Data Science Internship at ViSenze – Building Deep Learning Classification Models in the Fashion Domain

Project report submitted in partial fulfillment of the requirement for the degree of

Bachelor of Technology

In

Electronics and Communication Engineering

Submitted By

Agastya Seth

Roll No: 1610110047

Under supervision of

Jiangchun Li
Senior Data Scientist, ViSenze, Singapore

and

Dr. Upendra Kumar Pandey
Department of Electrical Engineering

Department of Electrical Engineering

School of Engineering

Shiv Nadar University

May 2020

Candidate Declaration

I hereby declare that the thesis entitled “Data Science Internship at ViSenze – Building Deep Learning Classification Models in the Fashion Domain” submitted for the B. Tech Degree program. This thesis has written in my own words. I have adequately cited and referenced the original sources.

_X_________________

Agastya Seth

Date: __________

Certificate

It is certified that the work contained in the project report titled “Data Science Internship at ViSenze – Building Deep Learning Classification Models in the Fashion Domain,” by Agastya Seth has been carried out under my/our supervision and that this work has not been submitted elsewhere for a degree.

_X___________________

Dr. Upendra Kumar Pandey

Department of Electrical Engineering

School of Engineering

Shiv Nadar University

May, 2020

Abstract

ViSenze is a Singapore-based startup which has pioneered their visual search platform. Simply put, it allows the user to search for a particular product, and similar products by simply clicking a picture. This platform is currently deployed by OEM phone manufacturers including Samsung, Huawei, etc. in their smartphones. It is also used by major e-commerce platforms like Glami and Rakuten for providing their users an option to find products of their interests using image search. It is also used by them for internal cataloging purposes. The fashion domain, in particular, is one of the major focuses for ViSenze as a company, since visual search is particularly helpful for searching for fashion products and accessories, which are otherwise too nebulous to be described in words.

In this report, I will begin by providing an introduction to ViSenze and its products and introduce some key concepts on Deep Learning which were important for me to learn to be successful in my deliveries at ViSenze) and discuss the projects undertaken by me and key learnings through the 5 month tenure of this wonderful internship.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 ViSenze, the Company

ViSenze is an innovative Singapore headquartered start-up which powers visual commerce at scale for retailers and publishers. The company delivers intelligent image recognition solutions that shorten the path to action as consumers search and discover on the visual web. Retailers like Rakuten and Myntra use ViSenze to convert images into immediate product search opportunities, improving conversion rates. Media companies use ViSenze to turn any image or video into an engagement opportunity, driving more new and incremental dollars. Currently specialized in fashion marketplaces, with that being the most explored vertical. ViSenze targets online local or global retailers of all sizes where access to visual search, computer vision and machine learning technologies is critical to their business.

Venture-backed by Rakuten and WI Harper, ViSenze is built by web specialists and computer scientists with deep machine learning and computer vision experience. ViSenze has offices in US, UK, India, China and Singapore. The company originally started as a part of NExT, a leading research center jointly established between National University of Singapore and Tsinghua University of China.

1.2 ViSenze Product Offerings

1.2.1 Visual Commerce Platform

ViSenze's AI-powered Visual Commerce platform enables personalized discovery by showing visually similar products. Three easy steps to quickly find the product the customer desires. This is described in Figure 1-1

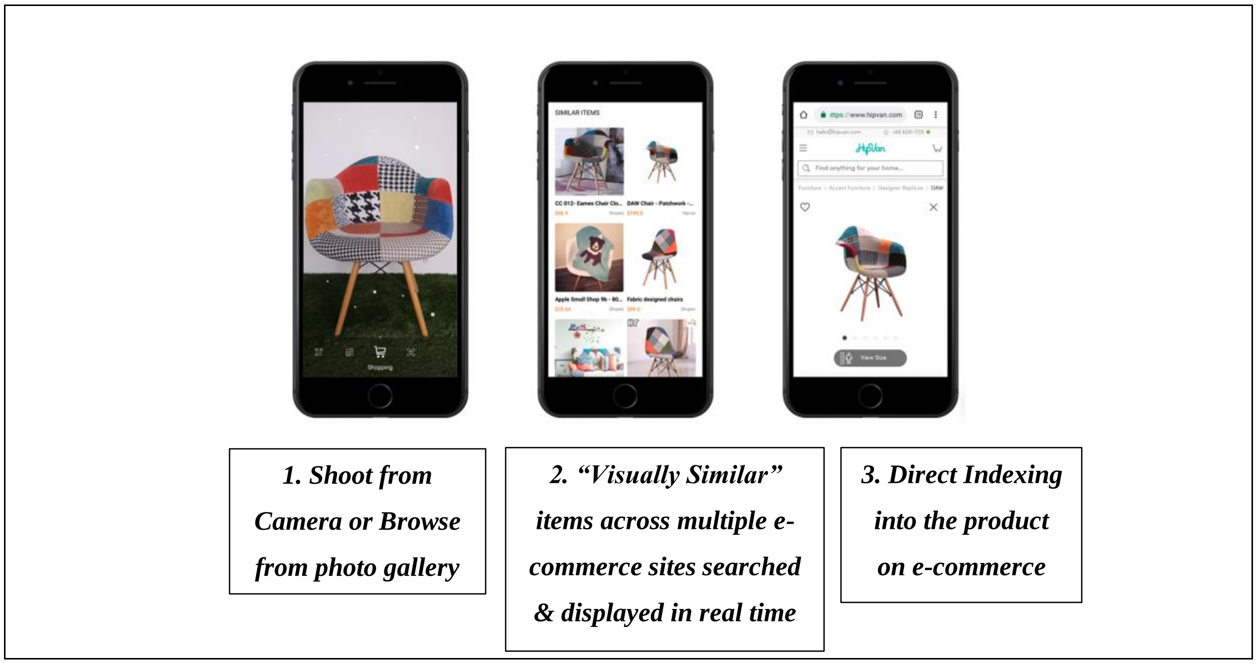

1. Shoot from Camera or Browse from photo gallery
2. "Visually Similar" items across multiple e-commerce sites searched & displayed in real time
3. Direct Indexing into the product on e-commerce

Figure 1-1 The 3-Step image search experience enables users to find contextually similar items based on the query image

Here are some salient features of their ViSearch Visual Search Tool:

- Natively integrated (visual shopping) on smartphone brands – Samsung, Huawei, LG & VIVO
- Pre-integrated & indexed most major e-commerce sites & products (400m + purchasable products)
- Powered by cutting-edge AI
- While Visual API's exist for customers to integrate ViSenze's visual search and recognition algorithms into their applications, this platform has many affiliate programs to start leveraging the capability without any special/custom integration requirement

1.2.2 Visual API’s

ViSenze provides SDKs in various languages, so that their APIs can be easily integrated into customer’s web and mobile applications.

Search By Image

This analyzes the contents of the image, information on what’s in the image sent back and similar images shown. Some marquee customers using ViSenze’s Search By Image are Uniqlo, Rakuten, Hipvan etc.

Visually Similar Product Recommendations

Analyzes the contents of the image and surfaces visually similar products from hundreds of online stores. Some marquee customers using ViSenze’s Visually Similar Product Recommendations are Caratlane, Zalora etc.

Automated Product Tagging

Analyzes content of the images (image quality, fashion attributes) and tag values describing the image content are returned.

1.3 ViSenze: Key Technology Differentiation

While many large companies like Google (Google Goggles), Microsoft (Microsoft Lens), Instagram etc have created and matured visual search solutions, what is the key differentiation ViSenze’s brings in? For one, Google Goggles and Microsoft Lens were designed to be “generalized” visual search engines which could identify the whole world for you, ViSenze has focused on the fashion vertical and with the huge amount of training dataset (including a lot of data (image) scraping and data augmentation), they have achieved best-in-class accuracy on their deep learning models for image search and recognition.

Besides, their world class internal framework that allows data scientists to generate, test and deploy multiple models in extremely efficiently. Its features include managing workflows of processing big archives of images data, training the models, working with several machine learning / deep learning algorithms and fine-tuning their parameters for optimum performance, and providing back office tools and APIs to interface with their platforms, at scale. The deep learning framework supports several deep learning libraries, including Caffe, MXNet and TensorFlow.

1.4 ViSenze Organization Structure

The figure below shows the organization structure of ViSenze's engineering team. The Data Science team, along with report to the CTO.

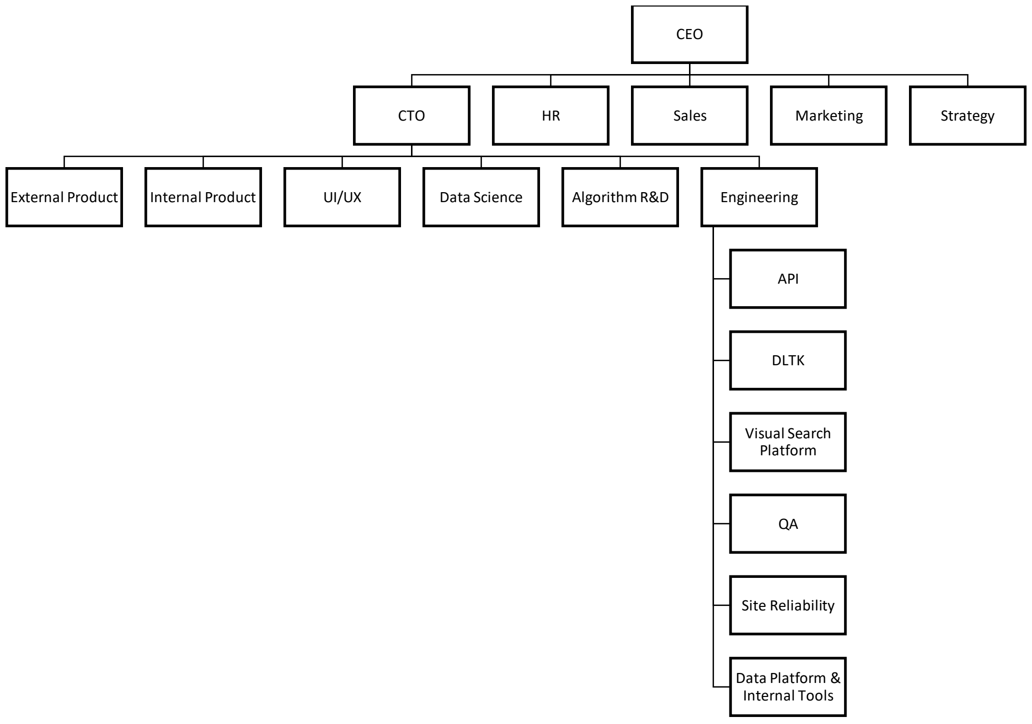


Figure 1-2 The organizational structure of ViSenze's engineering team

1.5 Role of a Data Science Intern

As a data science intern at ViSenze, I was required to work on tagging models across domains – but since a large part of the revenue from ViSenze's comes from fashion e-commerce websites, the models I worked on during my internship – fashion_accessories and view_angles models were bound to the fashion domain only. In the following sections, I will describe briefly the general process workflow, some of the deep learning models pertinent to the said image classification projects, as well as a summary of the various models and their results.

As a result of the confidentiality agreement signed with ViSenze, this project report will have some key features/details redacted, including but not limited to certain internal processes / workflows unique to ViSenze, customer data, exact taxonomy or guidelines used for the models, and/or any details or empirical conclusions derived from the experiments. I will however try my best to put forward in a generic way, a brief summary of the projects undertaken at ViSenze, as well as a logical conclusion w.r.t my experiences and learnings from this internship.

Chapter 2

Deep Learning

2.1 Machine Learning

The term deep learning has gone through several iteration of its definition over the years. A contemporary definition can be found in the Deep Learning book by Ian Goodfellow [1]. It defines it as the decomposition of complex concepts into simple ones and the recombination into new complex concepts. The algorithm must, thus, establish a hierarchy of concepts. A visualization of said hierarchy could be defined by a multi-layered graph, which could subsequently be called "deep" in a graph topological context. In this thesis, this term is used in a relatively narrow sense - as proposed by Skansi in his book Introduction to Deep Learning [2], which describes artificial neural networks as a subfield of machine. In this chapter, various concepts for understanding deep learning are presented and discussed.

Machine learning is a subfield of computer science. At its core, it is the foundation for a set of statistical tools that estimate complicated functions by learning from data. Machine learning can be divided into two main approaches, supervised and unsupervised learning. Supervised learning generally means that the program is given both input and the desired output, for example, pictures of objects with corresponding labels of what is depicted. The goal of the learning (or training) is to construct a map between those two. In contrast to supervised learning, the unsupervised learning approach does not provide the program with the correct output. Here, the goal of training is to find structure inside the given input; it is used, for example, in autoencoders [6].

This analysis aims to provide insight into one of the most successful types of neural networks in the field of image and pattern recognition, the convolutional neural networks. To do this, a deconvolutional network is built and its output analyzed.

This chapter provides a brief introduction into the important concepts of machine learning and neural networks as a part of machine learning. In the projects discussed further, convolutional neural networks are used to classify (tag) images under various concepts in the fashion domain, which are queried through ViSenze's various systems, including their visual search algorithm.

With this context, a rather useful definition for machine learning is given by Tom Mitchell in his book *Machine Learning* [7]:

> "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*."

The **task *T*** is the learning goal of the algorithm. It is defined by the user and is usually communicated to the algorithm by providing an example of how an event is to be processed. Here, event means a set of features describing a single data point. It is usually denoted as a vector $x \in R_n$ with entries x_i representing features of the event.

