Informatics

Autoencoders as Kolmogorov complexity based distance function in zero-shot learning

Wherein pictures of seahorses improve bird classification

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Logic and Computation

eingereicht von

Dorian Staudt, BSc.
Matrikelnummer 01227505

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.univ.Prof. Dr. Andreas Rauber

Wien, 1. Februar 2020

Dorian Staudt

Andreas Rauber

Informatics

Autoencoders as Kolmogorov complexity based distance function in zero-shot learning

Wherein pictures of seahorses improve bird classification

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Logic and Computation

by

Dorian Staudt, BSc.
Registration Number 01227505

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.univ.Prof. Dr. Andreas Rauber

Vienna, 1st February, 2020

Dorian Staudt Andreas Rauber

Erklärung zur Verfassung der Arbeit

Dorian Staudt, BSc.

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 1. Februar 2020

Dorian Staudt

Kurzfassung

Klassifikationsprobleme leiden oftmals unter einem Mangel an annotierten Trainingsdaten. Dies führte zu der Entwicklung von Zero-shot Learning Modellen, welche mit Klassen trainiert werden für die ausreichend Trainingsdaten zur Verfügung stehen, um dann unbekannte Klassen anhand von Beschreibungen zu erkennen. Oftmals sind diese Beschreibungen in der Form von Attributsvektoren, die allerdings ebenfalls selten zur Verfügung stehen und aufwändig zu erstellen sind. Manche Ansätze nutzen daher stattdessen Beschreibungen in natürlicher Sprache.

In dieser Arbeit wird eine neue Methode zum Vergleich von Daten aus verschiedenen Domänen, die Autoencoder Distance (AD), vorgestellt und getestet in einer Zero-shot Anwendung mit Bilddaten und Beschreibungen in natürlicher Sprache. Die Distanzfunkion basiert auf der Normalised Compression Distance von Cilibrasi und Vitányi, ein Verfahren bei dem verlustfreie Komprimierungsalgorithmen genutzt werden um gemeinsame Muster zu erkennen, in dem die Größe von kombinierten Eingangsdaten nach Komprimierung gemessen werden. Die Messung wird normalisiert mit den Größen der Eingabedaten wenn sie unabhängig voneinander komprimiert werden.
Für die Methode die in dieser Arbeit vorgestellt wird ist statt eines verlustfreien Komprimierungsalgorithmus ein Autoencoder im Einsatz. Dieser wird zuerst darauf trainiert zusammengehörige Eingabedaten zu assoziieren, also Bilder und die Beschreibungen der Klassen denen sie angehören. Die Distanz zwischen Eingabedaten wird dann approximiert indem die mittlere quadratische Abweichung zwischen der Beschreibung und der korrespondierenden Ausgabe berechnet wird. Für die Normalisierung werden für alle Beschreibungen Durchschnitt und Standardabweichung dieser Abweichung für alle Bilder in einem festgelegten Set genutzt.
Zur Klassifikationen eines Bildes werden alle Beschreibungen nach ihrem AD zu diesem Bild gereiht. Das Bild wird dann der Klasse die der erstgereihten Beschreibung entspricht zugeordnet.

Evaluiert wird das Modell anhand einer Variation des Caltech-USCD Vogel-Datensets mit Klassenbeschreibungen von Reed et al. Des Weiteren werden Bildersets von diversen Tieren und Alltagsgenständen zur Normalisierung genutzt.

Beim Klassifizieren mit 50 Beschreibungen die im Training nicht vorkamen konnte eine T1 Genauigkeit von 23,25% und eine T5 Genauigkeit von 57,14% erreicht werden, wobei Bilder von Seepferdchen zur Normalisierung genutzt wurden. Diese Werte sind geringer

als Genauigkeiten die von anderen Werken auf den gleichen Daten erreicht werden konnte, aber durch die neuartige Methode werden viele bisher unerforschte Ansätze für zukünftige Entwicklungen eröffnet.
Als ein Nebenziel wird zusätzlich gezeigt, dass die Ausgabe des Autoencoders für Explainability genutzt werden kann.

Abstract

Many classification tasks suffer a lack of labelled data. This led to the development of zero-shot learning models, which are trained on classes with available data to recognise unknown classes from descriptions. Often this is done with descriptions in the form of attribute vectors, but those are again rarely available and expensive to produce. Some approaches therefore use descriptions in natural language instead.

In this thesis a new method of comparing data from different domains, Autoencoder Distance (AD), is introduced and tested on a zero-shot application with image data and natural language descriptions. The distance function is based on the Normalised Compression Distance by Cilibrasi and Vitányi, a method that uses lossless compression algorithms to estimate shared patterns by measuring the size of combined inputs after compression, normalised by the compressed size of the inputs on their own.
For the method introduced in this thesis an autoencoder is used instead of lossless compression. It is first trained to associate related inputs, i.e., images and the descriptions of their class. The distance between inputs is then approximated by calculating the mean squared error between the input description and its reconstruction. Normalisation for each description is done with the mean and standard deviation of this error over a shared set of images.
For classification, descriptions are ranked by their AD to a given image. The image is then placed in the class associated with the top ranked description.

Evaluation is done on a variation of the Caltech-USCD bird dataset with descriptions provided by Reed et al. Further, image sets depicting various animals and commonplace items are used for normalisation.

Classifying by ranking 50 descriptions not encountered in training, a T1 accuracy of 23.25% and a T5 accuracy of 57.14% could be achieved using pictures of sea horses for normalisation. This is lower than what was previously achieved on the same data, but the new method opens many novel avenues for future work.
As a secondary objective it is also shown that the output of the autoencoder can be used for explainability.

Contents

CHAPTER 1

Introduction

1.1 Motivation and Aims

A common problem of classification in machine learning is the lack of labelled training data. As such, many approaches were tried to amplify the use of available data, or to train a model with little to no examples.
Recently one-shot, and especially zero-shot learning have experienced a rise in popularity.

One-shot learning describes the learning of classes by providing a single example and, usually, inferring further information from related classes with more available data. This requires a very broad generalisation, as, e.g., even images of the same object may look wildly different from varying angles.

Over time many methods were developed to extract comparable features. This ranges from learned metrics in early examples [Fin05], to modern approaches with the nowadays ubiquitous neural networks (NN) [Koc15][AZ18].

Zero-shot learning is in many way similar to one-shot learning, but instead of some classes only having few samples available, they lack them altogether. Instead, descriptions of those classes are used.
These descriptions take quite different forms. One very early approach went a very similar route to one-shot learning and used abstract image representations of the classes, e.g., specifically made 7×5 pixel illustrations of numbers and letters to represent handwritten or photographed ones [LEB08].

Many later ones instead use binary vectors that give information whether or not certain attributes are present in a given class. Animals, for example, could have attributes of the form ’brown’, ’white’, ’water’, ’wings’, ’two legs’, etc. [RPT15][XLSA19]
Such attribute vectors simplify zero-shot learning greatly, as they allow to treat the problem as a group of binary classification tasks, one for each attribute.

One noteworthy example even just used a normal classification network with the last matrix M replaced by $M_1 M_2$, where M_2 is a constant that maps attributes to classes. To add new classes, only M_2 had to be replaced [RPT15].

Even though this solves the issue of lacking examples of some classes, it still requires extensive labelling that is not readily available. So while attribute vectors are still commonly used, many newer approaches instead draw on methods of natural language processing (NLP), especially with the recent rise of transformers like BERT (see section 3.3.3).
There is still a lack of datasets with class descriptions of that format, but they can far more easily be generated, by experts of the field the data is related to, or even automatically from, e.g., Wikipedia [BSFS15].

Similar to many one-shot approaches, classification is often done by ranking a list of candidate classes. This means a distance function is used to compare a list of possible descriptors to a given sample, and ordering them by their score. If an actual classification is needed, the class with the closest description can be chosen.

A method that, to the author's best knowledge, has never been used in the context of zero-shot learning, is the use of Compression Distance as a distance function in the above. Compression Distance is based on Kolmogorov Complexity and, in simple terms, describes the difference between the individual complexity of two objects, and the complexity of both combined [CV05].

The Kolmogorov Complexity of an object is the length of its shortest description, a concept more extensively explained in section 2.1. Its exact value, and even a non-trivial lower bound, are uncomputable, making approximations necessary for practical applications [LV93].

Some previous approaches to classification and clustering have done this by using lossless compression algorithms. Since they are essentially made to produce unique, shorter descriptions, the filesize of samples before and after compression as approximation, leading to the term Compression Distance.
Another option is presented by Cilibrasi et al., who compared words and phrases by how many results Google returns in single and combined searches [CV07].

A common factor of these approaches is that they compare objects of the same type, i.e., images with images, text with text, and so on. Compression algorithms typically only consider the binary representation of anything they are applied to, not the actual encoded content.
Any common pattern between an image and a text that would lead to a better compression would therefore be as likely with a text describing the image and an unrelated text of similar length and structure.

That is not the case with autoencoders. An autoencoder trained on combined representations of objects and their descriptions over time learns to encode similar parts of the content in the same way, i.e., a red region in an image might be encoded the same way as the word 'red' in a text.

This thesis proposes a new Autoencoder Distance that approximates Compression Distance between objects of different types. Then this is applied in a zero-shot learning situation with image classes described in natural language. For the latter a dataset of bird species with descriptions provided by Reed et al. in [RALS16].

Using autoencoders this way comes with several challenges and opportunities:

1. Autoencoders normally compress to a constant size.

It is therefore not possible to take the size reduction due to compression as a measure of how many patterns the inputs share.

However, compression with autoencoders is lossy, i.e., information is lost when data is encoded and decoded. The better the process works for a given input, the better the reconstruction in the output. This makes it possible to approximate the combined complexity of both inputs by calculating the difference between input and output.

2. An autoencoder made to take representations of an object and the description of its class as input always requires both inputs.

Some objects are inherently less complex than others. Consequently, the reconstruction error of an image compressed with a very complex description is likely to be higher than that of the same image compressed with a far less complex description, even if it shares more patterns with the former.

When more general compression algorithms are used, this bias can be avoided by considering the compression size of the image and its descriptions on their own [CV07], which is not possible with the autoencoder here.

To resolve this, the reconstruction error of all descriptions combined with all images of a chosen set is calculated. The mean and standard deviation of this error is then used for normalisation.

3. To learn shared patterns, the autoencoder is only presented positive samples during training.

One-shot and zero-shot methods that use distance functions typically require both positive and negative samples during training. Since any combination of an image and a description of a class it does not belong to form a negative sample, they heavily outnumber positive samples. Due to this, it is a common problem which and how many negative samples should be chosen.

When the autoencoder is trained, the later distance function is not directly considered, only how well it reconstructed the samples it is given. What descriptions do not fit a given image is implicitly learned by the combination never appearing in training. This avoids the above problem completely.

4. The output of an autoencoder is in the same form as the input.

When the input to an autoencoder is in a human-interpretable form, or at least can be restored to one, then so is the output. This provides a unique opportunity of explainability, as the reconstruction of an image directly shows what parts the model focused on.

Given the above, this thesis aims to answer the following research questions:

1. What accuracy can the proposed model achieve, and how does it compare to other zero-shot methods on the same and similar data?
2. How much is the classification accuracy influenced by bias to certain classes, and what improvements can be made with the normalisation proposed?

As a secondary aim, the viability of using the output of the autoencoder for explainability is explored.

1.2 Structure of the Thesis

After the introduction, this thesis is structured into four more chapters.

Chapter 2: Essential Background Knowledge outlines the subjects required to understand the rest of the thesis. This includes a brief introduction into Kolmogorov Complexity, the definition of zero-shot learning and autoencoders, and some of the tools used in implementation. Further, four related works are summarised.

Chapter 3: Zero-Shot Classification first describes the datasets used and gives an overview of the method proposed. Then all involved models are explained in greater detail.

Chapter 4: Results presents what experiments were made and their outcomes. The first section is hereby devoted to zero-shot classification, whereas the second section is about explainability.

Chapter 5: Discussion gives an interpretation of the results achieved and compares them to previous work. Afterwards possible improvements and future work are described.

CHAPTER 2

Essential Background Knowledge

2.1 Kolmogorov Complexity

Kolmogorov Complexity is an area of algorithmic information theory named after Andrey Kolmogorov, who described it in a 1965 paper[Kol65]. A similar concept has also been introduced independently by Ray Solomonoff in 1964[Sol64].

Since a practical application of the field served as major inspiration for this project and may further help to explain the improvements in accuracy achieved in section 3.5, this section shall give an introduction to its basics and said application.

Everything in this section not otherwise cited is based on [LV93].

2.1.1 Basics

In simplest terms, the Kolmogorov Complexity of an object is the length of its shortest description, or a bit more formally,

$$C_U(x) := \begin{cases} \min\limits_{p} \{\, l(p) \mid U(p) = x \,\} & \exists p : U(p) = x \\ \infty & \text{otherwise} \end{cases} \tag{2.1}$$

where x is the object in question, p being descriptions, $l(\cdot)$ is a function giving the length of the argument, and $U(\cdot)$ is a method of getting an object from its description.

Of course some more restrictions must be set for this definition to be of any use in a formal context:

- The descriptions are finite, binary strings, i.e., $p \in \{0,1\}^* \wedge l(p) < \infty$
  - $l(p)$ is therefore the number of bits in the description

- The objects are binary strings, i.e., $x \in \{0,1\}^*$
- U is a universal Turing machine

It is trivial to show that the restriction to binary strings does not induce any loss of generality.

An important property of this definition is that it is equal up to an additive constant for any choice of universal Turing machine, i.e., for any object x and any universal Turing machines U_1, U_2:

$$|C_{U_1}(x) - C_{U_2}(x)| \leq c_{U_1,U_2}, \tag{2.2}$$

where c_{U_1,U_2} only depends on the Turing machines[1].

Since it is known that a universal Turing machine exists, one such machine U_0 can be chosen as fixed reference, thus leading to the definition of (unconditional) Kolmogorov complexity as

$$C(x) \coloneqq C_{U_0}(x)\,. \tag{2.3}$$

2.1.1.1 Pairs of strings

Further of interest, especially for section 2.1.2, is the complexity of pairs of strings. Clearly shared patterns can lead to a description shorter than the combined descriptions of both, due to patterns either shared by both strings or even only emerging after the combination.

Less intuitive, however, is that the complexity might even increase by more than a constant term. This stems from the fact that simply generating and combining the descriptions for both requires information about where to split the input. The best upper bound that can be provided is therefore

$$C(x,y) \leq C(x) + C(y) + O(\log(\min(C(x)\,, C(y))))\,. \tag{2.4}$$

2.1.1.2 Incompressibility

A string can be compressed if there is any description of it that is shorter than itself $(C(x) < l\,(x))$. Given that there are only $\sum_{k=0}^{n-1} 2^k = 2^n - 1$ binary strings with less than n characters, but 2^n such strings of length n, there are strings of any length which cannot be compressed, i.e., they are incompressible.

This definition can be extended to c-incompressibility, meaning that there are no descriptions of a string at least c characters shorter than itself $(C(x) > l\,(x) - c)$. By a similar argument as above, there are $2^n - 2^{n-c} + 1$ binary strings of length n.

[1]This assumes using the same enumeration for Turing machines. Comparisons between recursively isomorphic enumerations still only differ by another additive constant, but no such similarities necessarily exist between any other enumerations.

It can therefore be said that most objects are c-incompressible, already for $c = 0$ and even more so for larger c.

2.1.1.3 Notable variations

Not mentioned above is the conditional Kolmogorov complexity $C(x|y)$, which gives the length of the shortest description of x, given the information y. All properties listed in this section so far apply to this definition as well[2].

Also of note is the prefix Kolmogorov complexity $K(x)$ (or $K(x|y)$ in the conditional case). By requiring all descriptions x of computable strings, i.e., $U_0(x) < \infty$, to not be proper prefixes of any other such description, the definition acquires a number of desired properties that allow for the definitions in section 2.1.2, but shall not be further discussed here.

2.1.2 Information & Compression Distance

Vitànyi et al. describe several possible definitions for information distance, i.e., ways to describe how different two objects are, mostly based on conditional Kolmogorov complexity.[LV93][BGL+98] However, the most relevant metric to [CV07], and in further consequence to this thesis, is

$$E(x, y) \coloneqq K(x, y) - \min(K(x), K(y)). \tag{2.5}$$

This is then normalised to avoid the same difference having a larger impact on the distance between short strings compared to larger ones. Therefore, the normalised information distance (NID) is defined as

$$\mathrm{NID}(x, y) \coloneqq \frac{K(x, y) - \min(K(x), K(y))}{\max(K(x), K(y))}. \tag{2.6}$$

Since $K(x)$ like $C(x)$ is not computable, an approximation is required for practical applications. This can be achieved with compression algorithms, resulting in the Normalised Compression Distance

$$\mathrm{NCD}(x, y) \coloneqq \frac{Co(x, y) - \min(Co(x), Co(y))}{\max(Co(x), Co(y))}, \tag{2.7}$$

where Co is the size of the output of a compression algorithm.

[2] In fact, [LV93] introduces conditional Kolmogorov complexity first and defines unconditional Kolmogorov complexity as $C(x) \coloneqq C(x|\epsilon)$.

2.2 Zero-shot Learning

In classification tasks labelled training data is often hard to obtain. This can be the case for many reasons, from classes changing over time, to the amount of classes making collection unfeasible, to labelling simply being too expensive. Zero-shot learning describes the group of methods that work with some classes that have no data available at all. [WZYM19]

For this to be possible information about those classes must be provided in other forms, i.e, some sort of description. Which form this description takes is highly varied, especially if it is domain specific [WZYM19]. For example, a task involving the recognition of handwritten letters with only handwritten digits as training data might use 7×5 pixels representation of the characters in some digital font as description [LEB08].

An especially popular form of description are binary attribute vectors. Each position in such a vector describes a feature a class might have and is set to 1 for all classes it appears in, otherwise it is 0. [XLSA19][WZYM19]
Attribute vectors simplify the task of zero-shot learning greatly. As long as data for all attributes is available, the model can be trained with the vectors as output labels. For classification an input is then assigned to the class with the most similar vector.
In one especially simple approach, the model was trained on the known classes as it would be in a non-zero-shot situation, except for the last weight matrix M being replaced by $M_1 M_2$. M_2 was hereby a constant mapping consisting of all known attribute vectors. To add new classes after training it was only necessary to append the new attribute vectors to M_2. [RPT15]

However, while attribute vectors are often easier to obtain than entire sets of samples for each class, they are still not readily available for every dataset.
Descriptions in natural language are far easier to obtain in comparison. For this reason works in zero-shot learning increasingly focus on their usage. [BSFS15]
The approach introduced in this paper is one of them, two more are described in the following section.

2.3 Related Work

Learning Deep Representations of Fine-Grained Visual Descriptions[RALS16] provides the main dataset used throughout this thesis. To the author's best knowledge, it is still the state-of-the-art for the set of description used, and therefore serves as primary point of comparison. At the time it was released, it even attained accuracies higher than what was achieved with attribute vectors on the same image sets. (56.8% versus 50.4%, more details in section 4.1.4.)
Reed et al. present several models based on different text encoders. Since this paper

was published before the advent of transformers, the text encoders are all convolutional neural networks, recurrent neural network, or long short-term memory based. Images are encoded with a convolutional neural network.
Then a scoring function is trained based on the inner products of the features of the encodings. For classification descriptions are ranked by their score, as they are in this thesis. This suffers the previously described problem of requiring negative samples.

Predicting Deep Zero-Shot Convolutional Neural Networks Using Textual Descriptions[BSFS15] uses the same image dataset as the approach above, but generates class descriptions based on word frequencies in the Wikipedia articles of the classes.
Classification is done in a process involving word embeddings that is fully unrelated to this thesis, but the accuracies achieved give another point of comparison.
More importantly, they show a method to differentiate known and unknown classes by considering the uncertainty of the classifier. This allows to only focus on the classification of unknown classes in zero-shot learning and instead use a conventional classifier when a known class is detected. A similar method has yet to be implemented for the model introduced in this thesis, but an idea how it might be possible is described in section 5.2.

Clustering by Compression [CV05] is entirely unrelated to zero-shot learning, but instead introduces the normalised compression distance that much of this work is based on. The concept is more extensively described in section 2.1.2.
The normalised compression distance is based on the purely theoretical normalised information distance, which has been proven to be an optimal. Cilibrasi et al. show that their distance function is a good approximation in many clustering applications.
Their results are based on the assumption of a lossless compression algorithm and are therefore not directly transferable to this work.

