Applications of Deep Learning in Visual Recognition

by

Mirmohammad SAADATI

THESIS PRESENTED TO ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
IN PARTIAL FULFILLMENT FOR A MASTER'S DEGREE
WITH THESIS IN INFORMATION TECHNOLOGY ENGINEERING
M.A.Sc.

MONTREAL, MARCH 22, 2023

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

BOARD OF EXAMINERS

THIS THESIS HAS BEEN EVALUATED

BY THE FOLLOWING BOARD OF EXAMINERS

Mr. Patrick Cardinal, Thesis supervisor
Department of Software and Information Technology, École de technologie supérieure

Mr. Marco Pedersoli, Thesis Co-Supervisor
Department of Systems Engineering, École de technologie supérieure

Ms. Elsa Vasseur, Thesis Co-Supervisor
Department of Animal Science, McGill University

Ms. Sylvie Ratté, Chair, Board of Examiners
Department of Software and Information Technology, École de technologie supérieure

Mr. Matthew Toews, Member of the Jury
Department of Systems Engineering, École de technologie supérieure

THIS THESIS WAS PRESENTED AND DEFENDED

IN THE PRESENCE OF A BOARD OF EXAMINERS AND THE PUBLIC

ON MARCH 10, 2023

AT ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

ACKNOWLEDGEMENTS

First, I would like to acknowledge the work of P. Stecko (McGill University) for developing manual tracking (2.2.2) in the second chapter, A. Zambelis and V. Boyer (both from McGill University) for drafting the manuscript, and G.M. Dallago (McGill University) and V. Naghashi (UQAM) for further data analytics leading to the published version of the manuscript (Zambelis *et al.*, 2021).

Second, I am grateful for the support and guidance of my supervisors, Dr. Marco Pedersoli, Dr. Patrick Cardinal, and Dr. Elsa Vasseur. This work would have not been possible without their tremendous counsel and patience. They worked tirelessly to provide me with astounding opportunities to learn and improve. In the moments when I was stressed and disappointed with my performance, Dr. Pedersoli supported me profoundly and heartily and motivated me to pursue my work and enabled me to succeed.

I would also like to extend my gratitude to the rest of my thesis committee: Dr. Sylvie Ratté and Dr. Matthew Toews for their judicious comments and questions.

Also, I would like to thank my colleagues and friends at LIVIA who supported me and made me feel welcomed, specially Masih Aminbeidokhti who always helped me to learn and grow, Marie-Philippe Gill who set an example of hard-work and dedication, Sajjad Abdoli who always dispensed insightful advises, Behnaz Nasiri, and Saypraseuth Mounsaveng.

Applications de l'apprentissage en profondeur dans la reconnaissance visuelle

Mirmohammad SAADATI

RÉSUMÉ

La recherche sur le bien-être animal a soulevé des inquiétudes concernant l'intensification du logement des animaux de ferme des systèmes qui offrent des possibilités de mouvement limitées. Cependant, aucun système automatisé actuellement disponible le logiciel de suivi est capable de suivre efficacement et avec précision les mouvements des vaches laitières dans les stalles les systèmes de logement. L'application de modèles d'apprentissage en profondeur au suivi de localisation offre une opportunité pour une mesure précise et opportune du mouvement des vaches dans l'environnement du logement. La L'objectif de cette étude était de développer et de valider un outil de géolocalisation pour surveiller la mouvement des vaches laitières dans leurs stabulations entravées à l'aide d'une approche d'apprentissage en profondeur. Vingt-quatre en lactation Les vaches Holstein ont été enregistrées sur vidéo pendant une période continue de 24 heures les semaines 1, 2, 3, 6, 8 et 10. Des images individuelles montrant la position en stalle de chaque vache ont été extraites de chaque 24-h enregistrement à raison d'une image par minute. Trois coordonnées sur chaque vache ont été manuellement annoté sur les séquences d'images pour suivre l'emplacement de la hanche gauche, de la hanche droite et du cou. L'ensemble de données final utilisé pour valider l'approche d'apprentissage en profondeur consistait en 199 100 Images rouge-vert-bleu avec annotations manuelles des coordonnées. Le jeu de données a été séparé en ensembles de formation et de validation. Des variantes des modèles d'apprentissage en profondeur suivants ont été testées : VGG Net, Resnet, GoogLeNet et DenseNet. Les performances du modèle ont été exprimées en termes de pixels erreur pour chaque coordonnée annotée à partir de l'ensemble d'images de validation. L'erreur de pixel a été convertie en mesure standard en cm en utilisant le rapport pix/cm moyen pour chaque vache chaque semaine. ResNet18 avec des étiquettes augmentées ont nettement surpassé tous les autres modèles testés. Pour la validation ensemble d'images, l'erreur moyenne des 3 coordonnées équivalait à une erreur de 0,74 cm dans la valeur réelle placement physique des coordonnées dans l'environnement de décrochage. Sur la base de ce haut degré de précision, le modèle peut être utilisé pour analyser les modèles d'activité de vaches individuelles pour optimisation des espaces de stalle et amélioration de la facilité de circulation.

L'imagerie radar à synthèse d'ouverture (SAR) capture les propriétés physiques de la Terre en transmettre des signaux micro-ondes à sa surface et analyser le signal rétrodiffusé. Cela fait ne dépend pas de la lumière du soleil et peut donc être obtenu dans n'importe quelle condition, comme la nuit et temps nuageux. Cependant, les images SAR sont plus bruyantes que les images lumineuses et jusqu'à présent, il n'est pas clair que niveau de performance qu'un système de reconnaissance moderne pourrait atteindre. Ce travail présente une analyse des performances des modèles d'apprentissage en profondeur pour la tâche de segmentation des terres à l'aide de SAR images. Nous présentons des résultats de segmentation sur la tâche de classer quatre catégories de terres différentes (urbain, eau, végétation et ferme) sur six sites canadiens (Montréal, Ottawa, Québec, Saskatoon, Toronto et Vancouver), avec trois modèles de segmentation d'apprentissage

en profondeur à la fine pointe de la technologie. Résultats montrent que lorsque suffisamment de données et de variété sur l'apparence du terrain sont disponibles, l'apprentissage en profondeur Les modèles peuvent atteindre d'excellentes performances malgré le bruit d'entrée élevé.

Mots-clés: Automatisation, Vache laitière, Pistage, Comportement animal, Imagerie RSO RADARSAT-2, Réseau de neurones à convolution profonde, Classification de l'occupation du sol, Segmentation sémantique

Applications of Deep Learning in Visual Recognition

Mirmohammad SAADATI

ABSTRACT

Animal welfare research has raised concerns regarding the intensification of farm animal housing systems that offer limited opportunity for movement. However, no currently available automated tracking software is able to efficiently and accurately track dairy cow movement across stall-based housing systems. Applying deep learning models to location tracking provides an opportunity for accurate and timely measurement of cow movement within the housing environment. The objective of this study was to develop and validate a location tracking tool to monitor the movement of dairy cows in their tie-stalls using a deep learning approach. Twenty-four lactating Holstein cows were video recorded for a continuous 24-h period on weeks 1, 2, 3, 6, 8, and 10. Individual images showing the in-stall position of each cow were extracted from each 24-h recording at a rate of one image per minute. Three coordinates on each cow were manually annotated on the image sequences to track the location of the left hip, the right hip, and the neck. The final dataset used to validate the deep learning approach consisted of 199,100 Red-Green-Blue images with manual coordinate annotations. The dataset was separated into training and validation sets. Variants of the following deep learning models were tested: VGG Net, Resnet, GoogLeNet, and DenseNet. Model performance was expressed in terms of pixel error for each coordinate annotated from the validation image set. Pixel error was converted to a standard measure in cm using the average pix/cm ratio for each cow in each week. ResNet18 with augmented labels significantly outperformed all other models tested. For the validation image set, the average error from all 3 coordinates was equivalent to a 0.74 cm error in actual physical placement of the coordinates within the stall environment. Based on this high degree of accuracy, the model may be used to analyze the activity patterns of individual cows for optimization of stall spaces and improved ease of movement.

Synthetic Aperture Radar (SAR) imagery captures the physical properties of the Earth by transmitting microwave signals to its surface and analyzing the backscattered signal. It does not depends on sunlight and therefore can be obtained in any condition, such as nighttime and cloudy weather. However, SAR images are noisier than light images and so far it is not clear the level of performance that a modern recognition system could achieve. This work presents an analysis of the performance of deep learning models for the task of land segmentation using SAR images. We present segmentation results on the task of classifying four different land categories (urban, water, vegetation and farm) on six Canadian sites (Montreal, Ottawa, Quebec, Saskatoon, Toronto and Vancouver), with three state-of-the-art deep learning segmentation models. Results show that when enough data and variety on the land appearance are available, deep learning models can achieve an excellent performance despite the high input noise.

Keywords: Automation, Dairy cow, Tracking, Animal behavior, RADARSAT-2 SAR imagery, Deep convolutional neural network, Land cover classification, Semantic segmentation

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

CNN	Convolutional Neural Network
RCNN	Recurrent Convolutional Neural Network
DNN	Deep Neural Network
MSE	Mean Squared Error
ReLU	Rectified Linear Unit
SGD	Stochastic Gradient Descent
LRN	Local Response Normalization
SAR	Synthetic Aperture Radar
ETS	École de Technologie Supérieure
SPP	Spatial Pyramid Pooling

INTRODUCTION

Motivation

In recent years, deep learning in visual recognition has advanced exceedingly fast thanks to widely accessible data and faster GPUs. Training very deep convolutional neural networks (CNN) efficiently requires not only large-scale datasets but also substantial GPU time and memory. Many of the successful and commonly used CNN models are originally developed from and evaluated on standard benchmark datasets (e.g., ImageNet) to allow for practical and accurate comparison between model variations or against the state-of-the-art (SOTA). On the downside, this approach does not guarantee similar levels of performance on any data. In reality, data preparation contributes greatly to the time required for running machine learning end-to-end. Newer applications have shorter access to data, while deep learning requires even more data to compensate for automatic feature engineering (Roh, Heo & Whang, 2021). In practice, techniques like Transfer Learning and/or Domain Adaptation can mitigate scarce data (AlShehhi, Damiani & Wang, 2021). Although it is not feasible to evaluate a model on every available data, exploring the applications of said models in real-world data and evaluating their efficiency on non-standard datasets strikes as a compelling research area to explore.

Finding the right data is a challenging task. Data collection can often be efficient and atomized, specifically the type of data that is available on the internet (on demand). Raw data is useful for unsupervised learning (e.g. clustering, density estimation, etc.). However, for supervised learning, we need labeled data and acquiring ground-truth requires human experts and can be both expensive and noisy. While companies that generate revenue from learning models can afford annotating large datasets (mostly thanks to outsourcing), in academia we adopt a different perspective towards selecting the right data. We are motivated by the nature of the problem, how its solution can have positive impacts, and the potential of future work. In this work we are motivated by two different data and according tasks, 1) optical images of tie-stall cattle, 2) synthetic aperture radar (SAR) images of Canada's earth surface.

We evaluate the performance of deep convolutional neural networks on two different private datasets. The first part of this work aims to solve a localization task given a dataset of cow images annotated at image level with three coordinates of the hips and the back of the neck. The second part of this work aims to solve a segmentation task given a dataset of SAR images annotated at pixel level with the land cover classes. Here we introduce each work separately.

Problem Statement

In this work, we evaluate the accuracy of deep convolutional neural networks on real-world datasets. To start, we solve an object localization problem using regression. Then, we address a segmentation problem. Both of these tasks are a subset of visual recognition. In both of these tasks we aim to develop deep convolutional models that automatically discover regularities in image data. We can describe the visual recognition system by breaking down the process into three steps, namely, pre-processing, feature extraction, and prediction. In pre-processing, we perform operations to transform the raw input data (images) into normalized (size and value) samples ready for feature extraction. We also remove invalid or unnecessary information in pre-processing to avoid confusing the model. Extra operations such as data augmentation is performed at this step to improve the generalization of the system. In the next step, features are extracted from samples that are later used to make a decision by the model. These features are extracted automatically at different levels by convolutional layers. At first low level features such as edges are extracted and then high level features such as shapes or complex geometric patterns. At the last step, extracted features are mapped into the output space. In the task of object localization, the output space is a vector of six real numbers where each pair delimits a point of interest on the Cartesian plane. In the task of segmentation, the output space is a matrix of integers with the same width and height dimensions of the input sample where each value classifies the corresponding pixel (vector of 3 RGB values) in the input sample.

Figure 0.1 An example of the cattle localization problem. Three points of interest are localized by deep learning model. Red dots show the ground-truth location of the hips and the neck of the cow while blue dots are model predictions

Animal welfare research has raised concerns regarding the intensification of farm animal housing systems that offer limited opportunity for movement. However, no currently available automated tracking software is able to efficiently and accurately track dairy cow movement across stall-based housing systems. Applying deep learning models to location tracking provides an opportunity for accurate and timely measurement of cow movement within the housing environment. The objective of this study was to develop and validate a location tracking tool to monitor the movement of dairy cows in their tie-stalls using a deep learning approach. Figure 0.1 illustrates an example of the outcome of automatic localization compared to the ground-truth. Twenty-four lactating Holstein cows were video recorded for a continuous 24-h period on weeks 1, 2, 3, 6, 8, and 10. Individual images showing the in-stall position of each cow were extracted from each 24-h recording at a rate of one image per minute. Three coordinates on each cow were manually annotated on the image sequences to track the location of the left hip, the right hip, and the neck. The final dataset used to validate the deep learning approach consisted of 199,100 Red-Green-Blue images with manual coordinate annotations. The dataset was separated into

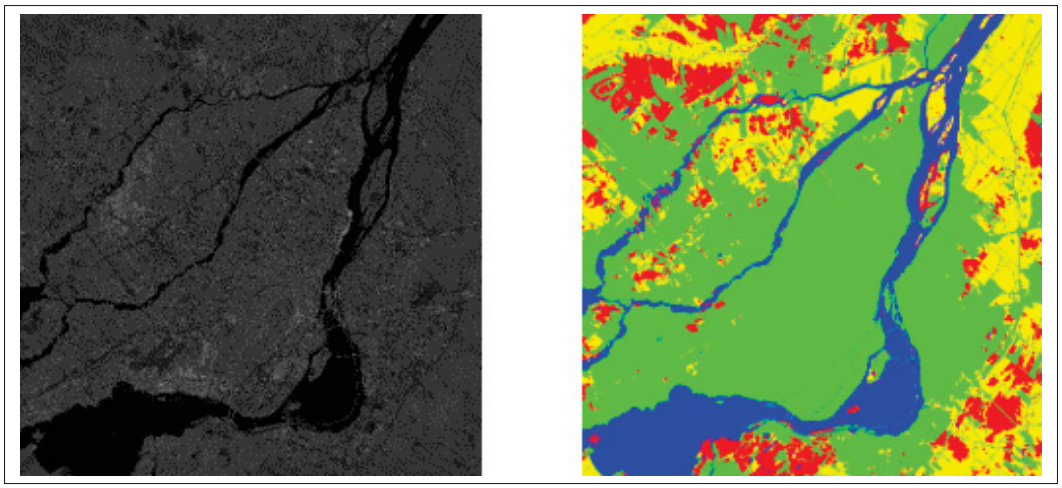

Figure 0.2 An example of the land cover segmentation problem. The SAR image is shown on the left and its corresponding semantic segmentation annotation on the right

training and validation sets. Variants of the following deep learning models were tested: VGG Net, Resnet, GoogLeNet, and DenseNet. Model performance was expressed in terms of pixel error for each coordinate annotated from the validation image set. Pixel error was converted to a standard measure in cm using the average pix/cm ratio for each cow in each week. ResNet18 with augmented labels significantly outperformed all other models tested. For the validation image set, the average error from all 3 coordinates was equivalent to a 0.74 cm error in actual physical placement of the coordinates within the stall environment. Based on this high degree of accuracy, the model may be used to analyze the activity patterns of individual cows for optimization of stall spaces and improved ease of movement.

Synthetic Aperture Radar (SAR) imagery captures the physical properties of the Earth by transmitting microwave signals to its surface and analyzing the backscattered signal. It does not depends on sunlight and therefore can be obtained in any condition, such as nighttime and cloudy weather. However, SAR images are noisier than light images and so far it is not clear the level of performance that a modern recognition system could achieve. This work presents an analysis of the performance of deep learning models for the task of land segmentation using SAR images.

Figure 0.2 illustrates the land cover segmentation problem. We present segmentation results on the task of classifying four different land categories (urban, water, vegetation and farm) on six Canadian sites (Montreal, Ottawa, Quebec, Saskatoon, Toronto and Vancouver), with three state-of-the-art deep learning segmentation models. Results show that when enough data and variety on the land appearance are available, deep learning models can achieve an excellent performance despite the high input noise.

In this thesis, we contributed by:

1. Developing an automatic video-based location tracking tool for dairy cattle in their tie stalls with high accuracy. This tool is useful for measuring the efficacy of different interventions aiming at improving comfort and welfare of dairy cows.
2. Experimenting and analyzing the results of high performance deep segmentation models trained on challenging SAR imagery data. This report is useful for comparing the efficacy of different deep learning models in land cover segmentation task using non-optical images.

The rest of this manuscript is organized as follows: Chapter 1 introduces the related background on deep learning and it's application in visual recognition tasks followed by a summary of modern convolutional neural network (CNN) architectures and previous work in animal localication and land cover segmentation. In chapter 2, we develop an automated video-based location tracking framework for dairy cows in their tie stalls. In chapter 3, we present an study on deep segmentation models and their application in land cover segmentation with SAR imagery data. Finally, in chapter 4, we conclude the manuscript and discuss the future work.

CHAPTER 1

BACKGROUND

1.1 Machine Learning

Machine learning is a subset of Artificial Intelligence, an approach to develop programs that can learn from experience (Domingos, 2012). Most real life experiences can be stored as data in a computer system (e.g., image, audio, video, graph). Machine learning algorithms extract meaningful features from training data and learn to identify and reason about unseen test data. Formally, "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." (Mitchell, 1997) The ability to learn from data enables us to develop algorithms that are not pragmatically complex to efficiently solve problems that are difficult for classical algorithms. The breakthroughs in learning algorithms are expedited by the availability of increasingly large labeled data and the leap of computational power thanks to advances in GPUs and TPUs manufacturing process.

1.1.1 Types of learning methods

Machine learning algorithms are classically categorized into three types, namely, supervised learning, unsupervised learning, and reinforcement learning. More recently, popular methods include semi-supervised learning, self-supervised learning, transfer learning, and active learning.

Supervised (predictive) learning is the most widely used form of machine learning in practice (Murphy, 2012). The goal of supervised learning is to learn to predict an associated value or vector y from an input vector x, where input is labeled by an instructor or human expert. Supervised algorithms observe features from each labeled example of a dataset and learn a mapping function $y = F(x)$ (estimating $p(y|x)$) given a set of pairs $D = \{(x_i, y_i)\}$ for $i = 1, \ldots, N$ where D is called training set and N is the number of training examples (Murphy, 2012). Input data x can simply be a D-dimensional vector of numbers or any complex object

(e.g., an image, a set of words, a graph). Depending on the problem being solved, we need to incorporate a way to represent the input data. For example, to predict a benign or malignant tumor, interesting variables include (not limited to) the tumor size, total tumor area, the tumor circularity, irregularity, and rectangularity (Zacharaki *et al.*, 2009). On the other hand, the output data can also take any form of representation depending on the application. Evidently, the output value y_i is usually a nominal variable from a finite set, $y_i \in \{1, \ldots, C\}$ (such as benign or malignant) for classification tasks or a real-valued number $y_i \in \mathbb{R}$ (such as the location of a cow in her stall) for regression tasks. The estimator F is classically modeled by algorithms like Linear/Logistic Regression, Decision Tree, Support Vector Machine (SVM), or Nearest Neighbours. However, here we are interested in models based on Neural Networks where algorithms learn by adjusting their inter connection weight combinations with the help of error values (Sathya & Abraham, 2013). In neural networks, learning is achieved by back propagating an error value through the network and calculating the local gradients in the following two passes:

- **Forward Pass**; where the input vector is passed forward neuron by neuron through the network and emerges at the output layer of the network as output vector y. This output is then compared with desired (ground truth) output and results in an error or loss value. This comparison is done by some loss function f_{loss}. The synaptic weights of the network are not changed during this pass (Sathya & Abraham, 2013).
- **Backward Pass**; where the loss value calculated at forward pass is propagated backward through the network. The local gradient for each neuron in each layer is calculated and is used to update the synaptic weights of the network according to the learning rate. Multi-layer NNs are optimized iteratively using back propagation as a way to implement *gradient descent* (Ibrahim, Chen & Lipsitz, 1999).

