GENERATIVE BASED ZERO-SHOT LEARNING

CLASSIFYING IMAGES USING TEXTUAL DESCRIPTIONS

MIGUEL Â. SIMÕES VALENTE

Artificial Intelligence
Utrecht University

February 2022

SUPERVISORS:
Dr. Melisachew Wudage Chekol m.w.chekol@uu.nl.
Prof. Dr. Yannis Velegrakis i.velegrakis@uu.nl
Dr. Klamer Schutte klamer.schutte@tno.nl

LOCATION: Utrecht, Netherlands

STUDENT NUMBER: 6876005

TIME FRAME: 08-02-2021 - 04-2-2022

ABSTRACT

Current studies in Zero-Shot Learning for image classification use a weak Zero-Shot condition by using curated attributes as semantics to guide the classification of unseen images. Instead, this work assumes a strict Zero-Shot condition by using the readiest at hand data as guiding semantics, raw text from Wikipedia. The Zero-Shot condition itself is solved by filling in the gap of the missing visual data with generated data, essentially simulating what is missing in hopes of classifying it when it comes along. The generation is done through Zero Flow: a generative neural network architecture based on normalizing flows. Zero Flow matches state of the art in the CUB dataset and surpasses the current state of the art by 12% on ImageNet.

ACKNOWLEDGEMENTS

This work was only possible through the backing, help and support of several people.

First, I would like to thank my supervisor Mel for the level of understanding during struggles and for the consistency of his help and feedback that led to the present work.

This entire project would not be possible without the weekly meeting with my supervisor at TNO, Klamer. His advice moulded this work's direction and provided the confidence needed to achieve it. Although not through name, I would also like to thank everyone from TNO who helped me with their insightful comments or with any technical difficulties.

Lastly, I would like to thank my parents and friends for pushing me to accomplish my goals and for making life better just by being here.

Miguel Valente
Utrecht, November 2021

CONTENTS

1 INTRODUCTION

Humans are exceptionally efficient at learning. We can learn to visually recognise instances of novel objects after one *training* example. Even more impressive is our ability to visually identify a novel object without prior visual cues solely using semantic information of an object's properties/components. This ability we possess is referred to in machine learning as zero-shot learning (ZSL). ZSL first appeared in machine learning in 2008 by Larochelle et al. [17] under the slightly different name, *zero-data*. Nonetheless, *zero-data* was soon adjusted to *zero-shot* in the styles of the already established *one-shot* and *few-shot* nomenclatures.

This thesis examination of ZSL came about due to a challenge proposed by TNO (Nederlandse Organisatie voor Toegepast Natuurwetenschappelijk Onderzoek), an independent research organisation in the Netherlands focusing on applied science. The challenge arose from the project *Learning With Less Labels* (LwLL). LwLL is a project funded by DARPA (Defense Advanced Research Projects Agency), a research and development organisation from the USA. In this project, TNO aims to make machine learning more efficient by reducing the number of labels for image classification. A way of reducing the number of labels is by removing them and performing classification on examples never found at training time, also known as ZSL.

The most clear-cut example of ZSL arises from the classification of fine-grained sets of objects and the inherent difficulty of labelling all the different classes of objects in existence for machine learning purposes. For example, when dealing with expert knowledge, such as identifying species of spiders, there is a lack of visual data for each species. We can leverage each species' semantic information to counteract a lack of visual data. Then, when presented with an unknown visual example, we can infer what semantic information fits better to obtain an image classification. Semantic information is an umbrella term for some form of description of an object. The description can be written in natural language form or as a binary vector of attributes indicating their presence or lack thereof.

Deconstruction of the previous sentences allows recognition of two elements, semantic information and visual data. Solving ZSL is to solve how these two pieces relate to each other. Although it is impractical to separate the aforementioned elements in the study of ZSL, this separation is required to exclude confounding variables and to limit the scope of the work ahead. Therefore, this research explores the options available for dealing solely with semantic information.

Specifically, this thesis explores generative neural network methods using textual descriptions as the source of semantic information for ZSL. At this instant, it is hard to grasp what is meant by the overall focus of the work. Therefore, background knowledge is necessary to have a good grasp of the thesis. Thus, available options and the exact expertise needed for understanding the thesis are presented through the remainder of the text.

2

PRELIMINARIES

This chapter lays down the basic concepts and intuitions needed to understand the essential methods and concepts referred to throughout the document.

2.1 DECONSTRUCTING HUMAN ZERO-SHOT LEARNING

AI (Artificial Intelligence) research and development take a significant amount of inspiration from human beings. The original direction of AI, called AGI (Artificial General Intelligence), was to create intelligence as we perceive it. Here, the plethora of definitions of intelligence is disregarded. Instead, it is more helpful to single out the source of inspiration for AI research, us humans.

Consciously or not, most AI researchers follow this heuristic: *if we, as humans, can accomplish a task, a machine should also be able to do it* [27]. We are the baselines to beat when it comes to AI development, including in benchmarks. There is a direct connection between the origin of zero-shot learning and our ability to perform it. As the baselines for ZSL, we should understand the inner workings and dependencies needed to perform human-ZSL (HZSL). A good understanding of the ZSL literature bequests a good knowledge of what makes HZSL, contrary to the current thought of data-driven AI empirical research. A good comprehension of the most crucial elements of HZSL enables the recognition of assumptions made by studies and, by extension, what is being ignored, forgotten, or circumvented. One-to-one engineering from biology to machine learning is not needed. Instead, only an understanding of our mechanisms' building blocks. A thoughtful discussion of this kind could in itself be a research proposal if the domain was philosophy or cognitive science. To avoid an unnecessary and lengthy discussion only the most prominent and definite parts of HZSL are considered.

1. HZSL needs nomenclatures of visual concepts.
2. HZSL has an imaginative component.
3. HZSL is compositional and relational
4. HZSL is open-ended and context dependent.

The rest of this sub-chapter explores the previous list of HZSL building blocks.

Nomenclatures of Visual Concepts

A visual concept is an object or action primarily recognizable by its visual perceptual qualities and other qualities such as purpose and function, statically or across a temporal dimension with an accompanying textual label. One can still point out that an accompanying label is not intrinsic for understanding a visual concept; hence the expression "*You would have to see it to understand it*". A concept can be recognized and identified without a label; when this happens, it is usually followed by "*something that cannot be put into words*". This human capacity is beyond the scope of current machine learning, and its solution is likely to involve a better understanding of the underlying unconscious processes. Nonetheless, several visual concepts are learned during a lifetime, and the acquisition never ends. However, someone or something labels it at one point or another, and there seems to be no way around it. Labels gain even more importance in machine learning, where everything needs to be quantifiable, especially for supervised classification. Therefore, *HZSL requires nomenclatures.*

Imaginative Component

Human beings can conduct counterfactual reasoning, a capacity to reason about what could or would have been, think about hypothetical scenarios and intervene on their hypothetical contents. This capacity extends itself into conjecturing visual concepts, referred to as the imaginative component. The next thought experiment will help in consolidating this notion.

Take a moment, focus and imagine a Downy Woodpecker, a red-headed bird with black and white wings. In your mind's eye, it is possible to visualize this species of Woodpecker, now go ahead and look at Figure 1 and identify the correct image of a Downy Woodpecker, being careful not to spoil the answer immediately by reading its caption. These images were carefully selected from the CUB [34] dataset to allow for some confounding elements, but only one of the birds fits the description above.

This thought experiment aims to bring attention to the imaginative component humans possess. Descriptions or depictions of visual concepts often produce imagery to the listener; assumptions about the visual characteristics act as a preview of the real thing. The preview usually goes unnoticed, and it depends on the visualization capacities of the individual. The preview in the above experiment is masked by how strong we are at performing ZSL; we often think that we only map semantic information to what is present in our retina. Once again, this is not true since we also map semantic information

to imagined abstract visual characteristics. *Therefore, HZSL contains an imaginative component.*

Figure 1: From left to right: *Downy Woodpecker*, *Pileated Woodpecker* and *American Three Toed Woodpecker* [34]

Compositionality

The previous thought experiment demonstrates how compositionality and relationality operate in HZSL. Woodpecker, bird, wing, head, red, black, and white provided semantic information. Redhead and black & white wings are two of the relations. This portion of HZSL is relatively straightforward, and it is easy to see how HZSL becomes increasingly easy to perform with an increment of concepts and relations. An increase of concepts and relations facilitates the zeroing in on a visual concept due to increased specificity, allowing to rule out other possibilities. Another underlying puzzle is how the relations and compositions arise and what kind of relations are possible. Once again, the unknown and its exploration are unnecessary, only its acknowledgement. *Therefore, HZSL is compositional and relational.*

Unbounded

We can classify anything if we know what it is; this is to say we recognize something stored in our memory banks. A regular classification task focuses more on recognizing and classifying and less on inferring: our knowledge of the world bounds classification tasks, similar to how supervised classification is limited to the labels it is trained on. On the contrary, HZSL needs more inferential reasoning based on current knowledge; new knowledge is added through the sum of recognizable parts: hence, HZSL is also bound by what can be recognized, similar to a ZSL task. We explore a space of seemingly infinite visual concepts through the combination of different kinds of semantic information. HZSL is not bounded and is also context-dependent. For example, if the thought experiment from Figure 1 contained a second redheaded Woodpecker with black and white wings but with

a different pattern, it would be impossible to guess which of the first two was the Downy Woodpecker. *Therefore, HZSL is unbounded and context depended*.

2.2 ZERO-SHOT LEARNING

Any field of science divides itself into many branches and leaves resembling a tree. This type of taxonomy helps researchers navigate a field's literature swiftly. ZSL's taxonomy can be divided into four sections of operation: the paradigm, data restriction, methods, and semantic information source. Figure 2 contains the current taxonomy of the field of ZSL [25]. Sections highlighted in green fall under the thesis's scope. In the remainder of the text, highlights in **bold** also signal the scope of the thesis. Generally, the choice's motivation is to make the overall research as applicable to real ZSL problems as possible.

