminating model as x-axis and true positive rate in y-axis.

Average Precision. While AUROC score gives us the idea about how well the novelty detection algorithm discriminated the known and novel examples, Average Precision score on the other hand tells us the ability of novelty detection algorithm to discriminate novel objects as novel. In real world scenarios, it is common to have a very small proportion of novel examples compared to the normal examples. Therefore, we want to make sure that our novelty detection algorithms are able to discriminate the relatively small population of novel examples. False negative (eg. an unknown or security-concerned landscape image classified as known when it is unknown) rate is very critical for the discriminating models like novelty detection and we would want to lower the false negative rates as much as possible.

2.4 Statistical Significance Test

In statistics, when we hypothesize relationships between two different distributions, we basically want to answer the following two questions: a) What is the probability that some relationship exists between these distributions? and b) if the relationship exists, how strong is it? One of the very common ways to address such questions in

statistical empirical studies is via a statistical significance test. T-test in particular is applicable to test two related samples (eg. two different sample groups of a variable) representing some distribution. T-test is a two-sided test for the null hypothesis that two related samples have identical expected values and is commonly used with small sample sizes, testing the statistical difference between the samples. We perform t-test in following steps:

1. State the research hypothesis.

2. State the null hypothesis.

3. Select the level of significance (α)

4. Calculate test statistics value (t-value) and p-value.

5. Interpretation (make statements about the findings)

For a given two sample distributions, with sample mean $\bar{X}$ and $\bar{Y}$ and variance $\sigma^2(X)$ and $\sigma^2(Y)$, respectively, we calculate the t-value as:

$$t\_value = \frac{\bar{X} - \bar{Y}}{\sqrt{\frac{\sigma^2(X)}{m-1} + \frac{\sigma^2(Y)}{n-1}}}$$

where,

$m = size(X)$

$n = size(Y)$

Later in our experimental section (Chapter 5), we use statistical t-test to compare performance results based on different sample groups.

2.5 Transfer Learning

Traditional machine learning techniques have the major assumption that the training and the testing data must have the same feature space and should come from the similar (if not same) probability distribution family [37]. In many field of study, even today, it is very common to learn different new models (supervised, semi-supervised and unsupervised) even for closely related domains and problems. The common approach of building algorithms with a fixed-variated dataset as input after initializing a new model with zero knowledge and training those models with the popular optimization techniques do not allow the algorithm developers and practitioners to transfer prior knowledge to the related problems in similar problem domains.

In recent years, transfer learning has emerged as one of the effective techniques in machine learning and data mining as a way to transfer learned knowledge from one domain task to another in order to make the generalizability of the learned models more robust. The investigation on transfer learning was motivated by the fact that humans intelligently use knowledge learned from one task and apply it to the tasks in other similar domains that allow humans to learn fast and efficiently. In the field of machine learning, transfer learning was fundamentally motivated from the NIPS-95 workshop on "Learning to learn", that was focused on the need of long-running machine learning algorithms that can reuse the knowledge learned previously. Learning to learn, life-long learning, knowledge transfer, inductive transfer, multi-task learning, knowledge consolidation, context sensitive learning, knowledge-based inductive bias, and so on are the alternative terms used for transfer learning. Figure 2.10 shows the general difference between the machine learning process with traditional pipeline and with transfer learning.

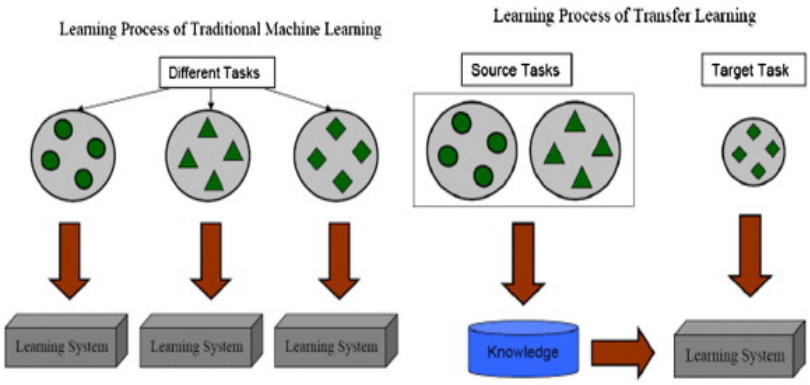


Figure 2.10: Traditional learning machine learning approach (left) and machine learning with transfer learning (right). Source: [37]

Transfer learning is also closely related to the domain-adaptation. For example, if there is a news spam filtering model learned from a specific language, then if one wants to filter out the spams in the data that is the news corpus in different language or may be even with differently structured corpus (blogs, forums, etc.), then this notion of transferring the learned parameters and weights from one model to a different but related domain is called domain adaptation.

In image classification tasks where models are developed using the complex convolutional neural networks, in practice, very few practitioners train the whole network from scratch every-time they want to classify new set of images. Transferring weights from the the previously learned models and using those weights to extract features from the new set of images or to re-weight the newly learned model are the commonly adopted techniques that conforms to the transfer learning techniques in the field of deep neural network.

In a nutshell, in this chapter, we discussed the technical background on various machine learning based topics relevant to the methodology and experimental design of our thesis work. In the next chapter, we present the literature review of different

research works relevant to our thesis.

CHAPTER 3

RELATED WORK

CNNs are widely applied in image classification and segmentation tasks. Transfer learning helps to modify already trained models (pre-trained models) on large scale datasets such as ImageNet [10] in a specific application domain. In fact, using pre-trained CNNs models (usually checkpointed binary weight files) as weight initializers for training same architecture in different application domains is common in visual recognition tasks. This process is referred to as transfer learning in the context of neural networks. Recent research shows that using pre-trained models obtained by training general data followed by application-specific fine-tuning yields significant performance improvement in the image classification task [20]. In this chapter, we discuss the related works relevant to our research from all the above-mentioned fields of study.

3.1 Novelty Detection

Relatively few studies address novelty detection with neural networks. In 2013, Smagghe et al. presented a variant of feedforward multi-layer perceptrons based Image Receptive Fields Neural Network (IRF-NN) algorithm adapted to image recognition which is further analyzed to detect unknown images [53]. Unlike our approach where we make use of deeper convolutional neural networks highly successful in visual

recognition, they have employed the multi-layer perceptrons and had demonstrated the novelty detection in the context of multi-object recognition and localization of images.

Smagghe et al. in 2015 introduced the concept of local learning for multi-class novelty detection tasks [3]. Even though the results from this paper are evaluated in different benchmark image datasets, the results are very poor (AUROC in the range of 0.6). They apply nearest neighbour algorithm to select the smaller sample size to train a local multi-class novelty detection algorithm which is then used to compute the AUROC score with test data including both the known and novel dataset. Additionally, unlike this work, our work importantly focuses on feature extraction from CNNs and then applying those features with the novelty detection algorithms. The features extracted from CNNs have been proven to provide enriched representational information about the images. The paper [3] actually concludes the investigation of deep CNNs in conjunction with novelty detection algorithm to be a potential field of study.

Giacomo et al. addresses the problem of automatic novelty detection in images by training a model to represent patches belonging to the training set of normal images [5]. Basically, they consider a model based on sparse representations of images and monitor the reconstruction error to improve the detection performance using OMP algorithm. Unlike our approach, their methodology does not make use of convolutional neural networks and the highly represented features extracted from CNNs. They compute the AUROC score using very small test data and the images in their experiments are from different domain than ours.

Many of novelty detection algorithms have been proposed in the last few decades. The main idea behind most novelty detection algorithms (esp. the ones used in this

thesis) is to identify the strict regions where the normal instances (not novel) lie in the feature space. The novelty score is given as a form of distance from these regions or sometimes as a probability that the example is not contained in any of these regions. The clustering based novelty detection techniques obtain their scores by using standard clustering algorithms such as K-Means [54] and Gaussian Mixture Models [1].

In the remaining paragraphs, we will further discuss about the application of novelty detection and outlier detection algorithms in image data that come from different type of sources. J. Theiler et al. developed an approach to identify the most unusual samples in the multispectral images. They used data attributes resampling technique to produce a background class which were then distinguished from the original dataset implementing binary classification [58]. S. Srinivasan et al. presented an approach to detect pornographic images in the internet leveraging techniques like texture analysis and face detection for non-adult content identification. They adopted techniques based on the analysis of object-level and pixel-level image contents [48]. L. Shamir et al. applied computational tools from Sloan Digital Sky Survey (SDSS) for automatic detection of peculiar galaxy pairs from the collection of nearly 400000 galaxy images. They found that the peculiar images detected by their technique involved lots of noise and weren't actually the peculiar images and thereby concluded the paper by settling up the peculiar images by manual inspection of the detected images.

In 1969, Frank E. Grubbs presented a concept of determining statistical outliers in the observed dataset [17]. This paper serves as one of the best baseline experimental works and a significant tutorial article for many successor researches in outlier detection evaluated based on the tests of significance. C. Zhu et al. in [66] introduced an

outlier detection approach which detects anomalies based on user-provided examples and a user-specified part of objects to be detected as outliers. All objects are at first mapped into a multi-granularity deviation factor (MDEF)-based feature space. This mapping is claimed to have captured the outlier-ness of the objects. The limited examples are then augmented preserving the statistical properties of the available examples and were classified using the marginal property of the SVM.

In [15], M. Goldstein et al. presented the thorough survey and evaluation of 19 different unsupervised anomaly detection algorithms on 10 different dataset from different application domains.The algorithmic performance, computational complexity, impact of parameter tuning and the behavior of local/global anomaly detection techniques have been studied in different setups. This paper suggests that local anomaly detection algorithms perform poorly on datasets containing global anomalies, however the global anomaly detection algorithms perform average on local problems; which suggests that it is wise to use a global anomaly detection algorithm if there is not enough information about the nature of anomaly we are dealing with on the available dataset. This paper also provides recommendation on case specific algorithmic usages. This paper, however, does not provide the use cases for image dataset and their experimental evaluations do not consider high dimensional feature set as well. V Hodge et al. also presented the extensive overview of outlier detection techniques developed in machine learning and statistics [19]. Many other similar surveys [40] were specifically focused towards particular sub-domain of the existing researches. M. Markou et al. presented the extensive study of neural network based [32] and statistics based [31] novelty detection techniques.

K. Koonsanit et. al presented a method[1] for automatically estimating the outlier

[1]http://www.ipcsit.com/vol11/1-ICNEE2011-N002.pdf

regions in unlabeled datasets. Their data mining algorithm implements co-occurence matrix techniques to determine outlier regions in highly pixelated satellite images. The co-occurrence matrix is first used to estimate the number of clusters, which is followed by a step that implements a standard clustering algorithm which discriminates outlier regions by automatic thresholding. Finally the outlier regions are depicted as indicated by the values less then some threshold. I. Delibalta et al. in [9] presented an online outlier detection algorithm applied on sequential dataset. The algorithm first constructs a score function and then fits a model to the observed data. This score is then used to evaluate newly seen observations. They constructed score function with adaptive nested decision trees. The original data space is partitioned with the nested soft decision trees in a hierarchical manner and an adaptive approach is used to mitigate overfitting issues.

