香港城市大學
City University of Hong Kong

Department of Computer Science

BSCCS Final Year Project Report
2021-2022

21CS089

Attribute-Directed Extended Cross-Domain Mapping for Zero-Shot Learning

(Volume 1 of 1)

Student Name : **DABAS Jayant**

Student No. : **55295490**

Programme Code : **BSCEGU4**

Supervisor : **Prof CHAN, Antoni Bert**

1st Reader :

2nd Reader :

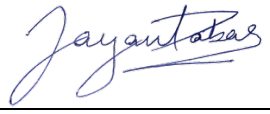
For Official Use Only

Student Final Year Project Declaration

I have read the project guidelines and I understand the meaning of academic dishonesty, in particular plagiarism and collusion. I hereby declare that the work I submitted for my final year project, entitled:

Attribute-Directed Extended Cross-Domain Mapping for Zero-Shot Learning

does not involve academic dishonesty. I give permission for my final year project work to be electronically scanned and if found to involve academic dishonesty, I am aware of the consequences as stated in the Project Guidelines.

Student Name: DABAS Jayant Signature: 

Student ID: 55295490 Date: March 21, 2022

Abstract

Zero-Shot Learning (ZSL) aims to learn an optimal mapping between attributes and classes by exploiting the visual knowledge from training data. It has gained popularity in many real-world applications like autonomous driving, detecting unforeseen diseases through medical imaging, speech recognition, and signal processing. Due to this, the number of proposed methods is increasing at a steady rate. However, we cannot analyze the progress or verify the results without any agreeable benchmarking protocols to evaluate these methods. Therefore, we propose new benchmarks that utilize the efficient techniques mentioned in previous papers to act as a standard reference for advancing this study area. An in-depth analysis is essential for comparing various approaches and understanding their good and bad sides. While previous studies treat all the attributes equally, the contribution of each attribute differs greatly in the training process. We propose to use an attention module to employ the discriminative properties of these attributes in our model framework. In recent years, Deep Neural Networks (DNNs) have gained increasing popularity due to their robustness in learning complex visual patterns. Moreover, extracting features with the help of a neural network makes our training process much faster and helps construct a better subspace for embeddings. Extending a state-of-the-art Deep-CDM method to a more realistic learning scenario, we display how existing techniques can be used to design new and better frameworks.

Acknowledgement

I would like to convey my sincere gratitude to my supervisor, Dr. Antoni Chan, for his continued support throughout this project. His scholarly advice and deep knowledge in the related areas have provided a lot of insights into this research. Even with his busy schedule, he never missed taking our doubts through biweekly meetings and emails, which provided the proper direction to this project.

I would also like to express my appreciation to the staff of the Department of Computer Science and the FYP lab, which offered excellent facilities and equipment that directly contributed to the completion of this project.

Special thanks to my sister, Mrs. Shivani Dabas Sihag, for taking out time to discuss important ideas and guide me in making my research better during the past year.

Moreover, I'm greatful to my good friends, who were ready to share computing resources to run the project code and save time.

Lastly, I want to thank all my family members, who have always been loving and supportive. Their constant encouragement has been my prime motivation.

TABLE OF CONTENTS

1 Introduction

1.1 Background and Motivation

The volume and availability of big data is increasing tremendously as the Internet integrates with our day-to-day lives. For example, an average of 2.6 quintillion bytes of data is generated every day [1]. However, most of this raw data is highly unstructured, and existing deep learning models cannot use it in the raw form to gain insights. The training data needs to be labeled by data science professionals, and it is impractical to label all the massive amounts of data manually. Meanwhile, humans identify classes of new objects like dog breeds, astronomical objects, or their environment all the time with very few samples.

Thus, innovative approaches like Zero-Shot Learning (ZSL) are essential as it can classify the data using only a few or even no labeled samples [2]. Zero-Shot Learning is a relatively new branch of machine learning that aims to solve a task without receiving data about all possible classes and objects in the training phase. This approach tries to identify new features in an unknown image and learns to classify them without fine-tuning or human interventions. The technique is very similar to how humans identify objects by looking at their attributes.

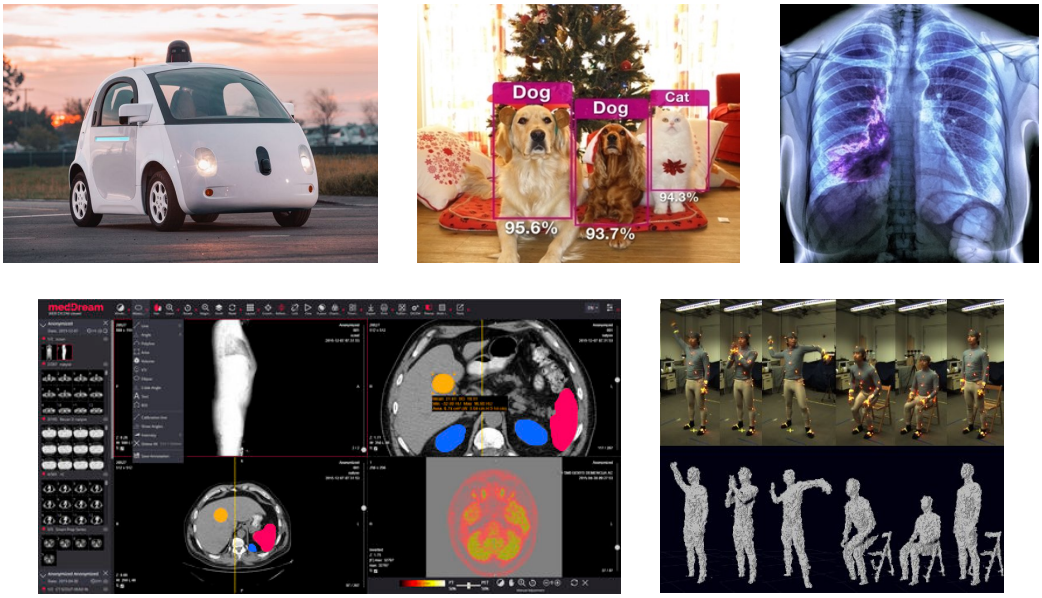


Figure 1 : Real-world applications of zero-shot learning – autonomous vehicle, medical imaging, action recognition, test-time optimizations, etc.

The motivation behind this research is the vast and growing number of new categories, making it impossible to collect and annotate all the instances, especially at the scale of big data. We continually observe new classes emerging every day, thus, increasing the complexity of the

whole process even further. For example, many statistical models estimated a total of 8.7 million eukaryotes species present on earth, of which only 1.64 million (around 19 percent) have been identified [3]. This project is also important because of the increasing applications of zero-shot learning in many real-world scenarios, like unfamiliar object classification in autonomous vehicles, natural language processing (NLP), test-time optimizations, action recognition, entertainment, human-computer interaction (HCI), or detection of unforeseen diseases (e.g., Covid-19) through medical imaging such as X-rays. That is why a proper analysis of existing designs and the development of new creative methods are necessary to push the boundaries of machine learning.

Existing approaches of ZSL models are classifiable as:

1.1.1 Embedding based approach

In this method, we try to map the image features and the semantic attributes of the object into a shared embedding space using a deep neural network. During the training phase, the aim is to reduce the loss between the projected vector and the ground truth for the semantic vector. Therefore, the same function can be generalized to predict attributes of unseen images while testing. After obtaining the semantics, we can find the nearest neighbor using k-means, use any similarity metrics like cosine similarity, or use direct matrix multiplication to acquire the class scores. The label corresponding to the nearest match is the predicted label of the instance. The figure below shows the structure of an embedding-based zero-shot learning method. First, the image features are obtained by passing the image through a feature extractor (PCA or deep neural network (DNN)). These image features represent the unique characteristics of the image that act as an input to the core method. The projection network then maps the features to semantic vectors using a compatible loss function. The semantic vectors are then used for the final classification.

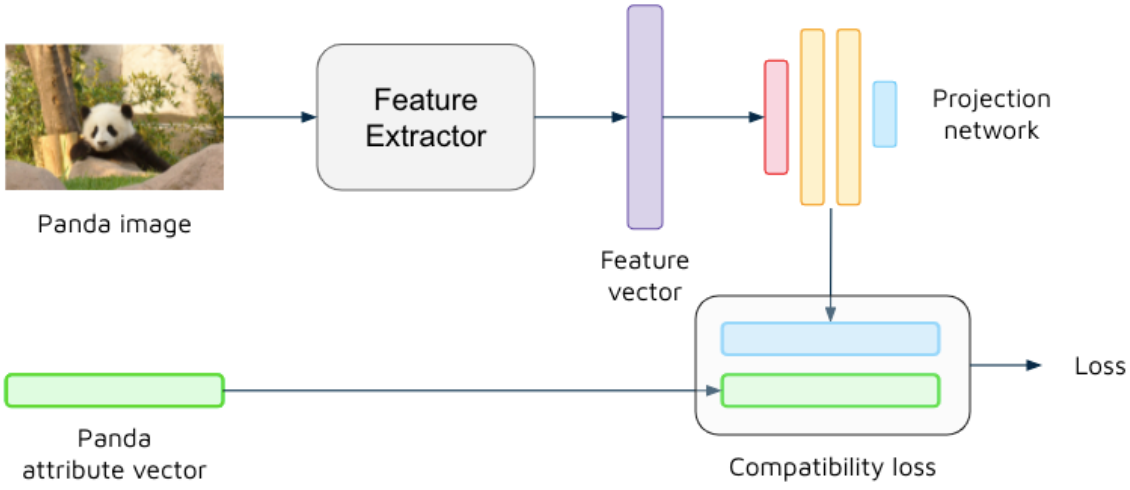


Figure 2 : Embedding approach in zero-shot learning

1.1.2 Generative model-based approach

The embedding-based methods face a significant disadvantage due to the problem of bias and domain shift. That is because, by learning only through the seen classes during training, the model is biased towards the seen categories. Thus, there is no certainty that the mapping of unseen labels would be correct.

The generative model-based approach overcomes this issue by training the model on both seen and unseen images. It generates image features of unseen classes with the help of a conditional generative adversarial network (cGAN) based on the semantic attribution of the given category. These synthesized features act as representatives of the test images to reduce the bias in the prejection of the learned embeddings into the semantic space.

The figure below illustrates a common generative model-based zero-shot learning. Similar to the embedding method, we first use a feature extractor to obtain the image features. The semantic attribute vector is input to the generative model to create an output vector representing the image from the unseen category. The generative model is trained in a way that the created output vector is similar to the original feature vector. When the model is developed, the generator's weights need to be adjusted, and the class attribute vector is passed as input to generate non-observed image features. After we have the image features of seen class (the training data) and the generated image features of unseen classes, we can train a more generalized mapping function with less bias.

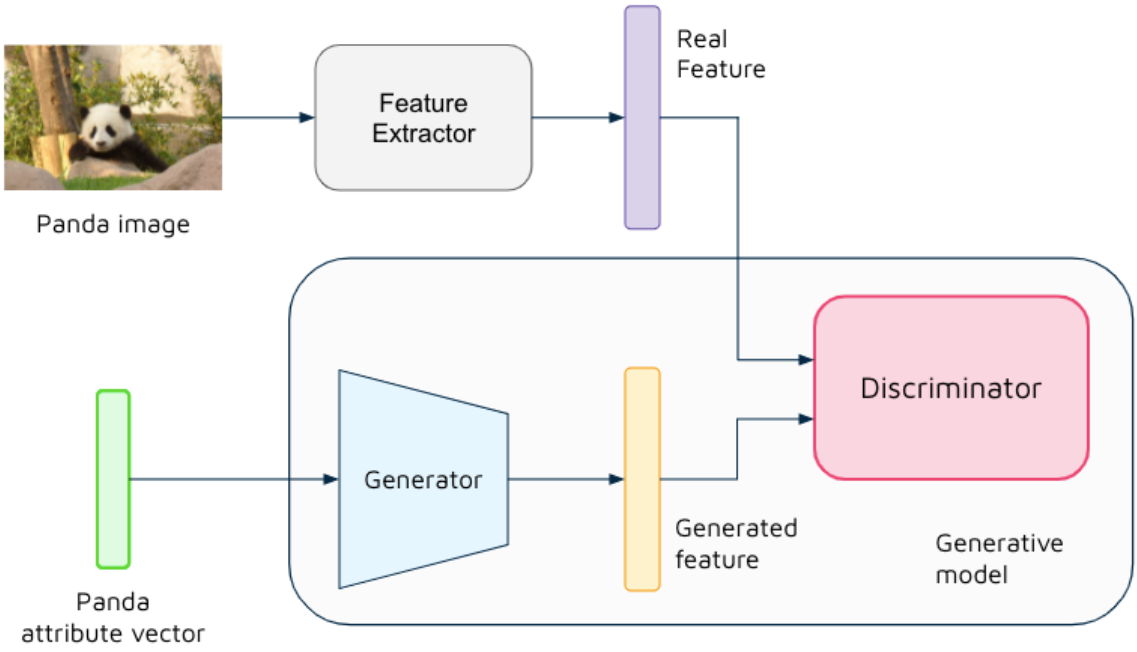


Figure 3 : Generative approach in zero-shot learning

1.2 Problem Statement

The number of zero-shot learning techniques has been growing each year. Even though modern methods have displayed promising progress in unsupervised image classification, the following cases require further study for proper implementation in real-world applications.

1.2.1 Accuracy Limited by Eminent Bias

Due to the inherent nature of the zero-shot learning paradigm, the models tend to be biased towards the seen classes. This bias appears because the model is trained only on the known categories. However, the embeddings for the unseen categories might not fit the learned projection mapping, giving us incorrect results. With a significant difference between accuracies of seen and unseen classes, methods that implement semantic embeddings perform poorly in generalized zero-shot learning (GZSL). There also exists another form of bias mentioned in [4]. It occurs when the image features extracted through a pre-trained deep neural network (DNN) on a large dataset containing test classes transgress the idea behind zero-shot learning, where training and testing classes are expected to be disjoint.

1.2.2 Hubness Issue in Word-Embeddings

The hubness problem can be described as an observation where the data points of a dataset, for increasing dimensionality, frequently occur in the k-nearest neighbor of other points. It happens due to distance concentration, with the points becoming highly skewed. As a result, hubs emerge, which might give us a list of the nearest neighbors with repeated points. It is a major issue when using multiple high-dimensional embeddings like in natural language processing or zero-shot learning. Therefore, increasing hubness can lead to incorrect classification of test images.