Combining elements from each section conditions what is possible by defining a research boundary. Before explaining each operation section, it is necessary to introduce some notation and basic ZSL principles. In ZSL, there are three main ingredients; semantic information, S, visual data, X, and the corresponding labels, Y. In turn, these ingredients follow a seen ^{s} and unseen ^{u} dichotomy. Methods are usually developed with S^{s}, X^{s}, and Y^{s} and tested on X^{u}, V^{u}, and Y^{u}; this separation depends on the type of data restriction. The main objective is then $f_{zsl} : X \rightarrow Y$.

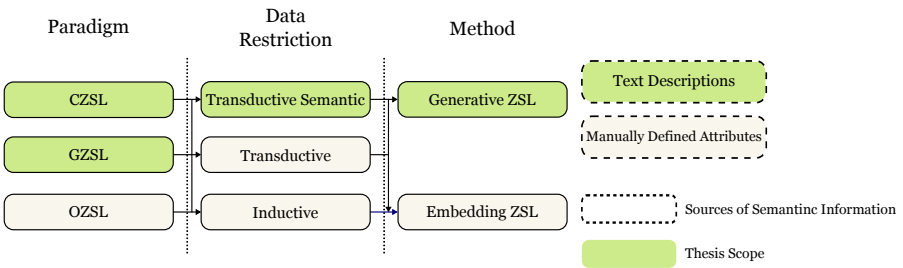


Figure 2: Diagram of ZSL taxonomy

2.2.1 *Paradigm*

The first section of operation paradigms reflects which sets of data can be expected at testing time. CZSL (classic-ZSL) was the first paradigm to show up, followed by **GZSL** (generalized-ZSL) and now recently followed by OZSL (open-ZSL). Research realized under CZSL assumes that training classes are not present during tests, $f_{czsl} : X \rightarrow Y^{u}$. **GZSL** came along to break the previous separation assumption; therefore, at test time, both train and test classes are expected, $f_{gzsl} : X \rightarrow Y^{u} \bigcup Y^{s}$. OZSL [19] breaks down the last assumption that considered ZSL a closed set classification, $f_{ozsl} : X \rightarrow Y^{u} \bigcup Y^{s} \bigcup Y^{o}$, where Y^{o} signifies every class not in $Y^{u} \bigcup Y^{s}$. Part of what makes HZSL is its

open-ended state, and in the same vein as HZSL, this last paradigm shifts to an open set. In this thesis, **GZSL** was chosen as it does not rely on unrealistic assumptions of CZSL, and possesses more maturity than the most recent more complex OZSL paradigm.

2.2.2 *Data Restriction*

Beyond the three base paradigms, three ways of using data are allowed for the experiment to be considered ZSL. The inductive approach is the most restrictive; it assumes that visual and semantic data from the training classes will not be present in subsequent tests; it uses X^s and S^s for training. **Transductive semantic** is the approach that most resembles HZSL; it permits visual and semantic data from training classes in addition to semantic information from test classes; for training, it uses X^s, S^s, and S^u. Transductive, not to be confused with transductive semantic, is the less restrictive of the three; it permits visual and semantic data from test classes and training classes, using X^s, S^s, S^u, and X^u. At first glance, it would appear that a full transductive restriction would violate the ZSL condition, but it breaks the HZSL condition instead. Studies only use unseen visual data to finetune network backbones responsible for visual feature extraction to avoid violating the ZSL condition.

Nonetheless, it breaks the HZSL condition, and it can be understood by realizing that unseen visual data is being used. The meaning of unseen changes to "*unlabelled visual data*" from "*not seen*". **Transductive semantic** is the data restriction of choice due to the need for S^u set. A requisite comes from using generative neural networks; the necessity for S^u is more apparent after explaining the methods under ZSL.

2.2.3 *ZSL Methods*

The next point in this exploration of ZSL's taxonomy is its methods, also among the primary points of this research. Two main methods of executing ZSL mark its literature. The first, most present in early work, usually involves; visual mapping features to a semantic space, from semantic space to visual or visual and semantic to a common latent space [25], referred to as embedding ZSL. The second method uses primarily generative neural networks referred to as **generative ZSL** not to be confused with generalized-ZSL aka GZSL. Generative ZSL works by training a generative model to generate X^s conditioned on the corresponding sets of S^s. After training, a generative model is primed to generate $\widetilde{X}^u$ conditioned on S^u. With both X^s and $\widetilde{X}^u$ the ZSL task becomes a supervised classification task. This choice's motivation is based on the results of GZSL techniques obtained in ZSL benchmarks plus natively incorporating the imaginative compo-

nent of HZSL. Section 3.1 contains a more thorough analysis of the current state-of-the-art techniques and neural network architectures regarding generative neural networks in ZSL.

2.2.4 *Sources of Semantic Information*

The last section of operation is the sources of semantic information, also one of the primary points of this research. Sources of semantic information are grouped into manually defined attributes and **textual descriptions**, used at different scales, image-level or class-level. Manually defined attributes used at the image-level have a Big O labelling time complexity of $\mathcal{O}(n^k)$, where n is the number of images and k the list of possible attributes for the dataset. Manually defined attributes used at the class level have a complexity of $\mathcal{O}(k \cdot c)$, where c are the number of classes. Textual descriptions have a complexity of $\mathcal{O}(c)$. The choice to use textual descriptions comes from taking a practical standpoint concerning a real-world ZSL task. A real-world problem enters a ZSL setting when for some reason, it is unfeasible to obtain the needed missing data: in the example of image classification, not being able to do supervised learning due to not having visual samples or their labels. Still, due to the principle defined in subsection 2.1, some form of labelling is required, and the one with the least amount of complexity is textual descriptions.

2.3 NORMALIZING FLOWS

Generative models are referenced in the title and the previous subchapter; Normalizing Flows (NF) are a specific type of generative models that broadly correspond to modelling a probability distribution over a random variable. Using machine learning terms the probability distribution is learned from a set of observed data, x, with a probability density, $p_x(x)$, parameterized by weights, θ. With a learned probability distribution, it is possible to use the distribution properties, $p_x(x)$, to generate data points and evaluate the density of new data points. Different setups and learning objectives lead to different types of generative models, such as the variational autoencoders (VAEs) [15] and generative adversarial networks (GANs) [11]. Each family of generative models contains one or more defining traits, which allow for their denomination.

The defining trait of NFs is the use of the change of variables formula 1, where $f(x)$ is an invertible and differentiable function, det stands for the determinant of a matrix, and $\mathbb{J}$ is the Jacobian of the partial derivatives of $f(x)$.

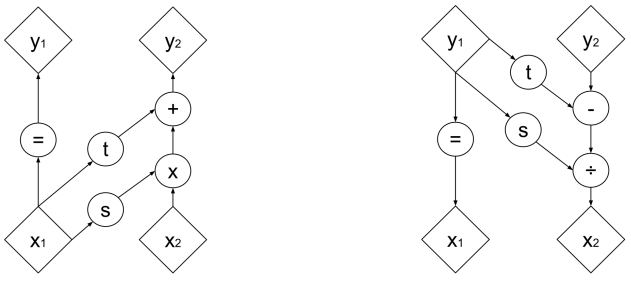
$$p_x(x) = p_z(f(x))|\det \mathbb{J} f(x)| \tag{1}$$

The formula above relates the change in the density distribution of a random variable x into the density distribution of a random variable z when x is passed through the invertible function $f(x)$. The random variable z, can follow any probabilistic distribution, the standard choice is a Gaussian, $z \sim \mathcal{N}(0, 1)$. Hence the name normalizing flow, the original arbitrarily complex distribution, $p_x(x)$ is transformed into a normal Gaussian, $p_z(z)$, the flow is the name given to the invertible and differentiable function $f(x)$. A flow needs to be dimension preserving due to the invertibility requirement. The parameters of $f(x)$, the flow, can be learned using max log-likelihood of the equation 1 leading to equation 2; θ stands for the weights that parameterize the transformation given a x.

$$\max_{\theta} \sum_{i=1}^{N} \log p_z(f(x_i|\theta)) + \sum_{k=1}^{K} \log |\det \mathbb{J} f_k(x_i|\theta)| \tag{2}$$

The sums present in Equation 2 represent the summation of flows chained together. Flows are chained together to build more complex transformations that can transform one distribution into another. The max log-likelihood uses the forward pass of a normalizing flow to learn its parameters. After learning the parameters for the normalizing flows, generating new data points is done first through sampling from the distribution $p_z(z)$ and then computing the inverse of the sampled point, $x = f^{-1}(z)$, also called the backward pass. A flow's inverse function and Jacobian determinant should also be efficiently computable for practical effects. To alleviate the previous issue the *modus operandi* of most state-of-the-art architectures is to use *coupling flows*.

Coupling flows work by splitting a vector $x = [x_1, x_2]$ in half from an initial arbitrarily complex probability distribution. The first half, x_1, is concatenated with a transformation of x_2 conditioned on x_1. This transformation can be any complex non-linear function. Figure 3 contains a diagram of affine coupling [7], a popular type of coupling flow. The transformation is affine and parameterized by $s(\cdot)$ and $t(\cdot)$, which can take the form of any random neural network.



(a) Forward propagation (b) Inverse propagation

Figure 3: Affine Coupling Block [7]

The splitting present in coupling flows makes it so that only the second half of the vector is operated on. Therefore, permutation of the vector in between chained flows is needed to ensure all data is used.

2.4 ENCODING IMAGES & TEXT

For this thesis, encoding signifies the changing of data from its original space and or format into some N-dimensional vector space. Noting that, the term **encoding** is used interchangeably with **embedding** or **vector** throughout the remainder of the document. For example, images predominantly follow the RGB colour model, having a 3-dimensional axis of arbitrary k size, $I \in R^{3\times k}$. The conventionality of the format plus the advent of agile development of deep learning python libraries, such as timm [35], makes the encoding straightforward, $I \rightarrow X^n$.

