International Journal of Engineering Trends and Technology
ISSN: 2231–5381 / https://doi.org/10.14445/22315381/IJETT-V73I7P142

Volume 73 Issue 7, 543-561, July 2025

Original Article

Hierarchical Framework to Reduce Zero-Shot Learning Complexity

Shaista Khanam[1], Poonam. N. Sonar[2]

[1,2]*Department of Electronics & Telecommunication Engineering, RGIT, Andheri, Mumbai, India.*
[1]*Department of Electronics & Telecommunication Engineering, VCET, Vasai Road, India.*

[1]*Corresponding Author : shaista.khan@vcet.edu.in*

Received: 27 February 2025 Revised: 07 July 2025 Accepted: 16 July 2025 Published: 30 July 2025

Abstract - *Zero-Shot Learning (ZSL) is an emerging machine learning approach that enables the classification of images belonging to categories absent from the training data. By leveraging semantic information, ZSL facilitates classification with minimal or no training images. This paper presents a novel approach for ZSL employing a hierarchical framework, designed to enhance accuracy while significantly reducing complexity. The proposed framework employs a two-stage hierarchical classification structure with primary and secondary classifiers specific to each stage of the hierarchy. A Convolutional Neural Network (CNN) works as the principal component of the primary classifier, which uses a deep hierarchical clustering technique to classify images into two larger categories (Subclass-0 and Subclass-1). The secondary classifier integrates fastText for semantic feature extraction and ResNet-50 for visual feature extraction, enabling the classification of unseen (zero-shot) images. The usefulness of the proposed approach is validated on three standard datasets, viz. SUN, AWA2, and CUB. According to experimental results, the hierarchical architecture achieves accuracy levels comparable to the best available methods while drastically reducing training complexity by almost 80%, training length by 25%, and testing time by 23%. The framework facilitates more effective learning by breaking the task up into smaller class subsets, which makes it ideal for large-scale ZSL applications.*

Keywords - *Hierarchical deep clustering, Hierarchical framework, Image classification, Model complexity, Zero-shotlearning.*

1. Introduction

Traditional image classification methods have shown considerable success in several fields. Practical issues arise because these approaches mostly depend on having a sizable amount of labelled training data for every class. Furthermore, these models cannot handle unknown classes because they can only recognize the classes that they have been trained on. This restriction results from the difficulty in obtaining labelled data, which can be attributed to either the high cost of annotating data or the lack of adequate data for certain categories.

ZSL offers a key to these challenges by enabling the recognition of unlabelled data without prior training in those specific classes [1]. The ZSL model undergoes training using a predefined group of known classes and is subsequently evaluated on how effectively it can identify its ability to categorize unknown classes. To address the disparity between these known and unknown classes, ZSL relies on semantic descriptions, such as attributes or natural language sentences, to capture the essential characteristics of each class. Throughout training, the model grasps the correlation between semantic attributes and visual characteristics extracted from the data. During prediction, the model maps unlabelled images to their semantic features and makes predictions grounded on the resemblance between the predicted and existing class features, often computed using Euclidean distance and cosine similarity. ZSL is a versatile technique with broad applications, holding promise to significantly advance areas such as computer vision, natural language understanding, and human activity recognition [2]. With its ability to classify unseen data using semantic relationships, ZSL holds great promise for the future, making it a crucial advancement in machine learning.

Manually created attributes were used in early ZSL techniques; however, these methods are not scalable and necessitate subject expertise. Later research improved automation but still suffered from semantic noise and ambiguity by introducing semantic embeddings utilizing language models such as word2vec, GloVe, and fastText. By synthesizing visual attributes for invisible classes, generative models like Conditional Variational Autoencoders (CVAE), Conditional Autoencoders (CAE), and Generative Adversarial Networks (GANs) carried the discipline even farther, but at the expense of higher computational overhead and system complexity. Although classification performance has increased because of these developments, its usefulness in large-scale or time-sensitive environments is limited by the need for deep architecture and

lengthy training periods. Although recent research has highlighted the significance of optimizing computational efficiency, the tradeoff between accuracy and resource utilization is not explicitly addressed by many ZSL systems.

1.1. Major Challenges to be Addressed in ZSL

1.1.1. Accuracy

High classification accuracy for unknown classes is still a major challenge, particularly when the gap between visual and semantic representations is large or when data domains shift.

1.1.2. Model Complexity

Simplifying the model structure and having low computational requirements without compromising on strong performance are crucial, especially for real-world deployment.

1.1.3. Training Efficiency

Improving the training process to be accelerated and to consume less data is vital, especially for large-scale deployments or systems with limited processing power.

Scalability

The ZSL techniques need to be scalable enough to accommodate a vast number of categories and should be effective while increasing the number of classes across diverse application areas. To counter these challenges, this work presents a novel hierarchical ZSL method that splits the classification process into two steps: a lightweight CNN-based coarse classifier and then a fine-grained secondary classifier consisting of ResNet-50 for visual features and semantic classification by utilizing fastText embeddings. The suggested method improves inference speed and overall scalability by reducing the number of candidate classes per prediction. Results from experiments on benchmark ZSL datasets SUN [3], AWA2 [4] and CUB [5], common ZSL datasets, show that the framework can strike a compromise between accuracy and efficiency, which qualifies it for practical use.

2. Literature Review

Classifying images is among the most demanding jobs in computer vision. ZSL is the task of identifying image categories that the model has never encountered during training. Recently, there has been a lot of research interest in this field. Lampert et al. [6], in their 2009 study, laid the groundwork for ZSL by prposing a method which utilizes manually crafted attributes as supplementary information for unknown categoriesLampert has extended the concept of attribute-based zero-shot image classification, employing Direct Attribute Prediction (DAP) and Indirect Attribute Prediction (IAP) as probabilistic classifiers [7], where visual features were matched to attributes and then used for classification. Zeynep et al. [8] proposed Attribute Label Embedding (ALE) by embedding attributes into a discriminative space, achieving better generalization and superior performance compared to DAP methods. Xuesong et al. [9] have proposed a method in which attribute labels participate in training deep neural networks and the deep neural network predicts attributes. Predicted attributes are weighted using the Sparse Representation Coefficient (SRC) as each attribute contributes differently to image recognition and improves accuracy.

The methods of [6-9] rely on handcrafted attributes which are limited and expensive. To overcome the limitations of handcrafted attributes, several researchers have turned to automatic semantic embedding techniques. The DeViSE model was presented by Andrea et al. [10], who combined skip-gram language models for semantic representation with CNN models for visual feature extraction. This allowed models to infer semantic similarity even for unseen classes. Scaling up this approach to larger dictionaries and training on extensive text corpora can further enhance prediction quality. Khanam et al. [11] have proposed the enhanced ZSL (EZSL) technique using ResNet-50, which exhibits higher accuracy and lower model loss than traditional CNN models. CNN and pre-trained CNN models are utilized to extract features, and subsequently employed to classify unknown images in methods. These techniques can scale better by utilizing vast text corpora and reducing manual labour. However, they may still suffer from noisy embeddings and semantic ambiguity, especially when utilizing unpolished word vectors. ZSL has been subjected to generative models including CVAE, CAE, and GANs in order further to enhance generalization and robustness [12–15]. These models transform the ZSL issue into a traditional supervised classification task by synthesizing visual characteristics for unseen classes using semantic attributes.

While generative approaches often lead to higher accuracy, they also significantly increase system complexity, requiring sophisticated designs and long training times. These features limit their use in large-scale or time-sensitive applications where efficiency and computational resources are crucial. These complications worsen with increasing dataset size and class diversity, underscoring the necessity for models that strike a compromise between computational viability and accuracy. Examining the space and temporal complexity of ZSL models, which are impacted by a variety of architectural and training factors, is crucial to resolving this issue. Lee et al. [16] showed that the number of layers, epochs, network depth, and optimizer selection all had an impact on CNN time complexity. In a similar vein, Freire et al. [17] suggested a methodical assessment of computational complexity in digital signal processing systems spanning neural network layers. The effects of factors including filter size, depth, number of filters, completely linked layers, and kernel size on runtime and performance were further investigated by Shah et al. [18]. These investigations provide insightful information on how deep learning models behave computationally.

The readers can refer to work on time and computational complexity of deep neural networks in [16-18]. Reducing training complexity without compromising performance has become a critical research goal, as many of the current ZSL

and image classification techniques become more complicated and resource-intensive as datasets expand. Investigating effective designs that can grow effectively while preserving competitive accuracy is therefore essential.