1.2.3 Poor Implementation of Attribute Coherency

Classes are defined based upon the relationship between various attributes and not just the attribute itself. However, initial ZSL models construct a semantic space by reducing the projection method to a binary classification function for each attribute. This approach loses the importance of the predicted attributes to a particular class and dramatically affects the performance of the overall model. The lack of coherent patterns in the semantic embeddings can render the attributes meaningless during the inference stage. That relationship is crucial in obtaining the correct labels for an image instance.

Hence, this project aims to target the above problems by developing a new and creative zero-shot model with state-of-the-art performance.

1.3 Objectives

The purpose of this project is threefold. These objectives will demonstrate promising improvements in zero-shot learning that may inspire future growth.

(1) First, we would like to present our **creative and new design** for a transductive model to solve zero-shot learning. The proposed method shows promising advancements. It is achieved by combining various innovative strategies to comprehend and target the underlying issues for ZSL tasks. For example, we use a multilayered image captioning model with better word embeddings to **reduce hubness** in attribute space. Over the years, text analysis has shown outstanding improvements in generalized scenarios, which is beneficial to our use case. Moreover, we power the captioner with generated semantics using proto-classes to **reduce bias** and expand the capabilities of our ADE design to multiple domains.

(2) Next, we try to implement a classifier that assigns bonuses and penalties to the class scores during the classification process to **make attributes more meaningful**. To achieve this, we utilize the attribute weights provided in the datasets. These weights represent the probability of the occurrence of a semantic in the respective class.

(3) Finally, to showcase the practical implementation of our model in a real-life scenario, we develop an **interactive web interface** that will use our trained model to demonstrate zero-shot classification. Along with the original code, the web demo is an essential part of the **technical aspects** of this project.

1.4 Scope

In order to achieve the objectives mentioned above, it is crucial to define the scope of this research. In the following sections, we will discuss the outcomes of the project and the importance of that scope. It will give a brief insight into the core modules that need to be developed. The detailed architecture and implementation of our design will be explained in the later sections.

1.4.1 Implement Generative Module

In the background, we discussed how the embedding-based approach suffers from significant bias towards seen classes. While GAN model-based approaches are popular for generating test features, we take a different approach by generating attributes using class prototypes with the help of semantic encodings []. This method allows our model to overcome the bias issue without the cost of extensive computing like in GANs.

1.4.2 Integrate Image Captioner

By exploring ways to integrate the image captioning module, we will be able to use their state-of-the-art performance in obtaining our semantic embeddings. The image captioner will allow our model to learn the projection mapping of visual features to attribute vectors in a multi-layer feed-forward approach. It will use both the training features and the features generated by our generative network to create the mapping. It is a creative approach that has never been implemented before in the existing methods of zero-shot learning.

1.4.3 Add Classification Penalties

While the classifiers of most ZSL methods use k-nearest-neighbor or class scores to label the instances, we propose to use class scores with added penalties to introduce attribute relations in our final class probabilities. This technique can prove to be a fast and efficient way to reduce hubness between classes. The classifier also adds a bonus point if the most important attributes are present for that class label. The classification process is an essential aspect of the model because even after the features are mapped to the embedding space, we need to identify the connection between the occurrence of various attributes for a particular class. Correctly identifying these incoherent attributes of classes can tremendously boost the performance of the model.

1.4.4 Create a Web Demo

The final product in this project's scope is our proposed ADE model and a web application to demo the saved model used in the observations. The web demo is a simple and intuitive way of presenting the results of this research project, demonstrating the effectiveness of our model. The details of the application developed are discussed in the implementation section of this report.

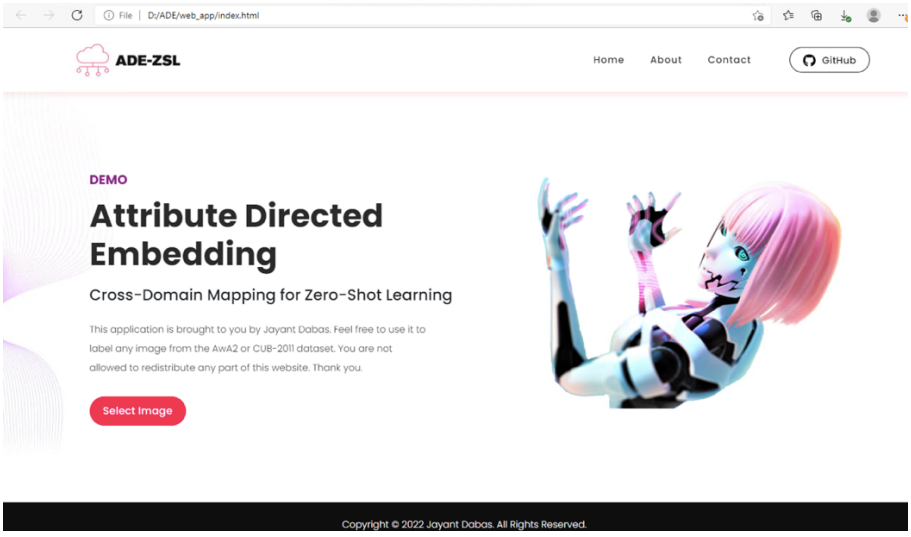


Figure 4 : ADE-ZSL Interative web demo

1.5 Deliverables

The project aims to deliver a new and innovative Attribute Directed Extended (ADE) zero-shot learning model. We also plan to use the trained model to present our demo as a web application for easy testing. The web demo consists of a web interface that connects to the python model and labels the uploaded images in real-time.

1.6 Major Technical Components

1.6.1 Datasets

Attribute dataset Animals with Attributes (AwA2) [5] will be used to train and evaluate the models used in this study. After carefully examining the available options, these datasets were selected because of their wide acceptance and usage in the major published papers. Moreover, zero-shot learning being a relatively new topic, only particular datasets with a good number of attributes and appropriate size can be used to evaluate the model's performance.

1.6.2 Language

Python is the primary language for programming the proposed model, while the web application will be designed in core HTML5, CSS, and JavaScript. Additionally, certain python packages may be written in C/C++. The demo will interact with the python program via ajax requests. It also uses bootstrap to smoothen the user experience when using our application.

1.6.3 Software

Software that will be useful in completing this project are - Visual Studio Code, MATLAB, Jupyter notebook, Terminal, Git, Flas, and Tensorflow.

1.7 Report Organization

The rest of the report is organized as follows. Section 2 reviews the related work to provide context to the subject domain and realize its importance. It would help us get an insight into the essential methods and technologies required by the project. Section 3 presents the design of the core modules and the overall architecture of our proposed network.

Section 4 discusses the methodologies used and their implementation in the project solution. In section 5, we look at the observations and the results of our ADE framework. We conclude the report by discussing the possible future improvements for the proposed method.

2 Literature Review

The idea of the Zero-Shot Learning approach focuses on retrieving a feature-rich space and developing a mapping function for precise accuracy. By understanding the process of obtaining features and the available methods, we try to find scopes of improvement for getting better attributes. The interpretation of existing ZSL-based models aims to realize the strengths and weaknesses of various processes employed to help design our solution. We also study some unconventional methods to look for ways of outperforming state-of-the-art models.

2.1 Image Transformations

2.1.1 Gaussian Smoothing

In image processing, a Gaussian kernel is widely used to downsample images or remove noise. It is a linear filter that has the shape of the 'Gaussian Distribution' function (Figure 1). The 2D convolution can be applied to an image to blur it like a mean filter where the standard deviation of the operator determines the degree of smoothness [6]. It uses the weighted average of surrounding pixels with more weight towards the center pixels to provide a gentler smoothing effect. Gaussian smoothing has various imperative use cases like photography, graphics software, and facial recognition because it is symmetric and easy to implement. However, the image loses details of edges and contrast by applying a Gaussian filter [7]. Hence, a separate filter is required to preserve the knowledge of nonlinear sample space.

$$G(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}} \quad (1)$$

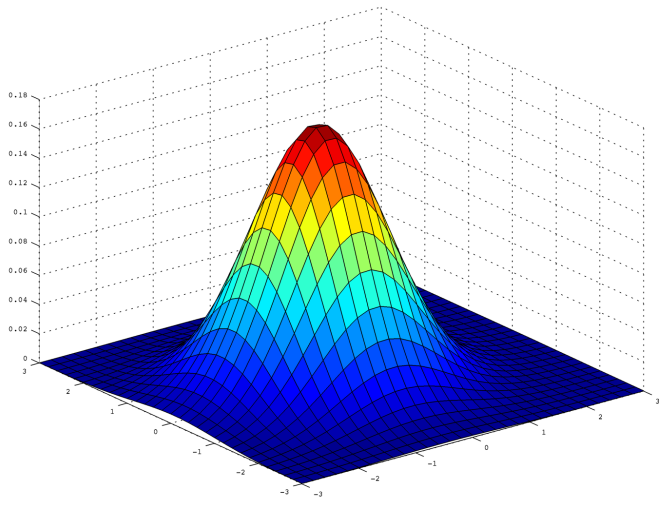


Figure 5: Gaussian curve

2.1.2 Median Diffusion Filtering

'Median filter' is another popularly used smoothing operator in image processing, time series processing, and signal processing. It is a nonlinear filter that helps in reducing salt-and-pepper or impulsive noise from the input signal. The median filter can remove extreme pixel values (noise) without having an impact on the surrounding pixels [8]. Its ability to eliminate input noise of remarkably high magnitudes gives median filters a significant advantage over linear filters like Gaussian blur. In linear filters, the output is degraded severely by even a small anomaly in noise. Though, a median filter of large size are slow and may introduce patterns or textures that were initially not existing in the original image. Therefore, we need to use a median filter of optimal size to receive a desirable feature rich image on performing transformation. Figure 3 shows our test results where median filter clearly outperforms gaussian smoothing.

Noisy Image Gaussian filtered Median filtered

Figure 6 : Comparison of Gaussian and Median filters

2.2 Feature Extraction

2.2.1 Principal Component Analysis

Principal Component Analysis (PCA) is a standard approach used to reduce the dimensionality of enormous datasets while maintaining valuable information. It is achieved by obtaining uncorrelated or independent variables that maximize the variance. These new variables are known as 'principal components,' which reduce the method to an eigenvector problem [9]. PCA is an adaptive data analysis method with multiple variants to deal with different types of data. Reducing the number of dimensions of a dataset always comes at the cost of accuracy. However, the goal in dimensionality reduction is to trade a little accuracy for clarity in data. The data with fewer dimensions is easier to explore and visualize, making it much simpler and faster for machine learning algorithms to process the data without any extra variables. In some instances, PCA

performs better at a low cost of accuracy by providing uncorrelated features, removing noise, and reducing overfitting.

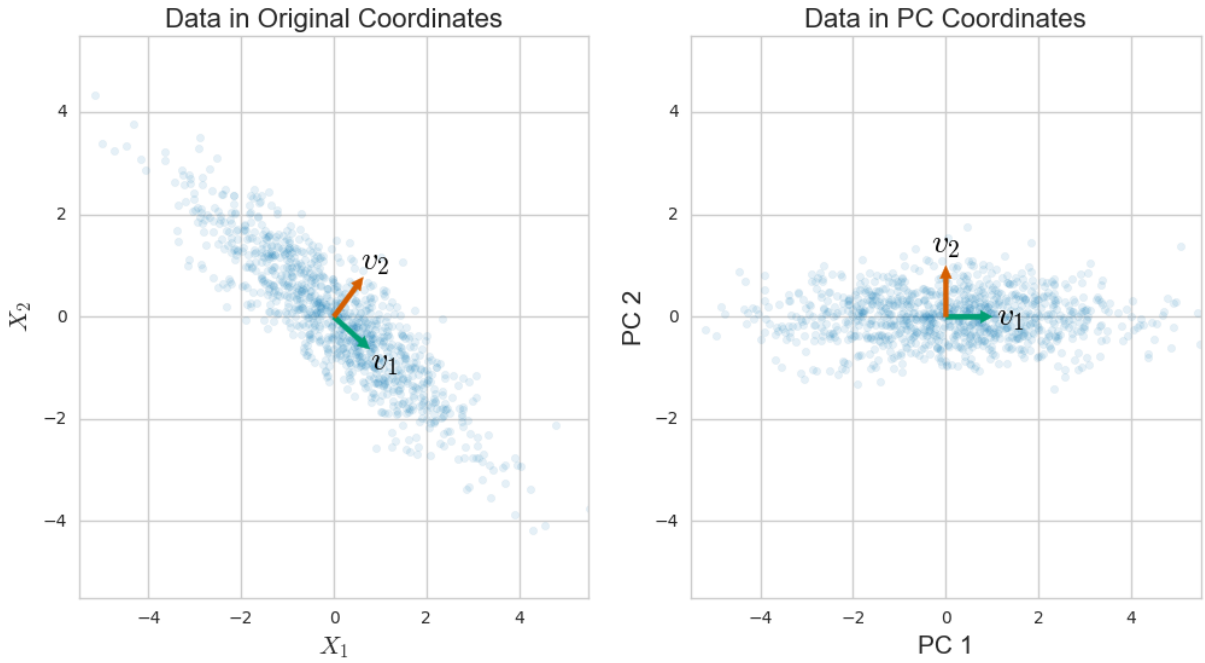


Figure 7 : Basic visualization of PCA

Steps involved in the technique of PCA are -

1) Standardization of range of the initial variable for equal contribution.
2) Calculation of covariance matrix to discern the correlation between data points with respect to each other.
3) Compute the eigenvectors and eigenvalues of the covariance matrix to obtain the principal components
4) Feature vector consisting of highly relevant principal components

Thus, Principal Component Analysis would be useful in reducing the number of non significant features in our solution design.

2.2.2 Scale-Invariant Feature Transform

The scale-invariant feature transform (SIFT) is an algorithm used in computer vision to extract, describe and match local features of images. This method is invariant to the rotation and scale of input images, making it highly applicable in real-world scenarios. It can generate a large number of features, even for a small image. No prior segmentation is required in SIFT because it

uses local features [10]. It is also easily extendible to a wide range of attribute types while being robust.

SIFT uses the following steps for finding the features:

1) Scale-space peak selection - To explore the possible locations of features.
2) Keypoint localization - For precisely obtaining the keypoints.
3) Orientation - Assigning orientation to the obtained feature keypoints.
4) Descriptors - Describing features as a vector.