Encoding text is not as straightforward as encoding images. Language suffers from inconsistencies and edge cases, and the text itself can have different lengths and formats, such as a book, an article, an e-mail, or a tweet. Even with the advent of python libraries such as HuggingFace plug-and-play transformers [36] and sentence-transformers [26] there is still much room for assumptions and interpretations in obtaining a more holistic representation of documents or representations of documents relative to other documents.

A hindrance of current text encoding technology is the size of the textual input. For example, suppose a need for a holistic encoding of a twenty thousand words document and a transformer with an input window of one thousand words that outputs a 512-dimensional vector. The original text must be split with or without an overlapping window between splits; considering no overlapping window; we obtain 20 pieces of text individually fed to a transformer for encoding. After encoding each split, there are twenty 512-dimensional vectors. Any shared information or context is lost in the splitting process. To obtain a holistic representation of the original text the twenty vectors have to be combined. This holistic representation can be done in many ways, through averaging, summing, concatenating, or even using an MLP (multilayer perceptron). Mentions of holistic representations are a single class descriptor computed from a set of embeddings of a text derived through splitting as in the above case.

3 LITERATURE REVIEW

This section explores the relevant and surrounding literature of the following topics; generative ZSL, semantic information, normalizing flows in ZSL.

3.1 GENERATIVE ZSL

Generative models and generative neural networks (GNNs) learn probability distributions of a set of data to generate similar data. In the event of the absence of visual data, as in the case of ZSL, generative models give the means to generate the missing visual data based on the learned correlation between semantic information and corresponding visual examples. Then, the conjunction of existing visual data with generated data is used in a supervised learning fashion. When it comes to the available machine learning methods for ZSL, GNNs have been the overall leaders on benchmarks of GZSL since introduced to the task [38].

In the following paper, BMCoGAN [29], the authors use a bidirectional GANs architecture for the generation of features, Figure 4. The bidirectionality comes from the generation of visual features, $\widetilde{x}$, using semantic attributes, a, and the subsequent reconstruction of semantic attributes, $\widetilde{a}$, using $\widetilde{x}$. The data restriction is of the transductive semantic type as it usually is when generative methods are in play.

The first key element is ensuring the consistency of generated visual features using bidirectional mapping. The second key element is using a discriminator, D, to separate the generated visual features from the real ones. The second key element aims at addressing the *domain shift problem* [25]. This problem arises when mappings of unseen features are a function of seen features without any adaptation causing a bias for the classification of unseen classes as seen; the ablation study further reinforces the previous point. The removal of D makes the accuracy between unseen and seen classes imbalanced towards the seen classes.

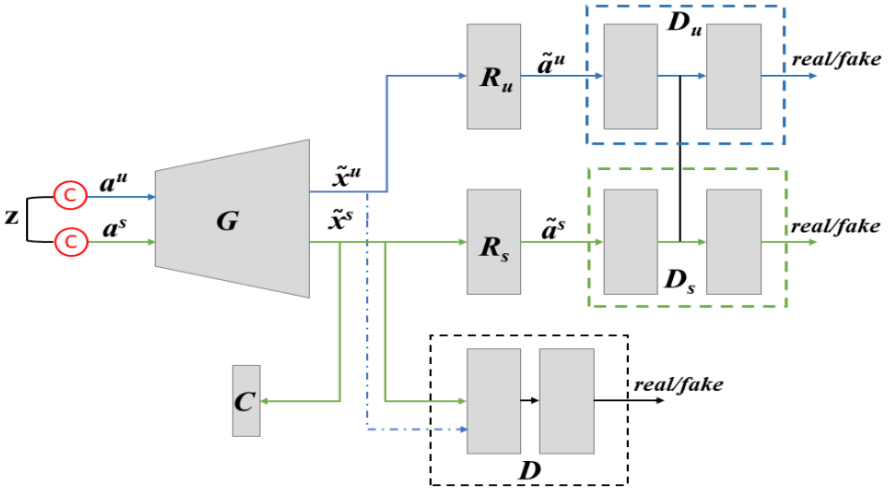


Figure 4: BMCoGAN architecture [29]

In TF-VAEGAN [21], the authors used cycle consistent VAEGAN architecture for feature generation, an architecture that combines VAEs with GANs. The data restriction can either be transductive or transductive semantic, Figure 5. The transductive setting adds another discriminator module, D [38]. The cycle-consistency is similar to the bidirectionality of BMCoGAN. The semantic embedding decoder module, *Dec*, transforms visual features into a latent vector $\widetilde{h}$. The feedback module, F, transforms $\widetilde{h}$ to $\widetilde{x}^f$ to be used by the Generator, G.

This approach's critical element is the use of *Dec* and *Feedback* modules to ensure semantic consistency in the feature generation.

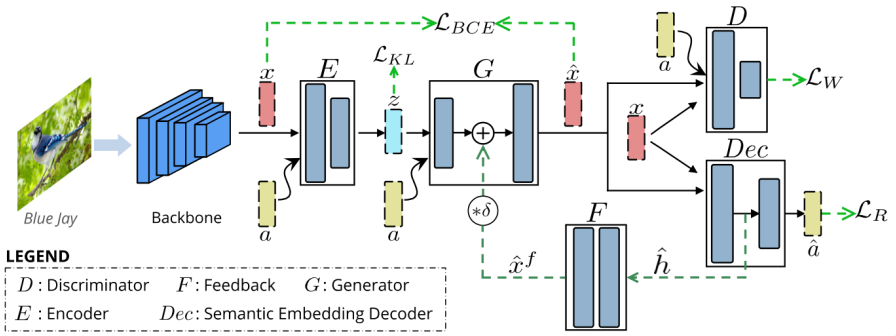


Figure 5: TF-VAEGAN architecture [21]

3.2 NORMALIZING FLOWS VS GANS & VAES

The biggest eye-catcher of Normalizing Flows is the ability to estimate exact probability density. Compared with other generative neural networks, NFs solve inherent problems of GANs (with which using ZSL regularization is complex due to the inability to perform probability density estimation) and VAEs (limited to approximate the probability density through ELBO).

The latest methods of solving ZSL using generative networks try to achieve a sort of cyclicity. The cyclicity can be found in generative models by continuous refinement of the generated visual data into generated semantic data and then into visual data. Normalizing Flows unlike GANs and VAEs, are inherently cyclic. No other networks are required to ensure cyclicity between semantic and visual information when using NFs. Achieving cyclicity with GANs and VAEs requires stacking GANs and VAEs modules. E.g., BMcoGAN requires three heads of discriminator modules to acquire cycle consistency, and TF-VAEGAN requires using a VAEGAN architecture with a complex interplay of different modules. The engineering complexity of the previous generative methods falls back to one single invertible function in Normalizing Flows.

3.3 NORMALIZING FLOWS IN ZSL

Normalizing flows based models are not well explored under ZSL; at the time of writing, there are three studies on them [28, 12, 6].

IZF [28] is the first study in ZSL that makes sole use of normalizing flows. IZF uses generative flow-based models in a transductive semantic data restriction setting. Like the former studies, one of the key elements is the cyclic approach to ensure a good generation of visual features. This cyclicity is inherent in normalizing flows; no extra module is required. From an initial visual sample v, a semantic factor $\widetilde{c}$ and non-semantic factor $\widetilde{z}^f$ are factored out through the usage of invertible functions in the *permutation-coupling block*. Figure 6 describes the architecture of IZF. Since the *permutation-coupling block* is invertible a semantic factor c and random non-semantic factor z^f can be fed to generate a $\widetilde{v}$.

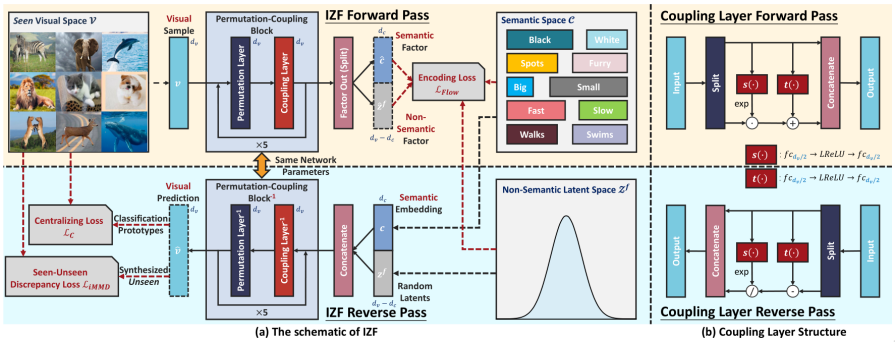


Figure 6: IZF architecture [28]

GSMF [6] is the most recent study in ZSL that uses normalizing flows. GSMF overall architecture is identical to IZF, and the main difference is in the conditioning of the visual data. IZF does so implicitly through its loss, and GSMF does it explicitly through the conditional affine coupling layers. The most relevant point is implement-

ing what the authors name relative positioning. Relative positioning works by continuously recreating the semantic embeddings during training and by adjusting their dimensions, this is further explored in Sub-chapter 4.2.6.

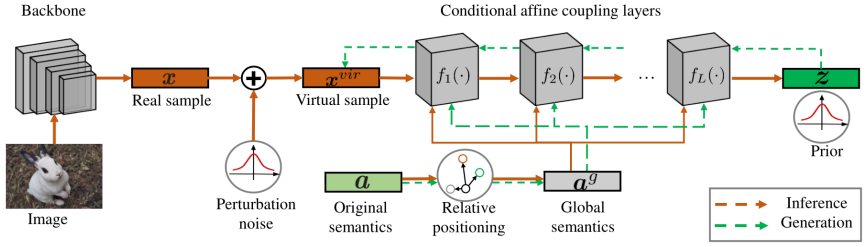


Figure 7: GSMF architecture [6] Errata: inference arrow should not be dotted

3.4 ENCODING TEXTUAL SEMANTIC INFORMATION

Sub-chapter 2.2.4 describes how the use of manual attributes does not translate to a real-world ZSL scenario. An alternative source of semantic information is class-level descriptions mined from websites such as Wikipedia. This alternative presupposes a naive assumption that the needed class information is present on Wikipedia. Notwithstanding, the assumption requiring semantic information to be present on Wikipedia holds for all the studies in the field.

