Deep Multi-Label Learning

by

Jan André Marais

Thesis presented in partial fulfilment of the requirements for the degree of Master of Commerce (Mathematical Statistics) in the Faculty of Economic and Management Sciences at Stellenbosch University

Supervisor: Dr. S. Bierman

December 2017

The financial assistance of the National Research Foundation (NRF) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the author and are not necessarily to be attributed to the NRF.

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: .....

Abstract

Deep Multi-Label Learning

J. A. Marais

Thesis: MCom (Mathematical Statistics)

December 2017

English abstract

Uittreksel

Diep Multi-Etiket Leer

("Deep Multi-Label Learning")

J. A. Marais

Tesis: MCom (Wiskundige Statistiek)

Desember 2017

Afrikaans abstract

Acknowledgements

I would like to express my sincere gratitude to the following people and organisations ...

Contents

List of Figures

List of Tables

Nomenclature

Constants

g =	$9.81\,\mathrm{m/s^2}$	

Variables

Re_D	Reynolds number (diameter)	[ ]
x	Coordinate	[m]
$\ddot{x}$	Acceleration	[$\mathrm{m/s^2}$]
θ	Rotation angle	[rad]
τ	Moment	[N·m]

Vectors and Tensors

$\overrightarrow{v}$	Physical vector, see equation ...

Subscripts

a	Adiabatic
a	Coordinate

Chapter 1

Introduction

1.1 Motivation

The motivation for this thesis is two-fold:

1. Multi-label learning is a highly relevant field in machine learning and statistics because of its wide range of applications. To varying degrees of success, it has been applied to problems in text categorisation, multimedia, biology, chemical data analysis, social network mining and e-learning among others (review list). Despite the rapid increase in multi-label learning literature (see Figure 1.1), the field is nowhere near the maturity level of its single-label counterpart. Consistently effective and efficient multi-label learning strategies are scarce. Researchers in the field have not yet reached consensus on many of the aspects when learning from multi-labelled data such as how to handle dependent labels or how to apply dimension reduction techniques. The field can gain from an up-to-date review of the literature (latest thorough review in 2014), more statistical perspectives on some of the challenges, additional benchmark datasets and quality empirical evaluations of the theory.

2. Deforestation is a massive global problem[1]. It contributes to reduced biodiversity, habitat loss, climate change and other devastating effects. It is said that the world loses an area of forest the size of 48 football fields per minute and the area most affected is in the Amazon basin (cite Kaggle). This problem can be fought more effectively by governments and local stakeholders if better data about the location of deforestation and human invasion on forests are continuously available to them - an ideal task for machine learning! Planet[2] and SCCON[3] constructed a dataset

[1] I saw in another paper that the rate of decline is decreasing (cite)

[2] Designer and builder of the world's largest constellation of Earth-imaging satellites - www.planet.com

[3] Remote sensing experts - www.sccon.com.br/eng

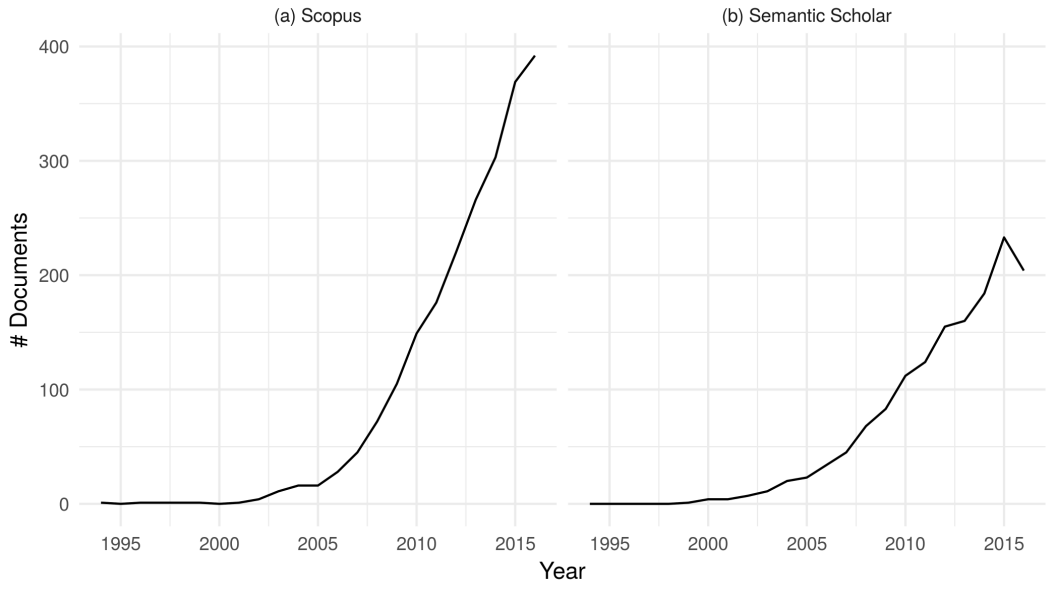

Figure 1.1: Line graphs illustrating the rise in multi-label learning publications per year for two databases. The database searches were done on 24-03-2017. The searches were not identical since they were limited to the search features of the databases. (a) The search on Scopus (cite) was for all documents (conference papers, articles, conference, articles in press, reviews, book chapters and books) in any subject area with either the words *multi-label* or *multilabel* and either the words *learning* or *classification* found in either their titles, abstracts or keywords. (b) The search on Semantic Scholar was based on machine learning principles and thus automatically decides which research documents are relevant to a specific search query. The query used was *multilabel multi-label learning classification*. The search only returns research in the computer science and neuroscience fields of study. More technical details can be found on the respective engine's websites.