When the Supervised learning has many applications in real life. Visual recognition tasks like: self-driving cars, face detection, image classification, cancer prediction, emotion detection, garbage sorting, etc. are examples where supervised learning is widely used. There are also

other examples that supervised learning can be helpful like: spam filtering, price prediction, content/product recommendation (e.g., Netflix movie suggestion, Facebook relevant ads).

The second major approach to machine learning is unsupervised (descriptive) learning. In this approach, algorithm is provided with unlabeled inputs and the goal is to find interesting patterns in dataset $D = \{x_i\}$ for $i = 1, \ldots, N$. This is also known as knowledge discovery (Murphy, 2012). Formally, unsupervised learning involves observing several examples of a random vector x, and trying to learn the probability distribution $p(x)$ that can be interpreted as pattern or structure. However, the boundary between supervised and unsupervised learning is not ultimately clear. Many techniques used in machine learning can address both of these tasks (Goodfellow, Bengio & Courville, 2016).

Behaviour detection, data clustering, anomaly/fraud detection, targeted marketing, and image segmentation are examples of applications that can benefit from unsupervised learning. It is also common to use ensemble models with a combination of supervised and unsupervised learning to solve problems.

As a combination of supervised and unsupervised learning, semi-supervised learning is used when only a subset of training data has labels. The goal of semi-supervised learning is to improve the performance of a supervised model by utilizing both labeled and unlabeled data, where the unlabeled portion of training data is larger than the labeled part due to the cost of labeling process. Another related technique is to use weak annotations, known as weakly-supervised learning, where labels contain noise (incomplete, inexact, inaccurate (Zhou, 2017)). A combination of weakly and semi supervised learning is also applicable, e.g., (Yan, Liang, Pan, Li & Zhang, 2017). In transfer learning, the goal is to train the model on a different dataset for some source task and transfer that knowledge by adapting and applying the model on the target task. Learning types and tasks are summarized in Figure 1.1

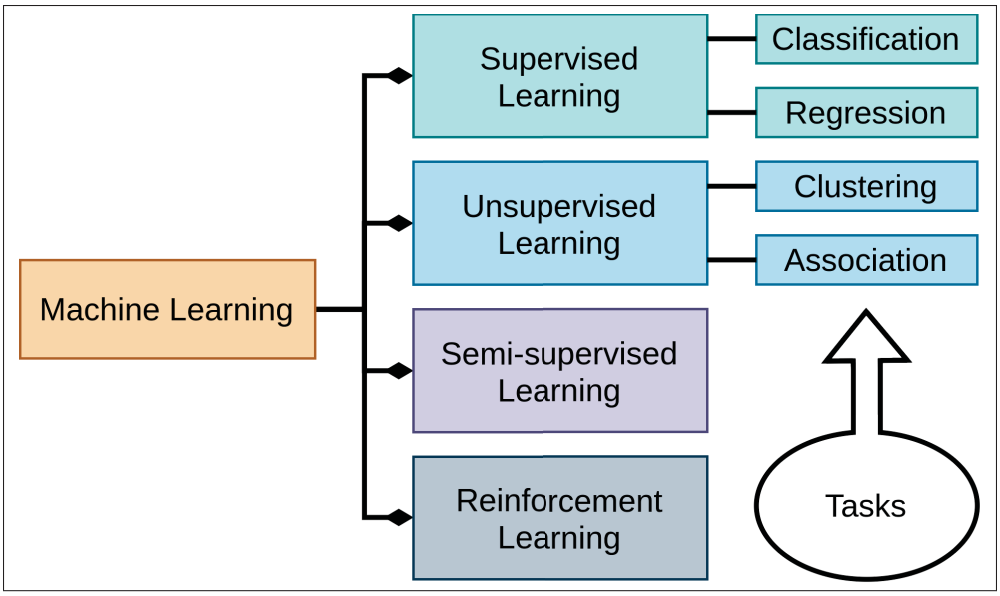

Figure 1.1 Types of machine learning and their tasks

1.1.2 Machine learning tasks

Machine learning enables us to solve tasks that are too difficult to solve with traditional – fixed – programs. In machine learning, the process of learning itself is not the *task*. Learning allows us to attain the intelligence needed to perform the task. We usually describe a task as how the algorithm should process an example (Goodfellow *et al.*, 2016). An example is a collection of features that are measured from an event or object that we are interested in. We represent each example as a vector $X \in \mathbb{R}^n$ where each entry x_i of the vector is a specific feature. Here we discuss three important tasks in machine learning.

- **Classification:** The goal of classification is to learn a mapping from inputs x to output y, where $y \in \{1, \ldots, C\}$. C is the number of classes. We refer to this problem as binary classification if $C = 2$ ($y \in \{0, 1\}$). If $C > 2$, the problem is referred to as multi-class classification. Sometimes, the class labels are not mutually exclusive (an object may be classified as expensive and heavy), in this case the problem is multi-label classification (Murphy, 2012). In this type of task, the computer is asked to specify which of C classes some input belongs to (Goodfellow *et al.*, 2016). There are other variants of classification problems where the algorithm learns a probability distribution over classes. Object recognition

problems are classification tasks that are best accomplished with deep learning (Goodfellow *et al.*, 2016).

- **Regression:** The goal of regression is to learn a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ from inputs x to some numerical value y. For example, the output value y may represent the estimated price of a house given some if its features like surface area, year built, neighbourhood, etc., in input vector x. Regression is similar to classification, except that the format of output is different (Goodfellow *et al.*, 2016).
- **Density estimation:** The goal of density estimation is to learn a probability density/mass function $p(X)$ (depending on whether the input is continuous or discrete) on the space that examples were drawn from (Goodfellow *et al.*, 2016). Here the algorithm learns the structure of the input data.

Figure 1.2 illustrates the difference between classification and regression using a linear example. Both classification and regression are general tasks that can be tailored to solve specific problems. For example, in visual recognition, classification can be applied both at image level (to predict a class label for an entire image or a crop of it) or pixel level (to predict a class label for each pixel). Combination of classification and regression is also applicable, for example, to locate and classify one or multiple objects within an image.

1.1.3 Challenges in machine learning

One of the most important challenges in machine learning is the problem of overfitting. An overfitted model performs very well on the training set, however, it flounders to achieve similar performance on the test set. Such models procure inadequate generalization, i.e., they lack the ability to perform well on previously unseen inputs (Goodfellow *et al.*, 2016). Modeling every minor variation in the training set is more rampant when fitting highly flexible models (e.g., deep neural networks). This results in an overfitted model that mimics the noise in the training data rather than learning the true trends (Murphy, 2012). Another factor that contributes to overfitting is the size (number of examples) of the training set. Statistically, an overfitted model contains more parameters than can be justified by the data. There are a few techniques to

overcome overfitting.

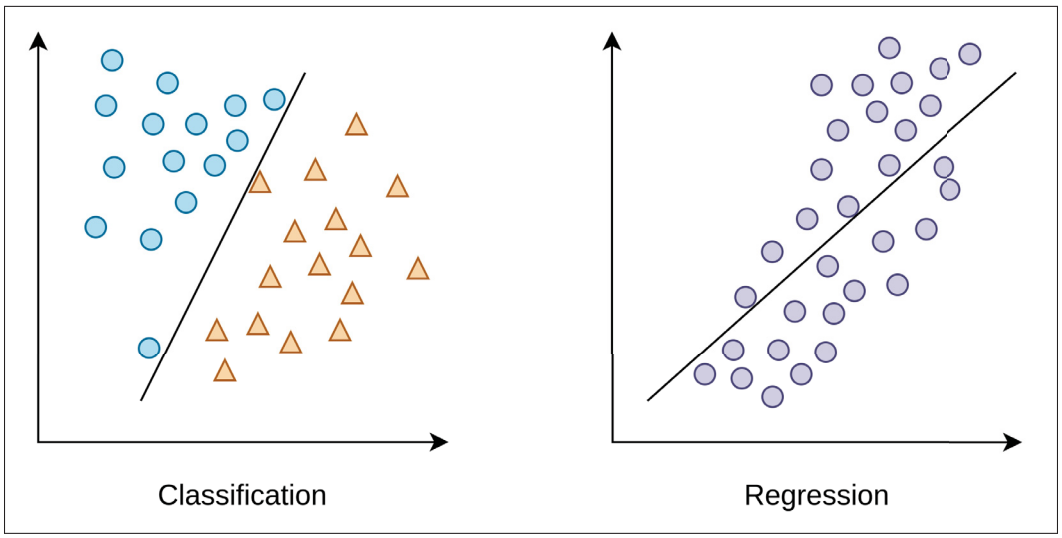

Figure 1.2 Difference between linear classification and linear regression

- **Regularization** is a technique which makes slight modifications to the learning algorithm such that the model generalizes better. Regularization penalizes the coefficients, such that no parameter in the model will get a relatively high value. There are different regularization techniques such as L1 and L2 regularization, and dropout (Srivastava, Hinton, Krizhevsky & Salakhutdinov, 2014).
- **Cross-validation:** In K Fold cross validation, the data is divided into k subsets. Now one of the k subsets is used as the validation set and the other k-1 subsets are put together to form a training set. This process is repeated k times. The error estimation is averaged over all k trials to get total effectiveness of our model. This significantly reduces bias as we are using most of the data for fitting, and also significantly reduces variance as most of the data is also being used in validation set.
- **Early stopping:** Early stopping is a form of regularization used to avoid overfitting when training a learner with an iterative method, such as gradient descent.

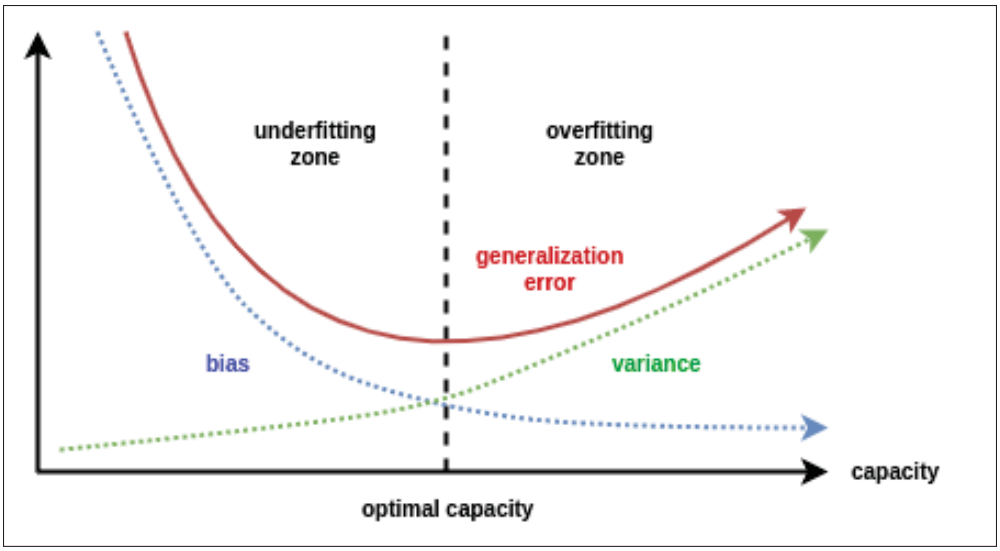

Figure 1.3 Early stopping

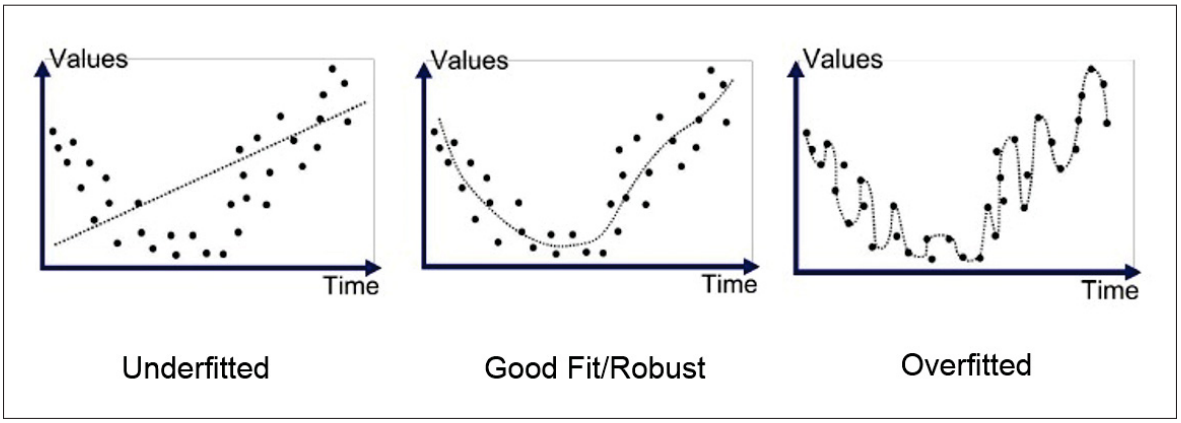

Figure 1.4 Overfitting

1.2 Image Recognition

Computer vision is the study of the most important functions of our visual system, e.g., extracting useful information from images (like identifying objects and people in context), and reconstructing the three-dimensional structure of an environment (Buhmann, Malik & Perona, 1999). The aim is to design models that learn to reproduce these functions. Vision is hard. Two images of the same object can be drastically different depending on the lighting, viewpoint, camera resolution, etc. Moreover, different recognition tasks require extracting information

at different levels of spatial and logical resolution. For example, in image classification, a single label describes an entire image. Where, in object segmentation, all pixels are labeled individually.

Image recognition is an application of **machine learning**. Recently, with the deep learning breakthrough (Tweedale, 2019) and larger yet standard datasets, a promising avenue has been paved in the most difficult recognition tasks on challenging datasets. Analyzing a scene and recognizing all of the constituent objects is the most challenging visual task for computers (Szeliski, 2010). The most challenging version of recognition is general category – class – recognition, that involves recognizing varied classes such as animals or furniture. In many instances, recognition depends heavily on the context of surrounding objects and scene elements. Now, we talk about image recognition tasks.

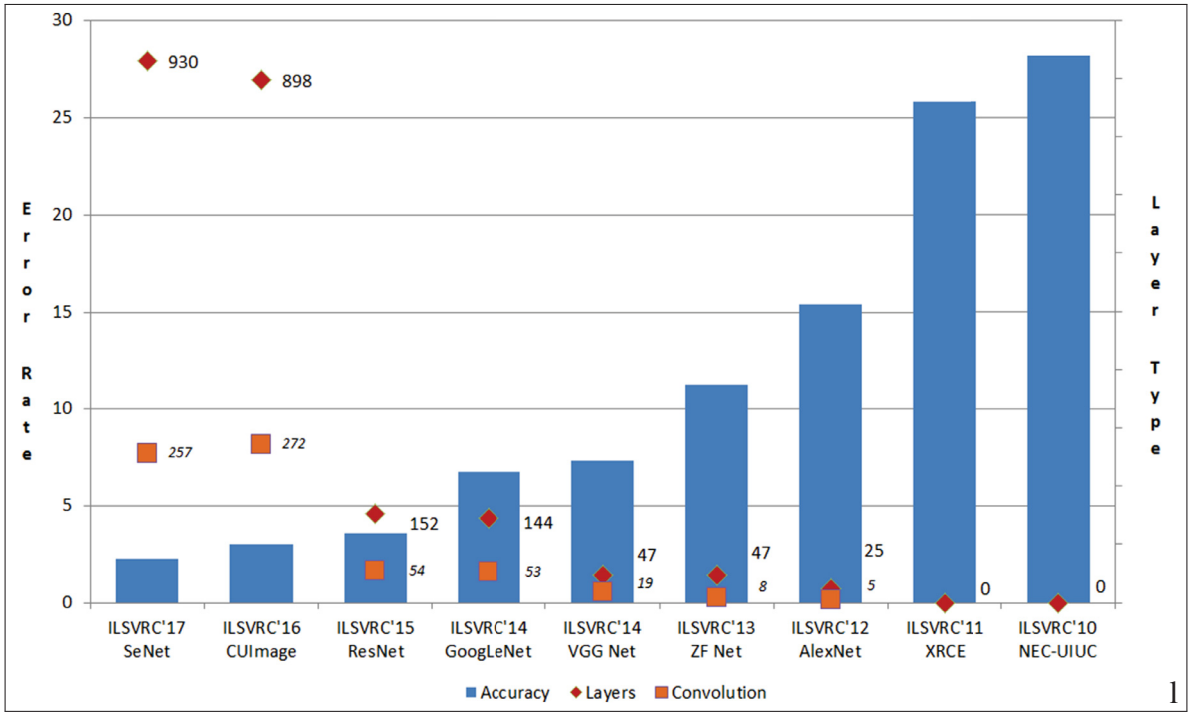

Figure 1.5 Deep learning breakthrough
Taken from Tweedale (2019)

1.2.1 Classification

As a fundamental task, image classification (also called category or class recognition) attempts to identify the category (class) to which a whole image belongs. Image classification is relatively a simple recognition task, since (typically) only one object appears in images. For example, in the problem of classifying images of single digits to number symbols, the classifier learns a mapping from the input image represented by a 2-D vector X to output label y where $y \in \{0, ..., 9\}$ (LeCun & Cortes, 2010). In order to classify a set of images into different categories, the relationship between the data and the classes must be well understood. Classification techniques were originally developed out of research in Pattern Recognition.

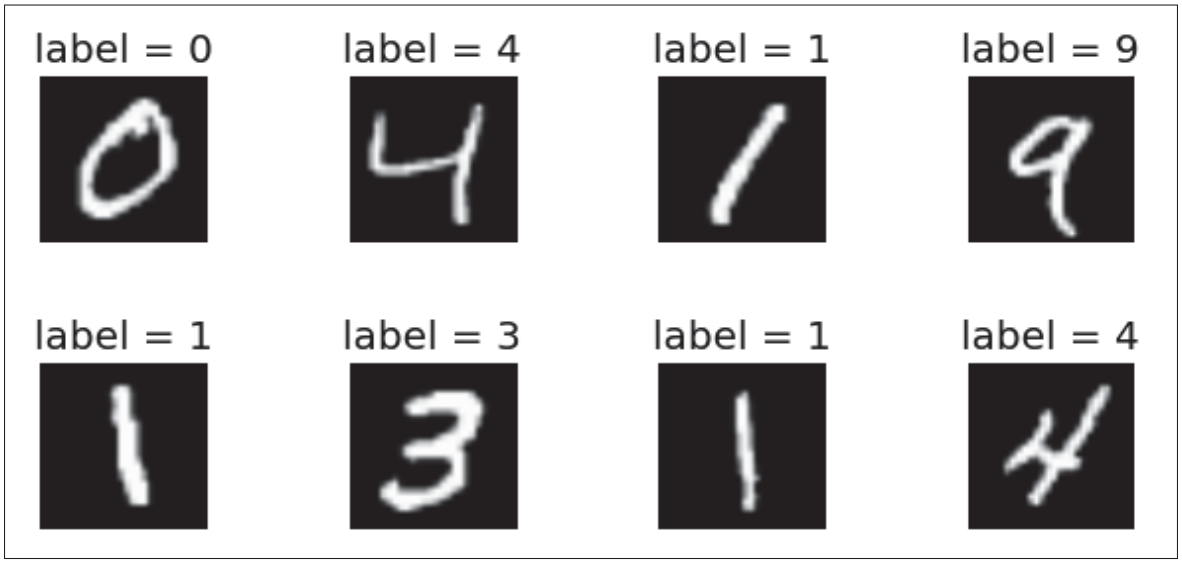

Figure 1.6 MNIST dataset

1.2.2 Detection

Object detection is the combination of two tasks; classification and localization. Detection is a challenging recognition task since multiple objects can appear in one image. In localization, a bounding box is drawn around each object of interest in the image. Then, a class label is assigned to each bounding box. Using brute force to apply an algorithm to every possible sub-window

in an image is slow and prone to error. In traditional methods, special purpose detectors are constructed to find regions where objects are more likely to to be.

Face detection is an example of early successful applications of object detection (Szeliski, 2010). We can also track objects in videos by applying object detection methods to a sequence of frames per second. For example, tracking the movement of cars and pedestrians in a street.