The main question to address is how to obtain the class level encodings of textual descriptions. Elhoseiny et al. [8] is the first encounter of textual descriptions used in ZSL. The process of feature extraction is done firstly by extracting textual features through the use of TF-IDF and subsequently performs dimensionality reduction using CLSI [40] for noise reduction: CLSI is a method that does rank reduction on documents term-frequency matrices. The recipe of using TF-IDF for initial feature extraction followed by some form of dimensionality reduction is a common approach. It can be found in [10, 41, 5, 23, 9, 39].

Instead of using TF-IDF, Le et al. [18] uses three alternative approaches with word embeddings to get feature representation of the noisy text descriptions. The first naive approach averages all the embeddings in the definition of a word to form a description representation. The second and third approaches seek to give weights to words by giving less weight to noise and bigger weights to more informative words. Visualness, the second approach, is based on the assumption that more *visual* words have a consistent visual feature representation, e.g., stripes, green and wrinkly. The last approach uses the second approach's visualness score to learn an attention model and predict visualness from word embeddings. The word embeddings used are Word2vec [20] and Glove [24].

Sub-chapter 2.4 alludes to some of the challenges of encoding text, with transformers. Precisely what is done in the work of Bujwid *et al.* [3]. Through the use of more recent text encoding technology like ALBERT [16] text is encoded in splits with or without an overlapping window. The text in question is part of ImageNet-Wiki. This dataset contains one or more textual descriptions per class, adding to the overall issue of obtaining a holistic vector for a class. For the case of more than one text description per class, the authors experimented with summing and averaging after obtaining embeddings for each document. Besides using transformers, holistic embeddings were obtained through the sum and averaging of word embeddings from GloVe and Word2Vec. Overall this last method surpassed results obtained with transformers.

4 METHODOLOGY

This chapter outlines the methods used through the thesis and how these work.

4.1 TEXT ENCODING

Text encoding is one of the focus points of this research, to obtain a single N-dimensional embedding from a natural language piece of text through TF-IDF (Term Frequency-Inverse Document Frequency), word embeddings and transformers. The term holistic vector is used interchangeably with semantic vector or semantic embedding throughout the remainder of the chapter. Long texts have to be split into chunks to encode them. The term holistic vector helps to keep in mind that it is the result of an aggregation of vectors, from different encodes.

4.1.1 *TF-IDF*

TF-IDF is a staple tool of natural language processing. It allows finding the relevant information from a piece of text in relation to a corpus. TF-IDF makes a vocabulary list of all different words across a corpus and gives uncommon words more weight and more common words less weight on a document-by-document basis. TF-IDF works by looking at the count of a word w in a document d divided by how many documents word w appears.

TF-IDF is used on all textual descriptions of a dataset to obtain a TF-IDF vector representation of each document. Documents are concatenated in the case of a class having one or more.

4.1.2 *Word Embeddings*

Word embeddings are essentially dictionaries whose entries or keys are words, and their corresponding definition or value is a vector. Word embeddings are pre-trained on a large corpus of text to obtain a vector representation for each word. This research used GloVe [24], a word embedding algorithm that encodes a form of global context documents into its embeddings through the use of a global co-occurrence matrix calculating the probability that a given word co-occurs with others. The conversion from textual description to N-dimensional vector is as follows: dictionary search of each word for its corresponding

GloVe vector representation followed by aggregation of the obtained vectors either through sum or average.

Transformers are a revolutionary force in NLP, allowing for large unsupervised training of models through masking or next token prediction. Unlike word embeddings, transformers act on the sentence level. Their usage for encoding long texts is not as straightforward as word embeddings, largely due to the input size's limiting factor. The former limitation was already introduced and discussed in Sections 2.4 and 3.4. For this work, the transformer used was "all-mpnet-base-v2" from sentence-transformers to encode all sequences of text during experimentation.

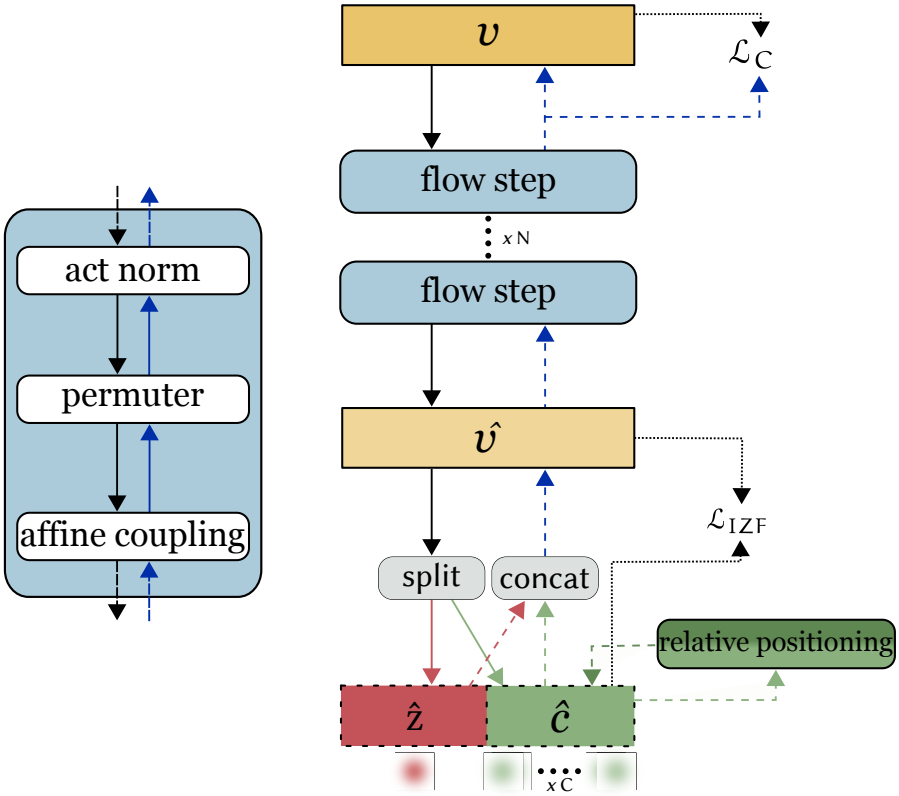
4.1.3 *Overall View of Methods*

The methods outlined in the previous sub-sections are often utilized in combination; what follows is a list of the different methods used. To summarise the encoders used are: mpnet, GloVe and TF-IDF

1. Encoding of the **sentences** of a text with **mpnet** and followed by aggregation through sum or averaging;
2. Encoding of the **words** of a text with, **mpnet** or **GloVe**, and followed by aggregation through sum or averaging;
3. Use TF-IDF on the set of all texts of a dataset to find the words of each document with the highest value followed by the encoding of the top n words with **mpnet** multiplied by their corresponding TF-IDF weight; aggregation is done through summation or averaging;
4. Concatenates the top n words according to TF-IDF and encode it with **mpnet**;
5. Use the TF-IDF raw vector or a concatenated version of it.

4.2 MODEL ARCHITECTURE

Figure 8 contains the structure of Zero Flow, the model architecture used to conduct generative ZSL. Sub-chapters 2.2 and 3.1 explain how to go from a ZSL setting to a supervised one using generative models. Zero Flow is a novel generative architecture that combines elements from IZF [28], GSMF [6] and Glow [14], two losses (IZF and centralizing), relative positioning and act norm plus 1×1 convolutions respectively.



a) Flow Step b) Zero Flow Architecture

Figure 8: Model architecture: forward pass (⟶), backward pass (⇢), multivariate Gaussian ( ), embeddings ( ) and functions ( )

4.2.1 *Act Norm*

Act Norm's [14] (activation normalisation) functions similarly to Batch Norm [13] by normalising features in between layers of a network, a type of data-dependent initialisation, in this case, present before permuting. Act Norms acts on the data through an affine transformation, scaling and translating. The first input of an Act Norm layer initialises its weights and ensures a post Act Norm output of zero mean and unit variance. Post initialisation Act Norm becomes data agnostic, and its scaling and translating parameters are learnable.

4.2.2 *Affine Coupling*

Affine Coupling [7] is a type of coupling flow, which explicitly uses affine transformations, scale and translation. Affine Coupling original paper used the exponential function for scaling. This scaling led to instability while training, so it was replaced by a sigmoid function whose output was bounden between $[0.5, 1]$ according to Behrmann

et al. [1], which improves stability and decreases exploding gradients: a sudden increase in value that causes *Nan* and *Inf*.

4.2.3 *Permuter*

Sub-chapter 2.3 gives an introduction to NFs, stating the problem and the reason why only half of a vector is used for transformations at a time. Therefore, permutation of the dimensions of a vector in between flows layers is needed. In the break-through paper for NFs [7] permutations are fixed, a type of inductive bias which was removed by introducing 1×1 convolutions [14]: a generalization of permutations. 1×1 convolutions are learnable parameters, initialized as a random rotation matrix. Instead of having fixed permutations, the permutations are learned during the training process to improve training.

4.2.4 *IZF Loss*

Zero Flow learns the distribution of a set of visual embeddings with the forward pass by training on seen classes and their respective class semantic embeddings through the IZF loss equation 3. The explicit loss defined in IZF [28] separates semantic information and non-semantic information from a visual embedding v. After v goes through N flow steps, $\hat{v}$ is obtained and split in half, a semantic half and a non-semantic half. A standard multivariate Gaussian represents the non-semantic half. The semantic half is represented by C multivariate Gaussians, where C corresponds to the number of classes of the used dataset. Each semantic Gaussian is centred at the semantic vector of each corresponding class and has a standard variance.