of labelled satellite images taken of the Amazon basin and released it as part of a competition on Kaggle[4], challenging competitors to build algorithms that can automatically label these images with atmospheric conditions and various classes of land use/cover[5]. Resulting algorithms will help the global community better understand where, how, and why deforestation happens all over the world - and ultimately how to respond.

[4]Runs programming contests to crowd source machine learning solutions - www.kaggle.com

[5]Land cover indicates the physical land type such as forest or open water whereas land use documents how people are using the land.

1.2 Thesis Objectives

This thesis works towards building a multi-label learner that can label satellite images of the Amazon as accurately as possible. The method thought best to achieve this is to:

1. Identify the most important and latest developments in the multi-label literature, as well as in satellite image classification.
2. Provide an extensive review and discussion of these methods and how they compare to each other.
3. Empirically evaluate and compare them on the satellite image data in order to find the best strategies for our labeling task.

The main focus points for this thesis are:

- Label dependence - What is it; can it be used to improve a learner's accuracy and/or complexity; and when?
- Resampling - What are effective resampling techniques for multi-label data to deal with the class imbalance problem and to estimate errors and standard deviations?
- Dimension reduction - How to reduce the number of dimensions of the input and output space in order to build more effective and efficient algorithms.

  This is still a rough list and should be updated as progress is made with the following chapters.

  Make sure this is how an introduction is allowed to look.

  should I have a section on contributions?

1.3 Data

This section covers an initial introduction to the data. The elements of the data important to know before moving on will be discussed here and the rest will be addressed throughout the thesis, as it becomes relevant to the discussion.

1.3.1 Image Format

The data for this task comes from a set of images (also referred to as chips). Each chip is a small excerpt from a larger image of a specific scene in the Amazon taken by satellites. The chip size in pixels is 256×256, representing roughly 90 hectares of land, and is taken from a larger scene of 6600×2200 pixels. All of the satellite images were taken between January 1, 2016 and

February 1, 2017. The format of these images differ from the standard image format. Each image contains four bands of data: red (R), green (G), blue (B) and near infrared (NIR), where the standard format images usually only contain R, G and B. The additional NIR colour channel is common in remote sensing[6] applications and supposedly allows for clear distinction between water and vegetation in satellite images, for example.

Another difference between these images and the usual format is that these have pixel intensities in 16-bit digital number format as opposed to the usual 8-bit of standard RGB images. This allows the colours in the images to have a much higher range since 16-bit pixel intensities have 65536 (2^16) levels, compared to 256 levels of 8-bit images. This becomes useful, for example, to distinguish between different levels of darkness in an image. Thus each chip can be represented by a vector of size 262144 ($256 \times 256 \times 4$). This might prove to require to much computational power but strategies to reduce the size of such a vector exist, *e.g.* filtering or resizing of the image.

1.3.2 Collection and Labelling of the Images

The image collection was created by first specifying a "wish list" of scenes containing the phenomena the creators wanted to be included and also a rough estimate of the number of such scenes that are necessary for a sufficient representation in the final collection. This set of scenes was then searched for manually on Planet Explorer[7]. From these scenes the 4-band chips were created. The chips were labelled manually by crowd sourcing. The utmost care was taken to get a large and well-labelled dataset, but that does not mean the labels all correspond to the ground-truth, *i.e.* the data will contain some inherent error. The creators believe that the data has a reasonable high signal to noise ratio.

Note, the training and test splits was determined by the Kaggle competition creators. The training chips are labeled but the test chips are not. Predicted labels for the test chips can be submitted for Kaggle to evaluate in terms of an evaluation metric. This setup prevents competitors from using the test chips for training a classifier. There are 40479 training chips and 40669 test chips.

1.3.3 Class Labels

The class labels for the images can be broken into three groups: atmospheric conditions, common land cover/use phenomena and rare land cover/use phenomena. Each chip will have one atmospheric label and zero or more common

[6]The use of satellite- or aircraft-based sensor technologies to detect and classify objects on Earth [https://en.wikipedia.org/wiki/Remote_sensing].

[7]A web based interactive map of Earth consisting of satellite images, similar to Google Earth - www.planet.com/explorer

and rare labels. Chips that are labeled as cloudy should have no other labels, but there are some labeling errors.

The atmospheric condition labels are: *clear*, *haze*, *partly cloudy* and *cloudy*. They are relevant to a chip when:

- clear: there are no evidence of clouds.
- haze: clouds are visible but they are not so opaque as to obscure the ground.
- partly cloudy: scenes show opaque cloud cover over any portion of the image but the land cover/use phenomena are still visible.
- cloudy: 90% of the image is obscured with opaque cloud cover.

