Paper Title* (use style: *paper title*)

*Note: Sub-titles are not captured in Xplore and should not be used

line 1: 1st Given Name Surname
line 2: *dept. name of organization (of Affiliation)*
line 3: *name of organization (of Affiliation)*
line 4: City, Country
line 5: email address or ORCID

line 1: 2nd Given Name Surname
line 2: *dept. name of organization (of Affiliation)*
line 3: *name of organization (of Affiliation)*
line 4: City, Country
line 5: email address or ORCID

line 1: 3rd Given Name Surname
line 2: *dept. name of organization (of Affiliation)*
line 3: *name of organization (of Affiliation)*
line 4: City, Country
line 5: email address or ORCID

line 1: 4th Given Name Surname
line 2: *dept. name of organization (of Affiliation)*
line 3: *name of organization (of Affiliation)*
line 4: City, Country
line 5: email address or ORCID

line 1: 5th Given Name Surname
line 2: *dept. name of organization (of Affiliation)*
line 3: *name of organization (of Affiliation)*
line 4: City, Country
line 5: email address or ORCID

line 1: 6th Given Name Surname
line 2: *dept. name of organization (of Affiliation)*
line 3: *name of organization (of Affiliation)*
line 4: City, Country
line 5: email address or ORCID

***Abstract*— In the realm of machine learning, zero-shot learning (ZSL) stands as a paradigm that enables the classification of images for categories not encountered during training, without any associated training data. This paper applies advanced ZSL techniques that leverage the power of convolutional neural networks (CNNs) and ResNet-50 models. Additionally, it introduces a novel hierarchical ZSL framework featuring both primary and secondary classification tiers. The primary classifier utilizes a CNN architecture, while two ResNet-50-based ZSL classifiers are tailored to specific subclasses. This innovative approach not only enhances accuracy but also reduces computational complexity. The paper also delves into a comprehensive analysis of factors affecting time complexity in CNN-based image classification, encompassing metrics such as real multiplications, bit operations, shift and add operations, and hardware logic gates. The proposed ZSL strategies demonstrate remarkable proficiency in classifying image categories never encountered before and achieve significantly improved accuracy levels in zero-shot scenarios.**

Keywords—word embedding, CNN, ResNet50 Zero-shot learning, visual semantic embedding, word embedding, CNN, ResNet50.

I. Introduction

Conventional classification techniques have achieved significant success across various domains. However, they are not without limitations. These methods heavily rely on the availability of a substantial amount of labeled training data for each class, which poses practical challenges. Moreover, these models are confined to recognizing only the categories that they have been trained on, making them incapable of handling unseen classes. This limitation stems from the difficulties associated with obtaining labeled data, whether due to the high costs of annotation or the scarcity of data for certain categories.

Zero-shot learning (ZSL) offers a solution to these challenges by enabling the recognition of unlabeled data without prior training in those specific classes. In ZSL, a model is initially trained on a set of "seen" classes and is then tested on its ability to categorize "unseen" classes. To bridge the gap between these seen and unseen classes, ZSL relies on semantic descriptions, such as attributes or natural language sentences, to capture the essential characteristics of each class. During training, the model learns to associate these semantic features with visual features extracted from the data. During prediction, the model maps unlabeled images to their semantic features and makes predictions based on the similarity between the predicted and existing class features, often computed using Euclidean distance.

This paper introduces an enhanced ZSL approach that leverages deep visual semantics. The model architecture involves the use of Convolutional Neural Networks (CNNs) and ResNet 50 for visual modeling, while FastText is employed for language modeling. This novel approach demonstrates the capacity to classify unknown image categories effectively. The proposed model's performance is evaluated on standard datasets, specifically SUN and AWA2, with an emphasis on assessing per-class average accuracy, which provides valuable insights into its effectiveness.

Furthermore, this paper presents a hierarchical ZSL framework that offers improved accuracy while reducing time complexity. This hierarchical approach comprises both primary and secondary levels of classification. The primary classification relies on a simplified CNN-based model trained on a reduced set of classes. In contrast, secondary classification employs two ResNet 50-based ZSL classifiers, each specialized for specific subclasses. This hierarchical strategy aims to enhance the accuracy of ZSL models and streamline computational demands. The proposed approach is rigorously tested for zero-shot classes on the standard datasets SUN and AWA2, with evaluations based on per-class accuracy and computational efficiency, providing a comprehensive assessment of its effectiveness in addressing ZSL challenges.

II. Literature Review

Image classification is one of the most demanding applications of complexity vision. Zero-shot learning is a kind of classification of images that the model has not seen. A lot of research has been done in the study of Zero-shot learning.

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE

In 2009 Lampert [4] et al. proposed attribute-based classification for unseen classes. Attributes are manually made features for groups of classes. Attributes like feather type, body structure, animal habitat, etc., are used as auxiliary information for unseen categories that are not available in training. Attribute-based zero-shot image classification was introduced in [5]. This method uses direct attribute prediction (DAP) and indirect attribute prediction (IAP), which are probabilistic classifiers. Attribute label embedding [ALE] suggested in [6] works better than DAP.

Frome [7] has developed deep visual semantic embedding (DeViSE), which extracts visual features with a convolutional neural network and semantic features using a skip-gram language model. The trained model is checked for its prediction using the Hinge loss function. This model can be used for larger dictionaries and trained on massively bigger text corpora can improve the quality of prediction.

The Deep Weighted Attribute Prediction method [8] uses a deep neural network for feature extraction and class prediction. And it does not require hand-crafted specific features. Models perform better with more accurate attribute prediction. SRC (Sparse representation coefficient) uses weighted attributes for prediction, improving classification accuracy.

The author in [9],[10],[11] have developed generative based methods. The main building blocks of these methods are Generative adversarial network and conditional Variational autoencoder which makes the system relatively complicated. These methods yield better accuracy leading to better ZSL classification.

The hybrid Feature model [12] uses a conditional autoencoder conditioned on semantic space. This method uses two autoencoders; one encoder uses visual and semantic information, and another autoencoder is provided with only visual information.

The proposed methodology of the Hierarchical approach of zero-shot learning emphasizes the computational complexity of models used for Zero-shot learning. The literature on computational and time complexity of models is as follows.

RICH LEE [13] has shown various factors in CNN image classification that affect the time complexities. It depends on different layers of CNN, depth of CNN, number of epochs, and model optimizer. Various models are evaluated for accuracy, loss, training time, training, and validation accuracy.

Pedro J. [14] has provided a systematic approach for evaluating and contrasting the computational complexity of neural network layers in digital signal processing. This paper explains how to calculate four metrics of computational complexity for feedforward and recurrent neural networks. The four metrics are Number of Real Multiplications, the Number of Bit Operations, the Number of Shift and Add Operations, and the Number of Hardware Logic Gates.

Number of shift and Add operations is a newly introduced metric that describes the bit width along with quantization used in the arithmetical operations.

The main idea of [15] is to evaluate the time complexity of CNN models. This work has identified the factors that affect the model performance, and the time taken by each layer to run. Time complexity analysis has been done on eight different models by changing the parameters such as size of filters, depth of CNN model, number of filters, number of fully connected layers, and kernel size.

The literature for zero-shot learning shows that the implemented approach with good accuracy is quite complicated. The accuracy of the existing method can be further improved with reduced complexity. The proposed Hierarchical approach of zero-shot learning is based on the deep visual semantic embedding [5] method and is evaluated for accuracy, computational, and time complexity.

III. Aim Of Study

The primary aim of this study is to develop an efficient approach with an optimized design architecture for zero-shot learning image classification. This approach is intended to effectively utilize both the training data and supporting information to create a robust classification model for the testing classes. In traditional classification approaches, the requirement for labeled training examples for each class poses a significant limitation. However, zero-shot learning (ZSL) aims to overcome this constraint by enabling the model to recognize and classify unseen categories, a critical capability in practical scenarios where labeled data for all possible classes may be unavailable. Therefore, the primary objective of this research is to devise a ZSL framework that leverages semantic features and visual information, as well as state-of-the-art deep learning models like CNN and ResNet 50, to classify unknown image categories accurately.