Equation 3 describes the IZF loss, given an embedding v the probability of it being classified as y. The former probability is given by the summation of the semantic term, $\log p_{c|y}(\hat{c} \mid y)$, with the non-semantic term $p_z(\hat{z}^f)$. Both terms are then summed by the accumulated sum of the determinant of the Jacobian of partial derivatives f given a v across N chain of flows. $\hat{z}$ and $\hat{c}$ are the vectors obtained from the splitting of the output embedding $\hat{v}$.

$$\mathcal{L}_{IZF} = \log p_{c|y}(\hat{c} \mid y) + \log p_z(\hat{z}^f) + \sum_{i=1}^{N} \log |\det \mathbb{J} f(v)| \tag{3}$$

For the splitting of $\hat{v}$ into non-semantic and semantic the following needs to be respected; for a set of semantic vectors with dimensions $c \in R^{C \times |x|}$ and a set visual features with dimensions $v \in R^{V \times |y|}$, $|x| < |y|$.

4.2.5 *Centralising Loss*

Centralising Loss works with the backwards pass, using generated samples in its equation. Equation 4 describes centralising loss, where c^s represents the semantic vector of each seen class, ^{s}, while $\overline{v}^s$ describes the mean vector of class ^{s}.

$$\mathcal{L}_C = \|f^{-1}([0, c^s]) - \overline{v}^s\|^2 \tag{4}$$

Generation works by sampling from $p_z(z)$ obtaining a vector, $\hat{z}$, sampling from $p_c(c \mid y)$ obtaining $\hat{c}$ followed by concatenating both vectors $\hat{v} = [\hat{z}, \hat{c}]$ and then performing the backward pass to obtain a generated $\tilde{v}$. In the case of centralizing loss the first half of $\hat{v}$ is replaced by zeros. The generated results from the concatenation of the semantic vectors of all seen classes with the non-semantic side left as zeros are subtracted from the mean visual vector of each class. The centralizing loss guides the output of the transformations to converge into the mean of the target semantic Gaussians.

4.2.6 *Relative Positioning*

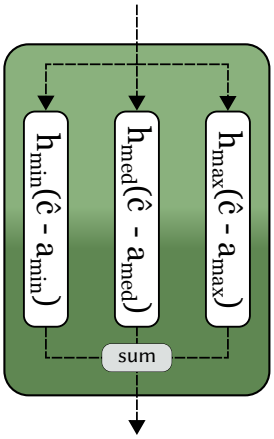


Figure 9: Relative Positioning

Relative Positioning acts on the set of semantic vectors of a dataset; it redimensions and readjusts the set of semantic vectors before and during training. Relative Positioning starts before training, calculating a similarity matrix of the set of semantic vectors. Three semantic anchors are calculated a_{min} a_{med} a_{max}, corresponding to the lowest, median and highest sum similarities to all other vectors. Relative Positioning is then initiated with three linear layers that correspond to h_{min}, h_{med} and h_{max}, with the corresponding offset that subtracts from input as depicted in Figure 9. The output of all layers is summed and used as a new semantic vector. Relative Positioning is part of the training, and semantic vectors are continuously adjusted. Changing

the dimension output of each linear layer makes it possible to adjust the dimensions to semantic vectors to any desired value.

5 EVALUATION

This chapter displays the results of employing the different methods detailed in the previous Chapter 4. The results are averaged across five random runs for a fair comparison, with the highest result of each table bolded and underlined.

5.1 DATASETS & BENCHMARKS

Currently there are three datasets available for ZSL that use text descriptions; CUB [34], NAB [30] and more recently ImageNet-Wiki [3]. This work only used CUB and ImageNet-Wiki due to their being more they are more distant in content than the NAB dataset. Images of objects with corresponding semantic information and labels comprise these datasets. CUB has one text description per class, composed of 200 classes of bird species and their corresponding Wikipedia textual description constituting a total of 11,788 images. ImageNet-Wiki has one or more text descriptions per class. The length of each text description is variable. The recent addition of ImageNet-Wiki to ZSL benchmarks solidifies text as an alternative to manual attributes. ImageNet-Wiki extends the ImageNet dataset where textual descriptions were added to its classes. Overall, the former dataset has 1.3 million images; manually labelling the attributes at the image level would be extremely expensive. The ImageNet-Wiki dataset creators also point out the need for expert knowledge depending on the data to be annotated, another disadvantage of using manual attributes.

5.1.1 *Data Splits*

Data splitting is a conventional procedure in machine learning research to evaluate the performance of models. Data splitting in ZSL serves the same evaluation purpose but adds another dimension if the paradigm is CZSL; the way data is split produces harder or easier sets for classification. The task becomes more challenging when less semantic information is shared between train and test datasets. In the example of the CUB dataset, if different species of albatross are present in both the train and test sets, then its classification becomes easier, a Super-Category-Shared (SCS) [10]. Having all different species of an albatross only present in the test split, it becomes a Super-Category-Exclusive (SCE). The former denominations are dependent on the super categories of a dataset; in CUB, the Super-Category is the *genus*.

The splits used for CUB are: for CZSL the hard-split [10](SCE) with 160 classes for training and 40 for testing, the easy-split [10](SCS) with 150 classes for training and 50 classes for testing and the more recent proposed split for GZSL [37] with 150 training and 50 test classeshere [4]. For ImageNet the split used was the mp500 [3] with 1000 classes for training and validation and 500 classes for measuring performance.

5.1.2 *Image Encoding*

Image encoding, as defined on Section 2.4, was done through the usage of timm [35] and solely for the CUB dataset due to time constraints in encoding ImageNet's dataset. The specific model from the available selection was *resenet50d*, the transforms used were; Resize(256) and CenterCrop(224,224).

5.2 RESULTS

This section contains the results covering CUB and ImageNet, the significance of the results obtained are at the end of this section separated by dataset.

5.2.1 *CUB2011 + WikiDescriptions*

There are four sources of data used to evaluate Zero Flow on the "*CUB2011 + WikiDescriptions*" benchmark, two for the visual features and two for the text features.

The first source for visual features is the CIZSL [9] study data which consists of a concatenation of seven 512D vectors extracted from the fully connected layer of a part-based convolutional neural network (1.head, 2.back, 3.belly, 4.breast, 5.leg, 6.wing and 7.tail), leading to a full vector representation of 3684D for each image. The second visual feature source is the outputs extracted from the last pooling layer of the resnet50d model found on timm [35] library.

The first source of textual features is from the CIZSL study, which consists of a 7551D vector representation for each CUB class obtained from running TF-IDF Porter stemmed WikiDescriptions. The second source corresponds to the text features extracted from WikiDescriptions with the methods delineated in Section 4.

The evaluation is initiated by fixing the parameters of Zero Flow and its training procedure, the base parameters for the initial trials are found here [31]. The evaluation starts with fixed parameters using the resnet50d visual features to evaluate the several text encoding methods across the easy split (Tables 1 ,3) and hard split (Tables 5 ,7). To assure a fair validation of the text methods the visual features of CIZSL are also used: easy split (Tables 2, 4) and hard split (Tables 6, 8).

Preprocessing, signalled by *, is done by lower-casing all text and removing all stopwords.

5.2.1.1 *Easy ZSL Split*

All following tables contain the Unseen Accuracy averaged across five random trials, random seeds, with the standard deviation in parenthesis. Table 1 and Table 2 diverge only in the visual encode used and share the text encoding; the same applies to Table 5 and Table 6. Summing was only done over a small number of vectors, namely the top 1024 sorted by TF-IDF weights; summing over a large number of vectors led to numerical issues and abysmal results. The "k" next to a TF-IDF result indicates the limitation of vocabulary respective to the top "k" results, e.g., 5k stands for top 5000 TF-IDF terms. Sentence 1024 weighed is similar to sentence 1024; the difference lies in the former being weighted by its corresponding TF-IDF value.

Text Encoding	Sum	Average
GloVe *	–	29.1 (0.6)
GloVe	–	24.0 (0.4)
Sentence *	–	32.1 (0.5)
Sentence	–	**34.0** (0.4)
Sentence 1024*	18.1 (1.5)	18.5 (1.0)
Sentence 1024	14.3 (1.0)	9.1 (0.7)

Table 1: Visual Features: resnet50d 2048D MinMax Normalized

Text Encoding	Sum	Average
GloVe*	–	29.1 (0.4)
GloVe	–	24.3 (0.9)
Sentence *	–	**33.2** (0.3)
Sentence	–	33.2 (0.7)
Sentence 1024*	18.6 (1.4)	18.9 (1.4)
Sentence 1024	14.5 (0.8)	9.7 (0.6)

Table 2: Visual Features: CIZSL 3583D Zero-Mean Standard Unit Normalized

Text Encoding	Pre-processed	Not Pre-processed
Sentence 1024 Weigthed	5.8 (0.7)	5.3 (0.5)
TF-IDF 5k	38.0 (0.6)	38.0 (0.2)
TF-IDF 7k	37.5 (0.3)	37.4 (0.5)
TF-IDF 8k	**38.1** (0.3)	37.8 (0.3)
TF-IDF Full	37.6 (0.4)	37.6 (0.6)

Table 3: Visual Features: resnet50d 2048D MinMax Normalized

Text Encoding	Pre-processed	Not Pre-processed
Sentence 1024 Weigthed	5.8 (0.5)	4.5 (0.5)
TF-IDF 5k	38.0 (0.3)	37.4 (0.3)
TF-IDF 7k	38.2 (0.3)	37.4 (0.1)
TF-IDF 8k	38.1 (0.1)	37.2 (0.4)
TF-IDF Full	**38.9** (0.2)	37.1 (0.2)

Table 4: Visual Features: 3583D Zero Mean Standard Unit Normalized

5.2.1.2 *Hard ZSL Split*

This CUB split follow the standards delineated in Sub-Section 5.2.1.1 but done with the hard-split version of the CUB dataset.