In the fashion products tagging, as the case for the projects presented in this report, each event is an image of a product (like a shoe, or a t-shirt) and its features are the values of the pixels in the image. The task is classification, where the algorithm is asked to assign a probability to the image with which it belongs to any of the k categories. To solve this task, the learning algorithm produces a function $f: R_n \rightarrow \{1, \dots, \mathrm{k}\}$, where *f* outputs a classifier distribution.

The **performance measure *P*** is the metric by which the competence of the algorithm is evaluated. The metric can be chosen by the user and generally has to be tailored to the task *T*. In the case of classification, the performance measure *P* usually employs a loss function that quantifies the disagreement between the predicted and true output. In a simple case, the loss function could be the amount of correctly classified events in proportion to the total number of events. In more sophisticated approaches, it is a continuous-valued score for each event. The loss function for the deep CNN models used in the projects are discussed later in this report. The performance is tested on data that were not included in the training sample. This independent sample is called the validation data set or validation sample. To estimate how well the algorithm will perform in a broader range of applications, additional performance tests not related to the chosen loss function are performed.

The **experience *E*** encompasses the provided data set and any additional information that the machine learning algorithm can use to learn. It is here where supervised and unsupervised algorithms can differ. They are not entirely exclusive and applications that combine both methods, called semi-supervised learning, exist. It is still a useful paradigm to categorize different machine learning goals. Unsupervised learning algorithms are presented with datasets with many features and learn properties of the structure of the dataset. They produce a probability distribution $\mathrm{p(x)}$ of a random event x. Supervised learning algorithm are trained on datasets containing features, where each event is also associated to a predetermined output via a label. They process several events x and an associated value or vector y, then learn to predict y from x. The dataset does not have to be fixed either. Other machine learning techniques, like reinforced learning, make use of a feedback loop that provides new data in response to the learning system. As noted previously, the presented classification models are based on supervised algorithms

2.1.1 Training

Training a machine learning algorithm can be seen as approximating two functions $y(x)$ and $\hat{y}(x)$, where the algorithm tries to find the closest distance from $y(x)$ to $\hat{y}(x)$, in a given metric. The basic principles of training can be illustrated with a linear regression:

$$\hat{y} = w^T x \qquad 2.1$$

Here, $\boldsymbol{w}$ is a vector of parameters that the algorithm can optimize, which in a machine learning context, are called weights. They determine how features xi correlate with the output $\hat{y}$; finding the closest "distance" between $\hat{y}$ and y is called predicting y from x. There are many possible ways for an algorithm to optimize the parameters. In the provided example, a possible learning method can be to minimize the mean squared error (MSE) from the following equation on the training set x:

$$\text{MSE} = \frac{1}{n}\sum_{i}(\hat{y} - y)_i^2 \tag{2.2}$$

Here, n is the number of events x with features i. $\hat{y}$ is called the prediction of the model on the training set. The MSE is minimized by solving the gradient with respect to weights w for 0:

$$\nabla_w \text{MSE} = 0 \tag{2.3}$$

To validate the training process, the MSE is also calculated for an independent validation set x_{val} with n_{val} events, that the algorithm does not use for training:

$$\text{MSE}_{\text{val}} = \frac{1}{n_{val}}\sum_{i}(\widehat{y_{val}} - y_{val})_i^2 \tag{2.4}$$

Here, y_{val} is the set of correct output values and y_{val} the algorithm prediction for yval based on xval. The algorithm iterates the training and validation process until the error is sufficiently small, a criterion specific to the task.

In general, the user has to define a model that describes the output y in terms of input x, like the linear regression above, and a learning method. Models that are used in deep learning are described later in this chapter. The learning method employed for this report, called Gradient-Based Learning, is described in section 2.2.4.

2.1.2 Overfitting and Underfitting

A successful machine learning algorithm should perform well on unseen input samples that are of the same type as the training and validation data sets. Therefore, the performance of the algorithm should not only be evaluated on its ability to minimize the training error, but also on its ability to minimize the difference between the training error and validation error. This difference is called the generalization error and is discussed in more detail in later chapters. Both, the training and generalization errors highly depend on the representational capacity of the algorithm, which is a measure of how well the algorithm is able to adapt to a range of outputs. In the example above, a linear regression was used as the model for training. If the user wants to model data whose underlying distribution is a higher order polynomial, the linear model will not be able to describe the desired behavior. The training error will never be sufficiently small. This problem is called underfitting the model.

In theory, if the algorithm’s representational capacity is allowed to be arbitrarily high, it will eventually perfectly fit any given finite event set to the outputs. But this limits the algorithm’s ability to perform on new data, since in this regime it is overly specialized to the training data. If the algorithm has the capacity to achieve a small training error but is not able to make the generalization error small, this problem is called overfitting. The key to avoiding overfitting is to find a sufficiently complex model that is still able to generalize. Figure shows the typical relationship between capacity and error.

2.1.3 Bayes Error and The No Free Lunch Theorem

There are statistical constraints on the classification performance of machine learning algorithms that have to be considered.

Assume there is an ideal model that matches the true underlying distribution of a data set X. The model will still incur errors on classification for multiple reasons. The mapping from events $x \in X$ to y may be inherently stochastic or X may be incomplete in terms of predicting y, because

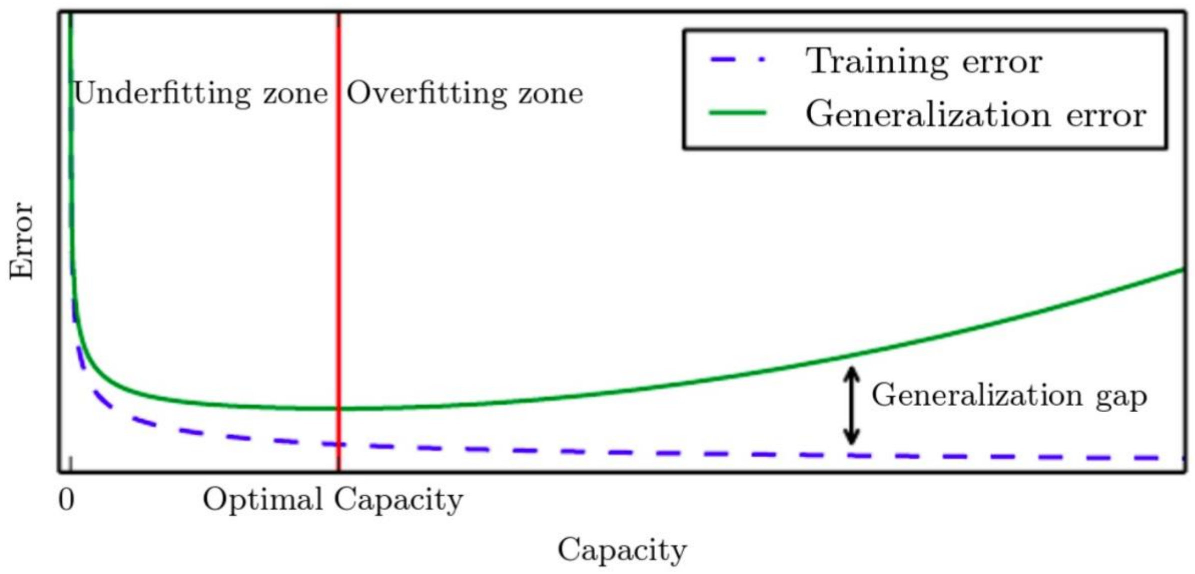


Figure 2-1 Training error and generalization error as functions of capacity. Between the underfitting and overfitting zone an optimal capacity exists. Figure taken from [1].

y may also depend on other variables outside of those that can be obtained from X. Finally, the distribution describing X may be noisy, meaning that an event x can belong to more than one class. The smallest possible error an ideal model can obtain is called Bayes error. In this limit, machine learning only offers probabilistic rules, i.e. it promises to find rules that are *probably* correct about *most* members of the set they concern.

The second constraint to consider is called the No Free Lunch Theorem [3]. It states that no one machine learning algorithm is inherently better performing on unseen data than any other algorithm if averaged over all possible data-generating distributions. This means that there is no single best algorithm to do all possible tasks. Therefore, assumptions about the given data distributions have to be made. This way algorithms can be tailored to the task.

2.1.4 Regularization

To address the aforementioned issues, the machine learning algorithm has to be designed with a set of preferences that align well with the given task. Building and modifying these preferences with the goal of decreasing the generalization error but not the training error is called regularization. Some important aspects of regularization are introduced here.

One way to efficiently adjust the capacity of the model is to examine the bias and variance of the algorithm. The bias of an algorithm A for sample size m at event x is defined as

$$bias(A, m, x) = \mathrm{E}\left(\hat{f}(x)\right) - f(x) \tag{2.5}$$

Here, f is the true distribution underlying the data, $\hat{f}$ the prediction for the data and $E\left(\hat{f}(x)\right)$ the expectation value of $\hat{f}(x)$ at event x. The prediction is called unbiased, if $bias(A, m, x) = 0$. This implies, that $E(\hat{f}\ ) = f$.

The variance (var) is given by

$$var(A, m, x) = \mathrm{E}\left[\left(\widehat{f_S} - \mathrm{E}\left(\hat{f}(x)\right)\right)^2\right] \tag{2.6}$$

where S are all training samples of size m. The variance describes random variations between different training sets S, that can result from noise in the training data, varying numbers of events in classes, or random behavior in the algorithm itself, such as different initial parameters. Both of the quantities above contribute to the generalization error (Err_{gen}) as discussed in [4]

$$Err_{gen}(A, m, x) = bias(A, m, x)^2 + var(A, m, x) \tag{2.7}$$

shows how this connection can be applied to find the optimal capacity to minimize the generalization error.

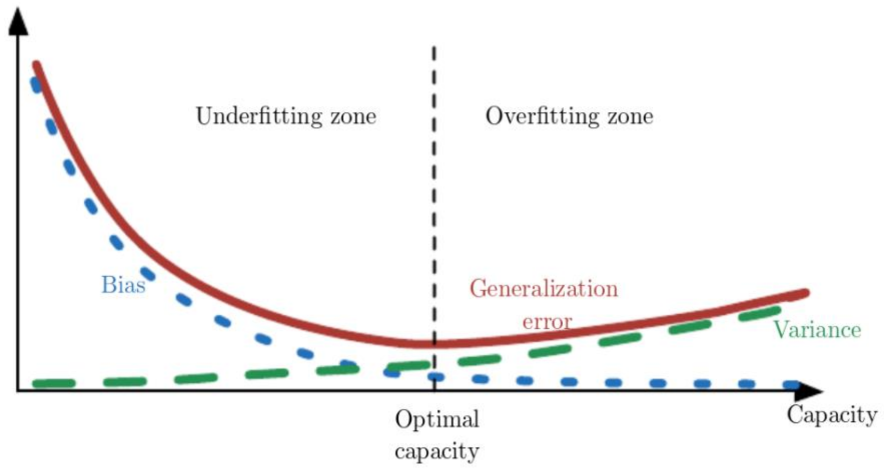


Figure 2-2 Generalization error as a function of capacity, bias and variance [1].

Commonly used regularization methods to avoid overfitting include Dropout [5] which is also employed in the projects undertaken and described in in later sections, and introducing variations into the training data during preprocessing.

2.2 Artificial Neural Networks

To build a machine learning algorithm, the user must define a model, a cost function, and an optimization procedure that befit the structure of the data. This chapter deals with the most prominent kinds of models in machine learning, artificial neural networks, and learning algorithms that can be used with these models.

As mentioned before, a machine learning algorithm is used to approximate a given function *f*. In the case of a classification, *f* maps an input x to a category y. An algorithm $\hat{f}$ will derive *f* with weights w such that *f* approximates

$$y = \hat{f}(x; w) \tag{2.7}$$

In general, *f* can be a composite function of any number of functions f_i in the form

$$\hat{f} = \widehat{f_n} \circ \ldots \circ \widehat{f_3} \circ \widehat{f_2} \circ \widehat{f_1} \tag{2.8}$$

This structure is called a network, where $\widehat{f_\imath}$ corresponds to the first layer, also called input layer, $\widehat{f_2}$ to the second layer, and so on. The final layer $\widehat{f_n}$ is called the output layer. The $n - 1$ layers between input and output layers are called hidden layers, because their behavior is only implicitly constrained by the training data, as the data do not show the desired output for each of those layers. The number of layers n defines the depth of the network. This simple chain structure is only one possible way to build a network and was first introduced in 1958 by Frank Rosenblatt [6]. In general, any connection structure between the different functions is possible. Some such alternative examples are deep residual networks [7], long/short term memory networks [8], and self-organized networks, that can decide which of its layers suit a given task [9].

All of the above networks are often referred to as artificial neural networks because they are loosely inspired by neuroscience. It is an established name used by the com- munity. However, the goal of neural networks is not to model the human brain. Neural network research is fundamentally based on and driven by mathematical and engineering disciplines rather than biological brain function.

2.2.1 Types of Units and Activation Functions

Each layer in a neural network consists of many units that act in parallel, where each unit represents a vector-to-scalar function. Most units can be described as accepting a vector of inputs x, computing the affine transformation

$$z = W^T x + b \tag{2.9}$$

where the matrix W describes the mapping from x to z with bias b applied to the transformation. Afterwards, a nonlinear function $g(z)$, called the activation function, is applied element-wise on z. The design of units is an active area of research and is not yet supported by definitive guiding theoretical principles. Most hidden units are distinguished from each other only by the choice of the form of the activation function $g(z)$.