In addition to the development of an efficient ZSL approach, this study also seeks to identify and address the key challenges and issues inherent in classification using zero-shot learning methods. These challenges may include the need to effectively bridge the gap between seen and unseen classes, the selection of relevant attributes or semantic features, and the optimization of model parameters to ensure accurate and reliable classification. By thoroughly examining these challenges, this research aims to contribute valuable insights into the practical implementation of ZSL techniques and provide solutions to mitigate potential obstacles.

Furthermore, an essential goal of this study is to enhance the performance of the ZSL model while simultaneously reducing its complexity. Achieving higher accuracy in zero-shot image classification is crucial for its practical utility, and this research aims to propose a hierarchical approach that combines primary and secondary levels of classification. By utilizing simplified CNN-based classifiers for primary classification and more advanced ResNet 50-based ZSL classifiers for secondary classification, the study aims to strike a balance between accuracy and computational efficiency. Through rigorous evaluation on standard datasets, such as SUN and AWA2, the proposed hierarchical approach will be assessed based on per-class accuracy and time complexity metrics, thereby providing insights into its effectiveness in improving ZSL performance and reducing computational burden. In summary, the overarching aim of this research is to advance the field of zero-shot learning image classification by developing an efficient and optimized approach, addressing

inherent challenges, and achieving improved performance with reduced complexity. This study seeks to contribute to the broader goal of enabling machines to recognize and categorize objects and entities not encountered during training, making it valuable in a wide range of real-world applications.

IV. Objective of Study

The primary objective of this research is to design and develop an approach that significantly enhances the performance of zero-shot image classification. In conventional image classification, models are typically trained on labeled data representing known classes. However, zero-shot learning (ZSL) challenges this paradigm by enabling the classification of unseen classes, for which no labeled training examples are available. To address this challenge, this study aims to introduce innovative techniques and strategies that improve the accuracy and reliability of ZSL methods. By leveraging semantic features, deep neural networks, and other advanced tools, the objective is to create a ZSL approach that excels in recognizing and categorizing previously unseen classes.

The second objective is to propose an architecture that not only enhances performance but also reduces the complexity of the classification model. Complexity in machine learning models can lead to increased computational requirements and longer processing times. Hence, this research seeks to design an architecture that strikes a balance between model accuracy and computational efficiency. By employing a hierarchical approach that combines simplified primary classification models with more sophisticated secondary models, the aim is to optimize the overall architecture, resulting in faster inference times and improved resource utilization.

The final objective involves a comprehensive evaluation of the proposed hierarchical approach against an enhanced zero-shot learning method utilizing the deep neural network ResNet 50. This evaluation will include a comparison of the proposed architecture's performance with existing methods in the field of zero-shot image classification. To ensure a robust assessment, standard datasets will be employed as benchmarks. Through rigorous testing and evaluation of datasets such as SUN and AWA2, the research aims to provide empirical evidence of the effectiveness of the proposed approach. This comparative analysis will shed light on how the proposed architecture outperforms or complements existing methods and its potential impact on the field of zero-shot image classification.

V. Methodology

Zero-shot learning has garnered increasing attention, driven by humans' remarkable ability to recognize and classify new visual classes they've never encountered before. In the Zero-Shot Learning (ZSL) approach, a labeled dataset is employed to train a model on seen classes, with the primary objective being the model's capability to identify and classify new classes that it has never encountered previously. ZSL significantly reduces annotation costs by leveraging existing knowledge. The typical phases in zero-shot learning encompass visual feature extraction, semantic representation, and visual-semantic mapping.

A. Visual Features Extraction

Visual feature extraction plays a pivotal role in tackling complex vision tasks. Employing effective methods for visual feature extraction is of paramount importance. Visual features encompass a range of attributes including color, texture, and shape, among others. Traditional feature extraction methods, such as Hue, Saturation Value, Histogram of Oriented Gradient (HOG), and gray-level co-occurrence matrix, have been widely utilized in this context. However, it's noteworthy that deep learning models have demonstrated significant advancements over traditional feature extraction methods. Presently, deep learning models such as VGG, GoogleNet, and ResNet have gained prominence and are commonly employed for feature extraction in complex visual tasks.

B. Semantic Representation

Zero-shot learning (ZSL) is a technique designed for recognizing previously unseen data based on knowledge acquired from seen data. In ZSL, the model is trained using labeled data from seen classes and subsequently tested on unseen classes, which were not available during the training phase. The challenge with traditional methods lies in their inability to effectively transfer knowledge acquired during training to the identification of these unseen classes.

To address this challenge, ZSL leverages additional information, referred to as a semantic feature space. These semantic features serve as a bridge between the known seen classes and the previously unseen ones. They can be attribute-based or word vector-based, and their role is to establish meaningful connections and relationships between the two categories, facilitating the recognition of previously unseen classes.

1. Attribute Space: In the attribute space, a set of human-understandable characteristics is defined to represent various properties of different classes. These characteristics, known as attributes, are expressed as words or sentences describing specific properties of the classes. For instance, attributes for animals can encompass properties like body color, habitat, and visual features. For example, attributes for a zebra might include "stripes" and "forest-dwelling," which are features that help describe and distinguish the zebra. These attributes are commonly referred to as semantic features, and they serve as shared characteristics found in some animals. Consequently, they can be effectively used to identify and classify previously unseen classes in zero-shot learning scenarios.

2. Word Vector Based: Natural language processing techniques often employ word embedding vectors generated using Word2Vec. Word2Vec is among the most widely adopted approaches for comprehending word embeddings through the use of shallow neural networks. It consists of two fundamental architectures, namely the continuous bag-of-words (CBOW) and skip-gram models, which are utilized to create Word2Vec vectors [16]. These Word2Vec vectors can be acquired through two distinct methods.

a. Continuous Bag of Words(CBOW): In CBOW, the context surrounding a word within a sentence is employed to predict the word positioned at the sentence's center. This

method is notable for its efficiency in training, as it typically outpaces the training speed of its counterpart, the skip-gram model. Moreover, CBOW exhibits a penchant for delivering superior accuracy when dealing with frequent words, making it a valuable choice in natural language processing tasks where the prevalence of certain words necessitates precise and efficient modeling.

b. Skip Gram Model: In this method, the input word serves as the basis for predicting the context words that surround it within a given sentence or text. Skip-Gram is especially renowned for its proficiency in effectively representing rare or infrequent words, a crucial asset in natural language processing tasks where understanding and leveraging the subtleties of less common terms are essential. Additionally, this method demonstrates a remarkable ability to deliver robust accuracy even when working with limited amounts of training data, making it a valuable tool for various applications in the field of language modeling and text analysis.

C. Visual Semantic Mapping

Visual features are aligned with their corresponding semantic features through a process known as visual-semantic mapping. This mapping is crucial for facilitating the prediction of unseen classes, whether it occurs in the visual space, semantic space, or an intermediate space. In fact, the entire landscape of Zero-Shot Learning (ZSL) methods can be categorized into three distinct mapping frameworks, as illustrated in Figure 5.1. These frameworks play a pivotal role in bridging the gap between visual and semantic information, enabling the model to generalize and recognize previously unseen classes.

1. Forward Mapping: Forward mapping represents a mapping approach wherein visual features are paired with their corresponding semantic features, ultimately leading to the recognition of unfamiliar or unknown classes within the semantic space.

2. Common Mapping: Image features and semantic features are harmoniously integrated into a shared common space, where the task of recognizing unseen classes is effectively carried out. This common space acts as the pivotal bridge facilitating the identification of classes that were previously unfamiliar to the model.

3. Reverse Mapping: Reverse mapping is a mapping strategy that involves the translation of semantic features into the visual space, ultimately leading to the recognition of previously unencountered classes within the visual domain.