Text Encoding	Sum	Average
GloVe*	–	**9.1** (0.6)
GloVe	–	9.0 (0.3)
Sentence *	–	8.0 (2.3)
Sentence	–	7.3 (0.6)
Sentence 1024*	6.6 (0.2)	6.3 (0.8)
Sentence 1024	6.2 (0.6)	5.5 (0.4)

Table 5: Visual Features: resnet50d 2048D MinMax Normalized

Text Encoding	Sum	Average
GloVe*	–	**9.3** (0.3)
GloVe	–	9.0 (0.4)
Sentence *	–	7.7 (0.2)
Sentence	–	7.5 (0.5)
Sentence 1024*	7.7 (0.8)	6.6 (0.9)
Sentence 1024	6.0 (0.4)	5.7 (0.7)

Table 6: Visual Features: CZISL 3582D Zero Mean Standard Unit Normalized

Text Encoding	Pre-processed	Not Pre-processed
Sentence 1024 Weigthed	5.3 (0.6)	5.6 (0.3)
TF-IDF 5k	**11.3** (0.2)	10.9 (0.2)
TF-IDF 7k	11.2 (0.4)	10.7(0.1)
TF-IDF 8k	11.0(0.3)	10.8 (0.2)
TF-IDF	10.8 (0.1)	10.9 (0.3)

Table 7: Visual Features: resnet50d 2048D MinMax Normalized

Text Encoding	Pre-processed	Not Pre-processed
Sentence 1024 Weigthed	5.5 (0.3)	5.4 (0.6)
TF-IDF 5k	11.6 (0.6)	11.3 (0.2)
TF-IDF 7k	11.1 (0.2)	**11.8** (0.6)
TF-IDF 8k	11.2 (0.1)	11.2 (0.3)
TF-IDF	11.4 (0.3)	11.5 (0.6)

Table 8: Visual Features: 3583D Zero Mean Standard Unit Normalized

5.2.1.3 *Optimizing Parameters*

After obtaining initial results, a hyper-parameter search is carried out on Zero Flow architecture and its training parameters. The hyper-parameter search is Bayesian in nature and done with the use of Weigths&Bias [2] with a total of 55 random seed runs.

The parameter searched are the following: the number of hidden layers of the affine coupling network, number of samples for training a classifier, number of affine coupling blocks, and the weight decay value of the optimizer. The parameters that carry the most significant importance are organized from the top (most important) to bottom (least significant) on Figure 10, and the overall results displayed in Figure 11: each line represents a training instance which is colour-coded according to the last column. These results and diagrams are available online and are free to consult and interact with here [32].

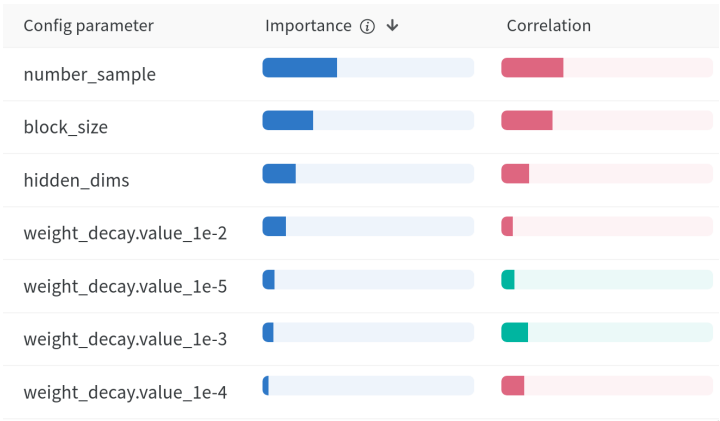


Figure 10: Hyper-Parameters correlation with unseen accuracy; red implies a negative correlation, green implies a positive correlation

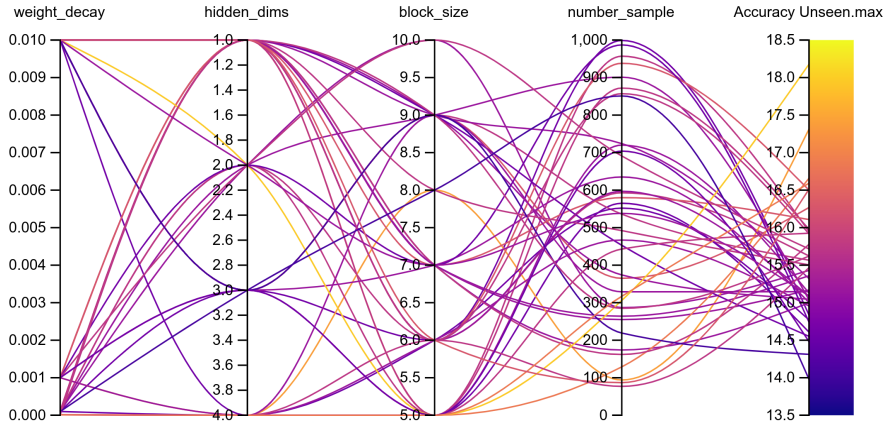


Figure 11: Results of hyper-parameters search CUB

After the search the parameters were fixed such as: *"number_samples"* was fixed at 300, the *block_size*, was fixed at 5, the number of *"hidden_dims"* was fixed at two, and the *"weight_decay"* fixed at 1e-3.

Before the search the parameters were fixed such as: *"number_samples"* was fixed at 2000, the *block_size*, was fixed at 5, the number of *"hidden_dims"* was fixed at two, and the *"weight_decay"* fixed at 1e-5.

Table 9, contains the final CUB results on both the hard and on the easy split, using CIZSL data and the text encodings delineated in earlier sections. Zero Mean Standard Unit was the normalization used on this experiment to remove the confounding element of using MinMax and Zero Mean Standard Unit normalization from previous experiments Table 2-8.

Methods	Visual Features	Text Features	Hard Split	Easy Split
Zero Flow	resnet50d	CIZSL	11.4 (0.5)	37.8 (0.2)
	resnet50d	TF-IDF 5k	10.4 (0.2)	39.9 (0.4)
	CIZSL	TF-IDF 5k	15.7 (0.4)	45.0 (0.4)
	CIZSL	CIZSL	15.7 (0.7)	**47.2** (**0.2**)
State of The Art Methods				
GAZSL + CIZSL-v1	CIZSL	CIZSL	14.4	46.6
CIZSL-v2+SeGC	CIZSL	CIZSL	**16.4**	42.4

Table 9: Increased Unseen Accuracy results after hyper-parameter search. TF-IDF 5k was solely used due to it being a smaller vector and because the performances between TF-IDF sizes was not significant. Results also show state-of-the-art is beaten on the easy split and in range of the standard variation of the hard split

5.2.1.4 *Proposed Split*

The trials with the proposed split were run with the resnet50d visual encodings and the TF-IDF 5k text encodings with the values of some hyper-parameters decided based on the search. There are no

other methods to the best of my knowledge making use of the GZSL proposed split on CUB.

	CUB		
Method (Encodings)	$\mathbf{A_S}$	$\mathbf{A_U}$	**H**
Zero Flow (resnet50d+ TF-IDF 5k)	16.1 (0.3)	16.8 (0.7)	16.6 (0.1)

Table 10: $\mathbf{A_S}$ stands for seen accuracy, $\mathbf{A_U}$ stands for unseen accuracy and **H** stands for the harmonic mean of $\mathbf{A_S}$ and $\mathbf{A_U}$

5.2.2 *ImageNet + ImageWiki*

The evaluation of the benchmark of *ImageNet + ImageWiki* is done in a similar fashion to CUB2011. There are two sources of text encodings and one source of visual encodings. The first source of text encodings comes from the authors of the ImageWiki dataset [3], and the second source from the text encoding methods delineated on Sub-Section 4.1. The source of visual embeddings for this benchmark comes from Xian *et al.*[37]. This evaluation starts with several text encodings and ends by doing a hyper-parameter search. The evaluation is only carried out with the ZSL paradigm.

5.2.2.1 *Proposed Split*

The proposed split experiments were run entirely with the text methods defined on Sub-chapter 4.1 except for ALBERT pre-encoded weights provided by the authors of ImageWiki, the first result on Table 11.

SOTA	**ALBERT**	**TF-IDF Full**	**TF-IDF 20k**	**Sentence Mean**	**Glove**
22.27	17.89	32.52 / 28.46	**34.15** / 30.08	21.95 / 11.38	19.51 / 7.31

Table 11: Results before hyper-parameter search on mp500 split. The left side of the "/" corresponds to pre-processed text in cells containing two values. The value under **SOTA** indicates the current state of the art result on the mp500 ImageNet+ImageWiki split

This table show that to the best of my knowledge the state-of-the-art was beaten by 12.88% [3]. The previous state-of-the-art was obtained using a simpler generative neural network when compared with Zero Flow, which could explain the disparity in values.

5.2.2.2 *Optimizing Parameters*

After obtaining initial results, a hyper-parameter search on Zero Flow architecture and some of its training parameters is carried out. The hyper-parameter search is done using Weights&Bias [2], in the same

fashion as Sub-chapter 5.2.1.3, therefore a Bayesian search with a total of 42 random seed runs.

The parameters searched are the following: the batch size of Zero Flow, number of hidden layers of the affine coupling network, number of samples generated for training a classifier and the number of affine coupling blocks. The parameters that carry the biggest importance are organized from top (most important) to bottom (least important) on Figure 12 and the overall results can be seen on Figure 13. These results and diagrams are available online and are free to consult and interact with here [33].