There has been a large increment in the volume of satellite images (the Earth Science data) over the recent years [2]. Analysis of such large and dynamic images is very challenging because of distributed storage across the geographical locations. K. Bhaduri et al. presented the outlier identifying algorithm that is operated in almost fully distributed manner - only less than 5% of the images were needed to be analyzed in a centralized manner. Vertical data partition and specialized pruning rule resulted in higher accuracy, maintaining a low communication cost while processing a terabytes of data.

3.2 Convolutional Neural Networks (CNNs)

Recently, Sabokrou et al. [46] dealt with the problem of anomaly detection in video by using convolutional transfer learning in order to detect anomalies in crowded scenes.

The main procedure was based on extracting vectors of features from an already pre-trained convolutional neural network (AlexNet without its fully connected layers) and use these vectors as input for a Gaussian classifier. In this work the authors state that the features extracted directly from the pre-trained neural network are in general representative enough and there is no need to fine-tune the pre-trained network to improve them. In this thesis we, however, show that fine-tuning the convolutional neural networks (in supervervised and unsupervised setups) improves the performance over the use of only pre-trained models (without fine-tuning).

L. Cavigelli et al. extended the image classification task employing CNNs to make use of higher-dimensional multispectral input images [6]. K. Nogueira et al. presented an analysis and extensive experimental evaluation of three possible ways of using the existing CNNs: full training, fine tuning and using models as feature extractors [36]. They discuss about the overhead (time and space) that researchers have to face while fully training the CNNs for the new dataset. They showed that fine-tuning specific layers of CNNs properly applied based on the nature of dataset being experienced produces better models that can generate sufficiently representative features. This approach saves researchers from spending a lot of time fully training the models for each domain-specific dataset they might experience in their research. Similar to this work, W. Zhou et al. [65] also evaluated pre-trained and fine-tuned CNN architectures on a public scene dataset to investigate whether these models are able to learn dataset-specific features or not.

3.3 Transfer Learning

In this section, we discuss some pioneering literature on state-of-the-art approaches on transfer learning that have been successful specifically in the field of deep neural networks and computer vision.

In many deep neural networks, selection of filters (*aka* kernels) and generated feature descriptors over each layer of the network in many competition winning CNN models exhibit exciting characteristics. In first few layers, CNNs tend to learn features like Gabor filters, edges and color blobs which are general properties that most of the natural images possess. In the later layers, these CNNs tend to learn more specific features associated with the particular data that the network is being trained on. J. Yosinki et al. investigated transferability of features learned from a source task to target task and they found that feature transfer has almost always a boosting effect in target domain even when the source and target tasks are distant in terms of their applicability [63]. However, practitioners should always be cautious about the drawback of negative transfer learning.

A. Razavian et al. presented a series of experiments based on several recognition tasks using the CNN model, *OverFeat* [49], investigated transfer of generic descriptors from the CNNs in range of dataset, and concluded that the the features extracted from these CNNs are indeed the most representative descriptors of the images and suggested that these features be used in most visual recognition tasks [50]. J. Donahue et al. released an open source project, *DeCAF*, that implements the deep convolutional activation features that have sufficient representational power and generalization ability due to transfer learning [11].

T. Chen et al. [7] introduced a technique that transfers information stored in one

neural network to other neural networks, which eventually accelerates the training task of the significantly larger neural networks. While working with the large scale data in a neural network, researchers spend time training and experimenting with a lot of networks, of which many are a waste. The authors presented two Net2Net methodologies that are based on the function-preserving transformations: in both wider (more units in hidden layers) and deeper (more hidden layers) target neural networks. The smaller networks were trained and the knowledge stored in such networks were acquired for the initialization of the larger networks.

Z. Tang et al. used transfer learning to train Recurrent Neural Network (RNN) using the knowledge already trained from the DNNs [57]. The investigation showed that transfer learning boosts the training performance of target domain even when the source domain has the lower representational power even with the limited training dataset. DNNs are assumed to be weaker than RNNs

In other popular surveys [37, 60], authors have generally presented the study of categorization of transfer learning. The major categories discussed are inductive transfer learning, transductive transfer learning, unsupervised transfer learning, Bayesian transfer learning and hierarchical transfer learning. These are categorized based on the availability of the labeled data on the source and target domain and also the nature of task - for instance, inducing predictive modeling, bayesian modeling, building complex models using the many simple models, etc.

CHAPTER 4

METHODOLOGY

In this chapter, we discuss in detail the algorithmic implementation of convolutional neural networks and novelty detection algorithms in supervised and unsupervised learning context. Just to recap the goal of our thesis, following is the methodological outline

Problem Definition. We define the problem of identifying unknown landscape image as novelty detection task. Given the multiple types of landscape images as training experiences to the novelty detection algorithms, we want to investigate how well our novelty detection algorithms perform in detecting unknown landscape images given the combination of known and unknown landscape images (test data) not seen during the training session. Thus we say that our applied machine learning algorithms experience landscape images in both training and test session and evaluate the performance in terms of novelty score with test data. We use "test phase" and "scoring phase" interchangeably in this chapter.

To achieve the above-mentioned goal of building novelty detection machine learning technique for our problem domain, we first present a baseline approach of using pre-trained neural networks to extract representational features of landscape images before applying standard novelty detection algorithms. Then we present more sophisticated ways of using and training CNN models with transfer learning approach in

supervised and unsupervised learning contexts.

4.1 Baseline procedure

A baseline procedure that can be used for unknown landscape type identification with transfer learning is to extract feature representations from pre-trained convolutional neural network models (already trained with large dataset like ImageNet [10]) and use these features (vector of real values) to train and evaluate standard novelty detection algorithm.

4.1.1 Feature Extraction from pre-trained CNNs

Pre-trained convolutional neural network models that were originally trained on a large scale natural image dataset like ImageNet can be directly generalized to other datasets specific to other application domains. The main advantage of these pre-trained networks is that they have already learned a rich set of features from a wide range of natural images. Traditionally, machine learning with image datasets used to require a lot of effort in hand-picking important features i.e. most of the work used to be focused in feature engineering. Since convolutional neural networks are very powerful in learning end-to-end hierarchical feature representations from raw images, it is very common in visual recognition tasks to use the convolutional network models as feature extractor. Like mentioned earlier, the downside of a neural network is that they require a lot of data to be properly trained and demand expensive computational resources and experimentation time. Fortunately, researchers from academia and industry have already trained such powerful CNN models with large-scale datasets such as ImageNet and have made the trained models (the learned parameters of

the CNNs) publicly available for other researchers to use in other relevant tasks. For example, the deep learning library Caffe [21] has it's model zoo from where we can obtain these pre-trained models and apply these quickly as feature extractors in just a couple of minutes. In this thesis, we use three pre-trained CNN models *Alexnet* [25], *VGG19* [52] and *GoogLeNet* [56] obtained from the Caffe library. Like every other neural network, convolutional neural networks also have hierarchical and layered-based network architecture. The raw input images are fed from the lowest input layer and the network successively learns from general (eg. edges, color blobs) to more abstract (eg. football stadium, river, airport etc) concepts in images through a hierarchical representational learning process. The pre-trained models that we are using in our implementation were all trained originally on millions of natural images over more than twenty thousand categories for multi-class classification tasks and these models have learned very general features (applicable to nearly any type of existing images) in the lower layers. Since our landscape dataset is very small in comparison to ImageNet, we were able to extract rich features for our landscape images even from the higher-level layers (fully connected layers on the output end). Usually, once these CNNs are used to convert the original images(raw landscape images in our case) into a set of meaningful features, these features can be then used to train a new classifiers such as linear SVM applied to a new task which the original CNN models were not trained for. This is one of the commonly employed and simplest forms of transfer learning in context of convolutional neural networks in visual recognition.

In this thesis, one of our major goals is to use the extracted features using pre-trained models and re-trained models in supervised and unsupervised context (to be discussed later), not with a classification task, but with standard novelty detection

algorithms. We describe the details of novelty detection algorithm with CNN feature extraction in next section.

4.1.2 Novelty Detection with CNN features extraction

Novelty detection as described in [42] is the task of detecting data that differs in some respect from the data that is available during training. By definition, novelties (or unknown observations) are not present in the training set and appear only on the test set. In order to identify unknown types of landscape, we define a novel example in our application domain, i.e. a type of landscape that is novel from those present in the training set (see [33]). In this thesis, we consider four commonly used effective novelty detection algorithms: (i) Gaussian Mixture Models, (ii) Bayesian Gaussian Mixture Models, (iii) One-class Support Vector Machines and (iv) Isolation Forests. Please refer to Chapter 2.2, [18] and [33] for a detailed description of each one of these methods. Implementation of novelty detection involves taking the extracted feature representation from CNN models as input, train each of these novelty detection algorithms using known landscape image samples (datasets detail and data preparation is explained Chapter 5) as training data and finally compute the novelty score for the landscape images in test data. These novelty detection algorithms return as output a score specifying the degree of novelty for each of the landscape images in the test data [1]. We describe our novelty detection procedure in algorithm 1.

Algorithm 1 takes in input a pre-trained CNN model N s.t. its last layer is exactly the one that we decide to extract the features from. In addition, we also input two set

[1] if the score is the belonging degree to the training set, the novelty score can be easily obtained multiplying it by -1

Algorithm 1 Novelty Detection with CCN feature extraction.

procedure ND(N, $IMGs$, $TestIMGs$)
Input : N is the pre-trained (or fine-tuned) model for feature extraction, $IMGs$ is a set of known images that will be used as a training set to train novelty detection algorithms, and $TestIMGs$ is the set of images (both known and unknown types) where our procedure will search for type of unknown landscape w.r.t. $IMGs$.
Output : *score* is the novelty score for each image in test set ($TestIMGs$).