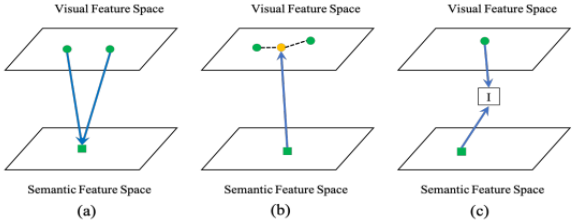


Fig 5.1 Three learning frameworks of zero-shot learning: (a) Forward Mapping (b) Reverse Mapping (c) Common mapping [17]

D. DeVISE using Deep Neural Network ResNet50

DeVISE using Deep neural network ResNet 50 is designed for the image classification of unseen data, specifically zero-shot classes. This approach combines two distinct models: a visual model and a language model. Operating under the forward mapping approach of visual-semantic mapping, DeVISE effectively bridges the gap between visual features and the semantic space, facilitating the recognition and classification of previously unencountered classes.

1. Visual Model: Visual models play a critical role in extracting essential visual features through the utilization of deep Convolutional Neural Networks (CNN). Within the vast array of available models in the literature, including notable ones like AlexNet, VGG19, GoogLeNet, and ResNet50, this approach employs both a fundamental CNN model and the ResNet50 model for the extraction of visual features from images. It's worth noting that both of these models are pre-trained on the extensive ImageNet dataset, harnessing the knowledge learned from this rich source of visual information.

a. Convolutional Neural Network: The Convolutional Neural Network (CNN) stands as a highly pervasive deep learning architecture, frequently harnessed for various tasks. At its core, CNN relies on matrix-based linear operations, with the initial layer often taking the form of a convolutional layer. This layer's primary role is to extract diverse image features through the use of filters. Subsequently, the pooling layer follows to reduce the network's dimensionality. After pooling, a fully connected layer connects all neurons from the preceding layer. The utility of CNN spans a wide range of applications in computer vision, encompassing tasks like image classification and object detection. Furthermore, CNN has found extensive application in the domain of natural language processing [18].

b. ResNet 50: ResNet 50 represents a Deep Convolutional Neural Network architecture specifically designed for the extraction of low, mid, and high-level features from images. While deep convolutional networks are effective for achieving high accuracy, they often encounter challenges related to gradient issues, including vanishing and exploding gradients. These challenges can result in slow convergence or oscillations during training. One effective solution employed in the ResNet 50 model to mitigate these gradient problems is the use of skip connections.

The ResNet 50 architecture, depicted in Figure 5.2, comprises numerous convolutional layers, pooling layers, and a fully connected layer, totaling 50 layers in all. These models are constructed with multiple convolutional layers and max pooling operations to extract intricate features from images. The distinctive feature of the ResNet 50 architecture lies in its incorporation of skip connections. These skip connections bypass certain layers within the ResNet 50

structure, enabling the output of one layer to serve as input for subsequent layers. This innovative approach significantly alleviates the challenges associated with vanishing gradients, resulting in more stable and efficient training.

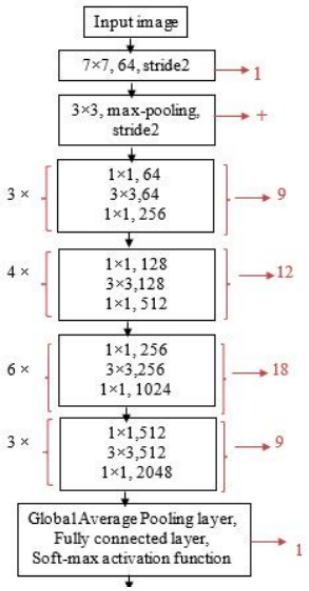


Fig 5.2 ResNet 50 Architecture

2. Language Model: Language models serve as a crucial tool for transforming words into word vectors, a task effectively accomplished using the Word2Vec method. Word2Vec employs a deep neural network model to comprehend word similarities, drawing insights from extensive text data. Once trained, this model gains the ability to predict words that are semantically similar. In practice, each label is represented through fixed-length vectors known as embedding vectors. This representation is achieved by observing that synonyms tend to co-occur in similar contexts, prompting the model to encode related words with similar embedding vectors.

To implement the language model, FastText, an open-source library developed by Facebook's AI Research (FAIR) lab, is utilized. FastText excels at converting words into embeddings based on textual data. In this context, the labels of classes are transformed into embedding vectors with dimensions set at 300. It's worth noting that the model can generate feature vectors of varying dimensions, such as 50, 100, 200, and 300. For this particular experiment, the choice is to employ 300-dimensional word vectors, as they provide richer information and enhance the model's ability to capture semantic nuances.

3. Deep Visual Semantic Model: The Deep Visual Semantic Model, as outlined in [1], merges a visual model with a language model. During training, models are exposed to training images alongside their corresponding label embedding vectors generated using the FastText library. This process establishes a mapping between visual features and language models, resulting in 300-dimensional representations.

In the testing phase, when confronted with a new image from a zero-shot class (an unseen class), the model first computes the visual feature vector complexity using the visual model. It then searches for the nearest labels within the embedding vector space, relying on cosine similarity as the similarity metric. The top 5 nearest embedding vectors are subsequently presented as the output.

The proposed DeVISE framework, employing a deep neural network with a CNN model, is depicted in Figure 5.3. This architecture comprises two convolutional layers followed by a max-pooling layer, two dense layers, and incorporates a ReLU activation function within the convolutional layer. To mitigate overfitting, a dropout layer is introduced. Notably, the CNN visual model generates a feature vector of 300 dimensions.

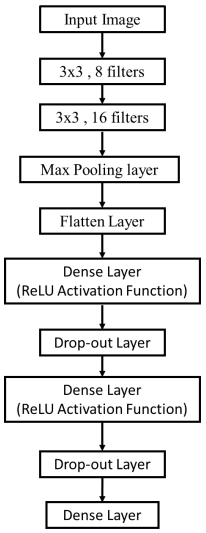


Fig 5.3 DeVISE using deep neural network CNN Model

The proposed DeVISE approach, employing a deep neural network with the ResNet 50 model, is illustrated in Figure 5.4. This ResNet 50 model is pre-trained on the extensive ImageNet dataset, and it's complemented by two dense layers and a dropout layer for enhanced performance. Notably, the ResNet 50-based visual model also yields a 300-dimensional feature vector.

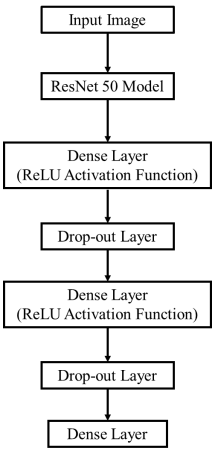


Fig 5.4 Enhanced zero-shot learning using deep neural network ResNet-50

E. Algorithm

Step 1: Initialize and Prepare Environment
Step 2: Convert Class Labels to Word Vectors
For each class label **do**:
Tokenize the label into words.
Average Word Vector = (Sum of Word Vectors of Words in Label) / (Number of Words in Label)
End
Store the calculated word vector as the class label vector.
Step 3: Calculate cosine similarity between class labels based on their word vectors.
Step 4: Define the architecture of a deep learning model, including data augmentation layers, a pre-trained ResNet50 base model, and additional dense layers for classification.
Step 5: Compile the model with an appropriate optimizer and loss function (Cosine Similarity) to evaluate model performance.
Step 6: Create custom data generators for training and validation data to handle data loading in batches.
Step 7: Train the Model
For each epoch (i ∈ [0, 1, 2, ..., steps_per_epoch]) **do**
Train the model on a batch of training data.
End
Step 8: Save the trained model to a file for future use.
Step 9: Evaluate the Model
For each batch of validation data (i ∈ [0, 1, 2, ..., validation_steps]), **do**
Evaluate the model's performance on a batch of validation data.
End
Step 10: Predict on Test Images
For each test image (i ∈ [0, 1, 2, ..., n]) **do**
Use the trained model to predict class labels for each test image based on cosine similarity.
End
Step 11: Save the predictions for test images to a JSON file.
Step 12: Optional Predict on Additional Images
If additional images are available, do
For each additional image (i ∈ [0, 1, 2, ..., n]) **do**
Use the trained model to predict class labels for each additional image based on cosine similarity.
End
Step 13: Display Predictions
For each selected image (i ∈ [0, 1, 2, ..., n]) **do**
Display the predicted class labels and images for each selected image.
End

F. Drawback of DeVISE approach

1. Prediction Quality:

One of the limitations of the Devise model is associated with prediction quality. While Devise serves as a valuable tool for zero-shot learning, it sometimes faces challenges in delivering high-quality predictions. This limitation becomes particularly evident when the model is tasked with classifying images from unseen categories. The root of this limitation can be traced back to Devise's heavy reliance on semantic descriptions. For instance, consider an image of a rare species of bird with unique feather patterns. Devise may struggle to accurately classify this image solely based on semantic attributes like "bird" and "feathers," as it may not capture the intricate visual details that distinguish this particular species. In contrast, other methods, such as those incorporating Fast Fourier Transform (FFT) to extract fine-grained visual features, may offer superior prediction quality by capturing these nuanced characteristics. Additionally, the semantic features used in Devise may not always offer a comprehensive representation of the classes, further contributing to the challenge of achieving consistently accurate predictions. Addressing this drawback is essential for enhancing prediction accuracy and the overall reliability of zero-shot learning.