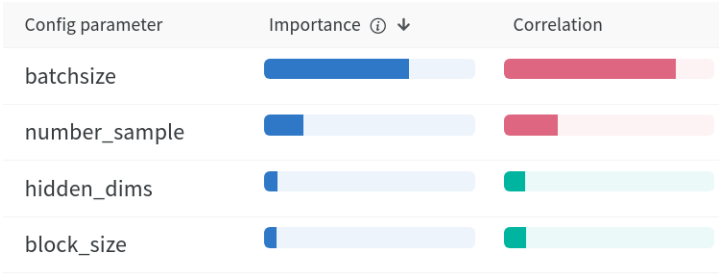


Figure 12: Hyper-Parameters correlation with unseen accuracy; red implies a negative correlation, green implies a positive correlation

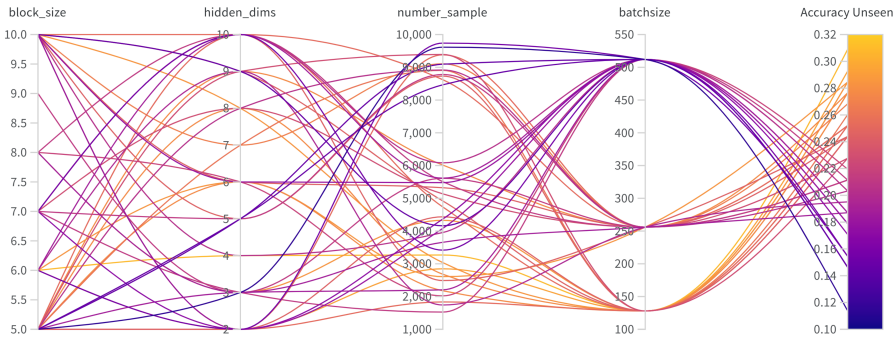


Figure 13: Results of hyper-parameters search ImageNet

There were no other experiments carried out after running running hyper-parameter optimization which suggests better accuracies are available.

5.2.3 *Qualitative Assement*

There are two prevalent generalisations across all datasets and splits; pre-processing overall indicates a better accuracy performance, this effect is more pronounced in Table 11 and less pronounced on Tables 5-8. TF-IDF outperforms all text encoding methods across the board regardless off the size of vocabulary restriction. A discussion of why this happens is present on Section 6.1.

5.2.3.1 *CUB2011*

On the easy split, TF-IDF performs the best, GloVe coming in second, followed by sentence embedding of the top 1024, and lastly, the sentence embedding weighted by TF-IDF corresponding weights. GloVe outperforms sentence transformers on the hard split, with the remaining text encoders following the same hierarchy as in the easy split. There are no other comparison studies on the proposed split using text descriptions.

The hyper-parameter analysis starts by looking at Figure 10, the parameter that correlates the most with Unseen Accuracy is the *"number_samples"*, the number of samples generated per class to train the classifier, the negative correlation indicates that a higher number of samples sizes leads to worse results. Figure 11 confirms the claim where yellow lines are predominantly present on a lower number of samples. The *block_size*, the number of flow steps is the second most important parameter, and the visual analysis of Figure 11 indicated that five number of blocks is the average optimum choice. Regarding *"hidden_dims"*, the number of hidden dimensions in an affine coupling network seem to best at two or four layers deep; the number of layers was fixed at two to minimize memory constraints.

Table 9, contains the final results of CUB on both hard and easy split after hyper-optimization. This table helps exclude the normalization technique of visual features as a confounding variable from the previous trials. It also shows permutations of the sources of visual and textual encodings. The worst performance happens with the combinations of encodes present in the first row. Using CIZSL visual features, there is a 4% increase in accuracy as seen on the fifth row; possibly due to CIZSL visual features higher dimensionality allowing better separation of semantic and non-semantic by ZeroFlow. The difference between CIZSL text features and TF-IDF 5k is less significant. Using CIZSL text data produced state-of-the-art results in the easy split. The hard split does not reach or surpass state-of-the-art due to it being analysed for five runs, an individual run did reach state-of-the-art as can be seen by analysing the standard deviation of 0.7. To the best of my knowledge, there are no studies on text encodings on the proposed split; the results obtained serve as a new benchmark.

5.2.3.2 *ImageNet*

The results of ImageNet, Table 11, reveal the importance of pre-processing text, most noticeable when using GloVe, showing a 12% increase, plausibly due to the aggregation working at the word level. Sentence mean has an 8% increase, not as significant, plausibly because this holistic encoding works at the sentence level. The baseline of Sentence is higher, which could be attributed to transformers more robustness to noise being pre-trained with natural, unprocessed language. AL-

BERT encoded the unprocessed text with an overlapping window on split sentences. Although the overlapping window may be helpful to ensure more shared information between encodings, it could also amplify the noise, leading to worse performance.

The hyper-parameter analysis starts by looking at Figure 12, the parameter with the biggest importance is *"batchsize"*; it correlates negatively with Unseen Accuracy. The negative correlation implied that bigger batch sizes lead to worse accuracy, visually confirmed by looking at Figure 13. The best value for batch size for ImageNet appears to be 128.

Similarly to CUB the second most important parameter is the number of samples generated for training the classifier; a larger amount of samples leads to lower unseen accuracy; the best value seems to reside between 3000 and 4000.

Both *hidden_dims* and *"block_size"* are harder to understand as their importance is being overpowered by the batch size. Still, the positive correlation indicates that a bigger number on both parameters is beneficial to reach higher accuracies.

In the end, the state-of-the-art was achieved surpassing the latest method by 12%.

6 DISCUSSION

This chapter discusses salient points that have been hinted at during the previous chapters.

6.1 PEERING INTO TF-IDF

TF-IDF outperforms all other encoding methods, which is surprising considering the current state of NLP, where the cycles of breakthroughs are getting increasingly smaller due to the advent of deep learning. Looking into some of the TF-IDF vectors gives a clue as to why their performance is so good.

Looking at the pre-processed TF-IDF 5k top 10 terms of the first three class vectors from CUB we see:

- **blackfooted albatross**: [albatross, blackfooted, islands, chick, convertkglbabbron, atoll, laysan, floating, pair, fishing]
- **laysan albatross**: [albatross, laysan, immutabilis, converttokglbabbron, underwing, hawaiian, mantle, rump, islands, dark]
- **sooty albatross**: [albatross, sooty, darkmantled, procellariiformes, nasal, sides,ocean, subantarctic, sulcus, shading]

Looking at the pre-processed ImageWiki top 10 terms of three chosen at random class vectors from ImageNet we see:

- **mountain tent**: [tents, tent, poles, ft, guy, pole, ropes, bullet, groundsheet, camping, 'fabric]
- **guacamole**: [avocado, guacamole, sauce, avocados, dip, mexican, de, dishes, mexico, pronounced]
- **jellyfish**: [jellyfish, polyp, tentacles, bell, species, medusae, medusa, fish, gfp, blooms]

The TF-IDF vectors obtained are constituted by continuous values that indicate the presence of a term; in our case, they resemble a type of weighted vector of binary values that correspond to that class's names, as clearly demonstrated by Figure 14. It appears that TF-IDF vectors give more weight to terms that are close to the class name or are semantic relevant. In the specific case of CUB, the top terms end up being the actual class name, likely due to the scientific tendency of the nature of Wikipedia text. It also explains the 30% drop-in unseen accuracy from the easy split to the hard split; the super-category, the

genus, such as albatross, is no longer shared between seen and unseen sets, making it harder to find relations.

However, quality depends on the source, such as a standardized form of description like Wikipedia, which TF-IDF can capture.

	albatros	laysan	... x 5k ...			sooty	blackfooted
sooty albatros	0.70	0				0.30	0
laysan albatros	0.65	0.47				0	0.07
blackfooted albatros	0.54	0.11				0	0.34

Figure 14: TF-IDF Values of first two terms from first three CUB classes; on top terms, on the left class names

6.2 CLASS LEVEL VS IMAGE LEVEL SEMANTIC REPRESENTATIONS

Sub-chapter 2.2.4 contextualized the motivation behind the use of textual descriptions but did not present an explicit discussion on the advantages and disadvantages of both sources of semantic information.

The first advantage of text descriptions is the dramatic decrease of labelling complexity illustrated by Figure 15.

Figure 15: Spectrum of semantic information labelling complexity

The scrapping of largely unsupervised descriptions discards the need for expert knowledge to define manual attributes. The spectrum is colour coded to imply that image level attributes are worse than class level, but one could quickly invert the colours. The spectrum's dichotomy depends on the observer; manual-level attributes also possess many advantages over text descriptions. Text descriptions are inherently more challenging to use than manual labels since they act at the class level, plus a loss in discrimination power is exhibited by the contrasts made in Table 12. Using semantic features at the image level is easier than using discriminatory features at the class level, mainly due to the granularity of discrimination gained by acting at the image level.

The perspective of an observer decision is marked by the rigidity of the ZSL condition of a real-world problem. The rigidity itself

Text Descriptions	Image Level Attributes
One semantic vector per class	One semantic vector per image
No access to components	Access to components
No intra-class variation	Intra-class variation
Lower inter-class variation	Inter-class variation
Noisy	No noise

Table 12: Disadvantages of text descriptions over image level attributes

is dictated by the supply of resources available to obtain semantic representations to reach a goal (*e.g.* image classification accuracy), in other words, the feasibility of obtaining the missing data. Assuming a hard limit on the sourcing of semantic information, meaning a setting where it is complicated to gather missing data, one has to turn to the least complex form of gathering, which for now stands as class-based semantics.

6.3 NORMALIZING FLOWS AND ZERO-SHOT LEARNING

Zero Flow started by borrowing the structural details of IZF, which after being coded, was not stable. Still, after a lot of work following the IZF IZF's manuscript, which lacked essential details, it was impossible to approximate the results they claimed with their toy dataset example. Most of the add-ons and borrowed elements from other Normalizing Flow's papers helped stabilize the network and made it usable for ZSL, starting by obtaining the same results as IZF on its toy dataset. The changes that had the most significance in stabilizing Normalizing Flows from most consequential to least are: changing the scaling function from exponential to bounded sigmoid, 1x1 convolutions, relative positioning, and act norm.