ReLU

The default choice for an activation function in modern neural networks is the max function

$$g(z) = \max\{0, z\} \tag{2.10}$$

A unit that employs this function is called a rectified linear unit (ReLU) [10]. ReLUs are quickly optimized since the derivative is either 0 or a positive constant value through the domain. This makes the gradient direction far more useful for learning than it would be with activation functions with non-vanishing and higher order derivatives. One drawback to ReLUs is that they cannot learn via gradient-based methods on examples for which the activation is zero.

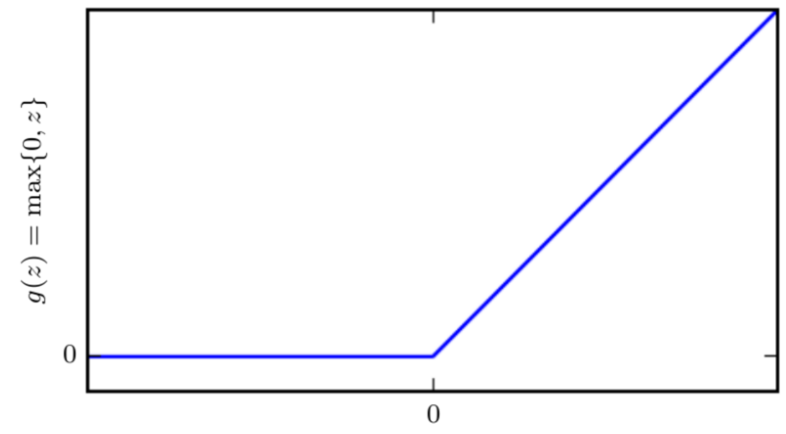


Figure 2-3 A rectified linear unit outputs zero across half its domain [1]

Maxout

Maxout units are generalized rectified linear units [11]. Instead of applying an element-wise function $g(z)$, maxout units divide z into groups of k values. Each maxout unit then outputs the maximum element of one of these groups:

$$g(z)_i = \max z_j\,, j \in G^{(i)} \tag{2.11}$$

Here, $G^{(i)}$ is the set of indices for group $i, \{(i-1)k+1, \ldots, ik\}$. This provides a way of learning a piecewise linear function that responds to multiple directions in the input x space. A maxout unit can facilitate learning for a piecewise linear, convex function with up to k pieces. Since each maxout unit is parametrized by k weight vectors instead of just one, it needs more regularization than an ReLU.

Sigmoid

The sigmoid function is used to represent a probability distribution over a binary variable. It is defined as

$$\sigma(z) = \frac{1}{1 + \exp(-z)} \tag{2.12}$$

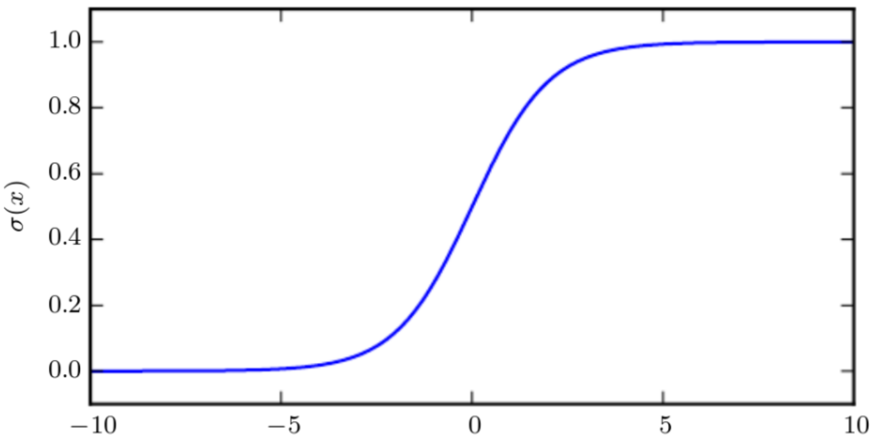


Figure 2-4 The sigmoid function [1]

This activation function saturates to 0 when *z* becomes very negative and saturates to 1 when *z* becomes very positive. For large absolute values of *z*, the gradient can become too small to be useful for learning even if the training data abundantly populate these regions.

Softmax

The softmax function of *z*, is a generalization of the sigmoid function that represents a probability distribution over a discrete variable with *n* possible values. Softmax functions are often used as the output units of a classifier. Formally, the softmax function is given by

$$softmax(z) = \frac{\exp(z_i)}{\sum_j \exp(z_i)} \tag{2.13}$$

Cost functions that do not use a log to undo the exp of the softmax cause a failure to learn when the argument to the exp becomes very negative, causing the gradient to vanish.

2.2.2 Back-Propagation

By default, information in a neural network propagates in a forward direction from the input nodes, through the hidden layers, and into the output node. This is called forward propagation. During training, forward propagation can continue until it produces a cost in an output node. The back-propagation algorithm [12] allows the information from the cost to then flow backward through the network in order to compute the gradient. The back-propagation algorithm computes the gradient of scalar z with respect to one of its ancestors x in the graph. First, the gradient with respect to z is taken. This is simply $\frac{dz}{dz} = 1$. From here on, the next gradient with respect to the next parent of z in the dz graph is obtained by multiplying the current gradient by the Jacobian of the operation that produced z. This is repeated backward through the graph until x is reached. When two or more paths can be chosen, the gradients of those paths are simply summed up.

2.2.3 Cost Function

As mentioned before, a learning algorithm needs a metric with which to evaluate the distance between two function. Such a metric is called a cost function or loss function in neural networks. Similar to the euclidean metric being the default choice of metric in euclidean space, the first choice for a cost function in a neural network is the cross-entropy between the training data and the predictions of the model, which makes use of the principle of maximum likelihood.

In most cases, parametric models define a distribution. Let $p_{model}(x;\theta)$ be a family of probability distributions over the same space indexed by θ that maps any configuration x to a real number estimating the true probability $p_{data}(x)$. The maximum likelihood θ_{ML} is then defined as

$$\theta_{ML} = \arg\max_{\theta} p_{model}(X;\theta) \qquad 2.14$$

Other popular cost functions are quadratic functions, such as the mean-square-error (MSE) and the root-mean-square-error (RMSE).

2.2.4 Gradient-Based Learning

Neural networks are usually trained using iterative optimizers based on Cauchy's gradient-descent [18]. Such methods merely drive the cost function to a very low value, in contrast to linear equation solvers used to train linear regression models, as well as convex optimization algorithms with global convergence guarantees used to train logistic regression. Convex optimization converges starting from any initial parameters. Stochastic gradient descent applied to non-convex loss functions has no such convergence guarantee and is sensitive to the values of the initial parameters. For feed-forward neural networks, it is important to initialize all weights to small random values. The biases may be initialized to zero or to small positive values.

Important for this work is the momentum learning method introduced in [13]. It is designed to accelerate learning, especially in the case of high curvature, small but consistent gradients, or noisy gradients. The momentum algorithm accumulates an exponentially decaying moving average of past gradients and continues to move in their direction. A variant of the momentum algorithm that was inspired by Nesterov's accelerated gradient method [14] was introduced in [15]. The difference between Nesterov momentum and standard momentum is where the gradient is evaluated. The former can be interpreted as attempting to add a correction factor to the standard method of momentum.

2.2.5 Data Structure

Frequently, data sets are too large be processed as a whole. Therefore, the data are provided to the network in batches. These batches take the form of multidimensional arrays. That way, the data can be treated as a matrix $X_{i,j} \in R^{i \times j}$, where rows i correspond to events x and columns j correspond to features of those events. The labels are provided in a vector y, where y_i contains the label for event *i*. The above only holds if the data are homogeneous, meaning all the events are of the same dimensionality. Methods for handling heterogeneous data can be found in [1]. In the image classification projects discussed in this report, the input is homogeneous.

Any type of input can be represented by a flattened vector [16], but in this case the data may lose intrinsic structure. An RGB image, for example, contains two order-sensitive axes for width and height, and one axis that is used to access different views of the data (i.e. the red, green, and blue channels of the color image), for which the ordering does not matter. This must be considered when choosing different layer types for the network, as some of the layers do not preserve this structure.

2.3 Important Layer Types

The key layers employed by the models in the presented work are the *convolutional* layer, the *pooling* layer, and the *dense* layer. A network composed of these layers is very typical for image recognition and is based on networks developed by Krizhevsky et al. for ImageNet [17]. In addition, the dropout layer is introduced because it has proven to be an important means of regularization.

2.3.1 Dense Layer

The dense layer uses the linear set of connections between input and output, described in equation 2.9. All units of a dense layer are fully connected to all units in neighboring layers in such a way that the units take every output from all units in the former dense layer as their input and pass their output to all units in the following layer.

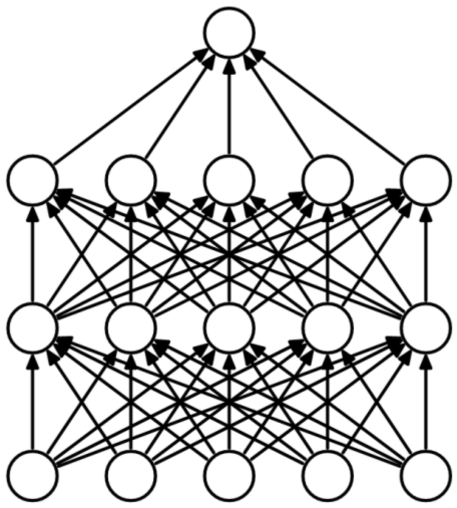


Figure 2-5 A densely connected neural net with two hidden layers.

This leads to a very simple matrix-vector-computation, but it also leads to a very large set of trainable parameters. A neural network that consists of dense layers can be very successful for low dimensional data, but computation can become very expensive for high dimensional data like images. To handle such tasks, layers with weight sharing are introduced.

A convolution is a special kind of linear operation on two functions of a real valued argument. A one-dimensional convolution $y(t)$ of two functions $x(t)$ and $w(a)$ is defined as the integral of the pointwise multiplication of these two functions:

$$y(t) = (x \star w)(t) = \int_{-\infty}^{+\infty} x(a)w(t-a)d\,a \qquad 2.15$$

The function $y(t)$ can be seen as a modified version of $x(t)$ weighed by $w(a)$. In a neural network context, $x(t)$ is the input, $w(a)$ the kernel, and $y(t)$ the output or feature map. Since algorithms only compute in discrete steps, the convolution operation has to be discretized:

$$y(t) = (x \star w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a) \tag{2.16}$$

Here, t is an integer. A convolution can be generalized for multiple dimensions, where functions x and w are defined on a set t of integers. With a 2-dimensional input $x(i,j)$, for example images with width and height coordinates i and j, the convolution can be written in the following form:

$$y(i,j) = (x \star w)(i,j) = \sum_{m}\sum_{n} x(i-m, j-n)w(m,n) \tag{2.17}$$

The first prominent implementation of convolutions in neural networks was carried out by Le Cun et al. [18], proving that the discrete convolution has important properties for use in image recognition. It is an operation that preserves the notion of ordering. Only a few input units feed into a given output unit, and parameters are systematically reused, such that the same weights are applied to multiple locations in the input.

The convolution operation can be visualized with the example in figure 2.6. The light blue grid is called the input feature map. The kernel (shaded blue area) slides across the input feature map. In this example, the Kernel is

$$w = \begin{pmatrix} 0 & 1 & 2 \\ 2 & 2 & 0 \\ 0 & 1 & 2 \end{pmatrix} \tag{2.18}$$

At each location, the product between each element of the kernel and the input element it overlaps with is computed and the results are summed to obtain the output at the current location. The result is called the output feature map (green grid).

The following parameters can be adjusted in a convolutional layer:

- the kernel size, also called filter size in neural networks;
- the step size, which is the distance between two consecutive positions of the kernel;
- zero padding, which is the number of zeros concatenated at the beginning and end of an axis.

The resulting effects of those parameter changes on the output are discussed in detail in [19].