2. Limited Training on Classes:

Another notable constraint in the Devise model is its training approach, specifically the limited number of classes on which it is trained relative to the total number of available classes. In the Devise framework, only 40 out of 100 classes are used for training. This selective training approach results in a significant coverage gap within the dataset. Consequently, the model lacks exposure to a substantial portion of the data, making it less adept at recognizing and classifying unseen categories effectively. In contrast, the proposed solution focuses on training on a more comprehensive set of 18 classes. This strategy aims to mitigate the limitation by providing the model with a broader understanding of the data distribution, which can lead to enhanced performance when dealing with zero-shot classification tasks. This aspect holds significant importance for achieving better generalization and improving the model's ability to handle a wider range of unseen categories.

G. Hierarchical Approach for Zero Shot Learning

The Enhanced Deep Visual Semantic Embedding approach to zero-shot learning employs two distinct strategies for classification. Firstly, it utilizes a ResNet 50-based single model, trained on the entire training dataset, to perform classification. However, a hierarchical approach is also introduced, as depicted in Figure 5.5.

In this hierarchical approach, the primary classification stage relies on a CNN model trained to categorize images into two sub-classes, namely, class-1 and class-2. This initial CNN model is specifically trained on a subset of the dataset, contributing to the effectiveness of zero-shot classification. It serves the purpose of broadly classifying images into either class-1 or class-2 subcategories.

Subsequently, the secondary classification phase involves two distinct models, one for each of the sub-classes, i.e., subclass-1 and class-2. These models are tailored for the task of zero-shot classification and are trained using ResNet 50-based architecture. Subclass-1 and class-2 are trained on subcategories of images, enabling more precise classification within their respective domains.

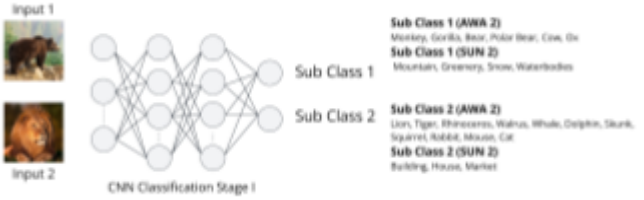


Figure 5.5 – Architecture of Proposed Hierarchical Approach for Zero shot learning.

The Zero-Shot Classification (ZSL) model, as depicted in Figure 5.6, is a fusion of both a visual model and a language model. The visual model harnesses the power of a pre-trained ResNet 50 model, skillfully trained on a wide array of images. Likewise, the language model is also a pre-trained entity, specifically tailored for the labels associated with images. Together, these models work in concert to perform zero-shot classification tasks.

In this particular approach, the dataset is categorized into two distinct groups based on their visual similarities and structural characteristics. Within the AWA2 dataset, the first category encompasses animals like monkeys, gorillas, bears, polar bears, cows, and oxen. Meanwhile, the second category is composed of animals with specific attributes, including four-legged creatures, aquatic animals, and those

with furry coats such as lions, tigers, rhinoceroses, walruses, whales, dolphins, skunks, squirrels, rabbits, mice, and cats.

Similarly, in the SUN dataset, a similar categorization strategy is applied. The first category comprises images depicting natural landscapes like mountains, greenery, snowscapes, and bodies of water. In contrast, the second category is focused on man-made structures, encompassing buildings, houses, and marketplaces. Figure 5.6 provides a visual representation of the dataset arrangement within the Proposed Hierarchical Approach for Zero-Shot Learning.

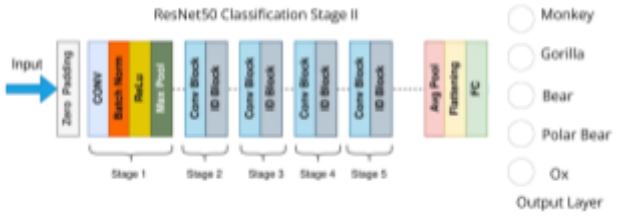


Figure 5.6 – ZSL Model of Proposed Hierarchical Approach for Zero shot learning.

The primary classifier plays a fundamental role in achieving broad classification within the proposed system, categorizing images into two distinct subclasses: Subclass 1 and Subclass 2. Focusing on the CNN-based classifier designed for the SUN dataset, as depicted in Figure 5.8, it encompasses key components like convolutional layers, batch normalization layers, max-pooling layers, and dense layers. Extensive experimentation involving different combinations of layers with varying kernel sizes was conducted to optimize the model. After careful refinement, the final configuration was selected. Notably, this CNN classifier exhibits robust performance, delivering a training accuracy of 85.86% and a validation accuracy of 87.26%. For the AWA2 dataset, a similar CNN classifier, depicted in Figure 5.9, was employed. This classifier employs five convolutional layers and, upon training, yields a training accuracy of 84.72% and a validation accuracy of 84.28%.

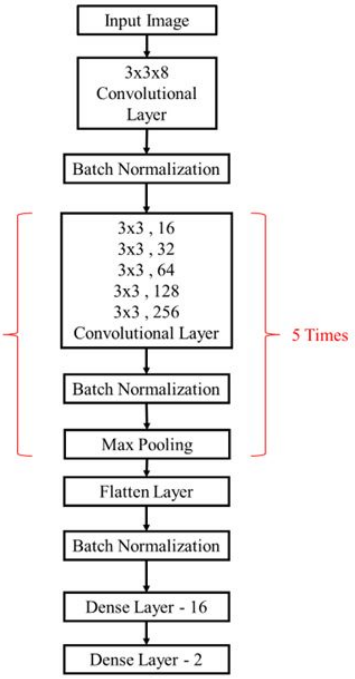


Fig 5.8 Proposed CNN based classifier for primary classification – SUN dataset

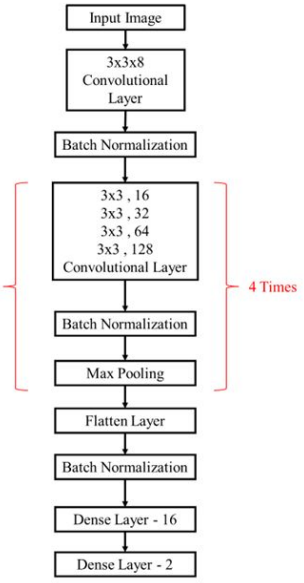


Fig 5.9 Proposed CNN based classifier for primary classification – AWA2 dataset

Accuracy results for both the SUN and AWA2 datasets are presented in Figures 5.10 and 5.11, illustrating the model's performance through visual accuracy plots.