Objects that Sound [AZ18] describes two models to compare video and audio data. Both models use still images and the spectrogram of the audio at the same timestamp as inputs. In the first model those inputs are processed with convolutional neural networks followed by two fully connected layers before they are connected. Then the euclidean distance between the processed inputs is evaluated by a small fully connected network to decide if they are related.
The second model is more important for this thesis. While the audio input is treated similar to the first model, only convolutional layers are used for the image data. The connection is then made with pairwise scalar products, and the relation is computed with max-pooling.
What makes this remarkable is that the last layer before the max-pool can be used as a localisation map showing what part of the image is connected to the sound. This shows how a neural network can learn to associate image regions with descriptions in a human-interpretable way, without the use of labelled image regions.

2.4 Autoencoders

Autoencoders are a type of Artificial Neural Network (ANN) that has been around since at least 1985, when they were proposed as a method of unsupervised backpropagation. In its original and still commonly used form, the network has the same input as output, but a hidden layer of a smaller size, forcing the model to learn a compressed (encoded) representation. [RHW86]

Since then many variations have been developed based on all types of ANN, for example Convolutional or Recurrent Neural Networks, for both encoding and pre-training purposes. Another popular use is the removal of noise by training with distorted input images and using the originals as output. [YZS+19]

2.4.1 Mish Activation

As with all forms of ANN, the choice of activation function plays a heavy role in the performance of autoencoders. It is common to use the same activation function throughout the network except for the output layer, which is often domain dependent.

The autoencoder in this thesis uses an identity function in the output layer as to not restrict what kind of inputs can be restored. All other layers use Mish Activation.

Mish is a recently introduced activation function defined as

$$f(x) \coloneqq x \cdot \tanh(\ln(1+e^x))\,. \tag{2.8}$$

Figure 2.1 shows the graph of the function.

While more expensive to calculate than the similar and often used ReLU and Swish, simply replacing the aforementioned functions with Mish show increased accuracy in several applications.

Misra suggests this might partly be due to small negative inputs being preserved, allowing for higher expressivity. [Mis19]

As of the time of this writing, the paper introducing Mish has not been published, but the advertised effect could be confirmed in tests. Compared to using ReLU in all layers but the last one, zero-shot classification accuracy was improved by 0.64% and the amount of training epochs required for this result was nearly halved. This more than makes up for the more time-consuming calculation.

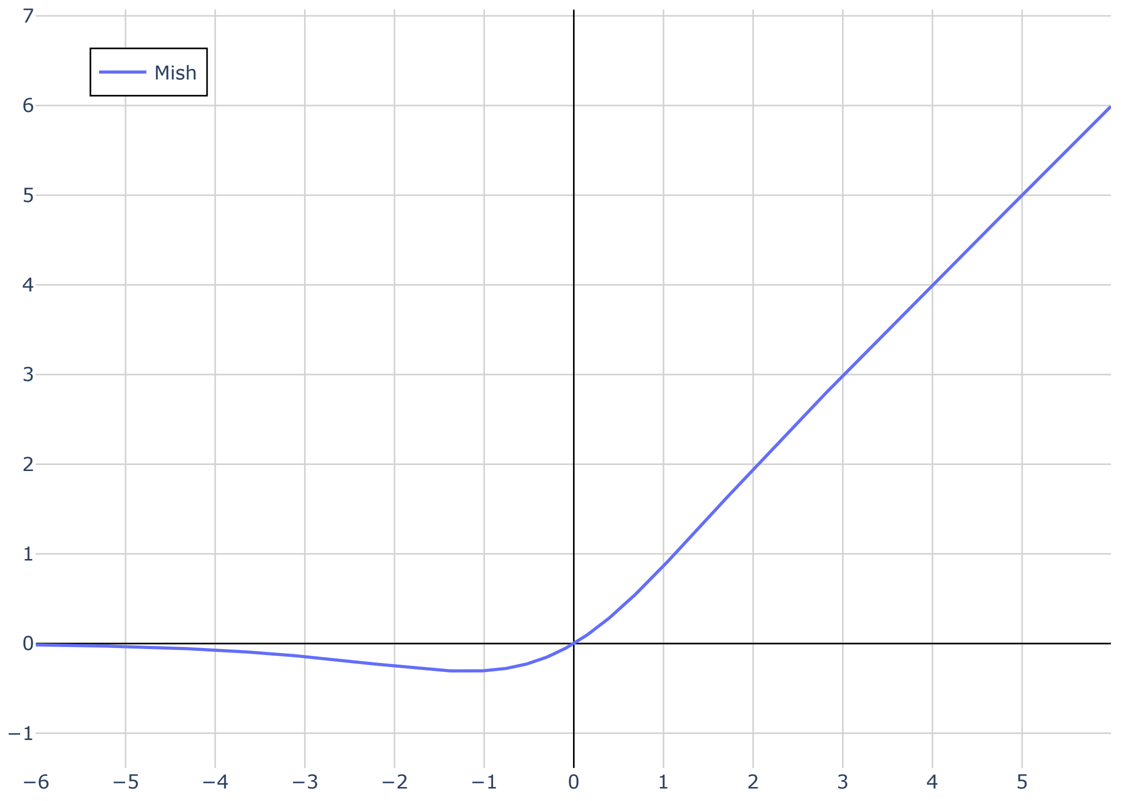


Figure 2.1: The Mish activation function.

2.4.2 Ranger Optimiser

Instead of fixed learning rates, most modern ANN use optimisers that assign individual learning-rates to every weight and adapt them throughout the training.

Ranger is a combination of two newly developed optimisers, Rectified Adam (RAdam) and Lookahead, and was suggested in a blog post by Less Wright[3]. Wright reports that their team achieved several new records on FastAI leaderboards by using Mish and Ranger.

Lookahead is designed to build upon another optimiser to increase long term stability and to reduce sensitivity to suboptimal hyper-parameters like the starting learning-rate. [ZLHB19]
RAdam improves upon the Adam optimiser by adding a rectifying term that removes the need for a warmup period [LJH+20].
Both methods are described in greater detail below.

[3]`https://medium.com/@lessw/new-deep-learning-optimizer-ranger-synergistic-combination-of-radam-lookahead-for-the-best-of-2dc83f79a48d`

Like Mish, Lookahead has not been yet published at the time of this writing. However, replacing the previously used Adam optimiser with Ranger led to faster convergence during the training of the autoencoder and showed no negative influence on classification.

The implementation of Ranger used in this project is provided by Less Wright[4] and is based on the official RAdam repository [LJH+20] and the Lookahead implementation by lonePatient[5].

2.4.2.1 RAdam

The first component of Ranger is RAdam, an improvement to the widely used Adam Optimiser, introduced in [LJH+20].

It has often been observed that Adam requires a warmup heuristic to prevent it from converging to bad local optima. Liu et al. identify the high variance of the adaptive learning rate in the early training stages as a likely culprit and attempt to amend it with a rectifying term

$$r_t = \sqrt{\frac{(\rho_t - 4)(\rho_t - 2)\rho_\infty}{(\rho_\infty - 4)(\rho_\infty - 2)\rho_t}}, \tag{2.9}$$

where

$$\rho_t = \frac{2}{1-\beta_2} - 1 - \frac{2t\beta_2^t}{1-\beta_2^t} \tag{2.10}$$

is the length of the simple moving average (SMA) approximated by the exponential moving average (EMA) used to calculate the second moment in the Adam optimiser at timestamp t. Accordingly, β_2 here is the second parameter of the Adam optimiser.

For $\rho_t > 4$ the update of weights ensues as with Adam, i.e.,

$$\theta_t \leftarrow \theta_{t-1} - \alpha_t r_t \frac{\widehat{m_t}}{\widehat{v_t}} \tag{2.11}$$

with r_t as in equation 2.9, and model weights θ_t, step-size α_t, bias-corrected EMA of the first and second moment $\widehat{m_t}$ and $\widehat{v_t}$ as in the regular Adam optimiser.

For $\rho_t \leq 4$ the step is simplified to

$$\theta_t \leftarrow \theta_{t-1} - \alpha_t \widehat{m_t}. \tag{2.12}$$

Experiments show RAdam to achieve faster convergence and improved accuracy compared to Adam in all attempted tasks[LJH+20].

[4]`https://github.com/lessw2020/Ranger-Deep-Learning-Optimizer`
[5]`https://github.com/lonePatient/lookahead_pytorch`

2.4.2.2 Lookahead

The second part of Ranger is Lookahead, which builds on any given existing optimiser to further speed up convergence and to reduce sensitivity to suboptimal hyperparameters [ZLHB19].

This is achieved by keeping two sets of weights called fast weights $\theta_{t,1\ldots k}$ and slow weights $\theta_{t,0}$. The fast weights are updated by another optimiser A,

$$\theta_{t,k+1} \leftarrow \theta_{t,k} + A\left(\theta_{t,k-1}, \ldots\right), \tag{2.13}$$

whereas the slow weights are only updated every k steps to

$$\theta_{t+1,0} \leftarrow \theta_{t,0} + \alpha\left(\theta_{t,k} - \theta_{t,0}\right) \tag{2.14}$$

with $\alpha \in (0, 1)$.

Informally, this allows for the fast weights to explore the loss surface, with the slow weights pulling them back from bad local minima.

2.5 Summary

This chapter gave a short overview of the theoretical backgrounds this work is based upon, as well as over newly developed methods employed in the code.

Section 2.1, and especially section 2.1.2, explained the distance measure approximated with autoencoders in section 3.4.

Then section 2.4 stated the origins and basic principles of said autoencoders.

In section 2.2 the concept of zero-shot learning is described.

Lastly, sections 2.4.1 and 2.4.2 respectively summarised the principles of the activation function and the optimiser used throughout the implementation of the models in section 3.4.

CHAPTER 3

Zero-Shot Classification

3.1 Choice of Datasets & Preparation

This thesis is mostly focused on image data with natural language class descriptions, for which few datasets exist so far. However, some datasets for image-labelling tasks can be adapted by either selecting representatives, or by combining the texts after vectorisation. More about this in section 3.3.

All images are reshaped to 224×224 pixels, adding black borders if necessary. Other methods of adding borders, for example by repeating the last row of pixels to both sides, showed little difference in performance.

3.1.1 Datasets with Descriptions: Birds

Reed et al. [RALS16] provide two datasets, one of birds and one of flowers. Both contain ten sentences describing each image, which when combined per species serve as the class descriptions. Due to complications with the format the flower data is provided in, only the bird data is used in this project.

The bird dataset contains 6033 images of 200 bird species, with around 30 images per species. Within their paper, Reed et al. further suggest a specific train/validation/test split of 100/50/50, which was also used in this project. A list of all classes sorted by this split can be found in appendix A.

3.1.2 Datasets without Descriptions

As the bias correction introduced in section 3.5 does not require additional labels, any set of images could be used. In order to evaluate the impact of domain overlap and image variance, several datasets are used for this purpose.

3.1.2.1 Animals with Attributes 2

Animals with Attributes 2 (AwA2 [XLSA19]) is a dataset intended for zero-shot learning. It contains 37322 images of 50 different animals, though the numbers per class vary greatly. Further, 85 binary features are provided for all classes, but are not used in this thesis. See section 5.2 for possible future applications.

3.1.2.2 Caltech 101

Caltech 101 (Cal101 [FFFP04]) supplies 9145 images in 102 fairly unconnected categories.

Note: Only 100 of the classes in the provided download have equivalents in the list of 101 categories in the associated paper, and even those often have different names.

3.2 Overview of the Zero-Shot Learning Procedure

The zero-shot classification method introduced in this thesis can be divided into four steps, with an optional fifth step for explainability. Figure 3.1 gives an overview of these steps.

1. **Vectorisation of Inputs**
   To keep the the autoencoder used for complexity estimation simple, feature extraction and vectorisation is done in a pre-processing step. This also allows for a modular setup, i.e, parts of the model can be exchanged quickly without everything needing to be redone.

   Two methods for image vectorisation are used in the following.
   One is based on ResNeXt, a model that achieved high accuracies on ImageNet datasets [XGD$^+$17]. A version trained on ImageNet-1K is first fine-tuned on the previously introduced bird dataset (section 3.1.1, then the last layer is removed.
   The second is a very simple convolutional autoencoder (SCAE) created for this thesis. It is both meant to test the capabilities of the proposed method with only rudimentary feature extraction, as well as to provide an image decoder to examine options for explainability.

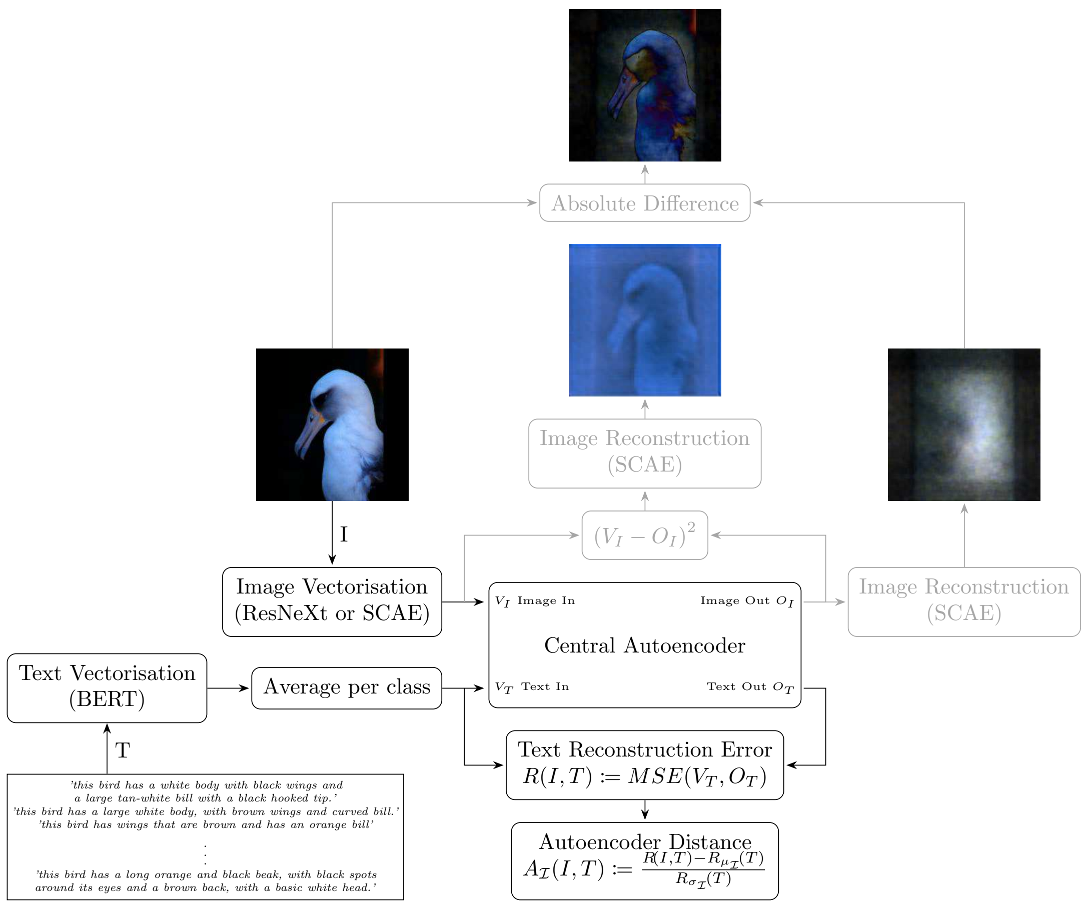

Absolute Difference
Image Reconstruction (SCAE)
$(V_I - O_I)^2$
I
Image Vectorisation (ResNeXt or SCAE)
V_I Image In
Image Out O_I
Central Autoencoder
V_T Text In
Text Out O_T
Image Reconstruction (SCAE)
Text Vectorisation (BERT)
Average per class
T
'this bird has a white body with black wings and a large tan-white bill with a black hooked tip.'
'this bird has a large white body, with brown wings and curved bill.'
'this bird has wings that are brown and has an orange bill'
'this bird has a long orange and black beak, with black spots around its eyes and a brown back, with a basic white head.'
Text Reconstruction Error $R(I,T) := MSE(V_T, O_T)$
Autoencoder Distance $A_{\mathcal{I}}(I,T) := \frac{R(I,T) - R_{\mu_{\mathcal{I}}}(T)}{R_{\sigma_{\mathcal{I}}}(T)}$

Figure 3.1: Diagram of the entire process except for the calculation of normalisation values. Grey parts are only for explainability and are not necessary for zero-shot classification.

For text vectorisation only one method is used in this thesis. BERT is a transformer based model that provides the option to encoder entire sentences as vectors [DCLT19].
Since the dataset used here includes descriptions for each image rather than for each class, all sentences are encoded separately and the average vector per class is used as class description.

Section 3.3 gives more details about input vectorisation.

2. **Compression Distance Approximation**
To approximate the combined complexity of an image and a description, first an autoencoder must learn to associate them. This is done by training it with only positive samples, i.e., vectorised versions of images and their corresponding class descriptions.

The distance between a given image and description is then estimated by calculating the mean squared error between the input description and its reconstruction.

From here on this autoencoder will often be referred to as central autoencoder to differentiate it from autoencoders used for input vectorisation.

More details on the process and a depiction of the central autoencoder can be found in section 3.4.

3. **Bias Reduction through Normalisation**
   The process in the previous step neglects to consider the different inherent complexity descriptions have. Since this would introduce a bias to less complex descriptions, a way to normalise the estimated distance must be found.
   Unlike with the Compression Distance the method is based on, the compression algorithm used here cannot be used to compress the descriptions alone. Instead the distances of all descriptions to all images of a chosen set is calculated. The mean and standard deviation of these distances is then used for normalisation.
   The normalised text reconstruction error is referred to as Autoencoder Distance.

   Section 3.5 describes the motivation behind the Autoencoder Distance and gives the exact definition. Different possible image sets for bias reduction are considered in section 4.1.1.

4. **Classification by Ranking of Descriptions** An image is classified by first choosing a set of possible classes, e.g., all classes of which descriptions are available. The descriptions are then sorted by their Autoencoder Distance to the image and the class associated with the top-ranked description is selected.

   Chapter 4 describes the results achieved this way.

5. **Image Reconstruction** When an autoencoder is used for input vectorisation, e.g., SCAE, the output of the central autoencoder can be reconstructed to human-interpretable form. This may give insights into the inner workings of the model.

   Section 4.2 explores two approaches to this.
   The first is to directly decode the output of the central autoencoder and to compare it with the input image.
   In the second approach the decoder part of SCAE is applied to the quadratic difference between the vectorised image and the output of the central autoencoder.

All models are implemented with PyTorch [PGM+19].

3.3 Input Vectorisation

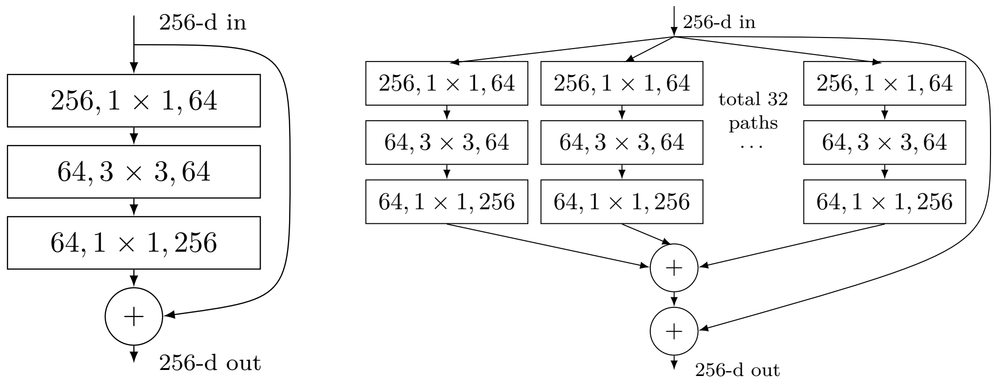
The central autoencoder is meant to take two inputs, encode them, and decode them as closely to the original as possible. For it to take images and texts as inputs directly, it would need to be able to restore these inputs as well. Besides increasing the required computational resources, this could also introduce additional points of failure into a model meant to test the viability of autoencoder distance.

Due to this all inputs, images and descriptions, are first vectorised in a separate pre-processing step. This can be done with several existing pre-trained models, though fine-tuning on the training data improves results.

As an additional benefit, this allows feature extraction methods to be easily interchangeable without having to redesign the entire models. Possible end-to-end train models are still a consideration for the future and are discussed in section 5.2.

3.3.1 Fine-Tuned ResNeXt-101

ResNeXt [XGD+17] is a convolutional network architecture based, both in name and in function, on the popular Residual Network (ResNet) [HZRS16] model.