Normalizing Flows are still under-researched compared to GANs and VAEs, but the work carried out in these theses proves they match the state-of-the-art research on the CUB dataset with highly engineered and optimized solutions such as the work of Elhoseiny et al. [9] who's been conducting work with generative ZSL on the CUB dataset since 2017 [10].

7 CONCLUSION & FUTURE WORK

7.1 CONCLUSION

On the CUB dataset, Zero Flow matches the current state of the art in the hard split, off by 0.7% and achieves state-of-the-art on the easy split an increase of 0.6%, respectively found on sub-sub-sections 5.2.1.2 and 5.2.1.1. On ImageNet, the state of the art was improved by 12%, result is present on sub-sub-section 5.2.2.1. These results are were obtained with two distinct types of datasets and Zero Flow should generalize as a method to other similar uses cases. The best performing text encoding method was TF-IDF with very discrete vector representations of the classes. Overall Normalizing Flows work for generative purposes and therefore for generative ZSL, but they require a lot of work as they are still a young topic in terms of total research.

7.2 FUTURE WORK

The current methods of averaging and summing separate vectors to obtain a holistic representation are crude and should be improved. Future work should focus on developing better class-level semantic representations to deal with the hard ZSL setting straight on instead of optimizing generative architectures and their losses with diminishing returns—a prime example was the work needed to make Zero Flow stable. Following the idea of a least resistant path, the focus should also shift to unsupervised methods such as the massive-scale pairing of text and image recently made by OpenAI's CLIP [22].

Solving ZSL is to unravel how semantic information relates to visual stimuli. Language is the only clue we have to access meaning, and therefore we derive that it must contain what we see. In machine learning terms, we assume text acts as an encoding of visual fields and learned concepts. Future work should also look into the possibilities for developing semantic sources free of text.

BIBLIOGRAPHY

[1] Jens Behrmann, Paul Vicol, Kuan-Chieh Wang, Roger Grosse, and Jörn-Henrik Jacobsen. "Understanding and mitigating exploding inverses in invertible neural networks." In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2021, pp. 1792–1800.

[2] Lukas Biewald. *Experiment Tracking with Weights and Biases*. Software available from wandb.com. 2020. URL: https://www.wandb.com/.

[3] Sebastian Bujwid and Josephine Sullivan. "Large-Scale Zero-Shot Image Classification from Rich and Diverse Textual Descriptions." In: *arXiv preprint arXiv:2103.09669* (2021).

[4] *CUB Splits*. en. URL: https://github.com/miguelvalente/CUB_WIKI (visited on 02/02/2022).

[5] Zhi Chen, Jingjing Li, Yadan Luo, Zi Huang, and Yang Yang. "Canzsl: Cycle-consistent adversarial networks for zero-shot learning from natural language." In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2020, pp. 874–883.

[6] Zhi Chen, Yadan Luo, Sen Wang, Ruihong Qiu, Jingjing Li, and Zi Huang. "Mitigating Generation Shifts for Generalized Zero-Shot Learning." In: *arXiv preprint arXiv:2107.03163* (2021).

[7] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. "Density estimation using real nvp." In: *arXiv preprint arXiv:1605.08803* (2016).

[8] Mohamed Elhoseiny, Babak Saleh, and Ahmed Elgammal. "Write a classifier: Zero-shot learning using purely textual descriptions." In: *Proceedings of the IEEE International Conference on Computer Vision*. 2013, pp. 2584–2591.

[9] Mohamed Elhoseiny, Kai Yi, and Mohamed Elfeki. "CIZSL++: Creativity Inspired Generative Zero-Shot Learning." In: *arXiv preprint arXiv:2101.00173* (2021).

[10] Mohamed Elhoseiny, Yizhe Zhu, Han Zhang, and Ahmed Elgammal. "Link the head to the" beak": Zero shot learning from noisy text description at part precision." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 5640–5649.

[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative Adversarial Nets." In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger. Vol. 27. Curran Associates, Inc., 2014, pp. 2672–2680. URL: https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

[12] Yu-Chao Gu, Le Zhang, Yun Liu, Shao-Ping Lu, and Ming-Ming Cheng. "Generalized Zero-Shot Learning via VAE-Conditioned Generative Flow." In: *arXiv preprint arXiv:2009.00303* (2020).

[13] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." In: *International conference on machine learning*. PMLR. 2015, pp. 448–456.

[14] Diederik P Kingma and Prafulla Dhariwal. "Glow: Generative flow with invertible 1x1 convolutions." In: *arXiv preprint arXiv:1807.03039* (2018).

[15] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes." In: *arXiv preprint arXiv:1312.6114* (2013).

[16] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. "Albert: A lite bert for self-supervised learning of language representations." In: *arXiv preprint arXiv:1909.11942* (2019).

[17] Hugo Larochelle, Dumitru Erhan, and Yoshua Bengio. "Zero-data learning of new tasks." In: *AAAI*. Vol. 1. 2. 2008, p. 3.

[18] Yannick Le Cacheux, Hervé Le Borgne, and Michel Crucianu. "Using Sentences as Semantic Representations in Large Scale Zero-Shot Learning." In: *European Conference on Computer Vision*. Springer. 2020, pp. 641–645.

[19] Federico Marmoreo, Julio Ivan Davila Carrazco, Vittorio Murino, and Jacopo Cavazza. "Learning without Seeing nor Knowing: Towards Open Zero-Shot Learning." In: *arXiv preprint arXiv:2103.12437* (2021).

[20] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. "Distributed representations of words and phrases and their compositionality." In: *arXiv preprint arXiv:1310.4546* (2013).

[21] Sanath Narayan, Akshita Gupta, Fahad Shahbaz Khan, Cees GM Snoek, and Ling Shao. "Latent Embedding Feedback and Discriminative Features for Zero-Shot Classification." In: *arXiv preprint arXiv:2003.07833* (2020).

[22] OpenAI. *CLIP: Connecting Text and Images*. 2021. URL: https://openai.com/blog/clip/ (visited on 01/09/2021).

[23] Tzuf Paz-Argaman, Yuval Atzmon, Gal Chechik, and Reut Tsarfaty. "ZEST: Zero-shot Learning from Text Descriptions using Textual Similarity and Visual Summarization." In: *arXiv preprint arXiv:2010.03276* (2020).

[24] Jeffrey Pennington, Richard Socher, and Christopher D Manning. "Glove: Global vectors for word representation." In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543.

[25] Farhad Pourpanah, Moloud Abdar, Yuxuan Luo, Xinlei Zhou, Ran Wang, Chee Peng Lim, and Xi-Zhao Wang. "A Review of Generalized Zero-Shot Learning Methods." In: *arXiv preprint arXiv:2011.08641* (2020).

[26] Nils Reimers and Iryna Gurevych. "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks." In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Nov. 2019. URL: https://arxiv.org/abs/1908.10084.

[27] Stuart Russell and Peter Norvig. "Artificial Intelligence: A Modern Approach, Global Edition 4th." In: *Foundations* 19 (2021), p. 23.

[28] Yuming Shen, Jie Qin, Lei Huang, Li Liu, Fan Zhu, and Ling Shao. "Invertible zero-shot recognition flows." In: *European Conference on Computer Vision*. Springer. 2020, pp. 614–631.

[29] Tasfia Shermin, Shyh Wei Teng, Ferdous Sohel, Manzur Murshed, and Guojun Lu. "Bidirectional Mapping Coupled GAN for Generalized Zero-Shot Learning." In: *arXiv preprint arXiv:2012.15054* (2020).

[30] Grant Van Horn, Steve Branson, Ryan Farrell, Scott Haber, Jessie Barry, Panos Ipeirotis, Pietro Perona, and Serge Belongie. "Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 595–604.

[31] *Weights & Biases*. en. URL: https://wandb.ai/mvalente/Zero_Flow_CUB/runs/h0e3nczw/overview (visited on 02/02/2022).

[32] *Weights & Biases*. en. URL: https://wandb.ai/mvalente/Zero_Flow_CUB/sweeps/8774skpy (visited on 02/02/2022).

[33] *Weights & Biases*. en. URL: https://wandb.ai/mvalente/zero_flow_imagenet/sweeps/3kao1t7a (visited on 02/02/2022).

[34] Peter Welinder, Steve Branson, Takeshi Mita, Catherine Wah, Florian Schroff, Serge Belongie, and Pietro Perona. "Caltech-UCSD birds 200." In: *California Institute of Technology* (2010).

[35] Ross Wightman. *PyTorch Image Models*. https://github.com/rwightman/pytorch-image-models. 2019. DOI: 10.5281/zenodo.4414861.

[36] Thomas Wolf et al. "Transformers: State-of-the-Art Natural Language Processing." In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. URL: https://www.aclweb.org/anthology/2020.emnlp-demos.6.

[37] Yongqin Xian, Christoph H Lampert, Bernt Schiele, and Zeynep Akata. "Zero-shot learning—A comprehensive evaluation of the good, the bad and the ugly." In: *IEEE transactions on pattern analysis and machine intelligence* 41.9 (2018), pp. 2251–2265.

[38] Yongqin Xian, Saurabh Sharma, Bernt Schiele, and Zeynep Akata. "f-VAEGAN-D2: A feature generating framework for any-shot learning." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 10275–10284.

[39] Cheng Xie, Hongxin Xiang, Ting Zeng, Yun Yang, Beibei Yu, and Qing Liu. "Cross knowledge-based generative zero-shot learning approach with taxonomy regularization." In: *Neural Networks* 139 (2021), pp. 168–178.

[40] Dimitrios Zeimpekis and Efstratios Gallopoulos. "CLSI: A flexible approximation scheme from clustered term-document matrices." In: *Proceedings of the 2005 SIAM International Conference on Data Mining*. SIAM. 2005, pp. 631–635.

[41] Yizhe Zhu, Mohamed Elhoseiny, Bingchen Liu, Xi Peng, and Ahmed Elgammal. "A generative adversarial approach for zero-shot learning from noisy texts." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 1004–1013.