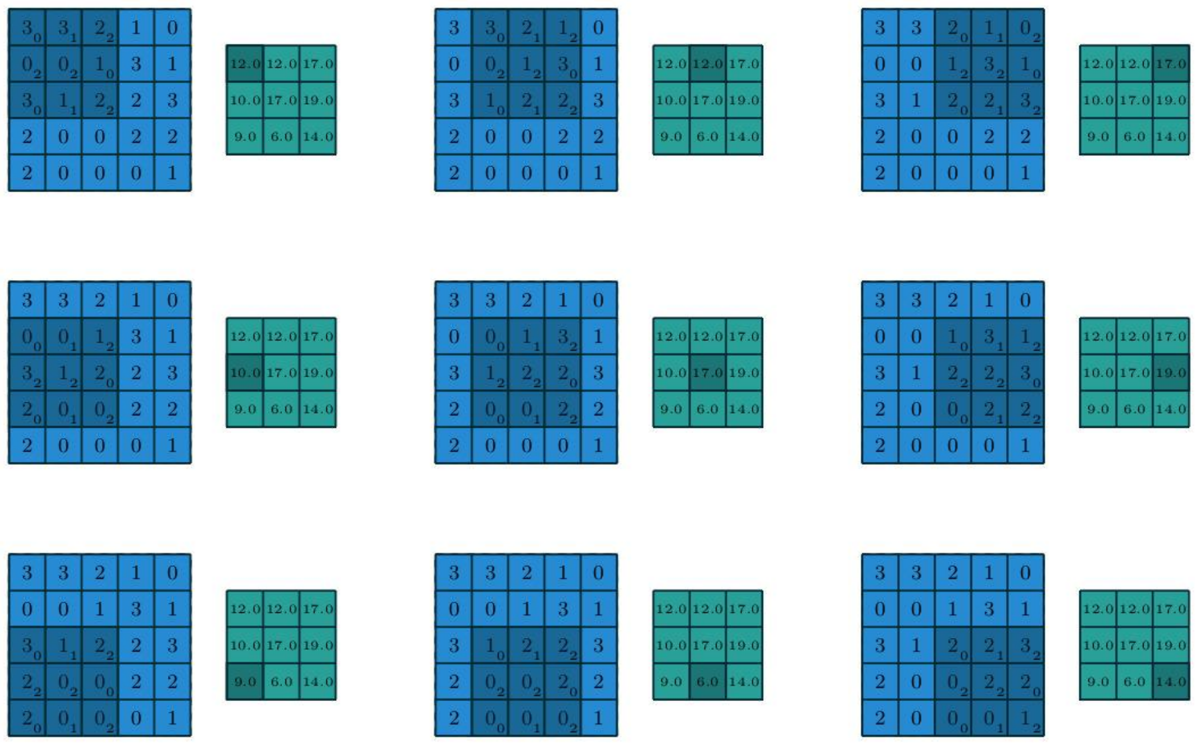


Figure 2-6 The convolution operation

The pooling layer is a tool that ensures invariance of y under small translations of the input, as well as for down sampling. It is very often used in combination with convolutional layers. The pooling operation computes a summary statistic of nearby inputs using an arithmetic function, and can be freely defined. One of the most popular pooling operations is called max pooling [20], which returns the maximum input within a rectangular neighborhood.

The pooling operation is similar to discrete convolution, but replaces the linear combination with some other function, such as the maximum or average of the input of the neighbors. Figure 2.7 shows the max pooling operation applied to the same input as in Figure 2-6 (shown in blue) and its output (shown in green) for the computation in nine steps.

2.3.2 Dropout

Dropout is a method for regularizing of machine learning algorithms. The key idea of a dropout layer is to randomly disable input units after each iteration of the training. A neural network with n units that employs dropout can be seen as a collection of 2^n possible neural networks. These possible networks have a smaller number of units but still share weights such that the total number of parameters is unchanged. Training a network with dropout can be seen as training a collection of 2^n smaller networks with extensive weight sharing.

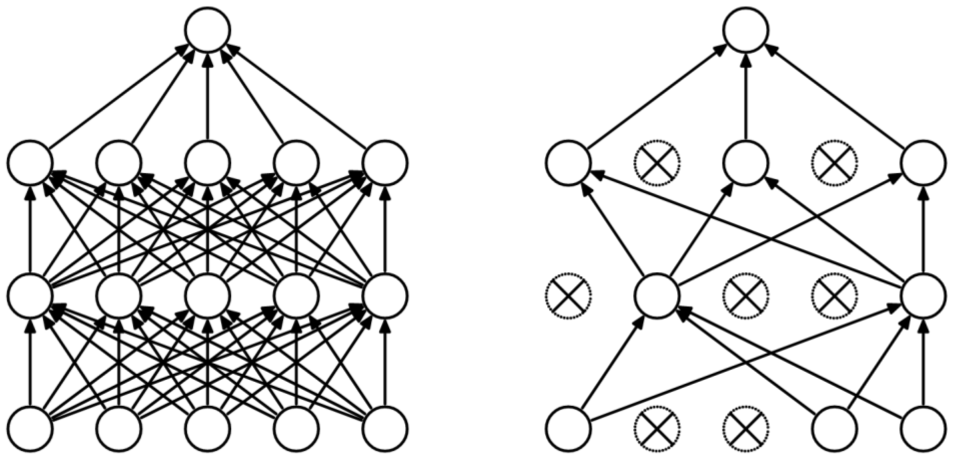


Figure 2-7 The same densely connected neural net as in Figure 2-5(left). An example of a smaller network produced by applying dropout (right). Crossed units have been deactivated by dropout [5]

At test time, it is easy to approximate the effect of averaging the predictions of the smaller networks by using a single network without dropout that has smaller weights. This significantly reduces overfitting and gives significant improvements over other regularization methods [5].

Chapter 3

Literature Review

3.1 ResNet-50

Microsoft ResNet [21] is a very deep CNN, with 152 layers. It has become popular since 2015, when it won both ILSVRC and COCO [22] challenges in five main tasks, becoming the new state-of-art. As name suggests, version of ResNet-50 contains 50 layers. Its use in this works, instead of other deeper versions of ResNet, is justify in the next chapter. For this section I just show a reduced architecture to explain the main concepts of this network, without showing the ResNet-50 architecture due to its huge number of layers. To make the network deeper an intuitive idea could be simply stack layers. Unfortunately, this solution lead to reach higher training error with respect to state-of-art. A properly way to stack layers improving the performance is perform an identical mapping to output of next layer, as it shown in following figure:

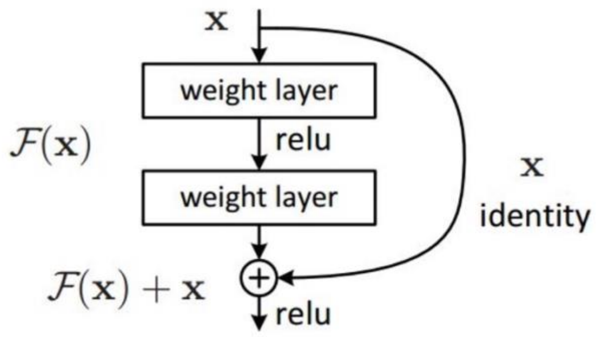

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 0-1 Residual block

Suppose x the input, $H(x)$ desired underlying mapping $F(x)$ and the residual mapping. Then last term is obtained in the following way:

$$F(x) = H(x) - x \tag{1}$$

To obtain the desired mapping $H(x)$ we have to just add x, which perform identical mapping. So, instead of adding a different layer we replicate the previous one in order to be sure to achieve same input value. The connection between input and output is called “shortcut connection’’. The advantages, further to obtain $H(x)$ is that none parameters are adds and the complexity does not increase. When layer size changed a zero padding is adding in order to avoid that number of parameters increased. The most popular ResNet version contains 152 layers, a huge architecture that it is omitted in this explanation for graphical dimensionality problem. A block diagram of the model architecture is shown in Figure 0-2.

3.2 MobileNet v1/v2

MobileNetsv1 [Howard et al., 2017] are another pioneer in efficient inference, their main idea was to use 3×3 depthwise separable convolutions followed by 1×1 pointwise convolutions. A MobileNetv1 block is shown in Figure 0-3 a. To balance accuracy and latency, the authors proposed to use a width multiplier and a resolution multiplier (using a lower resolution than 224 × 224 for the input). One year later, the second version of this network came out [Sandler et al., 2018]. The network is still based on depthwise separable convolutions, but the structure of the block has changed (see Figure 0-3 b.). A particularity of this design is that bottleneck blocks have a bigger number of intermediate channels while their inputs/outputs have less (the authors have a much bigger discussion about the architecture, see [Sandler et al., 2018]). The authors also used Relu6 instead of Relu because they argue it performs better with low-precision computations ([Krishnamoorthi, 2018] advocates the contrary based on experiences on MobileNetv1). Relu6 equation is:

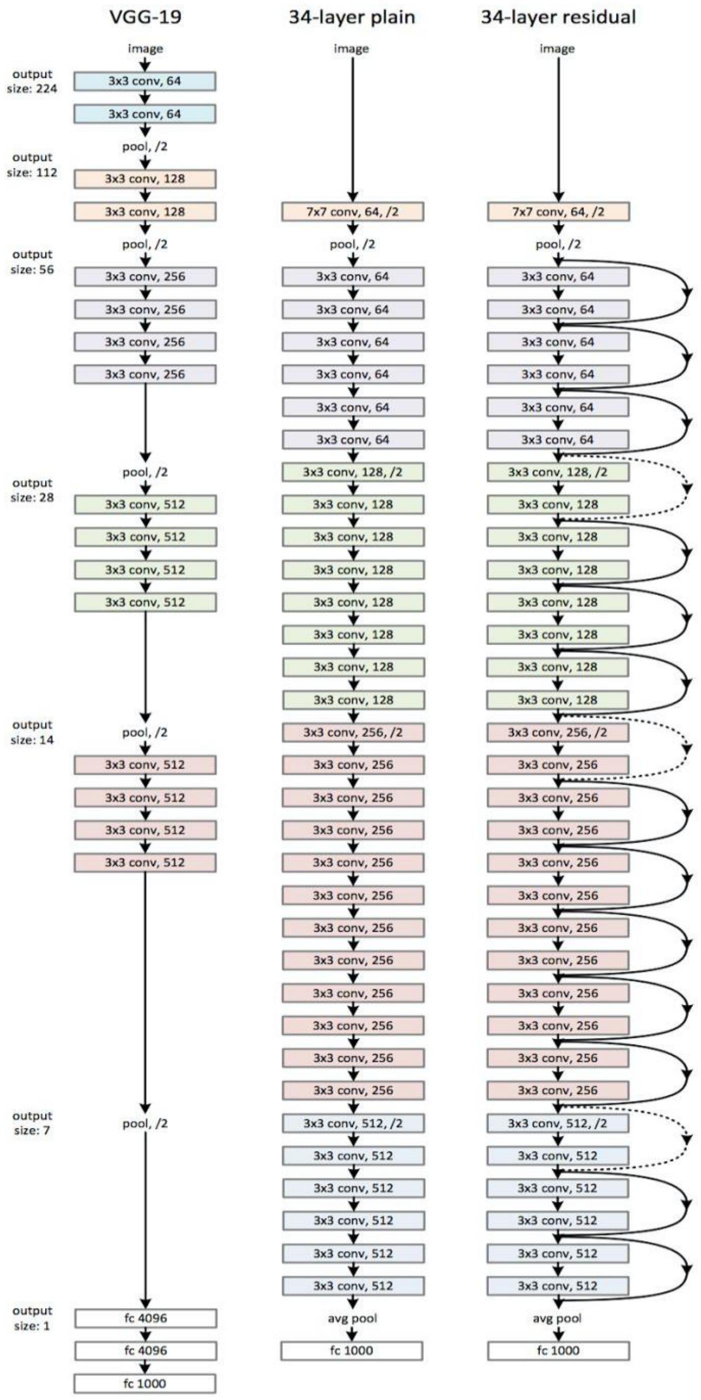
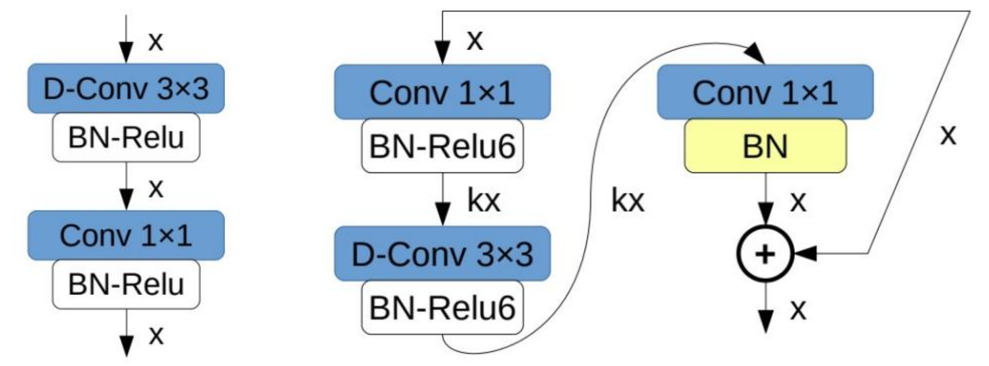


Figure 0-2 ResNet baseline architecture

$$\text{out} = \begin{cases} 6 & \text{if } in > 6 \\ in & \text{if } 0 \leq in \leq 6 \\ 0 & \text{otherwise} \end{cases}$$

Finally, they did not put an activation after the last layer of each block because this layer has a huge number of input channels (see paper for more details). Mobilenetv2 also use width and resolution multipliers (although width multipliers smaller than 1 do not affect the last convolutional layer).



(a) MobileNetv1 block, see [Howard et al., 2017]

(b) MobileNetv2 block, see [Sandler et al., 2018]

Figure 0-3 MobileNet Blocks

Stride might be used in the depthwise separable convolution in which case the skip connection is omitted for MobileNetv2, k is called expansion ratio and is > 1

Another idea developed in the paper is a way to trade inference speed for inference maximal memory consumption. In MobileNetv2 blocks, the biggest tensors are the ones produced inside of the feature blocks. Thus, by avoiding the storage of the bottlenecks intermediate tensors, we can reduce the maximal memory consumption during inference. This improvement is possible since the intermediate tensors are depthwise separable. Indeed, if we denote the output of the block by

$F(x)$ (not taking the skip connection into account), we have that $\mathcal{F}(x) = \mathcal{B}\left(\mathcal{N}\left(\mathcal{A}(x)\right)\right)$, with $\mathcal{N} = \left(\text{BatchNorm}\left(\text{Depthwise}Conv(Relu(\,)6)\right)\right)$ being a per-channel non-linear operation, $\mathcal{A} = BatchNorm\left(Conv(\,)\right)$ and $\mathcal{B} = \text{BatchNorm}\left(\text{Conv}(\,)\right)$ being linear operators, the whole operation is factorizable channel-wise, i.e. if we group the intermediate channels into t groups, $\mathcal{F}(x) = \sum_{i=1}^{t} \mathcal{B}_i\left(\mathcal{N}\left(\mathcal{A}_i(x)\right)\right)$. This idea reduces the maximal memory consumption since we do not have to store the whole tensors but is more computationally costly (more cache misses).