(a) A block of ResNet (b) A block of ResNeXt with Cardinality 32

Figure 3.2: Blocks of ResNet and ResNeXt with roughly equivalent complexity. Layers are described as (# in channels, filter size, # out channels).
The figure is based on figure 1 in [XGD+17].

Like ResNet, the ResNeXt model is separated into blocks after each of which the input of the block is added to the output (figure 3.2a). This makes the learning of an identity function trivial (the block only needs to output 0) and therefore greatly reduces the vanishing gradient problem.

Unlike ResNet, the blocks in ResNext are not simply a few stacked layers, but instead several such stacks executed in parallel and summed up at the end of a block. The number of stacks used is decided by a new hyper-parameter called Cardinality (figure 3.2b). Each of these branching paths has the same topology internally.

This project uses a pretrained 101-layer variant of ResNeXt (ResNeXt-101) provided by the Torchvision package contained in PyTorch [PGM+19], which was trained on the ImageNet-1K dataset [RDS+15].
It is fine-tuned by replacing the last linear layer (2048 to 1000) by one fitting the number of training classes (i.e. 2048 to 100 for the bird dataset), then training with the default Ranger optimiser (see section 2.4.2), a batch-size of 60, and using 20% of the training images for validation/early stopping.
The classification layer is then removed again for the creation of image vectors.
All inputs, both for fine-tuning and vectorisation, are normalised with a mean of $[0.485, 0.456, 0.406]$ and a standard deviation of $[0.229, 0.224, 0.225]$, as those are the values used when the model was pre-trained.

3.3.2 Simple Convolutional Autoencoder

As a second image vectorisation method a simple convolutional autoencoder (SCAE) was created. The encoder consists of three convolutional layers, each followed by max-pooling and the Mish activation function. It reduces the three-layered 224×224 pixel images to vectors of size 25088, which is about a sixth of their original size of $224 \cdot 224 \cdot 3 = 150528$.

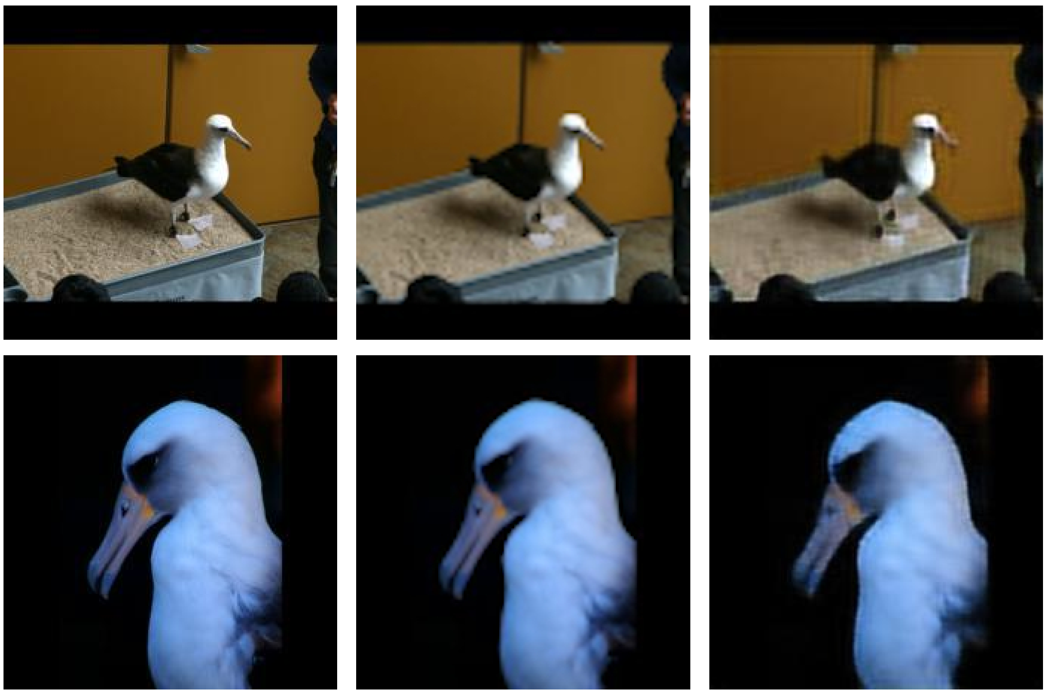
Figure 3.3 shows that SCAE is able to reconstruct images in a recognisable form, but achieves a lower resolution than resizing the image to 92×92 pixels and enlarging it again. The backgrounds being of a similar quality to the birds themselves further suggests that it did not learn to recognise bird-specific features.

SCAE was made to provide an decodable image vectorisation to explore explainability options (section 4.2) and to test the viability of the proposed method with less sophisticated feature extraction (section 4.1.1.2).

3.3.3 BERT

Bidirectional Encoder Representations from Transformers (BERT) [DCLT19] is, as the name implies, a transformer based language representation model. It is, as of the time of this writing, one of the most popular tools in the NLP community.

As a language representation model, BERT learns to encode natural language tokens as fixed length vectors. A token is hereby usually a number representing a word, but words may also be split up into multiple tokens in the case of common suffixes or unknown words. Additionally a [CLS] token is prepended to all inputs, and if a task requires two input sentences, they are separated with a [SEP] token.



(a) Original 224 × 224 pixels (b) Resized to 92 × 92 pixels (c) Reconstructed with SCAE

Figure 3.3: Images of Laysan Albatrosses with resolution reduced to approximately a sixth, compared to reconstruction with SCAE. Differences are especially apparent around the beaks.

Unlike with language embeddings, these vectors are not always the same for a given word, but depend on context. I.e., giving BERT the sentences *'Lead is a metal.'* and *'Lead the way.'* result in entirely different vectors for the word *'Lead'*.

Most models before BERT were trained in a left-to-right, or right-to-left fashion, always predicting the next word in the sentence. This leads to only the context on one side of a word being considered.
BERT instead uses bidirectional training. To do so, 15% of the input words are masked, i.e., replaced with as [MASK] token. The model is then tasked to predict the masked words.
In most applications BERT would afterwards be fine-tuned for the specific task at hand, e.g., sentence prediction.

When BERT is used on a sentence, all layers of the model keep internal representations for each input token known as hidden states. The hidden states of the last layer can then be used for downstream tasks. In this thesis the hidden state corresponding to the [CLS] token is used, which represents the entire sentence.

This project uses $\text{BERT}_{\text{BASE}}$ as implemented in the library PyTorch-Transformers [WDS+19].

Due to hardware limitations in the earlier stages of this thesis, the model is employed without fine-tuning. Changing this and reasons why that might be difficult are discussed in section 5.2.

3.3.4 Combination of Per-Image Descriptions

Most tests for this project were performed on the bird dataset introduced in section 3.1.1, which did not include descriptions of the bird species involved, but instead contained ten sentences for every image, each describing the bird pictured.

All sentences were individually transformed to vectors of size 768 with $\text{BERT}_{\text{BASE}}$, then several methods of aggregating all vectors of a class were attempted, including:

1. Mean
2. Selecting the sentence with the lowest distance to all others sentences (Medoid)
3. Selecting the sentence with the lowest distance to the mean

The distance measures used herein were euclidean, absolute, and cosine.

1. outperformed the other methods by a large margin in early tests, with the best of the others (3. with euclidean distance) reaching at best a third of the Top-1 (T1) and Top-5 (T5) accuracy on part of the validation data (T1 6% and T5 14% versus T1 18% and T5 46%. The results for method 1 have since been improved, as shown in chapter 4).

Together with methods 2 and 3 having a higher T1 accuracy than method 1 on training data, this suggests that method 1 leads to less overfitting. For this reason method 1 is used for the rest of this project.

3.4 Compression Distance Approximation with Autoencoders

The central part of the proposed method is the autoencoder depicted in figure 3.4. The encoder part takes the image and text vectors V_I and V_T separately, followed by still separate hidden layers E_I and E_T, then a combined hidden layer C. The decoder has the same topology in reverse, taking the input from C, then having two split hidden layers D_I and D_T and two output layers O_I and O_T. Of course the encoder and decoder are symmetrical, i.e., the size of V_I is the same as of O_I, the same is true for E_I and D_I, et cetera.

All layers use Mish (see section 2.4.1) as activation function, and the loss function L is

$$L(V_I, V_T) \coloneqq MSE(V_I, O_I) + MSE(V_T, O_T)\,, \tag{3.1}$$

where MSE is the mean square error, V_I and V_T are the vectorised image and text inputs, and O_I and O_T are the respective outputs.

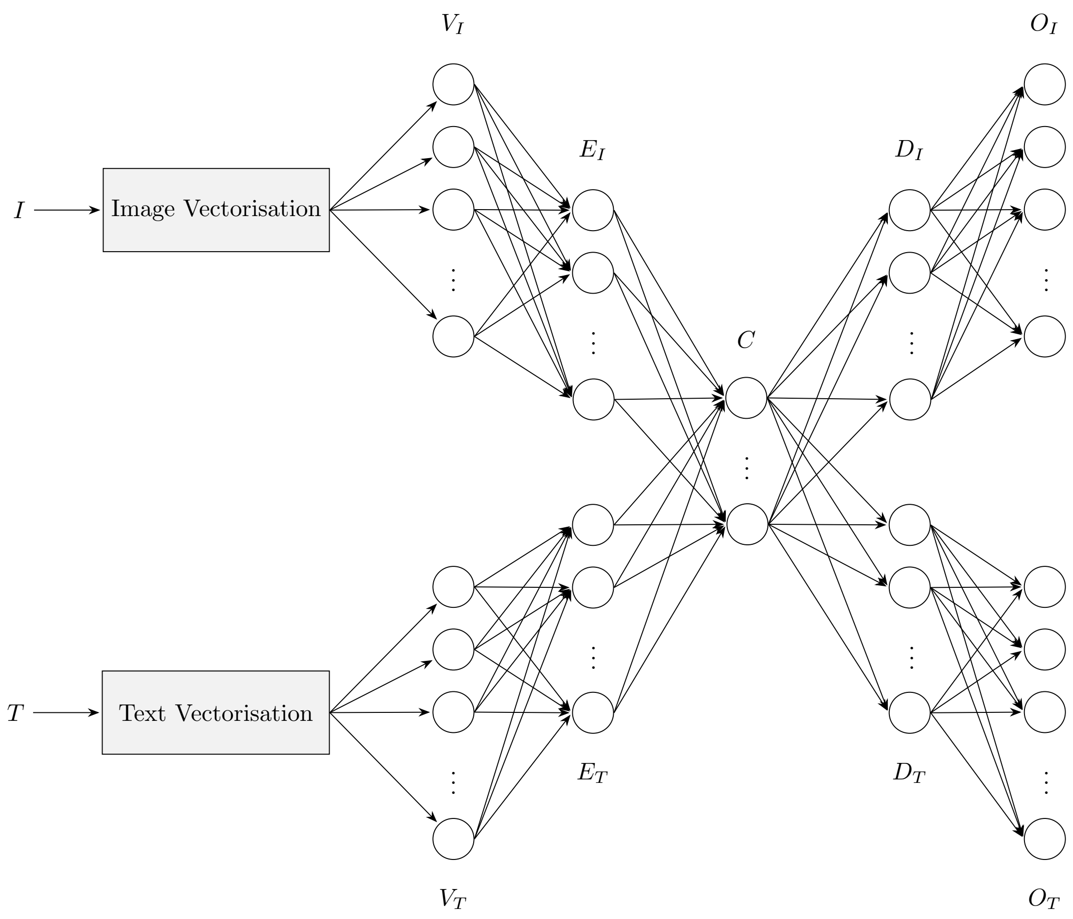


Figure 3.4: Topology of the central autoencoder.

In training, the Ranger optimiser (see section 2.4.2) was used with all values set to their defaults, i.e., $\beta_1 = 0.95, \beta_2 = 0.999$ and a threshold of 5 for RAdam[1], $k = 6$ for LookAhead[2], and a learning rate of 0.001[3].
A batch-size of 60 was chosen as a compromise between available time and computational resources.

[1] $\beta_1 = 0.9$ and $\beta_2 = 0.999$ are used in the papers Adam and RAdam were introduced in and have been shown to generally be a good choice[KB14][LJH$^+$20]. The author of Ranger suggests using $\beta_1 = 0.95$ instead, and further increases the threshold for regulation to 5.

[2] Also a value used in the respective paper, though some more optimisation could be advisable here.[ZLHB19]

[3] The choice of learning rate is luckily largely irrelevant with the use of Ranger.

Using BERT and ResNeXt results in V_I having size 2048 and V_T size 768. In early tests E_I, E_T and C were all set to size 1400, close to the average of the input sizes. To actually compress the text input as well, E_T was later set to size 700 and C to the average of E_I and E_T, i.e., 1050.

These values were kept for training with SCAE instead of ResNeXt. Due to the extreme input size differences, more tests on this may be done in the future.

To calculate the Compression Distance shown in equation 2.7, first an approximation Co of K is needed. Since the autoencoder used here only compresses data to a fixed size, this can only be done by further approximation. One readily available value for this is the reconstruction error of the restored data.
Due to the image consistently having a reconstruction error at least a magnitude higher than the description, likely because of the larger size, it dominated the overall error. Using this for classification on the validation set only gave results barely better than random, i.e., around 2%.
Considering only the text reconstruction error, significantly better results of around 17% could be achieved, which later were improved to the results presented in chapter 4. More formally, the reconstruction error R is defined as

$$Co(I, T) \approx R(I, T) \coloneqq MSE(V_T, O_T)\,, \tag{3.2}$$

where V_T and O_T are the vectorised text input and its reconstruction (see also figure 3.4) when the model is used with input image I and input text T. This error is still influenced by the input image, due to O_T depending on I and T.
Different vectorisation methods, especially ones where the text vector is larger than the image vector, might warrant a different definition including the image reconstruction error.

Using this as an approximation for the entire Compression Distance, images can be classified by choosing the class associated with the description with the lowest distance to a given image.
However, some texts are less complex than others, which leads to them always producing lower errors that introduce a bias for their associated classes. This effect is even more pronounced when those texts have been seen in training.
This is in line with what is known about Kolmogorov Complexity. Some objects are inherently less complex and therefore easier to compress. A given compressor may also contain information about some objects and can consequently create smaller encodings for them. [LV93]

Reducing this bias, and in the process producing an approximation more closely related to the Normalised Compression Distance, is discussed in the following section.

3.5 Class Bias Correction through Normalisation

As stated above, R only gives a very crude approximation of the Normalised Compression Distance (equation 2.7), fully ignoring the terms $\min(Co(I), Co(T))$ and $\max(Co(I), Co(T))$, which are specifically meant to account for size differences. It stands therefore to reason that bias may be reduced by finding appropriate approximations.

Given the larger size of the image vectors used, it is presumed that typically $Co(I) > Co(T)$, an assumption that is further made likely due to $MSE(I_I, O_I)$ consistently being at least one order of magnitude larger than $MSE(I_T, O_T)$. Thus we reach

$$\min(Co(I), Co(T)) = Co(T) \tag{3.3}$$

and

$$\max(Co(I), Co(T)) = Co(I)\,. \tag{3.4}$$

Substituting these equations into equation 2.7 results in

$$\frac{Co(I,T) - Co(T)}{Co(I)} \tag{3.5}$$

Since classification only requires the comparison of different descriptions to a constant image I, the term $Co(I)$ can be ignored, leaving only a need for an approximation of $Co(T)$.

This can, again, be done by calculating the error of the reconstruction. Though since the autoencoder is only able to compress image and text inputs together, this cannot be calculated as directly as the combined value. To circumvent this, a set of normalisation images $\mathcal{I}$ is chosen and the mean of the reconstruction error $R(I,T)$ is calculated for each description T over all images $I \in \mathcal{I}$.

More formal, the mean reconstruction error $R_{\mu_\mathcal{I}}$ for description T over a set of images $\mathcal{I}$ is defined as

$$R_{\mu_\mathcal{I}}(T) := \frac{\sum_{i \in \mathcal{I}} R(i,T)}{|\mathcal{I}|}. \tag{3.6}$$

Ignoring $Co(I)$ and approximating $Co(I,T)$ with $R(I,T)$ (see equation 3.2) and $Co(T)$ with $R_{\mu_\mathcal{I}}(T)$ leads to

$$R(I,T) - R_{\mu_\mathcal{I}}(T)\,. \tag{3.7}$$

This does indeed improve validation accuracy slightly for carefully chosen $\mathcal{I}$, but never by more than 2%.

However, when straying a bit further from the Compression Distance formula and accounting for variance within $\mathcal{I}$ this can be improved further. To do so, the above is divided by the standard deviation of the text reconstruction error over $\mathcal{I}$, i.e., $R(I,T)$ is normalised. Applying the formula of the standard deviation to $R(I,T)$ and using $R_{\mu_\mathcal{I}}$ as mean results in

$$R_{\sigma_\mathcal{I}}(T) := \sqrt{\frac{\sum\limits_{i\in\mathcal{I}} \left(R(i,T) - R_{\mu_\mathcal{I}}(T)\right)^2}{|\mathcal{I}|}}, \tag{3.8}$$

the standard deviation of the text reconstruction error over $\mathcal{I}$.

With this the Autoencoder Distance $A_\mathcal{I}$ can be defined as the normalisation described above, i.e.,

$$A_\mathcal{I}(I,T) := \frac{R(I,T) - R_{\mu_\mathcal{I}}(T)}{R_{\sigma_\mathcal{I}}(T)}. \tag{3.9}$$

Section 4.1 gives more details about the choice of $\mathcal{I}$ and the classification accuracies achieved by using the Autoencoder Distance.

3.6 Summary

This chapter introduced all the models required for the main part of the thesis project, and described how they fit into the concepts explained in chapter 2.

First, section 3.3 listed the methods used to vectorise the input images and texts.

For images two such models were shown, a fine-tuned version of ResNeXt used in most experiments, and a deliberately barely functional convolutional autoencoder herein called SCAE that demonstrates the concept to still work in less than ideal conditions.

Google's BERT$_{\text{BASE}}$ was the only model introduced for text vectorisation, other possibilities in that regard will be discussed in section 5.2. Further addressed was the problem of missing class descriptions, and how they were generated from image descriptions in the bird dataset.

Section 3.4 presented the central autoencoder of the project (see figure 3.4) and how it was trained. Additionally, it was described how compression size is estimated with the reconstruction error of said autoencoder to in turn crudely approximate the Compression Distance from section 2.1.2.

In section 3.5 the concept of Autoencoder Distance was introduced as a more accurate approximation of Compression Distance. This was done by taking the reconstruction error of the evaluated text inputs with all images of a given set and then normalising the reconstruction error with the resulting mean and standard deviation. How this reduces bias and therefore improves classification accuracy will be discussed in section 4.1 of the following chapter.

CHAPTER 4

Results

4.1 Zero-Shot Classification

The main goal of this project is to provide a new approach to zero-shot learning, so most of the focus is put on the aspect of classification. One of the method's biggest flaws, however, is the bias towards classes encountered in training, so most tests were made under the assumption of the training data being completely separate from the application. Further, the data used only provided very few images per class, removing some images from the training classes for test purposes was therefore considered as less important than the zero-shot aspects.
Data with different descriptions for training and test of the same classes might alleviate the first problem, but no such data was available.

All accuracies presented in this section were achieved on the bird dataset introduced in section 3.1.1 with their descriptions prepared as described in section 3.3.4.
Training, validation, and test data are in the same 100/50/50 split suggested by [RALS16]. Appendix A shows all classes and what set they are in.

As described before, classification is done by ranking. This means for each image, all class descriptions in the set are ordered by their Autoencoder Distance to that image, and the class associated with the description that has the lowest distance is the classification.
Top-1 (T1) accuracy refers to the percentage of images classified correctly this way. Top-5 (T5) accuracy means the percentage of images for which the correct class was among the descriptions with the five lowest Autoencoder Distances to the image.

4.1.1 Accuracies and Effect of Bias Reduction Measures

To allow for better evaluation on classes not seen in training, training classes are not considered in this section. This means all tests on validation and test data only rank the 50 class descriptions of their respective dataset.
In section 5.2 an idea is discussed how images belonging to classes already seen during training may be detected, which would allow this approach even in application. Similar methods have been suggested previously (e.g. [BSFS15]).
Section 4.1.3 describes the results when descriptions of training classes are included.

Table 4.1: T1 Classification Accuracy on Validation and Test data.