Examples of chips with atmospheric labels can be found in Figure 1.2. Each chip should only have one atmospheric label and therefore this classifying task simplifies to a multiclass problem. This allows for the option to break up the labeling task of all the labels into two tasks: a multiclass classification problem for the atmospheric labels and a multi-label classification problem for the land cover/use labels. This approach might save some computational time and give extra information to the multi-label learners for classifying the land cover/use labels. We will experiment with these approaches in Chapter ??.

The common land cover/use labels are: *primary*, *agriculture*, *water*, *habitation*, *road*, *cultivation* and *bare ground*. They are relevant to a chip when:

- primary: it is primarily consisting of rain forest (virgin forest), *i.e.* dense tree cover.
- agriculture: it contains any land cleared of trees that is being used for agriculture or range land.
- water: it contains any one of the following: rivers, reservoirs, or oxbow lakes.
- habitation: it contains human homes or buildings.
- road: it contains any type of road.
- cultivation: it shows signs of smaller-scale/informally cleared land for farming.
- bare ground: it contains naturally (not the caused by humans) occurring tree-free areas.

Examples of chips with common land cover/use labels are found in Figure 1.3. According to the competition page on Kaggle, small, single-dwelling habitations are often difficult to spot but usually appear as clumps of a few pixels that are bright white. Roads sometimes look very similar to rivers and therefore these two labels might be noisy. The NIR band might give a classifier additional information to help distinguish between the two. Cultivation is a subset of agriculture and is normally found near smaller villages, along major rivers or at the outskirts of agricultural areas. It typically covers very small areas.

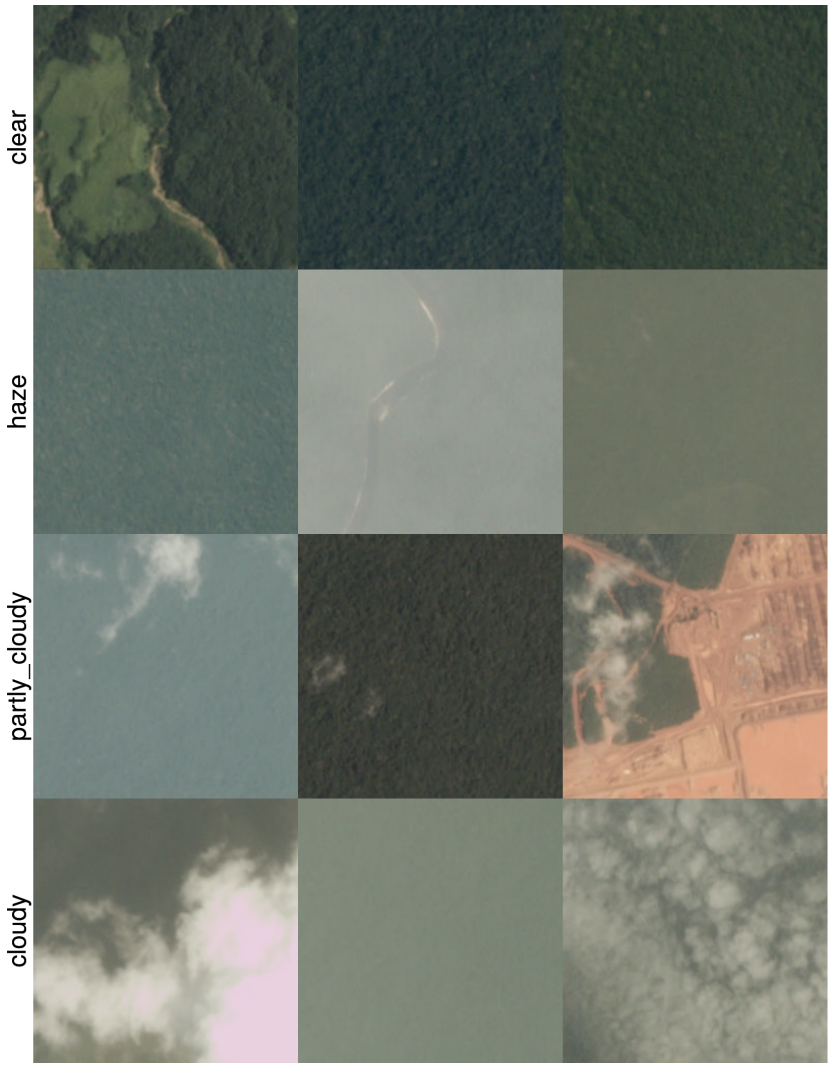

Figure 1.2: Examples of chips with atmospheric labels. These (along with all the other chips plotted throughout the thesis) are the JPEG conversions of the original 4-band, 16-bit images.