Recently, deep learning has dominated detection algorithms (Zou, Shi, Guo & Ye, 2019); (Liu *et al.*, 2018). Deep detection algorithms are designed to focus on achieving high accuracy or high efficiency. A highly accurate detection algorithm is robust to both interclass and intraclass variations. There are two categories of deep detection frameworks.

- **Two stage (region based) detection frameworks**: separating the preprocessing step for generating object proposals, e.g., R-CNN (Girshick, Donahue, Darrell & Malik, 2014), Faster R-CNN (Ren, He, Girshick & Sun, 2015), and Mask R-CNN (He, Gkioxari, Dollár & Girshick, 2017)
- **One stage (unified) detection framworks**: no region proposal generation, trained end-to-end, e.g., YOLO (Redmon, Divvala, Girshick & Farhadi, 2016), and YOLOX (Ge, Liu, Wang, Li & Sun, 2021)

Recent studies shows that with a weak supervision approach, the necessity of a well labeled datasets can be significantly reduced (Bilen & Vedaldi, 2015).

1.2.3 Segmentation

Image segmentation is a detection task where localization is performed at pixel level rather than placing a coarse bounding box around the objects. Segmentation helps to identify the shapes of different objects in images since the models predict a granular pixel-wise mask. In the simplest formulation – **semantic segmentation** – the problem is to classify each pixel with a class (semantic) label where objects are grouped based on a set of categories (e.g., human, car, tree, sky) as a single entity and different instances of the same object are not distinguished. In a more refined setup – **instance segmentation** – multiple objects of the same category

are delineated as distinct individual instances and are labeled uniquely. Figure 1.7 compares semantic segmentation with instance segmentation in the context of classification and detection using an example of cats and dogs. Some real-world applications of semantic segmentation are self-driving cars (identifying pedestrians, vehicles, road lanes, etc.), medical imaging (outlining tumors), satellite imagery (detecting forests, bodies of water, roads, farms, urban communities, etc.), and augmented reality (Minaee *et al.*, 2020). Deep convolutional models (e.g., DeconvNet (Noh, Hong & Han, 2015); SegNet (Badrinarayanan, Kendall & Cipolla, 2015); PSPNet (Zhao, Shi, Qi, Wang & Jia, 2016); DeepLabv3 (Chen, Papandreou, Schroff & Adam, 2017); DANet (Fu, Liu, Tian, Fang & Lu, 2018)) have made a paradigm change in image segmentation by achieving remarkable performance improvements over some advanced (yet classical) algorithms (e.g., graph cuts (Boykov, Veksler & Zabih, 2001), conditional and Markov random fields (Plath, Toussaint & Nakajima, 2009), and sparsity-based (Minaee & Wang, 2019)). The main challenge with segmentation relates to the high cost of generating accurate and consistent pixel-level annotations for large datasets (Rieder & Verbeet, 2019).

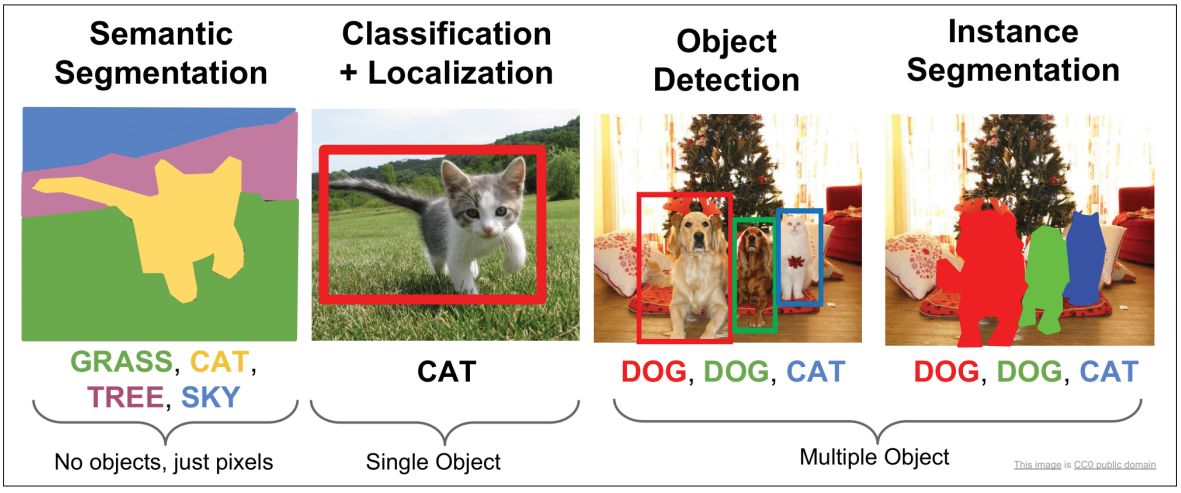

Figure 1.7 Classification VS Detection VS Segmentation (Semantic vs Instance)
Taken from Rieder & Verbeet (2019)

1.2.4 Benchmark datasets

Deep learning thrives on large training sets. The availability of impeccable training datasets has capacitated many of the major breakthroughs in AI (Wissner-Gross, 2016). However, producing high quality labeled datasets for supervised learning is expensive and difficult. Labeling thousands of data samples by expert annotators is a resource (time, cost, etc.) intensive process, exacerbated by human errors. In visual recognition, the cost of labeling an image for segmentation tasks is significantly higher compared to classification and detection. For example, the pricing of Google Cloud (https://cloud.google.com/ai-platform/data-labeling/pricing) for labeling 1,000 images is 35 USD for classification, 63 USD for bounding box, and 870 USD for segmentation. There are several public standard datasets that researchers in the field of visual recognition use to evaluate their algorithms and compare them to other models. Here we provide a list benchmark datasets and briefly explain the data and the respective recognition task.

- **MNIST** dataset (LeCun & Cortes, 2010) of handwritten digits (0 to 9) is a subset of NIST database. The digits are size-normalized and centered in square gray-scale images with a size of 28×28. MNIST contains a training set of 60,000 samples and a test set of 10,000 samples divided equally per 10 categories for classification.
- **CIFAR** dataset (Krizhevsky, 2009) of different Wordnet nouns (e.g., airplane, bird, dog, ship, etc.) has two variants – CIFAR-10 and CIFAR-100 – both of which contain 50,000 training samples and 10,000 test samples of square RGB images with a size of 32×32 divided equally per 10 and 100 categories in CIFAR-10 and CIFAR-100 respectively for classification. This leaves only 500 training samples for each category in CIFAR-100, making it a more challenging dataset compared to CIFAR-10. However, both of CIFAR datasets are already solved efficiently. For example, in the state-of-the-art, EffNet-L2 (Foret, Kleiner, Mobahi & Neyshabur, 2020) achieves 99.7 and 96.08 percent test accuracy on CIFAR-10 and CIFAR-100 respectively.
- **ImageNet** dataset (Russakovsky *et al.*, 2015) contains images that are organized according to the Wordnet hierarchy. Wordnet contains approximately 100,000 phrases and ImageNet has provided around 1000 images on average to illustrate each phrase.

1.3 Convolutional Neural Networks

As a specialized type of neural networks, convolutional neural networks (CNN) have been remarkably successful in practical applications (Krizhevsky, Sutskever & Hinton, 2012; He, Zhang, Ren & Sun, 2015). Convnetes are most suitable for processing data structures that have at least one dimension with a grid-like topology, e.g., time-series data (weather records, economic indicators, etc.), where samples are taken at defined intervals over a period of time constructing a 1D grid (vector) of real values, and image data (visual recognition) as a 2D grid (matrix) of pixels, where each pixel is a real value in Grayscale images, or a vector of three real values in RGB images (RGB images can also be expressed with integer values ranging from 0 to 255). The idea of convnets is to replace the linear transformation (matrix multiplication) in vanilla neural networks with a convolution operation (Goodfellow *et al.*, 2016).

We start introducing the convolution operation for 1D signals and then extend its definition in 2D spaces. The general form of convolution is defined as an operation on two functions of a real-valued argument. However, this definition is not practical when working with data on a computer. Therefore, we discretize this definition by assuming an input function $x(t)$ and a kernel (filter) function w. The convolution of x and w, denoted with an asterisk $x * w$, and referred to as the feature map (pre activation), is given by:

$$s(t) = (x * w)(t) = \sum_{\tau} x(\tau) w(t - \tau) \tag{1.1}$$

The definition above implies a flip-and-filter operation, where entries of $s(t)$ are calculated by flipping the kernel w, shifting it, and taking the dot product with the input x. Since convolution is commutative, it can be equivalently defined as:

$$s(t) = (w * x)(t) = \sum_{\tau} x(t - \tau) w(\tau) \tag{1.2}$$

The latter definition suggests a translate-and-scale operation, where multiple copies of the input x are translated and scaled by the kernel w (Grosse, 2018). The convolution operation can be

similarly defined over more than one axis, e.g., using 2D image data X and a 2D kernel W, we have:

$$S(i,j) = (X * W)(i,j) = \sum_{pq} X(p,q)W(i-p, j-q) \tag{1.3}$$

$$= (W * X)(i,j) = \sum_{pq} X(i-p, j-q)W(p,q) \tag{1.4}$$

Flipping of the kernel in convolution results in a nice mathematical property, however, the commutative property is not important in the implementation of convnets because the kernels are learned during the training process. Therefore, convnets are implemented using the cross-correlation function, i.e., convolution without flipping the kernel (Goodfellow *et al.*, 2016), given by:

$$S(i,j) = (X * W)(i,j) = \sum_{pq} X(i+p, j+q)W(p,q) \tag{1.5}$$

Convnets take advantage from the following two important assumptions.

- **Sparse connectivity** aims to limit the number of interactions between each pair of input and output units by defining the kernel smaller than the input. This leads to fewer stored parameters and consequently reduces required memory, improves statistical efficiency, and allows the network to efficiently describe complex interactions by constructing smaller building blocks for which their output values depend on a small portion of the input values. We illustrate the sparse connectivity in Figure 1.8.
- **Parameter sharing** aims to limit the number of parameters (weights) by sharing them between multiple functions in a model. In convnets this means that "rather than learning a separate set of parameters for every location of the input, we learn only one set" because feature detectors learned at one part of the data is probably useful in other parts.

In CNNs, the two assumptions above makes it possible to work with variable size inputs. A typical convolutional network layer consists of three sub-layers. The first sub-layer performs several convolutions in parallel to produce a set of linear activations. In next the sub-layer,

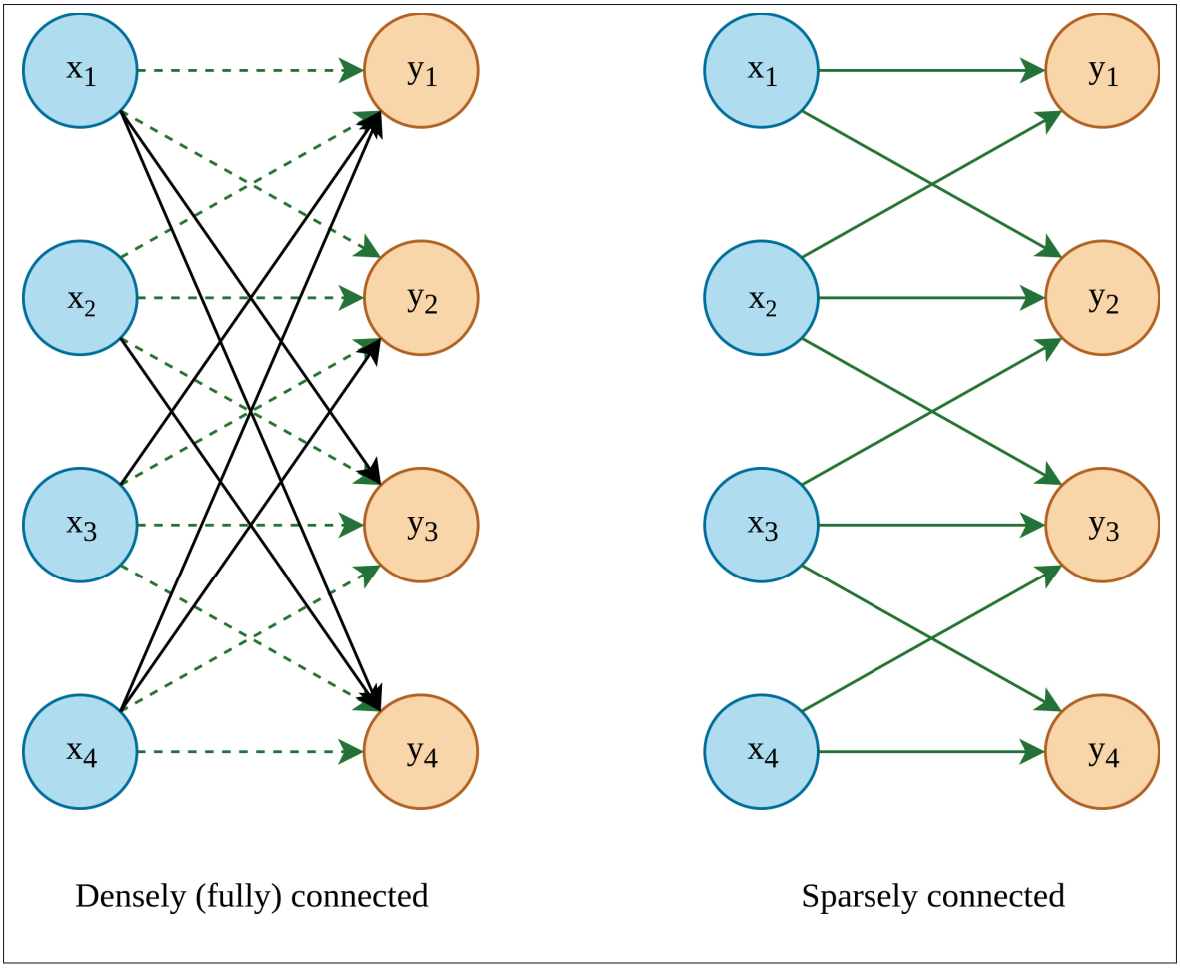

Figure 1.8 Sparse connectivity: The number of connections reduces significantly in sparsely connected layers compared to fully connected layers. This results in a smaller receptive field meaning that each output depends on fewer inputs

each linear activation in passed through a nonlinear activation function. At the last sub-layer, a pooling function modify the output of the layer. Pooling helps to make the representation become approximately invariant to small translations of the input (Goodfellow *et al.*, 2016). There are two types of popular pooling functions: **max pooling** and **average pooling**. Max pooling simply reduce a set of numbers defined by a windows by replacing them with the maximum value present at that window. Average pooling does the same using the average value of numbers present at each window.

Finally, after several convolutional and pooling layers, the final step is to generate the output. This is done by using a **fully connected** layer at the very end of the network. Neurons in a fully connected layer have connections to all activations in the previous layer, like regular neural networks. Figure 1.9 illustrates the architecture of a typical convolutional neural network.

There are many popular CNN architectures mainly designed for image classification and object localization problems. These models show state-of-the-art performance on benchmark datasets such as CIFAR-10, CIFAR-100, ImageNet and PASCAL VOC. Here we will discuss some of the most successful ones.

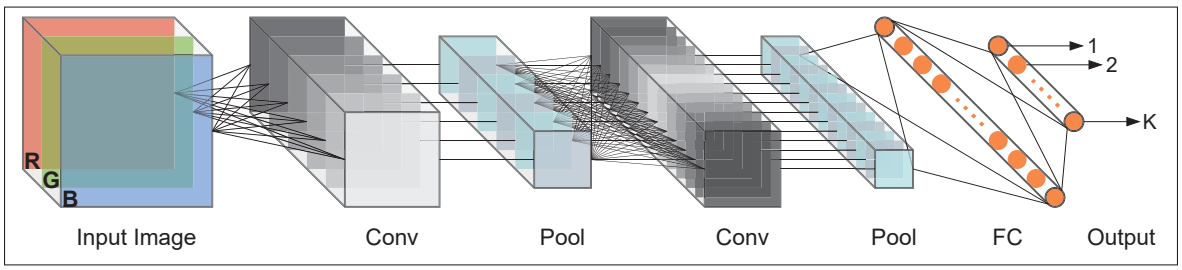

Figure 1.9 Typical CNN Architecture

1.3.1 AlexNet

Inspired by the architecture of LeNet (Lecun, Bottou, Bengio & Haffner, 1998), AlexNet (Krizhevsky *et al.*, 2012) is the first deep convnet trained efficiently on large subsets of ImageNet used in ILSVRC-2010 and ILSVRC-2012. These subsets contain around 1.2 million training images with 1000 different categories. Using label-preserving transformations to augment small datasets makes it possible to solve simple recognition tasks (like MNIST digit recognition) efficiently. However, in a more realistic setting, objects of the same class can have substantial spatial variability in different images. This calls for larger datasets and models with greater learning capacity. These models benefit from a strong architectural prior granted by convnets to make relevant assumptions about the nature of image data, and to adjust the learning capacity by tuning the depth and breadth of convolutional filters. Although training convnets is easier than fully connected neural networks of the same size due to sparse connectivity and parameter

sharing, applying deeper models in large scale to high-resolution images can get computationally expensive. Luckily, the continuous growth in GPUs processing power and memory makes it possible to train significantly deep models on large datasets in practical time by using an highly optimized implementation of 2D convolution.

The AlexNet model contains eight weighted layers of which the first five are convolutional and the remaining three are fully connected. Figure 1.10 summarizes the overall architecture of AlexNet. Every weighted layer is followed by the ReLU (Nair & Hinton, 2010) non-linearity. It is shown that using a non-saturating activation function (such as $f(x) = max(0, x)$) accelerates the training process several times compared to saturating coequals (such as $f(x) = tanh(x)$) when optimizing deep convnets with gradient descent (Krizhevsky *et al.*, 2012). To improve generalization in AlexNet, the first two convolutional layers are followed by a local response normalization scheme that entices local weights at different channels to compete for greater values by dividing the pre-activations by the sum of few adjacent kernel maps squared. The last note in the architecture of AlexNet is the employment of overlapping max pooling that is reported to slightly improve the results compared to the equivalent non-overlapping scheme. Similar to other classification tasks, AlexNet is trained by optimizing the log loss, i.e., maximizing the multinomial logistic regression objective (also known as cross entropy) after the network predictions are normalized into a probability distribution using softmax function $\sigma(x_i) = e^{x_i} / \sum_i e^{x_i}$.

In the original architecture of AlexNet, kernels are cut across and stored in two GPUs in a way that limits the communication necessary between the memories of each GPU. This technique was employed since each GPU had a limited memory of 3GB at that time; therefore, we do not provide any further details on how the splitting was performed exactly. AlexNet is very susceptible to overfitting since it has more than 60 million parameters where most of them belong to the last three fully connected layers. To reduce overfitting, in addition to data augmentation, dropout technique is applied in the last two fully connected layers to reduce complex co-adaptations between the weights of adjacent layers.

AlexNet achieved top-1 and top-5 test errors of **37.5%** and **17.0%** on ILSVRC-2010 and top-5 test error of **15.3%** with a pre-trained model on ILSVRC-2012.

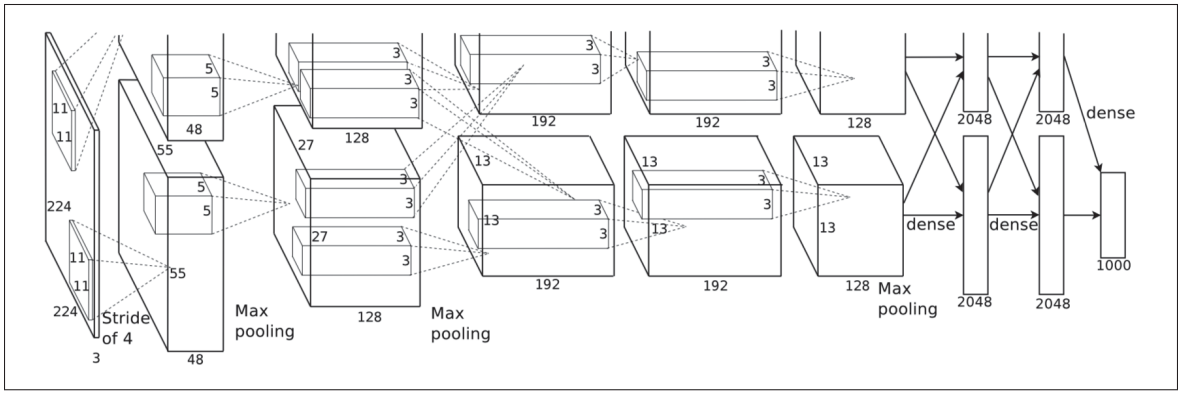

Figure 1.10 AlexNet Architecture
Taken from Krizhevsky *et al.* (2012)

1.3.2 VGG networks

Motivated by large public image repositories and high performance GPUs, VGG (Simonyan & Zisserman, 2015) builds upon the idea of smaller kernel size and stride in very first convolutional layers (Zeiler & Fergus, 2014; Sermanet *et al.*, 2014) and contributes by designing deeper and more accurate convnets. Training networks with increased depth is viable due to very small (3×3) kernels that are still capable of capturing the notion of *left*, *right*, *up*, *down*, and *center* efficiently. VGG achieved first and second place in ILSVRC-2014 localization and classification tasks respectively.