Normalisation Set $\mathcal{I}$	Validation Accuracy	Test Accuracy	Number of Images
(None)	17.33%	16.30%	
Validation	29.28%	22.60%	2961
Seahorse	28.03%	23.25%	57
Test	28.03%	23.87%	2933
Training & Validation	27.63%	23.08%	8855
Octopus	27.52%	23.59%	35
Buddha	27.02%	21.17%	85
Garfield	26.88%	21.58%	34
⋮	⋮	⋮	⋮
Random	22.97%	17.97%	500
⋮	⋮	⋮	⋮
Cannon	22.26%	19.98%	43
⋮	⋮	⋮	⋮
Dalmatian	19.42%	19.09%	67
Soccer Ball	19.12%	19.09%	64
Random (Not Normalised)	2.87%	7.02%	500

Table 4.1 shows the Top 1 (T1) classification accuracy of the model on validation and test data. The first row shows the values without de-bias efforts, the following rows use the process described in section 3.5 with a selection of image sets $\mathcal{I}$, and are sorted by the results on the validation data. The full table, including T5 accuracies, can be found in appendix B labelled with the image set names as they appear in the datasets.
'Random (Not Normalised)' refers to randomly generated image that were not normalised the way the ResNeXt model requires before vectorisation (see section 3.3.1). While it is not a situation that would realistically be encountered when choosing $\mathcal{I}$, the set was included to show that it is possible to choose images that reduce the classification accuracy when used for bias reduction.

Figures 4.1 and 4.2 show samples of the image sets $\mathcal{I}$ used.

It comes with no surprise that choosing $\mathcal{I}$ as exactly those images that are to be classified yields the best results. Since no labels are required for the normalisation process, this can be done when a large list of such images is available. As this will often not be the case, two main approaches, or combinations thereof, offer themselves:

1. Accumulating mean and standard deviation values each time a new image is classified.
2. Using a fixed image set $\mathcal{I}$ (unrelated or training data)

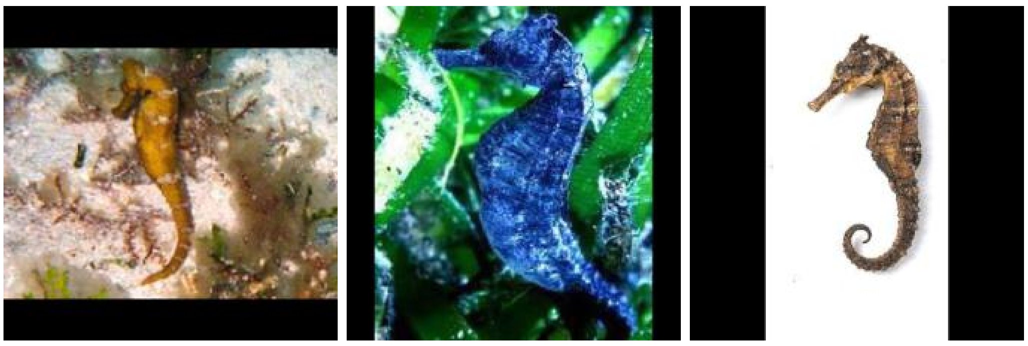
The first can fairly easily be calculated with equations 4.1 and 4.2, where i is the new image and $\mathcal{I}$ is the set of images accumulated so far. These equations are the result by substituting $\mathcal{I}$ with $\mathcal{I} \cup \{i\}$ in equations 3.6 and 3.8 respectively, and the realisation that $\sum_{i \in \mathcal{I}} (R(i,T) - R_{\mu_\mathcal{I}}(T))^2 = \sum_{i \in \mathcal{I}} \left(R(i,T)^2 - R_{\mu_\mathcal{I}}(T)^2 \right)$.

$$R_{\mu_{\mathcal{I}\cup\{i\}}}(T) = \frac{|\mathcal{I}| \cdot R_{\mu_\mathcal{I}}(T) + R(i,T)}{|\mathcal{I}| + 1} \tag{4.1}$$

$$R_{\sigma_{\mathcal{I}\cup\{i\}}}(T) = \sqrt{\frac{|\mathcal{I}| \cdot R_{\sigma_\mathcal{I}}(T)^2 + |\mathcal{I}| \cdot R_{\mu_\mathcal{I}}(T)^2 + R(i,T)^2 - (|\mathcal{I}| + 1) \cdot R_{\mu_{\mathcal{I}\cup\{i\}}}(T)^2}{|\mathcal{I}| + 1}} \tag{4.2}$$

It is therefore not required to save all images, unless new labels are added. However, this method still benefits from having base values for normalisations, at the very least for improved results on the first dozens of images classified.

For the other approach the image set has to be carefully chosen.
As table 4.1 shows, using images of sea horses or octopuses achieves an accuracy on validation and test data nearly as good as when the data itself is used, despite the small set sizes.

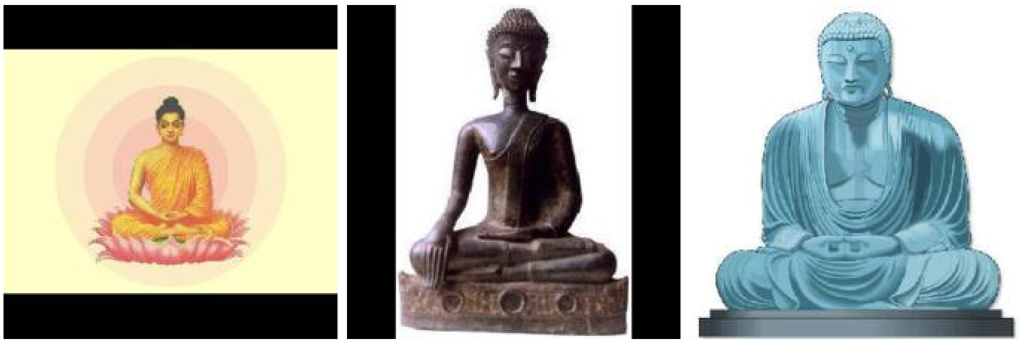


(a) Samples of 'sea_horse' from the caltech101 dataset.

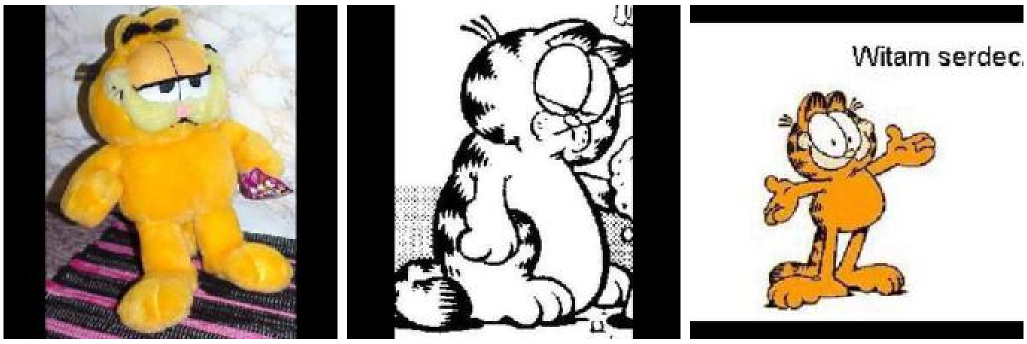
Figure 4.1: Samples of image sets that achieve good results at bias correction.

(b) A 'California Gull', a 'Rusty Blackbird' and a 'Sooty Albatross' from the training set.

(c) Samples of 'octopus' from the caltech101 dataset.



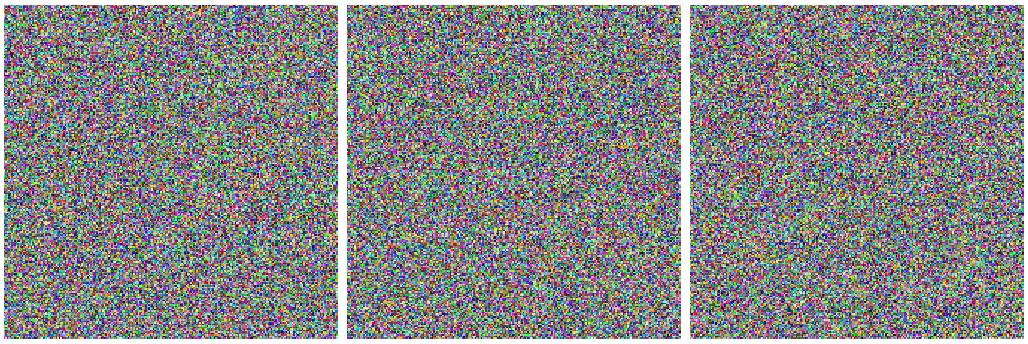
(d) Samples of 'buddha' from the caltech101 dataset.



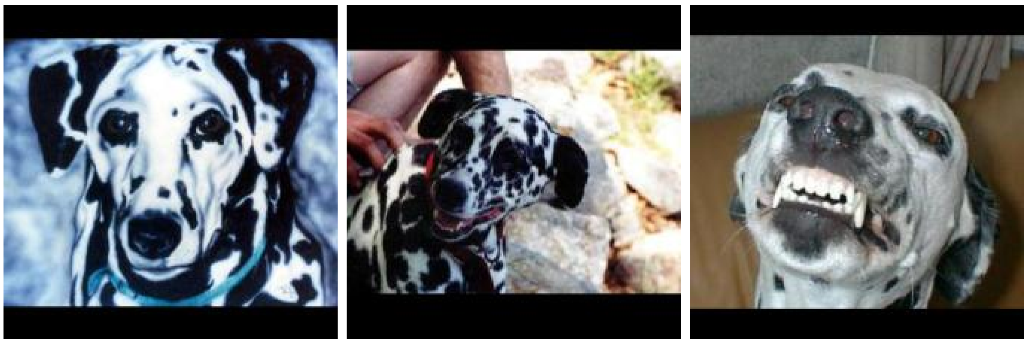
(e) Samples of 'garfield' from the caltech101 dataset.

Figure 4.1: Samples of image sets that achieve good results at bias correction.

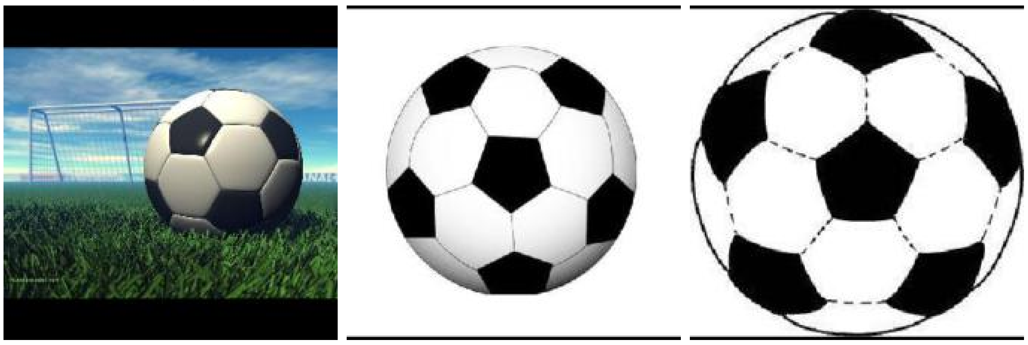
(a) Samples of 'hippopotamus' from the Animals with Attributes 2 dataset.



(b) Samples of random images.



(c) Samples of 'dalmatian' from the caltech101 dataset.



(d) Samples of 'soccer_ball' from the caltech101 dataset.

Figure 4.2: Samples of image sets that result in low to no benefit at bias correction.

While one could assume that images more similar to the evaluated data provide better results, this does not appear to be the case, as even the top values in the table include images of Buddha statues and the cartoon cat Garfield. A trend that can be observed, however, is that images from the AwA2 dataset more commonly appear in with the middling to worse results (see appendix B).
One observable difference between AwA2 and Cal101 is that the former tends to have more regulated and similar images, while the latter often includes decorations or even drawings. Especially the best images (figure 4.1) include a variety of shapes and especially colours, whereas the worst images (figure 4.2) are very uniform, with the worst two mostly being black and white.

There is no clear criterion for what image sets reduce bias best. Examination of the best sets show that they typically contain a wide variety of shapes in different colours, whereas lower ranked ones tend to have few colours and very similar shapes. Random noise also ranks rather low, likely due to no larger shapes being present.
Having said that, tests with combinations of image sets from AwA2 show that while slight improvements can be achieved by combining the best three image sets (hippopotamus, horse, walrus), the result of combining all AwA2 classes is generally worse than just using the best one.
It appears therefore to be best to start with an image set working well on the validation data (here for example sea horses) and to possibly add further images over time as by approach 1. Even then caution is recommended, as using too many similar images of the same class may over-correct the bias towards this class.

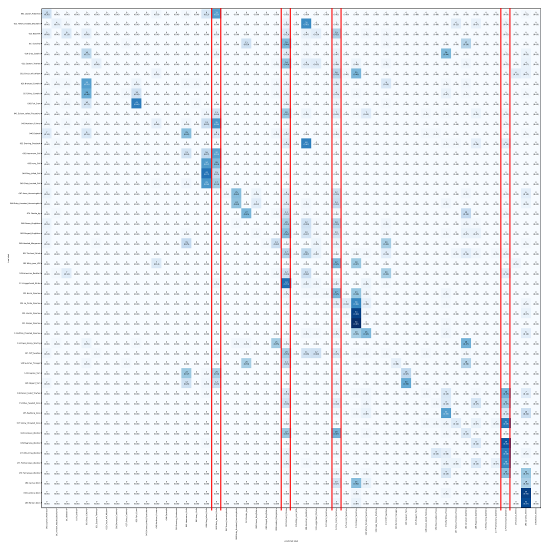
4.1.1.1 Where the Bias Lies

This section presents several confusion matrices with labels and axes that might be hard to read without zooming in. In all cases the x-axis shows the predicted label and the y-axis the true label. The labels follow the same order as the lists in appendix A.

Figure 4.3 shows the confusion matrices of T1 classification on test and validation data. As can be seen, both matrices have a clearly visible diagonal line indicating correctly classified bird species, and several vertical lines showing bias for certain classes. Red lines show examples of such bias indicators in figure 4.3a.

Additionally, some classes are especially often confused for a specific other. The most extreme example of the latter situation is 57 of 60 (95%) Indigo Buntings in the test data being classified as White-breasted Kingfishers (**red** box in figure 4.3b). Figure 4.4 shows two of them.

Normalising the errors with the test images themselves (figure 4.5) clearly reduces both kinds of errors strongly. Especially the misclassified Indigo Buntings are classified nearly 100% correctly, and the vertical lines vanished almost completely. (A red box indicates the same position as before, this time only a single Indigo Bunting is misclassified as White-breasted Kingfisher.)


(a) On validation data.

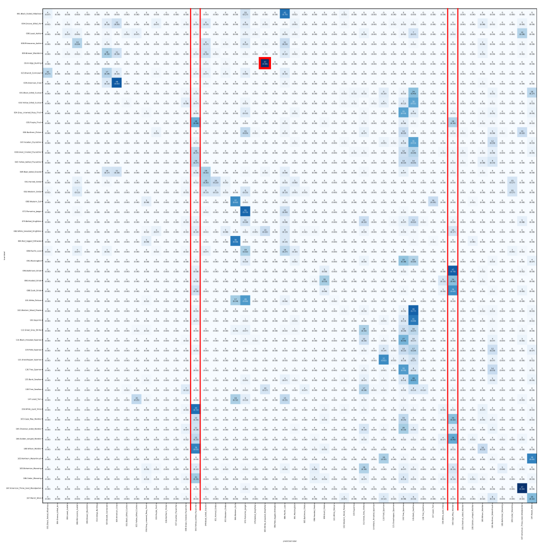

(b) On test data.

Figure 4.3: Confusion matrices of the T1 classification without any bias reduction measures.

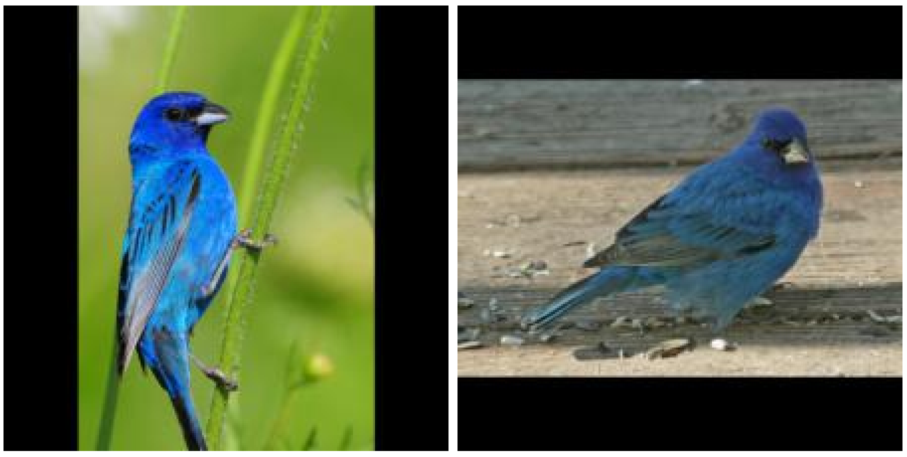
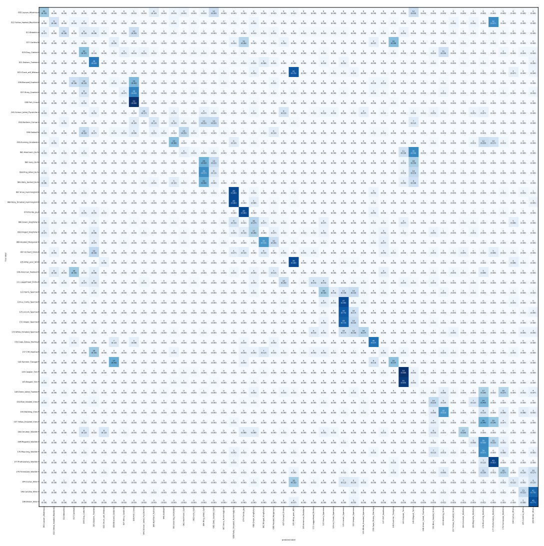


Figure 4.4: Samples of misclassified Indigo Buntings.


(a) On validation data.

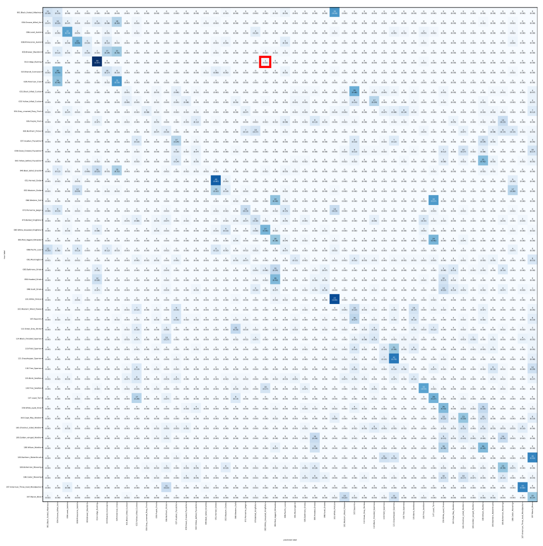
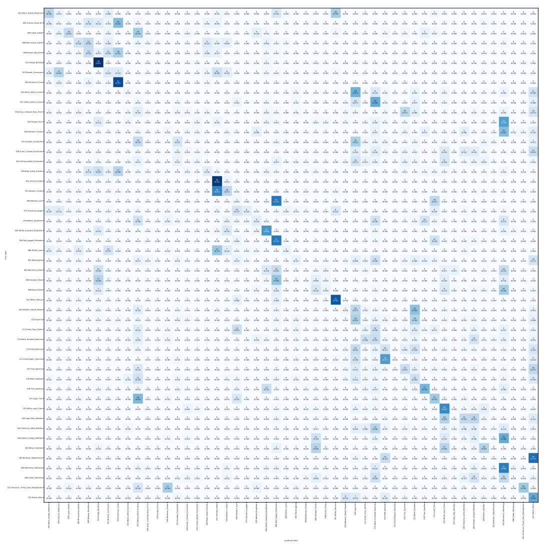

(b) On test data.

Figure 4.5: Confusion matrices of the T1 classification using the evaluated data for bias reduction.


(a) With reduced bias.

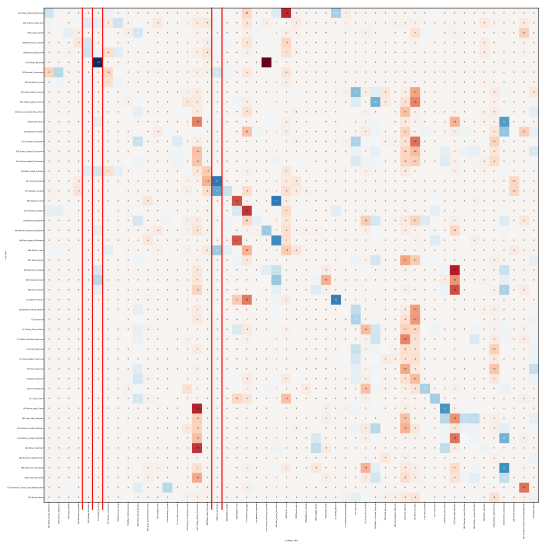

(b) Difference between figures 4.6a and 4.3b. Positive (blue) values are higher in the former.