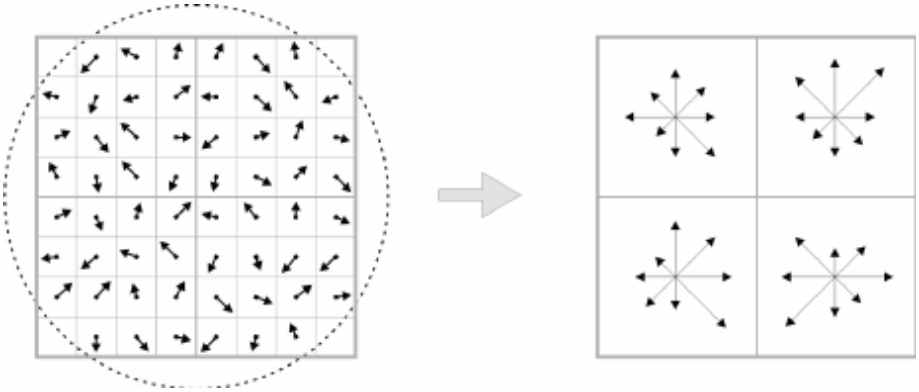


Figure 8 : Feature descriptors (Lowe, 2004) [22]

The SIFT features improve the model performance by solving the problem of orientation or scale. But SIFT still has some disadvantages of being slow, mathematically complex, computationally expensive, and unsupported on low-end devices [11]. That is why other alternatives like SURF and Oriented FAST and Rotated BRIEF (ORB) might be an effective solution if we cannot implement SIFT due to its patent.

2.2.3 Oriented FAST and Rotated BRIEF

Oriented FAST and Rotated BRIEF (ORB) is a modified combination of FAST keypoint detector and BRIEF descriptors to achieve high performance. It is one of the fastest and light weight technique for feature extraction [12]. Similar to the goals of SIFT, it uses the FAST algorithm to locate keypoints and apply Harris corner detection to select top N points. It uses an intensity weighted centroid to measure the orientation of the keypoints. ORB uses BRIEF descriptors for selected keypoints, but BRIEF has poor performance in rotation. Therefore, ORB "steers" BRIEF in the direction of the keypoints to enhance accuracy. It is a fast algorithm that will be profoundly beneficial in generating feature-rich space for semantic encoding in our proposed model.

Figure 9 : Feature matching in Computer Vision

2.2.4 ORB-PCA

In this method, the Oriented FAST and Rotated BRIEF (ORB) algorithm is coupled with Principal Component Analysis (PCA) to produce faster and more accurate results. While SIFT is the most popular feature extractor with rotational, scale, and illumination invariance, it takes relatively more time to extract the large set of features [11][13]. In some dimensions, ORB performs better than SIFT, and it is primarily applied in facial recognition [11]. That is why ORB is a good fit for our solution design. Integrating PCA allows us to reduce dimensions to save on time and computation costs.

Process for ORB-PCA based feature extraction

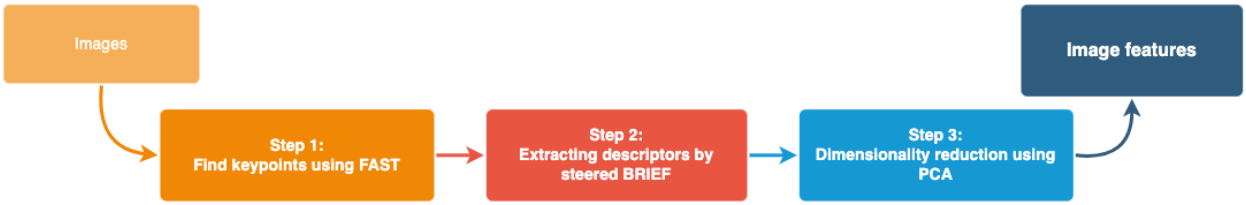


Figure 10 : ORB-PCA algorithm pipeline

2.2.5 CNN Features

Convolutional Neural Network (CNN) is a type of artificial neural network that is good at detecting patterns in input data. It is often used in computer vision for facial recognition, object classification, natural language processing, drug discovery, and video analysis. By solving the problem of spatial dependency, CNNs allow us to extract local features from an image which are scale and rotation invariant [14]. We can obtain abstract features, i.e., the initial layers detect low-

level features like edges and shapes, while deep layers get higher-level features such as faces and objects. Since CNN is fast, more reliable and easy to implement, using the features extracted by a CNN model like ResNet, Xception or YOLO is another suitable choice for generating semantic space for the proposed solution.

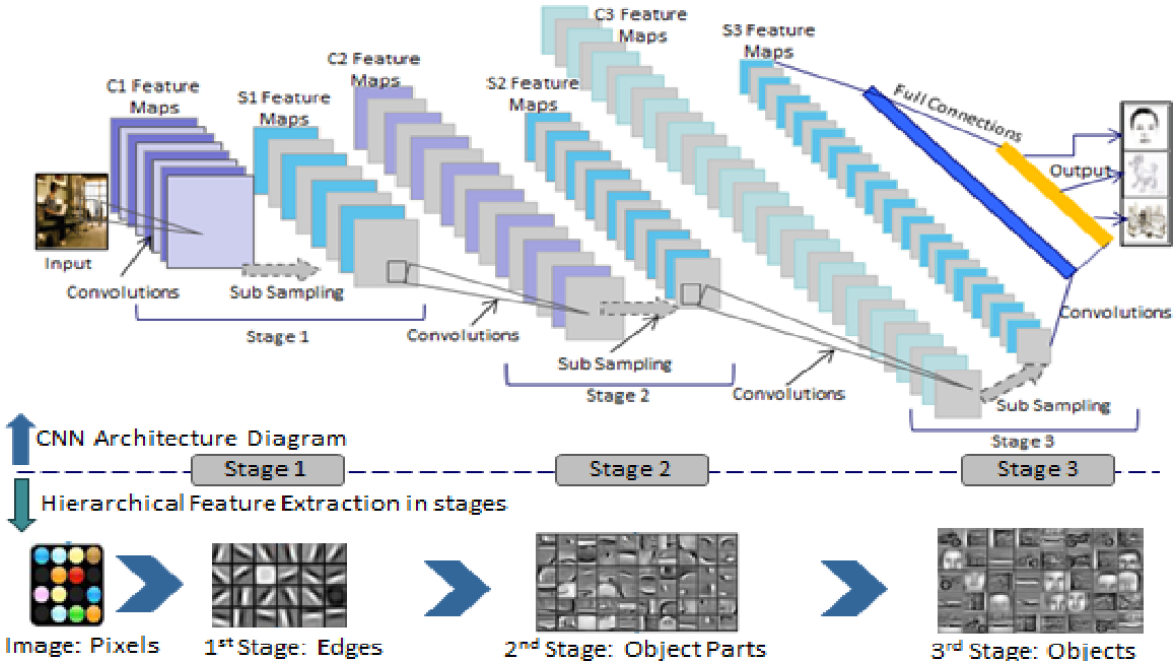


Figure 11 : Feature extraction using CNN architecture (Source: [27])

2.3 Learning Approaches

2.3.1 Supervised Learning

Supervised machine learning is a subcategory of deep learning algorithms which uses labeled datasets to train models to achieve accurate predictions. In this method, the training data (x) fed to the model is used to adjust the weights (w) of the evaluation function (f) by comparing the output (y) with the expected label during the cross-validation process. Supervised learning can be grouped into two types of problems - classification and regression. During classification, the algorithm attempts to label specific entities in the dataset, whereas regression helps to understand the relationship between dependent and independent attributes. Although supervised learning provides better performance, it is impossible to manually label everything in this world of big data. Therefore, the development of an automated approach like Zero-Shot Learning to label unseen classes is essential for the future of machine learning.

$$Y = \sum_{i=1}^{n} w_i \cdot x_i + b \qquad (2)$$

2.3.2 Semi-Supervised Learning

Semi-supervised learning is a fusion of supervised and unsupervised learning. This learning method is relevant to a plenitude of practical cases where it is expensive to label all the data. It requires less labeled data than supervised learning by applying pseudo labeling. The model is trained with a small amount of data until we see good results. It then trains on unlabeled data to predict pseudo labels by finding patterns in the data. The term 'pseudo' is used because the outcome may not be accurate. Now, the labeled and unlabeled training data is linked to retrain the model to increase its accuracy. A text document classifier is an ideal application of semi-supervised learning since it is difficult to find a large amount of labeled text data. Few-Shot learning and Zero-Shot learning are also regarded as semi-supervised approaches.

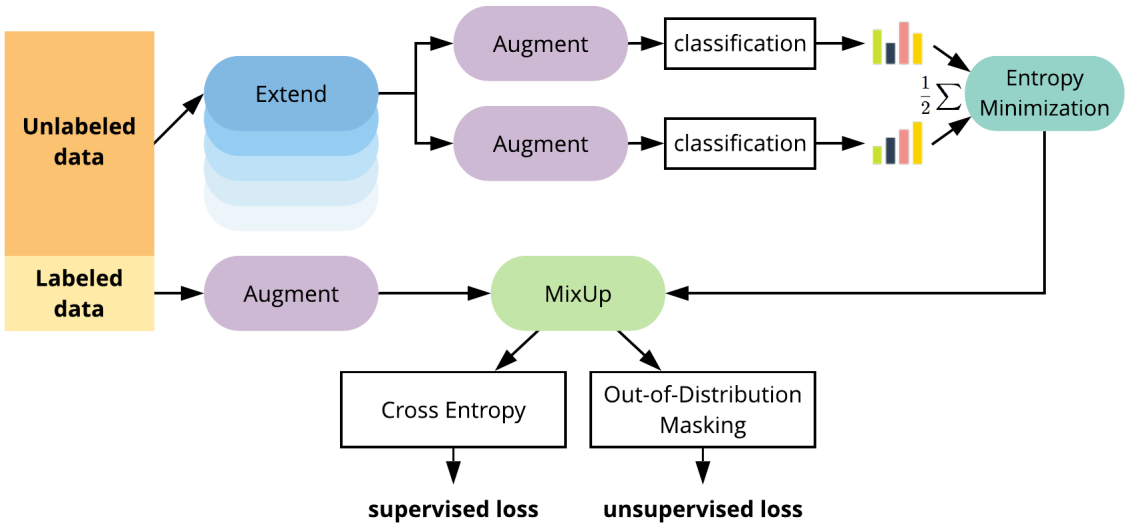


Figure 12 : Unsupervised vs Supervised Learning [14]

2.3.3 Unsupervised Learning

Unsupervised learning aims to examine and aggregate unlabeled data by discovering hidden patterns or groupings without human intervention. It is different from supervised learning as the input in unsupervised learning only contains attributes (x) and no label. That is why we cannot verify the predictions of the algorithm by comparing them with the expected output. Unsupervised learning is ideal for exploratory data analysis and image recognition due to its ability to identify similarities and differences in the provided information. A few popular unsupervised learning approaches are hierarchical clustering, probabilistic clustering, PCA, SVD, and autoencoders. It also supports many real-world applications like Google news, medical imaging, anomaly detection, and recommendation systems, allowing businesses to find patterns in large volumes of data in less time. The advantages of unsupervised learning is that it does not required labelled training data, it is robust and handles classification tasks pretty well. Although,

it is less accurate than supervised learning with less control over spectral grouping. Therefore, it is important to improve ZSL as semi-supervised learning and then extend its scope to unsupervised use cases.

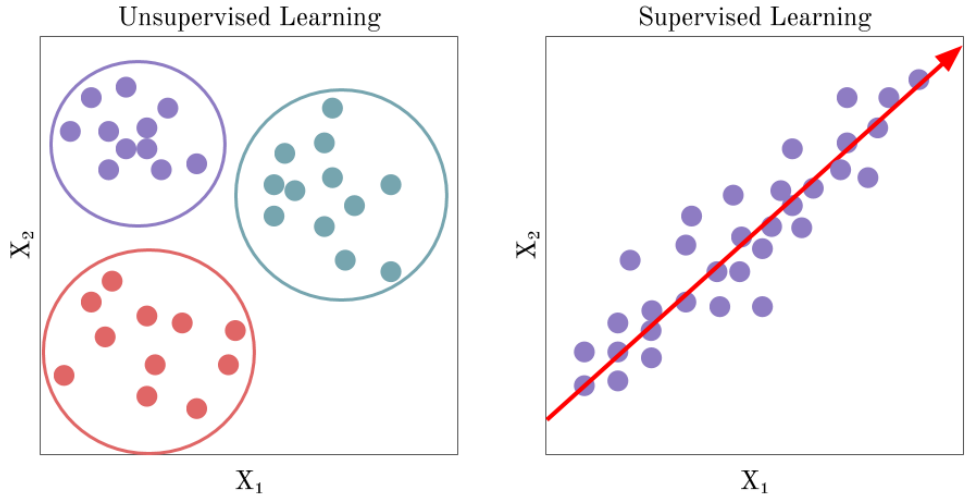


Figure 13 : Unsupervised vs Supervised Learning [22]

2.4 Types of Zero-Shot Learning models

2.4.1 Linear Layered networks

Zero-Shot learning applies feature descriptions of given instances to recognize new concepts. Although it seems to be a complex process, some simple approaches [15,16,17,18,19] have been able to achieve improved accuracies with a linear network that models the relationship between attributes and classes. These models would also provide opportunities to develop lightweight applications that can run on less powerful devices like smartphones and tablets. However, this linear framework does not solve the problem of domain shift and semantic loss. These issues can be solved by combining the linear model with a cross-domain mapping model to reduce the bias. It will enable us to take advantage of the strengths of linear ZSL while also learning a more powerful mapping function.