Trainig session
    $Feats \leftarrow$ getFeatures(N, $IMGs$)
    for ($i = 1$ to k) **do**
        $m_i \leftarrow$ trainNovelDetectModel($Feats$)
    end for

Scoring session
    $testFeats \leftarrow$ getFeatures(N, $TestIMGs$)
    for ($i = 1$ to k) **do**
        $score_i \leftarrow$ getScore(m, $testFeats$)
    end for
    $score = \frac{\sum_{i=1}^{k} score_i}{k}$

    return *score*
end procedure

of images: $IMGs$ is the set of known types of landscape images used as training and $TestIMGs$ is the set of images where we want to find the unknown type of landscape. The output of this procedure is a score for each image in $TestIMGs$. This score tells us how much the landscape type in this image is unknown. The procedure is divided in two parts: Training session and Scoring session. The Training session consists of two main tasks: first it extracts the training features, these training features are obtained by running the training images in $IMGs$ through the CNN N, and second uses these extracted features to train the novelty models m. We train all four novelty detection algorithms using these extracted features. After the novelty detection models m are trained we proceed with the Scoring session. The Scoring session is responsible for extracting the test features from the the images in $TestIMGs$, these features are again extracted using the CNN N, and scoring them with the trained novelty model m. Usually, all the standard novelty detection procedures are subjected to random choice. This make models unstable, meaning that different runs of the procedure can obtain different models (line 4), and scores. To overcome this issue and try to get a more stable result, we decided to execute the training procedure of the novelty detection model m k times (line 3). Once the $m_1, \ldots, m_k$ models are obtained, we score each training image in $TestIMGs$ with each one of the trained models and average the produced scores (line 10) to obtain a unique ensemble score.

Once the ensemble scores are obtained, two performance metrics - AUROC and Average Precision [18] - are computed to validate the results with respect to the the ground-truth (i.e. the information which images are known and which images are unknown landcapes in test data).

4.1.3 Parameter tuning

Standard novelty detection procedure as (i) Gaussian Mixture Models, (ii) Bayesian Gaussian Mixture Models, (iii) One-class Support Vector Machines and (iv) Isolation Forests are all different types of parametric algorithms. These parameters must be tuned to understand appropriate algorithmic parameter values for the specific application. Some tunable paramters for One-class SVM are choice of kernel, norm (L1 or L2) and kernel coefficient. For clustering based algorithms like Gaussian Mixture Models and Bayesian Gaussian Mixture Models, number of estimated cluster is a parameter that needs to be selected appropriately. Similarly for isolation forest, we need to decide on what type of and how many decision tree classifiers we want to apply. In a standard classification task one of the best ways to understand a parameter is to try all the possible values of that parameter and perform on the training set a K fold cross validation and average the obtained accuracy or AUROC score. In a novelty detection task the problem lies in the fact that the training set does not contain the novel class, so it is not possible to compute any AUROC score. One approach that can be used for our application, identification of unknown (or novel) landscape, is to consider as unknown images for the parameters decision the ones from dataset like Imagenet. The train of the procedure is always performed with the landscape images but the testing in all the folds is done with both, consequently it is possible to compute the AUROC by assuming the images of ImageNet unknown. We also use Bayesian Information Criterion (BIC) score to select among trained mixture models.

4.2 Novelty Detection Fine-tuning

One more sophisticated way to perform transfer learning in context of CNNs is by fine-tuning these CNNs. This procedure consists of refining the network (training it again) with the new specific dataset by continuing with the training checkpoint models that were initially used to train large scale natural dataset (in our case ImageNet). The fine-tuning is usually related to classification tasks. In this section we show how to fine-tune a network in the context of novelty detection. Once the network is fine-tuned, the baseline procedure can be applied. In the following sections we discuss two fine-tuning procedures, one more restrictive called supervised fine-tuning and the more general unsupervised fine-tuning procedure.

4.2.1 Supervised Fine-Tuning

Usually fine-tuning is performed in standard classification tasks where it is possible to provide examples of the new classes (at least two different classes) that the CNN is going to classify. In the novelty detection problem, only the known type of classes are available in the training, i.e. in our case only the known type of landscape. In order to perform the standard fine-tuning, we assume that our images belong to a set of disjointed categories that are all the different types of known landscape. Algorithm 2 describes our supervised fine-tuning procedure. This procedure takes in input a CNN N to fine-tune, a set of known images to be used as training set during fine-tuning, and a set of classes C pertaining to all the images in $IMGs$ representing specific type of known landscape. Note that the last layer of the input CNN N is exactly the layer chosen for the feature extraction.

First the procedure will extract the number of unique classes n from the classes

set C (line 2). Second a new fully connected layer is added to the last layer of the CNN N. The number of neurons of this new layer is equal to n (number of distinct classes). In the third step the $Train$ method is applied to fine-tune the CNN on the new dataset (i.e. landscape image dataset) (line 4). The learning rate for the $Train$ (gradient descend) method in the fine-tuning procedure usually decreases with the layer number that we are re-tuning. Usually the last layers have a learning rate coefficient bigger than the ones earlier in network layer. This is similar to freezing the earlier weight layers and training only the last layer. The motivation is that we want to operate a transfer learning procedure that moderately changes the weights by keeping most of the previously learned weights unaltered. In particular, the initial layers (near to input end) should have lower change because they mostly represent general features related to all natural images by itself (e.g. edges, color blobs etc.) while the higher level layers (near to output end) - the last fully connected layer in our case - should be learned more in order to allow the training process to learn features specific of the new application context (e.g. landscape images in our case). Finally, the added layer is removed and the procedure returns the new fine-tuned network that is further applied for feature extraction and novelty detection using the baseline procedure.

The fine-tuning based on types of the known type of landscapes allows us to linearly separate the features of each known landscape from the other (because we added only one layer more after one of the features). This separation allows the baseline approach to better bound the margins of the known type of landscapes. This is because the similar landscapes (same types) tend to be located close to each other and linearly separated from the others. Consequently, the boundaries for each class will be more tightly specialized for each specific class. This improves the ability

Algorithm 2 Supervised Fine-Tuning procedure.

1: **procedure** FINETUNE(N,$IMGs$,C)
Input : N is the CNN model to be used as weight initializer, $IMGs$ the set of known images to be used as training data, and C is a vector the classes (labels) for each image in $IMGs$ to be used for supervised fine-tuning.
Output : NF fine-tuned network (i.e. after transfer learning).
2: $n \leftarrow$ GETNUMBEROFCLASSES(C)
3: $NF \leftarrow$ ADDFULLYCONNECTEDLAYER(N, n)
4: $NF \leftarrow$ TRAIN($NF, IMGs, C$)
5: $NF \leftarrow$ REMOVELASTLAYER(NF)
6: **return** NF
7: **end procedure**

to recognize unknown landscapes, e.g. landscapes that are somehow a combination of the known ones will tend more to be considered unknown. In the experiments, in Section 5.3 we observe this phenomena.

Since the fine-tuning approach requires us to have the information of these classes, we call this procedure supervised fine-tuning. Unfortunately, providing the type of landscape of each known image can represent a big limitation. In the next section, we will provide completely new unsupervised fine-tuning procedure that does not need the class type information in input.

4.2.2 Unsupervised Fine-Tuning

In real world scenarios, it is usually difficult to obtain the categorization (classes) information of the known images. In this section, we provide an unsupervised procedure that groups similar images in base to the structure of the features extracted from the CNN. We use a clusterization procedure performed by a Gaussian Mixture Model [2] that as a result associates each image to a cluster. Then, we use each cluster as a

[2]Gaussian Mixture Model is a distance-based, probabilistic and parametric clustering approach less sensitive to noise. In density-based clusterization, it can be difficult to linearly separate clusters

specific landscape class and apply the supervised procedure explained in Section 4.2.1 to fine-tune the network. For stability reasons we decide to run the Gaussian Mixture Model training procedure multiple times and by using the consensus clustering [55] we ensemble all the results produced by the Gaussian Mixture Model execution in a unique clusterization. Once this unique clusterization (the classes) is obtained, the supervised fine-tuning procedure is performed and returns a fine-tuned network. The algorithm tends to clusterize the images with the property that the features of each cluster can be linearly separable from the ones of the other clusters. Algorithm 3 more formally describes this unsupervised procedure that requires input from the networks with the already trained weights N, the Images for the re-tuning, and the number of different classes contained in the known images. The output is the final fine-tuned network.

The algorithm extracts the features from the current CNN N (line 3). After features for each image are extracted, the algorithm trains k Gaussian Mixture Models (by performing different runs) and it adds all the clustering result (the association from image to cluster) to the set CS. In line 8 all the results of each Gaussian Mixture Model in CS are merged in a unique clusterization C by using the consensus clustering. As a last step (line 9) the original network N is fine-tuned with C obtaining a fine-tuned network Net. The effectiveness of the transfer learning process is always guaranteed by the different small learning factors of the supervised fine-tuning. This approach requires the estimation of the parameter NC, this can be estimated with the same procedure in Section 4.1.3 by obtaining the scores with the Algorithm 1.

---

from each other. Since this is one of the main effect of the supervised fine tuning, we avoid to use density-based clusterization approach. Distance-based clusterization is naturally more suitable to linear separation even if it still contains the quadratic factor.

Algorithm 3 Unsupervised Fine-Tuning Procedure.

1: **procedure** UNSUPERVISEDFT(N ,$IMGs$,NC)
Input : N is the CNN model to be used as feature extractor for GMM and weight initializer for fine-tuning, $IMGs$ is set of known images to be used as training data, and NC is hyperparameter which is effective number of unique clusters to be considered (obtained from clusterization algorithm, GMM) to obtain class labels for training data.
Output : $bestNet$ is the fine-tuned network in unsupervised setup.
2: $Cs \leftarrow \emptyset$
3: $Feats \leftarrow$ GETFEATURES(N,$IMGs$)
4: **for** ($i = 1$ to k) **do**
5: $m_i \leftarrow$ GMM($Feats$, NC)
6: $Cs \leftarrow Cs \cup \{m_i.getClusters(IMGs)\}$
7: **end for**
8: $C \leftarrow$ CLUSTERINGENSEMBLE(Cs, $IMGs$)
9: $Net \leftarrow$ FINETUNE(N,$IMGs$, C)
10: **return** Net
11: **end procedure**

4.2.3 Clustering Ensemble

We implement ensembles of clustering technique by Alexander et al. [55] in order to obtain and assign stable cluster labels for each landscape image in the training set before training the convolutional neural networks in an unsupervised setup as presented in algorithm 3. We train multiple instances (user defined k times) of Gaussian Mixture Model for one feature set of the training landscape images, and compute a single vector of cluster labels for each image in the training voted by a clustering ensemble algorithm. This enables us to guarantee that the cluster labels obtained are stable and were able to capture most of the variance characterized in the feature space while training the clustering based novelty detection algorithm. The quality of obtained cluster labels are crucial while training the CNNs and thus affects the overall novelty detection performance.