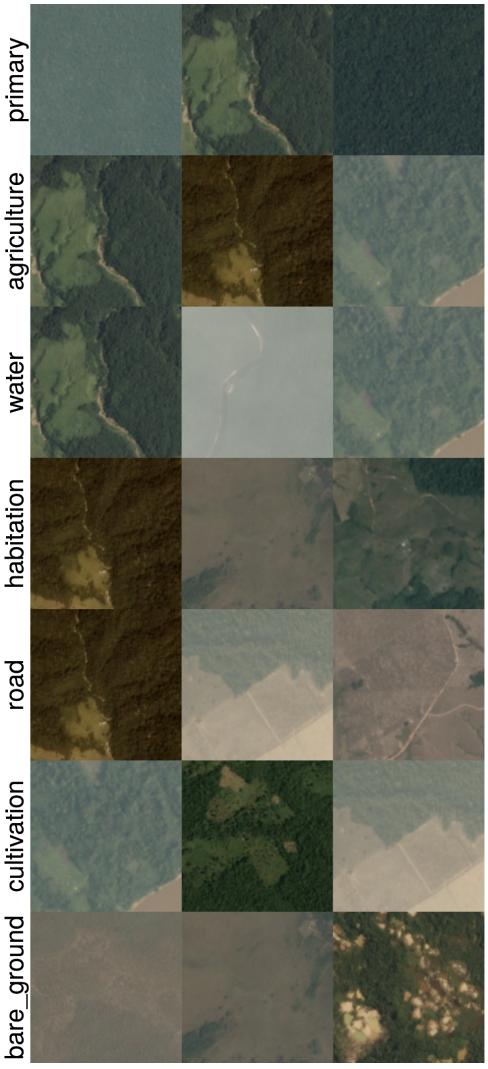

Figure 1.3: Examples of chips with common land cover/use labels.

The less common land cover/use labels are: *slash and burn*, *selective logging*, *blooming*, *conventional mine*, *artisinal mine* and *blow down*. Chips are tagged with these labels when:

- slash and burn: there are signs of the farming method that involves the cutting and burning of the forest to create a field. These look like cultivation patches with black or dark brown areas.
- selective logging: winding dirt roads are present adjacent to bare brown patches in otherwise primary rain forest. Selective logging is the practice of selectively removing high values tree species from the rainforest.
- blooming: there are signs of trees flowering. Blooming is a natural phenomena where particular species of flowering trees bloom, fruit and flower at the same time. These trees are quite big and the phenomena

can be seen in the chips. They usually appear as white dots.

- conventional mine: it contains signs of large-scale legal mining operations.
- artisinal mine: it contains signs of small-scale (sometimes illegal) mining operations.
- blow down: there are signs of trees uprooted or broken by wind. High speed winds (~160km/h) in the Amazon are generated when the cold dry air from the Andes settles on top of the warm moist air in the rainforest and then sinks down with incredible force, toppling larger rainforest trees. These open areas are visible from space.

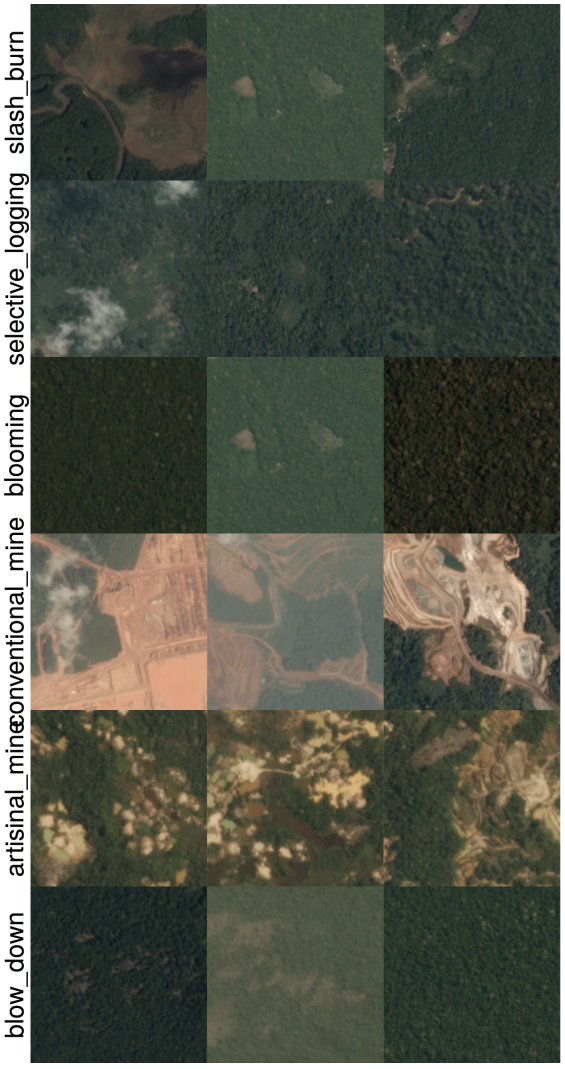

Figure 1.4: Examples of chips with less common land cover/use labels.

Examples of chips with these less common land cover/use labels are given in Figure 1.4. These labels are more challenging to identify in the chips and

since they also appear less frequently, it might be difficult for the classifier to learn these labels.