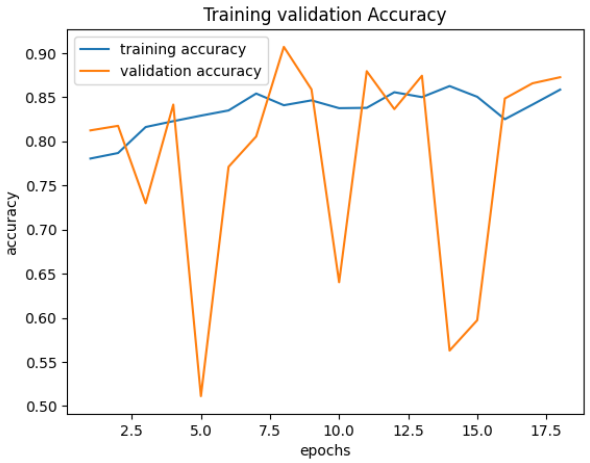


Fig 5.10 Accuracy plot of Proposed CNN based classifier for primary classification – SUN dataset

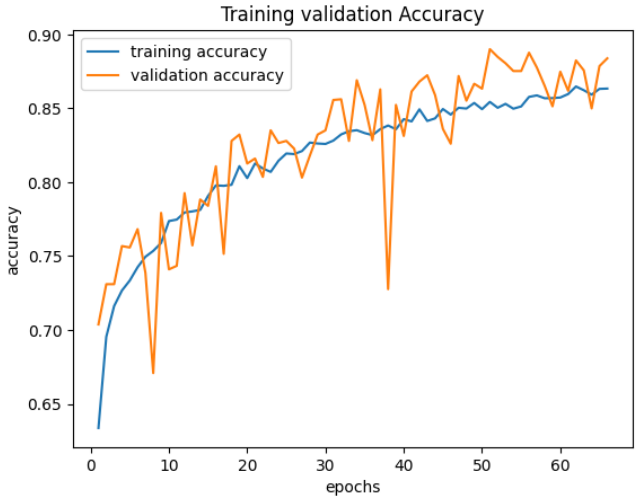


Fig 5.10 Accuracy plot of Proposed CNN based classifier for primary classification – AWA2 dataset

H. Metrics of Computational Complexity

The proposed Hierarchical Approach is evaluated on various metrics of computational complexity.

1. FLOPs: The term "Floating-Point Operations," often referred to as FLOPs, stands as a pivotal metric for gauging the complexity of a model. These operations encompass a range of mathematical operations involving floating-point values, including addition, subtraction, division, multiplication, and various others. It's essential to note that different layers within a Convolutional Neural Network (CNN) entail varying numbers of computational operations, contingent on the type of layer. FLOPs play a crucial role in the calculation of inference time, which signifies the time required for forward propagation. Below, I present the FLOPs calculations associated with different layers of a CNN.

a. Convolutional Layer: The convolutional layer stands as the foundational building block within a CNN, serving as the epicenter of most computational operations. This layer requires two primary components: the input image and the kernel. The input image is characterized by its dimensions of height, width, and depth, while the kernel, also known as a filter, functions as a feature detector. When applied to the input image, the filter initiates a process known as convolution, wherein a dot product is computed between the input image and the filter. This operation is represented by equation 1 and is commonly referred to as convolution.

$$y_i^f = \sum_{n=1}^{n_i} \sum_{j=1}^{n_k} x_{i+j-1,n}^{in} . k_{j,n}^f + b^f \qquad (1)$$

where yif i denotes the output, known as a feature map, of a convolutional layer built by the filter f in the i-th input element, nk is the kernel size, ni is the size of the input vector, xin represents the raw input data, kjf j denotes the j-th trainable convolution kernel of the filter f and bf is the bias of the filter f. FLOPs for convolutional layer is calculated using equation-5.2 [19].

FLOPs for convolutional layer = 2 *n* kernel shape*output shape (2)
n = Number of kernels

Kernel shape = W * H (3)
W and H are width and height of the kernel.

Output shape of convolutional layer = (M-W+1)(N-H+1) (4)
Where M and N are height and width of the input image.

For first convolutional layer of Model for AWA2 dataset is using 8 kernels of size 3 × 3 and input image is of 224 × 224 ×3.
M = 224, N=224, W = 3, H = 3

Output shape of convolutional layer =(224-3+1)(224-3+1)
= 222 × 222

FLOPs for convolutional layer = 2 ×8× 3×3 × 222 × 222
= 7,096,869

b. Fully Connected Layers:

In the fully connected layer, each node in the output layer connects directly to a node in the previous layer. FLOPs for fully connected layer is calculated using equation 5

Flops for fully connected layer = 2 × number of nodes in input layer × number of nodes in output layer
(5)

The fully connected layer of Model for AWA2 dataset is classified into 2 classes so number of nodes in output layer is 2. Input to layer is 16 neurons.

Flops for fully connected layer = 2 ×16 ×2= 64

c. Pooling Layer:

Pooling layer is used to reduce dimensionality reduction. Similar to the convolutional layer, the pooling operation sweeps a filter across the entire input, but the difference is that this filter does not have any weight. Instead, the kernel applies an aggregation function to the values within the receptive field. FLOPs for pooling layer is calculated using the equation 6

FLOPs for pooling layer =Height ×width ×Depth of an input image to pooling layer (6)

Last Pooling layer of Model for AWA2 dataset Height and width is 12 ×12 and depth is 128.

FLOPs for pooling layer = 12 ×12 × 128
= 18,432

d. Batch Normalization Layer

It is used to normalize the output of the previous layers. The activations scale the input layer in normalization. Using batch normalization learning becomes efficient also it can be used as regularization to avoid overfitting of the model.

FLOPs for batch normalization is given by equation 7
FLOPs for batch normalization =

$$C \times 4 \times Output\ shape \quad (7)$$

C is channels in the con.volution layer's output.

FLOPs for first batch normalization layer which consists of 8 channels and output shape of 222 × 222.

FLOPs for batch normalization = 8 ×4 ×222 × 222
= 1,577,088

FLOPs for EZSL ResNet 50, CNN based classifier for SUN dataset and CNN based classifier for AWA2 dataset are calculated using the equation 5.2 to 5.7. Details of the layers and calculated FLOPs are shown in Table-5.1, Table-5.2 and Table-5.3.

Table-5.1 Details of the layers and calculated FLOPs of DeVISE ResNet 50 Model

Sr No.	Layers of EZSL ResNet 50 Model	FLOPs
1.	ResNet 50 Pretrained Model [20]	3.8×10^9
2.	Dense Layer - 1	18,35,008
3.	Dense Layer - 2	3,44,064
4.	Dense Layer - 3	2,30,400
Total Number of FLOPs DeVISE ResNet 50 Model		**3.802409** × 10^9

Table-5.2 Details of the layers and calculated FLOPs of CNN based classifier model for SUN dataset

Sr No.	Layers of CNN based classifier model for SUN dataset	FLOPs
1.	Convolution layer – 1	7,096,896
2.	Batch Normalization Layer -1	1,577,088
3.	Convolution layer – 2	111,503,600
4.	Batch Normalization Layer - 2	3,097,600
5.	Max Pooling Layer -1	193,600
6.	Convolution layer – 3	107,495,424
7.	Batch Normalization Layer – 3	746,496
8.	Max Pooling Layer -2	93,312
9.	Convolution layer – 4	99,680,256
10.	Batch Normalization Layer – 4	692,224
11.	Max Pooling Layer - 3	43,264
12.	Convolution layer – 5	84,934,656
13.	Batch Normalization Layer – 5	294,912
14.	Max Pooling Layer - 4	18,432
15.	Convolution layer – 6	58,982,400
16.	Batch Normalization Layer – 6	10,240
17.	Max Pooling Layer - 5	6400
18.	Flatten layer	0
19.	Batch Normalization Layer – 7	1024
20.	Dense Layer - 1	204,800
21.	Dense Layer - 2	64
Total Number of FLOPs CNN based classifier model for SUN dataset.		**476.682** × 10^6

Table-5.3 Details of the layers and calculated FLOPs of CNN based classifier model for SUN dataset.