4.2.4 Iterative unsupervised fine tuning with Novelty Detection

We also combine algorithms 3 and 2 discussed in previous sections along with ensembles of clustering and conduct further iterative experiments on unsupervised novelty detection. We do so in order to evaluate the quality of unsupervised fine-tuning approach and stability of cluster labels obtained from clustering based novelty detection algorithm Gaussian Mixture Model. The cluster labels learned in each iteration are obtained using the consensus voting with ensembles of clustering algorithm (4.2.3). These consensus cluster labels are used to train convolutional neural networks in each iteration. Algorithm 4 performs h iterations, extracting the features of training images $IMGs$ and test images $TestIMGs$ from the current network Net (line 6 and line 7); Note that in the first iteration, the algorithm extracts the features from the initial pre-trained network N (line 4). After feature extraction, the algorithm trains k Gaussian Mixture Models (by performing different runs) and it adds all the clustering labels (the association from image to cluster) to the set CS and the computed novelty scores (AUROC and Average precision) to the set $score_{ens}$. In line 15, all the results of each Gaussian Mixture Model in CS are merged in a unique clusterization C by using ensembles of clustering to obtain the consensus label for the images in the training set. The aggregated scores for k instances of Gaussian Mixture Models are aggregated and added to a set $score$ as a score result for this iteration $iter$. As a last step, (line 17) the new network Net is fine-tuned with new consensus cluster labels and N as weight initializer.

This approach helps us avoid the ensemble results to drastically change at each iteration. The more we iterate the algorithm, the lower is the possibility that small clustering variations at each iteration can modify the ensemble result obtained with

all results of each iteration. Also this approach requires some parameter estimation procedure and the number NC can be estimated with the same procedure in Section 4.1.3 by obtaining the scores with Algorithm 1. In addition a similar procedure can be performed to establish how many iteration h are needed.

Algorithm 4 Iterative Unsupervised Fine Tuning Procedure

procedure UNSUPERVISEDFT(N ,$IMGs$, $TestIMGs$, NC)
Input : N is the CNN model to be used as feature extractor for GMM and weight initializer for fine-tuning. $IMGs$ is set of known images to be used as training data. $TestIMGs$ is set of images including both known and unknown types to be used as test data. NC is hyper-parameter which is effective number of unique clusters to be considered (obtained from clusterization algorithm, GMM) to obtain class labels for training data.
Output : $bestNet$ is the fine-tuned network in iterative unsupervised setup.
    $Cs \leftarrow \emptyset$
    $score \leftarrow \emptyset$
    $Net = N$
    for ($iter = 0$ to h) **do**
        $Feats \leftarrow$ GETFEATURES(Net,$IMGs$)
        $TestFeats \leftarrow$ GETFEATURES(Net,$TestIMGs$)
        $score_{ens} \leftarrow \emptyset$
        for ($i = 1$ to k) **do**
            $m_i \leftarrow$ GMM($Feats$, NC)
            $score_i \leftarrow$ GETSCORE(m_i,$TestFeats$)
            $Cs \leftarrow Cs \cup \{m_i.getClusters(IMGs)\}$
            $score_{ens} \leftarrow score_{ens} \cup score_i$
        end for
        $C \leftarrow$ CLUSTERINGENSEMBLE(Cs, $IMGs$)
        $score \leftarrow score \cup AVERAGE(score_{ens})$
        $Net \leftarrow$ FINETUNE(N,$IMGs$, C)
    end for
    return Net, $AVERAGE(score)$
end procedure

CHAPTER 5

EXPERIMENT AND RESULT ANALYSIS

In this section we first provide a description of the landscape image datasets (Section 5.1) used in our experiments and data pre-processing. We then explain the overall experimental settings and finally we present the detailed discussion about the obtained results.

5.1 Datasets and Preprocessing

We use two known landscape imagery benchmark datasets for our experiments: UC Merced Land Use Dataset (UCM)[61] and High-resolution Satellite Scene Dataset (HRS)[8]. Both datasets contain different landscape scenes representing general landscape objects.

UC Merced Land Use Dataset (UCM). The images in this dataset were extracted from the USGS National Map Urban Area Imagery collection for various urban areas around the USA. UCM contains 21 classes and 100 images per each class (2100 images in total). UCM has the following landscape classes: agricultural, airplane, baseball, diamond, beach, buildings, chaparral, dense residential, forest, freeway, golf course, harbor, intersection, medium residential, mobile home park, overpass, parking lot, river, runway, sparse residential, storage tanks and tennis court. The capture resolution of each image in this dataset is 1 foot per pixel. Images are in

Figure 5.1: Examples from UC Merced Land Use (UCM) Dataset.

Figure 5.2: Examples from High-resolution Satellite Scene (HRS) Dataset.

TIF format and contain 8bit three channel (RGB) information. Example landscape classes from UCM dataset are shown in Figure 5.1.

High-resolution Satellite Scene Dataset (HRS). This high-resolution satellite scene dataset contains 19 different landscape classes, including: airport, bridge, desert, farmland, football field, forest, meadow, mountain, park, parking, pond, port, railway station, river, viaduct, commercial area, industrial area, and residential area. For each class, there are at least 50 samples, totalling to 900+ images. Originally each image is 600x600 pixels in size. Images are in JPEG format and contain 8bit three channel (RGB) information. Example landscape classes from HRS dataset are shown in Figure 5.2.

Images representing unknown (novel) types. For landscape images to be used as unknown, we manually extracted two new types of landscape images from GoogleMaps: Nuclear Power Plants and Military Settlements. We processed these unknown images in the same way the benchmark datasets were processed to be used as raw input in our implementation pipeline. Each class contains 100 images and

Figure 5.3: Examples of military settlement (left) and nuclear power plant (right) images used as unknown data classes.

have been preprocessed in the exact same way as the two other landscape datasets. Example of landscape images used as unknown data points are shown in Figure 5.3. These images are only used for evaluating novelty detection algorithms and do not take part in the training phase of CNNs and novelty detection models.

Data preprocessing. At first, we obtain the above-mentioned landscape image datsets (all known and unknown types). The images in both of the known type datasets used are available with their respective class labels (i.e. landscape category). Implementation of Alexnet, VGG19 and Googlenet in deep learning library, Caffe [21], requires the input images to be preprocessed in strict ways. All three CNN models being used in this thesis require input landscape images to be resized to 256x256 pixels resolution followed by image transposition to CxHxW and channel order transformation from RGB to BGR. Additionally, the raw pixel values of each images were also required to be scaled up by a factor of 255 before feeding the input images to the CNN networks.

5.2 Experimental Setup

In order to create a feasible experimental environment for novelty detection, we define the distinction of landscape images to be considered as known (normal) and unknown (novel). By using the landscape image datasets described in the previous section and two classes of manually collected critical landscape images, we create the following 4 variation of our datasets:

- **UCM with two unknown types:** All the landscape classes in UCM are considered known with all its images. The only images representing the unknown types for the test are the unknown images extracted from Google Maps: nuclear power plants and military settlements.

- **UCM with three unknown types:** All the landscape classes in UCM except the class "storageTanks" are considered to be known with all its images. The only images representing the unknown types used for the test are the images of HRS related to the class "storageTanks" plus our unknown Google Maps images.

- **HRS with two unknown types:** All the landscape classes in the HRS dataset are considered known with all its images. The only images representing the unknown types for the test are our unknown images.

- **HRS with three unknown types:** All the landscape classes in HRS except for the class of "footballField" are considered known with all its images. The only images representing the unknown types used for the test are the images of HRS related to the class "footballField" plus our unknown Google Maps images.

The motivation behind the consideration of cases with 3 unknown types is that we wanted to verify if our image extraction procedure from Google Maps (extraction of nuclear power plants and military settlements images) can create some particular structure in the image s.t. the classification as unknown is trivial. For this motivation, we remove one class from the known image datasets in the training phase, consider those images of that class as unknown types in order to verify if our approach can still discriminate the known and unknown types regardless of the class of landscape images used in training and validation phase. In addition, in this way it is possible to check if the proposed procedures work only for recognizing as unknown nuclear power plants and military settlements or they work in general with all the possible unknown landscape types. These variants of image datasets with three unknown types provide us the way to make sure that our feature extraction procedure is not biased to any class of landscape dataset and is generalizable to any type of landscape dataset. We do so by applying (training and validating) these variants of dataset with our novelty detection algorithms.

Once each variant (dataset context) is properly defined, we proceed with a 10-folds cross validation [18] to evaluate the quality of our approach. We divide all the known images in 10 folds, each time we select one of the folds of known images and use this image plus our unknown types collectively as the test set. Accordingly, each time all the remaining images in the other 9 folds are used as a training set for our procedures. Note that the training set is constituted only by known images and the test set contains both known and unknown landscape images. In total we have 10 pairs of train and test sets for each of the 4 contexts. With proportions of 90% of known images for training and the remaining 10% of known images plus the two classes of the unknown type images as test set. In total we consider three procedures

in order to identify unknown classes of landscape:

- Novelty detection methods with feature extraction from pre-trained CNN models. We fix the number of training runs of all the novelty detection models k (Algorithm 1) to 25. We also fix the k to the same value for the following two novelty detection training cases involved in our experiments.
- Novelty detection methods with CNN feature extraction from a supervised fine-tuned networks.
- Novelty detection methods with CNN feature extraction from an unsupervised fine-tuned networks.

For each of above cases, we extract features of images in both training and test set from different CNN models and different layers specific to each CNN network architecture. The details about the used CNN network architecture, layers used for fine-tuning the models (both supervised and unsupervised) and feature extraction layers are reported in Table 5.1.

ID	Fine-tuning layer	Feature Extraction	Dim
A-fc7-fc6	Alexnet till fc7	fc6	4096
A-fc7-fc7	Alexnet till fc7	fc7	4096
GoogleNet	GoogleNet till fc_tune	loss3/classifier	1000
V-fc7-fc6	VGG19 till fc7	fc6	4096
V-fc7-fc7	VGG19 till fc7	fc7	4096

Table 5.1: CNN Models with details about Fine-tuning and Feature extraction layers.