1.4 Code and Reproducibility

Got this header from Arnu's thesis - not sure if I will include this. But it may be appropriate to indicate here where to find the code for the thesis, why it is important, etc.

1.5 Important Concepts and Terminology

Briefly introduce the important concepts to be grasped in order to follow the main thread of the thesis. It seems reasonable to introduce the problem of supervised learning here. The rest still needs to be decided on.

1.6 Outline

The structure of this thesis is built to mimic the workflow of fitting a supervised learning model to data. At each step, the relevant literature will be critically reviewed and discussed. Thereafter the proposed and recommended strategies will be applied to our data to see if the results match the literature and to find the best methods for our application.

In any supervised learning problem, it is essential to become familiar with the data and the task at hand before moving on to the training process. The background information of the data has already been discussed. In Chapter ?? the unique proppertes of multi-label data will be investigated and what steps are recommended to follow for data with certain properties. It is very important to clearly define the objective of the supervised learning task. For this thesis, prediction accuracy is more important than making inferences on the data (models also giving insight into the data is a bonus). The evaluation metric for our task will be introduced and discussed in Chapter ?? along with other ways to evaluate multi-label classifiers.

Other things still to mention:

- basic ML strategies
- ML resampling strategies for class imbalance and error estimates
- orders of complexity
- label dependence
- input space reduction

- output space reduction
- final predictions and evaluations (maybe MDS of actual vs predicted)
- might want to include short history/timeline of ML
- might want to do a meta analysis of the literature on main topics

Chapter 2

Image Classification with Neural Networks

2.1 Introduction

There are three main tasks in computer vision (CV), namely: image classification, object detection and image segmentation. Traditional image classification is the task of assigning one label from a fixed set of categories to an input image. More recently the task has been generalised to assigning multiple labels to an input image, *i.e.* multi-label classification (MLC). First, we will only look at the single label case.

Image classification is the core of computer vision tasks and probably the most explored since it has a large variety of practical applications. It can be shown that the other two CV tasks, detection and segmentation, can be reduced to classification. Classification will be the main theme of this thesis but we will have a look at segmentation and detection later on.

Instead of hard coding rules on how to classify images into an image classification model, it can learn to classify images by seeing many examples of images and its corresponding labels. In this way it learns the visual appearance of each class. This is sometimes referred to as a data-driven approach. A very intuitive approach to image classification (and supervised learning in general) is called the nearest neighbour approach.

2.2 Nearest-Neigbours

Although this approach is rarely used in practice, this description helps with the understanding of the image classification problem. The nearest neighbour classifier will take a test image, compare it to every single one of the training images, and predict its label to be the label of the closest training image. This leaves the question of how to measure the similarity between images.

An image is a grid of many small, square cells of different colors. These cells are known as pixels and one pixel represents one color. A grayscale image, 32 pixels wide and 32 pixels long, can be represented by a 32×32 matrix of integers, where each integer represents the 'brightness' (intensity) of each pixel. These integers are usually in $[0, 255]$, such that the greater the integer the brighter the pixel, *i.e.* a pixel with intensity 0 is totally black and a pixel with intensity 255 is totally white. Note that a color image consists of 3 color bands, red, green and blue (RGB), *i.e.* the color of one pixel is determined by 3 integers each representing the intensity of the color red, green and blue, respectively.

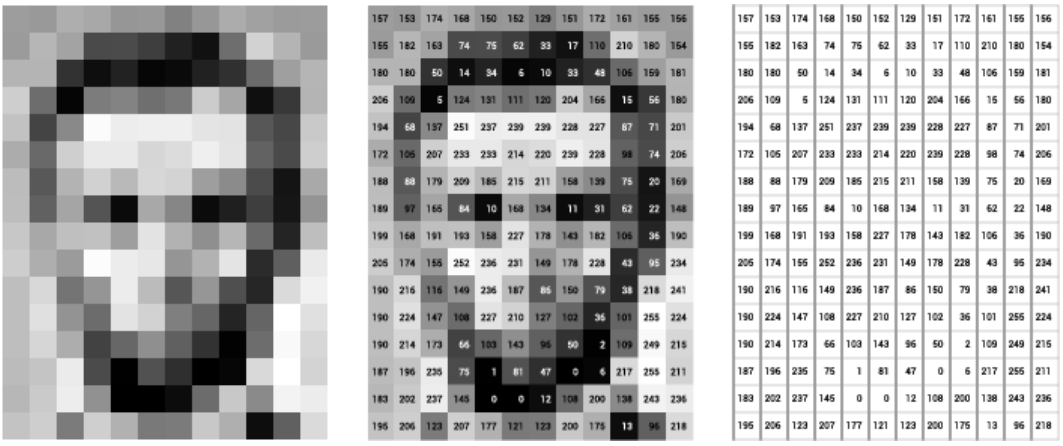

Figure 2.1: Greyscale intensities. http://ai.stanford.edu/~syyeung/cvweb/tutorial1.html