Key Observations from Literature Review :

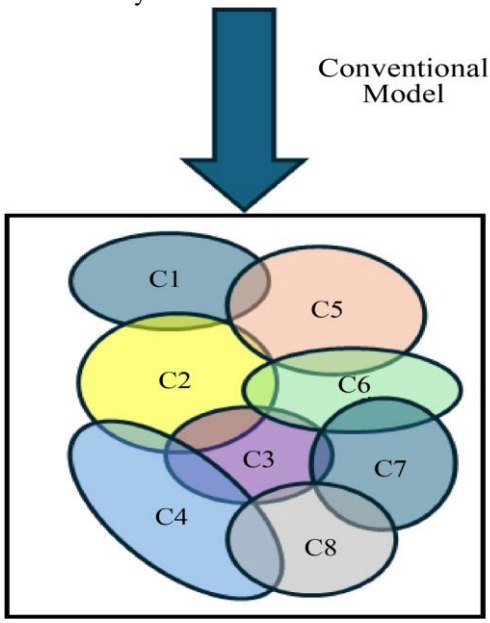
- Handcrafted Attributes
- Generative Models: Accuracy vs. Complexity
- Training and Testing Time
- Scalability Issues

Current work in Zero-Shot Learning (ZSL) identifies a number of challenges. Early techniques were largely based on manually designed attributes, which, although useful, are difficult to scale and involve a great deal of expert time. Although generative approaches like CVAE and GANs have been promising and can enhance accuracy through feature generation, they add system complexity, thus not being as appropriate for real-time or resource-constrained applications. In addition, numerous models incorporate deep architectures that lead to slow training and inference times, making them infeasible for deployment in time-sensitive scenarios. Another significant issue is scalability, with these models tending to fail in their ability to scale up well across large and varied datasets, highlighting the necessity of more efficient and robust ZSL methodologies.

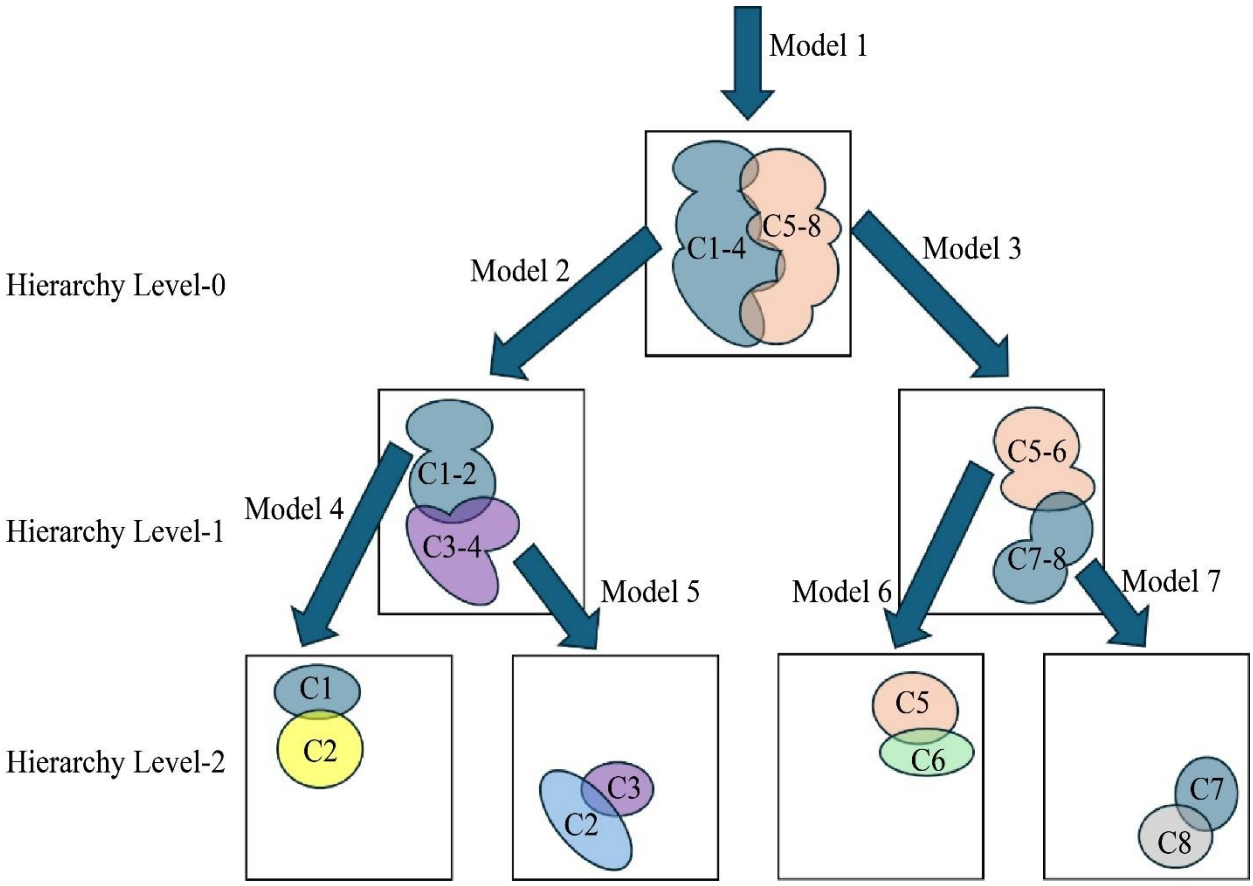
To address these challenges, this research suggests a novel hierarchical ZSL framework to overcome the drawbacks noted in earlier works, including reliance on manually created characteristics, noisy semantic embeddings, and the high complexity of generative techniques. The suggested approach reduces the number of classes per classifier and, as a result, minimizes model complexity by breaking down the classification process into two hierarchical steps, in contrast to conventional flat classification algorithms. The dataset is roughly divided into major subclasses using the primary classifier based on a lightweight CNN. A secondary classifier that uses fastText for semantic embedding and ResNet-50 for visual feature extraction handles each subclass, providing a good tradeoff between efficiency and performance. The system efficiently lowers training and inference complexity while preserving competitive accuracy by fusing deep hierarchical clustering with semantic guiding, which makes it appropriate for time-sensitive and large-scale ZSL applications.

3. Hierarchical Deep Clustering

A hierarchical approach for Zero-shot image classification combines primary and secondary stages of classification. Various classes in a dataset are clustered into two subclasses, Subclass-0 and Subclass-1, using Hierarchical deep clustering. Classifying images in one of the two subclasses is the primary classification. Hierarchical deep clustering organizes similar classes into clusters, forming a multilevel structure, as shown in Figure 1 (b). Hierarchy level -0 represents broader classes, while the higher Hierarchy levels depict more specific subclasses. In this approach, deep features are extracted through a CNN, and K-Means clustering groups classes based on the extracted features.