This will not be developed further in this work since we had no RAM consumption problem and our main interest was inference time. It is however mentioned as we found the idea interesting.

3.3 Data Visualization and coverage analysis using T-SNE

t-Distributed Stochastic Neighbor Embedding (t-SNE) is a modern machine learning algorithm used for dimensionality reduction. The algorithm is extremely useful for visualizing high-dimensional data on a 2- or 3-dimensional plot.

In contrast to principal component analysis (PCA), a linear dimensionality reduction technique, t-SNE is a non-linear dimensionality reduction technique. t-SNE is able to take high-dimensional data points that lie on or near a non-linear manifold and preserve the local structure when mapping onto a low-dimensional space, which is not possible with any linear technique [23].

Panels A and B of Figure 0-4 display the famous “Swiss Roll” dataset, which is a non-linear manifold in three dimensions [3]. Panel C shows the application of PCA to the data and subsequent plotting in two dimensions. The data is essentially flattened down without any regard to the underlying spiral manifold relationship among points because it is a linear mapping. The blue and red data points are neighbors in two dimensions despite being far apart on the original manifold.

When the non-linear dimensionality reduction method Isomap is used, however, we see in Panel D that the spiral manifold actually unrolls itself, preserving local structure of the original data.

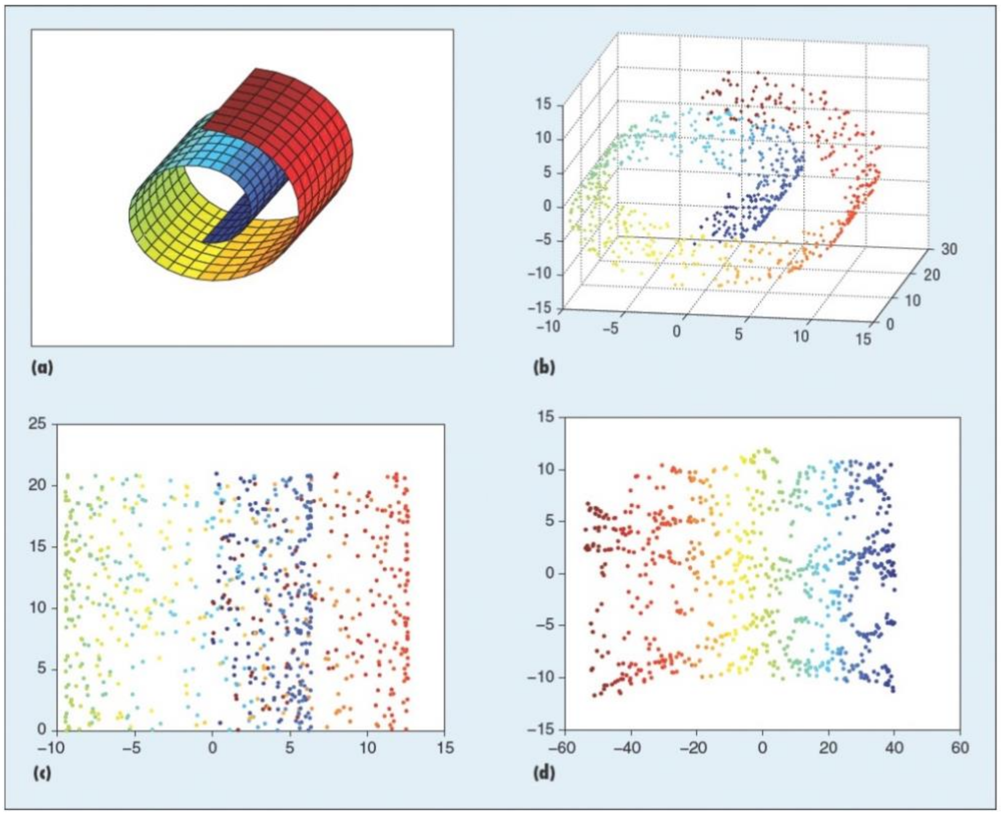


Figure 0-4 Application of PCA (c) and Isomap (d) to Swiss Roll Data

The full t-SNE algorithm is described in ALGORITHM 1

ALGORITHM 1 – T-SNE

```
Data: $X = x_1, x_2 \dots x_n$

Cost Function Parameter: Perplexity $Perp$;
Optimization Parameters: Number of iterations $T$, Learning rate $\eta$, Momentum $\alpha(t)$;

Result: Low-dimensional data representation $Y(T) = y_1, y_2 \dots y_n$

begin
    Compute pairwise affinities $p_{j|i}$ with perplexity $Perp$ using Equation 0.1;
    Set $p_{ij} = \frac{p_{ji}+p_{ij}}{2n}$;
    Sample initial solution $Y(0) = y_1, y_2 \dots y_n$ from $N(0,10^{-4}I)$;
    for $t = 1\ to\ T$ do
        Compute low-dimensional affinities $qij$ using Equation0.2;
        Compute gradient $\frac{\delta C}{\delta Y}$ using Equation 3.3;
        Set $Y^{(t)} = Y^{(t-1)} + \eta \frac{\delta C}{\delta Y} + \alpha(t)\left(Y^{(t-1)} - Y^{(t-2)}\right)$;
    end
end
```

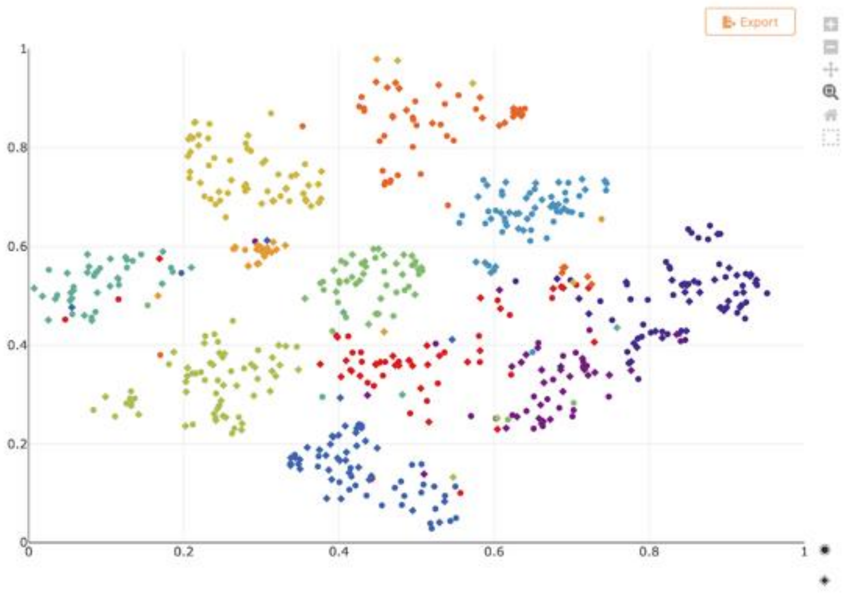
$$p_{ij} = \frac{\exp(-|x_i - x_{j\cdot}|^2)/2\sigma_{i^2}}{\Sigma_{k\neq i}\exp(-|x_i - x_{k\cdot}|^2)/2\sigma_{i^2}} \qquad 0.1$$

$$q_{ij} = \frac{(1 + |y_i - y_j|^2)^{-1}}{\sum_{k\neq l}(1 + |y_k - y_l|^2)^{-1}} \qquad 0.2$$

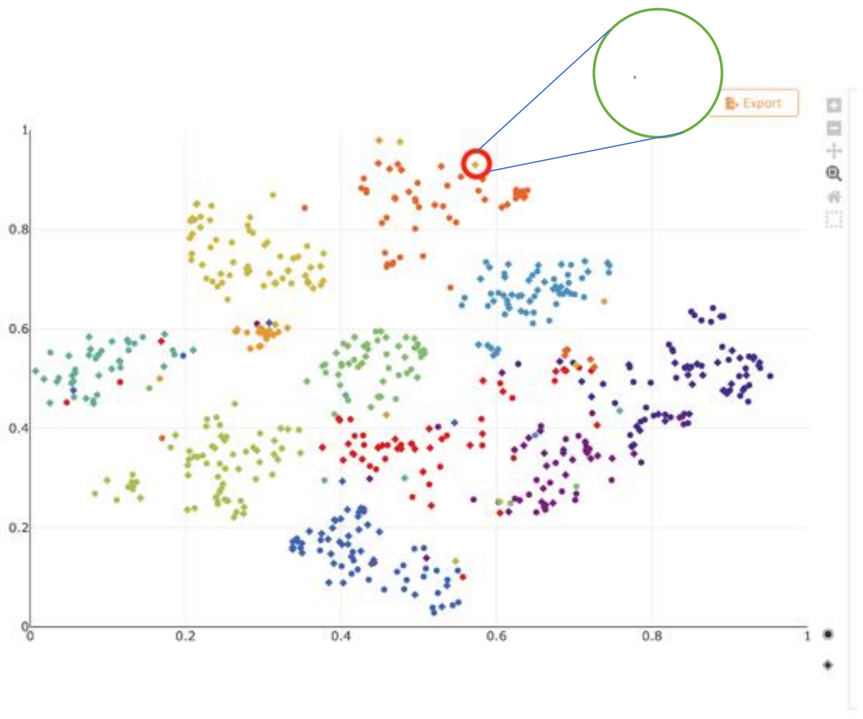
$$\frac{\delta C}{\delta y_i} = 4\Sigma_j(p_{ij} - q_{ij})(y_i - y_j)(1 + |y_i + y_j|^2)^{-1} \qquad 0.3$$

3.4 Exploring Data Coverage using T-SNE

This dimensionality reduction technique finds its way into very powerful visualization methods. Once the datasets (in our case images) are reduced to a 2D plane, we can cluster the images based on their cosine similarities. (**Error! Reference source not found.**).



(a)



(b)

Figure 0-5 T-SNE visualization for various fashion_accessories - (a) the orange concept shows category A and (b) shows a different category image with cosine similarity close to the category A concept misclassified.

3.5 Albumentations

Albumentations is a fast image augmentation library and easy to use wrapper around other libraries. In the training process workflow, we use this library extensively for image preprocessing and augmentation processes. Some of the key features of this library are as follows:

- Accurate, efficient augmentations based on highly-optimized OpenCV library.
- Simple and powerful interface for different tasks like (segmentation, detection, etc).
- Can be adopted to work with various CV frameworks.

We also list below some of the various augmentation used with the training models and their descriptions:

albumentations.augmentations.transforms.Blur(*blur_limit=7, always_apply=False, p=0.5*)

Blur the input image using a random-sized kernel.

Parameters:

- **blur_limit (*int, (int, int)*) –** maximum kernel size for blurring the input image. Should be in range [3, inf). Default: (3, 7).
- **p (*float*) –** probability of applying the transform. Default: 0.5.

Targets:

image

Image types:

uint8, float32

albumentations.augmentations.transforms.OpticalDistortion(*distort_limit=0.05, shift_limit=0.05, interpolation=1, border_mode=4, value=None, mask_value=None, always_apply=False, p=0.5*)

Parameters:	• **distort_limit (*float, (float, float)*) –** If distort_limit is a single float, the range will be (-distort_limit, distort_limit). Default: (-0.05, 0.05). • **shift_limit (*float, (float, float))*) –** If shift_limit is a single float, the range will be (-shift_limit, shift_limit). Default: (-0.05, 0.05). • **interpolation (*OpenCV flag*) –** flag that is used to specify the interpolation algorithm. Should be one of: cv2.INTER_NEAREST, cv2.INTER_LINEAR, cv2.INTER_CUBIC, cv2.INTER_AREA, cv2.INTER_LANCZOS4. Default: cv2.INTER_LINEAR**.**

- **border_mode (*OpenCV flag*)** – flag that is used to specify the pixel extrapolation method. Should be one of: cv2.BORDER_CONSTANT, cv2.BORDER_REPLICATE, cv2.BORDER_REFLECT, cv2.BORDER_WRAP, cv2.BORDER_REFLECT_101. Default: cv2.BORDER_REFLECT_101
- **value (*int, float, list of ints, list of float*) –** padding value if border_mode is cv2.BORDER_CONSTANT.
- **(int, float, (*mask_value*) –** list of ints, list of float): padding value if border_mode is cv2.BORDER_CONSTANT applied for masks.

Targets:

image, mask

Image types:

uint8, float32

albumentations.augmentations.transforms.HorizontalFlip(*always_apply=False, p=0.5*)

Flip the input horizontally around the y-axis.

Parameters: **p (*float*)** – probability of applying the transform. Default: 0.5.