Figure 4.6: Confusion matrix of the T1 classification on test data using images of sea horses for bias reduction.

As discussed in the previous section, classification often occurs online and a different set of images must be selected. Based on the results on validation data, sea horses are a good choice, resulting in figure 4.6a. A comparison with the previous results (figure 4.7) shows that while some biases are not removed quite as well and a notable error in the classification of Cape May Warblers is introduced (the only dark red entry on the diagonal), the overall difference is not overly large.

Figure 4.6b shows the difference between using sea horses for normalisation compared to the results without bias reduction. The red vertical lines clearly show where previous bias was strongly reduced.
Though some individual errors were introduced as well (red on the diagonal and blue anywhere else), no strong new bias was created this way, i.e., highly visible vertical blue lines on figure 4.6b.

This is, however, not always the case. Choosing a less ideal set of images, for example ibises (see figure 4.8), most of the same biases are removed, but a new one for Sayornis' is introduced (highly visible vertical blue line in both parts of the figure).

Comparing the variance of the reconstruction error $R_{\sigma_{ibis}}(T)$ of all test classes T, shows similar low values for the Sayornis class and the classes of which the bias was reduced, with a gap to the variance other classes produced.
Since standard deviation, and therefore variance too, has a large impact on the Autoencoder Distance, this may allow the detection of classes especially favoured by a given $\mathcal{I}$. Comparing the number of such classes may allow to filter out $\mathcal{I}$ that are prone to over-correction without any labelled data.
No attempts at this were made so far, the possibility is further discussed in section 5.2.

4.1.1.2 Using SCAE for Image Vectorisation

As could be expected, the quality of the image vectorisation plays a large role in the accuracy achieved.

Section 3.3.2 introduced SCAE, an autoencoder that only learned very rudimentary image features. Using this for image vectorisation results in a T1 accuracy equivalent to random if no bias reduction is used, du to all images being assigned to the same class. With bias reduction this can be improved to around 5% for test, and 7% for validation classes.
This shows that the proposed method in principle still works with very little information, though of course the performance suffers.

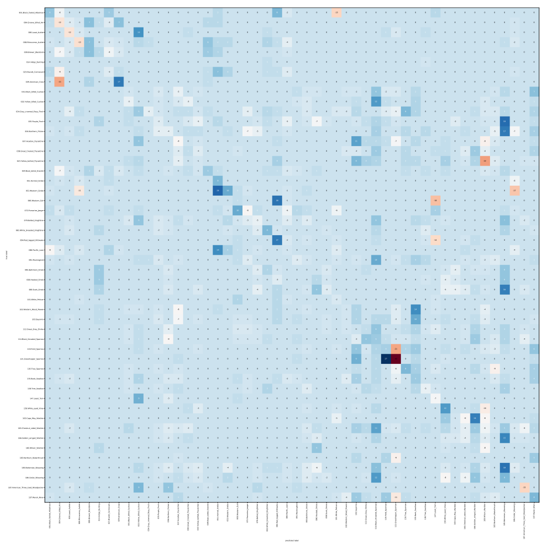
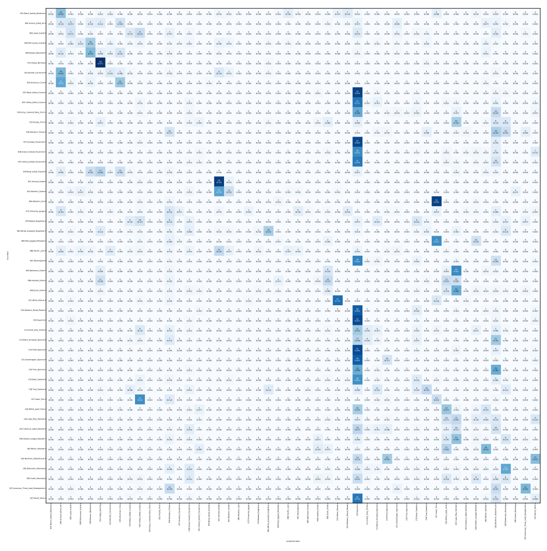
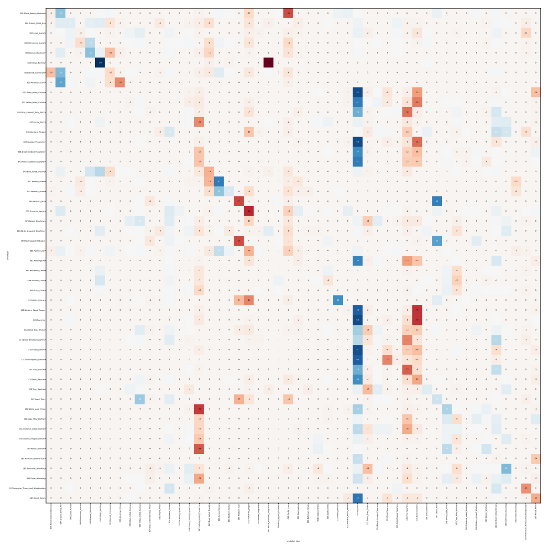


Figure 4.7: Difference between figures 4.6a and 4.5b. Negative (red) values are higher in the latter.



(a) With reduced bias.



(b) Difference between figures 4.8a and 4.3b. Positive (blue) values are higher in the former.

Figure 4.8: Confusion matrix of the T1 classification on test data using images of ibises for bias reduction.

4.1.2 Samples of Individual Results

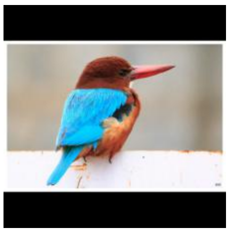
(a) Indigo Bunting 22

(b) Without bias reduction	(c) Sea Horses for bias reduction
083.White_breasted_Kingfisher	014.Indigo_Bunting
138.Tree_Swallow	083.White_breasted_Kingfisher
095.Baltimore_Oriole	138.Tree_Swallow
049.Boat_tailed_Grackle	009.Brewer_Blackbird
014.Indigo_Bunting	051.Horned_Grebe
156.White_eyed_Vireo	049.Boat_tailed_Grackle

'this small bird is mostly light turquoise blue, with a darker blue head and a yellow bill.'
'this small colorful bird has a blue belly and a short black bill.'
'a small bird with a blue breast and a grey belly with a sharp small beak.'

(d) Some of the texts describing images of Indigo Buntings

Figure 4.9: Classification results for a specific image of an Indigo Bunting



(a) White-breasted Kingfisher 22

(b) Without bias reduction	(c) Sea Horses for bias reduction
083.White_breasted_Kingfisher	014.Indigo_Bunting
098.Scott_Oriole	083.White_breasted_Kingfisher
163.Cape_May_Warbler	051.Horned_Grebe
086.Pacific_Loon	053.Western_Grebe
043.Yellow_bellied_Flycatcher	185.Bohemian_Waxwing
156.White_eyed_Vireo	084.Red_legged_Kittiwake

'this bird is bright blue and black with a large bill.'
'this bird has bright blue back and primary feathers, with a red crown and large bill.'
'this bird has wings that are blue and has a bornw [sic] *body and a large red bill'*

(d) Some of the texts describing images of White-breasted Kingfishers

Figure 4.10: Classification results for a specific image of a White-breasted Kingfisher

Figures 4.9, 4.10, and 4.11 show the top results of three individual images of an Indigo Bunting, a White-breasted Kingfisher, and a Tree Swallow respectively. All three figures also show the image classified, and some of the texts used to generate the combined vector (see section 3.3.4).
Sample texts were chosen by being the closest to the combined vector or the medoid of the class, using euclidean, absolute, and cosine distance. In all cases two or three of those sentences were duplicates of others and therefore omitted.

The full ranking of classes for each of those images can be found in appendix C, including the results for using test data as $\mathcal{I}$, and the values of $R(I, T)$ and $A_{\mathcal{I}}(I, T)$.

(a) Tree Swallow 41

038.Great_Crested_Flycatcher
095.Baltimore_Oriole
043.Yellow_bellied_Flycatcher
156.White_eyed_Vireo
112.Great_Grey_Shrike
031.Black_billed_Cuckoo

(b) Without bias reduction

138.Tree_Swallow
053.Western_Grebe
083.White_breasted_Kingfisher
051.Horned_Grebe
095.Baltimore_Oriole
185.Bohemian_Waxwing

(c) Sea Horses for bias reduction

'this bird has a white breast and belly, with a dark gray crown and back.'
'this bird has a shiny blue back and coverts with a blue and black crown and a white throat and brea
'this bird has a flat-shaped head, bright blue eyes, short bill and is white with bright blue wings.'
'the bird has a small bill that is black and white belly.'

(d) Some of the texts describing images of Tree Swallows

Figure 4.11: Classification results for a specific image of a Tree Swallow

As discussed in the previous subsection, most Indigo Buntings are misclassified as White-breasted Kingfishers before bias reduction. Figure 4.9 gives one such example, with the correct classification previously being on the fifth place.
Comparing the descriptions of the top ranked classes without bias reduction (see figures 4.10d and 4.11d for samples of two of them), the biggest similarity besides terms shared by all classes are mentions of blue and black, indicating that the appearance of a colour might be the primary feature considered, not so much the location and shape. However, those are not fully ignored, as, besides the White-breasted Kingfisher, most descriptions also mention short or sharp beaks.

This pattern also holds up after bias reduction (figure 4.9c), though it is noteworthy that the three top ranked classes are those most commonly described as blue, whereas the descriptions of those directly below rarely mention blue, but often black.
Among the top ranks shown in the figure it can also be observed that the classes ranked higher than before bias reduction were all slightly over-corrected, as can be seen in figure 4.6 (surrounded by red lines, in order: Brewer Blackbird, Indigo Bunting, Horned Grebe).

The class with the visibly biggest over-correction of those three is the Indigo Bunting itself. An example of this can be seen in figure 4.10, a White-breasted Kingfisher originally classified correctly, but considered an Indigo Bunting after bias reduction.
However, the correct class remains on place two, even though the rest of the top ranking is changed completely.

Appendix C shows the full ranking for figures 4.9, 4.10 and 4.11. This also includes the values of the reconstruction error R and the Autoencoder Distance $A_{\mathcal{I}}$ for $\mathcal{I} \in \{sea\_horses, test\}$, i.e., the reconstruction error normalised with images from those two sets.
Comparing these values suggests that, before bias reduction, the correct class was chosen due to actual similarities, while the other top ranking choices only had low distance

errors due to inherently being easier to reconstruct.
As an example using sea horses for bias reduction:
White-breasted Kingfisher (the true class) has $R = 0.000530$ and $A_{sea\_horses} = -3.548267$, moving it to place 2.
Scott Oriole has $R = 0.000592$ and $A_{sea\_horses} = -0.736513$, moving it from place 2 to place 14.
Cape May Warbler has $R = 0.000601$ and $A_{sea\_horses} = -0.059679$, moving it from place 3 to place 19.
Yellow-bellied Flycatcher has $R = 0.000665$ and $A_{sea\_horses} = 0.0383943$, moving it from place 5 to place 22.

This is again reaffirmed by several of those down-ranked classes showing clear vertical lines in figure 4.3b, Yellow-bellied Flycatcher and Cape May Warbler are marked with red lines.

Tree Swallows show a rather opposite problem to Indigo Buntings, as they are often misclassified as several different birds, few of which even have a noticeable bias. The example in figure 4.11 does not even contain the real solution in the top five (appendix C shows it to be place eleven), but is still corrected after bias reduction, without creating a significant bias for Tree Swallows in turn.

4.1.3 Including Descriptions of Seen Classes

As mentioned above, there were several reasons to not consider images labelled with training classes in the main part of this evaluation. While there are options to possibly detect if an image belongs to a class that was used in training, have yet to be fully explored. As such, some tests were run on validation and test image sets considering not only the 50 descriptions contained in each set, but also the 100 descriptions used in training.

Unsurprisingly, tripling the numbers of possible classes reduces accuracy in both cases. Accuracies for validation and test sets are reported in this section and in more detail for various $\mathcal{I}$ in appendix B. Confusion matrices will only be shown for validation data, as they show similar behaviour to those for test data, but stronger and therefore easier to visually distinguish.

Like in the previous section, the x-axis shows the predicted class and the y-axis the true class. A black line separates training and validation labels on the x-axis. All labels follow the same order as in appendix A.

Figure 4.12 shows the confusion matrix of the above without bias reduction. As expected, barely any images are classified correctly (4.46%), though a fine diagonal line is visible. On test data only 2.8% T1 accuracy is reached.
T5 accuracy is remarkably higher with 23.64% and 20.25%, an increase by a factor of over 5 in both cases. In comparison, the factor between T5 and T1 accuracy when only

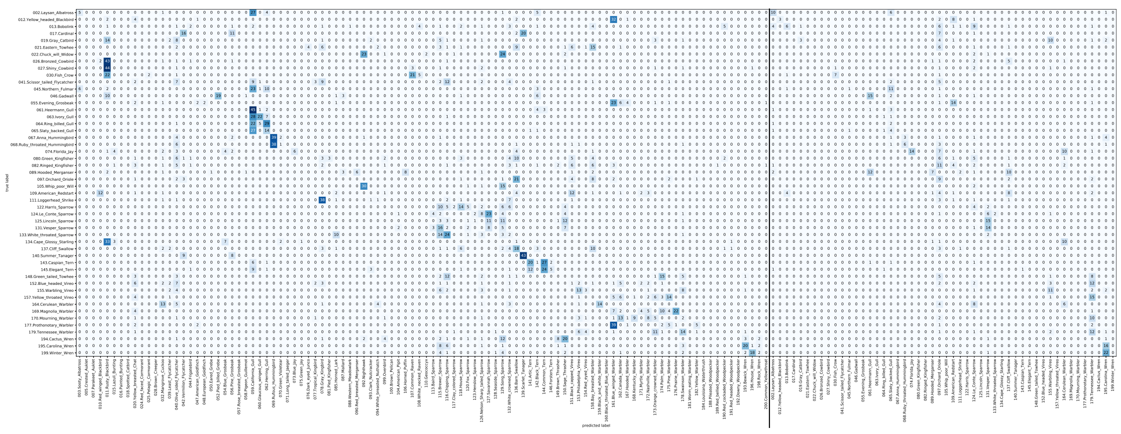


Figure 4.12: Confusion matrix of T1 classification on validation images with validation and training texts without any de-bias measures.

considering validation or set labels is slightly higher than 2. This shows that while there is s a strong bias towards some training classes, the true class still tends to rank highly.

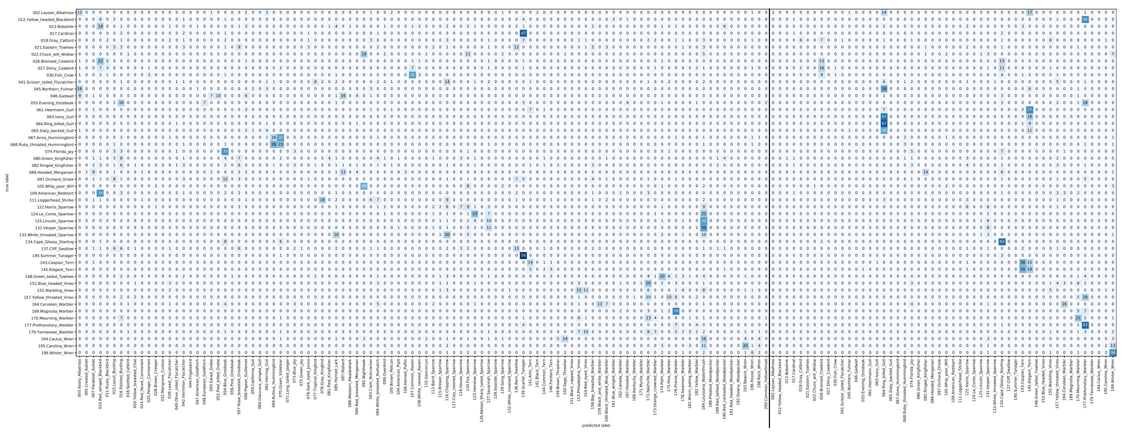


Figure 4.13: Confusion matrix of T1 classification on validation images with validation and training texts using images of cannons for bias reduction.

The best $\mathcal{I}$ for bias reduction here is a set of images of cannons, bringing T1 and T5 accuracy to 10.33% and 31.34% respectively. The utterly different choice is likely due to this set simply indiscriminately reducing scores for training classes, and would likely not work as well if images of those classes were being classified as well. A further indication for this is all image sets involving training images working especially well.
A confusion matrix for those results can be seen in figure 4.13, whereas figure 4.14 shows the difference of the previous two.

The best $\mathcal{I}$ for the validation set here also does not transfer well to the test set, where cannons only improve accuracies to 3.55% (T1) and 25.2% (T5). However, using training

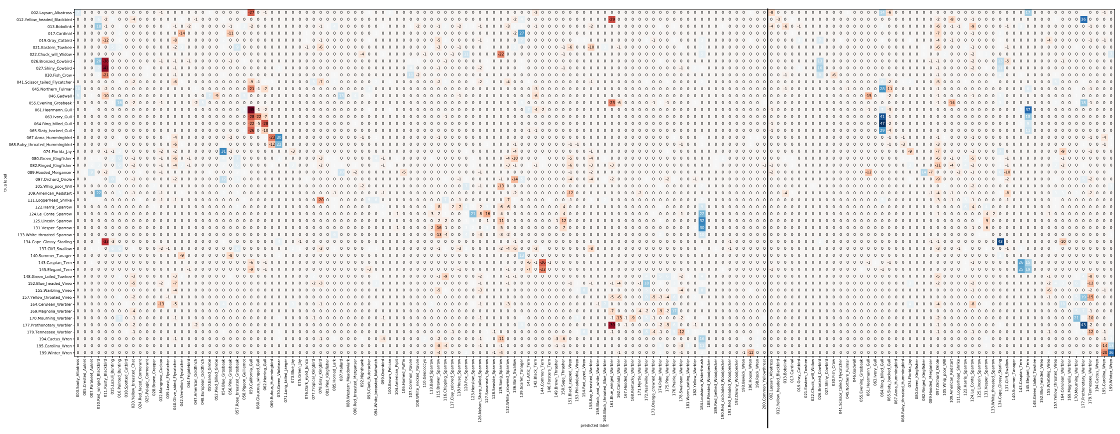


Figure 4.14: Difference between the confusion matrices in figures 4.13 and 4.12.

images for bias reduction works well there too, bringing T1 accuracy to 8.24% and T5 accuracy to 30.34%.

As can be seen, the right choice of $\mathcal{I}$ gives somewhat passable results, but it might be better to, as suggested before, use different descriptors for training classes in training and application.
Another possible approach would be to use the results on training classes without bias reduction as an indication if a given image belongs to one of them or a new class, and in the first case apply a normal non-zero-shot classifier. More about this idea in section 5.2.

4.1.4 Comparison to other Approaches

Not many previous works are directly comparable to this one, as there is no established benchmark for zero-shot learning with natural language descriptions.

One of the few is the paper that introduced the bird dataset used throughout this thesis [RALS16]. Reed et al. presented multiply models, several of which achieved a higher T1 classification accuracy than the approach presented here.