Five unique network architectures are introduced by VGG (excluding one architecture that performs local response normalization (LRN) after the first convolutional layer, a technique inherited from AlexNet). These five architectures share the same design principles in layer configurations except for the depth. VGG is trained using RGB input images with fixed spatial dimensions of 224×224, pre-processed by subtracting the mean RGB value from each pixel. The architecture is designed by stacking five groups of 3×3 convolutional filters (excluding one architecture that also contains 1×1 kernels) where each group is followed by a 2×2 max

pooling with a stride of 2. The architecture is concluded by three fully connected (FC) layers, where two 4096 channels layers are followed by a 1000 channels layer outputting classification scores. Naturally, all weighted layers are followed by ReLU non-linearity (Nair & Hinton, 2010). The stride and padding of convolutions are fixed to 1, preserving spatial resolution. Figure 1.11 shows the increasing depth in different configurations of VGG. In its deepest configuration, with 144 million parameters, VGG-19 only has 8% more than 133 million parameters of VGG-11, granted that the extra layers are convolutional.

As the best performing configuration, VGG-19 achieved top-1 and top-5 test errors of **25.5%** and **8.0%** on ILSVRC-2014 using a single test scale setup. The top-1 and top-5 errors are further decreased to **24.4%** and **7.1%** using a combination of multiple crops and dense evaluations (Simonyan & Zisserman, 2015).

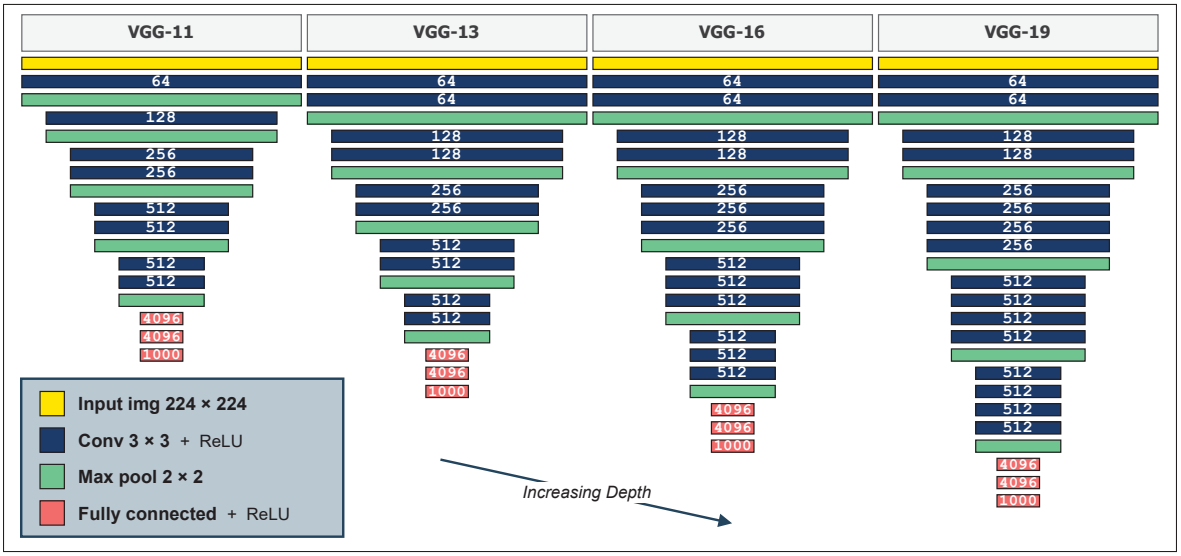

Figure 1.11 VGG Networks. More than 80% of parameters are in fully connected layers. VGG-11 with LRN and VGG-16 with Conv 1×1 are omitted

1.3.3 Inception

Inception is the Google's codename for the computational budget aware deep architecture that achieved 1st place in ILSVRC-2014 classification task with a particular "incarnation" called GoogLeNet (Szegedy *et al.*, 2015a).

With the careful design of repeating components called *inception module*, this architecture tackles the problem of computational expenses and overfitting while benefit from increasing depth and width of the network. Inception is motivated not only by the increased processing power and larger datasets; but mainly from improved algorithms and architectures that were designed while capping computations at "1.5 billion multiply-adds". Inception is inspired by **Network in Network** to enhance local modeling within the receptive fields, in practice however, to increase local discriminability, coequal 1×1 convolutions are used in place of MLPs originally proposed by (Lin, Chen & Yan, 2014).

While increasing the size (depth, width) of convnets can improve their performance, it comes with two major drawbacks: 1) overfitting, due to the increased number of parameters, and 2) increased use of computation and memory. Overfitting can be moderated by larger training set and regularization, however, limiting computation and memory usage can only be achieved by adopting sparsely connected architectures inside the convolutions. This is inherently different from convolution's parameter sharing. Here, the idea is to build non-uniform sparse models motivated by (Arora, Bhaskara, Ge & Ma, 2013) that constructs the optimal network topology, layer by layer, from a large, very sparse deep neural network by clustering neurons from the last layer that present highly correlated activations.

1.3.4 Residual networks

Deep convolutional neural networks achieved remarkable results in Visual Recognition tasks. Deep networks naturally integrate from low-level details like colors and edges, to more abstract concepts like eyes and lips. As the importance of network depth revealed itself in modern CNN architectures (Simonyan & Zisserman, 2015; Zilly, Srivastava, Koutník & Schmidhuber, 2017),

residual networks (He *et al.*, 2015) were introduced. CNN models can benefit from increased depth, however, training deep CNNs is challenging due to vanishing/exploding gradients and degradation (saturating accuracy). While the former is mitigated by normalized initialization ($\text{Bias}_{ij} = 0, \text{Weight}_{ij} = \text{Uniform}[-\frac{\sqrt{6}}{\sqrt{n_j+n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j+n_{j+1}}}]$ where n is the number of weights in layer j (Glorot & Bengio, 2010)) and batch normalization (Ioffe & Szegedy, 2015), the latter is exposed in deeper models indicating the levels of optimization difficulty. Experiments show that training deeper models produce higher training error compared to their shallower counterparts. However, by adding *identity* mapping layers to a trained model, one can construct a deeper solution with the very same training error, yet SGD cannot find such solutions.

Deep residual learning framework proposes a network architecture to address degradation problem by letting each few stacked layers to fit a residual mapping. Let's assume that the desired mapping is $H(X)$, instead of finding this mapping, the algorithm tries to find residual mapping $F(X) = H(X) - X$. The author hypothesize that it is *easier to optimize a residual mapping than to optimize the unreferenced mapping*. This formulation implies shortcut connections in the network architecture. We may let shortcut connections simply perform identity mapping. These deep residual networks are **easy to optimize** and benefits easily from the **accuracy gains** with increased depth. There are five different ResNet architectures that are mainly different in their depth. The first convolution layer of these networks consists of a 7×7 filter with a stride of 2 followed by a max pooling of 3×3 with a stride of 2. All of the ResNet architectures benefit from an average pooling and a single fully-connected layer followed by softmax. Figure 1.12 illustrates the architecture of ResNet-34 comparing it with VGG19. Table 1.1 compares the accuracy of AlexNet, VGG, and Resnet on ImageNet dataset.

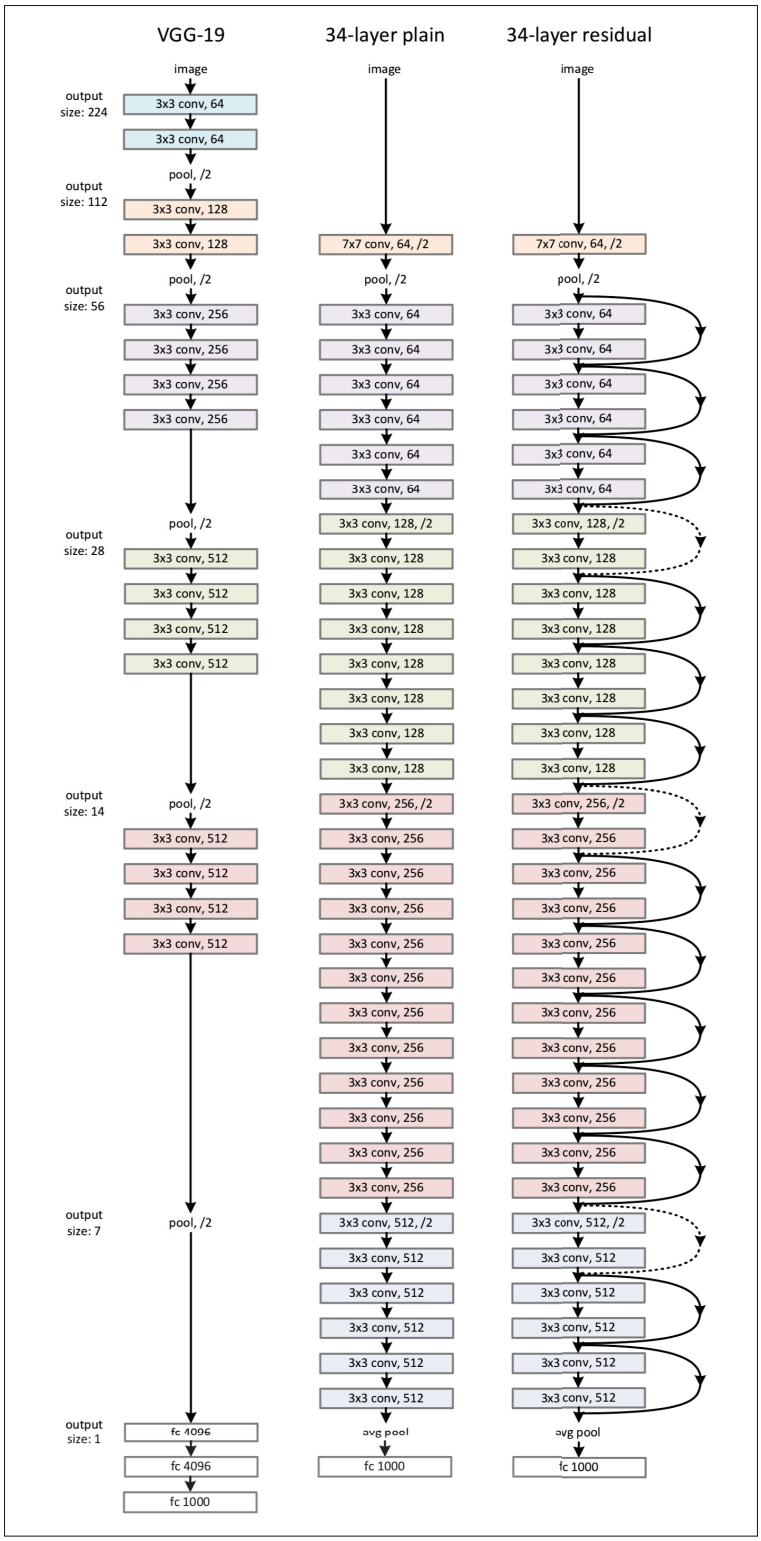

Figure 1.12 ResNet Architecture
Taken from He *et al.* (2015)

Table 1.1 AlexNet vs VGG vs ResNet

Model	Top-1 error	Top-5 Error
AlexNet	36.7	15.4
VGG (v5)	24.4	7.1
ResNet-34	21.53	5.60
ResNet-50	20.74	5.25
ResNet-101	19.87	4.60
ResNet-152	**19.38**	**4.49**

1.4 Related Work

In this section, we review the literature of deep learning applications in localization and land cover segmentation separately.

1.4.1 Localization

In localization, learning algorithms predict a set of real numbers. Deep CNN models can be trained using multiple loss functions that measure how far the predictions are from the ground truth. The most basic form of localization loss calculates the mean absolute error (MAE) between predictions and target. MAE is also known as *L1 loss* i.e. L1 norm $||W||_1$ of vector $W = Y - \hat{Y}$ where $Y/\hat{Y}$ are prediction/target vectors. In a batch training setup, a reduction function (like sum or mean) is needed to calculate a scalar loss for the entire batch. Assuming both Y and $\hat{Y}$ vectors contain n elements, the MAE is calculated as in Equation 1.6 with sum reduction for a batch size of N.

$$\text{MAE} = \sum^{N} \frac{1}{n} \sum_{i}^{n} |Y_i - \hat{Y}_i| \tag{1.6}$$

The next common localization loss calculates the mean squared error (MSE) between predictions and target. MSE is also known as L2 loss i.e. squared L2 norm $||W||_2^2$ of vector $W = Y - \hat{Y}$ as shown in Equation 1.7 with mean reduction.

$$\text{MSE} = \frac{1}{N}\sum^{N}\frac{1}{n}\sum_{i}^{n}(Y_i - \hat{Y}_i)^2 \tag{1.7}$$

MSE penalizes extreme errors and makes small (< 1) errors less effective. It can be observed that MSE exhibits quadratic growth while MSE increases proportionally, therefore it is more sensitive to outliers. To combines the desirable properties of both the absolute and the quadratic loss, Huber loss (Mangasarian & Musicant, 2000) introduces a conditional criterion that calculates the MSE when the absolute element-wise error falls bellow δ and a δ scaled MAE term otherwise. Equation 1.8 defines the Huber loss with a $\delta = 1$, also known as as the smooth L1 loss.

$$\text{smooth}_{L1} = \begin{cases} 0.5(Y - \hat{Y})^2, & \text{if } |Y - \hat{Y}| < 1 \\ |Y - \hat{Y}| - 0.5, & \text{otherwise} \end{cases} \tag{1.8}$$

Human Pose Estimation

Human pose estimation (HPE) is an example of a localization problem where the classical computer vision methods were found to be outperformed by deep learning solutions that have been rapidly developed in recent years. HPE is the problem of building a representation of human body by locating the body parts from images or videos. We can solve this problem for 2D and 3D pose annotations. Describing such representation requires human body modeling. Three main models of human body modeling are: kinematic model (2D and 3D), planar model (2D), volumetric model (3D). N-joints rigid kinematic model is most commonly used in HPE methods where a human body is defined as an entity with joints and limbs, containing body kinematic structure and shape information (Zheng *et al.*, 2020). Figure 1.13 illustrates three main approaches to human body modeling.

Although HPE is not directly related to our work in this thesis, learning representations of pose annotations modeled by joints and limbs; whether for human or cattle, resembles an analogous problem with challenges similar to those of HPE. Therefore, the previous work in HPE is motivating for Animal Localization.

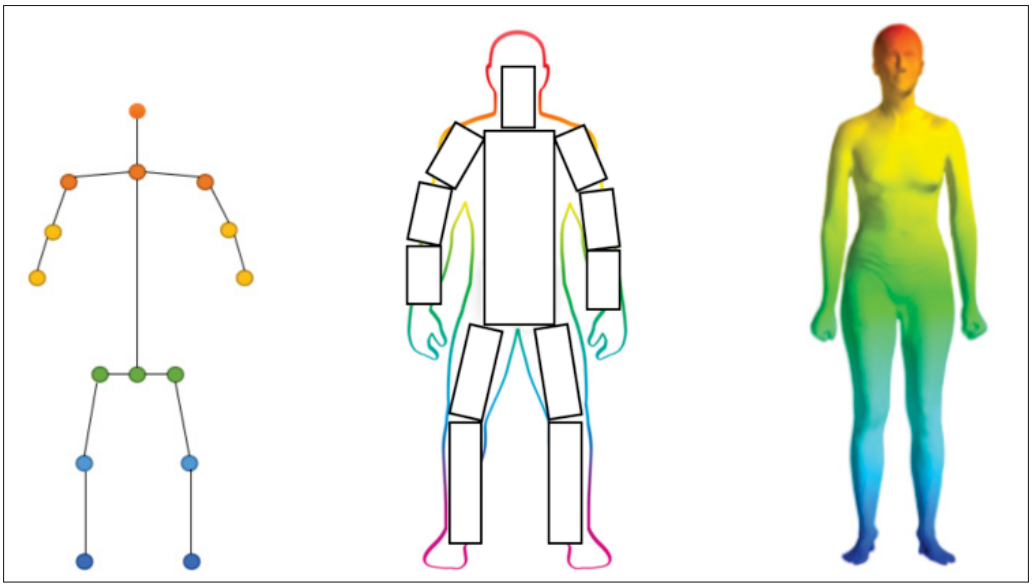

Figure 1.13 Human body modeling. From left to right
(1) Kinematic, (2) Planar, (3) Volumetric
Taken from Zheng *et al.* (2020)

HPE has many applications including human-computer interaction (HCI), action recognition, motion analysis, video surveillance, medical assistance, animation, gaming, augmented reality (AR), and virtual reality (VR). Deep learning solutions have achieved high performance in 2D HPE of a single-person, driving the research to the more challenging problem of highly occluded multi-person HPE in complex scenes. Highly occluded multi-animal scenes, e.g., free-stall barns where dozens of cattle can move around freely, are also challenging in Animal Localization. Other challenges in HPE includes insufficient training data, depth ambiguities, and strong articulations. Here we summarize a few important deep learning methods in HPE. The first major work that applied CNN-based regression to HPE is DeepPose (Toshev & Szegedy, 2014). DeepPose reviews the limitations of part-based models and emphasizes on the need for a holistic view of HPE, i.e., estimating the joint location using a complex nonlinear transformation of the full image. The employed CNN is an AlexNet (Krizhevsky *et al.*, 2012) with the final FC layer outputting a pose vector (with a size of $2k$) that contains x and y coordinates for k joints. The model is trained on FLIC (Sapp & Taskar, 2013) and LSP (Johnson & Everingham, 2010, 2011) datasets using L2 loss. Evaluation is performed by Percentage of Correct Parts (PCP) and Percentage of Correct Joints (PCJ) metrics.

1.4.1.1 Animal Localization

Reliable and robust tracking of individual cows is necessary for analyzing the behaviors of dairy cattle to maintain their health and welfare (Mahmud, Zahid, Das, Muzammil & Khan, 2021). Tracking can be done by either using sensors (e.g., motion, bubble) or using cameras. Here, we focus on research in tracking using optical cameras. Installing cameras in dairy farms is constrained by the type of housing environment. Although this work only utilizes cows in tie-stall, we review previous work in developing automatic tracking systems for cows in free-stall as well. Mainly because we can easily generalize this approach to develop tracking methods in different housing environments.

In (Guzhva, Ardö, Nilsson, Herlin & Tufvesson, 2018), a 2-step CNN architecture is used to develop a robust tracking system to detect cows in a free-stall. First a deep VGG network without fully connected layers produces a five-channel probability map to detect the landmark class (e.g., background, cow center, cow head) that serves as the input for a second shallow CNN to detect the cows and their orientations (described by 32 different classes) given a top-view input image containing multiple cows.

1.4.2 Land Cover Segmentation

The research in remote sensing, specifically land cover classification, is mostly motivated by the lack of large annotated SAR datasets. In (Huang, Dumitru, Pan, Lei & Datcu, 2020), a very deep pre-trained residual network is fine tuned on natural images for remote sensing and validated on SAR data using transitive learning. A cost-sensitive top-2 smooth loss function is used to reduce the effect of imbalanced dataset with mislabeled training samples. In (Wu, Li, Zhang, Li & Guo, 2018), a new dataset is developed to evaluate the performance of deep residual CNN models on polarimetric SAR scene classification. Using transfer learning, manifold polarimetric decompositions are incorporated into the model without losing the spatial features. In (Seferbekov, Iglovikov, Buslaev & Shvets, 2018), a feature pyramid network (FPN), pretrained on ImageNet, is utilized to develop a model for multi-class segmentation of land cover on

satellite imagery collected by DigitalGlobe's satellite. To reduce overfitting, a spatial dropout unit is deployed on the output layer. In (Zhang *et al.*, 2019), the labels of land cover and land use are utilized jointly to train a deep convolutional neural network for land cover classification.