Targets:

image, mask, bboxes, keypoints

Image types:

uint8**,** float32

albumentations.augmentations.transforms.RandomSizedCrop(*min_max_height, height, width, w2h_ratio=1.0, interpolation=1, always_apply=False, p=1.0*)

Crop a random part of the input and rescale it to some size.	
Parameters:	• **min_max_height ((*int, int*))** – crop size limits. • height (*int*) – height after crop and resize. • **width (*int*)** – width after crop and resize. • **w2h_ratio (*float*)** – aspect ratio of crop. • **interpolation (*OpenCV flag*)** – flag that is used to specify the interpolation algorithm. Should be one of: cv2.INTER_NEAREST, cv2.INTER_LINEAR, cv2.INTER_CUBIC, cv2.INTER_AREA, cv2.INTER_LANCZOS4. Default: cv2.INTER_LINEAR. • **p (*float*)** – probability of applying the transform. Default: 1.
Targets: image, mask, bboxes, keypoints	
Image types: uint8, float32	

Table 3-1 A list of frequently augmentations used in the project, along with their example and parameters

3.6 Web Scraping

3.6.1 Introduction

The World Wide Web is currently the largest data source in the history of mankind [24]and consists of mostly unstructured data, which can be hard to collect. Extracting the World Wide Webs unstructured data can be done with traditional copy-and-paste, as some websites provide protection against an automated machine accessing the website. However, this is a highly inefficient approach for larger projects [25]. Sometimes websites or web services offer APIs to fetch or interact with the data. However, it is not uncommon that APIs are absent or that the available solutions do not cover the user's needs. Using APIs also require some programming skill [26]. If APIs are not available or are insufficient for the task, a technique known as web scraping can be applied. Web scraping, also known as web data extraction, web data scraping, web harvesting or screen scraping [27] can be defined as

"A web scraping tool is a technology solution to extract data from web sites in a quick, efficient and automated manner, offering data in a more structured and easier to use format" [24]

In essence, web scraping is used to fetch unstructured data from web pages and transform it to a structured presentation, or for storage in an external database. It is also considered an efficient technique for collecting big data, where gathering large amounts of data is important [28]. Search engines use web scraping in conjunction with web crawling to index the World Wide Web, with the purpose of making the vast number of pages searchable. The crawlers, also called spiders, follow every link that they can find and store them in their databases. On every website metadata and site contents are scraped to allow for determining which site best fit the users search terms. One example of a way to "rank" the pages is by an algorithm called PageRank[1] . PageRank looks at how many links are outgoing from a website, and how often the website is linked from elsewhere. Three different phases build up web scraping:

Fetching phase

First, in what is commonly called the fetching phase, the desired web site that contains the relevant data has to be accessed. This is done via the HTTP protocol; an Internet protocol used to send requests and receive responses from a web server. This is the same techniques used by web

[1] *https://www.google.com/intl/ALL/search/howsearchworks/*

browsers to access web page content. Libraries such as **curl**[2] and **wget**[3] can be used in this phase by sending an HTTP GET request to the desired location (URL), getting the HTML document sent back in the response [12].

Extraction phase

Once the HTML document is fetched, the data of interest needs to be extracted. This phase is called the extraction phase, and the technologies used are regular expressions, HTML parsing libraries or XPath queries. XPath stands for XML Path Language and is used to find information in documents [19]. This is considered the second phase.

Transformation phase

Now that only the data of interest is left it can be transformed into a structured version, either for storage or presentation [12]. The process described above can be summarized in Figure 0-6

[2] https://curl.haxx.se/

[3] https://www.gnu.org/software/wget/

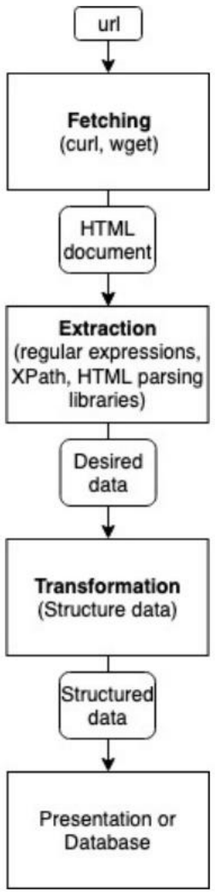


Figure 0-6 Web Scraping Process Flow

3.7 Challenges

There are some challenges that arise when doing web scraping. As the Internet is a dynamic source of data, things may change. This includes site structure, the technology used and whether the website follows standards or not. When a website changes it is likely that a built web scraper needs to be altered as well [27]. Other challenges include unreliable information and ungrammatical language. Even if the information exists and is scrapable, it might not actually be correct. Grammar and spelling can be an issue in the parsing phase, as information might be missed or falsely gathered [19]. While there has been constant evolvement in development of web scraping tools, the legalities concerning web scraping are somewhat unexplored. Some legal frameworks can be applied, a websites terms of use can state that programmatic access should be denied and can lead to a breach of contract if broken. A problem with this however is that the website user needs to explicitly agree to these terms and unless the web scraper does this, the breaching may not be able to lead to a prosecution. If prohibited material is obtained and used can lead to prosecution as well as overloading the website can lead to charges [15]. The ethical side is for the most part ignored. For example, a research project that involves collecting large amounts of data via web scraping might accidentally compromise the privacy of a person. A researcher could match the data collected with another source of data and thus reveal the identity of the person who created the data. Companies and organizations privacy can be unintentionally revealed via trade secrets or other confidential information through web scraping. It can also harm websites that rely on ads for income, since utilizing a web scraper causes ads to be viewed by software and not a human [15].

3.7.1 Tools and techniques

There are several approaches one can take when implementing a web scraper. A common path is to use libraries. Using this approach, the web scraper is developed in the same vein as a software program using a programming language of choice. Popular programming languages for building web scrapers include Java, Python, Ruby or JavaScript, in the framework Node.js [6]. Programming languages usually offer libraries to use the HTTP protocol to fetch the HTML from a web page. Popular libraries for using the HTTP protocol include *curl* and *wget*. After this process regular expressions or other libraries can be used to parse the HTML [12]. Using a web scraping framework is an alternative solution. These frameworks usually take care of each step in the web scraping process, removing the need to integrate several libraries. For example, a Python framework called *Scrapy* allows defining web scrapers as inheriting from a *BaseSpider* class that provides functions for each step in the process. Some popular libraries for implementing web scrapers include *Jsoup*, *jARVEST*, *Web-Harvest* and *Scrapy* [12]. If programming skills are absent, desktop-based environments can be used. These allow for creating web scrapers with minimal technical skills, often providing a GUI-based usage with an integrated browser. The user can then navigate to the desired web page and simply point-and-click on data that should be fetched. This process avoids the use of XPath queries or regular expressions for picking out data [12]. Web scraping tools can be split up into two subgroups, partial and complete tools. Partial tools are often focused on scraping one specific element type and can come in the form of a browser extension. It is more light weight and requires less configuration. Complete tools are more powerful and offer services such as a GUI, visual picking of elements to scrape, data caching and storage [6].

XPath

XPath[4], which stands for XML Path Language, is used to access different elements of XML documents. It can be used to navigate HTML document as HTML is an XML-like language and shares the XML structure. XPath is commonly used in web scraping to extract data from HTML documents and uses the same notation used in URLs for navigating through the structure. Some examples are shown using the file in Listing 2.1, describing a small book store taken from the Microsoft .NET XML Guide[5].

```xml
<?xml version='1.0'?>
<bookstore xmlns="urn:newbooks-schema">
<book genre="novel" style="hardcover">
		<title>The Handmaid's Tale</title>
		<author>
				<first-name>Margaret</first-name>
				<last-name>Atwood</last-name>
		</author>
		<price>19.95</price>
</book>

<book genre="novel" style="other">
```

[4] https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Selectors

[5] https://docs.microsoft.com/en-us/dotnet/standard/data/xml/select-nodes-using-xpath-navigation

```
		<title>The Poisonwood Bible</title>
		<author>
			<first-name>Barbara</first-name>
			<last-name>Kingsolver</last-name>
		</author>
		<price>11.99</price>
</book>

<book genre="novel" style="paperback">
		<title>The Bean Trees</title>
		<author>
			<first-name>Barbara</first-name>
			<last-name>Kingsolver</last-name>
		</author>
		<price>5.99</price>
</book>

</bookstore>
```

Figure 0-7 An XML file describing a book store

Following examples are ran using the command line tool xpath on the file shown in Figure 0-7.

Running the command *//book* will extract all the book elements in the document, as shown in Figure 0-8.

```
<book genre="novel" style="hardcover">
		<title>The Handmaid's Tale</title>
		<author>
			<first-name>Margaret</first-name>
			<last-name>Atwood</last-name>
		</author>
		<price>19.95</price>
</book>

<book genre="novel" style="other">
		<title>The Poisonwood Bible</title>
		<author>
```

```xml
                <first-name>Barbara</first-name>
                <last-name>Kingsolver</last-name>
        </author>
        <price>11.99</price>
</book>

<book genre="novel" style="paperback">
        <title>The Bean Trees</title>
        <author>
                <first-name>Barbara</first-name>
                <last-name>Kingsolver</last-name>
</author>
```

Figure 0-8 The resulting XML when fetching all book elements

Headless browsers

Many popular browsers offer a way to run in headless mode. This means that the browser is launched without running its graphical interface. A common use case of utilizing a headless mode browser is for automated testing, but it can also be used for web scraping. Benefits of using a headless browser include faster performance, as the CSS, HTML and JavaScript code needed to render the web page can be avoided. One drawback that can be encountered with a browser running in headless mode is that websites may be configured to try preventing automatic access of its content. If a non-headless browser is utilized, this is not a problem, it will behave as if a human is controlling it.

3.7.2 Survey of Web Scraping tools

In this section twelve state of the art tools used for web scraping are presented. They have been found through searching the web or having heard about them due to their popularity. These twelve tools are considered for the evaluation moving forward.

Scrapy

Scrapy is an open source Python framework originally designed solely for web scraping, but now also supports web crawling and extracting data via APIs. XPath or CSS selectors is the built in way to do the data extraction, but external libraries such as BeautifulSoup and lxml can be imported and used. It allows for storing data in the cloud[6].

Selenium

A way of automating and simulating a human browsing with a web browser can be accomplished by using a tool called Selenium. It is primarily used and intended for testing of web applications, but is a relevant choice for web scraping. Using the Selenium WebDriver API in conjunction with a browser driver (such as ChromeDriver for the Google Chrome browser) will act the same way as if a user manually opened up the browser to do the desired actions. Because of this, loading and scraping web pages that makes use of JavaScript to update the DOM is not a problem. The Selenium WebDriver can be used in Java, Python, C#, JavaScript, Haskell, Ruby and more[7].

Jaunt

Jaunt is a web scraping library for Java that, compared to Selenium, uses a headless browser. It allows for controlling HTTP headers and accessing the DOM, but does not support JavaScript.

[6] *https://scrapy.org/*
[7] *https://www.seleniumhq.org/*

Because of lacking the ability to work with JavaScript it is more lightweight, faster and can be easily scaled for larger projects. If JavaScript is a must, a crossover between Jaunt and Selenium called Jauntium[8] can be used. The idea with Jauntium is to overcome both the limitations of Jaunt and Selenium and keep the design and naming philosophy similar as with Jaunt, making for an easy transition over if needed. If JavaScript is not needed however, regular Jaunt is recommended[9].

jsoup

jsoup is an open source, Java based tool for manipulating and extracting HTML data. It implements the WHATWG HTML5 11 standard and offers DOM methods for selecting HTML elements as well as CSS Selectors and jQuery-like extraction methods. It is designed to deal with any type of HTML, malformed or not[10].

HtmlUnit

HtmlUnit is a headless Java browser allowing commonly used browser functionality such as following links, filling out forms and more. Similarly, to other headless browsers it is typically used for testing web applications but can also be used for web scraping purposes. The goal is to simulate a "real" browser, and thus HtmlUnit includes support for JavaScript, AJAX and usage of cookies[11]. HtmlUnit code is written inside JUnit[12] tests, which is a popular framework for Java unit testing.

PhantomJS

[8] *https://jauntium.com/*
[9] *https://jaunt-api.com/*
[10] *https://jsoup.org/*
[11] *http://htmlunit.sourceforge.net/*
[12] *https://junit.org/junit5/*

PhantomJS is a headless web browser that is controlled with JavaScript code. It is open source and is commonly used to run browser-based tests but can be used for any type of automated browser interaction. QtWebKit is used as the back end and supports fast, native web standards such as CSS selection, DOM handling, JSON, Canvas and SVG. As of recently, PhantomJS is no longer being maintained[13].

Puppeteer

Puppeteer is an open source tool maintained by the Google Chrome DevTools team, used to control a headless Chromium (or Chrome, if configured) instance. It is used as a Node.js module and is thus written and configured by JavaScript code. The intention of Puppeteer is to present a browser automation tool that is fast, secure, stable and easy to use[14].

CasperJS

CasperJS is a JavaScript tool for web automation, navigation, web scraping and testing. It is used as a Node.js module and uses a headless browser. CSS Selectors and XPath queries are used for content extraction. However, as of now, it is no longer being maintained[15].

Nightmare.js

Nightmare.js, a JavaScript tool, was originally designed to perform tasks on web sites that do not have APIs but has evolved into a tool that is often used for UI testing and web scraping. Exposing

[13] *http://phantomjs.org/*
[14] *https://www.chromium.org/Home*
[15] *https://pptr.dev/*

just a few simple methods (goto, type and click) makes for a simple yet powerful API[16]. Under the hood it uses Electron[17] as the browser.

BeautifulSoup

BeautifulSoup is a Python HTML extracting library. While not being a complete web scraping tool, it can be used in conjunction with the requests package, which allows for doing HTTP calls in Python. BeautifulSoup is a simple, pythonic way to navigate, search and modify parse trees, such as an HTML tree. It is intended to be easy to use and provides traversal functionality such as finding all links or all tables matching some condition.