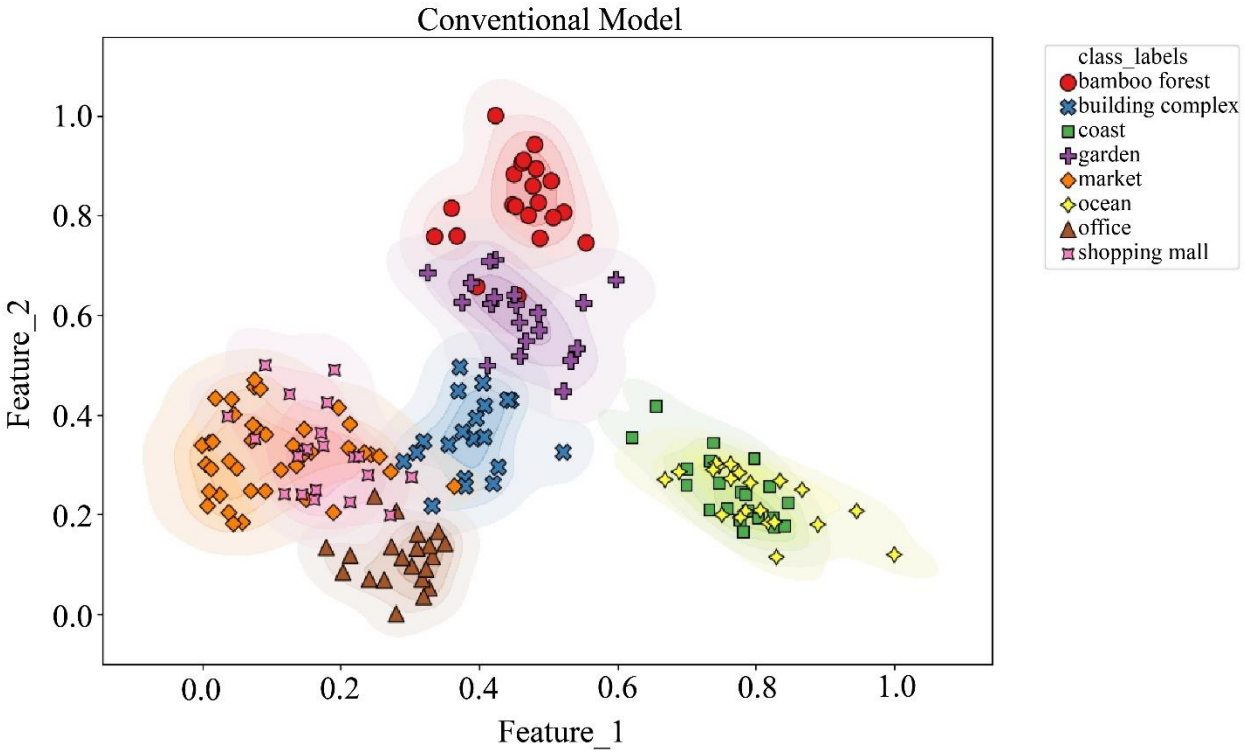


(a) Conventional clustering

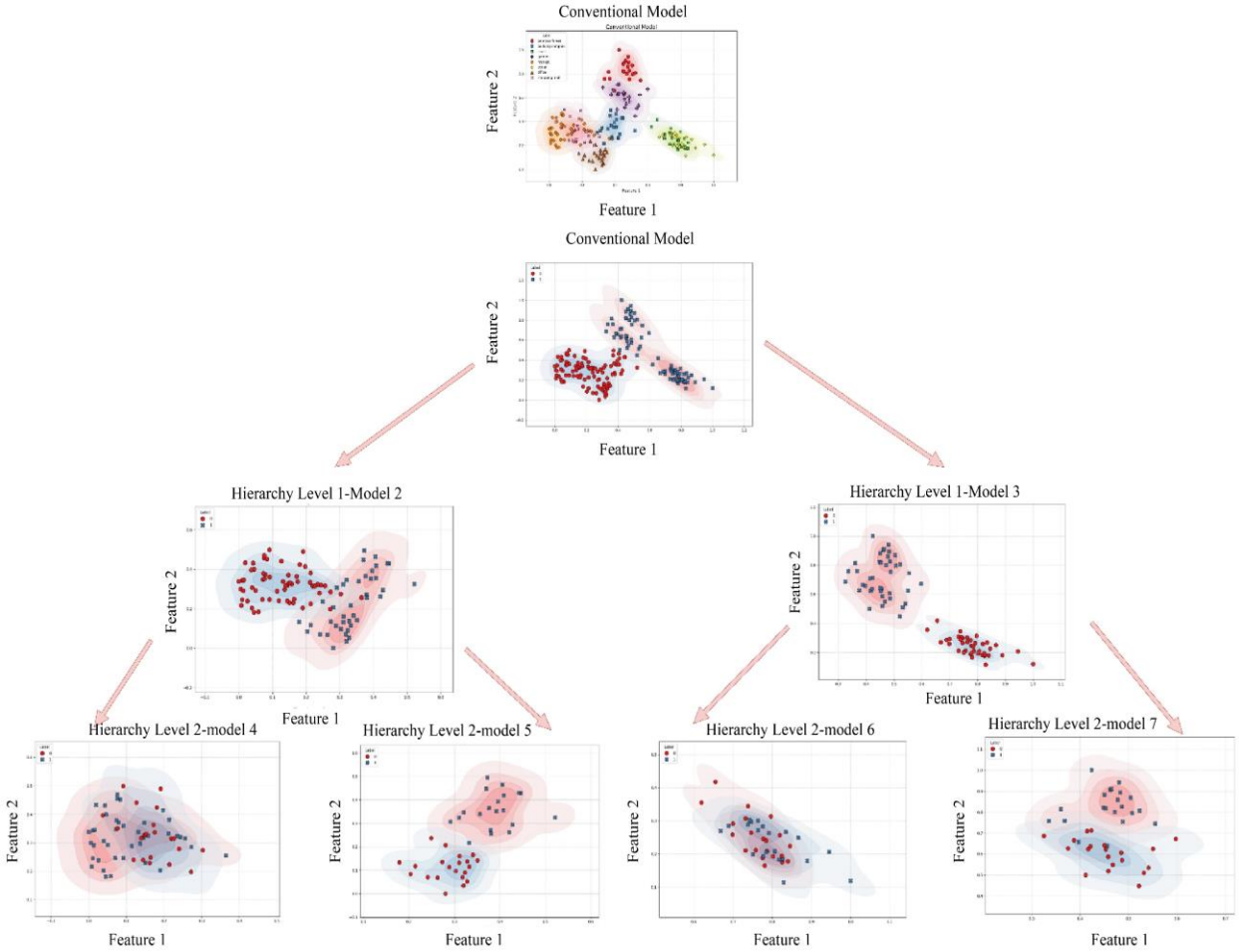


(b) Hierarchical deep clustering

Fig. 1 Conventional v/s Hierarchical deep clustering



(a) Conventional clustering



(b) Deep hierarchical clustering

Fig. 2 Example of conventional v/s Hierarchical deep clustering on 8 classes of the SUN dataset

Hierarchical Deep Clustering
Step 1: Feature Extraction: Each image's feature vector is extracted using a CNN. Step 2: K-Means Clustering: Feature vectors are clustered based on similarity. Step 3: Counting Images Assigned to Each Cluster: The number of images per class assigned to each cluster is computed. Step 4: Majority Voting: Each class is assigned to the cluster where most of its images belong. Step 5: Final Cluster Reassignment: All images of a class are reassigned to the majority cluster.

In conventional methods, a single model is used to classify all test classes as shown in Figure 2 (a). This model is trained in all training classes simultaneously, requiring it to learn a wide range of features, which increases model complexity and makes training more challenging. This also makes the model bulky, and the extracted features start overlapping each other. This results in a reduction in the accuracy of the model. Hierarchical deep clustering adopts a divide-and-conquer approach as used in the Fast Fourier Transform (FFT). Initially, only two broad subclasses are used at hierarchy level 0 for classification, which will make the model simple and yield higher accuracy. This is depicted as model 1 in Figure 1 (b). The hierarchy level-1 specialized model for subclassification of these two classes is used. In the next hierarchy, level-2 further subclasses are partitioned using specialized models focusing on the given subclasses that are depicted as models 2,3,4,5,6,7 in Figure 1 (b). This reduces learning complexity and improves classification accuracy by focusing on fewer

features at a time. To validate the hypothesis, the experiments were conducted on a sample SUN dataset covering eight classes, as shown in Figure 2.

3.1. Steps for Hierarchical Deep Clustering

The proposed Hierarchical deep clustering consists of five steps. Each step is explained in the following brief section.

3.1.1. Step 1: Feature Extraction

Each image $x_{i,j}$ In class, *I* is passed through a CNN to extract its feature representation ($f_{i,j}$) using Equation (1),

$$f_{i,j} = CNN(\ )x_{i,j} \quad (1)$$

$i \in \{1,.....,C\}$ represents the class index.

$j \in \{1,.....,n_i\}$ represents the image index within class *i*.

$f_{i,j} \in \mathbb{R}^d$ is the extracted d-dimensional feature vector for an image $x_{i,j}$.

3.1.2. Step 2: K-Means Clustering

Initialization

Two cluster centroids $\{C_1, C_2\}$ represented by Equation (2) are randomly initialized,

$$C_k \in \mathbb{R}^d, K \in \{1,2\} \quad (2)$$

Represents the centroid of cluster *k*.

Cluster Assignment

Equation (3) assigns each feature vector to the closest cluster based on the Euclidean distance.

$$Cluster(f_{i,j}) = arg\ min_k ||f_{i,j} - c_k||^2 , K \in \{1,2\} \quad (3)$$

Where:

$||f_{i,j} - c_k||^2$ It is the feature vector's squared Euclidean distance from the cluster centroid.

Centroid Update

After assigning all feature vectors to clusters, the centroids S_k are updated as the mean of all assigned points given by Equation (4),

$$S_k = \{ f_{i,j} \mid f_{i,j}\ is\ assigned\ to\ cluster\ c_k\}$$

$$C_k = \frac{1}{|S_k|} \sum_{f_{i,j} \in S_k} f_{i,j} \quad (4)$$

Where:

- S_k Is the set of feature vectors assigned to cluster *k*?
- $|S_k|$ is the number of elements in S_k,
- The new centroid c_k It is the mean of all assigned feature vectors.

The above steps (assignment and update) repeat iteratively until convergence, i.e., until the centroids do not change as given by Equation (5),

$$C_K^{(t+1)} = C_K^{(t)} \quad (5)$$

Where *t* represents the iteration number.

3.1.3. Step 3: Counting Images Assigned to Each Cluster

After convergence, the number of images from each class assigned to each cluster is counted by Equation (6),

$$N_{i,k} = \{ f_{i,j} \mid f_{i,j} \in X_i\ and\ f_{i,j}\ is\ assigned\ to\ cluster\ c_k\} \quad (6)$$

Where:

- The total number of class *i images allocated to the cluster* c_k *is denoted* $N_{i,k}$
- X_i It is the set of all images belonging to class *i*.

3.1.4. Step 4: Majority Voting and Reassignment

Majority Cluster Selection

For each class *i*, the cluster that contains the majority of its images is determined by Equation (7),

$$C_K^* = argmax_k N_{i,k} \quad (7)$$

Where:

- C_K^* Is the cluster that has the most images from class *i*?
- $argmax_k$ finds the cluster index *k* with the highest $N_{i,k}$

3.1.5. Step 5: Final Cluster Reassignment

All images of the class i are reassigned to the majority cluster by Equation (8),

$$Final_{cluster(f_{i,j})} = C_K^*, \forall j \in \{1, \dots . n_i\} \quad (8)$$

Ensuring that all images from the same class belong to a single cluster.

4. Methodology

ZSL has gathered increasing attention, driven by humans' remarkable ability to recognize and classify new visual classes they have never considered before. In the ZSL approach, a labelled dataset is employed to train a model on seen classes, with the primary objective being that the model can identify and classify new classes that they have never considered previously. ZSL significantly reduces annotation costs by leveraging existing knowledge. This study's primary goal is to increase the ZSL model's performance while minimizing its computing complexity. Achieving higher accuracy in zero-

shot image classification is crucial for its practical utility, and this research aims to propose a hierarchical framework that combines primary and secondary classification levels. The hierarchical deep clustering method is proposed to create the hierarchy level of classifier models to reduce complexity and enhance the classification. By utilizing a simplified CNN classifier model 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 complexity. Through rigorous experimentation on benchmark datasets, viz. SUN, AWA2 and CUB, the proposed hierarchical framework is evaluated based on per-class accuracy, training and time complexity metrics, thereby providing insights into its effectiveness in improving ZSL performance and reducing computational burden.