Most of the research in land cover classification focuses on particular tasks over similar regions. Optical imagery is usually incorporated for label generation. More importantly, models are developed for image-level land cover classification. In this work, we train and validate deep segmentation models using SAR imagery only. Also, rather than image-level, we perform the classification on pixel-level.

1.4.3 Discussion

Firstly, we discussed the importance of robust tracking of individual dairy cattle for analyzing their behavior and maintaining their welfare. In view of constraints imposed by the housing environments, it is challenging to install and maintain optical cameras in dairy farms. The previous work in cattle localization focuses on detecting landmark classes, the cows, and their orientation in free-stall. The focus in the first part of this thesis is on developing a robust localization tool for dairy cattle in tie-stall that can be generalized to different housing environments.

Secondly, we reviewed the previous work in remote sensing, specifically in the context of land cover classification. Motivated by the lack of large annotated synthetic aperture radar (SAR) datasets, several studies focus on utilizing transfer learning to fine tune models mitigating the lack of data for remote sensing applications. Most of the previous work focuses on land cover classification task over similar regions with optical imagery. In the second part of this thesis, we focus on land cover segmentation using deep convolutional neural networks (CNN) on SAR imagery only.

CHAPTER 2

AUTOMATION OF VIDEO-BASED LOCATION TRACKING TOOL FOR DAIRY COWS IN THEIR HOUSING STALLS USING DEEP LEARNING

2.1 Introduction

Tracking animal movement is a tool widely used in animal behavior research. Animal movement can be tracked from a large scale such as further understanding the migratory patterns of birds (Guilford *et al.*, 2011), to a small scale such as understanding the movements and behaviors of parasitoid wasps (Abram, Parent, Brodeur & Boivin, 2016). While the scale of behavioral tracking may vary, the objectives and challenges of developing new tracking software remains the same, particularly in terms of maximizing accuracy (Shepley, Berthelot & Vasseur, 2017).

Automated tracking of production animal movement within the housing environment is a useful tool to measure the efficacy of different interventions aiming at improving comfort and welfare. With the intensification of dairy cow husbandry systems, welfare concerns associated with limited opportunities for movement have become a major research focus (Vasseur, 2017). Tracking of cows' movement within their home stalls could provide, in conjunction with other common outcome measures (e.g., activity levels, physiological parameters), a greater insight into the level of comfort provided to cows in different environmental set-ups than with those common measures alone. This can be used to better support the development of recommendations for animal welfare housing improvements (Boyer, de Passillé, Adam & Vasseur, 2021a; Boyer *et al.*, 2021b; McPherson & Vasseur, 2021; St John, Rushen, Adam & Vasseur, 2021). Yet, commercially available video-based automated tracking software for researchers in animal behavior and welfare sciences (e.g., Ethovision XT®, iDTracker©, Bio-Tracking©) are expensive and show limitations in their ability to accurately and efficiently track the movement of dairy cows due to variation in the housing environment (e.g., different surfaces, colors and lights, people walking through frame, neighboring cows). As an alternative, manual tracking of farm animals is costly and time-intensive (i.e., several hours for 1 h of video recording), since observers need to be properly trained and location coordinates need to be visually marked over subsequent images. Recent

research in computer vision have allowed the development of novel methods for tracking animals in their production environment using algorithm-based ability to detect, identify, and follow objects in a continuous manner, overtime, and under varying conditions. Deep learning has been successful in visual tracking applications due to its accuracy in feature learning. Developing a software that uses deep learning techniques to progressively extract higher level features from raw image input may be more effective than available software at discriminating environmental variation. Software that uses convolutional neural networks (CNN) to track the movement of an object against a diverse background have shown to be more accurate and time efficient, especially those that are developed with deep features (Li, Wang, Wang & Lu, 2018). As an example of CNN application, (Guzhva *et al.*, 2018) showed that an individual cow could be successfully tracked for over 20 min in a mildly crowded free-stall environment. Unfortunately, one model is not generalizable to all situations and requires validation. The objective of our study was to validate whether it is feasible to develop an accurate and low-cost alternative to manual cow tracking of spatial use in their tie-stall using deep learning techniques.

2.2 Materials and Methods

The certified Animal Care Committee of McGill University and affiliated hospitals and research institutes reviewed and approved the use of animals in this trial and all procedures used (#2016 – 7794). All aspects of this study met the high standards established by the Canadian Council on Animal Care to ensure the continued humane and ethical use of animals in research.

2.2.1 Sample and Recordings

Lactating Holstein cows (n = 24, average 129 days in milk) were randomly selected from a herd at the Macdonald Campus Dairy Complex (Ste-Anne-de-Bellevue, Québec, Canada). The trial period extended for 10 consecutive weeks from February 20 to May 1, 2017. The cows were housed in two opposite rows of tie-stalls, each facing a wall. Stall dimensions were in accordance with current recommendations following the individual body size of each cow (DFC-NFACC, 2009; Anderson, 2014; Valacta, 2014). The stall base consisted in KKM longline rubber mats

(Gummiwerk Kraiburg Elastik GmbH & Co. KG, Tittmoning, Germany) coated with a thin layer of sawdust bedding ($< 2cm$). Bedding was added once per day in the morning. Cleaning of the stalls and the gutters was performed as needed from 5:00 a.m. until 9:00 p.m. Cows were milked twice per day in stall at 5:00 a.m. and 5:00 p.m. Cows were fed a total mixed ration (TMR) 4 times daily with feed pushups occurring 6 times per day. Water was available *ad libitum* from self-serving water bowls shared between adjacent cows.

To obtain an overhead view of each cow, a network surveillance camera was used (Smart Turret 2.8, Hikvision Digital Technology Co., Ltd., Hangzhou, China; 720p at 60 frames per s; height: $338cm$ centered above each stall). Each cow was filmed for a continuous 24-h period on weeks 1, 2, 3, 6, 8, and 10. Video recording was performed on the same day each week with the camera placed in the same position. Images showing the position of each cow within their stall were obtained from the overhead video recordings. The network video recorder (NVR; Hikvision Digital Technology Co. Ltd., Hangzhou, China) automatically saved the 24-h video recordings into multiple video files of less than 1GB at a resolution of 1280×720 pixels. Using a computer, the FFmpeg© software (64-bit Static; ffmpeg.org) was used to concatenate the separate video files for each 24-h period of each cow into a single file. Following that, the FFmpeg© software was used to extract individual images from each 24-h video file starting at 00:00:00 (hh:mm:ss) with a rate of one image per minute of video recording. The resulting image sequence consisted of 1441 frames per 24-h of video recording for each cow.

2.2.2 Manual Tracking

Cow position within each of the image sequences was then tracked using the Manual Tracking plugin within the FIJI distribution (https://imagej.net/) of ImageJ (https://fiji.sc/), which is a software used for image processing and analysis. This was done by manually annotating three coordinates on each cow to track the location of: 1) the left hip, 2) the right hip, and 3) the neck (Figure 2.1). While other points such as the cow's head, shoulder tips, pin bones and tailhead were considered for tracking, the hips and neck base base were retained as they were consistently visible on the video recordings, even when the cow was lying down. The two hip bones were

chosen to represent the widest point of the body and provide information on the use of the space at the back of the stall, while the base of the neck was tracked to follow the orientation of each cow while providing information on the cow's use of the space at the front of the stall. Each coordinate was composed of two integer numbers representing one point on a Cartesian coordinate system. These coordinates were defined as $P_1 = (X_1, Y_1)$, $P_2 = (X_2, Y_2)$, $P_3 = (X_3, Y_3)$. P_1 localized the tip of the left hip, P_2 the tip of the right hip, and P_3 the base of the neck. Once the image sequence was imported into ImageJ, the Manual Tracking plugin was opened, and a track of P_1 was created. For the entire image sequence, P_1 was manually selected by an observer while the Manual Tracking plugin recorded the pixel location (X_1, Y_1) and then switched to the next image in the sequence. After completing the annotation of P_1 for the entire image sequence, the process was repeated for the other body points (P_2, P_3). At the end of the manual tracking process, all 1441 frames of each image sequence were annotated with the pixel location of coordinates P_1, P_2, and P_3. This was repeated for all 24 cows over each of the 6 weeks considered.

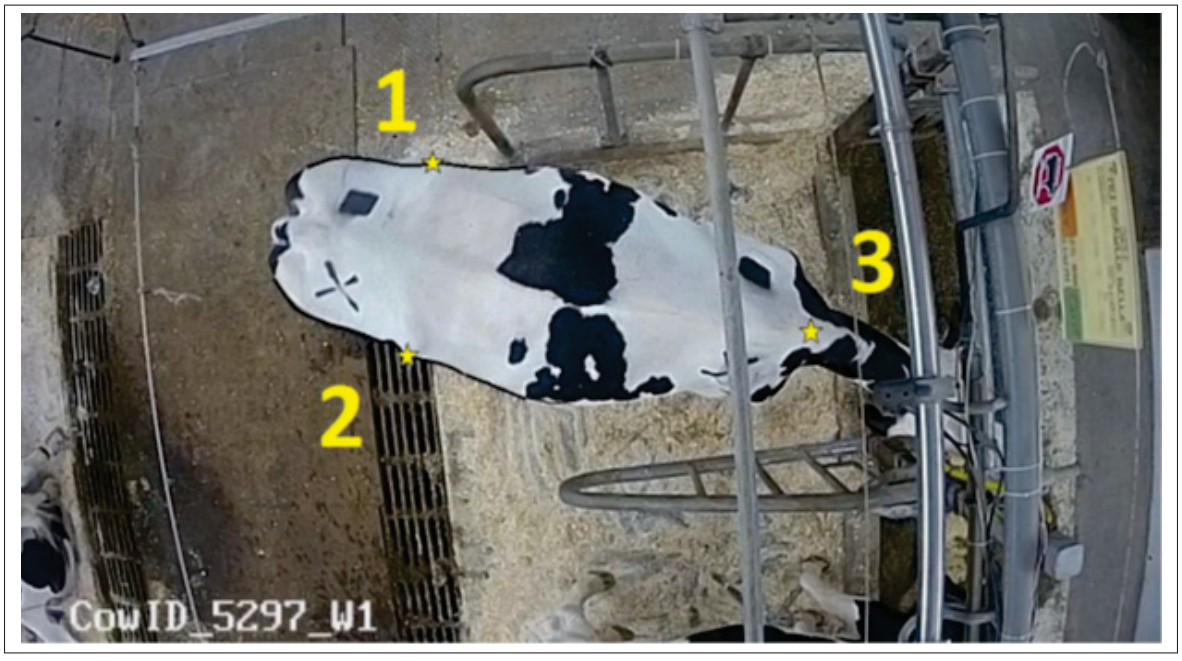

Figure 2.1 An overhead video camera image of a tie-stall cow with the three manually tracked body coordinates labelled (yellow stars) to show (1) the tip of the left hip bone, (2) the tip of the right hip bone, and (3) the base of the neck

To confirm the precision of the manual tracking method, a Coefficient of Variation (CV) was used to measure the variability in the hip-to-hip distance in pixels across an image sequence. For manual tracking to be carried out properly, the distance between the two tracked hip points was expected to remain constant for the same cow over a 24-h period. To ensure a consistent measurement of the hip-to-hip distance, the optical distortion and height of the cow relative to the camera over distinct stall areas was considered. As such, 4 areas ($A1 \ldots A4$) of the stall were defined by pixel coordinates at the boundaries of the stall. The coordinates in (Figure 2.2) illustrate the manual tracking of a cow over a 24-h period within the tie-stall (red dot = location of the left hip, blue dot = location of the right hip, and yellow dot = location of the base of the neck). Four areas ($A1 \ldots A4$) of the stall are demarked by pixel boundaries (black dotted line) relative to the confines of the tie-stall (pink lines). $A1$ shows the alley behind the stall, where cows occasionally exit the stall area. $A2$ shows the back 25% of the stall. $A3$ shows the second 25% from the back of the stall. $A4$ shows the remainder of the image. The pixel coordinates for the confines of the stall were determined using the Manual Tracking plugin, by selecting the extremes of each corner of the stall, which remained the same throughout a single image sequence.

For the image sequences of each cow, the CV and a pixel to cm (pix/cm) conversion ratio were calculated separately for all 4 stall areas. First, the P_1 and P_2 coordinates were categorized for each image based on which area of the stall the hips were found. The CV was then calculated for each stall area by dividing the standard deviation of the hip-to-hip pixel distances by the mean hip-to-hip pixel distances for the entire image sequence. The pix/cm ratio was calculated for each stall area by dividing the hip-to-hip distance in pixels by the hip-to-hip distance in cm for the entire image sequence. The CV was then used to identify sets of tracking coordinates (P_1 and P_2) that were problematic and needed to be reassessed. A CV of less than 5% was deemed acceptable based on the system by (Lin, 1989).

The final dataset used to evaluate deep learning accuracy on regressing tracking coordinates consists of 199,100 samples. Each sample includes a red-green-blue (RGB) image with three channels manually annotated with tracking coordinates (P_1, P_2, P_3) by human experts. Images

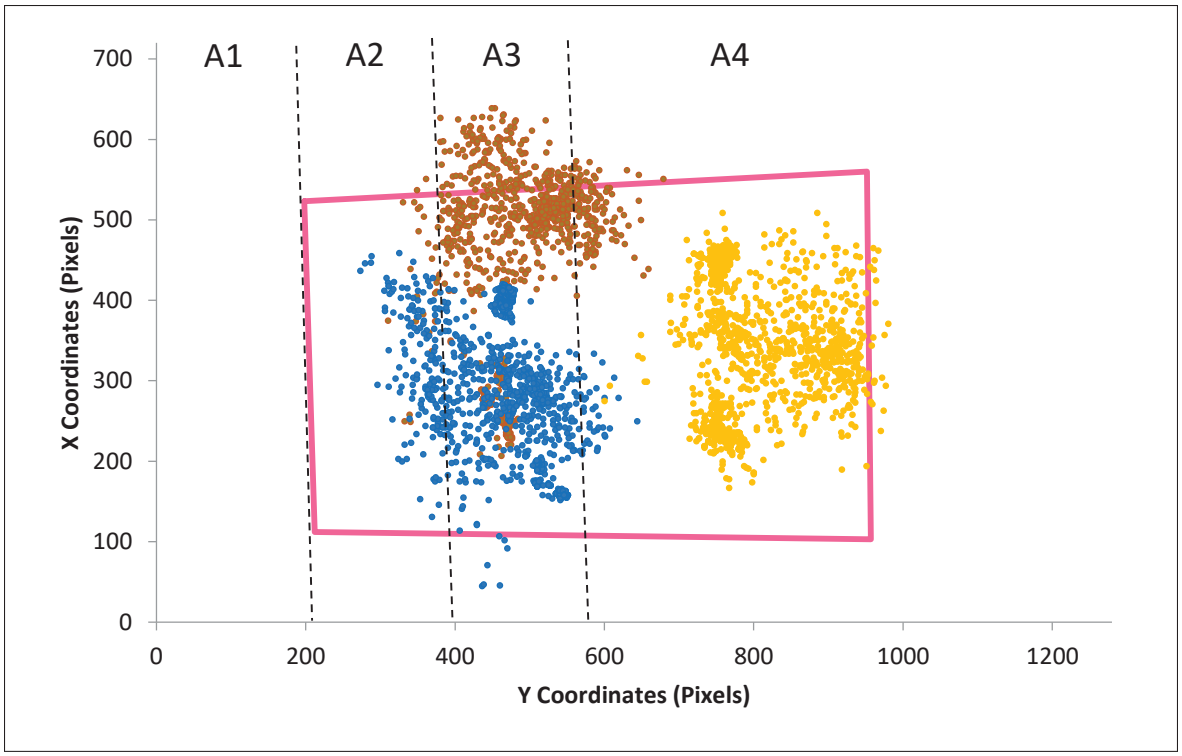

Figure 2.2 A Cartesian coordinate system with the X and Y axes showing the dimensions of a single 1280×720 pixel image

were taken from 24 cows over 6 weeks with at least 5,600 unique captures per cow (5,683 to 8,683).

2.2.3 Automated Tracking

Deep residual networks (He *et al.*, 2015) have shown excellent generalization performance on visual recognition tasks by addressing the degradation problem using residual learning with identity connections. Candidate models used to experiment with were developed using the architecture of Resnet-18 (He *et al.*, 2015) and other top performing CNN models namely VGG (Simonyan & Zisserman, 2015), GoogLeNet (Szegedy, Vanhoucke, Ioffe, Shlens & Wojna, 2015b), DenseNet (Huang, Liu & Weinberger, 2016), and U-Net (Ronneberger, Fischer & Brox, 2015). Batch normalization (Ioffe & Szegedy, 2015) was integrated in models under experiment. Rescaling the sample images by a factor of 0.25 resulted in a spatial dimension of 180 pixels height and 320 pixels width. This allowed for accommodating deeper models in GPU memory

as well as faster training epochs. No data augmentation was applied during experiments, because image transformations (random flip/crop/rotation) used in data augmentation change the position of cow, thus invalidating its tracking coordinates.

The optimization objective employed to train models calculates the distance between model predictions and the ground truth (i.e., the manual tracking coordinates). Leave-one-out cross-validation was used to train the models. By this method, the number of folds was set to 24 since there were data from 24 dairy cows. All models were trained 24 times having one cow as the validation set and the remaining 23 cows as the training set. In addition, the models were trained using L1, L2, and Huber loss functions. L1 loss calculates the absolute difference between respective values of model predictions and the ground truth. L2 loss calculates the average of squared difference between the same values. Huber loss (Mangasarian & Musicant, 2000) is a combination of Mean Absolute Error (L1) and Mean Squared Error (L2) functions defined with a value of δ. This criterion calculates the L2 loss when the absolute element-wise error falls below the $\delta = 1$, and L1 loss otherwise. Huber loss (also called L1 smooth loss) is more robust to outliers. Computing the gradient of loss function with respect to model parameters provides the optimizer with the amount and direction of adjustments for parameters of every layer. Stochastic gradient descent optimizes model parameters using the computed gradient scaled by a learning rate of 1×10^{-3}, accelerated by a momentum (Sutskever, Martens, Dahl & Hinton, 2013) of 0.9, and regularized by a weight decay of 5×10^{-4}.

Every training epoch started with splitting all samples from the shuffled training set using an 80:20 ratio, which was then used to train and test the model, respectively. In addition, the training split was further split into mini batches of size 32. Model parameters were updated through every iteration of batches. Decaying the learning rate every 20 epochs by scaling it by a gamma of 0.1 helped the model to learn more complex patterns (You, Long, Jordan & Jordan, 2019). At the end of each epoch, the accuracy of the trained model was measured on the testing split by averaging the respective distances of tracking coordinates between model prediction and the ground truth in pixels. Elaborately, for tracking coordinates of the i^{th} sample in the validation set, average pixel error (APE) calculates the Euclidean distance between model prediction (x, y)

and the ground truth $(\hat{x}, \hat{y})$ for every point $P_j^i = (x_{P_j}^i, y_{P_j}^i)$ where $j \in \{1, 2, 3\}$, as shown in Equation 2.1.

$$\text{APE} = \frac{1}{3} \sum_{j=1}^{3} \sqrt{(x_{P_j}^i - \hat{x}_{P_j}^i)^2 + (y_{P_j}^i - \hat{y}_{P_j}^i)^2} \tag{2.1}$$

During training, early stopping criterion was adopted to reduce the chance of overfitting and to reduce training time (Coulibaly, Anctil & Bobée, 2000). Training was allowed until the prediction accuracy did not improve for 8 consecutive epochs. The pixel error values have been reported for all 3 points of tracking coordinates (P_1, P_2, and P_3). Converting the unit from pixels to a standard measure (cm) was carried out using the calculated pix/cm conversion ratio. The implementation of these experiments is available at: https://github.com/CowLifeMcGill/biotracking.