Note that when we fine-tune these networks, we use fc7 for Alexnet and VGG19, and fc_tune (additionally appended fully connected layer with 1000 neurons before the classification layer) for GoogleNet. Intuitively, figure 5.4 presents the experimental

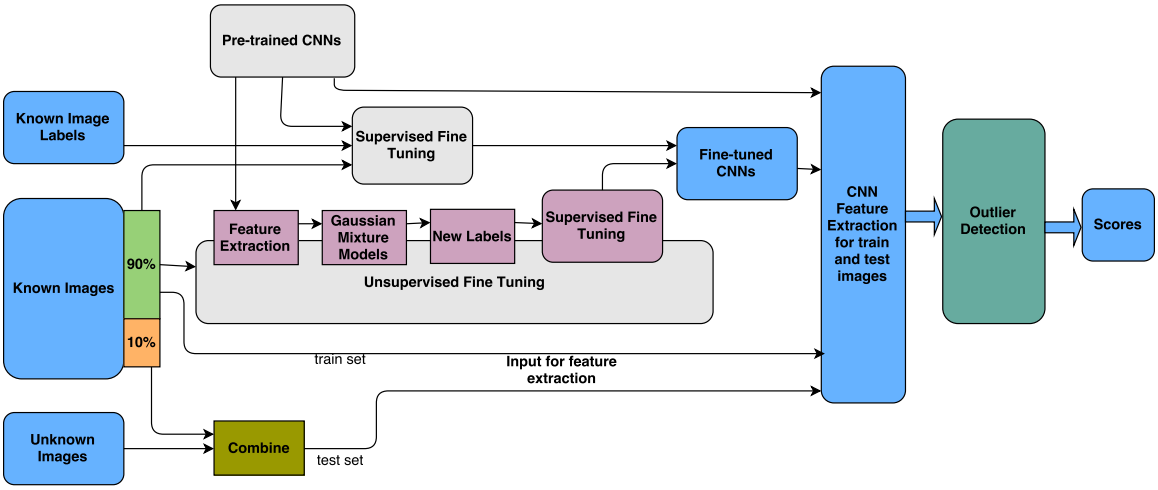
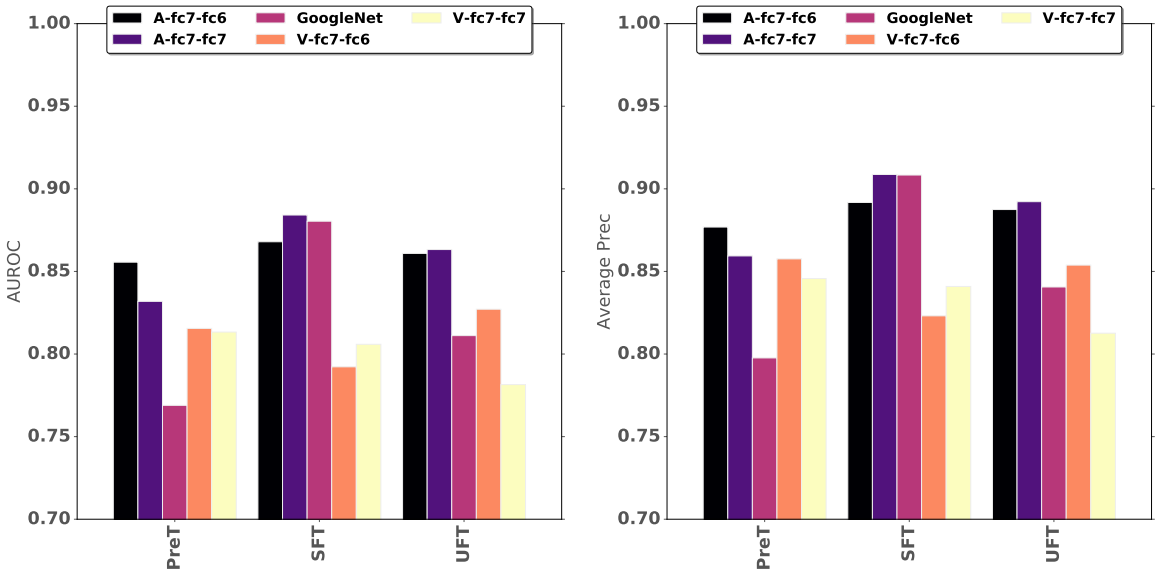


Figure 5.4: Overall workflow and experimental setting for all pre-trained and fine-tuned CNNs.

pipeline: contains a summary of experiments conducted with pre-trained, supervised fine-tuned and unsupervised fine-tuned CNN models, CNN feature extraction, application of novelty detection algorithms and performance score computation. Note that this figure shows the workflow of one fold for each variant of dataset (combination of known and unknown types). Finally, AUROC and Average precision scores are computed as performance metrics for all the experimental instances which are reported and analyzed in the next section (Section 5.3).

5.3 Results and Discussion

Next, we will present the results obtained from our experiments and present the statistical analysis of our performance metrics (AUROC and Average Precision) for different cases. Note that the AUROC and Average Precision scores reported in the following sub-sections are computed as the mean of the values obtained from all 10-folds used during the training and validation of CNN models and novelty detection



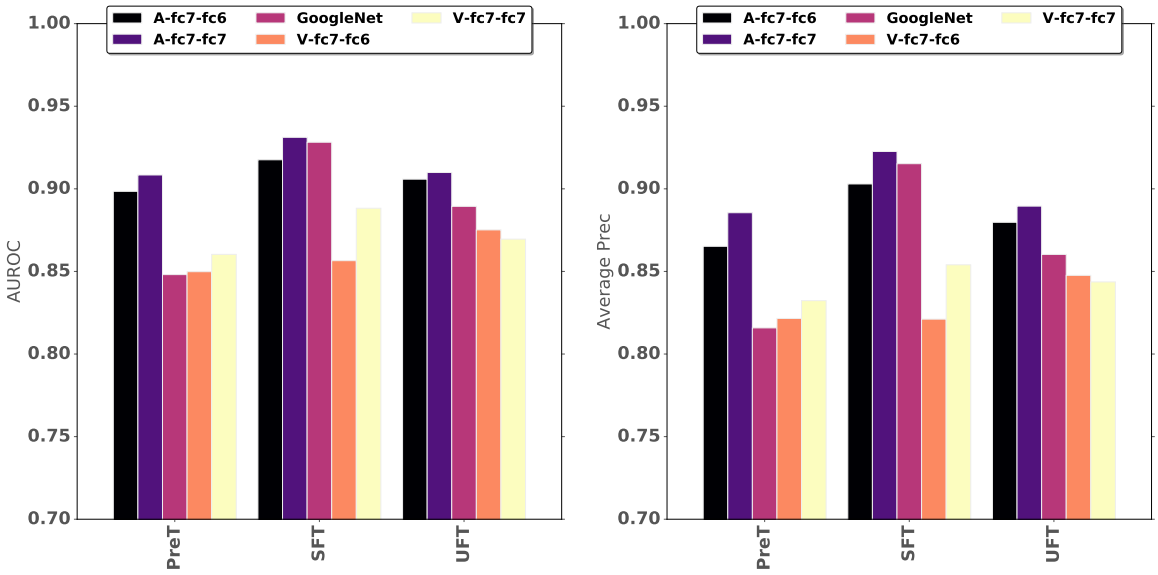
(a) Auroc (b) Average Precision

Figure 5.5: UCM with two unknown types.

algorithms.

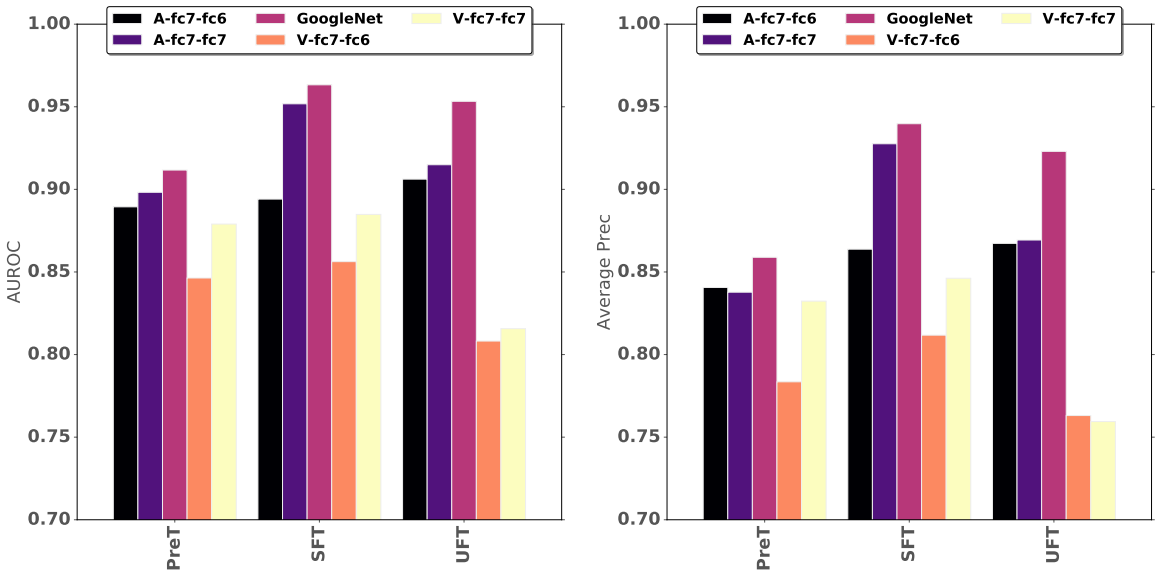
The charts in Figure 5.5 represent the comparison among pretrained (PreT), supervised fine-tuning (SFT) and unsupervised fine-tuning (UFT) procedures in term of AUROC and Average Precision in the case of UCM with two unknown types. In particular, we consider all the CNN models described in Table 5.1 and report the results obtained with the Gaussian Mixture Model. We only show scores for Gaussian Mixture Model because we will see in the statistical significance test that GMM obtains better scores than other novelty detection algorithms - see H4-H6 in Table 5.3 considering the results from overall experiments. While observing the figures 5.5, 5.6, 5.7 and 5.8, keep in mind that when we consider results for pre-trained models (PreT), the fine-tuning layer is not relevant since it is not fine-tuned (so we just use the feature extraction layers of the original network architecture - see 5.1).

Similar to Figure 5.5 representing the UCM dataset with two unknown types, the



(a) Auroc (b) Average Precision

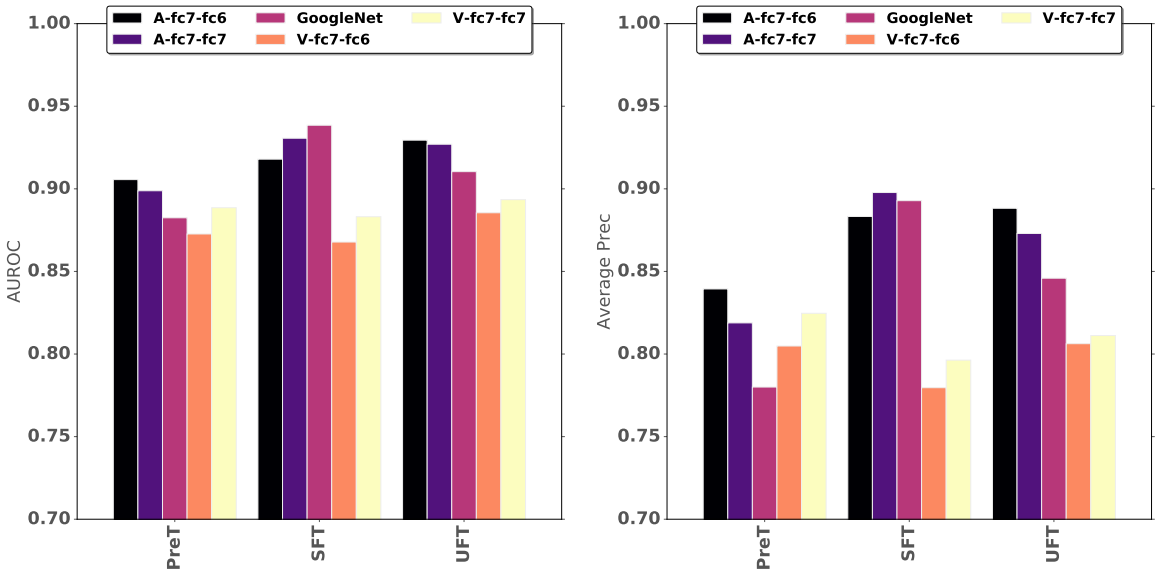
Figure 5.6: UCM with three unknown types.