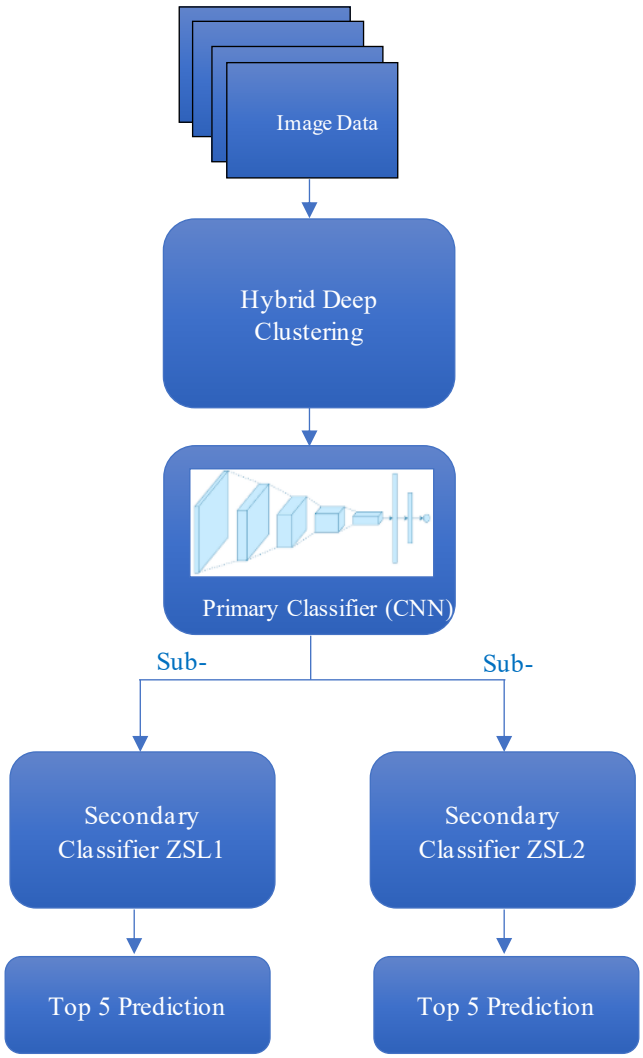


Fig. 3 Architecture of the proposed hierarchical framework for zero-shot image classification

4.1. Proposed Hierarchical Framework

The proposed hierarchical framework is based on primary and secondary classification stages. The primary classification stage relies on a CNN-based classification model trained to categorize images into two broad subclasses, Subclass-0 and Subclass-1, as depicted in Figure 3. Subclass-0 and Subclass-1 are created using hierarchical deep clustering till hierarchy level - 0. It 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 Subclass-0 or Subclass-1 subcategories. Subsequently, the secondary classification stage performs zero-shot image classification using the ZSL model trained using the ResNet-50 architecture. Due to the use of a hierarchy of primary and secondary classifiers, a limited number of classes are used to train each classifier model, leading to fewer feature learning, reducing the complexity of model learning and increasing accuracy.

4.1.1. Primary Classifier - CNN Classification Model

The CNN classification model is a primary classifier that plays a fundamental role in achieving major classification within the proposed system. The proposed framework uses hierarchy level 0 as it categorizes images into two distinct subclasses: Subclass-0 and Subclass-1. Focusing on the CNN classifier designed for the SUN dataset, as depicted in Figure 4. (a), it encompasses key components like convolutional layers, batch normalization layers, max-pooling layers, and dense layers. In CNN architecture, the initial layer typically consists of a convolutional layer responsible for extracting diverse image features through the application of filters. In contrast, a filter is also called a kernel or a feature detector [19]. Sliding the kernel over the image and calculating the sum of element-wise products at each place is the process of convolution, which determines the dot product between the input image and a kernel. The pooling layer reduces the dimensionality of the network, while the fully connected layer follows it, connecting all neurons from the preceding layer. Extensive experimentation involving different combinations of layers with varying kernel sizes has been conducted to optimize the model. After careful refinement, the final configuration has been selected as shown in Figures 4 (a), (b) and (c).

4.1.2. Secondary Classifier - ZSL Model

The ZSL model serves as a secondary classifier that categorizes the classes not encountered during training, e, zero-shot classes. Figure 5. depicts the ZSL model, which is a combination of 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. In the visual semantic embedding space, the visual feature vectors produced by the visual model and the semantic feature vectors produced by the language model using word2vec are linearly mapped. Cosine loss is used to train the ZSL model. These visual and language models collaborate to accomplish zero-shot classification tasks.

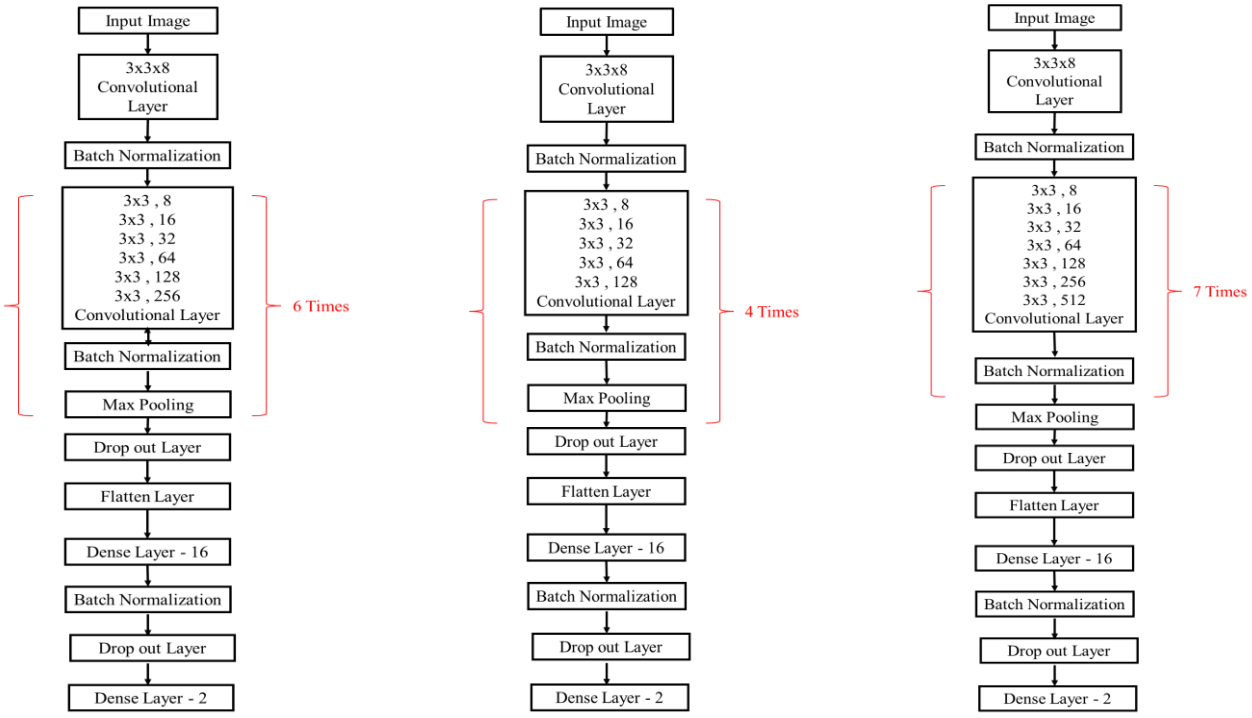


Fig. 4 Primary classifier - CNN classification model – (a) SUN dataset, (b) AWA2 dataset, and (c) CUB dataset