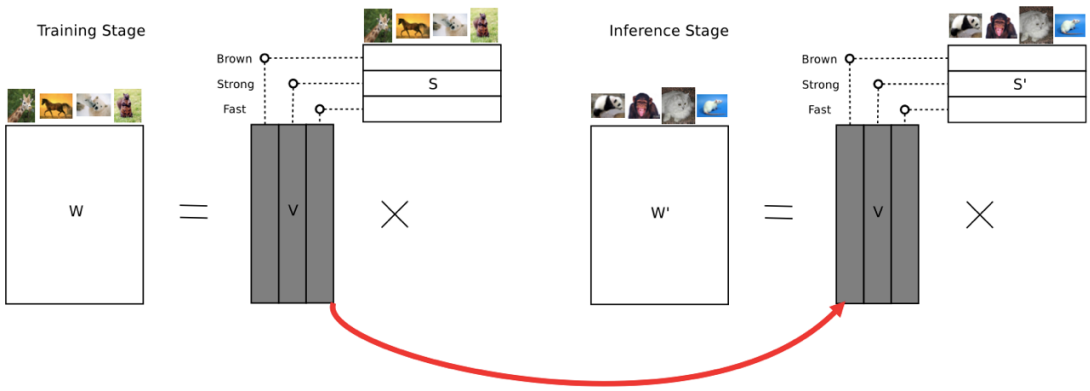


Figure 14 : Linear framework for ZSL as described in [15]

2.4.2 Semantic Embedding Models

This category of Zero-Shot Learning models mainly focuses on encoding semantic information in the model to achieve better performance. The semantic embedding space allows ZSL to transfer knowledge. This space can be an attribute space or word vector space [20], in which names of seen and unseen classes are represented as high dimension vectors known as 'class prototypes' [21]. Semantic embedding models take advantage of this feature space to attain high accuracies. However, it commonly suffers from the problem of semantic loss, which will be resolved in our solution.

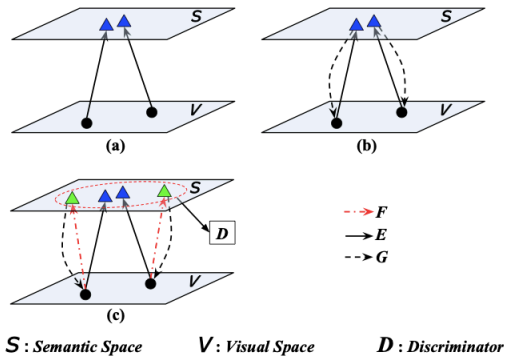


Figure 15 : SP-AEN framework [20]

(a) Semantics-Preserving Adversarial Embedding Networks (b) Semantic autoencoder (c) SP- AEN, with an independent visual-to-semantic F and an adversarial-style discriminator D [20]

2.4.3 Cross Domain Models

The Cross-Domain mapping (CDM) approach is a fusion of linear and embedded models that takes advantage of both forms. It also enables us to address the problem of domain adaptation by learning projection even for unseen classes. However, we need to introduce a threshold to minimize the loss caused by the domain shift. The CDM model iteratively calculates the class prototypes and optimizes the mapping to better fit the unseen classes [23]. It tries to extend inductive ZSL to transductive methods. Therefore, Cross-Domain mapping can prove to be beneficial in employing an unsupervised learning network.

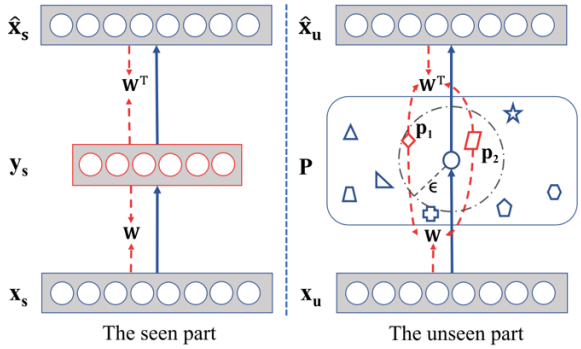


Figure 16 : CDM-SAE model proposed in [23]

2.5 Generative Adversarial Networks

Generative models are one of the most promising innovations to analyze the massive amount of cross-related information that is accessible today. To train these models, we require a large amount of data in a specific domain (e.g., millions of images, texts, or sounds). Later, the model is capable of generating new samples in that domain using the distinctive properties of that object in an unsupervised manner. Generative Adversarial Networks (GANs) are generative models, first proposed in 2014 [24]. They use a creative way of training the model by framing the unsupervised task as a supervised problem. It is done with the help of a generator and a discriminator, where both sub-models cooperate to boost the performance. The generator model is responsible for generating new fake examples using a latent vector. Then, the discriminator ensures whether the generated sample is classifiable as an object of that class. In this approach, we try to improve both models to obtain realistic results. GANs are an exciting and rapidly improving field, able to deliver on the promise of generative models. They have been successfully applied in a range of domains, like image translations, noise removals, editing image environments, super-resolution, and many more.

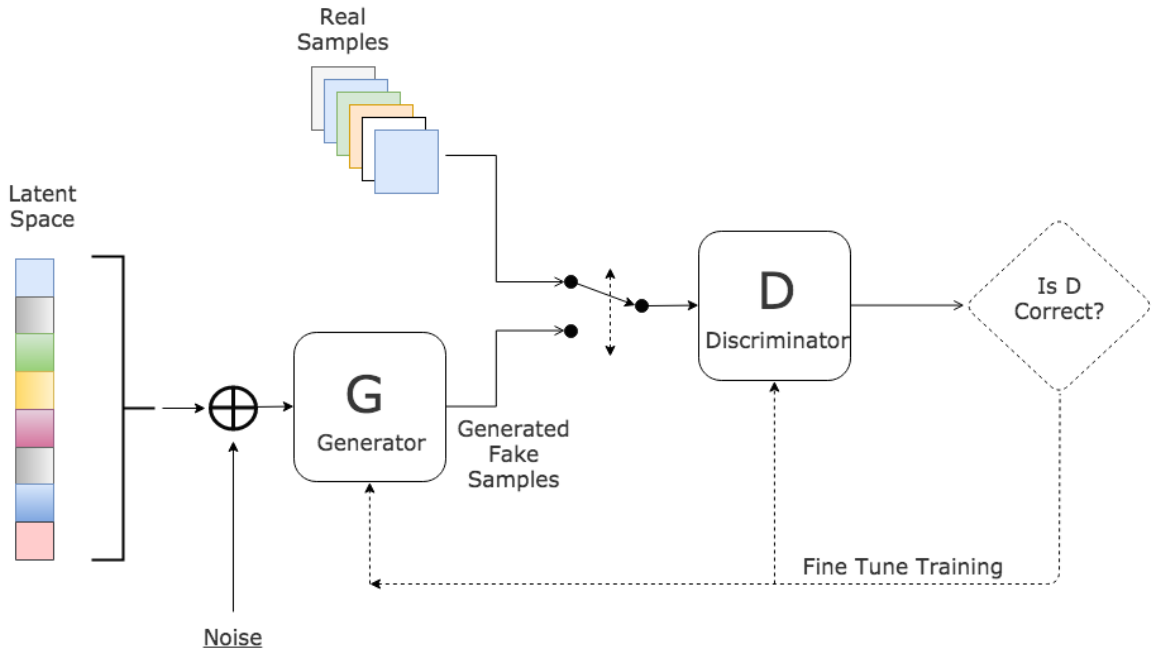


Figure 17 : Generative Adversarial Network (GAN) architecture

$$\min_G \max_D V(D,G) = E_{x\sim p(x)}[logD(x)] + E_{z\sim p(z)}\left[\log\left(1 - D\left(G(z)\right)\right)\right] \quad (3)$$

The above equation (3) shows the objective function of a GAN model, where the first half of the equation represents log probability of discriminator predicting that the real data is genuine and second half is the log probability predicting if the generated sample is not genuine.

2.6 Auto-Image Captioning

Linguistic indexing, popularly known as image captioning, is a method used to generate textual descriptions for visual data. It is a two-step process that requires a profound understanding of the image features and the correct linguistic model to match the syntax and semantic embedding of the language. The advancements in deep learning techniques have shown major improvements in natural language processing, where now it is possible to build models that can generate realistic captions [25]. Most linguistic models use the power of Long Short-Term Memory (LSTM) or Gated Recurrent Units (GRU) networks. These are a type of Recurrent Neural Network (RNN) that can recognize ordered patterns in sequential data. The first step in linguistic indexing is to extract image features using an encoder. According to the requirements, the encoder can be any feature extraction method like PCA, ORB, CNN, SURF, or SIFT. Convolutional neural network (CNN) is one of the most widely adopted feature extractors because of its effectiveness in different scenarios. Once we get the features, the next step is to use a decoder to map the visual features to linguistic embeddings. The embeddings can then be converted back to captions. The structure of a typical decoder is shown in the figure below.

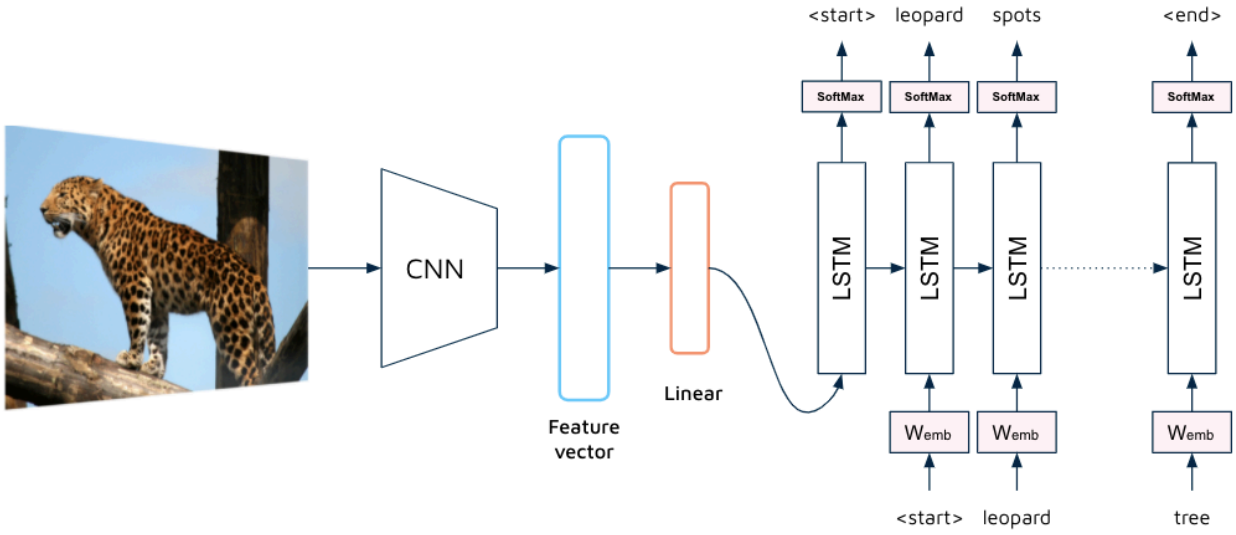


Figure 18 : General architecture for image captioning

2.7 Visual Attention

The attention mechanism is a technique used to focus a model's attention on the essential parts of an image and is commonly used in sequence-based models. The current algorithms use all the features in the image to train the model, but the presence of extraneous features may impact its performance [33]. That is why we learn attention mapping to map the relevant local features to

their representation in semantic space. This kind of mapping can significantly improve the overall performance of our solution by generating a meaningful feature space. Attention mechanisms work on the idea that the human brain intrinsically selects specific portions of the visual information and focuses on those regions. An Attention Network is responsible for feature localization in the model to focus the attention of the decoder on specific regions of an image. By doing so, we express the features obtained by their relevance to the task making the training process smoother and faster.

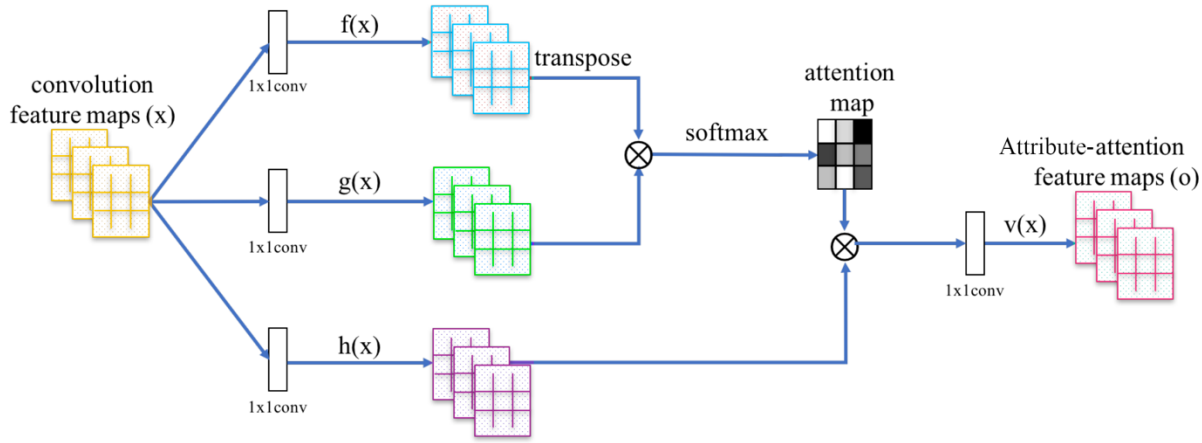


Figure 19 : Structure of an Attribute-weighted Attention Network [26]

3 Design Analysis

3.1 Environment Consideration

As this research largely deals with the software aspects, it is crucial to prepare an environment to meet our design requirements. First, we need a graphics processing unit (GPU) enabled system to ensure that we can train the models and carry out analysis in a reasonable amount of time. It is possible by using the Compute Unified Device Architecture (CUDA) application programming provided by Nvidia. An essential requirement for the success of this project is the availability of data. All of the datasets used in our analysis are publicly available, and a backup will be created to keep our evaluation results consistent during later readings.

3.2 System Design

The below diagram represents the core design of our proposed model. It primarily consists of 6 main components, which will be discussed in the later section. First we separate the data into training and testing for the seen and unseen classes, respectively. The first module is responsible

to provide augmented image for the training samples. It is done to increase the variety of images available for training our model. The second module is a feature extractor based on the ResNet-101 architecture. It takes the 3-dimensional image vector as input and predicts the visual feature, $V \in \mathbb{R}^{N \times m}$. These image features are then forwarded to the ADE-Generative module to generate semantic vectors, $S_G \in \mathbb{R}^{U \times D}$, for each sample using pseudo class. These generated vectors allows our model to reduce bias when using the model in a generalized setting. In the next step, the train, augmented, and test features are fed to the auto-captioning module along with their respective tokens. It allows the model to create well-defined global embedding space for both seen and unseen class captions. Since the captions are a textual representation of the class semantics, we need to decode the predicted caption to the binary attribute vector. This is done with the help of semantic translator, implemented specially for this task. The translated attributes, $S_p \in Seen$ and $S_p \in Unseen$ are passed on to the classifier to calculate class scores and apply penalties. The final scores are then used as class probabilities to classify the image instance.