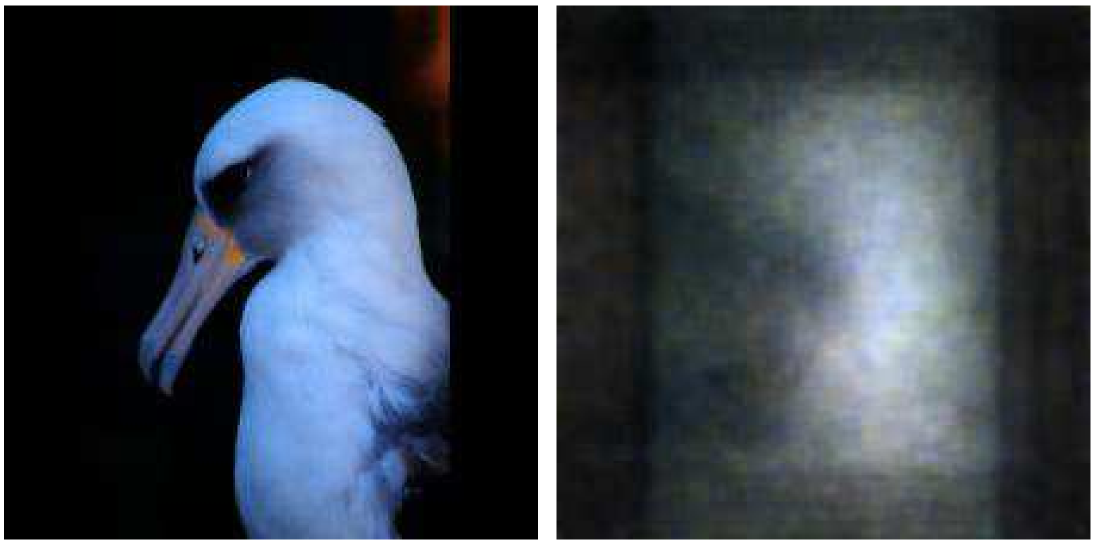
Their best model uses a word-level convolutional neural network (CNN) to reduce input dimensions and then continues with a recurrent neural network (RNN). This RNN is also fed information from another CNN that encoded the input image and learns to accumulate a score. For zero-shot classification a per-class average of text embeddings is used.
With this they achieve a T1 accuracy of 56.8% on the bird dataset, compared to the maximum of 23.87% reached here. However, their model requires several descriptions per image to be available during training, whereas this model already uses a single vector per class that could be replaced with a singular description as long as it contains enough information about the class. Their T5 accuracy is not reported.

Another work that uses at least similar data is by Ba et al. [BSFS15]. They use the same image dataset, but use word frequencies from Wikipedia articles about the classes as descriptions. Doing so in this thesis was considered as well, but many of the necessary article contained no visual description of the birds, only their behaviour. While this apparently did not make classification impossible, it likely contributed to the lower accuracy Ba et al. achieved. Further, they use a different split with only 40 unseen classes. All of this limits comparability.
Part of the paper is the evaluation of different performance metrics, most of which are only possible due to their models being binary classifiers for each class rather than a single multi-label classifier. The best accuracies on unseen classes they reach are T1 12% and T5 44%.

4.2 Explainability through Image Reconstruction

As mentioned before, having an autoencoder as distance function also offers a unique way of providing explainability, since the output can be restored to a human-"readable" form. The grey areas in figure 3.1 show two ways this can be done.

As explainability was not a main focus of this project, only SCAE, a very simple autoencoder, was constructed to test this (section 3.3.2). This leads to limited success, but gives hope for future work with more sophisticated methods. Section 5.2 gives some ideas for such improvements and how it may be applied to the text input or non-autoencoder vectorisation methods.



(a) Original image. (b) Reconstructed image.

Figure 4.15: Image of a Laysan Albatross before and after reconstruction.

Assuming the input was vectorised with an autoencoder as well, the process of reconstruction is simply to decode the output of the central autoencoder (uppermost image in figure 3.1). As can be seen in figure 4.15, the direct result of this is only a vague approximation of the original image.

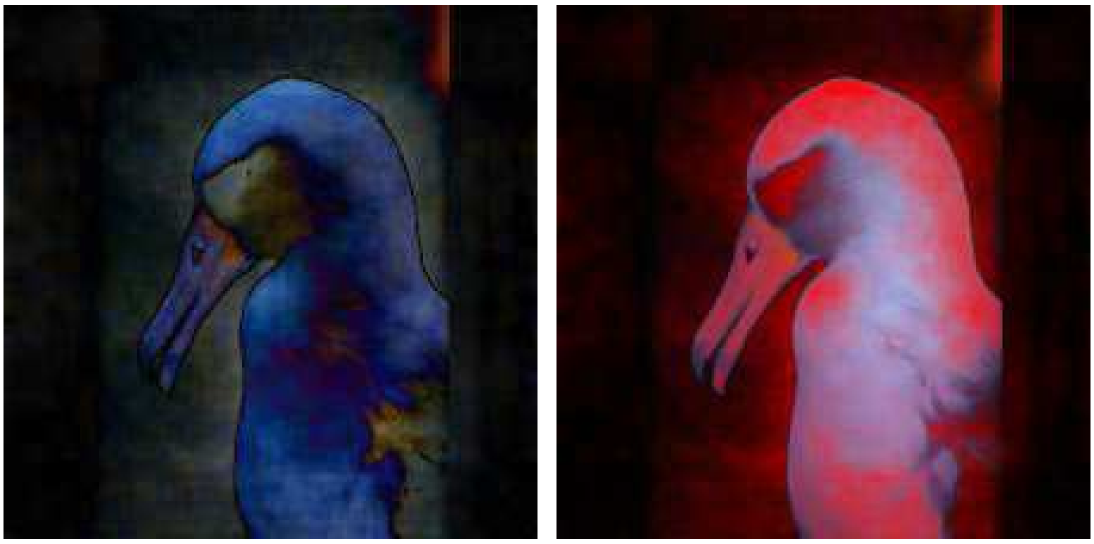

(a) Absolute difference. (b) Colour-mean of the difference as red overlay on the original image.

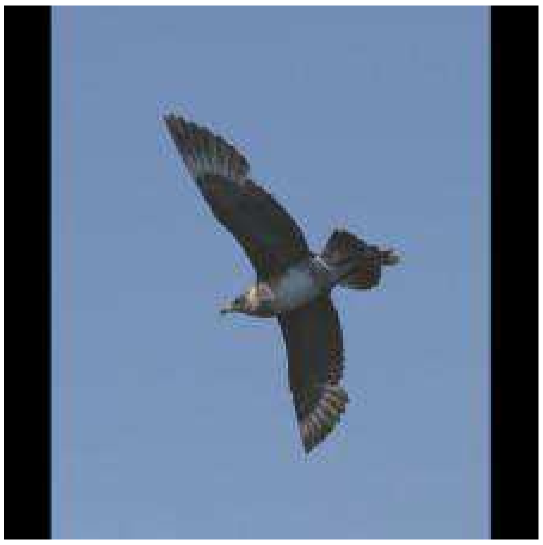
Figure 4.16: Difference between the images in figure 4.15.

The absolute difference between this and the original image already gives a more useful result. Figure 4.16 shows this difference on its own, and as a red overlay on the original image. The overlay is created using the mean of the colour values of the image as alpha. Lighter (left) or redder (right) indicate greater differences.

Similar to the results of the previous section, this mostly shows that large swathes of similar colours are the primary feature recognised, but approximate shape is considered as well. Further, an outline can be observed in many images (see also figure 4.17), but this might also stem from jpeg artefacts in the original image.

To reduce the influence of such artefacts, another option is to decode the squared difference between the image input and output of the central autoencoder, instead of comparing the final reconstruction (central reconstructed image in 4.17). This leads to figure 4.18. Due to the error going through the decoder, it is not completely clear what constitutes a better reconstruction here. The lighter areas on the border of the image suggest that those are probably lower errors, which leads to similar insights as the previous method.

While these results show what parts of the image the combined training of image and texts focusses on, it is perhaps more interesting to compare the outputs using different texts. Sadly, with the current model this only produces very scattered images with little more use than interpreting configurations of tea leaves.


(a) Original Image.

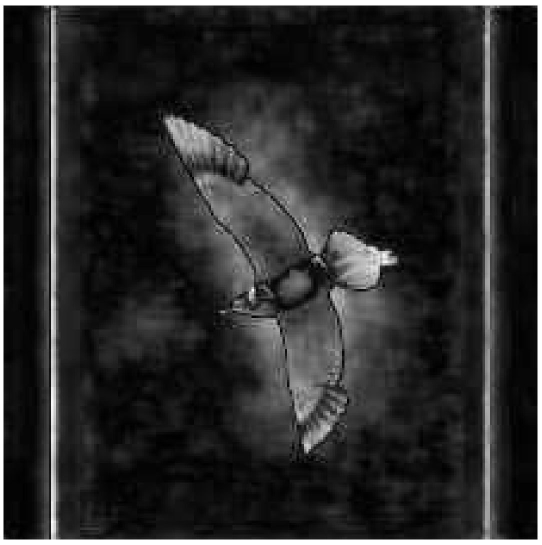
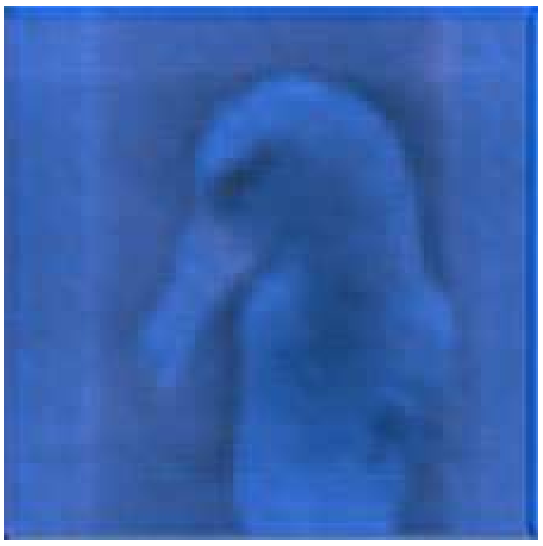

(b) Colour-mean of the absolute difference to the reconstruction.

Figure 4.17: Error on an image of a Pomarine Jaeger.


(a) Reconstruction.

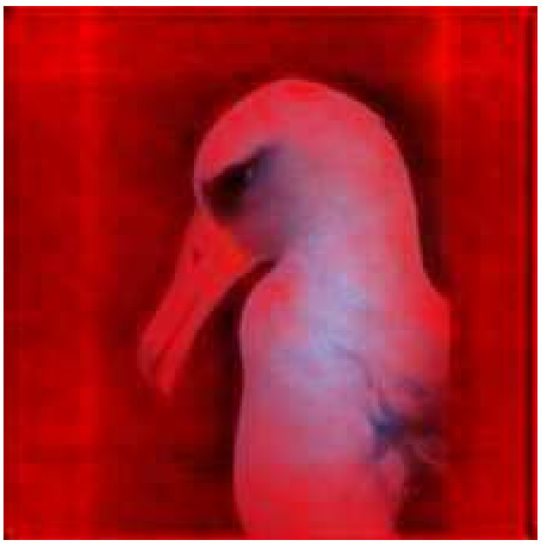

(b) Colour-mean of the reconstruction as red overlay on the original image.

Figure 4.18: Reconstruction of the squared error.

However, since some small differences can be observed, we may hope that using a more accurate autoencoder for vectorisation also leads to more interpretable results in that regard.

4.3 Summary

This chapter displayed and shortly discussed all results achieved with the model introduced in this thesis. Further discussion can be found in section 5.1.

The larger first part focussed on classification, especially in the case of unknown classes. As has been supposed in previous chapters as well, purely using the reconstruction error leads to accuracies only a bit better than random, but reducing bias with the methods described in section 3.5 led to large improvements. Using sea horses as normalisation set, a T1 accuracy on test data 23.87% could be reached and using the test data itself even 23.87%.
Previous work has achieved better accuracies on the same data, but with less versatile methods in regard to necessary training data. The method proposed here can compare any objects that can be encoded as a vector, whereas the outperforming work requires sequential descriptions for every single image.

No clear answer was found as to which image sets work best for bias reduction, though besides extreme similarity to the images classified, sets with large but not random variety in colour and shapes seemed to work best.
The results proof that the concept works as intended, with many avenues being open to improve accuracies in the future.

Further, the option of using the output of the autoencoder for explainability was explored. This as well shows promising first results, though so far nothing is shown not already suggested by the classification results.

CHAPTER 5

Discussion

5.1 Viability

The results shown in the previous chapter demonstrate it is generally possible to approximate Compression Distance with autoencoders.
While the pure reconstruction error on its own already showed the concept to work in principle, it still a showed major bias towards some classes, as surmised in the introduction. A normalisation method using unlabelled image sets was proposed to reduce the influence of those biases and improved T1 test accuracy by a factor of about 1.4 and T5 test accuracy by a bit over 1.3. It is assumed that this error normalisation may also improve accuracies in other comparison based methods, like those based on Siamese networks.
With that, the main contribution of this thesis is the Autoencoder Distance proposed in equation 3.9.

However, even with the best choice of normalisation set $\mathcal{I}$, the accuracies achieved are still below those achieved by Reed et al. [RALS16], the only work the author is aware of that uses the same inputs for both images and class descriptions. The differences between the models used are described in section 4.1.4.

Although this may be seen as an argument against the method proposed, it may still be worthwhile to continue exploring Kolmogorov based methods in this context. Most of the work in this thesis is not based on established zero-shot methods and therefore presents a new approach that still may be improved in many ways. The accuracies achieved are comparable to, or better than, other methods in their beginnings (e.g. [BSFS15], see section 4.1.4 for details).
Some possible paths for improvements are given in section 5.2.

Further, it was demonstrated that it is possible to restore the output of the central autoencoder to a human-interpretable format in two methods.

In the first, the image output is directly decoded and then compared to the input image. For the other the quadratic difference between the input and output vectors of the central autoencoder representing the image is calculated and the result is decoded. Details on both methods can be found in section 4.2.
Due to this only being a secondary objective, the option was not as thoroughly investigated as it may have been. In the current form it was still possible to see what parts of the image were restored best, which depend on the same encoding as the text reconstruction used for the autoencoder distance.
Due to this, the reconstructed image is slightly different when the same image is encoded with different descriptions. So far the differences are minimal and therefore not pictured here, so future work is needed to discover if these fluctuations follow a pattern.

5.2 Improvements and Future Work

As the model proposed in this thesis was mainly intended as a proof of concept, there are still many opportunities for improvements.

The most obvious avenue is the design of the central autoencoder. Information Distance and consequently Autoencoder Distance is based on shared patterns in the compared objects. If the autoencoder can be taught to more actively combine descriptions in its central layer, classification should improve as well.
A possible way to achieve this is the use of attention, specifically as it is used in unsupervised cases [YKW+19].

Many of the problems encountered were, as it commonly happens, due to lack of training data and computational resources. While a fine-tuned ResNeXt worked well enough for a proof of concept here, a more specialised network might improve classification results significantly. Especially a more complex convolutional autoencoder would further help with explainability, as described before.
It would also be possible to use the attribute vectors some datasets provide as input here. This has not been done as the focus was laid on the use of natural language description, and it is likely that a model designed for the use of such vectors makes far better use of them.

Similarly the BERT model could not be fine-tuned, due to both lack in training data, as well as lacking access to the necessary computational power during the early phases of the project.
Besides fine-tuning BERT, other methods of sentence vectorisation may be worth a try. Many offshoots of BERT have been developed in recent times, and some modern methods not based on transformers show promise, like the convolutional sentence encoder in [PDCT19].
The convolutional and recurrent neural network based methods in the work of Reed et al. are designed for a different form of comparison, but could be adapted for use here. This is unlikely to improve results, as transformer based methods like BERT, that were not

yet available when the paper was written, typically perform better on natural language related tasks.

This leads to another interesting opportunity. Using a decodable text vectorisation like in [PDCT19] may allow for similar reconstruction options as shown for images. In theory this is possible for BERT as well, but the implementation used sadly did not provide any such option.

Of course both image and text explainability is not limited to the use of autoencoders for vectorisations. As explainability became more and more popular, methods came up to map the input activations required to achieve a certain state in later layers. Using this, it would be possible to compare the mappings of the input and output of the central autoencoder, and the changes caused by different descriptions.
Further, it may be possible to train approximate decoders for vectorisations not intended for such. A lot of information required for a fully accurate reconstruction is likely lost in such a vectorisation, but a comparison of a decoding of the vectorisation with a decoding of the image output of the central autoencoder may still give a visualisation of what information is retained.

Another option considered in designing this model was to train the input vectorisation in combination with the central autoencoder. This, however, could only work if both vectorisation methods are autoencoders as well, since training with the error of only the central autoencoder would likely lead to the vectorisations only producing constants. Further research on autoencoders (e.g. [LBH18]) suggests that stacked training achieves better results. Fine-tuning afterwards would, again, only be possible with autoencoder-based vectorisations.

A different problem that came up is the reduced accuracy on new classes when training classes are considered as well (see section 4.1.3). One possible solution already mentioned is the use of two descriptions for each training class, one for training and one for application. Due to lack of such data, this could not be tested so far.
However, it might be possible to avoid the problem altogether. Ba et al. suggest a method where they measure the uncertainty in their classifier to detect images belonging to known labels and use a standard classifier for those [BSFS15]. It may be possible to use a similar approach with the model proposed in this thesis by considering the average reconstruction error an image has on known classes, but far more testing is required to find an appropriate threshold if one exists.

Lastly, the method proposed for bias reduction introduced a new parameter, the set of images $\mathcal{I}$ used for normalisation. Choosing one based on validation data generally seems to work well, but not perfectly so. Further research into the choice of $\mathcal{I}$ is therefore warranted.
One path to do so is the comparison of the standard deviation of the error different $\mathcal{I}$ produce. Due to how the normalisation process works, i.e., classes tend to be ranked lower after bias reduction when their standard deviation was low. It might therefore help to remove $\mathcal{I}$ that produce low standard deviations on classes few other $\mathcal{I}$ do so as well.

APPENDIX A

List of Classes in the Bird Dataset

Tables A.1, A.2, and A.3 list all classes in the bird dataset described in section 3.1.1 sorted by the split suggested by [RALS16].

Table A.1: Training classes of the bird dataset (section 3.1.1).

#	Class
005	Crested_Auklet
007	Parakeet_Auklet
010	Red_winged_Blackbird
011	Rusty_Blackbird
015	Lazuli_Bunting
016	Painted_Bunting
018	Spotted_Catbird
020	Yellow_breasted_Chat
024	Red_faced_Cormorant
025	Pelagic_Cormorant
028	Brown_Creeper
032	Mangrove_Cuckoo
039	Least_Flycatcher
040	Olive_sided_Flycatcher
042	Vermilion_Flycatcher
044	Frigatebird
047	American_Goldfinch
048	European_Goldfinch
050	Eared_Grebe
052	Pied_billed_Grebe
054	Blue_Grosbeak
056	Pine_Grosbeak
057	Rose_breasted_Grosbeak
058	Pigeon_Guillemot

#	Class
059	California_Gull
060	Glaucous_winged_Gull
062	Herring_Gull
069	Rufous_Hummingbird
070	Green_Violetear
071	Long_tailed_Jaeger
073	Blue_Jay
075	Green_Jay
076	Dark_eyed_Junco
077	Tropical_Kingbird
078	Gray_Kingbird
081	Pied_Kingfisher
085	Horned_Lark
087	Mallard
088	Western_Meadowlark
090	Red_breasted_Merganser
092	Nighthawk
093	Clark_Nutcracker
094	White_breasted_Nuthatch
099	Ovenbird
100	Brown_Pelican
104	American_Pipit
106	Horned_Puffin
107	Common_Raven
108	White_necked_Raven
110	Geococcyx
113	Baird_Sparrow
115	Brewer_Sparrow
116	Chipping_Sparrow
117	Clay_colored_Sparrow
118	House_Sparrow
120	Fox_Sparrow
123	Henslow_Sparrow
126	Nelson_Sharp_tailed_Sparrow
127	Savannah_Sparrow
128	Seaside_Sparrow
129	Song_Sparrow
132	White_crowned_Sparrow
136	Barn_Swallow
139	Scarlet_Tanager
141	Artic_Tern
142	Black_Tern
144	Common_Tern
146	Forsters_Tern
149	Brown_Thrasher
150	Sage_Thrasher
151	Black_capped_Vireo

#	Class
153	Philadelphia_Vireo
154	Red_eyed_Vireo
158	Bay_breasted_Warbler
159	Black_and_white_Warbler
160	Black_throated_Blue_Warbler
161	Blue_winged_Warbler
162	Canada_Warbler
167	Hooded_Warbler
168	Kentucky_Warbler
171	Myrtle_Warbler
172	Nashville_Warbler
173	Orange_crowned_Warbler
174	Palm_Warbler
175	Pine_Warbler
176	Prairie_Warbler
178	Swainson_Warbler
181	Worm_eating_Warbler
182	Yellow_Warbler
184	Louisiana_Waterthrush
188	Pileated_Woodpecker
189	Red_bellied_Woodpecker
190	Red_cockaded_Woodpecker
191	Red_headed_Woodpecker
192	Downy_Woodpecker
193	Bewick_Wren
196	House_Wren
198	Rock_Wren
200	Common_Yellowthroat

Table A.2: Validation classes of the bird dataset (section 3.1.1).