(a) Auroc (b) Average Precision

Figure 5.7: HRS with two unknown types.

remaining three situations are reported for the similar performance comparison among PreT, SFT and UFT approaches in Figure 5.6 (UCM dataset with three unknown



(a) Auroc (b) Average Precision

Figure 5.8: HRS with three unknown types.

types), Figure 5.7 (HRS dataset with two unknown types(and Figure 5.8 (HRS dataset with three unknown types). A common important observation to note from these figures is that often times (however, not in all cases) SFT and UFT fine tuning approaches improve the performance in terms of AUROC and Average Precision compared to the scores obtained from the pretrained models (PreT) - at least with CNNs Alexnet and GoogleNet. We can see that the performance of UFT approach in general performs better than PreT even with the fact that UFT fine-tuning approach doesn't include the true labels of the known landscape images during training phase. Table 5.2 shows the best results among all the considered networks for each situation procedure in terms of performance metrics (AUROC and Average Precision). By analyzing this table, we can see that the best results obtained from novelty detection with fine-tuning approaches (SFT and UFT) are mostly better than the best case of the PreT approach. Next, we consider the results from our overall experiment and

analyze the statistical significance of those results in various scenarios to find the relevant statistical conclusion about the stated hypotheses.

Table 5.2: Best results for each procedure, dataset variant, and metric.

Dataset	PreT	SFT	UFT
	AUROC	AUROC	AUROC
UCM 2 Unknown Types	0.85	0.88	0.86
HRS 2 Unknown Types	**0.91**	**0.96**	**0.95**
UCM 3 Unknown Types	0.90	0.93	0.91
HRS 3 Unknown Types	0.90	0.94	0.93
	Avg. Prec.	Avg. Prec	Avg. Prec
UCM 2 Unknown Types	0.87	0.91	0.90
HRS 2 Unknown Types	0.85	**0.94**	**0.92**
UCM 3 Unknown Types	**0.88**	0.92	0.89
HRS 3 Unknown Types	0.84	0.90	0.89

5.3.1 Statistical Analysis

As described in Section 5.2, since we conducted extensive experiments (due to the consideration of 10-fold cross validation, different dataset variants, multiple standard novelty detection methods and different CNN models), we decided to formulate these results with several tests of hypothesis. In this way, it is possible to summarize all these results and make sure that our conclusions are supported by statistical evidence. The statistical evidence is verified by the use of statistical t-test which tries to figure out the relationship between any two related sample distributions (distribution of AUROC and Average Precision scores in our case) and provides the degree of confidence to support that relationship via some statistical test values and p-value. We use the level of significances ($\alpha = 0.01, 0.05$) in order to interpret the statistical relationship among two distributions after obtaining the p-value for the

t-test. p-value if obtained less than chosen α suggests that the hypothesis statement maintains the statistical significance with the significance level of α.

We enlist our statistical hypotheses statements with their unique ids in Table 5.3.

Hypothesis ID	Hypothesis Description
H1	Supervised fine-tuning (SFT) approach performs better than non fine-tuning (PreT)
H2	Unsupervised fine-tuning (UFT) approach performs better than non fine-tuning (PreT)
H3	Supervised fine-tuning (SFT) approach performs better than unsupervised fine-tuning (UFT)
H4	Gaussian Mixture Model (GMM) performs better than One-class SVM (OCS)
H5	Gaussian Mixture Model (GMM) performs better than Bayesian Gaussian Mixture Model (BGMM)
H6	Gaussian Mixture Model (GMM) performs better than Isolation Forest (ISOF)
H7	A-fc7-fc6 CNN model performs different than A-fc7-fc7
H8	V-fc7-fc6 CNN model performs different than V-fc7-fc7
H9	A-fc7-fc7 CNN model performs better than Googlenet
H10	Googlenet CNN model performs better than V-fc7-fc7
H11	A-fc7-fc7 CNN model performs better than V-fc7-fc7

Table 5.3: ID and description of our statistical hypotheses statements.

For each of the hypotheses shown in Table 5.3, there are two sample distribution of AUROC and Average Precision scores to be tested for statistical significance. All the corresponding p-values and statistical description of compared sample distributions for all the hypotheses can be looked up in tables 5.4 and 5.5 for AUROC and Average Precision scores, respectively. Collectively, table 5.6 reports the significance of each of the hypothesis tested by statistical t-test. ** represents that there is strong relationship with 0.01 level of significance between two sample distributions being compared characterized by the sample mean values. Similarly, * represents the relationship with 0.05 level of significance between two sample distributions.

HID	S1 (left) vs S2 (right)	S1 (mean, std)	S2 (mean, std)	Absolute mean difference	p-value
H1	SFT vs PreT	0.892, 0.044	0.866, 0.037	0.026	0.002
H2	UFT vs PreT	0.905, 0.041	0.866, 0.037	0.038	0.000
H3	UFT vs SFT	0.905, 0.041	0.892, 0.044	0.013	0.073
H4	GMM vs OCS	0.913, 0.029	0.881, 0.026	0.033	0.000
H5	GMM vs BGMM	0.913, 0.029	0.898, 0.020	0.015	0.000
H6	GMM vs ISOF	0.913, 0.029	0.716, 0.082	0.197	0.000
H7	A-fc7-fc7 vs A-fc7-fc6	0.819, 0.094	0.813, 0.140	0.006	0.289
H8	V-fc7-fc7 vs V-fc7-fc6	0.775, 0.111	0.769, 0.110	0.006	0.169
H9	A-fc7-fc7 vs Googlenet	0.819, 0.094	0.791, 0.146	0.027	0.000
H10	Googlenet vs V-fc7-fc7	0.791, 0.146	0.775, 0.111	0.016	0.012
H11	A-fc7-fc7 vs V-fc7-fc7	0.819, 0.094	0.775, 0.111	0.044	0.000

Table 5.4: Table for AUROC scores: Statistical significance test table obtained by applying statistical t-test against various combinations of experimental results. *S1 (left) vs S2 (right)* column represent two related sample distributions (array of scores). Third and forth columns contains mean and standard deviation of sample distributions S1 and S2 respectively. Statistical t-test has been applied according to the hypothesis as shown in HID column. Refer Table 5.3 for detailed hypothesis description. *Absolute mean difference* column contains the absolute difference of mean values of sample distributions S1 and S2. *p-value* represents the level of significance of the comparison being done.

We prepare the various sample distributions from our experimental results for statistical tests. We basically aggregate the results based on overall fine-tuning approach (H1 - H3), standard novelty detection algorithms (H4 - H6) and CNN models used (H7 - H11). We aggregate the results for both AUROC and Average Precision scores. After the interpretation of these statistical results as shown in table 5.6, we

HID	S1 (left) vs S2 (right)	S1 (mean, std)	S2 (mean, std)	Absolute mean difference	p-value
H1	SFT vs PreT	0.871, 0.046	0.834, 0.028	0.037	0.002
H2	UFT vs PreT	0.885, 0.025	0.834, 0.028	0.051	0.000
H3	UFT vs SFT	0.885, 0.025	0.871, 0.046	0.013	0.106
H4	GMM vs OCS	0.895, 0.018	0.853, 0.031	0.042	0.000
H5	GMM vs BGMM	0.895, 0.018	0.872, 0.016	0.023	0.000
H6	GMM vs ISOF	0.895, 0.018	0.647, 0.040	0.249	0.000
H7	A-fc7-fc7 vs A-fc7-fc6	0.769, 0.119	0.767, 0.168	0.002	0.841
H8	V-fc7-fc7 vs V-fc7-fc6	0.714, 0.115	0.728, 0.136	0.014	0.033
H9	A-fc7-fc7 vs Googlenet	0.769, 0.119	0.743, 0.169	0.026	0.003
H10	Googlenet vs V-fc7-fc7	0.743, 0.169	0.714, 0.115	0.029	0.001
H11	A-fc7-fc7 vs V-fc7-fc7	0.769, 0.119	0.714, 0.115	0.055	0.000

Table 5.5: Table for AVERAGE PRECISION scores: Statistical significance test table obtained by applying statistical t-test against various combinations of experimental results. *S1 (left) vs S2 (right)* column represent two related sample distributions (array of scores). Third and forth columns contains mean and standard deviation of sample distributions S1 and S2 respectively. Statistical t-test has been applied according to the hypothesis as shown in HID column. Refer Table 5.3 for detailed hypothesis description. *Absolute mean difference* column contains the absolute difference of mean values of sample distributions S1 and S2. *p-value* represents the level of significance of the comparison being done.

state the following conclusions for each of the hypotheses described in Table 5.3.

H1: For both of the performance metrics, SFT is better than PreT with 0.01 level of statistical significance.

H2: For both of the performance metrics, UFT is better than PreT with 0.01 level of statistical significance.

H1	H2	H3	H4	H5	H6	H7	H8	H9	H10	H11
AUROC										
**	**		**	**	**			**	**	**
Average Precision										
**	**		**	**	**		*	**	**	**

Table 5.6: Comprehensive result table of statistical significance test with p-values for all hypotheses as presented in Table 5.3. ** means that statistical significance level (p-value) is lower or equal to 0.01 and * means that p-value is lower or equal to 0.05. Corresponding values can also be looked up in tables 5.4 and 5.5.

H3: Even though the collective average scores of UFT is higher than SFT for both AUROC and Average Precision, the relationship was not proven to be statistical significant. Thus, we conclude that we fail to obtain statistically significant results that would suggest one is better than the other.

H4 - H6: We conclude that Gaussian Mixture Models performed better than other novelty detection algorithms such as One-class SVM, Isolation Forest and Bayesian Gaussian Mixture Models. We prove the relationship by a level of significance of 0.01 and by comparing the sample average values of scores for each of the pair distribution being compared in hypotheses H4-H6.

H7: No statistical evidence of strong relationship among the results obtained from A-fc7-fc7 and A-fc7-fc6.

H8: No statistical evidence of strong relationship among the results obtained from V-fc7-fc7 and V-fc7-fc6 in terms of AUROC score. However, V-fc7-fc6 is better than V-fc7-fc7 in terms of Average Precision with 0.05 level of significance.

H9: A-fc7-fc7 is better than Googlenet with 0.01 level of significance.

H10: Googlenet is better than V-fc7-fc7 with 0.01 level of significance.

H11: A-fc7-fc7 is better than V-fc7-fc7 with 0.01 level of significance.