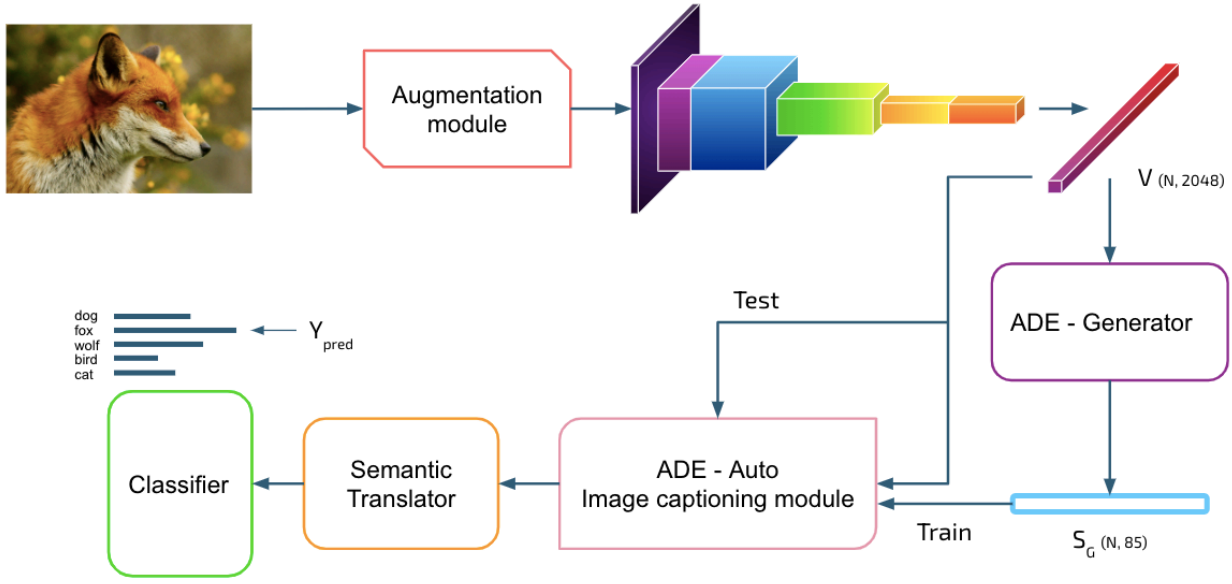


Figure 20 : ADE ZSL - System Design

3.3 System Components

In this section, we will discuss the primary components that will be included in the system design of the proposed model.

3.3.1 Image Augmentation Module

Neural networks work by matching similarities and patterns in the features of class images. This task requires a large number of samples to converge the objective function and provide a

good accuracy. However, we might not always have the amount of data needed to train a model. In such scenarios, image data augmentation is employed. It is a technique used to artificially expand the number of training samples by making random modifications to the image while keeping the main features intact. It also applies to the datasets used in this research, where the CUB dataset only has a maximum of 60 images per class. Since the given set of images is not enough to train our model, we propose to include an image augmentation module. This module is specifically designed to apply random flips, rotations, cropping, translation, and other necessary augmentations to maintain design consistency. This addition is expected to further boost the model's overall performance by improving the quantity of visual data used by our framework.

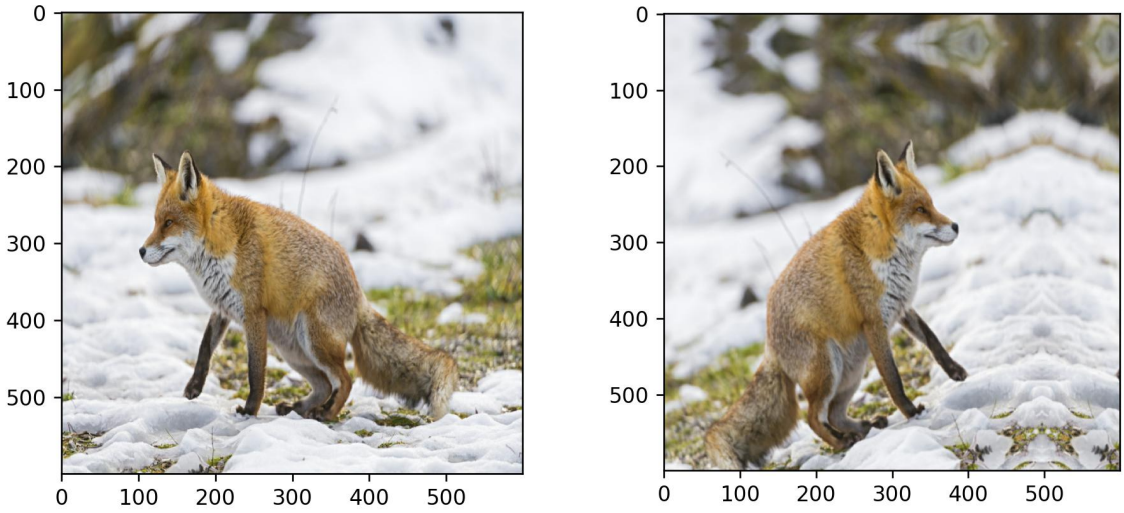


Figure 21 : Image augmentation - Original image (left), Aumented image (right)

3.3.2 Feature Extractor – ResNet-101 (DNN)

Before we begin, the most crucial step of any computer vision program is to extract visual features from the input image. These features allow the model to understand the patterns in the mathematical vector form to be able to get processed by the system. It is necessary because, unlike humans, the machine only understands the numerical representations of patterns like corners, edges, and gradients. In recent years, incredible progress in the extraction process has made it possible to obtain more reliable features in considerably less time.

We explored various popular extraction methods like PCA, SIFT, ORB, ORB-PCA, and CNN features, which have been discussed in the literature review. Among these methods, deep neural networks like ResNet-50 and VGG16 have been widely adopted in state-of-the-art machine learning algorithms. However, we decided to employ the ResNet-101 model with frozen layers as a feature extractor in our solution. Our choice is based upon the number of model size, performance, input shape, and the number of channels in the output feature. The ResNet-101 is a

suitable network because it has only 44.7 million parameters, compared to 66.7 million and 138.4 million trainable parameters in Efficient-B7 and VGG-16, respectively. Moreover, the top-5 accuracy for ResNet-101 trained on the imagenet dataset is 92.8%, which is a little higher than the other models like VGG-16.

The deeper EfficentNet models, i.e., B4~B6, were avoided as they do not improve the performance significantly, use more parameters, and need more processing time. Therefore, we use ResNet-101 as a pre-trained model with frozen layers to predict our features.

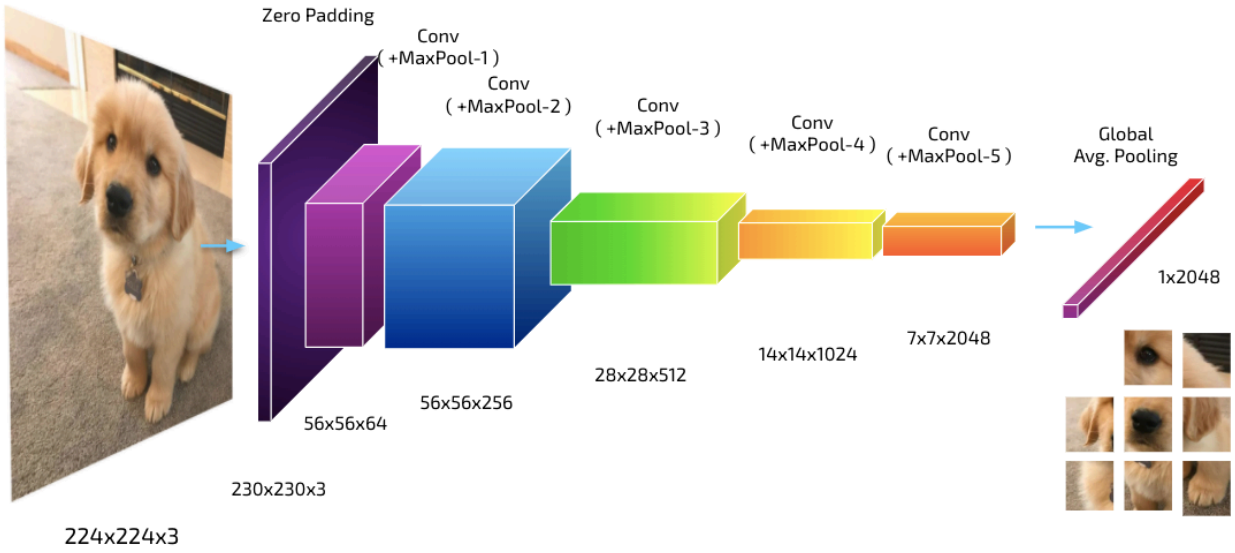


Figure 22 : Feature Extractor – ResNet 101 model

3.3.3 ADE – Generator Module

The generator module is responsible for providing synthesized attribute vectors (attribute directed) for each test instance. This process is done by simply projecting the test features to a normalized semantic space using the weight matrix obtained from the training data [27]. We use it to replace GANs, which perform well but are computationally expensive. While the generated vectors cannot be directly used for classification in generalized zero-shot learning, we can achieve almost the same test accuracy as conventional ZSL by combining it with our captioning module.

3.3.4 ADE - Multi-Layered Auto Image Captioner

Image captioning is the process of creating a textual description of an image. Originating from the domains of natural language processing and computer vision, 'automatic image annotations' have led to breakthrough innovations in the field of computer science. Today, we can produce image descriptions in real-time with the help of highly efficient and optimized embedding spaces. The captioner uses the image features to map them to the word-tokens in a separate

embedding space. The projection mapping is done in a two-way process using an encoder and a decoder. The encoder is responsible for extracting image patterns or features that will be passed as an input to the decoder. Since we have already discussed the feature extractor as a separate module, we can directly move on to the decoder. The decoder is a Recurrent Neural Network (RNN), which handles the language modeling. Most of the existing zero-shot learning solutions that implement captioning methods use a single-layer feed-forward long-short term memory (LSTM) network. However, our module uses three bidirectional Gated Recurrent Unit (GRU) layers where the features are also input at each level to decode visual patterns. GRU is similar to LSTM, although it has a fewer parameters making training faster.

Moreover, before inputting the image features to our decoder, we pass them through a 'transfer map' layer, which prepares the features for embedding. The first time step receives the mapped feature and the start token. We stop the time steps on receiving the end token or reaching the caption word limit.

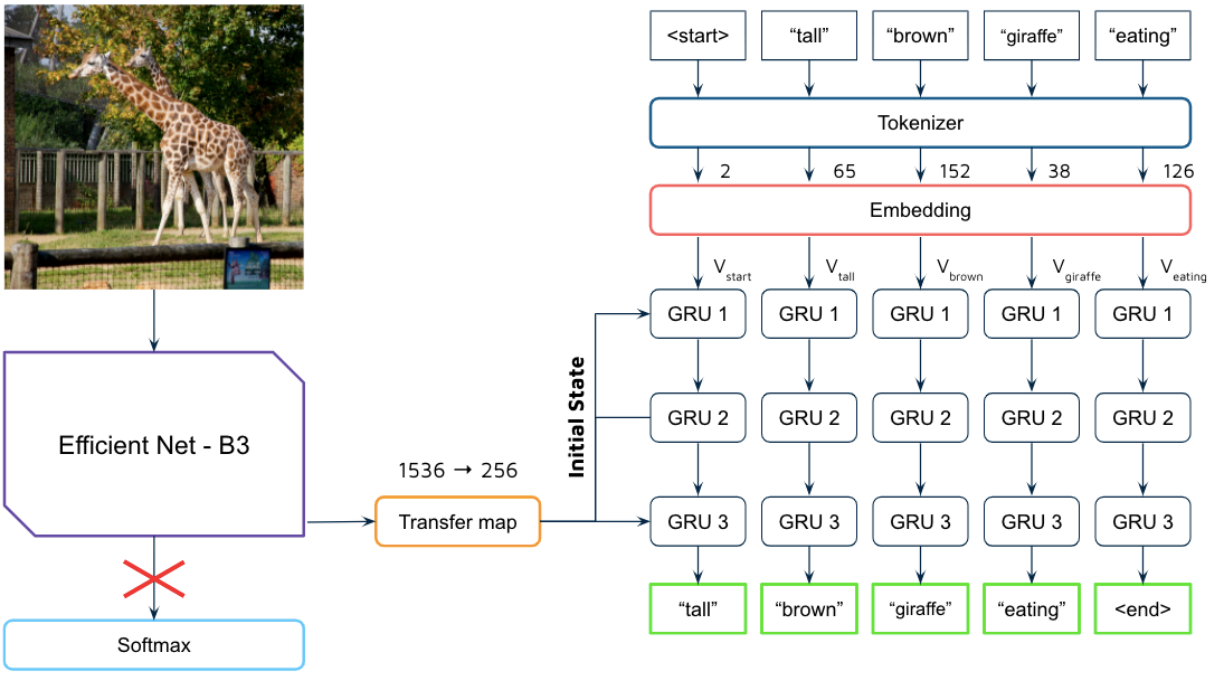


Figure 23 : Structure of our captioning module

3.3.5 Semantic Translator

The semantic translator is a simple module that helps us to convert a caption to its representative semantic vector. This conversion is an essential step because captions cannot be directly used by the classifier to predict class scores. Therefore, it acts as a secondary projection from the word embedding space to the semantic space. We loop through the attribute names in the order of semantic vector and identify the names present in captions. If the attribute is correctly

annotated, we represent it as 1 and 0 otherwise. Therefore, it allows us to preserve the structure of semantic space because changing the order of attributes would directly affect the dimensions of the generated vector.

3.3.6 Classification Module

The classification module consists of two main processes. We realize that directly obtaining the class scores is not enough because when generating the scores, all binary attributes are treated equally. Therefore, we use class penalties to reduce the score for classes that have extreme outliers. This approach makes it easy to divide the scores and obtain the correct class label. It is also a more realistic approach for a generalized setting where both seen and unseen labels are present during the testing phase.

$$score\left(y^{(i)}\right) = dist(W, x_u, P_j) \tag{5}$$

$$bias^{(i)} = bonus\left(pos, S_y^{(i)}\right) - penalty\left(neg, S_y^{(i)}\right) \tag{6}$$

$$Y_{pred}{}^{(i)} = \arg min_j \ (score(y) + bias) \tag{7}$$

3.4 Algorithms

Algorithm 1 The Image Captioning algorithm

Input: The total number of epochs N, training set S with n labeled samples of visual features V, and the word tokens T.