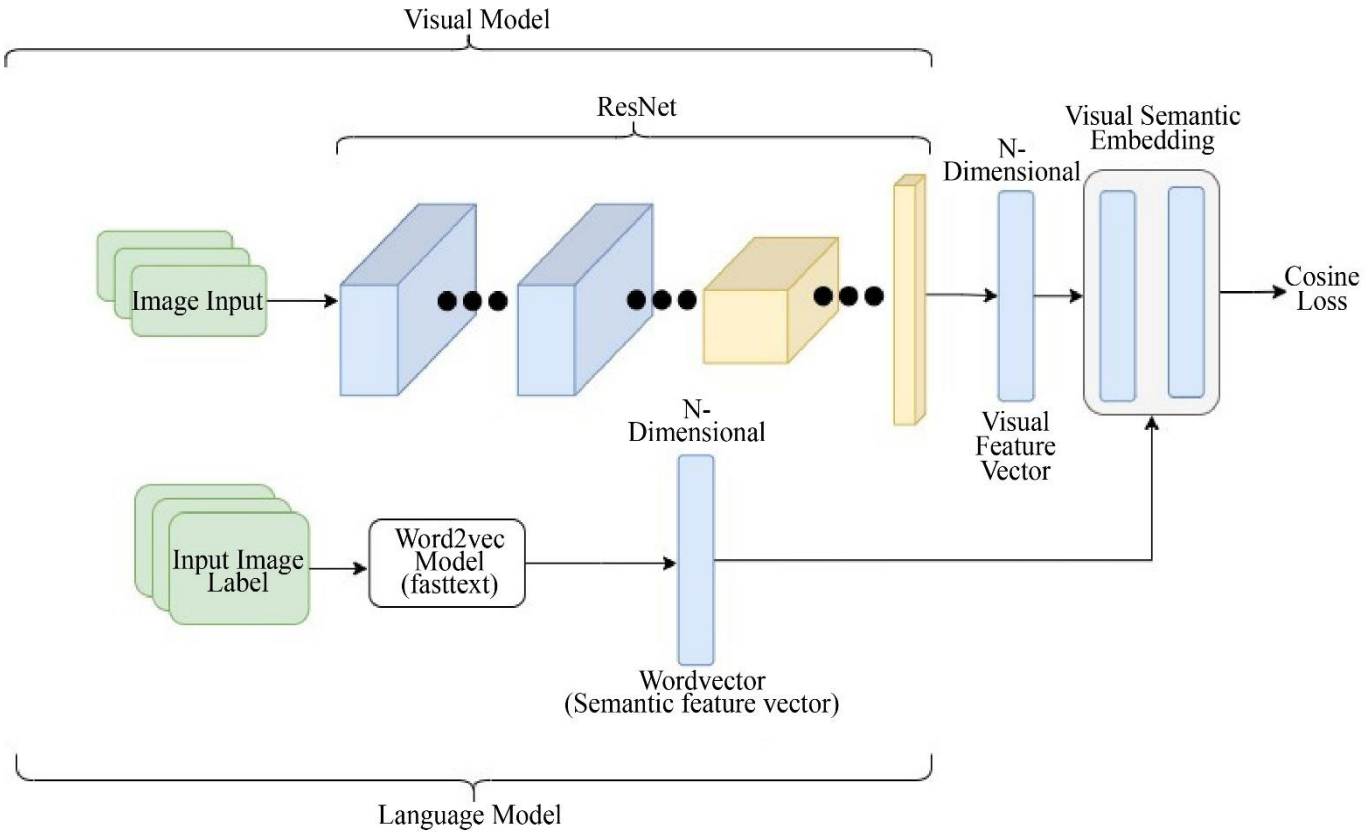


Fig. 5 Secondary classifier - ZSL model

Visual Model

Visual models play a crucial role in extracting visual features through deep CNN. For this approach, the ResNet50 model is selected to extract visual features from images after experimenting with various pre-trained models, such as AlexNet, VGG19, DenseNet, GoogleNet, and ResNet-50. Among these models, ResNet-50 emerges as the most suitable choice for the proposed method. Although experimentation was also conducted with ResNet-101, which yielded better results, it increased the complexity of the model compared to ResNet-50. The proposed approach aims to improve accuracy while reducing complexity. ResNet-50 strikes an optimal balance, offering sufficient accuracy with less computational overhead than the more complex ResNet-101.

Utilizing the pre-trained ResNet-50 model on the ImageNet dataset enhances its capacity to grasp a wide array of high-level visual features. ResNet-50 represents a deep CNN architecture designed for extracting diverse features at low, medium, and high levels of features from images. While stacking layers in a deep network enhances accuracy, it introduces challenges like vanishing and exploding gradients, causing slow convergence or oscillation. To address this, ResNet-50 employs skip connections, a pivotal solution to mitigate vanishing gradient and exploding gradient issues. The ResNet-50 architecture, as shown in Figure 6, consists of several convolutional layers, a pooling layer, and a fully connected layer comprising 50 layers [20]. These layers, including convolutional and max-pooling operations, are instrumental in extracting intricate features from images. Notably, the hallmark of ResNet 50 lies in its skip connections, which strategically skip.

Language Model

Language models play a crucial role in transforming words into word vectors, and the Word2Vec method serves as a prominent approach for language modelling. This algorithm leverages a deep neural network model, trained on substantial text data, to recognize word similarity and predict semantically analogous words. To find word similarity for labels, the model converts each label into a fixed-length vector called an embedding vector.

The fastText open-source library, developed by Facebook's AI Research (FAIR) lab, is used in the language model. FastText efficiently converts words into embedding vectors using the Word2Vec method based on textual data. The labels of classes are transformed into embedding vectors of 300 dimensions. The model exhibits versatility by generating feature vectors of 50, 100, 200, and 300 dimensions, with the experiment opting for a 300-dimensional word vector to capture richer semantic information. This strategic choice recognizes that a higher-dimensional word vector provides more detailed information, enhancing the model's capacity to understand and represent the nuances of language [21].

5. Evaluation Metrics

The proposed hierarchical approach is evaluated based on floating-point operations (FLOPs), training complexity, and time complexity.

5.1. FLOPs

One important indicator for evaluating a model's complexity is its floating-point operations. These operations can include addition, subtraction, division, multiplication, or any other calculation involving a floating-point value. Different layers of convolutional neural networks perform different computational operations based on the type of layer. The following are the formulas for calculating FLOPs for different layers of a CNN.

5.1.1. Convolutional Layer (CL)

The input image is convolved with the kernel in the convolutional layer. FLOPs for the convolutional layer [22] are calculated using Equation (9),

$$FLOPs_{CL} = 2 \times n \times kernel\ shape \times output\ shape \tag{9}$$

$$n = number\ of\ kernel$$

$$Kernel\ shape = W \times H$$

$$Where\ W\ and\ H\ are\ the\ width\ and\ height\ of\ the\ kernel.$$

$$Output\ Shape_{CV} = (M - W + 1)(N - H + 1)$$

5.1.2. Fully Connected Layer

Every node in the output layer is directly connected to every node in the layer above it in the fully connected layer. Equation (10) is used to determine FLOPs for fully linked layers, which are twice the product of the input layer (I) and output layer (O) node counts.

$$FLOPs_{fully\ connected\ layer} = 2 \times\ I \times\ O \tag{10}$$

5.1.3. Pooling Layer

The pooling layer reduces the dimensionality of the input layer. The FLOPs for the pooling layer are calculated by multiplying the height (H), width (W), and depth (D) of the pooling layer. FLOPs for the pooling layer are given by Equation (11),

$$FLOPs_{pooling\ layer} = H \times\ W \times\ D \tag{11}$$

5.1.4. Batch Normalization Layer

The batch normalization layer is used to normalize the outcome of the preceding layers. It enhances learning efficiency and acts as a regularizer to prevent model overfitting. FLOPs for batch normalization are given by Equation (12),

$$FLOPs_{batch\ normaliztion\ layer} = C \times 4 \times output\ shape \tag{12}$$

C is the number of channels in the convolution layer's output

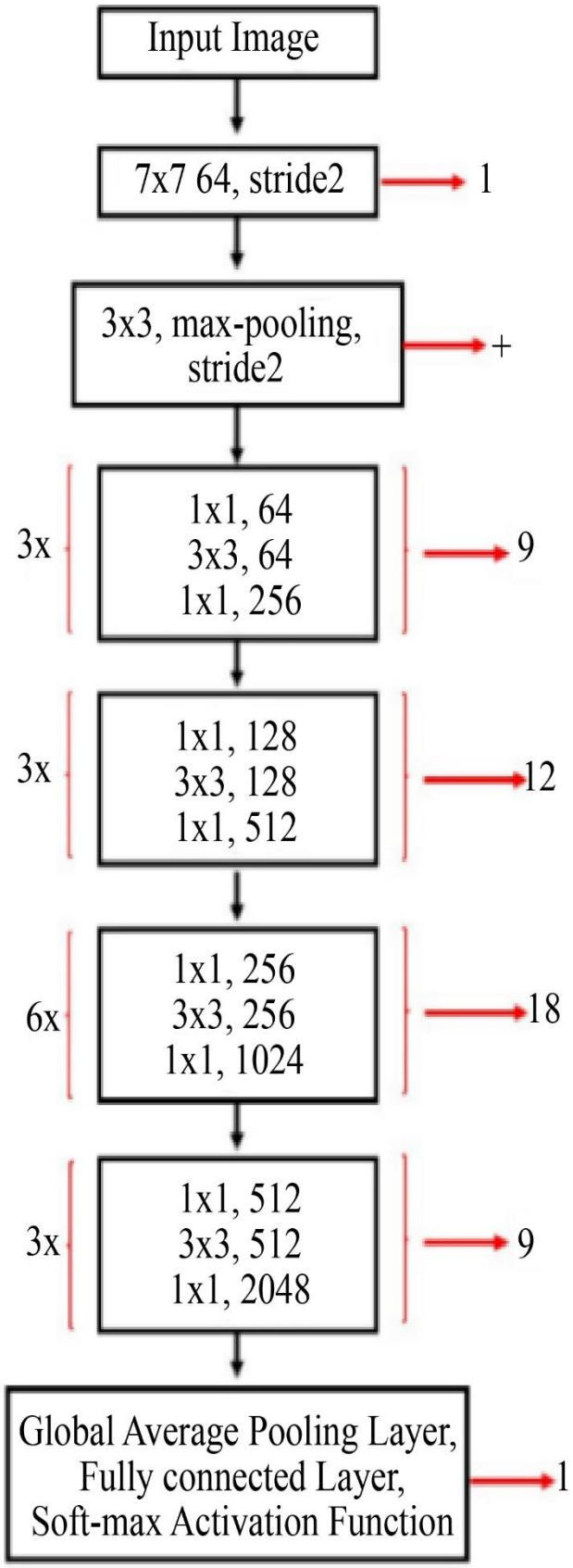


Fig. 6 ResNet-50 architecture [20]

5.2. Training Complexity

The training complexity (*TC)* is measured as the number of computations of the model during training. It relies on the number of FLOPs (*F*), the number of epochs (*E*), and the number of input images (*I*) for training the model. It is given by Equation (13),

$$TrainingComplexity\,(TC) = F \times E \times I \tag{13}$$

The quantity of input images used during training affects a model's training complexity. Consequently, the training complexity is greatly impacted by the dataset's size. The model's training complexity rises with dataset size since it must learn more features from the images.