2.3 Results and Discussion

The manual tracking method was validated to precisely annotate the coordinates of P_1, P_2, and P_3. For this method, a very low CV was found with an average of 2.5% variability in the hip-to-hip distance calculated within each 24-h period across all cows. Based on prior literature, a CV of less than 5% has been deemed acceptable based on the system by (Lin, 1989). The precision of these annotations was important to ensure proper training of the deep learning models experimented with. For applications in which the deep learning model cannot be used, the manual tracking method offers a precise way to track the movement of coordinates through a series of images. While this method offers low variability, it is not considered the most time efficient approach to tracking, taking between 3 to 5 hours per set of 1441 images for annotation depending on observer abilities. As such, deep learning models were developed as a more feasible alternative.

Of the several models we tested in our experiments, Resnet-18 was the most promising candidate model. The models based on Resnet architecture achieved the lowest normalized average pixel error on the validation dataset across all 3 annotation points (Table 2.1). In Resnet models, the spatial dimensions of the input image are significantly reduces by the first convolutional/pooling layer, which could potentially reduce the details extracted and needed to train an accurate

regressor. However, this was not an issue in the present study, since the Resnet-18 had the best performance. The training set contained more than 180,000 samples, however, only from 24 cows. Considering the unique skin patterns of cows, the limited number of cows in the training set could had reduced the generalization ability of the evaluated models.

The pixel error of Resnet-18 models for annotation of the validation image set was lowest for P_1 at 1.20 pixels, intermediate for P_2 at 1.34 pixels, and highest for P_3 at 1.81 pixels (Table 2.1). Except for the U-Net models, the same trend was observed for all models evaluated (Table 2.1). The lowest pixel error when annotating the right and left hips compared to the base of the neck was likely due to the hip coordinates lying at the boundary of the cow where an edge is present to delineate the cow from the surrounding environment. The base of the neck however exists within the body of the cow where no edge or delineating point is present. As such, variation in the physical attributes between cows (for instance, color and skin patterns) may have a larger impact on the annotation accuracy of the base of the neck relative to the hips.

Using the pix/cm ratio, the overall pixel error for annotation of the validation image set was equal to an average of 1.44 cm error for Resnet-18 in actual physical placement of the points within the stall environment (Table 2.1). In other words, the Resnet-18 models were able to locate the hips and base of the neck of each cow relative to the stall environment with an average error of 1.44 cm. Therefore, this model offers a high degree of accuracy to track the movement of cows in an automated fashion within and around the stall environment. The annotation of the training image set had a lower pixel error for all coordinates compared to the validation image set in all models except for the U-Net models (Table 2.1). This difference in pixel error was associated with a negligible difference in actual physical placement of the coordinates within the stall environment (average difference ranging from 1.20 to 9.25 cm; Table 2.1). This is expected given that the images from the training set had been previously present in an annotated form to each model, while the validation set images had not. For application of the models, the outlined methodology should be replicated as precisely as possible in terms of camera placement and type to avoid introducing novel forms of variability to the images. Visual inspection of image annotation by the models can be used to ensure that a high pixel accuracy is being maintained.

Table 2.1 Average degree of error presented in pixels and cm of the ResNet18 with distance/angle label augmentation for coordinate annotation of the training and validation datasets, where P1 is the location of the left hip bone, P2 is the right hip bone, and P3 is the base of the neck

Model	Dataset	M.	P1 E.	P2 E.	P3 E.	Overall E.
Resnet-18	Training	pixel	1.92 (0.70)	2.04 (0.73)	3.16 (0.94)	2.48 (0.90)
		cm	0.48 (0.17)	0.51 (0.18)	0.79 (0.23)	0.92 (0.23)
	Validation	pixel	4.82 (2.68)	5.35 (2.46)	7.25 (3.23)	5.76 (2.61)
		cm	1.20 (0.67)	1.34 (0.61)	1.81 (0.81)	1.44 (0.65)
VGG-11	Training	pixel	4.98 (1.45)	5.38 (1.58)	6.97 (1.88)	5.78 (1.63)
		cm	1.25 (0.36)	1.35 (0.39)	1.74 (0.47)	1.44 (0.41)
	Validation	pixel	6.08 (3.75)	6.06 (3.63)	8.24 (4.17)	6.79 (3.74)
		cm	1.52 (0.94)	1.52 (0.91)	2.06 (1.04)	1.70 (0.94)
GoogLeNet	Training	pixel	3.97 (2.04)	4.23 (2.09)	5.67 (2.86)	4.62 (2.32)
		cm	0.99 (0.51)	1.06 (0.52)	1.42 (0.71)	1.15 (0.58)
	Validation	pixel	23.9 (3.59)	24.2 (3.23)	25.9 (2.86)	24.7 (2.88)
		cm	5.97 (0.90)	6.06 (0.81)	6.47 (0.71)	6.16 (0.72)
DenseNet	Training	pixel	4.07 (2.47)	4.17 (2.48)	5.37 (2.71)	4.53 (2.55)
		cm	1.02 (0.62)	1.04 (0.62)	1.34 (0.68)	1.13 (0.64)
	Validation	pixel	5.70 (2.45)	5.96 (2.52)	7.61 (2.76)	6.42 (2.39)
		cm	1.43 (0.61)	1.49 (0.63)	1.90 (0.69)	1.61 (0.60)
U-Net	Training	pixel	38.2 (0.86)	36.9 (0.31)	38.5 (0.93)	37.3 (2.68)
		cm	9.54 (0.22)	9.23 (0.08)	9.63 (0.23)	9.33 (0.67)
	Validation	pixel	37.6 (9.34)	35.9 (8.00)	37.8 (7.62)	36.4 (7.55)
		cm	9.39 (2.33)	8.99 (2.00)	9.45 (1.91)	9.11 (1.89)

2.4 Conclusion

The results obtained in this study validated the potential of Resnet-18 models to be used in the development of an accurate and low-cost alternative to manual cow tracking and space use in their tie-stalls. Future steps in the development of a tool usable for cow comfort and behavior research purposes will seek to address the limitations of this experiment such as the small number of images available for annotation. Yet, our results have demonstrated that training an existing algorithm not developed for the tracking of cow movement in a tie-stall setting can yield high accuracy, as the models tested and presented proved able to accurately pinpoint the position of the hips and of the base of the neck. The position of the hips and base of the neck can be compared with the pixel boundaries of the stall confines to determine the percentage of

time and distance with which each cow spends outside the stall, which represents biologically relevant information usable in a behavior research context, including the assessment of possible stall design improvements through the evaluation of how cows move within their surrounding environment. This evaluation is currently done 'manually' by an observer analyzing video data; the methodology we worked on will, following further validation steps, allow for this process to be automatized, substantially reducing the time required to collect this data. Future use of this methodology includes the analysis of individual movement patterns demonstrated by dairy cows in their housing environment for comparison with other commonly used outcome measure (for instance, activity level or animal posture) to assess ease of movement and stall comfort, and better support the development of recommendations for animal welfare housing improvement.

CHAPTER 3

RADARSAT-2 SYNTHETIC-APERTURE RADAR LAND COVER SEGMENTATION USING DEEP CONVOLUTIONAL NEURAL NETWORKS

3.1 Introduction

Land use and land cover (LULC) information has many applications in remote sensing (Yumus & Ozkazanc, 2019) both in military and civil fields (Gao *et al.*, 2018). Automatic LULC classification plays a significant role in urban planning, natural resource management (Liu *et al.*, 2017), forest monitoring (Lapini *et al.*, 2020), and understanding the rapid changes on the surface of the Earth (Zhang *et al.*, 2019). Optical images can be obtained from the Earth's surface using aerial photography or satellite imagery. Another technique in satellite imagery i.e. Synthetic Aperture Radar (SAR) imagery, captures physical properties by transmitting electromagnetic energy (microwave signals), toward the surface of the Earth and measuring the distance between the sensor and the point on the Earth's surface where the signal is backscattered. Unlike optical satellite imagery which depends on natural light, SAR signals can penetrate through clouds and other obstacles e.g. rain and snow, and can be taken during night. Detecting changes in the Earth's surface e.g. changes in habitats, levels of water and moisture, effects of natural or human disturbance, etc. are some applications of SAR imagery (wha).

Complex scattering characteristics caused by different wavelengths and incidence angles (Huang *et al.*, 2020) introduce significant levels of noise in SAR images. The lack of large public labeled SAR datasets along with noisy images and imbalanced land cover categories contribute to the challenging nature of land cover classification using SAR imagery. In this work, a private dataset of multiple SAR images obtained by RADARSAT-2 satellite on major Canadian cities (Montreal, Ottawa, Quebec, Saskatoon, Toronto and Vancouver) is used to train different deep learning models. We compare the performance for pixel-level classification of four different land cover classes: Urban, Water, Vegetation and Farm. Multiple SAR samples obtained at different times over each city are averaged to reduce noise and increase the quality of the SAR image. Using stacked SAR data, three top performing deep semantic segmentation models are trained and

evaluated. The proposed networks are proved to have excellent generalization performance. The achieved segmentation performance on unseen data, promises a general robust solution for automatic remote sensing, specifically land cover pixel-level classification.

3.2 Proposed Method

In this section, we present the proposed method. Starting by an overview to the method, and then presenting the architecture of deep CNNs used for semantic segmentation, we describe the details of our method.

3.2.1 Overview

We are given a set of data with land cover annotations, $D = \{I_n, Y_n\}_{n=1}^{N}$ where N is the number of images, I_n is the nth image, and Y_n is the pixel-level annotations. The land cover annotations have the same dimensions as the SAR image with only one channel. The value for each pixel represents the class label of that pixel. The land cover classes are annotated at 30-meter resolution. We aim to learn a CNN-based model in an end-to-end fashion for land cover segmentation by using the training data $X \subset D$.

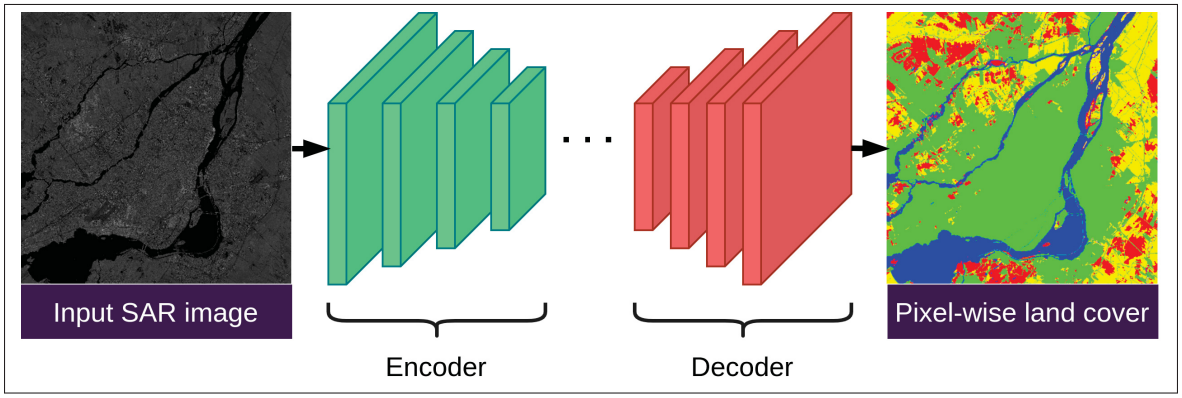

Figure 3.1 Overview of our method. The illustrated CNN model is based on the Encoder-Decoder approach for semantic segmentation

The overview of our method is given in Figure 3.1. Segmentation models that are used in this work, are designed based on two approaches. 1. *Convolutional Encoder-Decoder*, and 2. *Multi-Scale Pyramid Pooling*. In the **encoder-decoder** approach, first, encoder part of the network extracts meaningful features from the SAR image by transforming the image to a multidimensional vector representing features of the image (Noh *et al.*, 2015). The represented features are input to the decoder part of the network, where the target segmentation map is reconstructed by predicting pixel-wise class probabilities from extracted patterns (Minaee *et al.*, 2020). In the **multi-scale pyramid pooling** approach, a residual network (He *et al.*, 2015) works as the feature extractor. Using pyramid pooling, the feature maps are downsampled at multiple scales. Summarized pooled features are upsampled and concatenated with the initial feature maps to reconstruct the segmentation map by generating pixel-wise predictions (Zhao *et al.*, 2016). The entire network is trained using cross entropy loss. Working with extremely large SAR images – over 100 million pixels per image – it is impractical to train deep models without a proper sampling method where samples are smaller overlapping images preserving information around the borders, and only containing valid category labels. We use Deconvnet (Noh *et al.*, 2015), Segnet (Badrinarayanan *et al.*, 2015), and PSPNet (Zhao *et al.*, 2016) for evaluating the performance of deep convolutional segmentation models with SAR images. Now, we provide more details about each model's architecture.

3.2.2 Encoder-Decoder

Most common deep models for image segmentation are based on the convolutional encoder-decoder architecture. The encoder network takes the input image, e.g. an image of a street full of cars, and extracts different patterns. The decoder network takes the extracted features and reconstructs a map of pixel-wise class probabilities, e.g. segmentation masks of cars, pedestrians, and trees.

3.2.2.1 Deconvnet

Based on the architecture of VGG 16-layer net, the encoder of Deconvnet consists of five pooling layers following five groups of convolutional layers with {64, 128, 256, 512, 512} channels. Two fully connected layers with 4096 features follow the last pooling layer. The decoder – deconvolution – network is designed by mirroring the encoder, replacing convolutional layers with transposed convolution operation and max pooling layers with max unpool operation. Every weighted layer is followed by batch normalization (Ioffe & Szegedy, 2015) and rectification (Nair & Hinton, 2010) layers. The encoder network reduces the size of activations through convolution and pooling layers, and the decoder network enlarges the size of activations through deconvolution and unpooling layers (Noh *et al.*, 2015). The last layer of Deconvnet is a 1×1 convolution producing the score map for segmentation.

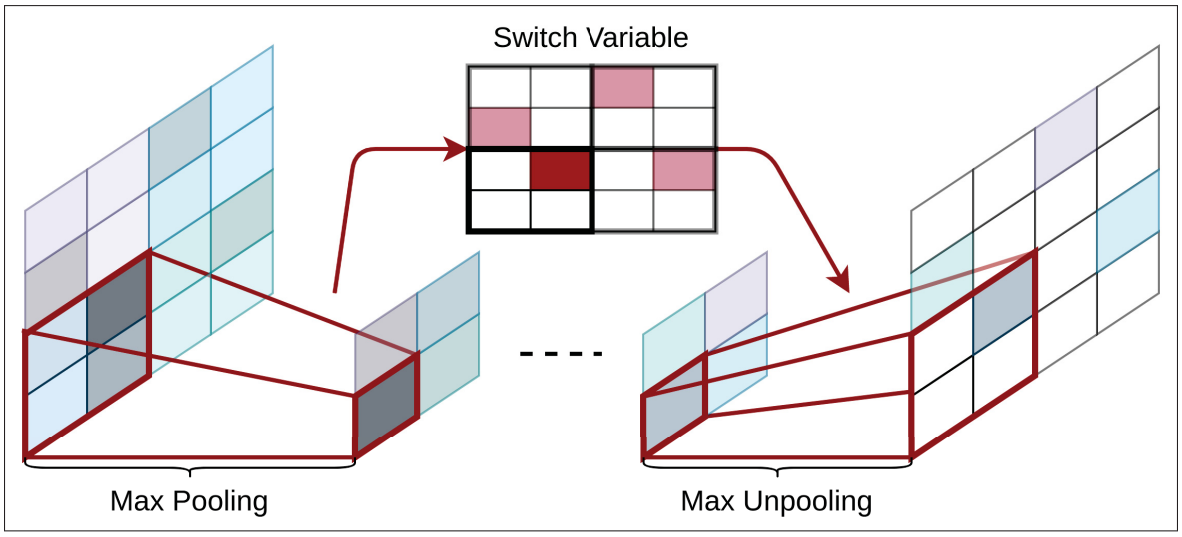

Figure 3.2 Illustration of unpooling operation

3.2.2.1.1 Unpooling

This operation is used to reconstruct the spatial structure of input image for segmentation. The indices of selected activations during pooling are stored in a switch variable. The saved indices

are then used for placing each activation back to its original pooled index. This operation is illustrated in Figure 3.2.

3.2.2.1.2 Deconvolution

This operation is used to associate the sparse enlarged activations obtained by unpooling to a dense activation map using multiple learned filters. These filters learn to reconstruct the shape of input image. While the shape of SAR images are captures by the filters in lower layers of the decoder network, the filters in higher layers tend to learn the details of textures, that is used subsequently to predict the land cover labels for each pixel.

3.2.2.2 Segnet

The architecture of Segnet is almost identical to Deconvnet. However, Segnet promises better efficiency by removing the two fully-connected layers. Therefore, the pooled activations obtained from the last layer of encoder network are immediate inputs to the first layer of decoder network. One advantage of removing fully connected layers is the significant reduction in number of model parameters – from 134M to 14.7M. This improves generalization performance and reduces computational cost. Also, the feature maps at the deepest layer of decoder network retain higher spatial resolution. This helps the decoder network to reconstruct the segmentation map with finer details. Similar to Deconvnet, Segnet stores the indices of pooled activations in a switch variable used by unpooling layers to reconstruct the shape of segmentation map. The architecture of Segnet is illustrated in Figure 3.3.

3.2.3 Pyramid Pooling

Spatial pyramid pooling (SPP) (He, Zhang, Ren & Sun, 2014) is a pooling mechanism that is usually applied to the activations of the last convolutional layer. Assuming a fix number of bins, the feature maps are spatially divided into bins with sizes proportional to the input image. Using max pooling operation, SSP downsamples feature maps inside each bin individually. This results

in an output vector of size $n \times C$, where n and C are the number of bins and the number of channels of input activations respectively. Bins are captured at different pyramid scales. Smaller bins contour fine-grained information; where larger bins include coarse-grained information. Lastly, the pooled features of bins within each scale are concatenated together forming a single feature vector. SPP is illustrated in Figure 3.4.

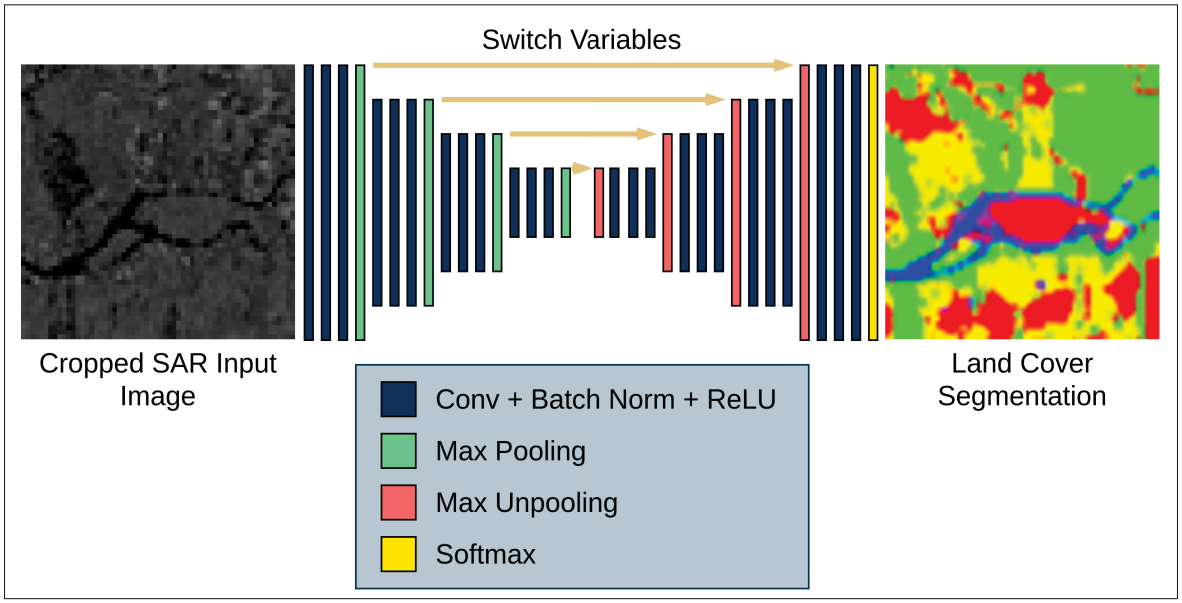

Figure 3.3 Illustration of Segnet architecture. Switch variables are saved during downsampling and used to unpool the activations during upsampling

3.2.3.1 PSPNet

Pyramid scene parsing network (PSPNet) proposes an effective global context prior for tackling complex scene parsing problems. Problems related to contextual relationship and global information e.g. mismatched relationship, confusion categories, and inconspicuous classes are addressed by pyramid pooling module in PSPNet. The architecture of PSPNet is illustrated in Figure 3.5.