Output: Trained captioning network

for each epoch N do
    for each i do
        1. $W_e \leftarrow E(T_j)$;
        2. $Y_1 \leftarrow GRU1(V_i,\ W_e)$;
        3. $Y_2 \leftarrow GRU2(V_i,\ Y_1)$;
        4. $Y_3 \leftarrow GRU3(V_i,\ Y_2)$;
        5. $Y \leftarrow Softmax(Y_3)$;
    end for
    4. Compute the cross-entropy loss L_{CE};
    5. Minimize the overall loss;
end for

Algorithm 2 The attribute generation algorithm

Input: Seen and unseen visual features $\mathbf{X}s$, $\mathbf{X}u$
Semantic embedding features $\mathbf{Y}s$, A

Output: The predicted semantics $\mathbf{S}_G$

SAE weights [27]

1. $P \longleftarrow A * A^T$
2. $Q \longleftarrow \lambda * X * X^T$
3. $R \longleftarrow (1 + \lambda) * A * X^T$
4. $W \longleftarrow sylvester(P, Q, R)$

Generative module

1. $D \longleftarrow 1 - cosine(S_{pred}, A_{test})$
2. $C \longleftarrow argmin\ (D)$
3. $S_G \longleftarrow A[C]$
4. Assign $\mathbf{S}_G$ to test features as generated semantics to reduce hubness.

end for

3.5 Experimental Setup

The experiments are designed to meet this research's primary objectives: to define standard benchmarks, discuss an in-depth comparative analysis, and propose an attribute-directed extended cross-domain method. We conduct the experiments on AwA2 [5] dataset. Instead of using the standard splits originally provided by the datasets, we decided to use the splits proposed in [4] to prevent unintended bias towards seen classes. We propose to use an 80-20 split to keep our training and validation phase consistent among all the datasets. Then we evaluate several popular state-of-the-art models in zero-shot learning and compare their results on our proposed split. Therefore, the observations in the proposed benchmark might slightly differ from the original results because it will use a different subset of seen and unseen classes. The experiments will be carried out in a generalized setting. The difference between the two is that in the generalized case, testing data even includes a few training classes, representing a more realistic scenario. Based upon the results, we will conduct a comparative analysis to showcase the good and bad designs in the studied models. Our analysis would be a significant contribution, allowing everyone to quickly understand how such methods function and discuss the scope of future advancements. Finally, we will test our proposed model on the newly defined benchmark and publish the results in this paper and online. The proposed method will be tested on two sets of features: (1) Features provided in [4], and (2) Self-extracted features using our feature extraction module. This is to ensure that our results are reproducible, convincing, and comparable with other methods.

4 Methodology and Implementation

4.1 Dataset Statistics

After exploring many popular datasets that have been widely used among state-of-the-art zero-shot learning methods, we reviewed two coarse-grained, one medium and one small scale, and two fine-grained datasets, both of medium size. All of the datasets include attribute information for each image sample. As shown in Table 1, Animal with Attributes 2 (AwA2) [5] is medium in size compared to large-scale datasets like ImageNet. AwA2 has 30,475 image samples and 85 attributes. There are 40 seen animal classes for training a model and 10 unseen classes that are used for testing. AwA2 is a coarse-grained dataset in terms of details of the attributes presented. The second dataset is CUB-200-2011 Birds (CUB) [28], which is an extended version of the CUB-200 dataset by Caltech with fine attribute details. It contains 11,788 instances of bird images that are defined by 312 different attributes. The CUB dataset provides a standard split of 150 seen classes and 50 unseen classes for testing. We also review Attribute Pascal and Yahoo (aPY) [29], a coarse-grained dataset consisting of 15,339 images with 64 attributes. The dataset is originally split into 20 seen and 12 unseen classes. The fourth dataset, SUN [30], is a fine-grained dataset with 14,340 images of 717 types of scenes and 102 attributes. While it provides a good number of classes, the number of images per class is significantly reduced. There are 645 seen classes and 72 seen classes.

Out of these datasets, we decided to present our results for **AwA2** dataset as it is coarse-grained and the attribute clusters are at a good distance from one another. Moreover, the number of instances per class in AwA2 is more ideal to train our model compared to other datasets.

Dataset	**Details**	**Images**	**Semantic Space**	**Attributes**	$\mathcal{Y}_{train}$	$\mathcal{Y}_{test}$
AwA2 (selected)	coarse	37,322	A+W	85	40	10
CUB	fine	11,788	A	312	150	50
aPY	coarse	15,339	A	64	20	12
SUN	fine	14,340	A+W	102	645	72

Table 1 : Statistics of Benchmark Datasets

4.2 Evaluation Protocol

We need to define some evaluation protocols before carrying out any experiments. These protocols are essential to propose benchmarks described previously in the objectives. The evaluation protocols followed in this research are:

Image and Class Embedding: We apply image-preprocessing before extracting features of images for AwA2 dataset. The ResNet-101 model [31] pre-trained on ImageNet [32] dataset to extract visual features, which results in 2048-dimensional vectors, where $V \in \mathbb{R}^{N \times 2048}$. By dimensionality reduction, the dimensions of features are further reduced to 256 state vector with the help of a linear transfer layer. It allows the model to train faster while keeping the top level of accuracy. Our model uses a transductive approach to zero-shot learning which is a more realistic scenario. Furthermore, it is possible to extend any well-performing inductive model to transductive using our method by simply using it as a prototyping function in the generation module to provide synthesized attributes for the test images.

Dataset Split: In zero-shot learning, we assume that the training and testing classes are distinct, and no test class should be used in the training phase. Therefore, the datasets like ImageNet used to train DNN based feature extractors should not include those test classes. However, Xian et al. [4] pointed out that in the standard splits of the benchmark datasets, 6 AwA2 test classes out of 10, and 1 CUB test class out of 50 are present in the 1000 classes of the ImageNet dataset. That is why we decided on the proposed splits mentioned in [4] to prevent this unintended error in our results.

Evaluation Metric: Picking the right evaluation metric is crucial to understanding how a machine learning model performed overall and when compared to other similar methods. After analyzing various metrics available, we selected F1 score and Cross-Entropy Loss. While accuracy is the general way to qualify a model, the F1 score or the harmonic mean of precision and recall is an evaluation metric that punishes extreme values more. It is a specific case of the F_β score and is a perfect choice in zero-shot learning, where we need good accuracies for both training and testing.

$$F_1\ score = 2 \times \frac{precision \cdot recall}{precision + recall} \tag{8}$$

$$F_\beta\ score = (1 + \beta^2)\ \cdot\ \frac{precision\ \cdot\ recall}{\beta^2\ \cdot\ precision + recall} \tag{9}$$

To calculate the F_1 score, we utilize per class accuracy to make sure that model is performing good on every ZSL class. For example, without this metric, if one class consists of most testing samples then it might effect our results for other classes. Per class accuracy also allows us to remove any descrepancies because of the uneven number of samples in each class.

$$Accuracy = \frac{1}{|C|}\sum_{i=1}^{C} accuracy(c_i)$$

We will also propose to use the Cross-Entropy Loss as a secondary criterion to evaluate the models. It is based upon the idea of entropy from information theory and is one of the most popular ways to assess the performance of a classification problem.

$$L_{CE} = -\sum_{c=1}^{M} y_{o,c} \cdot \log(p_{o,c}) \tag{10}$$

4.3 Methods

4.3.1 Domain Adaptation

Learning a projection function for an unseen domain is crucial in zero-shot learning because the data of both seen and unseen classes are embedded in the same semantic space. The unsupervised subspace alignment technique is highly relevant among all the proposed approaches in domain adaptation, like covariate shift, self-labeling, and clustering. On the other hand, generative adversarial nets (GANs) use a discriminator to minimize the distance between visual features. However, subspaces spaces constructed by GAN models do not hold semantic meaning and cannot be used to draw relationships between the seen and unseen classes. Therefore, our model uses the generalizing power of natural language processing (NLP) using a captioning model.

4.3.2 Gated Recurrent Unit

Gated Recurrent Units (GRUs) was created to overcome the issue of short-term memories in recurrent neural networks. These cells use internal mechanisms called gates to regulate the flow of information passing through them. It is capable of learning which data in the sequence is essential to keep or forget. Many state-of-the-art results based on recurrent networks are using GRUs. It is a newer generation of networks and is pretty similar to the design of an LSTM. However, a GRU cell only has two gates, a reset gate, and an update gate. Therefore, GRUs use lesser tensor operations and train faster than LSTM. We tried both LSTM and GRU in our model before deciding to use GRU in the final implementation.. It is achieved by the following equations:

They are also applicable in speech recognition and text generation. The cell state and gates are the core components of a GRU, where the state acts as the memory of the network. Moreover, the gates contain sigmoid activations capable of learning relevant and unnecessary knowledge during training.

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z)$$

$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r)$$

$$\hat{h}_t = \phi_h(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h)$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h}_t$$

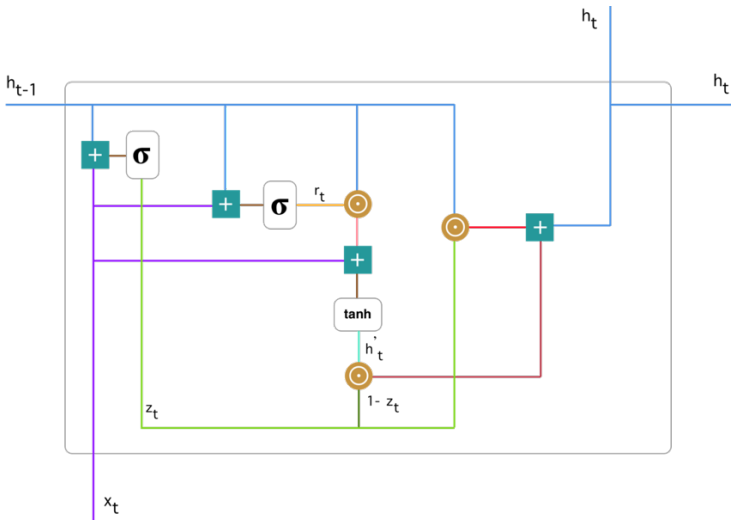


Figure 24 : Architecture of a GRU cell

4.4 Implementation

This section will explain the implementation used in our attribute-directed extended (ADE) zero-shot learning approach. To begin, the images provided in the AwA2 dataset are of different sizes, so images are resized to a standard shape of 224x224x3 to be able to feed it in our feature extractor. We also create a set of augmented images using the training data to artificially expand the available dataset. It is achieved with the help of an augment function that applies random flips, rotation, and translations to an image. The output of the augmentation is shown in Fig. 21. Hence, every augmented image provides a different perspective to the same class object to allow better generalization. With the normal and augmented data sets available, the images are fed to a ResNet-101 model with pre-trained weights and frozen layers. It provides us with more holistic features.

The obtained feature vectors consist of 2048 dimensions. On obtaining features for all the instances, we try to generate semantic vectors for testing images in an inductive approach using

linear transformations. This generation module is one of the creative aspects of the proposed model because the inductive semantics will be later used in a transductive manner to generalize the semantic space for both seen and unseen classes. We use semantic autoencoding (SAE) as proposed in [27] to create the weight matrix for attribute embeddings. This way is much faster than a generative GAN approach as it uses linear transformation and does not require any training. It is important to note that this weight matrix is biased and unsuitable for classification in a generalized scenario. It is only used to generate possible semantics for the testing sample.

Moreover, one extraordinary capability of our model is that the base method to generate test semantics can be replaced with any traditional method of choice, and the ADE model can extend it to a generalized setting. The generated semantic is a binary vector of 85 dimensions. Here, each dimension represents if the respective attribute is present or not.

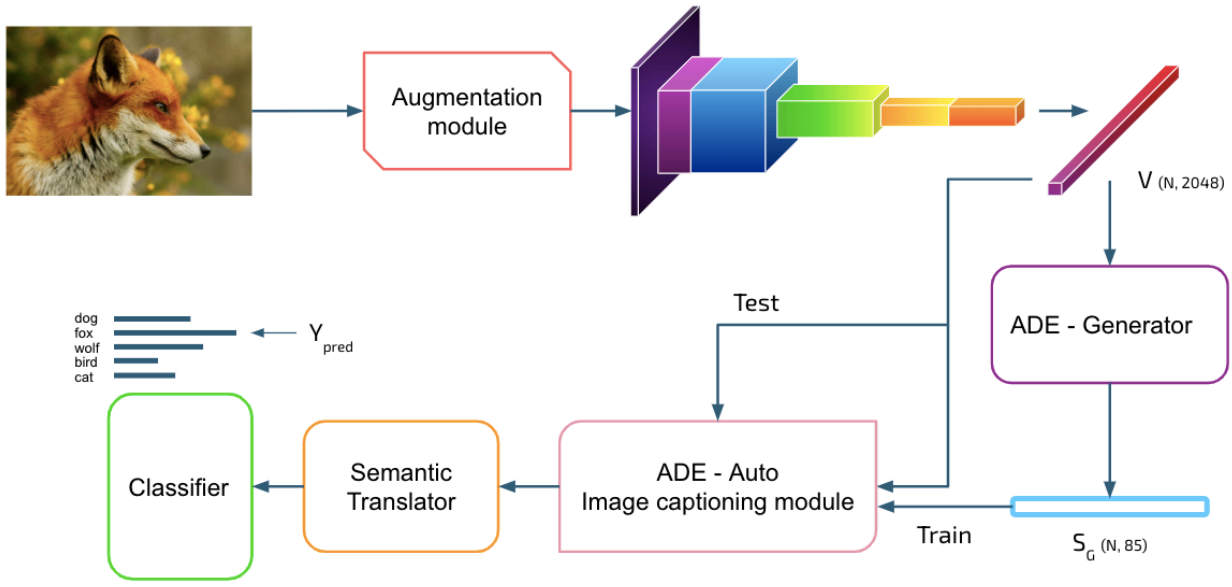


Figure 25 : ADE ZSL – System Design

As shown in the above figure, the artificial testing data is combined with training data to train the image captioning module. The captions are obtained by converting the binary attributes to a textual representation containing the names of the attributes present in the image. The design of the proposed captioning module has been described in Fig. 23. The captions are first converted to word tokens using a tokenizer that creates a vocabulary and uses the word to index format to replace each word with an integer token of that word's index in the vocabulary. We limit the vocabulary size to 87 words because we have 85 attributes and two extra words added with each token to denote the start and end of the sentence. The tokens are then embedded into a 256-dimensional vector using an embedding layer. Each of these vectors represents the respective word in a well-defined embedding space. Next, we feed the input features and the embeddings into the

GRU layers. The output of the previous GRU is passed on to the next GRU layer with the feature inputs to prevent any information loss during backpropagation. Thus, we use a multi-layered GRU approach to predict the captions.