MechanicalSoup

MechanicalSoup is a combined solution of BeautifulSoup in conjunction with the requests Python package. It is written and configured by Python code. It does not support JavaScript loaded content but can automatically send cookies, follow browser redirections and links and supports form submissions.

rvest

rvest is a web scraping tool for the statistic-focused programming language R. Its intention is to be easy to use and claims to be inspired by Python web scraping tools such as BeautifulSoup and RoboBrowser. It allows usage of both CSS Selectors and XPath queries for element extraction

[16] *http://www.nightmarejs.org/*
[17] *https://electronjs.org/*

Chapter 4
ViSenze Processes & Development Environment

4.1 Process Workflow

This section describes briefly the process workflow for building image classification models at ViSenze. Again, for confidentiality reasons, a specific, detailed explanation of the processes and internal tools are redacted. We do however see many of the data science processes in action and how they fit into the entire production workflow. ViSenze uses the industry standard AGILE methodology for managing the development & releases.

4.1.1 Problem Definition

Taxonomy

Taxonomy is the definition of the available concepts and structure and relationships between them. Exactly one taxonomy must be followed in each classification and detection dataset. One single taxonomy is usually represented as a JSON dictionary. **Error! Reference source not found.** illustrates an example of a taxonomy for single concept-group multi-label problem.

Some other terminology related to taxonomy and defining a classification project is listed below:

```
{
  "concepts": [
    "chair",
    "table",
    "sofa"
  ],
  "concept_groups": [
    {
      "name": "furniture",
      "type": "MULTI_LABEL"
    }
  ],
  "cascade_relations": []
}
```

Table 4-1 Sample definition of a taxonomy, in this case, for the concept group furniture

Concepts: defines the label information

Concept_groups: defines a virtual group of concepts and the problem type of the group.

Cascade_relations: A taxonomy may contain cascade relationships.

Other taxonomy examples:

- single concept_group, multi-class
- single concept_group, multi-label
- multiple concept_groups, multi-class
- multiple concept_groups with cascade relations

Guideline

One guideline that contains the detailed definition of each concept must be there in each taxonomy. A guideline is generally a list of slides which tries to explain all kinds of details for each concept. The algorithm engineer is supposed:

- view the guideline carefully
- make sure one understands the definition of each concept
- gives some feedbacks on the guideline whenever he/she feels it is confusing, unclear or incomplete.

The guideline won't be complete in the first place, in most cases. It always needs incremental improvement and complement.

4.1.2 Dataset

Types of dataset

It's recommended to have four datasets for each tagging project.

- **Training set and Training Dev Set (validation set):** These two datasets are split from the same dataset built by an algorithm developer so they should be from the same distribution. Training dev set usually only contains enough data samples for model evaluation. Most data samples should go to training set for model training. **Training set** is the dataset used to train the model.

- **Training dev set (validation set)** is the dataset used to evaluate the model during the learning process. This dataset would be used to tune parameters, select features, and make other decisions regarding the learning algorithm.

- **Product dev set and product test set:** When the model is in production, these two datasets are usually provided by the product team as the representation of the real-world problem. They are assumed to be from the same distribution. The only difference between them is that algorithm developers is permitted to look at the content of the product dev set but not the product test set.

  This is because algorithm engineers are expected do error analysis on the product dev set and improve model performance based on the analysis. The model is therefore highly likely to overfit the product dev set. This makes the model performance on the product dev set not trustworthy anymore. It is required to keep one set untouched from the beginning and only use it to evaluate model performance, to approximate the model performance in production environment.

- **Dev set** is like Training dev set except that it is provided by product team and assumed to be from the same distribution as the test set.

- **Test set** is used to check whether the model performance meets product acceptance standard.

4.1.3 Component of a dataset

A dataset contains a set of ground-truths. Ground-truth defines the unique level of a dataset.
Each ground-truth should, for classification problem, contain:

- one image
- some labels (concepts, neg_concepts, skip_concepts)
- (optional) a bounding box

Each ground-truth should, for detection problem, contains:

- one image
- one or more objects, each object should contain:
- one box
- some labels (concepts, neg_concepts, skip_concepts)

4.1.4 Dataset general process

A data science engineer can be required to build his/her own dataset under certain situations – if the model he/she is working on is a brand new one, or the previous datasets are lost or disqualified furthermore. Or one may want to clean up the current dataset further because one finds some issues emerging from it.

Some general processes related to dataset are summarised. All dataset related processes aim to collect or modify parts of the ground-truths:

- **image**: collect more images through data sourcing
- **box**: add bounding boxes to the dataset by applying our product_detection model on it
- **label**: add labels to the dataset or modify the current labels through annotation tasks

4.1.5 Data Sourcing

The aim of data sourcing is to collect images. Some potential ways for data sourcing:

- write local scripts to crawl images from certain data sources
  - potential tools: BeautifulSoup, Selenium, etc.
- use internal data crawling tools.

A detailed discussion on web scraping and various tools used for scraping images from the web is discussed in chapter 3.

4.1.6 Apply product detection

Building a dataset with bounding boxes around the interested objects is recommended. One can apply a product detection model on the dataset to obtain the bounding boxes. The returned result contains box coordinates, tag and score for each of the object detected. One can apply customised process and get the final result, based on the returned information.

For example, to add bounding boxes for the toe_shape training set, we would:

- Apply product detection on the original dataset
- Only keep the bounding boxes with tag shoe. In this way, some images might not have any valid bounding box, you can choose to remove all those images or just keep them as full images
- Choose among the valid bounding boxes, for which possible strategies could be:
    - keep all valid bounding boxes (one image might have multiple bounding boxes)
    - for each image only keep the box with the highest score (one image only has one bounding boxes at most) etc.

4.1.7 Annotation

After getting enough images as well as bounding boxes (optional) for the dataset, annotation tasks need to be created to get or modify labels. After the annotation, a wiki page needs to be created to record the dataset creation process.

4.1.8 Model training

To train models, one needs to prepare and put all the following materials on the same:

- Taxonomy
- Dataset - You need to at least have both **training set** and **training dev set** (validation set) on the platform.

A model can be trained using various general training workflows for some common SOTA classification models. A brief list of steps in the training workflow is as follows:

- Prepare a train, validation or test dataset
- Train a deep learning model
- Test a trained model
- Build a trained model to a release version

4.1.9 Model evaluation

After a model is obtained, one can evaluate the model performance on customized test sets. Theoretically one can evaluate the model on any dataset with ground-truth labels. For example, the test set could be one of training set, training dev set, product dev set and product test set.

The evaluation workflow consists of model evaluation workflow, which would inference the model on the given test set. The evaluation system provides a simple result browsing and some filters to better understand multi-class performance and misclassifications made by the model. This is an important step during error analysis (as discussed)

4.1.10 Model Improvement

After training a model is finished, one will always think about how to improve the model performance further. Here are some potential strategies to help spot the issues of the current model.

Data coverage analysis

Sometimes it is not known whether the training set has covered most of the common cases for the problem. The training set might be biased towards certain sources and thus lack of diversity. Or it could totally miss out some parts of the whole data space. If it is realized that the test set distribution is highly different from their training set, one shouldn't expect their model to perform well.

Data coverage analysis gives a way to explore how different is the current test set from the training set. If severe data coverage problem is spotted, where a large portion of the testing samples are not covered by the training set, it is advisable to enlarge the training set coverage by adding certain amount of data samples to the uncovered space and then re-train a model on it. It's assumed that a more diverse training set could help the model generalize better.

In summary, data coverage analysis helps to answer the following questions:

- Whether the test set is similar to the training set
- If you want to improve the training set data coverage, what kind of data samples should you add

Confusion area analysis

If the prediction result in examined in detail, the model may be found to keep making mistakes for certain testing samples or keeps confused between two classes. The reasons underneath could be complicated, the most common one we found is that, the problem itself is very difficult and the guideline is not clear enough to help annotators differentiate between the confusing cases. Thus, data samples in the confusion area are not well annotated and model also can't learn how to draw clear boundaries. If the testing samples fall right on these areas, the model won't be confident about its prediction.

Under these circumstances, it is advisable to find out these confusion areas, sample a part of data points from these areas and clean up the labels through annotation tasks. Besides, it's necessary to give feedback on the current guideline based on the discovery and require the product team to refine the guideline further.

Detailed error analysis

Error analysis refers to the process of examining dev set examples that your algorithm misclassified, so that you can understand the underlying causes of the errors [29].

Apart from the high-level evaluation, it is advised to do error analysis in detail to understand the types of error. So that one can prioritize the effort on solving the severe ones. We are trying to follow the error analysis method suggested by Andrew Ng in his book Machine Learning Yearning [29]. Some of the key takeaways are:

- Not starting off trying to design and build the perfect system - Instead building and training a basic system quickly
- Carrying out error analysis by manually examining ~100 examples from the dev set that the algorithm misclassifies and counting the major categories of errors. We use this information to prioritize what types of errors to work on fixing.
- We consider splitting the dev set into an “Eyeball” dev set, which we manually examine, and a Blackbox dev set, which we will not manually examine. If the Eyeball dev sees an improvement over the Blackbox dev, we have overfit the Eyeball dev set and should consider acquiring more data for it.

4.1.11 Deliverables

- The classification model.
- Detection model to use
- Threshold file for the model
- Tag mapping file when you have some mapping relation between tags (optional)
- A yaml file indicating the necessary configuration for model deployment

4.1.12 Model deployment

Release a Recognition model

After finishing building the model, one should follow the steps below to deploy the model:

- Pre-release evaluation of the model
- Deploy the model into production
- Post-release check

4.1.13 Product Acceptance

After the model is pushed online, and the algorithm engineer finishes the post-release check, PM needs to be notified and involved into the product acceptance check. In detail, the algorithm engineer needs to:

- Create a wiki page to note down the necessary information about the released model.
- Send the release email to inform both the product team and algorithm team.

4.2 AGILE Best Practices

While not directly pertinent to the project, I believe it's important to understand some of the AGILE best-practices which are adopted for a data science for project management and meeting targets and customer objectives.

4.2.1 Iterative Development

Instead of tackling a big problem – like building a classification model, as a single block of work, it is instead broken into several subtasks, and are distributed and managed in in repetitive cycles. Through this process, the developers get a perspective of the new developments and the different features that need to be added which contribute towards a more flexible product development.

4.2.2 Daily Meetings

This is one of the key features of an AGILE framework. Regular meetings, which are concise and have fixed agendas allows monitoring of team's performance and check if there are any obstacles in the way of product development. In these meetings, each member of the team explicitly states the progress of the tasks and what needs to be done. These meetings could take form as stand-up meetings, sprint meetings, project meetings, etc.

4.2.3 Using Jira and other professional tools

Planning scrums and sprints are some of the most common strategies to implement the AGILE best practices. These can benefit greatly with the use of professional tools such as Jira. Jira lets the teams plan their sprints and scrums through various tools such as Kanban boards, visualizing tools for workflows, etc. It also lets the team manage their tasks through tickets, which can be assigned to the members based on their duties and responsibilities.

Chapter 5
Fashion Accessories Model

fashion_accessories is one of the most searched items in ViSensze's query logs. However, it was found that there was still plenty of room for improvement for this category. The then online model (which is one cascade level above fashion_accessories) could only recognize 16 types of (jewelries, headwear, belt, gloves, etc.) as one concept. We target to expand the concept with a more detailed taxonomy.

A dedicated fashion_accessories model is desirable as this can serve as a cascade logic model for several top-level tag groups, as well as be deployed as a single model for internal use. Moreover, the evaluation datasets generated for the classification models can be used as a standalone dataset for benchmarking various models' performance on fashion_accessories.

In this project, we are tasked to build a new deep learning classification model. This includes the workflow discussed in section 4.1 – problem definition, guideline building, compiling and building evaluation datasets, sourcing training datasets, and multiple iterations of error analysis, updating training sets, as well as changing model parameters and architectures to improve model performance and mitigate any data coverage issues.

Evaluation datasets further include 3 datasets from 3 different domains - product dev sets, product test sets, and query log sets for various domains i.e. User Generated Content (UGC), Professionally Generated Content (PGC), and Marketplace datasets (these are explained in detail in section 4.1.2).

5.1 Datasets

5.1.1 Evaluation Datasets

- UGC (User Generated Content)
- PGC (Professionally Generated Content)
- Marketplace (Images from online marketplace)

The tagging model must perform well on all 3 domains of images to be deemed robust. Dividing the evaluation into these different tests would also allow for a better understanding on where the model falls short.

5.1.2 Summary of Evaluation Datasets Released

Dataset Name	**Sources**
fashion_accessories Product Dev Set	1. Internal datasets for previous models 2. Various eCommerce websites
fashion_accessories Product Test Set	Same distribution as dev set
fashion_accessories Querylog Set	1. Internal datasets for previous models 2. Querylog exported from new queries

Table 5-1 Summary of Evaluation Datasets Released

5.1.3 Training Datasets

Through several rounds of process iterations, data coverage issues are identified and mitigated. After each iteration, the training dataset is cleaned and/or appended more images to, depending upon the error analysis. Each iteration, thus, requires the training set to be updated. A brief summary of the various key updates is listed under table Table 5-2.