The (dis)similarity between two images can now be measured pixel by pixel. It is possible to represent the grayscale image mentioned above in a vector of length 32×32. Suppose a grayscale Image 1 is flattened out to be represented by the vector $\boldsymbol{I}_1 = \{I_{11}, I_{12}, \ldots, I_{1p}\}$ and similary, Image 2 by $\boldsymbol{I}_2$, where $p = 32 \times 32$. Then the dissimilarity between Image 1 and Image 2 can be calculated by the L_1-distance:

$$d_1(\boldsymbol{I}_1, \boldsymbol{I}_2) = \sum_{j=1}^{p} |I_{1j} - I_{2j}|.$$

Now, suppose we want to predict the label of an test image a, then the nearest neighbour approach would assign the label of train image b^* to test image a if:

$$b^* = \arg\min_b d_1(\boldsymbol{I}_a, \boldsymbol{I}_b),$$

for $b = 1, 2, /dots, N$, where N is the number of training images. Of course there are other ways of measuring the dissimilarity between images. Another example would be to use the L_2-distance:

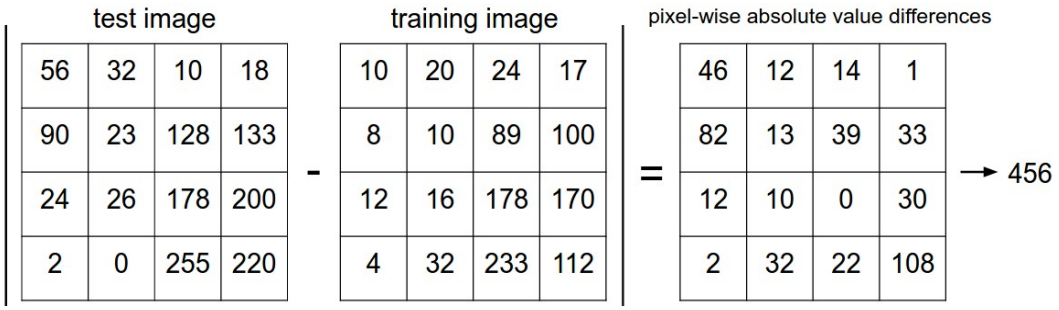

Figure 2.2: Pixelwise difference

$$d_2(\boldsymbol{I}_1, \boldsymbol{I}_2) = \sqrt{\sum_{j=1}^{p} (I_{1j} - I_{2j})^2}.$$

The chosen metric depends on the use case.

The nearest neighbour approach can be generalised to use more than 1 nearest neighbour when predicting the label of a test image. This approach is called the k-Nearest Neighbours (k-NN). The only difference is that, you now search for the k (instead of just 1) images with the smallest dissimilarity with the test image and then combine the labels of these k images, either through averaging or majority voting, to predict the label of the test image. Choosing the right value of k is important and is usually done by cross-validation. See Hastie ref.

The advantage of using k-NN is that it is simple and requires no time to train. Unfortunately, when it comes to test time, the algorithm needs to calculcate the distance between the test image and all the other images in the training set, which is computationally very expensive. Also in [Haste ref], they show that k-NN suffers severely from the *curse of dimensionality* and that it is mostly only useful to classify lower dimensional objects. Images are very high-dimensional objects.

The dissimilarity measures discussed above are actually proven to be very poor in discriminating between images in an image classification problem. Images that are nearby in terms of the L_1 and L_2 distances are much more of a function of the general color distribution of the images, or the type of background rather than their semantic identity. Refer to the t-SNE figure.

2.3 Linear Classification

The following simple approach to image classification naturally extends to neural networks and convolutional neural networks and is therefore very important to comprehend. This approach has to major components: a score function and a loss function. The score function maps raw data (*e.g.* an image) to a set of

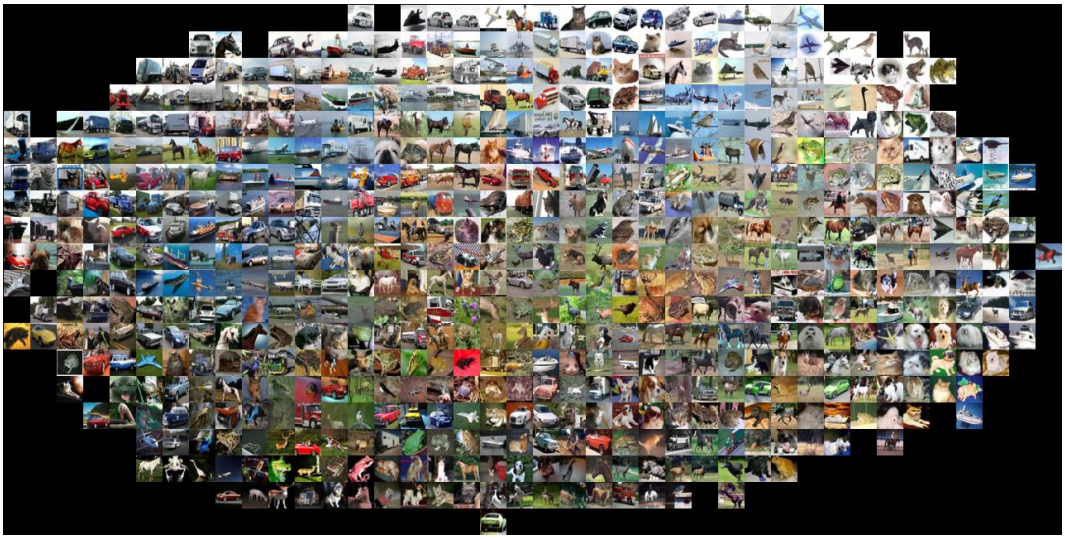

Figure 2.3: Caption

class scores, and a loss function quantifies the agreement between the predicted class scores and the actual ground truth labels associated with the raw data. This approach can then be described as an optimization problem in which the minimisation of the loss function with respect to the parameters of the score function is the main goal.