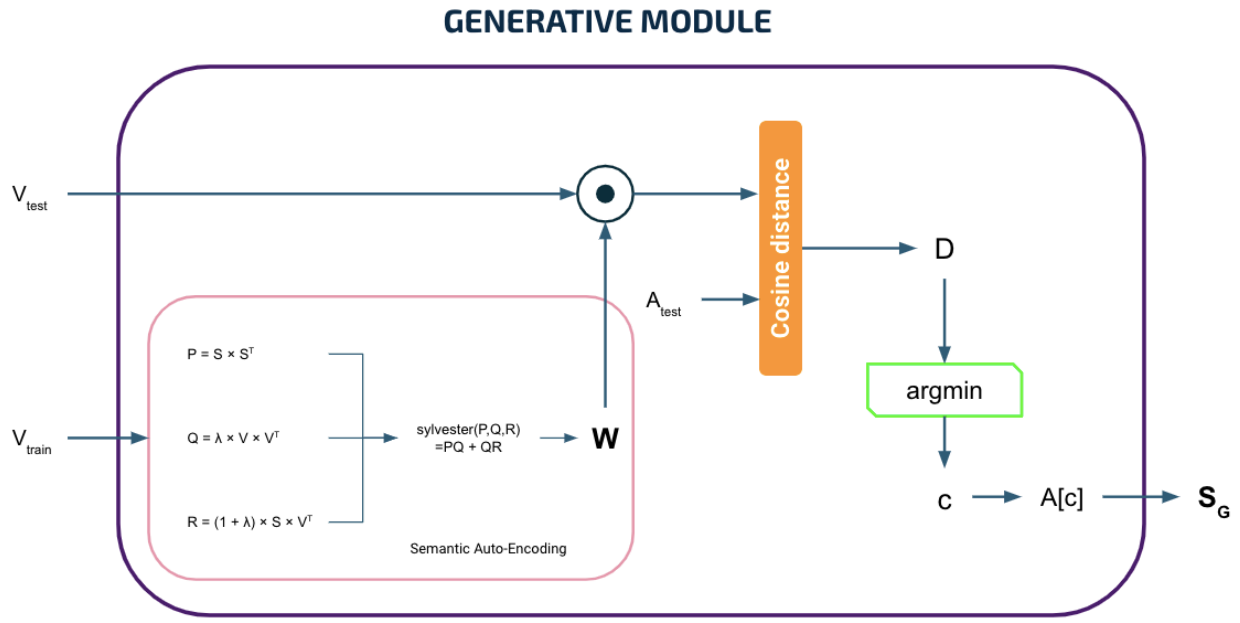


Figure 26 : Architecture of ADE-Generator

The predicted captions are then translated to binary attribute vectors using the semantic translator. We parse the predicted caption in the sequence of default attribute sequence and define one if the attribute is present in the caption or zero if it is absent. The semantics obtained via this strategy is much closer to the ground truth and thus provide better results. The reason behind the excellent performance of our implementation is that method like image captioning in natural language processing has been specially designed to perform well in a generalized setting using available patterns.

In the end, the predicted semantic vectors are passed to the classifier that calculates the class scores using matrix multiplication. The distinguishing factor of our classifier is that it selects positive and negative attributes using values from continuous attributes. It assigns a bonus if the positive attributes are present and a penalty if they are missing from the predicted vector. Similarly, it adds a bonus if a negative attribute is not present and gives a penalty if it is present in the semantic vector. It results in an Nx50 score matrix, and the class with the highest score is labeled as the predicted class. Our classification process uses the attribute space consisting of both training and testing classes, which is an essential requirement for generalized zero-shot learning.

The interactive web demo uses flask to connect with our trained python model and provide live predictions of captions and labels for uploaded images.

5 Results and Evaluation

Table 2 shows the results of our comparative analysis on the selected benchmark dataset: AwA2 [5] under the proposed splits (PS) [4]. The below models have been evaluated in a generalized setting, i.e., both seen and unseen classes are present in the final classification. From our observations, we clearly see that generative models outperform the traditional methods. However, our model is successfully able to extend the performance of SAE method when combined with the ADE approach as the harmonic accuracy (H) increases from 2.2% to 64.82%. This boost in performance is obtained by using the traditional method as a generator to our ADE-captioning that generalizes the embedding domain, thus, reducing the bias between seen and unseen classes.

Method	**AwA2**		
	S	U	H
Inductive			
SAE	82.2	1.1	2.2
ESZSL	77.8	5.9	11.0
SJE	73.9	8.0	14.4
SSE	82.6	8.1	14.8
SYNC	90.5	10.0	18.0
ALE	81.8	14.0	23.9
DEVISE	74.7	17.1	27.8
RelationNet	93.4	30.0	45.3
Transductive			
ALE-tran	73.0	12.6	21.5
DSRL	74.7	20.8	32.6
GDAN	67.5	32.1	43.5
Generative			
SE-GZSL	68.1	58.3	62.8
f-VAEGAN-D2	70.6	57.6	63.5
SAE+ADE **(Ours)** (Given features)	**98.27**	**48.36**	**64.82**
SAE+ADE **(Ours)** (Our features)	96.19	44.29	60.66

Table 2 : Classification accuracies

In table 3, we have provided the cross-entropy loss for both the models. It is seen that the $\mathcal{L}_{CLE}$ for testing phase is higher than that of the training loss. A similar pattern is observed for the accuracies where the testing accuracies (U) are much lower than the training accuracies (S). It is an acceptable outcome in zero-shot learning because unlike supervise or semi-supervised learning, models in zero-shot learning do not train on testing classes and solely rely on semantic relationships to predict a new class.

Method	**Cross-Entropy Loss**	
	$\mathcal{L}_{train}$	$\mathcal{L}_{test}$
ADE **(Ours)** (Given features)	3.18	3.44
ADE **(Ours)** (Our features)	3.19	3.49

Table 3 : Cross-Entropy Loss for classification

The figures below show the accuracy and loss of our models during testing. The plots show the high performance of the ADE-captioner on the training samples, proving that if we can obtain really good semantic vectors through our generator, it will directly enhance the classification accuracy of ADE model on unseen class images.

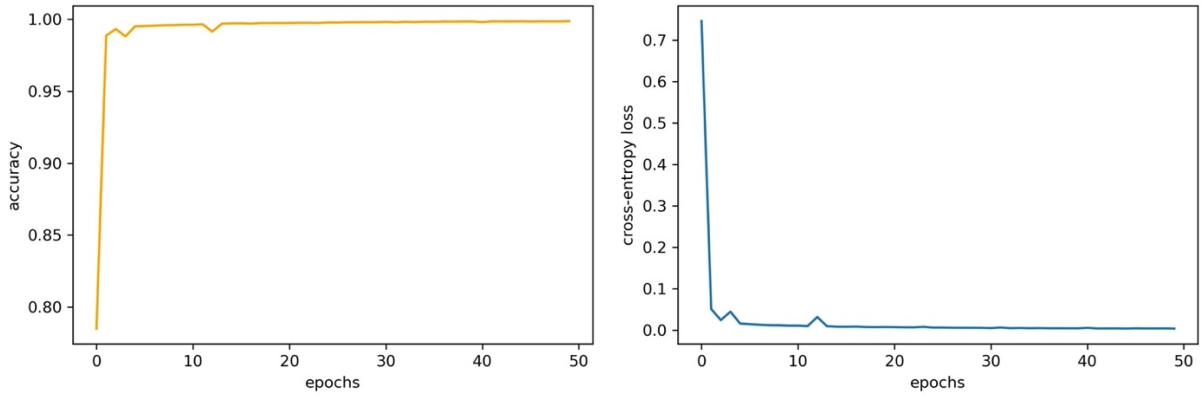


Figure 27 : Accuracy (left) and loss (right) curve of decoder for extracted features

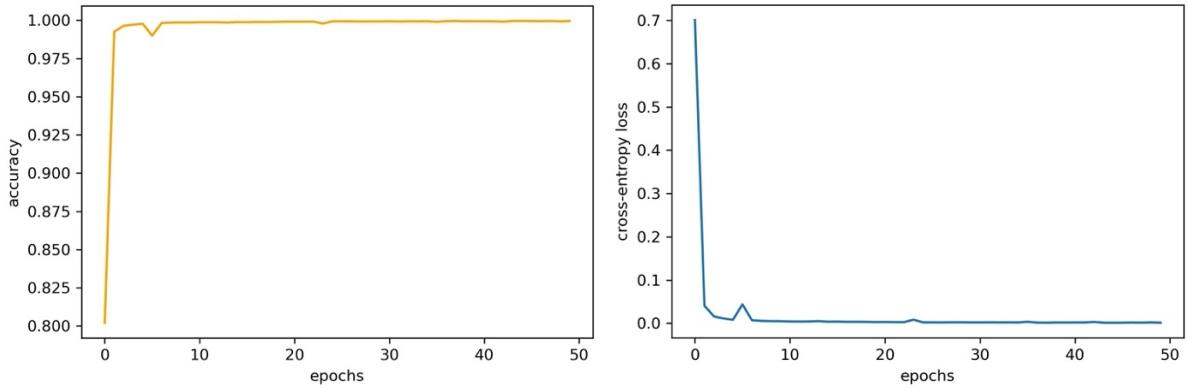


Figure 28 : Accuracy (left) and loss (right) curve of decoder for provided features

6 Challenges

In our preliminary analysis of benchmark datasets, we observed the difference between attribute distribution of fine-grained vs coarse datasets. As it can be seen in Fig. 29, the images attributes of fine-grained dataset such as CUB, lie very close to one another in embedding space, while in coarse dataset like AwA2, they are far apart. Therefore, it was determined that Caltech's CUB-200 requires additional preprocessing steps to form distinguishable clusters for applicable use in zero-shot learning.

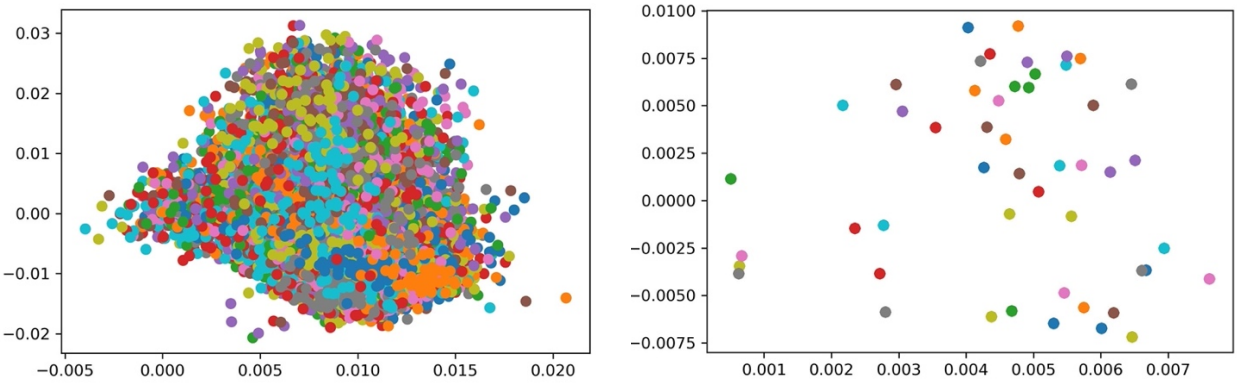


Figure 29 : Attribute cluster : CUB original attributes(left) and AwA2 attributes (right)

Furthermore, we observed that using binary attributes for the primary embedding of the model may generate poor results due to lack of relationship or importance of attributes. In binary notation, all attributes that are present are treated equally even though one attribute might be less important than the other. To overcome this issue, we used attribute temperatures (continuous attributes) provided with the dataset to introduce penalties in our class scores. This step showed a significant improvement in the final predictions.

7 Future Improvements

Zero-Shot Learning has come a long way since the first papers on its use in natural language processing and computer vision appeared in 2008. It is now possible to apply ZSL in real-world applications like autonomous vehicles, medical imaging, signal processing, and voice assistants with the help of advancements in deep neural networks. In this project, we exemplified some of the innovative ways to build a zero-shot learning method and compared it with state-of-the-art models. However, there is still a lot to be achieved in this domain. Some of the possible future improvements are:

Realistic Learning Strategy: After analyzing the existing models, we found that most methods have put the effort in a restrictive and unrealistic setting. The current approaches presume

that the unseen classes are known even if no training data is provided. However, in reality, it cannot be assumed that the new samples will come from a set of known unseen classes. Moreover, we want the model to learn from the unseen classes for future classification to perform well in a generalized setting.

Attribute term frequency for attention mechanism: This idea is based on the term frequency-inverse document frequency (TF-IDF) used in text analysis. Similarly, we believe that it might be possible to extract more relevant attributes to learn distinguishable embeddings for the semantic space. The limiting factor for this feature is that we require a large number of attributes to form the attribute term index. While this method has never been tested before, it appears to be a great improvement as an attention mechanism.

8 Conclusion

In conclusion, we designed and developed a new innovative approach to target zero-shot learning by using generated attributes in place of generated features as proposed in existing solutions. It allows us to significantly reduce the training time and hardware requirements to train a GAN. It is also observed that only using binary attributes might not be helpful in constructing a meaningful embedding space. Therefore, we propose to implement a classifier with penalties based on attribute weights to increase the distances between predicted class probabilities.

We believe that the interactive web demo will be able to demonstrate the results of the project and its applications in real-world scenarios like captioning images of unknown classes. As described in future works, using a realistic learning strategy where the model keeps learning from test data can increase the use cases in a variety of fields. Recently, interactive zero-shot learning models have been introduced in medical imaging. With the help of test-time optimizations, the doctors can interact to help the model improve results learned for future predictions. The web demo of our model aims to demonstrate a similar use case where the model can learn on the go.