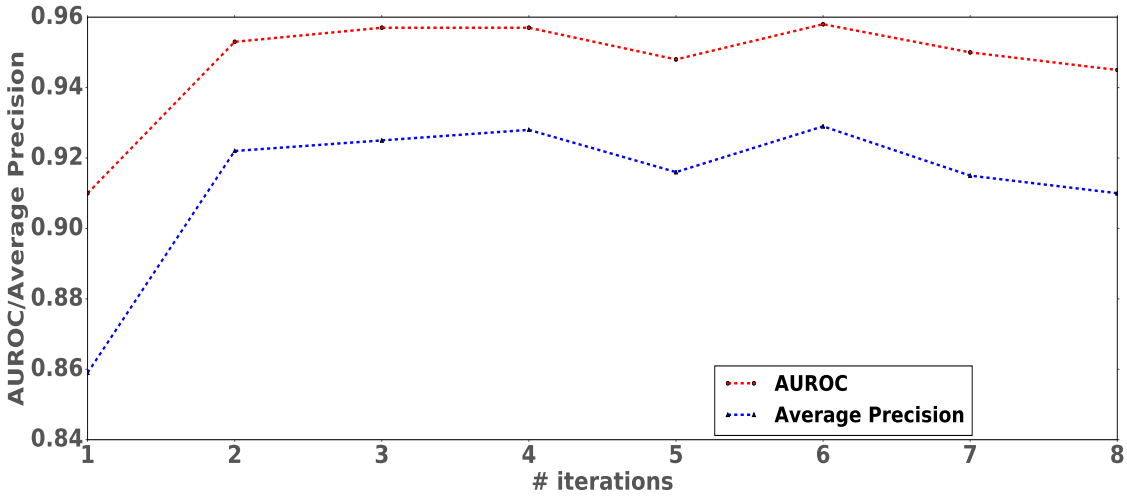
5.3.2 Iterative unsupervised fine-tuning.

In order to evaluate the stability of unsupervised fine-tuning with convolutional neural networks, we iteratively fine-tuned subsets of experiment instances. We basically repeat unsupervised fine tuning algorithm 3 in order to validate whether the subsequently trained model also produces stable results or not. In the first iteration, we take the training landscape images without labels, extract features using the pre-trained CNNs and train ensemble of Gaussian Mixture clustering algorithms to generate new labels for the training images. These new labels generated in the current iteration is then used to train CNNs in supervised manner in the next iteration.

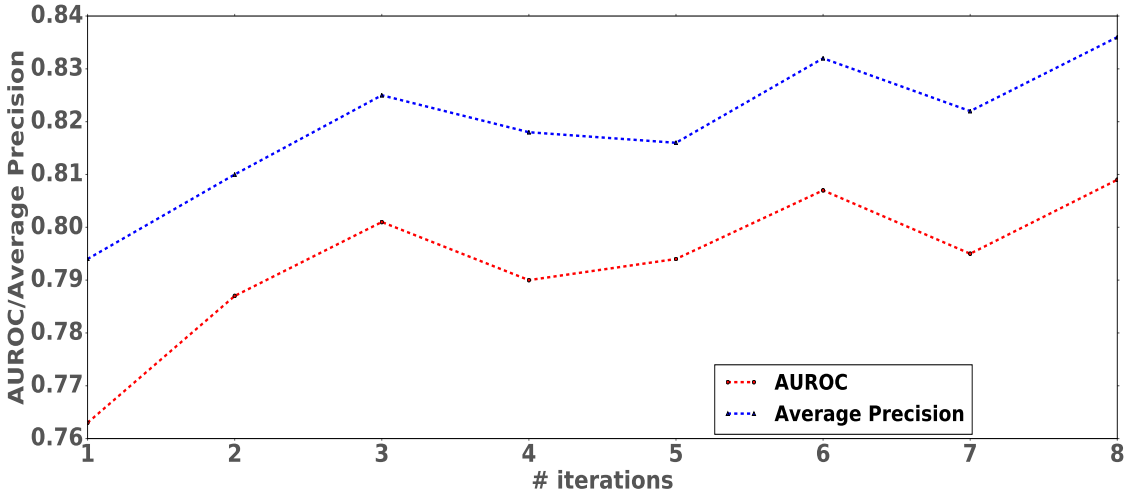
Figures 5.9a, 5.9b, 5.10a and 5.10b depict the stability of AUROC and average precision performance metrics values obtained from an iterative unsupervised fine tuning approach. We can observe that the AUROC and average precision values starts from lower values in the first iteration, increases in the second iteration and then maintain nearly-constant variation in the later iterations. The values in the first iterations are obtained from the features extracted from the pre-trained models (without fine tuning). Then the ensemble of clustering with Gaussian Mixture Models is then used to obtain the labels for fine-tuning the model in second iteration. The same process is repeated and the AUROC and average precision values are computed for numerous iterations.

Figure 5.9 presents the obtained AUROC and precision values for iterative unsupervised fine tuning for both of the datasets with two unknown types for eight iterations. Similarly, Figure 5.9 shows the result for both datasets with three un-

known types repeated for twenty two iterations. Both of the presented results were experimented for Googlenet CNN.

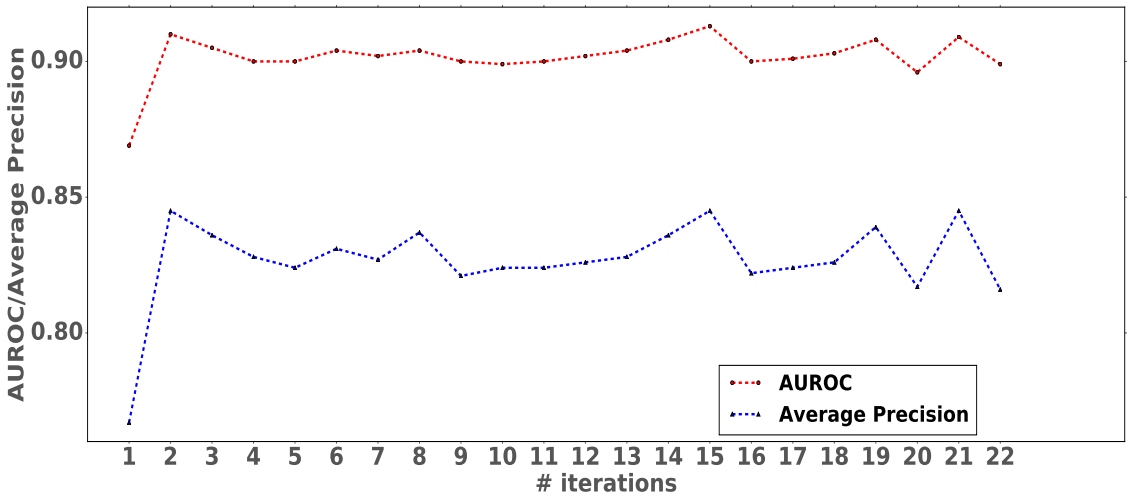


(a) HRS dataset with two unknown types

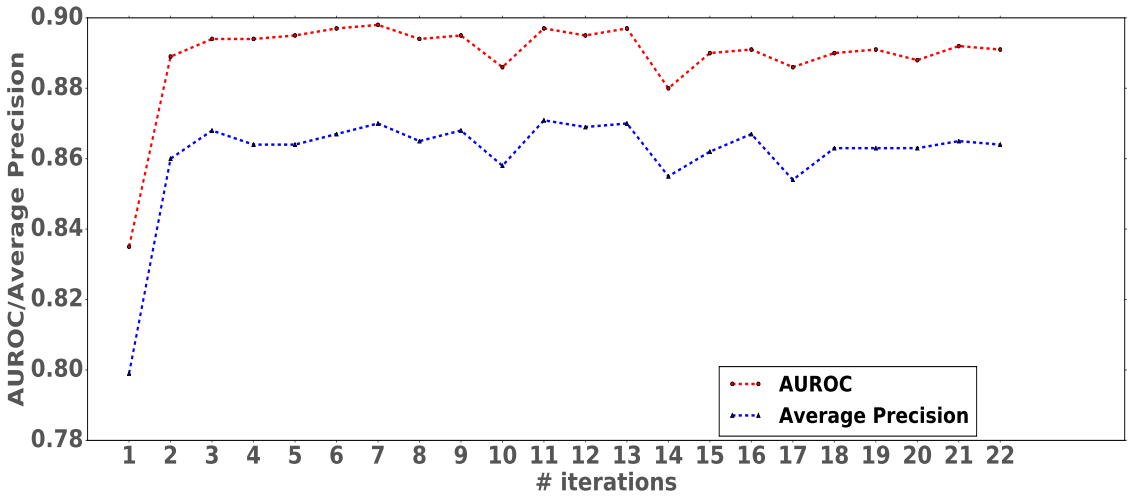


(b) UCM dataset with two unknown types

Figure 5.9: Iteratively computed AUROC and Average Precision scores for unsupervised fine tuning algorithm for (top) HRS dataset with two unknown types and (bottom) UCM dataset with two unknown types. CNN model: Googlenet.



(a) HRS dataset with three unknown types



(b) UCM dataset with three unknown types

Figure 5.10: Iteratively computed AUROC and Average Precision scores for unsupervised fine tuning algorithm for (top) HRS dataset with three unknown types and (bottom) UCM dataset with three unknown types. CNN model: Googlenet.

5.3.3 Autoencoder

We also conducted partial experiments by implementing an autoencoder neural network on top of features extracted from CNN models from the "CNN Feature Extrac-

tion for train and test images" block of figure 5.4. With autoencoder (see Chapter 2 for more details on how autoencoder works), we use reconstruction error as a metric to measure the novelty score of images in test set. However, the obtained results were very low and were not comparable at all to the results discussed in the previous sections with PreT, SFT and UFT novelty detection approaches.

CHAPTER 6

CONCLUSION

In this thesis, we defined the new problem of identifying unknown type of landscape by using convolutional neural networks in the context where large training datasets are not available. This problem has high impact in security and safety aspects in visual recognition. In this regard, we effectively provided a new paradigm for working on novelty detection in the supervised and unsupervised context using convolutional neural networks with transfer learning.

We proposed three novelty detection approaches using CNN transfer learning: baseline approach with pre-trained CNNs, supervised fine-tuning approach and non supervised fine-tuning approach. The baseline procedure requires the feature extraction from pre-trained CNNs and apply standard novelty detection algorithms. This procedure is a faster approach as we don't need to train any CNNs and provided fair results. With supervised fine-tuning procedure, we were able to obtain outstanding results: 0.96 of AUROC score and 0.94 of Average Precision score in the best cases. The primary contribution of our thesis is the unsupervised fine-tuning approach. Supervised fine-tuning approach require the knowledge of categories in in the known training data. Unsupervised-fine tuning approach was able to avoid this limitation while still obtaining the outstanding and comparable results. In the best case, we obtained 0.95 of AUROC score and 0.92 of Average Precision score in the unsupervised

fine tuning approach to novelty detection. Additionally, we perform a solid statistical analysis of the results that provides guideline and best practices related to our new problem.