Dataset Version	Source/update details
v1	**Source -** Crawled from eCommerce websites
v2	• Added more images for the following concepts: o **suspenders** o **others_fashion_accessories**
v3	Cleaning **others_fashion_accessories/wristband**
v4	Added more images of buckles for the concept **belt_and_buckle**
v5	• Resampled from v4 for balanced train/val sampling
v6	• Added more images for the the following concepts: o **suspenders** o **hair_accessories** o **belt_and_buckle** (only buckles)

Table 5-2 Summary of key training set updates for fashion_accessories model

5.2 Training Summary

Once the training/validation datasets are consolidated, we decide a model architecture – at ViSenze popularly used classification models are ResNet-50/ResNet-101 and MobileNet. These are discussed in detail in Chapter 3. We then use the internal training platform to adjust model parameters, and start training. As discussed in Chapters 3 and 4, the model architectures only afford us marginal improvement in our model performance. Hence, we keep a default parameter configuration for initial model builds, and focus solely on data coverage issues in the models. When the performance nears the target performance, and there are no more major data bias/data insufficiency in the training set, we move to try various preprocessing methods, as well as changing model parameters (like batch size, learning rate, dropout rates, and optimizer methods).

Table 5-3 summarizes the various model iterations – including the CNN architecture used, dataset updates, any change in the default model parameters, and the corresponding validation performance.

Iteration No. & Finish Time	**Parameter Change (from previous iterations)**	**Performance Metric <F1-Score>**
Iteration 5 10/02/20 16:22	**model**: resnet101 **parameters/preprocessing changed**: • Dropout rates • Training augmentations o *scale_rotate_resized_crop*	0.9152
Iteration 6 11/02/20 14:05	**model:** mobilenet_v2 **parameters/preprocessing changed**: • dropout_rates • train.num_epochs • train.optimizer • train.augmentation	0.9254

	○ scale_rotate_resized_crop	
Iteration 7-8 12/02/20 21:56	**model**: mobilenet_v2 **Dataset Updated:** v6 **parameters/preprocessing changed**: • train.augmentation ○ course_dropout ○ jpeg_compression ○ optical_distortion	0.9219
Iteration 9 13/02/20 10:50	**model:** ResNet-50	0.9284

Table 5-3 Summary of model parameters and corresponding performance on the validation set for fashion_accessories model

5.3 Experiment Summary

This section summarizes the key experiments performed on the trained models and their brief analysis, as well as the key issues addressed in each iteration. The workflow followed for every iteration remains the same as discussed in section 4.1.

5.3.1 Iteration 1 - Data Insufficiency

Key issues addressed:

- Data insufficiency for **others_fashion_accessories** (and sub-levels)
- Data insufficiency for **suspenders**

Training Dataset change/training data set name if any:

- Add images for **others_fashion_accessories**
- Add images for **suspenders**

Summary of sub-method experiments

Methods	Evaluation Summary
Sourcing more training data for **others_fashion_accessories**	Recall and Precision for **others_fashion_accessories** increases
Sourcing more training data for **suspenders**	Recall and Precision for **suspenders** increases

Table 5-4 Summary of changes in model iteration 1

5.3.2 Iteration 2: Data Augmentation

Key issues addressed

- Improve learning by adding rotation jitter
- Using bigger network (resnet101 instead of mobilenet)

Summary of sub-method experiments

Methods	Evaluation Summary
Resnet101 (instead of mobilenet) Rotation_jitter (parameter changed)	Overall validation F1 increases
Mobilenet Rotation_jitter (parameter changed)	Overall validation F1 drops from previous mobilenet model
Mobilenet Rotation_jitter (parameter changed)	No change in either validation or test results

Table 5-5 Summary of changes in model iteration 2

5.3.3 Iteration 3: Augmentation and Model Architectures

Key issues addressed

- Data coverage - bad cases - **suspenders, hair_accessories, belt_and_buckle**
- Using bigger network
- Augmentations to improve learning

Training Dataset change/training data set name if any:

- Add bad case images for **suspenders, hair_accessories, belt_and_buckle**

Summary of sub-method experiments

Methods	**Evaluation Summary**
More pgc/mkp images of suspenders added to training set	Recall and Precision for suspenders increases
Augmentations used: - name: optical_distortion - name: jpeg_compression - name: coarse_dropout	Overall accuracy and F1 of model improves
Using resnet101 instead of mobilenet	The overall F1 improves

Table 5-6 Summary of changes in model iteration 3

5.3.4 Iteration 4: Model Architecture to ResNet-50

Key issues addressed

- Used small network for faster inferencing (resnet50 instead of resnet101)

Summary of sub-method experiments

Methods	Evaluation Summary
Using resnet50 instead of resnet101	Decreased evaluation and validation performance - Validation F1 drops (trade-off for faster inferencing)

Table 5-7 Summary of changes in model iteration

Chapter 6
View Angle Models

Many of ViSenze's clients are e-commerce websites in the fashion domain; these are companies that wish to either use ViSenze's recognition platform to better tag and organize their product catalogues, or use ViSenze's image search engine to offer their users an image search option on their websites or mobile apps. It is thus valuable to have models which can be used to identify (classify) the angles of products given in the images.

These models can be used to profile the various attributes of a given dataset, which could be useful when training other models. Since "view angles" could have different meanings for different products (for example – for a t-shirt, the concepts "top view" and "bottom view" do not keep any significance), we develop 3 models corresponding to 3 different problem definitions and guideline, namely – shoe angle, bag angle, and apparel angle. We discuss briefly the challenges involved with these particular type of classification models, as well as a summary of experiments involved with developing the same in future sections.

6.1 Challenges involved

There was one particular challenge with these model types, as opposed to a regular classification model – the lack of pre-labelled or annotated images. Most of the training set for classification models (as discussed in previous sections) comes from crawling e-commerce websites. These websites usually show products from a search result as a list in the XML source of the webpage, with some label for each image, which is usually the title for the product page. However, most e-commerce websites do not provide meta-labels for these images, and these images, since they are thumbnails, are usually picked to showcase the most prominent product angle. For example, the most prominent view angle for a shoe would be the side angle. As a result, we are unable to use pre-existing scripts to gather

6.2 Evaluation Datasets

Product Dev Set:

Sources:

- Querylog Q3 2019
- Shoe_angle dev/test sets

Product Test Set

Sources:

- Querylog Q3 2019
- Shoe_angle dev/test sets

6.3 Experiment Summary

The following table (Table 6-1) summarizes the key experiments undertaken for the shoe_angle model, along with the key parameter changes.

Finish Time and Version	**Training Dataset**	**Parameter Change**	**Organic Dev Set Performance (<metric>)**
Revision 1	v1	Reproduced old version and old dev set	0.8052
Revision 2 -	v2	• Switched to **resnet-50** from **mobilenet_v2** • Updated evaluation datasets • Image augmentations: • optical_distortion • jpeg_compression • coarse_dropout	0.8381

Revision 3	v3	Switched to resnet-101 from resnet-50 tag_group changed, guideline update training_set updated	0.8599

Table 6-1 Experiment summary for shoe_angle model

6.4 Experiment Detail

6.4.1 Experiment 1: Reproduce latest model & evaluation set update

Key issues addressed:

- Reproduce latest model
- Update evaluation sets to include querylog images from Q3 2019

Methods	**Evaluation Summary**
Updated dev set - added latest querylog support	The performance drops from the original dataset because of edge cases and some wrong images in evaluation set

Table 6-2 Experiment 1 – Evaluation Summary for shoe_angle model

6.4.2 Experiment 2: Augmentation and ResNet-50

Key issues addressed:

- General improvement to reach target F1
- Cleaning dev/test set

Methods	Evaluation Summary
Adopted ResNet-50 architecture	Overall (micro) F1 Score improves
Added various image augmentation methods in the training pipeline	Overall (micro) F1 Score improves

Table 6-3 Experiment 2 – Evaluation Summary for shoe_angle model

6.4.3 Experiment 3: Augmentation and ResNet-50

Key issues addressed:

- Sourced images for bad-cases from e-commerce websites - for flat sandals, sneakers
- Cleaning dev/test set
- General tweaking/improvement to reach target F1

Methods	Evaluation Summary
Sourcing images from asos for the flat sandals, sneakers footwear concepts	Improved recall for side_view and front_angle view
Adopted ResNet-101 architecture	Overall (micro) F1 Score improves Inferencing would be possibly slower.
Added various image augmentation methods in the training pipeline	Overall (micro) F1 Score improves

Table 6-4 Experiment 2 – Evaluation Summary for shoe_angle model

Chapter 7
Conclusion

Through the duration of my employment at ViSenze as a data science intern, I primarily worked on two major projects – the fashion_accessories model, and view angle models (bag_angle, shoe_angle, and apparel_angle). These models were eventually deployed to the ViSenze's internal systems, and have also been used

I learned and consolidated the basics of deep learning and data science, as well as the practical limitations and issues in the domain when adapting at a large scale. By streamlining the process workflow for data management and model training through the various internal tools, rapid prototyping of deep learning models becomes much more efficient. The entire process workflow for deep classification model development – starting from problem definition and defining a valid taxonomy, to sourcing training and evaluation datasets, to training the models, to eventually performing error analysis – needs to be done for several iterations.

One of the biggest learnings for me was working in a software development environment. Since my experience with deep learning and computer vision so far was limited to strictly prototyping and researching in the domain through school projects, it was very revealing to learn how data science, when deployed in production at scale, leverages a lot of the software development processes and best practices. As discussed in chapter Chapter 4, we adopted an AGILE-based project management system, where we set goals for sprints to manage project timelines, discuss blocking problems, and eventually meet product deliveries.

Chapter 8

Future Prospects

- My experience at ViSenze has enabled me as a data scientist to work on deep leaning and computer vision projects at an industrial scale, and has equipped me with the right skillset to work at companies as a data scientist.
- Most of the models from this project, which were deployed still have certain badcases which could be addressed to improve their performance further.
- I am currently also working on improving certain classification models for fashion attributes, namely toe_shape and toe_type. I intend to continue working on the same till the end of May 2020.

Bibliography

[1] I. Goodfellow, Y. Bengio, and A. Courville, "Deep Learning," MIT Press, 2016.

[2] S. Skansi, Introduction to Deep Learning, Cham: Springer, 2018.

[3] D. H. Wolpert, "The lack of a priori distinctions between learning algorithms," *Neural Computation,* vol. 8, no. 7, pp. 1341-1390, Oct 1996.

[4] S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias/variance dilemma," *Neural Computation,* vol. 4, no. 1, pp. 1-58, 1992.

[5] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research,* vol. 15, p. 1929–1958, 2014.

[6] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review,* p. 65–386, 1958.

[7] X. Zhang, S. Ren, J. Sun. K. He, "Deep Residual Learning for Image Recognition," *ArXiv e-prints,* vol. 2015.

[8] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.,* vol. 9, no. 8, p. 1735–1780.

[9] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological Cybernetics,* vol. 43, no. 1, pp. 59-69, January 1982.

[10] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?," *IEEE 12th International Conference on Computer Vision,* p. 2146–2153, 2009.

[11] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout Networks," *ArXiv e-prints,* February 2013.

[12] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature,* vol. 323, no. 533, October 1986.

[13] B. Polyak, "Some methods of speeding up the convergence of iteration methods," *USSR Computational Mathematics and Mathematical Physics,* vol. 4, pp. 1-17, December 1964.

[14] Y. Nesterov, "A method of solving a convex programming problem with convergence rate O(1/k2)," *Soviet Mathematics Doklady,* vol. 27, p. 372–376, 1983.

[15] "On the importance of initialization and momentum in deep learning," Atlanta, Georgia, USA, 2013.

[16] S. Kozielski, D. Mrozek, P. Kasprowski, B. Malysiak-Mrozek and D. Kostrzewa, "Beyond Databases, Architectures, and Structures," *Springer, Cham,* 2014.

[17] A. Krizhevsky, I. Sutskever and F. P. C. J. C. B. L. B. a. K. Q. W. E. C. A. G. E. Hinton in 25, *Imagenet classification with deep convolutional neural networks,* 25, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., Curran Associates, Inc., 2012, p. 1097–1105.

[18] Y. L. Cun, L. Bottou and Y. Bengio, "Reading checks with multilayer graph transformer networks," *IEEE International Conference on Acoustics, Speech, and Signal Processing,* vol. 1, p. 151–154, April 1997.

[19] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," *ArXiv e-prints,* vol. arXiv: 1603.07285, March 2016.

[20] Y. T. Zhou and R. Chellappa, "Stereo matching using a neural network," *ICASSP- 88, International Conference on Acoustics, Speech, and Signal Processing,* vol. 2, p. 940–943, April 1988.

[21] K.He; X. Zhang; S. Ren; J. Sun, "Deep Residual Learning for Image Recognition," *arXiv: 1512.03385v1[cs.CV],* 10 Dec 2015.

[22] "COCO Challenge," [Online]. Available: http://mscoco.org/.

[23] L.J.P. van der Maaten and G.E. Hinton, "Visualizing High-Dimensional Data Using t-SNE," *Journal of Machine Learning Research,* no. 9, pp. 2579-2605, Nov 2008.

[24] O. Castrillo-Fernández, "Web scraping:applications and tools," *Euro- pean Public Sector Information Platform,* 2015.

[25] A. Mehlführer, "Web scraping: A tool evaluation," *Technische Universität Wien,* 2009.

[26] Y. Neil, "Web scraping the easy way.," *Georgia Southern University,* 2016.

[27] S. Sirisuriya, "A comparative study on web scraping," *International Research Conference, KDU,* p. 8:135–139, November 2015.

[28] B. Zhao, "Web scraping," *Oregon State University, 2017,* p. 1–2.