Some notation is needed to formally define this approach. Suppose we have N training images $\boldsymbol{x}_i \in \mathbb{R}^p$ each associated with a label $y_i \in \{1, 2, \ldots, K\}$, where $i = 1, 2, \ldots, N$ and K is the number of possible categories an image can belong to and p the number of pixels of each image. The score function is then defined as the function f that maps the raw image pixels to class scores:

$$f : \mathbb{R}^p \to \mathbb{R}^K.$$

The simplest possible score function is a linear mapping:

$$f(\boldsymbol{x}_i, W, b) = W\boldsymbol{x}_i + \boldsymbol{b}.$$

In the above equation, Image i is flattened out to be represented by a p-dimensional vector. The paramters of f are the matrix $W : K \times p$ and the vector $\boldsymbol{b}$, often called the weights and biases, respectively. These terms are comparable to the coefficient and constant terms in a statistical linear model and thus should not be confused with bias in the statistical sense.

We assume the pairs $(\boldsymbol{x}_i, y_i)$ to be fixed, but we do have control over the W and $\boldsymbol{b}$ terms. Our goal will be to set these in such a way so that the computed class scores for each image in the training set match the associated ground truth label as close as possible. What we have described thus far is very similar to the approach taken by convolutional neural networks, but instead the function,

f, which maps the raw pixels to class scores, is much more complicated with plenty more parameters to tune.

Notice that this score function determines the score for each class as a weighted sum of the pixel values across all 3 of its color bands. We would imagine that a linear classifier trained to classify, say, ships would have a weight matrix that assigns heavier weights to blue pixels on the sides of an image, which loosely corresponds to water.

If we picture the images as points in a high-dimensional space, f is a hyperplane, W determines the angle of the hyperplane and $\boldsymbol{b}$ translates the hyperplane through the space. Another interpretation of this linear classifier is that each row of the weight matrix is a so-called template for the corresponding class. The linear classifier matches the input image with each of the class templates in W by calculating a dot product. A high class score would translate to a higher similarity between the input image and the class template. This interpretation is closely related to the nearest neighbour approach, but here only the test image's distance (here the negative of the inner product) to each of the K class templates are calculated instead of its distance to each of the N images in the training set.

Later on it becomes too cumbersome to keep track of two sets of parameters, W and $\boldsymbol{b}$, and therefore, for the rest of the thesis we will write the linear classifier as:

$$f(\boldsymbol{x}_i, W) = W\boldsymbol{x}_i,$$

where $\boldsymbol{b}$ is now contained in the last(/first?) column of W and the last element of $\boldsymbol{x}_i$ is now the constant, 1. This is the so-called bias trick.

Note that thus far we have used raw pixel values in the range of $[0, 255]$ as input. However, in practice, it is more common to subject the input images to some preprocessing before inputting them into the score function. The benefits of this will be made clear in the optimisation section. Common preprocessing techniques are the centering and scaling of the pixels so that their values lie in the range of $[-1, 1]$. To center the input image, is to calculate a *mean image* from the training images and subtract each of its pixel values to from the corresponding pixel values of each image in the training set. This is identical to zero mean centering for standard statistical learning tasks - each pixel is seen as in input feature. Scaling is done by dividing each pixel by a function of its variance accross the whole training set.

2.4 Loss Function

To evaluate the agreement between the score function and the ground truth labels, we need a loss function. A loss function, also known as the cost function or the objective, is high when the score function does a poor job of mapping

the input images to the class scores, and low when it does so accurately. There are multiple ways of defining such a loss function.

2.4.1 Multiclass Support Vector Machine Loss

A commonly used loss is the Multiclass Support Vector Machine (SVM) loss. In statistical learning this is more commonly known as the Hinge Loss. The SVM loss is designed in such a way that it wants the correct class for each image to have a score higher than the incorrect classes by some fixed margin Δ. More precisely, the multiclass SVM loss for the i-th example with label y_i can be given by:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + \Delta),$$

where $s_j = f(x_i, W)_j$ is the score for the j-th class computed for image i. Here L_i consists of $K - 1$ components, each representing an incorrect class. A component will make no contribution to the loss if the calculated class score for the corresponding incorrect class is less than the correct class score by a margin of Δ, *i.e.* $s_{y_i} - s_j > \Delta$. It will make a positive contribution otherwise. As an example, suppose we have three predicted class scores for an image $s = [4, 5, -3]$ and that the second class is the true label. Let $\Delta = 2$. The loss computed for this image will then consist of 2 components:

$$\begin{aligned} L_i &= \max(0, 4 - 5 + 2) + \max(0, -3 - 5 + 2) \\ &= 1 + 0 \end{aligned}$$

We see that although the predicted class score for class 1 was smaller that the predicted class score for the true label, class 2, it was still within a margin of $\Delta = 2$ and there had a positive contribution to the loss. The predicted class score for class 3 was far lower than predicted class score for the true label and therefore did not make any contibution to the loss. In summary, the SVM loss function wants the score of the correct class to be larger than the incorrect class scores by at least Δ, if not, we will accumulate a loss.