Our problem of identifying unknown landscape images represents a strong challenge because it can be used for decision about safety and security. In these contexts, reducing the false alarms (false positive rates) and improvement of even 1% or 2% of our selected metrics can have greater significance towards better methodological improvement. Our work also shows that pre-trained convolutional neural networks contain precious information to solve the problem of novelty detection in images when training dataset are not so big. The real challenge consists in extrapolating this information especially in the case of unsupervised fine-tuning.

Future Work. Our work applied the CNN transfer learning methodologies in the context of unknown type identification of landscape. We showed that our unsupervised fine tuning approach performed comparable to the supervised fine-tuning approach (in terms of mean value of obtained scores but with no statistical evidence), and we believe that there is room for improvement in terms of both the performance scores and methodology. The result plots of iterative unsupervised fine-tuning novelty detection approach also showed that the computed AUROC and Average Precision scores after some iterations tend to reach the performance score of SFT approach in some cases (for eg. compare scores of HRS with two unknown types in Table 5.2 with the same scores in 4^{th} and 6^{th} iteration of figure 5.9a). We believe that such comparable and near-equal performance of UFT approach in comparison to SFT approach is very promising. We also suggest the implementation of our approach to other application domains such as medical applications, scanning images, etc. in order to tackle the novelty detection problem.

Similarly, in unsupervised fine-tuning approach, in our current work, we use the same CNN models for both feature extraction before applying Gaussian Mixture model to generate new cluster and fine-tuning. For example, if we use Alexnet to obtain the initial cluster labels for the training dataset, with our current implementation, we use those new labels to fine-tune only Alexnet, and not VGG19 and Googlenet. We strongly believe that hybrid implementation (feature extraction from one type of CNN and fine tuning with another) of unsupervised fine tuning approach might achieve even better performance results.

Another possibility for the extension of our work is to fine-tune ensemble of CNN models. Ensemble of CNN models have also achieved state-of-the-art accuracy scores in various image classification tasks. By combining different CNN networks such as Alexnet and VGG19, and training them efficiently might produce more representational CNN models which might in turn produce better feature representations of images that eventually improve the performance of standard novelty detection algorithms. Similarly, we partially experimented with autoencoder neural network using only the reconstruction error. We also look forward to using stronger denoising and variational autoencoders which are gaining success as adversarial neural networks recently. We believe that, with the swift improvements in terms of representational power of convolutional neural networks and other kinds of neural networks, we can try different architectures to improve the performance score of our novelty detection approach.

REFERENCES

[1] D. Agarwal. Detecting anomalies in cross-classified streams: a bayesian approach. *Knowledge and information systems*, 11(1):29–44, 2007.

[2] K. Bhaduri, K. Das, and P. Votava. Distributed anomaly detection using satellite data from multiple modalitie. In *CIDU*, pages 109–123, 2010.

[3] P. Bodesheim, A. Freytag, E. Rodner, and J. Denzler. Local novelty detection in multi-class recognition problems. In *Applications of Computer Vision (WACV), 2015 IEEE Winter Conference on*, pages 813–820. IEEE, 2015.

[4] R. J. Bolton, D. J. Hand, et al. Unsupervised profiling methods for fraud detection. *Credit Scoring and Credit Control VII*, pages 235–255, 2001.

[5] G. Boracchi, D. Carrera, and B. Wohlberg. Novelty detection in images by sparse representations. In *Intelligent Embedded Systems (IES), 2014 IEEE Symposium on*, pages 47–54. IEEE, 2014.

[6] L. Cavigelli, D. Bernath, M. Magno, and L. Benini. Computationally efficient target classification in multispectral image data with deep neural networks. In *SPIE Security+ Defence*, pages 99970L–99970L. International Society for Optics and Photonics, 2016.

[7] T. Chen, I. Goodfellow, and J. Shlens. Net2net: Accelerating learning via knowledge transfer. *arXiv preprint arXiv:1511.05641*, 2015.

[8] D. Dai and W. Yang. Satellite Image Classification via Two-Layer Sparse Coding With Biased Image Representation. In *IEEE Geoscience and Remote Sensing Letters*, volume 8, pages 173 – 176, 2011.

[9] I. Delibalta, K. Gokcesu, M. Simsek, L. Baruh, and S. S. Kozat. Online anomaly detection with nested trees. *IEEE Signal Processing Letters*, 23(12):1867–1871, 2016.

[10] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[11] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *Icml*, volume 32, pages 647–655, 2014.

[12] Z. Ferdousi and A. Maeda. Anomaly detection using unsupervised profiling method in time series data. In *ADBIS Research Communications*. Citeseer, 2006.

[13] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fernández, and E. Vázquez. Anomaly-based network intrusion detection: Techniques, systems and challenges. *computers & security*, 28(1):18–28, 2009.

[14] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448, 2015.

[15] M. Goldstein and S. Uchida. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PloS one*, 11(4):e0152173, 2016.

[16] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[17] F. E. Grubbs. Procedures for Detecting Outlying Observations in Samples. *Technometrics*, 11(1):1–21, 1969.

[18] J. Han. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[19] V. J. Hodge and J. Austin. A survey of outlier detection methodologies. *Artificial intelligence review*, 22(2):85–126, 2004.

[20] F. Hu, G.-S. Xia, J. Hu, and L. Zhang. Transferring deep convolutional neural networks for the scene classification of high-resolution remote sensing imagery. *Remote Sensing*, 7(11):14680–14707, November 2015.

[21] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[22] A. Karparthy. Connecting Images and Natural Language. `http://cs.stanford.edu/people/karpathy/main.pdf`, 2017. [Online; accessed 13-June-2017].

[23] A. Karparthy. CS231n Convolutional Neural Networks for Visual Recognition. `http://cs231n.github.io/convolutional-networks/`, 2017. [Online; accessed 13-June-2017].

[24] H. Kataoka, K. Iwata, and Y. Satoh. Feature evaluation of deep convolutional neural networks for object recognition and detection. *arXiv preprint arXiv:1509.07627*, 2015.

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

[26] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

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

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

[29] K. Lenc and A. Vedaldi. R-cnn minus r. *arXiv preprint arXiv:1506.06981*, 2015.

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

[31] M. Markou and S. Singh. Novelty detection: a reviewpart 1: statistical approaches. *Signal processing*, 83(12):2481–2497, 2003.

[32] M. Markou and S. Singh. Novelty detection: a reviewpart 2:: neural network based approaches. *Signal processing*, 83(12):2499–2521, 2003.

[33] S. Marsland. Novelty detection in learning systems. *Neural Comp. Surveys*, 2003.

[34] T. M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997.

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

[36] K. Nogueira, O. A. Penatti, and J. A. dos Santos. Towards better exploiting convolutional neural networks for remote sensing scene classification. *Pattern Recognition*, 61:539–556, 2017.

[37] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[38] A. Papazoglou and V. Ferrari. Fast object segmentation in unconstrained video. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1777–1784, 2013.

[39] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[40] M. Petrovskiy. Outlier detection algorithms in data mining systems. *Programming and Computer Software*, 29(4):228–237, 2003.

[41] C. Phua, V. Lee, K. Smith, and R. Gayler. A comprehensive survey of data mining-based fraud detection research. *arXiv preprint arXiv:1009.6119*, 2010.

[42] M. A. F. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko. Review: A review of novelty detection. *Signal Process.*, 99:215–249, June 2014.

[43] L. Portnoy, E. Eskin, and S. Stolfo. Intrusion detection with unlabeled data using clustering. In *In Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001*. Citeseer, 2001.

[44] 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*, pages 91–99, 2015.

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

[46] M. Sabokrou, M. Fayyaz, M. Fathy, et al. Fully convolutional neural network for fast anomaly detection in crowded scenes. *arXiv preprint arXiv:1609.00866*, 2016.

[47] M. Sakurada and T. Yairi. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, page 4. ACM, 2014.

[48] S. H. Sengamedu, S. Sanyal, and S. Satish. Detection of pornographic content in internet images. In *Proceedings of the 19th ACM International Conference on Multimedia*, MM '11, pages 1141–1144, 2011.

[49] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

[50] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 806–813, 2014.

[51] J. Sigholm and M. Raciti. Best-effort data leakage prevention in inter-organizational tactical manets. In *MILITARY COMMUNICATIONS CONFERENCE, 2012-MILCOM 2012*, pages 1–7. IEEE, 2012.

[52] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[53] P. Smagghe, J.-L. Buessler, and J.-P. Urban. Novelty detection in image recognition using irf neural networks properties. In *ESANN*, 2013.

[54] R. Smith, A. Bivens, M. Embrechts, C. Palagiri, and B. Szymanski. Clustering approaches for anomaly based intrusion detection. *Proceedings of intelligent engineering systems through artificial neural networks*, pages 579–584, 2002.

[55] A. Strehl and J. Ghosh. Cluster ensembles — a knowledge reuse framework for combining multiple partitions. *J. Mach. Learn. Res.*, 3:583–617, Mar. 2003.

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

[57] Z. Tang, D. Wang, and Z. Zhang. Recurrent neural network training with dark knowledge transfer. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 5900–5904. IEEE, 2016.

[58] J. P. Theiler and D. M. Cai. Resampling approach for anomaly detection in multispectral images. In *AeroSense 2003*, pages 230–240. International Society for Optics and Photonics, 2003.

[59] S. Thiprungsri and M. A. Vasarhelyi. Cluster analysis for anomaly detection in accounting data: An audit approach. 2011.

[60] L. Torrey and J. Shavlik. Transfer learning. *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*, 1:242, 2009.

[61] Y. Yang and S. Newsam. Bag-of-visual-words and spatial extensions for land-use classification. In *Proceedings of the 18th SIGSPATIAL International Conference*

on Advances in Geographic Information Systems, GIS '10, pages 270–279, New York, NY, USA, 2010. ACM.

[62] D.-Y. Yeung and Y. Ding. Host-based intrusion detection using dynamic and static behavioral models. *Pattern recognition*, 36(1):229–243, 2003.

[63] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[64] D. Zhang, O. Javed, and M. Shah. Video object segmentation through spatially accurate and temporally dense extraction of primary object regions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 628–635, 2013.

[65] W. Zhou, Z. Shao, and Q. Cheng. Deep feature representations for high-resolution remote sensing scene classification. In *Earth Observation and Remote Sensing Applications (EORSA), 2016 4th International Workshop on*, pages 338–342. IEEE, 2016.

[66] C. Zhu, H. Kitagawa, S. Papadimitriou, and C. Faloutsos. Outlier detection by example. *Journal of Intelligent Information Systems*, 36(2):217–247, 2011.