Finally, zero-shot learning is a relatively new domain of research and is still advancing every day. Its enormous benefits in real-world applications like autonomous vehicles, medical imaging, action recognition, and gesture recognition, will keep pushing the boundaries of this field of study with more amazing innovations yet to come.

9 References

[1] K. Mannanuddin, S. Aluvala, Y. Sneha, E. Kumaraswamy, E. Sudarshan, and K. Mahender, "Confluence of Machine Learning with Edge Computing for IoT Accession," in Proc. IOP conference series. Materials Science and Engineering, 2020, vol. 981, no. 4. [Online]. Available: https://iopscience.iop.org/article/10.1088/1757-899X/981/4/042003/pdf

[2] C. Zhan, D. She, S. Zhao, M. M. Cheng, and J. Yang, "Zero-Shot Emotion Recognition via Affective Structural Embedding," in *Proc. 2019 IEEE/CVF Int. Conf. on Computer Vision (ICCV)*, 2019, pp. 1151–1160. [Online]. Available: https://openaccess.thecvf.com/content_ICCV_2019/papers/Zhan_Zero-Shot_Emotion_Recognition_via_Affective_Structural_Embedding_ICCV_2019_paper.pdf

[3] D. Strain, "8.7 Million: A New Estimate for All The Complex Species on Earth," Science (American Association for the Advancement of Science), vol. 333, no. 6046, pp. 1083–1083, 2011, doi: 10.1126/science.333.6046.1083.

[4] Y. Xian, C. H. Lampert, B. Schiele and Z. Akata, "Zero-Shot Learning—A Comprehensive Evaluation of the Good, the Bad and the Ugly," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 41, no. 9, pp. 2251-2265, 1 Sept. 2019, doi: 10.1109/TPAMI.2018.2857768.

[5] C. H. Lampert, H. Nickisch, and S. Harmeling, "Attribute-Based Classification for Zero-Shot Visual Object Categorization," IEEE transactions on pattern analysis and machine intelligence, vol. 36, no. 3, pp. 453–465, 2014, doi: 10.1109/TPAMI.2013.140.

[6] M. Sajjad, N. Ejaz, I. Mehmood, and S. W. Baik, "Digital image super-resolution using adaptive interpolation based on Gaussian function," Multimedia tools and appl., vol. 74, no. 20, pp. 8961–8977, 2013, doi: 10.1007/s11042-013-1570-1.

[7] V. Grau, Alcañiz, Juan, Monserrat, and Knoll, "Automatic Localization of Cephalometric Landmarks," J. of biomedical inform., vol. 34, no. 3, pp. 146–156, 2001, doi: 10.1006/jbin.2001.1014.

[8] P. Shivakumara, S. Noushat, and G. Hemantha Kumar, "New Filter Based Unsupervised Rules for Boolean Blur Metric," in Proc. 2007 Int. Conf. on Computing: Theory and Applications (ICCTA'07), 2007, pp. 611–617. [Online]. Available: https://ieeexplore.ieee.org/document/4127438

[9] T. Celik, "Unsupervised Change Detection in Satellite Images Using Principal Component Analysis and k-Means Clustering," IEEE geoscience and remote sensing letters, vol. 6, no. 4, pp. 772–776, 2009, doi: 10.1109/LGRS.2009.2025059.

[10] M. Hayashida, H. Koyano, and T. Akutsu, “Measuring the similarity of protein structures using image local feature descriptors SIFT and SURF,” in Proc. 2014 8th Int. Conf. on Systems Biology (ISB), 2014, pp. 164–168. [Online]. Available: https://ieeexplore.ieee.org/document/6990750

[11] X. Xie et al., “A study on fast SIFT image mosaic algorithm based on compressed sensing and wavelet transform,” J. of ambient intelligence and humanized computing, vol. 6, no. 6, pp. 835–843, 2015, doi: 10.1007/s12652-015-0319-2.

[12] D. Phillips, A. Pooransingh, and S. Guven, “ORB-Based Multiple Fixed Resolution Approach for On-Board Visual Recognition,” in Proc. Artif. Intell. and Mobile Services – AIMS 2019, 2019, pp. 54–71. [Online]. Available: https://link.springer.com/chapter/10.1007%2F978-3-030-23367-9_5

[13] C. C. Yeh, Y. L. Chang, P. H. Hsu, and C. H. Hsien, “GPU Acceleration of UAV image splicing using oriented fast and rotated brief combined with PCA,” in *Proc. IGARSS 2018-2018 IEEE Int. Geoscience and Remote Sensing Symposium*, Jul. 2018, pp. 5700-5703. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8519046

[14] M. Jogin, Mohana, M. S. Madhulika, G. D. Divya, R. K. Meghana and S. Apoorva, "Feature Extraction using Convolution Neural Networks (CNN) and Deep Learning," in *Proc. 2018 3rd IEEE Int. Conf. on Recent Trends in Electron., Inf. & Commun. Technol. (RTEICT),* 2018, pp. 2319-2323. [Online]. Available: https://ieeexplore.ieee.org/document/9012507

[15] B. Romera-Paredes and P. Torr, “An embarrassingly simple approach to zero-shot learning,” in *Proc. Int. Conf. on machine learning*, 2015, vol. 37, pp. 2152-2161. [Online]. Available: http://proceedings.mlr.press/v37/romera-paredes15.pdf

[16] E. Kodirov, T. Xiang, Z. Fu, and S. Gong, “Unsupervised domain adaptation for zero-shot learning,” in *Proc. IEEE Int. Conf. on computer vision*, 2015, pp. 2452-2460. [Online]. Available: https://www.cv-foundation.org/openaccess/content_iccv_2015/papers/Kodirov_Unsupervised_Domain_Adaptation_ICCV_2015_paper.pdf.

[17] Z. Zhang and V. Saligrama, “Zero-shot learning via joint latent similarity embedding,” in *Proc. IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2016, 6034-6042. [Online]. Available: https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Zhang_Zero-Shot_Learning_via_CVPR_2016_paper.pdf.

[18] Y. Shigeto, I. Suzuki, K. Hara, M. Shimbo, and Y. Matsumoto, “Ridge Regression, Hubness, and Zero-Shot Learning,” in *Machine Learning and Knowledge Discovery in Databases*, Cham: Springer International Publishing, 2015, pp. 145-151.

[19] A. Zhao, M. Ding, J. Guan, Z. Lu, T. Xiang, and J.-R. Wen, “Domain-Invariant Projection Learning for Zero-Shot Recognition,” *Advances in Neural Inf. Process. Syst.*, 2018, pp. 1027-1038.

[20] L. Chen, H. Zhang, J. Xiao, W. Liu, and S.-F. Chang, “Zero-Shot Visual Recognition using Semantics-Preserving Adversarial Embedding Networks,” 2018, pp. 1043-1052.

[21] Y. Jiao, J. Hare, and A. Prügel-Bennett, “What Remains of Visual Semantic Embeddings,” 2021.

[22] R. Ramachandran, J. Zhang, M. Maskey, and T. Lee, “Riding the Type Wave: Evaluating New AI Techniques for Their Applicability in Earth Science,” 2016.

[23] M. Ding, Z. Wang, and Z. Lu, “Cross-domain mapping learning for transductive zero-shot learning,” *Computer vision and image understanding*, vol. 187, p. 102784, 2019, doi: 10.1016/j.cviu.2019.07.004.

[24] X. Yu, “Emerging Applications of Generative Adversarial Networks,” *IOP conference series. Materials Science and Engineering*, vol. 740, no. 1, p. 12132, 2020, doi: 10.1088/1757-899X/740/1/012132.

[25] M. Hossain, F. Sohel, M. Shiratuddin, and H. Laga, “A Comprehensive Survey of Deep Learning for Image Captioning,” *ACM computing surveys*, vol. 51, no. 6, pp. 1–36, 2019, doi: 10.1145/3295748.

[26] C. Yuan, Z. Bin, and M. Bing, “An Approach to Labeling Audio Tags Based on Self-Attention Generative Adversarial Networks,” in *Proceedings of the 2019 2nd International Conference on algorithms, computing and artificial intelligence*, 2019, pp. 297–302, doi: 10.1145/3377713.3377772.

[27] E. Kodirov, T. Xiang and S. Gong, "Semantic Autoencoder for Zero-Shot Learning," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, 2017 pp. 4447-4456.

[28] C. Wah, S.Branson, P. Welinder, P. Perona, and S. Belongie, “The caltech- ucsd birds-200-2011 dataset,” California Inst. Technol., Pasadena, CA, USA, Tech. Rep. CNS-TR2011-001, 2011.

[29] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva and A. Torralba, "SUN database: Large-scale scene recognition from abbey to zoo", *Proc. IEEE Conf. CVPR*, pp. 3485-3492, Jun. 2010.

[30] M. Sajjad, N. Ejaz, I. Mehmood, and S. W. Baik, “Digital image super-resolution using adaptive interpolation based on Gaussian function,” *Multimedia tools and appl.*, vol. 74, no. 20, pp. 8961–8977, 2013, doi: 10.1007/s11042-013-1570-1.

[31] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.

[32] J. Deng, W. Dong, R. Socher, L. -J. Li, Kai Li and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248-255, doi: 10.1109/CVPR.2009.5206848.

10 Appendices

Appendix I. Project Schedule

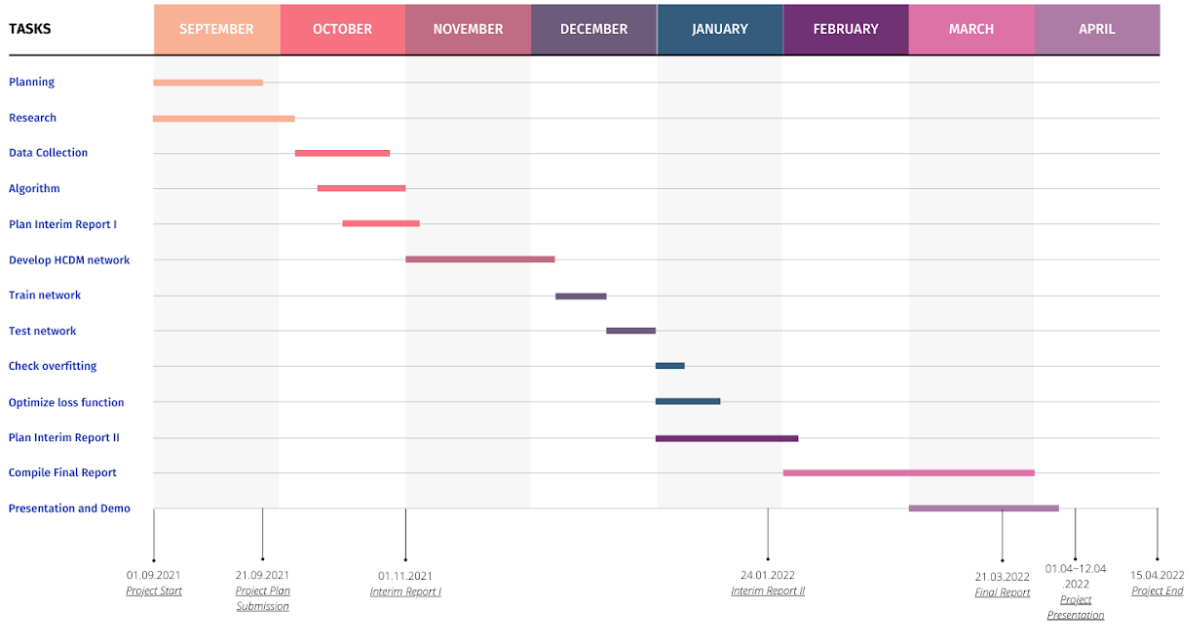


Figure 30 : Project Schedule

Appendix II. Code snapshots

Augmentation module

```
def augment_func(image_path):
    image = tf.io.read_file(image_path)
    image = tf.io.decode_jpeg(image, channels=3)
    image = tf.keras.layers.Resizing(img_size, img_size)(image)
    image = tf.keras.Sequential([
                tf.keras.layers.RandomFlip(),
                tf.keras.layers.RandomRotation(0.2),
                tf.keras.layers.RandomTranslation(0.2, 0.2)
            ])(image)
    return image
```

Generator

```
P = S*S.T
Q = lam*X*X.T
R = (1+lam)*S*X.T
W = solve_sylvester(P,Q,R)

W = normalize(W)

test_pre_attribute =  testfeatures.dot(W.T)
test_attributetable = normalize(test_attributetable.T).T

dist = 1-cosine_similarity(test_pre_attribute,test_attributetable)

lis = np.expand_dims(np.unique(test_classes), 1)
```

Appendix III. Table of Figures

Appendix IV. Milestones

Stage 1 : Research Review

Stage 2 : Data Collection and Processing

Stage 3 : Framework Design

Stage 4 : Model Architecture and Development

Stage 5 : Testing and Evaluation

Stage 6 : Optimization

Appendix V. Monthly Log

October, 2021

1) Stage 1 and 2
2) Review existing solutions of Zero-Shot Learning
3) Discuss project objectives and scope
4) Pick benchmarks
5) Determine hardware and software requirements

November, 2021

1) Stage 2
2) Collect essential datasets
3) Data Preprocessing

December, 2021

1) Stage 3
2) **Data processing** - Understand features, try the semantic extraction methods stated in the literature review, and study ways to improve the features obtained
3) **Framework** – Design system framework
4) **Environment setup** - Prepare the system environment by installing required python packages for running models. Install CUDA integration for GPU support.
5) **Testing** - Run and observe the performance of existing ZSL solutions.

January, 2022

1) Stage 3 and 4
2) **Framework** – Design system framework

3) **Comparative Analysis** – Analyze the working of state-of-the-art models in zero-shot learning
4) **Implementation** – Started developing the proposed model architecture using attention-based image captioning

February, 2022

1) Stage 4 and 5
2) **Architecture** – Tested multiple strategies to finalize the design of our model
3) **Implementation** – Used the final design to program our method pipelines

March, 2022

1) Stage 5 and 6
2) **Development** – Finished developing the model in the first week
3) **Testing and Evaluation** – Tested the method using the specified evaluation metrics
4) **Optimizations** – On comparing the results with existing methods, we optimized our model a few times to generate the published results