Sr No.	Layers of CNN based classifier model for AWA2 dataset	FLOPs
1.	Convolution layer – 1	7,096,896
2.	Batch Normalization Layer -1	1,577,088
3.	Convolution layer – 2	111,503,600
4.	Batch Normalization Layer - 2	3,097,600
5.	Max Pooling Layer -1	193,600
6.	Convolution layer – 3	107,495,424
7.	Batch Normalization Layer – 3	746,496
8.	Max Pooling Layer -2	93,312
9.	Convolution layer – 4	99,680,256
10.	Batch Normalization Layer – 4	692,224
11.	Max Pooling Layer - 3	43,264
12.	Convolution layer – 5	84,934,656
13.	Batch Normalization Layer – 5	294,912
14.	Max Pooling Layer - 4	18,432
15.	Flatten layer	0
16.	Drop out layer	0
17.	Dense Layer - 1	58,982,400
18.	Batch Normalization Layer – 6	1024
19.	Drop out layer	0
20.	Dense Layer - 2	64
Total Number of FLOPs CNN based classifier model for AWA2 dataset.		**417.774**× 10^6

Total number of of FLPOs for Hierarchical Approach based model for SUN dataset =

FLOPs of CNN based classifier model for SUN dataset + EZSL ResNet 50 Model for subclass -1 + EZSL ResNet 50 Model for subclass -2

$= 476.682 \times 10^6 + 3.802409 \times 10^9 + 3.802409 \times 10^9$

= 8.0815*10^9

Total number of of FLPOs for Hierarchical Approach based model for AWA2 dataset =

FLOPs of CNN based classifier model for AWA2 dataset + EZSL ResNet 50 Model for subclass -1 + EZSL ResNet 50 Model for subclass -2

$= 417.774 \times 10^6 + 3.802409 \times 10^9 + 3.802409 \times 10^9$

= 8.022*10^9

2. Training Complexity: The training complexity of model depends on number of FLOPs, number of epochs, number of input image for training.

Training complexity = Number of FLOPS ×Number of epohcs ×number of input (5.8)

Training complexity of model depends on dataset on which it is trained. If Size of dataset increases the training complexity of model increases. At the same time for large dataset, it takes a smaller number of epochs to fit the model then models training complexity can be reduced.

Training complexity of EZSL ResNet 50 = $Number\ of\ FLOPS \times Number\ of\ epohcs \times number\ of\ inpu$

$Number\ of\ FLOPS\ EZSL\ ResNet\ 50 = 3.8024 \times 10^9$

$Number\ of\ epohcs = 19$

$number\ of\ input = 11,735$

Training complexity of EZSL ResNet 50

$= 3.8024 \times 10^9 \times 19 \times 11,735$

$= 84.78 \times 10^{13}$

Similar to above calculations Training complexity of Enhanced ZSL Model, Hierarchical Approach based model for SUN dataset, Hierarchical Approach based model for AWA2 dataset are evaluated and shown in Table 5.4.

Sr No.	Model		Number of FLOPs	No. of epochs	No. of input image	Training Complexity
1.	Enhanced ZSL Model - SUN		3.802409×10^9	19	11735	**84.78**$\times 10^{13}$
2.	Enhanced ZSL Model - AWA2		3.802409×10^9	15	24270	**1.38426**$\times 10^{15}$
3.	Hierarchical Approach based model for SUN dataset	primary classifier Model	476.682×10^6	18	2715	2.3295×10^{13}
		Sub class-1 model	3.802409×10^9	25	991	9.4204×10^{13}
		Sub class-2 model	3.802409×10^9	23	454	3.9704×10^{13}
		Total Training Complexity				**19.69** $\times 10^{13}$
4.	Hierarchical Approach based model for AWA2 dataset	primary classifier Model	417.774×10^6	66	9767	2.6931×10^{14}
		Sub class-1 model	3.802409×10^9	11	3278	1.37×10^{13}
		Sub class-2 model	3.802409×10^9	10	6408	2.4365×10^{13}
		Total Training Complexity				**6.499** $\times 10^{14}$

Table – 5.4 Training complexities of evaluation of various models.

I. Algorithm

Step 1: Initialize and Prepare Environment

Step 2: Data Loading and Preprocessing (Code1)

Step 3: Specify layers, activation functions, and output layer based on the problem (BaseModel)

Step 4: Compile the BaseModel

- Compile the model with appropriate optimizer, loss function, and metrics.
- Use Cosine Similarity as the loss function for evaluating model performance.

Step 5: Training Loop (BaseModel)

For each training epoch, do:
Train the model on a batch of training data.
Update model weights.
Continue until the desired number of epochs is reached.

Step 6: Model Evaluation (BaseModel)

- Evaluate the trained model using validation data.
- Calculate and print the model's loss.

Step 7: Save the Trained BaseModel

Step 8: Load Pretrained ResNet50 Models

- Load pretrained deep learning models (**BaseModel** and secondary models, e.g. **SubModel_1**, **SubModel_2**, etc.) for image classification.

Step 9: Prediction Loop (Code2 and Code3)

For each image in the specified directory, do:
1. Load and preprocess the image.
2. Use the primary model (**BaseModel**) to predict the initial class lab
3. Optionally, based on the primary model's output (e.g., **out** variable secondary models (**SubModel_1**, **SubModel_2**, etc.) for further classification.
4. Use the selected model to predict class labels.
5. Display the image and predicted labels.

Step 10: Counters and Class Identification

- After predicting labels with different models, update counters for each animal cla
- Determine the predicted class for each image based on the highest counter value.

Step 11: Display Predictions and Save Results

- Save the predictions in a dictionary (**save_pred**) for further analysis.

VI. Result

In today's data-driven landscape, the exponential growth in digital content has led to a substantial increase in unlabeled data. While traditional image classification methods excel when equipped with labeled data, the surge in unlabeled data presents a formidable challenge. This dilemma has spurred the emergence of zero-shot learning, a promising solution designed to adeptly classify previously unseen data instances.

Deep Visual Semantic Embedding represents a notable approach within the zero-shot learning paradigm, specializing in the classification of these hitherto unseen classes. The study at hand implements a hierarchical approach to zero-shot learning, employing a CNN-based classifier for the initial, broad classification of unseen classes into either subclass-1 or subclass-2. Subsequently, dedicated models for Subclass-1 and Subclass-2 are employed to undertake zero-shot learning classification for these specific categories. The method is then rigorously benchmarked against Enhanced Zero-Shot Learning using ResNet 50, providing valuable insights into the effectiveness of this hierarchical approach in addressing the unique challenges posed by zero-shot learning.

Results are tested on the following datasets:

- **SUN Dataset:**

  The SUN [2] dataset comprises encompasses a collection of 14,340 scene images depicting various landscapes, including rivers, hills, churches, beaches, and more. For the purpose of zero-shot learning, this dataset is thoughtfully divided into two segments: a training set featuring 645 seen categories, and a testing set comprising 72 unseen classes.

- **AWA2 Dataset:**

  The AwA-2 [3], encompasses a total of 50 distinct categories, featuring an extensive collection of 37,322 animal images. In the context of this experiment, 40 of these categories are thoughtfully employed for training purposes, while the remaining 10 categories serve as the test set for evaluation.

Results are evaluated using following parameters:

- **Per Class Average Accuracy** = $\frac{1}{N}\sum_{i=1}^{N}\left[\frac{Y_{correct\ class}^{class\_i}}{Y_{Total}^{class\_i}}\right]$
  N is total number of classes in dataset
- **Model Loss** = $-\sum_{i=1}^{N}\left[l^2norm(Y-true)*\ l^2norm(Y-pred)\right]$

Cosine similarity model loss is used. Its value is a number between -1 and 1. When it is a negative number between -1 and 0, 0 indicates orthogonality and values closer to -1 indicate greater similarity. The values closer to 1 indicate greater dissimilarity

- **Flops** – Number of floating-point operations of model Training
- **Time** – Time to train the model.
- **Testing Time** – Time to test one image.
- **Training complexity**– Training complexity depends on the number of FLOPs, number of epochs and number of input images

A. DeVISE using deep neural network ResNet50 – Results:

The development of DeVISE, leveraging the power of a deep visual semantic model, involves the creation of a visual model utilizing both CNN and ResNet 50, while the language model is constructed using FastText. This model, once implemented, demonstrates the capability to classify previously unknown image categories. To rigorously evaluate its performance, the proposed model undergoes testing on well-established benchmark datasets, namely SUN [2] and AWA2 [3].

In the testing phase, when the model is presented with unseen images, it promptly extracts visual features and proceeds to predict the corresponding embedding vectors. Subsequently, it generates predictions by identifying the top 5 most similar embedding vector labels. The outcomes of these predictions for both the SUN and AWA2 datasets are thoughtfully depicted in Figures 6.1 and 6.2, offering valuable insights into the model's classification prowess.