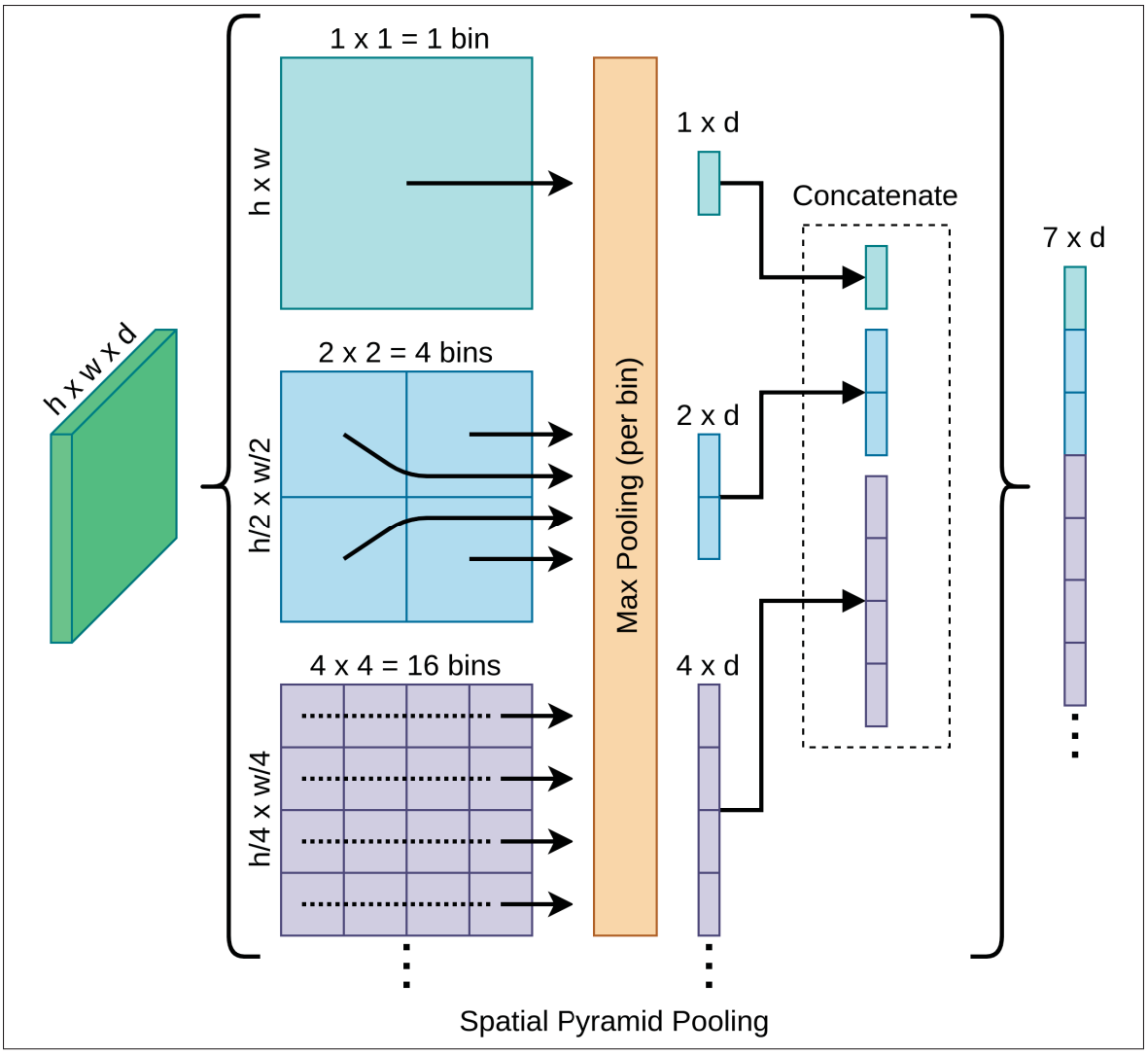

Figure 3.4 Spatial pyramid pooling layer

3.3 Experimental Results

In this section, we evaluate the proposed method. We describe the dataset and evaluation metrics. We also present our final results.

3.3.1 Dataset

The LCSAR dataset contains 116 high resolution SAR images from regions of Montreal, Ottawa, Quebec, Saskatoon, Toronto and Vancouver with land cover labels annotated at 30-meter

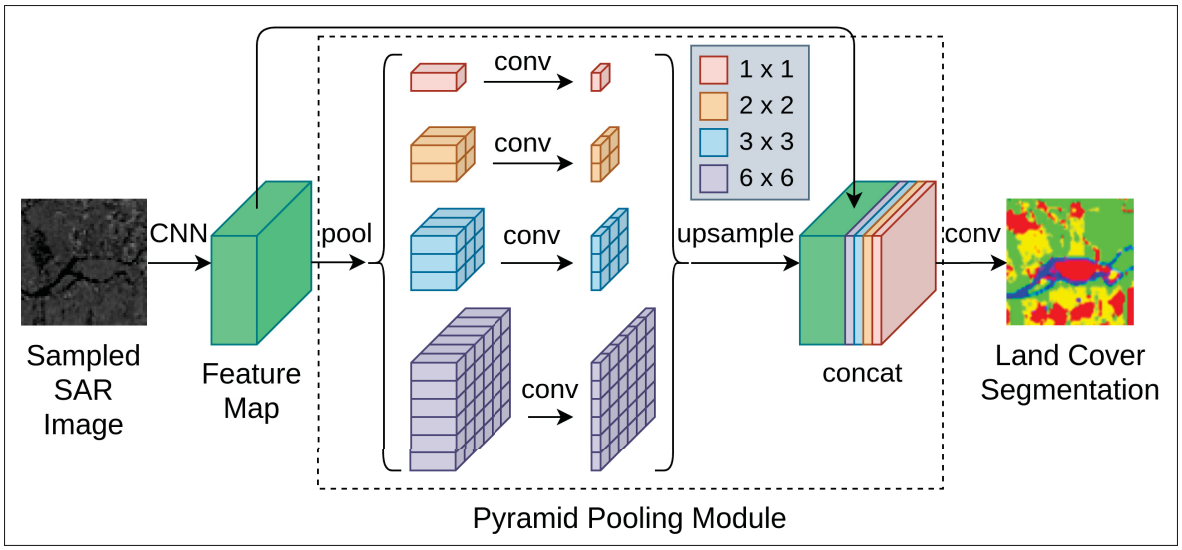

Figure 3.5 Illustration of PSPNet architecture. Pyramid pooling is applied at four different resolutions to preserve global context information as well as finer details

resolution. The SAR images are taken using different beam modes resulting in more variance in the dataset. Most of the SAR images are obtained 24 days apart. To reduce the noisy backscattered data, images from each region are stacked by averaging pixel values resulting in 6 final SAR images to sample from. These stacks include 19 images from Montreal, Ottawa, Saskatoon and Toronto, 18 images from Quebec and 23 images from Vancouver. The SAR images were obtained from 2010 to 2015.

3.3.1.1 Land cover classes

The land cover segmentation labels are summarized in 4 classes. Each class is identified with an integer number as the pixel value in the ground-truth segmentation image. Table 3.1 describes the four classes used in land cover labels. The land cover images contain two auxiliary classes that are identified by ID codes -999 and 0 labeled *Unknown* and *Empty* respectively. The unknown class describes that the correct land cover category is not determined; where, the empty class fills the blank area surrounding originally tilted SAR images – due to the angle offsets.

Table 3.1 Land cover classes. Vegetation class includes natural parks and forests, while Farm class labels include agricultural lands

ID Code	Class Description	Color Code
21	Urban	Green
31	Water	Water
41	Vegetation	Red
51	Farm	Yellow

3.3.1.2 Sampling method

Loading very high resolution, single channel, gray-scale SAR images into GPU memory is not feasible, while experimenting on very deep segmentation models. Therefore, we adapt a sampling approach to reduce the resolution/size of the input image. A set of overlapping crops are generated from each SAR image with a kernel size of $K = (K_h, K_w)$ and a stride of $S = (S_h, S_w)$. In our experiments, square crops are sampled with $K_h = K_w$ and $S_h = S_w$. Discarding crops that contain invalid pixel labels (namely *Unknown* and *Empty*) helps to keep the data clean and inhibits outliers. The empty area surrounding tilted SAR images amounts to a considerable area labeled *(*Empty), matching pixels where water is present that are captured almost black in SAR imagery. The number of sample crops, $|D|$, generated from an image with dimensions of (H, W) can be found using Equation 3.1. This method is illustrated in Figure 3.6.

$$|D| = \lfloor \frac{H - K_h}{S_h} + 1 \rfloor \times \lfloor \frac{W - K_w}{S_w} + 1 \rfloor \tag{3.1}$$

We experiment with the variants defined in Table 3.2. The size of training and validation sets decreases proportionally to the size of sampling stride. Therefore, using a smaller stride, we expect to gain generalization performance at the cost of longer training time.

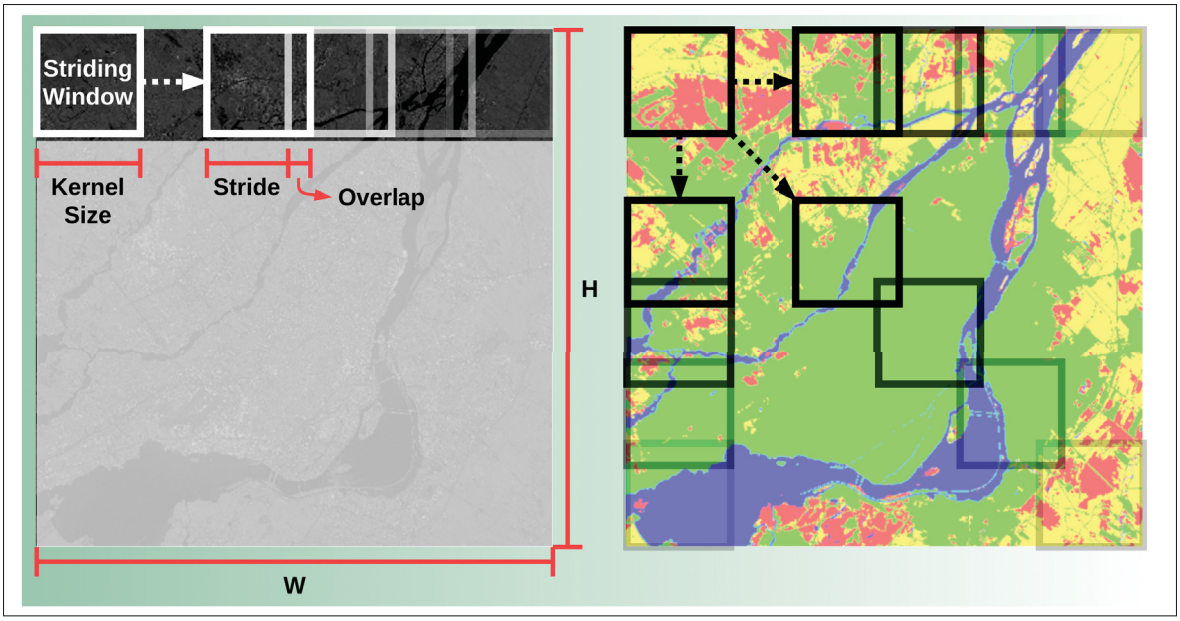

Figure 3.6 Illustration of the sampling method. A crop of the SAR image on Montreal is shown on left with its corresponding land cover label on the right

Table 3.2 Sampling modes. Larger stride size reduces the number of samples and the amount of overlap per sample

Mode	Kernel Size	Stride	Overlap
(**L**)arge	224×224	112	112
(**M**)edium		152	72
(**S**)mall		192	32

3.3.1.3 Training and validation sets

The initial experiments on every SAR image are performed by fixing sampling size, model architecture, hyperparameters, and random initialization to evaluate the performance of each SAR image both for training and validation. The results of these experiments are reported in Table 3.3.

The difference in accuracy of each class is caused by two factors. First, the number of pixels of that class in training and validation samples. Second, the difficulty of classifying that class. In order to further investigate the performance of selected segmentation models and evaluate

Table 3.3 Initial results. Summary of validation performance using pairs of SAR stacks for training and validation sets. These results are obtained using SegNet model

Train	Validation	Urban 21	Water 31	Veg. 41	Farm 51	Avg.	Overall
Montreal	Ottawa	83.9	82.9	74.6	89.8	80.4	80.86
	Quebec	91.7	91.1	78.2	78.2	87.1	
	Saskatoon	81.5	0.7	6.3	92.8	75.0	
	Toronto	92.9	97.5	61.8	83.8	89.6	
	Vancouver	92.4	96.1	35.1	64.9	72.2	
Ottawa	Montreal	87.7	88.7	84.8	74.0	83.7	76.3
	Quebec	90.6	89.1	84.4	60.5	84.9	
	Saskatoon	78.3	2.0	24.9	88.0	73.8	
	Toronto	92.6	95.9	59.3	79.3	88.0	
	Vancouver	91.7	91.8	95.4	89.5	51.1	
Quebec	Montreal	91.0	81.4	70.4	65.0	79.8	72.44
	Ottawa	77.3	71.0	66.9	75.6	71.5	
	Saskatoon	84.6	0.7	18.7	64.7	61.8	
	Toronto	92.7	94.4	29.3	70.7	83.9	
	Vancouver	86.4	92.1	25.1	41.2	65.2	
Saskatoon	Montreal	92.6	0.0	0.2	86.4	66.2	52.98
	Ottawa	88.3	0.0	0.2	90.2	38.9	
	Quebec	93.4	0.0	0.2	83.5	63.0	
	Toronto	93.4	0.0	0.3	81.4	69.2	
	Vancouver	79.0	13.7	4.5	82.8	27.6	
Toronto	Montreal	89.9	83.9	68.8	86.9	85.4	76.42
	Ottawa	78.9	70.3	59.3	95.7	71.8	
	Quebec	91.2	85.1	65.8	81.2	84.2	
	Saskatoon	82.5	0.1	9.1	91.3	74.8	
	Vancouver	93.0	95.3	17.3	92.4	65.9	
Vancouver	Montreal	84.9	80.5	88.8	0.1	61.8	60.48
	Ottawa	71.8	66.9	94.8	0.1	69.3	
	Quebec	79.1	85.8	88.6	0.1	70.3	
	Saskatoon	72.7	0.2	94.3	0.5	32.4	
	Toronto	87.3	95.0	82.1	0.2	68.6	

variables such as sampling size, we split the dataset into two fixed training and validation sets. We define the following split for training and validation:

- **Training Set**: SAR(Montreal, Quebec, Saskatoon, Vancouver)
- **Validation Set**: SAR(Ottawa, Toronto)

This is used to produce the final results from here on; unless otherwise specified. The first consideration for splitting the dataset is to make sure every region in a SAR image appears only in one set (training or validation). The next consideration is to make sure the total number of pixels per class divides as equally as possible in training and validation sets. Using this split, pixels with class C21 and C31 are divided almost equally; where, pixels with class C41 are divided with 9 percent error, and pixels with class C51 are divided with 5 percent error. This is the optimal combination of SAR images given the dataset D. Since Quebec, Saskatoon, and Vancouver SAR images are taken with lower quality beam mode, we select their split as the training set. This ensures more practical evaluation results assuming cheaper SAR images, and leaves Ottawa and Toronto stacks for the validation set.

3.3.2 Final Results

We implemented our experiments using *PyTorch*. Cross entropy loss is deployed for training pixel level classifier on the task of distinguishing four different land cover classes. The largest possible mini-batch size with respect to GPU memory constrains is 32 which is used for all experiments. The segmentation models are optimized using Stochastic Gradient Descent, with learning rate of 1^{-3}, momentum of 0.9, and weight decay of 5^{-4} as regularization hyperparameter.

Since all the four categories in sample images are important for segmentation, the Union over Intersection (UoI) reduces to average pixel accuracy as the evaluation metric. The accuracy is reported for correctly classified pixels of validation set in percentage, individually per class and in average.

We compare three sampling modes — **L**arge, **M**edium, **S**mall — with strides of 112, 152, and 192 respectively. The kernel size 224 is fixed for all experiments, mainly because larger input images require more GPU memory than available to us without reducing the mini-batch size considerably. Results using different sampling modes are presented in Table 3.4. Finally, we compare three deep CNN-based segmentation models using sampling mode **S**. The final results are presented in Table 3.5.

Table 3.4 Evaluation of segmentation results using different sampling modes. These results are obtained using SegNet model

Sampling mode	Urban 21	Water 31	Vegetation 41	Farm 51	Average
(**L**)arge	90.4	94.1	82.0	87.7	88.3
(**M**)edium	90.3	93.4	80.8	87.8	87.8
(**S**)mall	90.6	91.5	78.7	87.8	87.2

Table 3.5 Final results. Summary of land cover segmentation performance of our trained deep CNN models on the validation set using LCSAR dataset. These results are obtained by *Small* sampling mode

Model	Urban 21	Water 31	Vegetation 41	Farm 51	Average
DeconvNet	89.4	84.0	72.4	75.5	82.4
SegNet	**90.6**	91.5	78.7	**87.8**	87.2
PSPNet	90.2	**93.3**	**82.3**	86.8	**88.9**

3.4 Conclusion

In this work, we implemented three models to evaluate the performance of deep convolutional neural networks on land cover pixel-level classification using Synthetic Aperture Radar imagery. To alleviate the inherited noise in SAR imagery, we stacked several SAR images by averaging the pixel values. We also developed a sampling method to build training and validation sets from extremely large SAR images. Achieving near 90% average segmentation accuracy on validation set, our trained models promise a practical and robust solution for applications in automatic remote sensing and land cover classification. Future work can include incorporating image transformations with a smaller dataset to evaluate the efficiency of data augmentation in SAR imagery, experimenting with more complex networks e.g. Dual attention segmentation model (Fu *et al.*, 2018), and exploring the applications of automatic land cover segmentation in challenging tasks of remote sensing where the accuracy of non-deep learning models is not sufficient.

CONCLUSION AND RECOMMENDATIONS

In this dissertation, we demonstrated that deep convolutional neural networks are highly effective in achieving prominent accuracy when applied to custom datasets for various recognition tasks. By experimenting with different network architectures on localization and segmentation tasks using optical and SAR images respectively, we analyzed the efficacy of CNN models on non-standard data. Additionally, the performance of these networks surpass that of classical computer vision methods when applied on the same tasks. Moreover, we investigated the limitations of such models when working with very high resolution images and introduced a workaround to mitigate such limitations.

In the first part of this work, we focused on a localization task and validated the potential of CNN models by developing an accurate and low-cost alternative to manual cow tracking and space use in their tie-stalls. We trained several standard convolutional networks and proved that our presented models are able to accurately pinpoint the position of the hips and of the base of the neck of cows on the test data. Moreover, our results demonstrated that residual networks outperform other evaluated models by a considerable margin. Our developed models can be used to track dairy cows accurately and efficiently over long periods of time. Furthermore, the acquired tracking data can be compared with the pixel boundaries of the stall confines to determine the percentage of time and distance with which each cow spends outside the stall, representing biologically relevant information usable in a behavior research context, including the assessment of possible stall design improvements by evaluating how cows move within their surrounding environment.

In the second part of this work, we focused on a segmentation task and validated the potential of deep convnets to develop an accurate and efficient land cover pixel-level classification method using Synthetic Aperture Radar imagery. We averaged the pixel values of several stacked SAR images to reduce the inherited noise in SAR imagery and developed a sampling

method to efficiently train deep convnets using very high resolution SAR images. Moreover, we demonstrated that highest accuracy can be achieved by incorporating pyramid pooling module to capture both global context information and local details. Our developed models promise a practical and robust solution for applications in automatic remote sensing and land cover classification.

Overall, this study demonstrates the potential of deep learning in solving practical problems in various vision domains, from animal behavior research to remote sensing. It also highlights the importance of carefully selecting appropriate CNN architectures and methods for specific tasks and datasets. The results obtained can motivate the future work on localization and segmentation tasks having to deal with challenging data. By providing insights into the training and validation process of applying various CNN models to different image data, our study can help research developers by reducing the time and resources required for experimenting and achieving better results.