5.3. Time Complexity

Training and testing durations both affect time complexity. Testing time is the amount of time needed to use the trained model to evaluate a single image from the test dataset, whereas training time is the amount of time needed to fit the model to the training data.

5.4. Per-Class Average Accuracy

Per-class accuracy, also referred to as class-wise accuracy or class-specific accuracy, is a performance metric utilized for the evaluation of classification models, particularly in scenarios involving multi-class classification problems. Per-class Average Accuracy is calculated with Equation (14),

$$Per-class\ Average\ Accuracy = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{Y_{correct\,class}^{class_i}}{Y_{Total\,classes}^{class_i}}\right) \tag{14}$$

The proportion of accurately predicted images to all of the images in each class is calculated. The average accuracy for each class in the dataset is then obtained by aggregating it for all N classes.

6. Results and Discussion

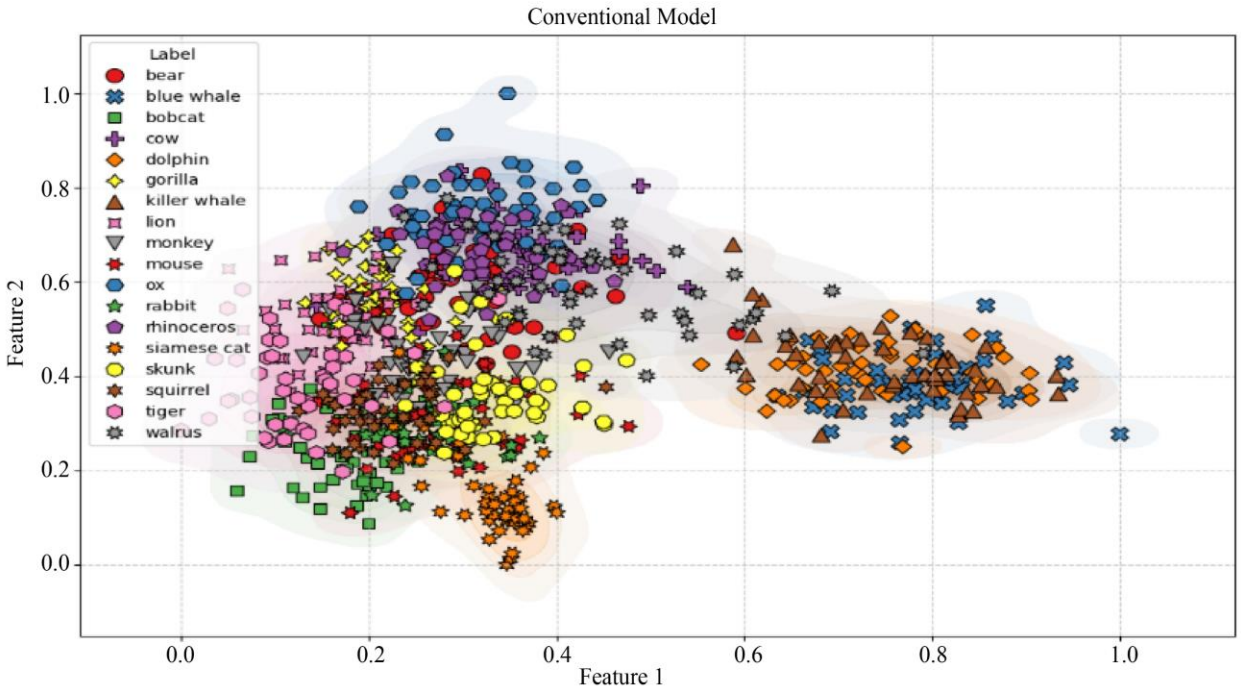
The hierarchical framework of ZSL is a novel approach to classifying unseen data by using a hierarchy of classifiers. It is implemented using a primary classifier (CNN classifier) to broadly classify unseen classes into Subclass-0 or Subclass-1. Each subclass's secondary classifier (ZSL Model) is used for zero-shot image classification. The hierarchical ZSL framework and the EZSL framework [11] are compared based on accuracy, training complexity, and training-testing time. The implementation and evaluation of results are performed on the SUN, AWA2 and CUB datasets.

6.1. Dataset

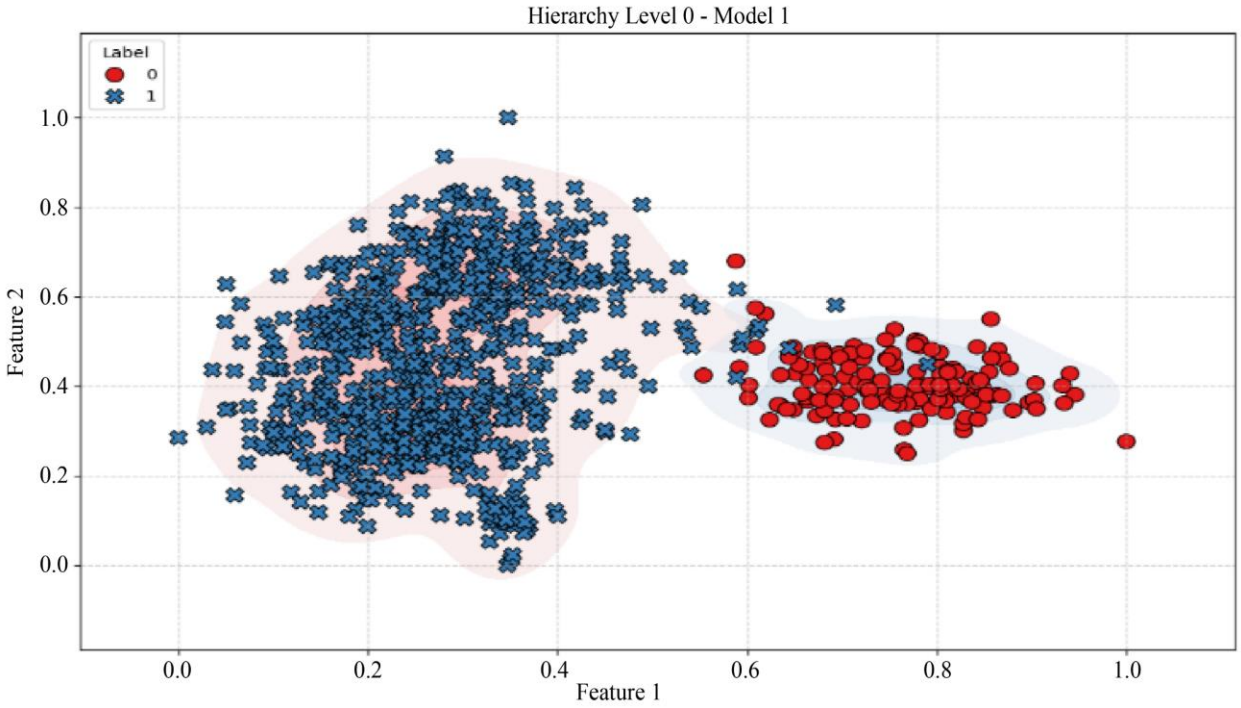
Three benchmark datasets are used to assess the efficacy of the suggested hierarchical framework: SUN, AWA2, and CUB. The SUN dataset consists of 14,340 scene images across 645 Training classes, with 72 zero-shot test classes.

Table 1. The statistics of the three benchmark datasets were employed for the experiments

Dataset	Seen classes	Unseen classes	Seen classes	Unseen classes
	Benchmark split [27]		Proposed approach split	
AWA2	40	10	18	10
SUN	645	72	60	72
CUB	150	50	101	50



(a) Conventional clustering



(b) Hierarchical deep clustering

Fig. 7 Conventional v/s Hierarchical deep clustering – AWA2 dataset

The hierarchical approach significantly reduces the training set to just 60 classes while maintaining classification effectiveness. Similarly, the AWA2 dataset, which includes 37,322 images of animals categorized into 50 classes (40 for training and 10 for zero-shot testing), undergoes a reduction in training classes from 40 to 18. There are 11,788 bird images from 200 species in the CUB dataset (150 for training and 50 for testing). The approach narrows the training set from 150 to 101 classes. The detailed Statistics of the Benchmark split with the proposed split are shown in Table 1. This reduction ensures the training process is more efficient while preserving the necessary semantic information for classification. The method chooses training classes according to how semantically similar they are to test classes in order to accomplish this reduction. The original training set and the test set are defined as $C_T = \{T_1, T_2, \ldots . T_{40}\}$, and as $C_S = \{S_1, S_2, \ldots \ldots \ldots . S_{10}\}$ respectively. For each test class S_j a similarity score $S(T_i, S_j)$ is computed for all $T_i \in C_T$ and $S_j \in C_S$.

Then the model is retained only on the top 10 most similar training classes for each test class, forming a refined subset. $C_T, \subseteq C_T$. Any training class that does not appear in the top 10 for any test class is discarded, resulting in a final filtered set of training classes (e.g., 18 for AWA2). This selection process ensures that only the most semantically relevant training classes contribute to the learning process, reducing unnecessary complexity.