#	Class
012	Yellow_headed_Blackbird
013	Bobolink
017	Cardinal
019	Gray_Catbird
021	Eastern_Towhee
022	Chuck_will_Widow
026	Bronzed_Cowbird
027	Shiny_Cowbird
030	Fish_Crow
041	Scissor_tailed_Flycatcher
045	Northern_Fulmar
046	Gadwall
055	Evening_Grosbeak
061	Heermann_Gull

#	Class
063	Ivory_Gull
064	Ring_billed_Gull
065	Slaty_backed_Gull
067	Anna_Hummingbird
068	Ruby_throated_Hummingbird
074	Florida_Jay
080	Green_Kingfisher
082	Ringed_Kingfisher
089	Hooded_Merganser
097	Orchard_Oriole
105	Whip_poor_Will
109	American_Redstart
111	Loggerhead_Shrike
122	Harris_Sparrow
124	Le_Conte_Sparrow
125	Lincoln_Sparrow
131	Vesper_Sparrow
133	White_throated_Sparrow
134	Cape_Glossy_Starling
137	Cliff_Swallow
140	Summer_Tanager
143	Caspian_Tern
145	Elegant_Tern
148	Green_tailed_Towhee
152	Blue_headed_Vireo
155	Warbling_Vireo
157	Yellow_throated_Vireo
164	Cerulean_Warbler
169	Magnolia_Warbler
170	Mourning_Warbler
177	Prothonotary_Warbler
179	Tennessee_Warbler
194	Cactus_Wren
195	Carolina_Wren
199	Winter_Wren

Table A.3: Test classes of the bird dataset (section 3.1.1).

#	Class
004	Groove_billed_Ani
006	Least_Auklet
008	Rhinoceros_Auklet
009	Brewers_Blackbird
014	Indigo_Bunting
023	Brandts_Cormorant
029	American_Crow

#	Class
031	Black_billed_Cuckoo
033	Yellow_billed_Cuckoo
034	Gray_crowned_Rosy_Finch
035	Purple_Finch
036	Northern_Flicker
037	Acadian_Flycatcher
038	Great_Crested_Flycatcher
043	Yellow_bellied_Flycatcher
049	Boat_tailed_Grackle
051	Horned_Grebe
053	Western_Grebe
066	Western_Gull
072	Pomarine_Jaeger
079	Belted_Kingfisher
083	White_breasted_Kingfisher
084	Red_legged_Kittiwake
086	Pacific_Loon
091	Mockingbird
095	Baltimore_Oriole
096	Hooded_Oriole
098	Scotts_Oriole
101	White_Pelican
102	Western_Wood_Pewee
103	Sayornis
112	Great_Grey_Shrike
114	Black_throated_Sparrow
119	Field_Sparrow
121	Grasshopper_Sparrow
130	Tree_Sparrow
135	Bank_Swallow
138	Tree_Swallow
147	Least_Tern
156	White_eyed_Vireo
163	Cape_May_Warbler
165	Chestnut_sided_Warbler
166	Golden_winged_Warbler
180	Wilsons_Warbler
183	Northern_Waterthrush
185	Bohemian_Waxwing
186	Cedar_Waxwing
187	American_Three_toed_Woodpecker
197	Marsh_Wren

APPENDIX B

Accuracies for various $\mathcal{I}$

Table B.1 shows the T1 and T5 accuracies achieved using various image sets $\mathcal{I}$ for bias reduction (see section 3.5). The image vectorisation is done with the fine-tuned ResNeXt model described in section 3.3.1.
Columns labelled 'Valid' and 'Test' show evaluations on only the classes appearing in the validation and test sets respectively (50 classes each), whereas columns labelled with 'Valid + Train' also consider training classes, though still only use images from the validation set (150 classes total). Entries are sorted by their validation T1 accuracy.
The names of image sets $\mathcal{I}$ are as they appear in their respective databases (see sections 3.1.2.1 and 3.1.2.2), with the following exceptions:

- 'valid': Images from the validation set
- 'test': Images from the test set
- 'train' Images from the training set
- 'trval': Images from the training and validation sets
- 'trtest': Images from the training and test sets
- Any with suffix '_awa2': Images from AwA2. The names before the suffix are as they appear in the database.

Excerpts of this table are shown in section 4.1.

Table B.1: T1 and T5 accuracies achieved with different images sets $\mathcal{I}$ for bias reduction, sorted according to T1 validation accuracy.

Image Set $\mathcal{I}$	Valid T1	Test T1	Valid + Train T1	Test + Train T1	Valid T5	Test T5	Valid + Train T5	Test + Train T5	#
valid	29.28%	22.60%	6.92%	2.80%	66.80%	57.38%	39.11%	28.50%	(2961)
sea_horse	28.03%	23.25%	7.57%	2.76%	63.69%	57.14%	33.16%	27.17%	(57)
test	28.03%	23.87%	7.67%	3.95%	65.52%	58.23%	38.87%	30.21%	(2933)
trval	27.63%	23.08%	7.90%	4.02%	65.38%	58.27%	37.35%	30.38%	(8855)

Image Set $\mathcal{I}$	Valid T1	Test T1	Valid + Train T1	Test + Train T1	Valid T5	Test T5	Valid + Train T5	Test + Train T5	#
octopus	27.52%	23.59%	9.22%	2.80%	63.05%	55.71%	35.73%	28.44%	(35)
buddha	27.02%	21.17%	6.42%	2.56%	61.50%	55.47%	33.74%	27.24%	(85)
garfield	26.88%	21.58%	7.77%	3.03%	62.21%	55.61%	34.45%	24.82%	(34)
trtest	26.82%	23.70%	8.00%	4.43%	64.57%	58.03%	37.28%	30.31%	(8827)
crayfish	26.58%	21.41%	5.91%	3.24%	63.86%	57.35%	33.94%	28.81%	(70)
scorpion	26.48%	22.50%	9.79%	5.32%	61.26%	57.31%	35.46%	30.48%	(84)
butterfly	26.41%	21.24%	7.94%	4.71%	63.19%	57.55%	35.26%	28.06%	(91)
metronome	26.34%	21.10%	6.89%	2.69%	62.65%	54.48%	33.54%	25.16%	(32)
lamp	26.27%	22.60%	6.55%	3.07%	63.80%	56.53%	33.64%	28.50%	(61)
minaret	26.17%	22.91%	7.23%	3.07%	62.14%	56.46%	33.91%	27.51%	(76)
ant	26.14%	20.35%	7.80%	2.93%	63.09%	57.55%	33.81%	27.17%	(42)
umbrella	26.14%	21.21%	6.55%	3.75%	59.51%	55.78%	31.81%	27.75%	(75)
train	26.11%	22.84%	8.24%	5.05%	63.53%	57.07%	37.05%	30.34%	(5894)
ceiling_fan	26.07%	21.28%	7.06%	3.10%	62.95%	57.18%	34.62%	27.55%	(47)
BACKGROUND_Google	26.00%	23.25%	6.69%	3.17%	60.55%	56.29%	33.03%	26.90%	(467)
strawberry	25.84%	21.62%	7.70%	3.99%	64.10%	57.07%	33.03%	25.40%	(35)
platypus	25.80%	20.15%	5.88%	2.86%	60.93%	54.72%	31.78%	25.03%	(34)
bonsai	25.77%	21.58%	7.40%	4.67%	60.93%	55.71%	33.60%	29.01%	(128)
nautilus	25.73%	22.91%	9.02%	3.14%	61.43%	56.56%	35.33%	29.22%	(55)
brontosaurus	25.63%	21.58%	6.42%	3.48%	58.09%	53.77%	30.83%	25.54%	(43)
electric_guitar	25.63%	21.62%	6.55%	3.51%	60.11%	55.71%	33.10%	26.53%	(75)
hippopotamus_awa2	25.60%	21.62%	5.30%	1.33%	60.49%	56.12%	31.27%	24.28%	(684)
horse_awa2	25.50%	21.58%	7.73%	2.90%	59.74%	55.27%	33.70%	26.63%	(1645)
laptop	25.50%	20.73%	9.08%	3.41%	60.05%	55.27%	33.16%	27.34%	(81)
walrus_awa2	25.50%	21.96%	5.30%	2.08%	60.93%	57.28%	30.53%	25.95%	(215)
chandelier	25.43%	22.20%	8.88%	3.51%	62.68%	57.07%	33.98%	28.40%	(107)
dolphin	25.36%	22.60%	5.61%	2.28%	60.25%	55.06%	30.06%	25.16%	(65)
seal_awa2	25.36%	21.92%	5.88%	2.08%	59.61%	56.02%	30.23%	24.96%	(988)
pig_awa2	25.26%	21.38%	8.14%	1.77%	61.33%	55.71%	32.96%	25.91%	(713)
water_lilly	25.26%	19.71%	8.98%	2.18%	63.42%	52.06%	34.35%	25.16%	(37)
bat_awa2	25.23%	21.55%	8.00%	2.59%	62.85%	57.52%	34.08%	26.87%	(383)
mandolin	25.23%	19.47%	7.73%	3.07%	62.31%	56.94%	33.50%	25.84%	(43)
german+shepherd_awa2	25.13%	20.32%	6.55%	2.42%	58.93%	54.01%	33.74%	24.51%	(1033)
pyramid	25.09%	22.67%	5.94%	3.44%	61.94%	55.44%	32.32%	27.58%	(57)
persian+cat_awa2	25.06%	20.35%	6.69%	1.60%	60.86%	52.95%	33.43%	25.23%	(747)
chihuahua_awa2	25.03%	20.22%	9.29%	3.00%	59.37%	54.79%	34.92%	26.49%	(567)
mayfly	25.03%	19.33%	5.74%	2.90%	64.51%	54.35%	30.33%	24.00%	(40)
ewer	24.99%	21.65%	7.84%	3.51%	61.70%	55.64%	33.20%	26.36%	(85)
pagoda	24.96%	20.01%	7.73%	4.13%	59.24%	52.81%	31.17%	23.15%	(47)
hawksbill	24.86%	20.42%	5.57%	3.55%	60.49%	55.74%	31.88%	27.14%	(100)
pigeon	24.86%	19.09%	6.32%	2.73%	58.63%	52.44%	33.27%	24.79%	(45)
sheep_awa2	24.86%	20.63%	6.82%	3.17%	59.64%	55.03%	30.40%	27.55%	(1420)
trilobite	24.86%	20.56%	8.21%	4.98%	60.59%	54.65%	33.74%	27.04%	(86)
siamese+cat_awa2	24.79%	20.22%	7.77%	3.55%	58.46%	54.18%	34.11%	25.95%	(500)
stapler	24.79%	20.73%	5.94%	2.42%	59.84%	53.94%	30.80%	24.38%	(45)
crab	24.76%	20.93%	8.38%	3.65%	63.15%	55.98%	34.58%	28.30%	(73)
wolf_awa2	24.76%	20.76%	6.32%	2.28%	62.88%	55.51%	32.35%	26.22%	(589)
airplanes	24.72%	21.00%	5.37%	2.49%	59.54%	53.87%	32.62%	26.32%	(800)
snoopy	24.69%	21.99%	8.14%	2.69%	58.97%	54.42%	33.54%	26.05%	(35)
dolphin_awa2	24.65%	21.48%	4.80%	2.52%	58.66%	55.06%	30.23%	25.37%	(946)
wrench	24.65%	22.16%	7.87%	2.97%	60.11%	55.13%	30.87%	26.15%	(39)
barrel	24.62%	21.99%	10.10%	3.03%	60.11%	56.73%	34.35%	27.11%	(47)
stop_sign	24.62%	20.42%	5.91%	3.07%	59.78%	53.43%	30.43%	24.04%	(64)
binocular	24.59%	22.84%	7.19%	3.95%	58.76%	55.81%	32.86%	27.96%	(33)
cow_awa2	24.59%	21.45%	7.53%	3.78%	57.89%	54.79%	32.35%	25.40%	(1338)
kangaroo	24.59%	22.37%	7.63%	4.40%	62.21%	56.36%	33.03%	27.11%	(86)
menorah	24.59%	21.10%	7.70%	4.02%	61.63%	54.86%	33.06%	26.87%	(87)
anchor	24.55%	20.01%	8.98%	4.36%	60.49%	55.44%	33.91%	26.49%	(42)
dollar_bill	24.55%	22.84%	7.84%	2.73%	60.45%	53.15%	30.70%	24.14%	(52)
otter_awa2	24.52%	20.87%	6.01%	2.22%	59.51%	55.61%	30.12%	24.62%	(758)
ketch	24.48%	22.20%	7.77%	2.73%	60.96%	54.21%	33.77%	24.89%	(114)
llama	24.45%	20.87%	6.96%	3.92%	60.55%	55.47%	31.27%	27.58%	(78)
lobster	24.42%	20.83%	8.85%	2.18%	63.26%	57.07%	33.47%	24.55%	(41)
brain	24.35%	22.57%	6.42%	4.36%	59.47%	55.98%	32.86%	28.47%	(98)
chair	24.35%	21.38%	8.31%	3.48%	62.92%	56.12%	33.57%	24.58%	(62)
accordion	24.32%	20.97%	6.52%	2.90%	58.56%	54.59%	32.12%	25.47%	(55)
stegosaurus	24.28%	22.30%	7.77%	3.27%	61.80%	56.90%	31.81%	25.78%	(59)
yin_yang	24.28%	20.56%	7.26%	2.42%	57.58%	53.46%	32.59%	24.38%	(60)
weasel_awa2	24.25%	19.81%	8.85%	3.34%	61.20%	56.77%	32.62%	27.07%	(272)
Motorbikes	24.21%	19.95%	8.85%	2.56%	56.50%	53.22%	32.39%	23.59%	(798)

Image Set $\mathcal{I}$	Valid T1	Test T1	Valid + Train T1	Test + Train T1	Valid T5	Test T5	Valid + Train T5	Test + Train T5	#
flamingo	24.21%	19.50%	6.92%	3.03%	60.89%	54.48%	28.88%	26.49%	(67)
gramophone	24.18%	22.67%	7.43%	3.03%	62.04%	57.01%	34.72%	29.32%	(51)
deer_awa2	24.15%	21.68%	8.14%	3.48%	62.78%	56.05%	34.11%	27.51%	(1344)
grizzly+bear_awa2	24.15%	19.91%	7.97%	3.00%	59.57%	55.23%	31.37%	24.86%	(852)
humpback+whale_awa2	24.15%	20.66%	3.75%	3.03%	56.06%	51.89%	28.67%	23.32%	(709)
killer+whale_awa2	24.05%	19.71%	5.03%	2.52%	55.89%	51.59%	29.69%	23.12%	(291)
wheelchair	24.05%	21.00%	9.83%	3.89%	59.30%	55.88%	34.75%	26.08%	(59)
panda	24.01%	20.87%	5.61%	4.09%	56.60%	53.36%	30.97%	27.51%	(38)
chimpanzee_awa2	23.98%	22.57%	6.65%	3.10%	58.93%	55.51%	30.19%	26.12%	(728)
cougar_face	23.94%	20.25%	8.34%	2.32%	60.79%	53.56%	32.46%	24.17%	(69)
elephant_awa2	23.94%	21.99%	6.21%	2.80%	57.24%	54.72%	30.06%	23.05%	(1038)
giant+panda_awa2	23.91%	20.70%	7.40%	3.14%	56.23%	51.89%	32.49%	26.83%	(874)
scissors	23.91%	21.34%	7.19%	1.60%	60.66%	54.86%	33.27%	26.73%	(39)
cup	23.88%	19.77%	6.89%	3.78%	58.60%	53.94%	30.23%	26.53%	(57)
watch	23.88%	20.46%	9.05%	3.00%	61.77%	53.49%	32.49%	25.67%	(239)
helicopter	23.78%	19.40%	5.10%	2.83%	59.64%	54.86%	31.04%	24.11%	(88)
lotus	23.78%	18.62%	7.29%	3.99%	62.41%	52.68%	30.23%	24.11%	(66)
blue+whale_awa2	23.74%	21.21%	3.34%	3.61%	57.58%	53.02%	28.74%	23.83%	(174)
saxophone	23.71%	19.95%	9.49%	3.07%	60.01%	52.78%	31.98%	23.08%	(40)
dragonfly	23.67%	18.82%	7.33%	2.76%	62.01%	55.54%	34.31%	26.76%	(68)
hamster_awa2	23.67%	20.05%	8.38%	2.69%	58.87%	52.30%	32.93%	25.64%	(779)
spider+monkey_awa2	23.67%	21.21%	8.75%	3.17%	60.35%	54.99%	33.94%	25.67%	(291)
bass	23.64%	20.15%	7.46%	2.93%	61.94%	55.81%	32.02%	24.86%	(54)
moose_awa2	23.61%	21.17%	7.63%	3.14%	58.29%	55.03%	31.44%	23.42%	(704)
antelope_awa2	23.57%	21.24%	9.05%	3.68%	60.11%	54.04%	33.40%	26.90%	(1046)
mouse_awa2	23.51%	20.56%	7.50%	2.49%	61.23%	54.35%	31.88%	27.34%	(185)
car_side	23.47%	20.70%	5.74%	2.52%	57.62%	53.43%	28.84%	24.92%	(123)
ferry	23.47%	20.01%	6.72%	2.39%	59.71%	54.76%	31.64%	25.20%	(67)
rabbit_awa2	23.47%	20.25%	9.52%	4.88%	59.47%	54.59%	34.38%	27.28%	(1088)
rooster	23.47%	17.35%	8.34%	2.83%	55.18%	49.91%	30.06%	23.90%	(49)
buffalo_awa2	23.40%	22.64%	6.99%	3.92%	56.81%	53.53%	30.90%	24.21%	(895)
collie_awa2	23.40%	19.60%	7.73%	2.22%	56.30%	53.73%	32.52%	23.56%	(1028)
grand_piano	23.40%	20.90%	8.54%	3.07%	61.20%	55.23%	30.87%	23.73%	(99)
joshua_tree	23.37%	20.73%	7.73%	4.19%	59.17%	53.49%	32.12%	25.67%	(64)
pizza	23.37%	20.90%	8.21%	2.90%	60.96%	55.54%	33.70%	26.29%	(53)
tick	23.34%	20.05%	7.97%	3.31%	62.11%	57.11%	35.56%	24.68%	(49)
fox_awa2	23.27%	20.32%	9.15%	2.32%	61.53%	53.43%	34.04%	25.03%	(664)
hedgehog	23.24%	20.97%	8.00%	3.38%	60.96%	54.65%	31.14%	26.73%	(54)
inline_skate	23.17%	19.13%	9.35%	3.20%	55.93%	52.54%	30.29%	22.37%	(31)
raccoon_awa2	23.10%	19.50%	9.35%	2.86%	57.85%	54.11%	33.27%	24.96%	(512)
giraffe_awa2	23.03%	20.90%	7.63%	3.78%	59.47%	52.78%	30.97%	25.23%	(1202)
gorilla_awa2	23.03%	21.75%	6.86%	2.69%	57.21%	54.82%	31.78%	25.54%	(872)
rat_awa2	23.03%	20.29%	9.25%	3.07%	58.05%	52.54%	32.59%	25.26%	(310)
rhinoceros_awa2	23.03%	21.17%	6.65%	3.10%	56.40%	53.87%	29.48%	23.66%	(696)
headphone	23.00%	20.12%	8.14%	3.65%	57.85%	53.73%	32.76%	25.30%	(42)
cougar_body	22.97%	19.26%	8.65%	3.24%	61.06%	55.64%	30.97%	24.58%	(47)
elephant	22.97%	19.95%	6.75%	2.39%	56.60%	53.73%	28.03%	22.13%	(64)
random	22.97%	17.97%	7.23%	3.65%	60.05%	52.47%	30.87%	21.10%	(500)
Faces	22.90%	21.58%	7.57%	2.32%	57.11%	53.63%	29.89%	24.72%	(435)
okapi	22.90%	19.57%	9.32%	3.07%	58.02%	53.15%	32.69%	25.88%	(39)
cellphone	22.83%	20.12%	8.14%	2.59%	58.26%	53.97%	31.81%	24.55%	(59)
mole_awa2	22.83%	18.96%	8.27%	3.03%	55.93%	51.45%	30.50%	24.31%	(100)
beaver_awa2	22.80%	20.73%	6.72%	2.08%	57.85%	54.28%	29.85%	23.32%	(193)
Faces_easy	22.73%	20.56%	8.48%	3.44%	57.62%	54.21%	29.89%	24.41%	(435)
polar+bear_awa2	22.73%	21.41%	6.86%	2.69%	61.06%	54.28%	28.06%	24.58%	(868)
bobcat_awa2	22.70%	18.34%	8.38%	3.48%	60.69%	51.11%	31.61%	24.04%	(630)
camera	22.56%	21.07%	5.61%	2.93%	58.26%	51.52%	28.57%	22.71%	(50)
beaver	22.49%	18.41%	5.78%	2.80%	56.74%	50.87%	27.69%	20.87%	(46)
Leopards	22.39%	18.41%	7.90%	4.06%	59.54%	53.97%	31.00%	23.35%	(200)
gerenuk	22.36%	18.72%	6.86%	3.07%	60.93%	53.05%	31.71%	21.82%	(34)
ox_awa2	22.36%	20.32%	7.70%	3.55%	55.86%	53.46%	30.53%	23.90%	(728)
emu	22.29%	18.99%	4.90%	4.71%	56.23%	53.32%	26.75%	23.87%	(53)
cannon	22.26%	19.98%	10.33%	3.55%	58.56%	54.14%	31.34%	25.20%	(43)
rhino	22.26%	19.30%	5.71%	3.51%	57.51%	53.53%	29.11%	22.20%	(59)
schooner	22.22%	21.07%	7.67%	3.07%	59.84%	52.98%	31.04%	25.47%	(63)
tiger_awa2	22.19%	18.17%	8.17%	3.65%	58.93%	51.99%	30.77%	22.84%	(877)
lion_awa2	22.12%	17.76%	7.50%	3.89%	59.78%	51.35%	30.16%	22.71%	(1019)
ibis	22.05%	17.15%	6.01%	3.72%	56.33%	51.04%	24.62%	22.95%	(80)
windsor_chair	22.05%	18.62%	8.71%	4.06%	61.53%	52.10%	30.73%	25.40%	(56)
starfish	22.02%	20.46%	6.01%	2.18%	62.14%	53.87%	30.90%	23.15%	(86)