Actual Label: Valley
Top 5 Prediction for an image: mountain, **valley**, hillside, hill, tree-line

Actual Label: forest
Top 5 Prediction for an image: garden, back-garden, house, farm, gardens

Fig 6.1 Results for SUN dataset - DeVISE ResNet 50

Actual Label: chimpanzee
Top 5 Prediction for an image: gorilla, gorillas, chimp, **chimpanzee**, orangutan

Actual Label: tiger
Top 5 Prediction for an image: tiger, bobcat, tigers, **leopard**, tiger-striped

Fig 6.2 Results for AWA2 dataset - DeVISE ResNet 50

The training process spans 20 epochs and employs the callback mechanism. Within this framework, the early stopping optimization technique is deployed to effectively prevent overfitting. The model optimization leverages the Adam optimizer.

Visualizing the model's training progress, Figures 6.3 and 6.4 illustrate the loss dynamics for both the SUN and AWA2 datasets. Notably, as the training epochs progress, the model's loss steadily diminishes, signifying the refinement and improved performance achieved through iterative training.

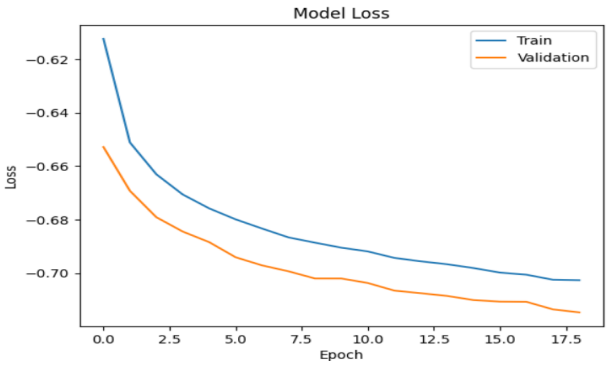


Fig.6.3 Model loss – SUN dataset

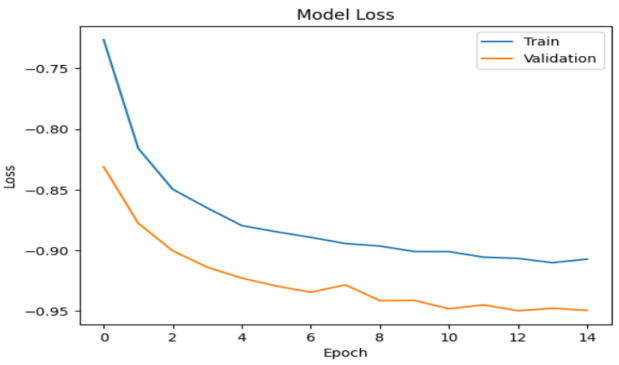
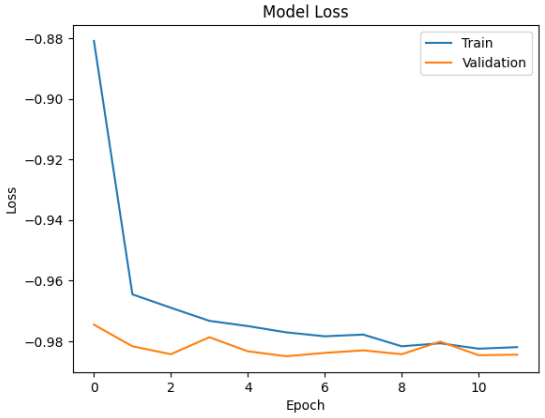


Fig 6.4 Model loss - AWA2 dataset

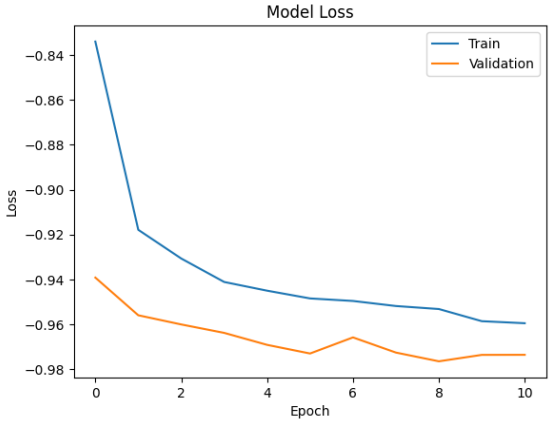
B. Hierarchical Approach for Zero shot learning – Results:

The Hierarchical Approach is structured with a dual-level classification system, comprising a primary and secondary tier. At the primary classification level, a CNN-based classifier takes the lead, providing a broad classification of images into two distinct subclasses. Subsequently, the secondary classification level is executed using a ResNet 50-based model.

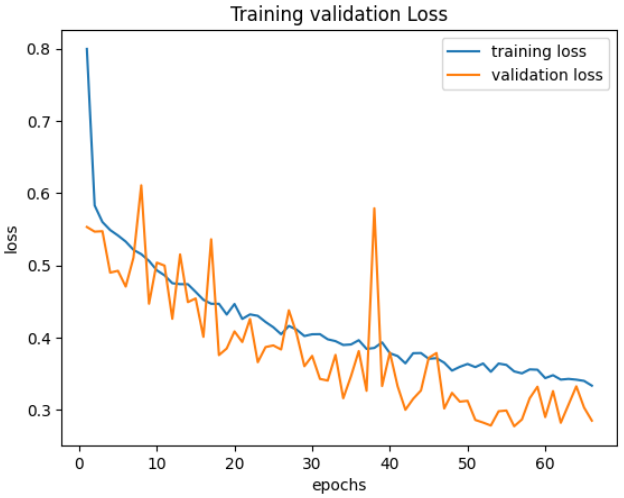
In the context of zero-shot class images, the CNN-based classifier plays a pivotal role in categorizing the image into either Subclass-1 or Subclass-2 based on visual similarities. The Zero-Shot Learning (ZSL) classifier for Subclass-1 or Subclass-2 then takes over, extracting visual features via ResNet 50 and proceeding to predict the corresponding embedding vectors. As a result, the model delivers predictions by identifying the top 5 most similar embedding vector labels. The outcomes of these predictions for both the SUN and AWA2 datasets are visually depicted in Figures 6.5 and 6.6, offering a comprehensive view of the model's classification capabilities.



(a) Model loss – ZSL model -1

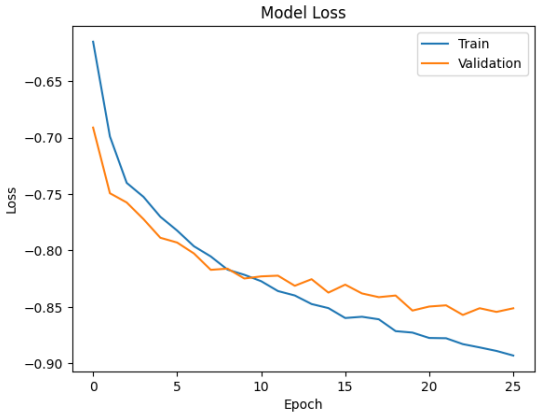


(b) Model loss – ZSL model -2

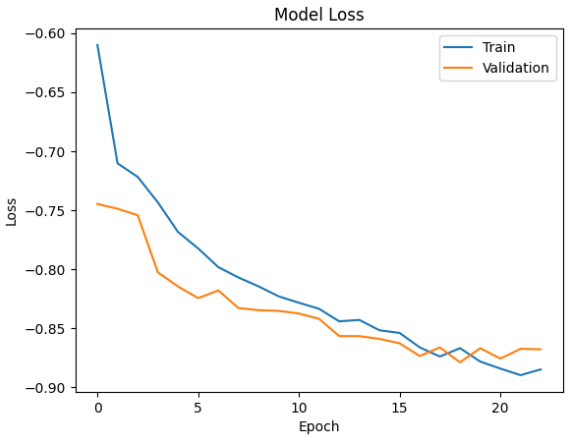


(c) Model loss - Primary model

Fig 6.7 Model loss – AWA2 dataset - Hierarchical Approach



(a) Model loss – ZSL model -1



(b) Model loss – ZSL model -2

(c) Model loss - Primary model

Fig 6.7 Model loss – SUN dataset - Hierarchical Approach

C. *Comparative analysis of Quality, Computational and time Complexity*

Upon training both the DeVISE and the Hierarchical Approach, these models are rigorously tested on unseen classes, specifically zero-shot classes. The resulting accuracy metrics are conveniently presented in Table 6.1. Notably, the Hierarchical Approach demonstrates a notable improvement in accuracy compared to the Enhanced Zero-Shot Learning approach, with accuracy gains ranging from approximately 10% to 15%.