Note that the loss is typically evaluated on a set of images and not just one, as we have describe thus far. The average loss of a set with N images can be written as $L = \frac{1}{N} \sum_{i=1}^{N} L_i$. Another variation of the SVM loss is to replace the $\max(0, \cdot)$ term with the term, $\max(0, \cdot)^2$, which results in the squared hinge loss or the L_2-SVM loss. This penalises violated margins more heavily and may work better in some cases. [https://arxiv.org/abs/1306.0239]

There is still one problem with the SVM loss described thus far. Suppose we have found a weight matrix W that correctly classifies all input images and by the correct margins, *i.e.* $L_i = 0$, $\forall i$, then setting the weight matrix to λW, for $\lambda > 1$ will have the same solution. This means the solution to the optimisation problem is not unique. It would make the optimisation task

easier if we could remove this ambiguity. This can be done by adding a penalty term to the loss function, also know as regularisation. The mosty common regularisation penalty, $R(W)$, is the L_2-norm:

$$R(W) = \sum_k \sum_l W_{k,l}^2,$$

which is simply the sum of the squared elements of the weight matrix. The full SVM loss can now be defined as:

$$L = \frac{1}{N} \sum_i L_i + \lambda R(W).$$

The two components of the loss can be called the *data loss* and the *regularisation loss*. λ determines how much regularisation should be done. If λ is large, more regularisation will take place.

The regularisation penalty ensures a unique (or less solutions?) solution to the optimisation problem by restricting the weight parameters in size. Greater weight parameters will result in bigger loss, if everything else remain constant. Another appealing property is that penalising large weights tends to improve generalisation, because it means that no input dimension can have a very large inlfuence on the scores all by itself.

Typically, only the weight parameters are regularised, since the bias terms do not control the strength of influence of an input dimension. Howerver, in practice the often turns out to have a negligible effect.

- http://cs231n.github.io/classification/

2.5 Neural Networks

2.6 Deep Learning

Chapter 3

Convolutional Neural Networks

3.1 Introduction

3.2 Core Layers

3.2.1 Convolutional Layers

3.2.2 Pooling Layers

3.2.3 Activation Layers

3.2.4 Fully Connected Layers

3.3 Optimization (or Training ?)

3.3.1 Back Propogation

3.3.2 Stochastic Gradient Descent

3.3.3 Learning Rates

- cyclical
- decay
- momentum

3.3.4 Freezing Layers

- https://arxiv.org/abs/1706.04983

3.4 Loss Functions

3.5 Summary

Chapter 4

Convolutional Neural Networks in Practice (other title)

4.1 Introduction

Data-driven approach: provide the computer with many examples of each class and the develop learning algorithms that look at these examples and learn the visual appearance of each class.

4.2 Visualizing CNN's

4.3 Transfer Learning

- ImageNet

4.4 Famous Architectures

- AlexNet, Inception, ...

4.4.1 VGG

4.4.2 ResNet

4.4.3 DenseNet

4.5 Regularization

4.5.1 Normalization Layers (maybe move to core)

4.5.2 Data Augmentaion

4.5.3 Pseudo-Labelling and Knowledge-Distillation

4.5.4 Dropout

4.6 Generalization (?)

- https://arxiv.org/abs/1706.01350

Chapter 5

Multi-Label Convolutional Neural Networks

5.1 General Multi-Label Learning Approaches

5.2 Spatial Regularization Networks

- https://arxiv.org/pdf/1702.05891.pdf

5.3 From Single to Multi Output Paper ()

5.4 RCNN paper ()

- Other object detection

Chapter 6

Things that need a place:

- challenges for image classification: (maybe in CNNs in practive)
  - http://cs231n.github.io/classification/
- Feature learning
- one-shot learning:
  - https://github.com/sorenbouma/keras-oneshot
  - https://github.com/fchollet/keras/blob/master/examples/mnist_siamese_graph.py
  - https://sorenbouma.github.io/blog/oneshot/
- multi-task learning:
  - https://arxiv.org/abs/1706.05137
- test time augmentation
- relational learning:
  - https://arxiv.org/pdf/1706.01427.pdf
- Fully-Convolutional Networks
- Spatial Pyramid Pooling: https://github.com/yhenon/keras-spp
- AutoML:
  - https://research.googleblog.com/2017/05/using-machine-learning-to-explore.html

Appendices

Appendix A

Benchmark Datasets

Appendix B

Software

Bibliography