Finally, we recommend the following future work based on the conclusions above:

- Applying the presented methods to more challenging data: While the presented methods achieved high accuracy and performance on custom datasets, future work can explore the generalization of these models when applied on related but different data. For example, we evaluated our localization method on Holstein cows in tie-stall setup, however, its efficacy when applied on images from a different cattle breed with unalike body patterns (e.g. Jersey cows) hosted in an inconsistent environment (e.g. free-stall farm) is yet to be evaluated. Further experiments can involve applying different augmentation techniques, network architectures, and optimizations to ensure such models can handle more diverse data.
- Exploring the potential of transfer learning: To reduce the computational resources needed for training CNN models, one can adopt pre training such models. Future work can explore

the advantages of using pre-training on similar recognition tasks and fine-tuning the models on the custom datasets. For example, we trained and validated our segmentation method on SAR imagery only, however, these models can be pre-trained on optical satellite images and fine-tuned on SAR data which can potentially lead to faster training and higher accuracy.

- Optimizing the inference of the presented methods: While we promise high accuracy and performance with deep CNN models, the actual effectiveness of the presented methods when used inside a real-time tool is yet to be evaluated. Future work can explore techniques to improve the inference performance while reducing the size of these models. For example, using quantization and quantization-aware training can potentially reduce the size of trained CNN models and the inference time in trade-off of a small accuracy loss.

LIST OF REFERENCES

[Accessed: 2020-10-17]. What is SAR. Retrieved from: https://asf.alaska.edu/information/sar-information/what-is-sar/.

Abram, P. K., Parent, J.-P., Brodeur, J. & Boivin, G. (2016). Size-induced phenotypic reaction norms in a parasitoid wasp: an examination of life-history and behavioural traits. *Biological Journal of the Linnean Society*, 117(3), 620-632. doi: 10.1111/bij.12658.

AlShehhi, M., Damiani, E. & Wang, D. (2021). Toward Domain Adaptation for small data sets. *Internet of Things*, 16, 100458. doi: https://doi.org/10.1016/j.iot.2021.100458.

Anderson, N. G. (2014). Confort des vaches laitières - Dimensions des stalles de stabulation entravée. Guelph, Ontario, Canada: Ontario Ministry of Agriculture, Food and Rural Affairs. Retrieved from: http://www.omafra.gov.on.ca/french/livestock/dairy/facts/tiestaldim.htm.

Arora, S., Bhaskara, A., Ge, R. & Ma, T. (2013). Provable Bounds for Learning Some Deep Representations. *CoRR*, abs/1310.6343. Retrieved from: http://arxiv.org/abs/1310.6343.

Badrinarayanan, V., Kendall, A. & Cipolla, R. (2015). SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *CoRR*, abs/1511.00561. Retrieved from: http://arxiv.org/abs/1511.00561.

Bilen, H. & Vedaldi, A. (2015). Weakly Supervised Deep Detection Networks. *CoRR*, abs/1511.02853. Retrieved from: http://arxiv.org/abs/1511.02853.

Boyer, V., de Passillé, A., Adam, S. & Vasseur, E. (2021a). Making tiestalls more comfortable: II. Increasing chain length to improve the ease of movement of dairy cows. *Journal of Dairy Science*, 104(3), 3316-3326. doi: https://doi.org/10.3168/jds.2019-17666.

Boyer, V., Edwards, E., Guiso, M., Adam, S., Krawczel, P., de Passillé, A. & Vasseur, E. (2021b). Making tiestalls more comfortable: III. Providing additional lateral space to improve the resting capacity and comfort of dairy cows. *Journal of Dairy Science*, 104(3), 3327-3338. doi: https://doi.org/10.3168/jds.2019-17667.

Boykov, Y., Veksler, O. & Zabih, R. (2001). Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11), 1222-1239. doi: 10.1109/34.969114.

Buhmann, J. M., Malik, J. & Perona, P. (1999). Image recognition: Visual grouping, recognition, and learning. *Proceedings of the National Academy of Sciences*, 96(25), 14203–14204. doi: 10.1073/pnas.96.25.14203.

Chen, L., Papandreou, G., Schroff, F. & Adam, H. (2017). Rethinking Atrous Convolution for Semantic Image Segmentation. *CoRR*, abs/1706.05587. Retrieved from: http://arxiv.org/abs/1706.05587.

Coulibaly, P., Anctil, F. & Bobée, B. (2000). Daily reservoir inflow forecasting using artificial neural networks with stopped training approach. *Journal of Hydrology*, 230(3), 244-257. doi: https://doi.org/10.1016/S0022-1694(00)00214-6.

DFC-NFACC. (2009). Code of practice for the care and handling of dairy cattle. Ottawa, Ontario, Canada: Dairy Farmers of Canada and the National Farm Animal Care Council. Retrieved from: https://www.nfacc.ca/codes-of-practice/dairy-cattle.

Domingos, P. (2012). A Few Useful Things to Know About Machine Learning. *Commun. ACM*, 55(10), 78–87. doi: 10.1145/2347736.2347755.

Foret, P., Kleiner, A., Mobahi, H. & Neyshabur, B. (2020). Sharpness-Aware Minimization for Efficiently Improving Generalization. *CoRR*, abs/2010.01412. Retrieved from: https://arxiv.org/abs/2010.01412.

Fu, J., Liu, J., Tian, H., Fang, Z. & Lu, H. (2018). Dual Attention Network for Scene Segmentation. *CoRR*, abs/1809.02983. Retrieved from: http://arxiv.org/abs/1809.02983.

Gao, F., Huang, T., Sun, J., Wang, J., Hussain, A. & Yang, E. (2018). A new algorithm of SAR image target recognition based on improved deep convolutional neural network. *Computation*. doi: 10.1007/s12559-018-9563-z.

Ge, Z., Liu, S., Wang, F., Li, Z. & Sun, J. (2021). YOLOX: Exceeding YOLO Series in 2021. *CoRR*, abs/2107.08430. Retrieved from: https://arxiv.org/abs/2107.08430.

Girshick, R., Donahue, J., Darrell, T. & Malik, J. (2014). Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 580-587. doi: 10.1109/CVPR.2014.81.

Glorot, X. & Bengio, Y. (2010, 13–15 May). Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 9(Proceedings of Machine Learning Research), 249–256. Retrieved from: https://proceedings.mlr.press/v9/glorot10a.html.

Goodfellow, I., Bengio, Y. & Courville, A. (2016). *Deep Learning*. MIT Press.

Grosse, R. (2018). Lecture 11: Convolutional Networks. University of Toronto. Retrieved from: https://www.cs.toronto.edu/~rgrosse/courses/csc321_2018/.

Guilford, T., Åkesson, S., Gagliardo, A., Holland, R. A., Mouritsen, H., Muheim, R., Wiltschko, R., Wiltschko, W. & Bingman, V. P. (2011). Migratory navigation in birds: new opportunities in an era of fast-developing tracking technology. *Journal of Experimental Biology*, 214(22), 3705–3712. doi: 10.1242/jeb.051292.

Guzhva, O., Ardö, H., Nilsson, M., Herlin, A. & Tufvesson, L. (2018). Now You See Me: Convolutional Neural Network Based Tracker for Dairy Cows. *Frontiers in Robotics and AI*, 5, 107. doi: 10.3389/frobt.2018.00107.

He, K., Zhang, X., Ren, S. & Sun, J. (2014). Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *CoRR*, abs/1406.4729. Retrieved from: http://arxiv.org/abs/1406.4729.

He, K., Zhang, X., Ren, S. & Sun, J. (2015). Deep Residual Learning for Image Recognition. *CoRR*, abs/1512.03385. Retrieved from: http://arxiv.org/abs/1512.03385.

He, K., Gkioxari, G., Dollár, P. & Girshick, R. (2017). Mask R-CNN. *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2980-2988. doi: 10.1109/ICCV.2017.322.

Huang, G., Liu, Z. & Weinberger, K. Q. (2016). Densely Connected Convolutional Networks. *CoRR*, abs/1608.06993. Retrieved from: http://arxiv.org/abs/1608.06993.

Huang, Z., Dumitru, C. O., Pan, Z., Lei, B. & Datcu, M. (2020). Classification of Large-Scale High-Resolution SAR Images With Deep Transfer Learning. *IEEE Geoscience and Remote Sensing Letters*, 1–5. doi: 10.1109/lgrs.2020.2965558.

Ibrahim, J. G., Chen, M. H. & Lipsitz, S. R. (1999). Monte carlo EM for missing covariates in parametric regression models. *Biometrics*, 55(2), 591–596. doi: 10.1111/j.0006-341X.1999.00591.x.

Ioffe, S. & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *CoRR*, abs/1502.03167. Retrieved from: http://arxiv.org/abs/1502.03167.

Johnson, S. & Everingham, M. (2010). Clustered Pose and Nonlinear Appearance Models for Human Pose Estimation. *Proceedings of the British Machine Vision Conference*.

Johnson, S. & Everingham, M. (2011). Learning effective human pose estimation from inaccurate annotation. *CVPR 2011*, pp. 1465-1472. doi: 10.1109/CVPR.2011.5995318.

Krizhevsky, A. (2009). *Learning multiple layers of features from tiny images*. (Master's thesis, Department of Computer Science, University of Toronto).

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, (NIPS'12), 1097–1105.

Lapini, A., Pettinato, S., Santi, E., Paloscia, S., Fontanelli, G. & Garzelli, A. (2020). Comparison of Machine Learning Methods Applied to SAR Images for Forest Classification in Mediterranean Areas. *Remote Sensing*, 12(3), 369. doi: 10.3390/rs12030369.

Lecun, Y., Bottou, L., Bengio, Y. & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278-2324. doi: 10.1109/5.726791.

LeCun, Y. & Cortes, C. (2010). MNIST handwritten digit database. Retrieved from: http://yann.lecun.com/exdb/mnist/.

Li, P., Wang, D., Wang, L. & Lu, H. (2018). Deep visual tracking: Review and experimental comparison. *Pattern Recognition*, 76, 323 - 338. doi: https://doi.org/10.1016/j.patcog.2017.11.007.

Lin, L. I.-K. (1989). A Concordance Correlation Coefficient to Evaluate Reproducibility. *Biometrics*, 45(1), 255–268. Retrieved from: http://www.jstor.org/stable/2532051.

Lin, M., Chen, Q. & Yan, S. (2014). Network In Network. *CoRR*, abs/1312.4400.

Liu, L., Ouyang, W., Wang, X., Fieguth, P. W., Chen, J., Liu, X. & Pietikäinen, M. (2018). Deep Learning for Generic Object Detection: A Survey. *CoRR*, abs/1809.02165. Retrieved from: http://arxiv.org/abs/1809.02165.

Liu, X., He, J., Yao, Y., Zhang, J., Liang, H., Wang, H. & Hong, Y. (2017). Classifying urban land use by integrating remote sensing and social media data. *International Journal of Geographical Information Science*, 31(8), 1675-1696. doi: 10.1080/13658816.2017.1324976.

Mahmud, M. S., Zahid, A., Das, A. K., Muzammil, M. & Khan, M. U. (2021). A systematic literature review on deep learning applications for precision cattle farming. *Computers and Electronics in Agriculture*, 187, 106313. doi: https://doi.org/10.1016/j.compag.2021.106313.

Mangasarian, O. & Musicant, D. (2000). Robust linear and support vector regression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(9), 950-955. doi: 10.1109/34.877518.

McPherson, S. & Vasseur, E. (2021). Making tiestalls more comfortable: IV. Increasing stall bed length and decreasing manger wall height to heal injuries and increase lying time in dairy cows housed in deep-bedded tiestalls. *Journal of Dairy Science*, 104(3), 3339-3352. doi: https://doi.org/10.3168/jds.2019-17668.

Minaee, S. & Wang, Y. (2019). An ADMM Approach to Masked Signal Decomposition Using Subspace Representation. *IEEE Transactions on Image Processing*, 28(7), 3192-3204. doi: 10.1109/TIP.2019.2894966.

Minaee, S., Boykov, Y., Porikli, F., Plaza, A., Kehtarnavaz, N. & Terzopoulos, D. (2020). Image Segmentation Using Deep Learning: A Survey. *CoRR*, abs/2001.05566. Retrieved from: https://arxiv.org/abs/2001.05566.

Mitchell, T. M. (1997). *Machine Learning* (ed. 1). New York, NY, USA: McGraw-Hill, Inc.

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. The MIT Press.

Nair, V. & Hinton, G. E. (2010). Rectified Linear Units Improve Restricted Boltzmann Machines. *Proceedings of the 27th International Conference on International Conference on Machine Learning*, (ICML'10), 807–814.

Noh, H., Hong, S. & Han, B. (2015). Learning Deconvolution Network for Semantic Segmentation. *CoRR*, abs/1505.04366. Retrieved from: http://arxiv.org/abs/1505.04366.

Plath, N., Toussaint, M. & Nakajima, S. (2009). Multi-Class Image Segmentation Using Conditional Random Fields and Global Classification. *Proceedings of the 26th Annual International Conference on Machine Learning*, (ICML '09), 817–824. doi: 10.1145/1553374.1553479.

Redmon, J., Divvala, S., Girshick, R. & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779-788. doi: 10.1109/CVPR.2016.91.

Ren, S., He, K., Girshick, R. & Sun, J. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, (NIPS'15), 91–99.

Rieder, M. & Verbeet, R. (2019, 09). Robot-Human-Learning for Robotic Picking Processes. doi: 10.15480/882.2466.

Roh, Y., Heo, G. & Whang, S. E. (2021). A Survey on Data Collection for Machine Learning: A Big Data - AI Integration Perspective. *IEEE Transactions on Knowledge and Data Engineering*, 33(4), 1328-1347. doi: 10.1109/TKDE.2019.2946162.

Ronneberger, O., Fischer, P. & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pp. 234–241.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C. & Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3), 211-252. doi: 10.1007/s11263-015-0816-y.

Sapp, B. & Taskar, B. (2013). MODEC: Multimodal Decomposable Models for Human Pose Estimation. *In Proc. CVPR*.

Sathya, R. & Abraham, A. (2013). Comparison of Supervised and Unsupervised Learning Algorithms for Pattern Classification. 2(2). doi: 10.14569/IJARAI.2013.020206.

Seferbekov, S. S., Iglovikov, V. I., Buslaev, A. V. & Shvets, A. A. (2018). Feature Pyramid Network for Multi-Class Land Segmentation. *CoRR*, abs/1806.03510. Retrieved from: http://arxiv.org/abs/1806.03510.

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R. & LeCun, Y. (2014). OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. Retrieved from: http://arxiv.org/abs/1312.6229.

Shepley, E., Berthelot, M. & Vasseur, E. (2017). Validation of the Ability of a 3D Pedometer to Accurately Determine the Number of Steps Taken by Dairy Cows When Housed in Tie-Stalls. *Agriculture*, 7(7), 53. doi: 10.3390/agriculture7070053.

Simonyan, K. & Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Retrieved from: http://arxiv.org/abs/1409.1556.

Srivastava, N., Hinton, G., Krizhevsky, A. & Salakhutdinov, R. (2014). *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*. Retrieved from: http://jmlr.org/papers/volume15/srivastava14a.old/srivastava14a.pdf.

St John, J., Rushen, J., Adam, S. & Vasseur, E. (2021). Making tiestalls more comfortable: I. Adjusting tie-rail height and forward position to improve dairy cows' ability to rise and lie down. *Journal of Dairy Science*, 104(3), 3304-3315. doi: https://doi.org/10.3168/jds.2019-17665.

Sutskever, I., Martens, J., Dahl, G. & Hinton, G. (2013). On the Importance of Initialization and Momentum in Deep Learning. *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, (ICML'13), III–1139–III–1147.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. & Rabinovich, A. (2015a). Going deeper with convolutions. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1-9. doi: 10.1109/CVPR.2015.7298594.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. & Wojna, Z. (2015b). Rethinking the Inception Architecture for Computer Vision. *CoRR*, abs/1512.00567. Retrieved from: http://arxiv.org/abs/1512.00567.

Szeliski, R. (2010). *Computer Vision: Algorithms and Applications* (ed. 1st). Berlin, Heidelberg: Springer-Verlag.

Toshev, A. & Szegedy, C. (2014). DeepPose: Human Pose Estimation via Deep Neural Networks. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1653-1660. doi: 10.1109/CVPR.2014.214.

Tweedale, J. (2019). An Application of Transfer Learning for Maritime Vision Processing Using Machine Learning (pp. 87-97). doi: 10.1007/978-3-319-92028-3_9.

Valacta. (2014). THE BARN; A SOURCE OF COMFORT: Practical guide to evaluating and improving comfort in the barn. Ste-Anne-de-Bellevue, Quebec, Canada: Valacta. Retrieved from: https://www.valacta.com/en-ca/library/practical-guide-evaluating-improving-comfort-in-barn.

Vasseur, E. (2017). ANIMAL BEHAVIOR AND WELL-BEING SYMPOSIUM: Optimizing outcome measures of welfare in dairy cattle assessment1. *Journal of Animal Science*, 95(3), 1365-1371. doi: 10.2527/jas.2016.0880.

Wissner-Gross, A. [Accessed: 2021-7-31]. (2016). Datasets Over Algorithms. Retrieved from: https://www.edge.org/response-detail/26587.

Wu, W., Li, H., Zhang, L., Li, X. & Guo, H. (2018). High-Resolution PolSAR Scene Classification With Pretrained Deep Convnets and Manifold Polarimetric Parameters. *IEEE Transactions on Geoscience and Remote Sensing*, 56(10), 6159-6168.

Yan, Z., Liang, J., Pan, W., Li, J. & Zhang, C. (2017). Weakly- and Semi-Supervised Object Detection with Expectation-Maximization Algorithm. *CoRR*, abs/1702.08740. Retrieved from: http://arxiv.org/abs/1702.08740.

You, K., Long, M., Jordan, M. I. & Jordan, M. I. (2019). How Does Learning Rate Decay Help Modern Neural Networks. *arXiv: Learning*.

Yumus, D. & Ozkazanc, Y. (2019). Land Cover Classification for Synthetic Aperture Radar Imagery by Using Unsupervised Methods. *2019 9th International Conference on Recent Advances in Space Technologies (RAST)*, pp. 435-440.

Zacharaki, E. I., Wang, S., Chawla, S., Soo Yoo, D., Wolf, R., Melhem, E. R. & Davatzikos, C. (2009). Classification of brain tumor type and grade using MRI texture and shape in a machine learning scheme. *Magnetic Resonance in Medicine*, 62(6), 1609-1618. doi: 10.1002/mrm.22147.

Zambelis, A., Saadati, M., Dallago, G., Stecko, P., Boyer, V., Parent, J.-P., Pedersoli, M. & Vasseur, E. (2021). Automation of video-based location tracking tool for dairy cows in their housing stalls using deep learning. *Smart Agricultural Technology*, 1, 100015. doi: https://doi.org/10.1016/j.atech.2021.100015.

Zeiler, M. D. & Fergus, R. (2014). Visualizing and Understanding Convolutional Networks. *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I*, 8689(Lecture Notes in Computer Science), 818–833. doi: 10.1007/978-3-319-10590-1_53.

Zhang, C., Sargent, I., Pan, X., Li, H., Gardiner, A., Hare, J. & Atkinson, P. M. (2019). Joint Deep Learning for land cover and land use classification. *Remote Sensing of Environment*, 221, 173 - 187. doi: https://doi.org/10.1016/j.rse.2018.11.014.

Zhao, H., Shi, J., Qi, X., Wang, X. & Jia, J. (2016). Pyramid Scene Parsing Network. *CoRR*, abs/1612.01105. Retrieved from: http://arxiv.org/abs/1612.01105.

Zheng, C., Wu, W., Yang, T., Zhu, S., Chen, C., Liu, R., Shen, J., Kehtarnavaz, N. & Shah, M. (2020). Deep Learning-Based Human Pose Estimation: A Survey. *CoRR*, abs/2012.13392. Retrieved from: https://arxiv.org/abs/2012.13392.

Zhou, Z.-H. (2017). A brief introduction to weakly supervised learning. *National Science Review*, 5(1), 44-53. doi: 10.1093/nsr/nwx106.

Zilly, J. G., Srivastava, R. K., Koutník, J. & Schmidhuber, J. (2017, 06–11 Aug). Recurrent Highway Networks. *Proceedings of the 34th International Conference on Machine Learning*, 70(Proceedings of Machine Learning Research), 4189–4198. Retrieved from: https://proceedings.mlr.press/v70/zilly17a.html.

Zou, Z., Shi, Z., Guo, Y. & Ye, J. (2019). Object Detection in 20 Years: A Survey. *CoRR*, abs/1905.05055. Retrieved from: http://arxiv.org/abs/1905.05055.