In Table 6.2, a detailed view of the model's computational complexity parameters is provided, with a focus on FLOPs (Floating-Point Operations). It's worth noting that FLOPs tend to increase within the Hierarchical Approach; however, there is a simultaneous reduction in training complexity. This reduction is primarily attributed to the hyperparameter that determines the number of input images, which significantly impacts the model's training complexity. Furthermore, the Hierarchical Approach is trained on more compact datasets, further contributing to the improved training complexity, which experiences an enhancement ranging from approximately 23% to 47%.

Additionally, when considering training and testing times, the Hierarchical Approach exhibits a notable enhancement in comparison to the Enhanced Zero-Shot Learning method. In summary, the Hierarchical Approach not only delivers superior classification accuracy but also demonstrates a reduction in timing complexity, rendering it a more efficient and effective choice for zero-shot learning tasks.

Sr No.	Method	Dataset	Accuracy
1.	DeVISE using deep neural network ResNet50	SUN	39.5%
		AWA2	44.786
2.	Hierarchical Approach for Zero Shot learning	SUN	50.65%
		AWA2	60.438%

Table 6.1 Quantitative analysis of quality

Sr No.	Method	Dataset	Training Time (sec)	Testing Time (sec)	FLOPS	Training Complexity
1.	DeVISE using deep neural network ResNet50	SUN	809.9183	0.2201	3.802409*10^9	84.78*10^13
		AWA2	817.5256	0.3912	3.802409*10^9	13.8426*10^14
2.	Hierarchical Approach for Zero Shot learning	SUN	656.3926	0.264	8.0815*10^9	**19.69*10^13**
		AWA2	4,859.7809	**0.297**	8.022*10^9	**6.499*10^14**

Table 6.2 Quantitative analysis of quality

VII. Conclusion

Zero-shot learning is a machine learning technique designed to classify classes not present in the training dataset. This process involves several key steps: visual feature extraction, semantic feature representation, visual-semantic mapping, and the classification of previously unseen classes. In this context, an enhanced zero-shot learning approach employing the ResNet-50 deep neural network has been proposed, demonstrating superior accuracy when compared to CNN-based visual models.

Furthermore, a hierarchical approach to zero-shot learning has been introduced and applied to two standard datasets, SUN and AWA2. A comparative analysis has been conducted, evaluating the quality and computational complexity of the enhanced zero-shot learning using ResNet-50 with the hierarchical approach. The results indicate a significant improvement in accuracy, with the hierarchical approach outperforming the EZSL ResNet-50 approach by 10% to 15%. Additionally, the training complexity of the hierarchical approach has been optimized, reducing it by 23% to 47%.

It's important to note that the hierarchical approach employs three models for zero-shot learning classification, which increases both floating-point operations (FLOPs) and

training time. However, this is offset by the improved training complexity and testing time.

Looking ahead, future research will investigate ZSL classification using autoencoders and optimized feature extraction through autoencoders to further enhance zero-shot learning.

Reference

[1] Palatucci M, Pomerleau D, Hinton G, Mitchell T (2009) "Zero-shot learning with semantic output codes", Adv Neural Inf Proces Syst 1:1410–1418

[2] Sun Attribute Database: Discovering, annotating, and recognizing scene attributes (no date) SUN Attribute Dataset. Available at: https://cs.brown.edu/~gmpatter/sunattributes.html (Accessed: December 20, 2022).

[3] Available at: https://cvml.ist.ac.at/AwA2/ (Accessed: December 20, 2022).

[4] Lampert, C.H., Nickisch, H. and Harmeling, S. (2009) 'Learning to detect unseen object classes by between-class attribute transfer', 2009 IEEE Conference on Computer Vision and Pattern Recognition [Preprint]. doi:10.1109/cvpr.2009.5206594.

[5] Lampert, C.H., Nickisch, H. and Harmeling, S. (2014) 'Attribute-based classification for zero-shot visual object categorization', IEEE Transactions on Pattern Analysis and Machine Intelligence, 36(3), pp. 453–465. doi:10.1109/tpami.2013.140.

[6] Zeynep Akata, Florent Perronnin, Zaid Harchaoui, and Cordelia Schmid. "Label-embedding for image classification". IEEE TPAMI, 38(7):1425–1438, 2015.

[7] Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Marc'Aurelio Ranzato, and Tomas Mikolov. Devise: A deep visual-semantic embedding model. In Advances in neural information processing systems, pages 2121–2129, 2013.

[8] Xuesong Wang, Chen Chen, Yuhu Cheng , Xun Chen and Yu Liu "Zero-Shot Learning Based on Deep Weighted Attribute Prediction" IEEE Transactions on Systems, Man, and Cybernetics: Systems (Volume: 50, Issue: 8, Aug. 2020)

[9] Yongqin Xian, Tobias Lorenz, Bernt Schiele, Zeynep Akata "Feature Generating Networks for Zero-Shot Learning", Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 5542-5551

[10] Gao, R., Hou, X., Qin, J., Liu, L., Zhu, F., Zhang, Z. (2019). A Joint Generative Model for Zero-Shot Learning. In: Leal-Taixé, L., Roth, S. (eds) Computer Vision – ECCV 2018 Workshops. ECCV 2018. Lecture Notes in Computer Science (), vol 11132. Springer, Cham. https://doi.org/10.1007/978-3-030-11018-5_50

[11] Varun Khare, Divyat Mahajan, Homanga Bharadhwaj, Vinay Verma, Piyush Rai, "Generative Framework for ZSLwith Adversarial Domain Adaptation", in 2020 IEEE Winter Conference on Applications of Computer Vision (WACV).

[12] Al Machot, F.; Ullah, M.; Ullah, H. HFM: A Hybrid Feature Model Based on Conditional Auto Encoders for Zero-Shot Learning. J. Imaging 2022, 8, 171. https://doi.org/10.3390/jimaging8060171

[13] LEE, R. and CHEN, I.-Y. (2020) 'The Time Complexity Analysis of neural network model configurations', 2020 International Conference on Mathematics and Complexity in Science and Engineering (MACISE) [Preprint]. doi:10.1109/macise49704.2020.00039.

[14] Freire, P. J., Srivallapanondh, S., Napoli, A., Prilepsky, J. E., & Turitsyn, S. K. (2022) 'Computational Complexity Evaluation of Neural Network Applications in Signal Processing', ArXiv. /abs/2206.12191

[15] Shah, B. and Bhavsar, H. (2022b) 'Time complexity in deep learning models', Procedia Computer Science, 215, pp. 202–210. doi: 10.1016/j.procs.2022.12.023.

[16] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, pages 427–431, 2017.

[17] Socher, R.; Ganjoo, M.; Manning, C.D.; Ng, A. Zero-shot learning through cross-modal transfer. Adv. Neural Inf. Process. Syst. 2013, 26, 1–10.

[18] S. Albawi, T. A. Mohammed and S. Al-Zawi, "Understanding of a convolutional neural network," 2017 International Conference on Engineering and Technology (ICET), 2017, pp. 1-6, doi: 10.1109/ICEngTechnol.2017.8308186.

[19] Lecture 41: Space and Computational Complexity in DNN. nptel. Available at: https://www.youtube.com/watch?v=hGu2VIaEbHE.

[20] He, K. et al. (2016) 'Deep residual learning for image recognition', 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) [Preprint]. doi:10.1109/cvpr.2016.90. – ResNet FLOPs

IEEE conference templates contain guidance text for composing and formatting conference papers. Please ensure that all template text is removed from your conference paper prior to submission to the conference. Failure to remove template text from your paper may result in your paper not being published.