This approach offers several advantages. By eliminating irrelevant training classes, the model complexity is reduced, leading to faster training and improved generalization. The computational complexity of classification is optimized from $O(n.40)$ instead of $O(n.k)$where k is the reduced number of training classes (e.g., 18 for AWA2). Additionally, by focusing solely on semantically similar classes, the model avoids overfitting to unrelated categories and learns more meaningful distinctions. As a result, classification performance remains effective while requiring fewer training resources. The novelty of this approach lies in its ability to successfully classify all test classes despite using a significantly reduced training set. By strategically filtering out less relevant classes, this method lowers training complexity and computational cost and enhances learning efficiency without compromising classification accuracy. After reducing the training classes based on semantic similarity, Hierarchical deep clustering is applied to all reduced training classes to divide the reduced training classes into subclass-0 and subclass-1, as shown in Figure 7(b). AWA2 dataset reduced training classes are shown in Figure 7(a), and Hierarchy level – 0 is shown in Figure 7(b) for the AWA2 dataset.

6.2. Results Analysis of Training Complexity and Accuracy

The proposed framework is tested on a laptop running Windows 11 with an AMD Ryzen 9 5900HX processor and an NVIDIA GeForce RTX 3060 GPU operating at 3.30 GHz with 16MB of RAM. The Adam optimizer and categorical cross-entropy loss function optimization are employed in the principal classifier model. ResNet-50, which has been pretrained on the ImageNet dataset, is used to create the secondary classifier visual model. The language model of the secondary classifier uses the fastText open-source library to generate embedding vectors and find word similarity. The secondary classifier is trained with the Adam optimizer and cosine loss function for optimization. Early stopping criteria are used for primary and secondary classifiers to prevent overfitting.

6.2.1. Training Complexity

The number of FLOPs, epochs, and input images needed to train the model is used to assess training complexity.

Table 2. FLOPs evaluation of the EZSL model

Sr No.	Layers of the EZSL Model	FLOPs
1.	ResNet 50 Pretrained Model [23]	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 EZSL Model		$\mathbf{3.802 \times 10^6}$

Table 3. FLOPs evaluation of the primary classifier model for the SUN dataset

Sr No.	Layers of a CNN-based classifier model for the 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.	Drop-out layer	0
20.	Batch Normalization Layer – 7	1024
21.	Dense Layer - 1	204,800
22.	Drop-out layer	0
23.	Dense Layer - 2	64
Total Number of FLOPs of the primary classifier model for the SUN dataset.		**476.682×10^6**

Table 4. FLOPs evaluation of the primary classifier model for the AWA2 dataset

Sr No.	Layers of a CNN-based classifier model for the 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 of the primary classifier model for the AWA2 dataset		**417.774×10^6**

Table 5. FLOPs evaluation of the primary classifier model for the CUB dataset

Sr No.	Layers of a CNN-based classifier model for the CUB 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.	Convolution layer – 7	21,233,664
19.	Batch Normalization Layer – 7	18,432
20.	Flatten layer	0
21.	Drop out layer	0
22.	Dense Layer - 1	589,824
23.	Drop out layer	0
24.	Dense Layer - 2	64
Total Number of FLOPs of the primary classifier model for the CUB dataset.		**497.934×10^6**

Table 6. Training complexity evaluation of the EZSL Model

Framework	Dataset	Number of FLOPs	Number of epochs	Number of input images	Training Complexity - (number of computations)
EZSL Model	SUN	3.802×10^9	19	11735	**84.771×10^{13}**
	AWA2	3.802×10^9	15	24270	**138.411×10^{13}**
	CUB	3.802×10^9	80	6,060	**184.224×10^{13}**

Table 7. Training complexity evaluation of the hierarchical framework model for the SUN dataset

Framework	Models	Number of FLOPs	Number of epochs	Number of input images	Training Complexity- (number of computations)
The proposed Hierarchical Framework model – SUN dataset	Primary classifier Model	476.682×10^6	18	2715	2.329×10^{13}
	Subclass-0 model	3.802×10^9	25	991	9.419×10^{13}
	Subclass-1 model	3.802×10^9	23	454	3.970×10^{13}
Total Training Complexity of Hierarchical Framework Model for the SUN Dataset					**15.718×10^{13}**

To calculate training complexity, FLOPs for the layers of the EZSL model and the CNN-based Primary classifier model for the SUN, AWA2 and CUB datasets are calculated using equations (9) to (12) and shown in Tables 2, 3, and 4. FLOPs calculated in Table 2, Table 3, Table 4 and Table 5 are used in evaluating the training complexity of the EZSL Model and the Proposed Hierarchical framework. Training complexity evaluation of the EZSL Model for SUN, AWA2 and CUB datasets is shown in Table 6. The training complexity of the individual model is calculated using Equation(13).

Tables 7, 8 and 9 show the training complexity evaluation of the hierarchical framework for the SUN, AWA2 and CUB datasets. The hierarchical framework model consists of three models: the primary classifier model, the Subclass-0 model and the Subclass-1 model. The total training complexity of all the models determines the training complexity of the hierarchical framework model. The number of input images, a crucial hyperparameter, significantly influences the training complexity of a model. According to Table 6, the EZSL model for the SUN dataset is trained on 11,735 images. In contrast, the Hierarchical framework model, as shown in Table 7, is trained on a total of 4,160 images (the sum of 2,715, 991, and 454 images), a reduced number of images due to the hierarchy structure. Whereas the EZSL model is trained on a larger dataset compared to the Hierarchical framework model. Therefore, the number of computations, i.e. the training complexity, is higher for the EZSL model. The performance analysis of the proposed hierarchical strategy in terms of training complexity in comparison to the EZSL approach is shown in Table 10. The findings show that the hierarchical model's training complexity was reduced by an average of about 80%. The key reason for the improvement in training complexity is the reduced number of computations while training the model.

As the number of training classes is decreased, fewer computations are required because of the hierarchical structure. Tables 11 and 12 present the performance analysis of the proposed framework based on training and testing time. The hierarchical framework demonstrates a 25% improvement in training time compared to the EZSL framework. Despite involving three models for training, the hierarchical framework achieves lower training time due to its structured learning process. Additionally, the hierarchical structure reduces testing time, making the system faster for classification. Overall, the time complexity of the proposed framework is improved by approximately 23%. While hierarchy increases the number of models for training, it simultaneously reduces complexity by training models on subclasses.

6.2.2. Accuracy

The results of the proposed framework in Figures 8, 9, and 10 illustrate the top 5 predictions for the SUN, AWA2, and CUB datasets, respectively. A detailed zero-shot image classification report of the proposed hierarchical framework model for the AWA2 dataset is shown in Table 13. The accuracy, recall, and F1-score metrics provide insightful information about the model's performance for every class score.

Table 8. Training complexities evaluation of the hierarchical Framework model for the AWA2 dataset

Framework	Models	Number of FLOPs	Number of epochs	Number of input images	Training Complexity - (number of computations)
The proposed Hierarchical Framework model – AWA2 dataset	Primary classifier Model	417.774×10^6	66	9767	26.930×10^{13}
	Sub class-1 model	3.802×10^9	11	3278	1.371×10^{13}
	Sub class-2 model	3.802×10^9	10	6408	2.436×10^{13}
Total Training Complexity of Hierarchical Framework Model for the AWA2 Dataset					$\mathbf{30.737 \times 10^{13}}$

Table 9. Training complexities evaluation of the hierarchical framework model for the CUB dataset

Framework	Models	Number of FLOPs	Number of epochs	Number of input images	Training Complexity - (number of computations)
The proposed Hierarchical Framework model – CUB dataset	Primary classifier Model	497.934×10^6	100	6060	30.175×10^{13}
	Sub class-1 model	3.802×10^9	11	3278	1.371×10^{13}
	Sub class-2 model	3.802×10^9	10	6408	2.436×10^{13}
Total Training Complexity of Hierarchical Framework Model for the CUB Dataset					$\mathbf{33.982 \times 10^{13}}$

Table 10. Performance analysis of the proposed hierarchical framework based on training complexity

Sr.No	Dataset	Training Complexity –EZSL Framework	Training Complexity – Hierarchical Framework	Reduction in Training Complexity (%)
1.	SUN	84.771×10^{13}	$\mathbf{15.718 \times 10^{13}}$	81.46
2.	AwA2	138.411×10^{13}	$\mathbf{30.737 \times 10^{13}}$	77.79
3.	CUB	184.224×10^{13}	$\mathbf{33.982 \times 10^{13}}$	81.55
Average Reduction in Training Complexity				**80.26**

Table 11. Performance analysis of the proposed hierarchical framework based on training time

Sr.No	Dataset	Training Time (Seconds) –EZSL Framework	Training Time (Seconds) – Hierarchical Framework	Reduction in Training Time (%)
1.	SUN	809	**656**	18.91
2.	AwA2	2854	**1953**	31.56
3.	CUB	2356	**1717**	27.12
Average Reduction in Training Time				**25.86**

Table 12. Performance analysis of the proposed hierarchical framework based on testing time

Sr.No	Dataset	Testing Time –EZSL Framework	Testing Time – Hierarchical Framework	Reduction in Testing Time (%)
1.	SUN	0.260	**0.224**	13.84
2.	AwA2	0.391	**0.297**	24.04
3.	CUB	0.356	**0.245**	31.17
Average Reduction in Testing Time				**23.02**