Image Set $\mathcal{I}$	Valid T1	Test T1	Valid + Train T1	Test + Train T1	Valid T5	Test T5	Valid + Train T5	Test + Train T5	#
leopard_awa2	21.92%	18.17%	7.67%	3.55%	59.57%	52.17%	31.27%	21.75%	(720)
squirrel_awa2	21.92%	19.13%	9.02%	2.97%	57.78%	52.85%	32.76%	25.03%	(1200)
sunflower	21.85%	19.23%	8.98%	3.00%	62.95%	52.51%	31.54%	24.17%	(85)
skunk_awa2	21.82%	19.16%	8.88%	3.00%	54.00%	50.49%	31.64%	22.47%	(188)
wild_cat	21.75%	19.84%	8.68%	2.39%	61.47%	54.65%	30.97%	23.97%	(34)
crocodile	21.68%	17.70%	8.95%	3.14%	58.90%	52.74%	30.77%	22.50%	(50)
revolver	21.68%	19.81%	6.92%	3.41%	54.58%	52.34%	31.34%	23.56%	(82)
dalmatian_awa2	21.48%	19.77%	8.17%	2.01%	55.32%	50.12%	29.52%	20.90%	(549)
flamingo_head	21.24%	19.16%	5.50%	2.15%	58.60%	51.21%	27.56%	23.90%	(45)
zebra_awa2	20.91%	18.62%	9.56%	4.47%	55.35%	51.41%	30.63%	23.39%	(1170)
euphonium	20.80%	18.92%	10.06%	3.58%	56.97%	53.32%	31.41%	22.91%	(64)
crocodile_head	20.67%	19.88%	8.48%	2.93%	59.00%	52.71%	28.44%	23.56%	(51)
dalmatian	19.42%	19.09%	7.46%	2.08%	57.35%	52.27%	28.47%	20.53%	(67)
soccer_ball	19.12%	19.09%	8.31%	2.56%	55.72%	52.81%	25.60%	20.70%	(64)
None	17.33%	16.30%	4.46%	2.80%	47.38%	43.81%	23.64%	20.25%	(0)

APPENDIX C

Selected Individual Results

Full results for the three individual images discussed in section 4.1.2. $R(I,T)$ is hereby the text reconstruction error as defined in equation 3.2, and $A_{\mathcal{I}}(I,T)$ the Autoencoder Distance defined in equation 3.9.

To save space, columns labelled $\#T$ only give the numbering of classes. The names can be found in appendix A.

Table C.1: Full results for I as the Indigo Bunting in figure 4.9.

#	$R(I,T)$	$\#T$	$A_{sea\_horses}(I,T)$	$\#T$	$A_{test}(I,T)$	$\#T$
1	0.000549	083	−4.564322	014	−2.528211	014
2	0.000745	138	−3.478106	083	−1.884309	083
3	0.000793	095	−2.665579	138	−1.560511	138
4	0.000804	049	−2.504240	009	−1.465517	009
5	0.000935	014	−2.252132	051	−1.190418	051
6	0.000951	156	−1.558213	049	−1.007489	049
7	0.000984	006	−0.820472	095	−0.412795	095
8	0.000999	043	−0.685278	053	−0.303238	006
9	0.001026	008	−0.304119	185	−0.252425	008
10	0.001062	163	−0.292540	096	−0.162729	023
11	0.001081	051	−0.077329	008	−0.130145	053
12	0.001086	035	0.148064	084	0.066383	029
13	0.001134	079	0.646196	006	0.077494	185
14	0.001173	098	0.715314	023	0.204967	096
15	0.001179	038	0.941319	156	0.435466	084
16	0.001186	096	1.361169	029	0.733414	079
17	0.001201	023	1.479536	186	0.837785	186
18	0.001237	031	1.633388	036	0.857270	156
19	0.001253	053	1.944241	079	0.982149	004
20	0.001264	066	2.251958	098	1.182192	036

#	$R(I,T)$	#T	$A_{sea\_horses}(I,T)$	#T	$A_{test}(I,T)$	#T
21	0.001277	086	2.254295	043	1.402957	035
22	0.001290	185	2.504939	038	1.519953	098
23	0.001316	091	2.642324	163	1.583461	163
24	0.001330	186	2.875505	147	1.629241	043
25	0.001347	112	2.946767	034	1.679566	037
26	0.001414	034	3.054409	086	1.719536	031
27	0.001428	180	3.144106	035	1.782594	033
28	0.001435	009	3.406167	114	1.823253	038
29	0.001462	135	3.548604	112	1.914408	121
30	0.001468	037	3.822119	033	1.927405	147
31	0.001469	084	3.951225	066	1.972980	086
32	0.001500	033	4.239897	004	2.171114	114
33	0.001513	114	4.397454	166	2.187786	119
34	0.001523	187	4.818616	165	2.217805	103
35	0.001541	072	4.937165	031	2.224244	112
36	0.001566	029	5.200638	037	2.304851	066
37	0.001578	130	5.594457	130	2.345478	091
38	0.001586	119	5.612985	121	2.355086	034
39	0.001772	121	5.857708	180	2.548031	135
40	0.001887	036	6.264458	119	2.615060	180
41	0.001913	103	6.514968	135	2.675150	130
42	0.001914	183	6.604708	103	2.755081	197
43	0.002033	165	6.668827	187	2.824020	187
44	0.002070	102	8.515936	101	2.824239	166
45	0.002088	197	8.723020	091	2.862465	165
46	0.002153	166	8.875541	072	2.879075	102
47	0.002288	147	8.927152	197	2.939401	001
48	0.002503	001	10.490552	183	2.970409	101
49	0.002595	004	10.998594	001	3.293729	183
50	0.003146	101	12.320220	102	3.600309	072

Table C.2: Full results for I as the White-breasted Kingfisher in figure 4.10.

#	$R(I,T)$	#T	$A_{sea\_horses}(I,T)$	#T	$A_{test}(I,T)$	#T
1	0.000530	083	−4.020404	014	−2.216745	014
2	0.000592	098	−3.548267	083	−1.925928	083
3	0.000601	163	−2.663526	051	−1.413098	051
4	0.000611	086	−2.535479	053	−1.390855	053
5	0.000665	043	−2.238555	185	−1.234865	084
6	0.000683	156	−2.102324	084	−1.219072	009
7	0.000690	079	−2.017088	009	−1.160596	049
8	0.000723	053	−1.862865	049	−1.118919	185
9	0.000730	049	−1.859066	096	−0.994207	138
10	0.000751	096	−1.783666	138	−0.987591	096
11	0.000774	084	−0.905983	036	−0.798818	079
12	0.000798	185	−0.889191	079	−0.738117	023

#	$R(I,T)$	$\#T$	$A_{sea\_horses}(I,T)$	$\#T$	$A_{test}(I,T)$	$\#T$
13	0.000798	095	−0.791174	095	−0.638717	036
14	0.000815	034	−0.736513	098	−0.450270	098
15	0.000866	112	−0.530423	023	−0.431332	086
16	0.000914	038	−0.420436	156	−0.389948	095
17	0.000939	023	−0.373882	086	−0.198223	147
18	0.000956	035	−0.327960	147	−0.170542	029
19	0.000963	051	−0.059679	163	−0.150026	156
20	0.000976	066	0.220640	034	−0.108257	163
21	0.000986	187	0.352224	008	0.017809	008
22	0.000991	138	0.383943	043	0.102628	006
23	0.001003	033	0.585653	166	0.262717	043
24	0.001027	031	0.720087	186	0.282322	033
25	0.001075	130	0.847673	029	0.315197	186
26	0.001092	180	0.901831	033	0.351714	034
27	0.001110	072	0.959939	112	0.476177	166
28	0.001117	008	1.064986	038	0.616946	004
29	0.001125	006	1.448543	006	0.657053	112
30	0.001135	036	1.787624	165	0.850133	038
31	0.001157	014	1.981721	114	0.858438	035
32	0.001159	166	2.028339	066	0.927682	187
33	0.001162	135	2.300109	035	0.968592	031
34	0.001182	186	2.788797	130	1.000586	165
35	0.001231	091	2.935952	187	1.161564	066
36	0.001248	114	3.225982	031	1.264340	130
37	0.001263	147	3.400896	004	1.278289	180
38	0.001367	165	3.493834	180	1.329583	114
39	0.001370	183	4.368231	135	1.431522	101
40	0.001446	029	5.092729	101	1.612215	001
41	0.001549	119	5.331520	072	1.615944	135
42	0.001595	037	5.701634	121	1.805276	183
43	0.001618	009	6.015319	119	1.913921	072
44	0.001788	121	6.115742	037	1.956855	121
45	0.001815	197	6.356506	103	2.023326	091
46	0.001846	102	6.499019	183	2.064498	037
47	0.001872	103	7.200656	197	2.083365	119
48	0.001950	001	7.247634	001	2.117669	103
49	0.002397	004	7.792199	091	2.130418	197
50	0.002451	101	10.396680	102	2.308183	102

Table C.3: Full results for I as the Tree Swallow in figure 4.11.

#	$R(I,T)$	$\#T$	$A_{sea\_horses}(I,T)$	$\#T$	$A_{test}(I,T)$	$\#T$
1	0.000315	038	−3.445313	138	−2.061203	138
2	0.000410	095	−3.030115	053	−2.016731	095
3	0.000442	043	−2.971871	083	−1.792153	186
4	0.000483	156	−2.957348	051	−1.727896	053

#	$R(I,T)$	#T	$A_{sea\_horses}(I,T)$	#T	$A_{test}(I,T)$	#T
5	0.000487	112	−2.877356	095	−1.614431	096
6	0.000493	031	−2.850474	185	−1.584014	083
7	0.000500	163	−2.682474	096	−1.572137	051
8	0.000502	114	−2.342418	186	−1.497379	185
9	0.000517	035	−2.193628	038	−1.352039	038
10	0.000523	096	−2.036952	114	−1.157119	079
11	0.000527	138	−1.743499	014	−1.044561	114
12	0.000564	135	−1.551768	079	−0.985666	035
13	0.000581	053	−1.442100	156	−0.939257	031
14	0.000586	186	−1.369947	009	−0.912910	014
15	0.000587	079	−1.301589	049	−0.905756	156
16	0.000598	098	−1.121450	031	−0.894723	008
17	0.000617	091	−1.098301	008	−0.891690	009
18	0.000642	185	−1.078201	112	−0.878518	049
19	0.000679	086	−0.866234	043	−0.670993	006
20	0.000683	066	−0.704820	098	−0.650694	043
21	0.000686	072	−0.686284	036	−0.576847	112
22	0.000691	083	−0.648515	163	−0.481178	036
23	0.000702	130	−0.558229	035	−0.476925	163
24	0.000708	034	−0.359358	033	−0.429375	098
25	0.000719	187	−0.271704	084	−0.365603	033
26	0.000788	119	−0.264862	034	−0.296322	091
27	0.000788	033	−0.080811	006	−0.242145	135
28	0.000807	008	−0.021456	086	−0.184170	086
29	0.000856	006	0.049330	166	−0.158630	023
30	0.000868	049	0.073768	066	−0.038093	119
31	0.000869	183	0.088770	135	−0.014744	187
32	0.000880	051	0.137042	165	−0.013335	165
33	0.000918	037	0.711589	130	−0.005071	034
34	0.000975	180	0.724188	023	−0.000562	066
35	0.001005	165	0.953709	119	0.017003	037
36	0.001019	166	1.080820	187	0.019526	029
37	0.001024	121	1.089863	091	0.030979	121
38	0.001105	103	1.152362	147	0.123896	084
39	0.001138	197	1.248207	037	0.145803	166
40	0.001200	036	1.259613	029	0.219827	130
41	0.001203	023	1.679590	121	0.254125	072
42	0.001247	102	1.766523	103	0.265870	103
43	0.001340	084	1.843386	072	0.437297	183
44	0.001543	029	2.667693	180	0.583848	197
45	0.001737	147	2.830554	183	0.782807	102
46	0.001778	001	2.926120	197	0.784030	147
47	0.001861	009	4.480021	004	0.811106	180
48	0.002085	014	5.257147	102	1.086671	004
49	0.002652	004	6.085886	001	1.201159	001
50	0.002659	101	6.117440	101	1.892176	101

List of Figures

List of Tables

Bibliography

[AZ18] Relja Arandjelović and Andrew Zisserman. Objects that sound. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *European Conference on Computer Vision (ECCV)*, pages 451–466, Cham, 2018. Springer International Publishing. doi:10.1007/978-3-030-01246-5_27.

[BGL^{+}98] Charles H. Bennett, Péter Gács, Ming Li, Paul M. B. Vitányi, and Wojciech H. Zurek. Information Distance. *IEEE Transactions on Information Theory*, 44(4):1407–1423, July 1998. doi:10.1109/18.681318.

[BSFS15] Jimmy Lei Ba, Kevin Swersky, Sanja Fidler, and Ruslan Salakhutdinov. Predicting Deep Zero-Shot Convolutional Neural Networks Using Textual Descriptions. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4247–4255, December 2015. ISSN: 2380-7504. doi:10.1109/ICCV.2015.483.

[CV05] R. Cilibrasi and P. M. B. Vitányi. Clustering by Compression. *IEEE Transactions on Information Theory*, 51(4):1523–1545, April 2005. doi:10.1109/TIT.2005.844059.

[CV07] Rudi L. Cilibrasi and Paul M. B. Vitányi. The Google Similarity Distance. *IEEE Transactions on Knowledge and Data Engineering*, 19(3):370–383, March 2007. doi:10.1109/TKDE.2007.48.

[DCLT19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. URL: https://www.aclweb.org/anthology/N19-1423, doi:10.18653/v1/N19-1423.

[FFFP04] Li Fei-Fei, R. Fergus, and P. Perona. Learning Generative Visual Models from Few Training Examples: An Incremental Bayesian Approach Tested on 101 Object Categories. In *Workshop on Generative-Model Based Vision*, IEEE. CVPR 2004, June 2004.

[Fin05] Michael Fink. Object classification from a single example utilizing class relevance metrics. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in neural information processing systems 17*, pages 449–456. MIT Press, 2005. URL: `http://papers.nips.cc/paper/2576-object-classification-from-a-single-example-utilizing-class-relevance-metrics.pdf`.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, June 2016. ISSN: 1063-6919. `doi:10.1109/CVPR.2016.90`.

[KB14] Diederik Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations (ICLR)*, December 2014.

[Koc15] Gregory R. Koch. Siamese Neural Networks for One-Shot Image Recognition. In *International Conference on Machine Learning (ICML)*, 2015.

[Kol65] Andrei Nikolaevich Kolmogorov. Three approaches to the definition of the concept 'quantity of information. *Problemy Peredachi Informatsii*, 1:3–11, 1965.

[LBH18] Guifang Liu, Huaiqian Bao, and Baokun Han. A Stacked Autoencoder-Based Deep Neural Network for Achieving Gearbox Fault Diagnosis. *Mathematical Problems in Engineering*, 2018:5105709, July 2018. Publisher: Hindawi. `doi:10.1155/2018/5105709`.

[LEB08] Hugo Larochelle, Dumitru Erhan, and Yoshua Bengio. Zero-data learning of new tasks. In *National Conference on Artificial Intelligence*, volume 2 of *AAAI'08*, pages 646–651, Chicago, Illinois, July 2008. AAAI Press.

[LJH$^+$20] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. In *International Conference on Learning Representations (ICLR)*, April 2020. URL: `https://www.microsoft.com/en-us/research/publication/on-the-variance-of-the-adaptive-learning-rate-and-beyond/`.

[LV93] Ming Li and Paul Vitànyi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer, New York, Berlin, Heidelberg, 1993.

[Mis19] Diganta Misra. Mish: A Self Regularized Non-Monotonic Neural Activation Function. *arXiv:1908.08681 [cs, stat]*, October 2019. arXiv: 1908.08681 version: 2. URL: `http://arxiv.org/abs/1908.08681`.

[PDCT19] Gabriele Prato, Mathieu Duchesneau, Sarath Chandar, and Alain Tapp. Towards Lossless Encoding of Sentences. In *Annual Meeting of the Association for Computational Linguistics*, pages 1577–1583, Florence, Italy, July 2019. Association for Computational Linguistics. URL: `https://www.aclweb.org/anthology/P19-1153`, `doi:10.18653/v1/P19-1153`.

[PGM+19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL: `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

[RALS16] Scott Reed, Zeynep Akata, Honglak Lee, and Bernt Schiele. Learning Deep Representations of Fine-Grained Visual Descriptions. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 49–58, June 2016. URL: `https://github.com/reedscot/cvpr2016`.

[RDS+15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. `doi:10.1007/s11263-015-0816-y`.

[RHW86] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning Internal Representations by Error Propagation. In *Parallel distributed processing: explorations in the microstructure of cognition, vol. 1: foundations*, pages 318–362. MIT Press, Cambridge, MA, USA, January 1986.

[RPT15] Bernardino Romera-Paredes and Philip H. S. Torr. An embarrassingly simple approach to zero-shot learning. In *International Conference on Machine Learning*, ICML'15, pages 2152–2161, Lille, France, July 2015. JMLR.org.

[Sol64] R. J. Solomonoff. A formal theory of inductive inference. Part II. *Information and Control*, 7(2):224–254, June 1964. `doi:10.1016/S0019-9958(64)90131-7`.

[WDS+19] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *ArXiv*, abs/1910.03771, October 2019.

[WZYM19] Wei Wang, Vincent W. Zheng, Han Yu, and Chunyan Miao. A Survey of Zero-Shot Learning: Settings, Methods, and Applications. *ACM Transactions on Intelligent Systems and Technology*, 10(2):13:1–13:37, January 2019. doi:10.1145/3293318.

[XGD+17] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated Residual Transformations for Deep Neural Networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5987–5995, July 2017. ISSN: 1063-6919. doi:10.1109/CVPR.2017.634.

[XLSA19] Yongqin Xian, Christoph H. Lampert, Bernt Schiele, and Zeynep Akata. Zero-Shot Learning—A Comprehensive Evaluation of the Good, the Bad and the Ugly. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(9):2251–2265, September 2019. URL: http://cvml.ist.ac.at/AwA2/, doi:10.1109/TPAMI.2018.2857768.

[YKW+19] Chao Yang, Taehwan Kim, Ruizhe Wang, Hao Peng, and C.-C. Jay Kuo. Show, Attend, and Translate: Unsupervised Image Translation With Self-Regularization and Attention. *IEEE Transactions on Image Processing*, 28(10):4845–4856, October 2019. Conference Name: IEEE Transactions on Image Processing. doi:10.1109/TIP.2019.2914583.

[YZS+19] F.-N Yuan, L. Zhang, J.-T Shi, X. Xia, and G. Li. Theories and applications of auto-encoder neural networks: A literature survey. *Jisuanji Xuebao/Chinese Journal of Computers*, 42:203–230, January 2019. doi:10.11897/SP.J.1016.2019.00203.

[ZLHB19] Michael R. Zhang, James Lucas, Geoffrey Hinton, and Jimmy Ba. Lookahead Optimizer: k steps forward, 1 step back. *arXiv:1907.08610 [cs, stat]*, December 2019. arXiv: 1907.08610 version: 2. URL: http://arxiv.org/abs/1907.08610.