Actual Label: Farm
Top 5 Predicted labels: field, meadow, garden, fields, farm, back-garden

Actual Label: Forest
Top 5 Predicted labels: Forest, Meadow, Treeline, Forestry, Woodlot

Fig. 8 Results hierarchical framework - SUN dataset

Actual Label: Raccoon
Top 5 Predicted labels: Raccoon, Bobcat, Bobcats, Cat, Housecat

Actual Label: Leopard
Top 5 Predicted labels: Leopard, Tiger, BobCat, Tigers, Tigress

Fig. 9 Results hierarchical framework – AWA2 dataset

Actual Label: Laysan_Albatross
Top 5 Predicted labels: Laysan_Albatross, Pied_billed_Grebe, Palm_Warbler, Ivory_Gull, Red_eyed_Vireo

Actual Label: Parakeet_Auklet
Top 5 Predicted labels: Parakeet_Auklet, Rhinoceros_Auklet, White_breasted_Kingfisher, Scott_Oriole, Red_eyed_Vireo

Fig. 10 Results hierarchical framework – CUB dataset

Table 13. Zero-shot image classification report of the proposed hierarchical framework for the AWA2 dataset

Class. No	Class Name	Precision	Recall	F1-score	Support
0.	Chimpanzee	0.64	0.73	0.68	728
1.	Giant_panda	0.56	0.38	0.45	874
2.	Hippopotamus	0.59	0.62	0.60	684
3.	Humpback_whale	0.70	0.98	0.81	709
4.	Leopard	0.63	0.62	0.63	720
5.	Persian_cat	0.68	0.90	0.78	747
6.	Pig	0.39	0.25	0.30	713
7.	Raccoon	0.55	0.67	0.60	512
8.	Rat	0.44	0.80	0.57	310
9.	Seal	0.54	0.27	0.36	988
	Accuracy			0.59	6985

Table 14. Performance analysis of the proposed hierarchical framework based on per-class accuracy

Sr.No	Dataset	Per-class Accuracy – EZSL Framework	Per-class Accuracy - Hierarchical Framework	Improvement in Per-class Accuracy (%)
1.	SUN	39.5	**50.6**	28.10
2.	AwA2	44.7	**59.9**	34.00
3.	CUB	43.2	**52.93**	22.52
Average Improvement in Per-class Accuracy				**28.21**

Table 15. Comparison of the performance of the proposed framework with other existing approaches using per-class accuracy

Sr No.	Approach	SUN	AWA2	CUB
1	EZSL [11]	39.5	44.7	43.2
2	SAE [24]	40.3	54.1	33.3
3	DAP [7]	39.9	46.1	40.0
4	DeViSE [10]	56.5	59.7	52.0
5	ESZSL [25]	54.5	58.6	53.9
6	LATEM [26]	55.3	55.8	49.3
7	JG-ZSL [13]	60.3	69.4	52.1
8	HFM [15]	53.8	65.5	69.5
9	SYNC [28]	56.3	46.6	55.6
	Proposed Hierarchical Framework for Zero-Shot Image Classification	50.6	59.9	52.9

Additionally, Persian_cat (class 5) has a strong F1-score, recall, and precision. Pigs (class 6) score lower than other classes in terms of precision, recall, and F1. The overall accuracy is 59.9% for the AWA2 dataset. Table 14 presents the improvement in per-class accuracy achieved by the proposed hierarchical framework for these datasets.

The results indicate an average improvement of 28% compared to the EZSL framework in Table 14. The Hierarchical framework is compared with other existing approaches in Table 15. The proposed hierarchical model can classify all zero-shot classes with better accuracy, even though it was trained on a smaller dataset. This improvement is due to the use of a hierarchy of primary and secondary classifiers. By training these classifiers on fewer classes, the model needs to learn fewer features, which reduces the likelihood of misclassification and thus increases accuracy.

Comparing existing methods to the proposed framework, the DAP [7] and JG-ZSL [13] methods use manually crafted attributes as visual features, while SYNC [28] employs colour histograms, SIFT, and PHOG as shallow visual features. These visual features are provided with the dataset. In contrast, the proposed framework uses a deep CNN, specifically the ResNet-50 model, for visual feature extraction. ResNet-50 model extracts diverse high-level features from the images, thereby eliminating the need for handcrafted features. However, the HFM [15] and JG-ZSL [13] methods achieve better accuracy than the proposed hierarchical framework. The HFM approach utilizes two variational autoencoders, while the JG-ZSL approach employs a generative adversarial network. Both variational autoencoders [15] and generative adversarial networks [13] are computationally more complex, as variational autoencoder uses encoder and decoder networks, and generative adversarial networks use a generator and a discriminator. Our proposed hierarchical framework, on the other hand, provides optimal accuracy with reduced computational complexity.

7. Conclusion

The ZSL model gets complicated and slow when there are many classes. The proposed novel hierarchical framework that uses a dissimation technique is offered to overcome this problem. This hierarchical framework employs less complex but more accurate classifiers at each level of the hierarchy, similar to the Fast Fourier Transform (FFT) approach. The quantitative analysis on three benchmark datasets (AWA2, SUN, CUB) shows that our proposed framework reduces training complexity by 80.26%, training time by 25.86% and testing time by 23.02% without compromising the accuracy of ZSL.

This paper primarily contributes a hierarchical framework and hierarchical deep clustering, which hold the potential to significantly increase the use of ZSL by providing an excellent tradeoff between complexity and accuracy. The proposed hierarchical framework uses a CNN-based primary classifier and ResNet 50, along with the fastText language model, to form a secondary classifier that introduces a two-stage hierarchy. The benefits of this framework hold promise for future applications, especially as datasets (number of classes) continue to grow in size and complexity. Future iterations could also explore a multilevel hierarchy to enhance performance further.

References

[1] Xiaohong Sun, Jinan Gu, and Hongying Sun, "Research Progress of Zero-Shot Learning," *Applied Intelligence*, vol. 51, no. 6, pp. 3600-3614, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[2] Wei Wang et al., "A Survey of Zero-Shot Learning: Settings, Methods, and Applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1-37, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[3] Genevieve Patterson, Xu Chen, and James Hays, Sun Attribute Database: Discovering, Annotating, and Recognizing Scene Attributes, SUN Attribute Dataset, 2011. [Online]. Available: https://cs.brown.edu/~gmpatter/sunattributes.html

[4] Animals with Attributes 2: A Free Dataset for Attribute-Based Classification and Zero-Shot Learning, Institute of Science and Technology Austria (ISTA), 2016. [Online]. Available :https://cvml.ista.ac.at/AwA2/

[5] Catherine Wah et al., "*The Caltech-Ucsd Birds-200-2011 Dataset*," California Institute of Technology, 2011. [Google Scholar] [Publisher Link]

[6] Christoph H. Lampert, Hannes Nickisch, and Stefan Harmeling, "Learning to Detect Unseen Object Classes BY Between-Class Attribute Transfer," *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Miami, FL, USA, pp. 951-958, 2009. [CrossRef] [Google Scholar] [Publisher Link]

[7] Christoph H. Lampert, Hannes Nickisch, and Stefan 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, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[8] Zeynep Akata et al., "Label-Embedding for Image Classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 7, pp. 1425-1438, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[9] Xuesong Wang et al., "Zero-Shot Learning Based on Deep Weighted Attribute Prediction," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 8, pp. 2948-2957, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[10] Andrea Frome et al., "Devise: A Deep Visual-Semantic Embedding Model," *Advances in Neural Information Processing Systems (NIPS 2013)*, vol. 26, pp. 2121-2129, 2013. [Google Scholar] [Publisher Link]

[11] Ansari Shaista Khanam, and Poonam. N. Sonar, "Enhanced Zero-shot Learning using Deep Neural Network ResNet50," *2023 4th International Conference for Emerging Technology (INCET)*, Belgaum, India, 2023, pp. 1-6, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[12] Yongqin Xian et al., "Feature Generating Networks for Zero-Shot Learning," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, pp. 5542-5551, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[13] Minwan Zhang et al., "Zero-Shot Learning with Joint Generative Adversarial Networks," *Electronics*, vol. 12, no. 10, pp. 1-18, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[14] Varun Khare et al., "A Generative Framework for ZSL with Adversarial Domain Adaptation," *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, Snowmass, CO, USA, pp. 3090-3099, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[15] Fadi Al Machot, Mohib Ullah, and Habib Ullah, "HFM: A Hybrid Feature Model Based on Conditional Auto Encoders for Zero-Shot Learning," *Journal of Imaging*, vol. 8, no. 6, pp. 1-12, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[16] Rich Lee, and Ing-Yi Chen, "The Time Complexity Analysis of Neural Network Model Configurations," *2020 International Conference on Mathematics and Computers in Science and Engineering (MACISE)*, Madrid, Spain, pp. 178-183, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[17] Pedro Freire et al., "Computational Complexity Evaluation of Neural Network Applications in Signal Processing," *arXiv Preprint*, pp. 1-25, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[18] Bhoomi Shah, and Hetal Bhavsar, "Time Complexity in Deep Learning Models," *Procedia Computer Science*, vol. 215, pp. 202-210, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[19] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi, "Understanding of a Convolutional Neural Network," *2017 International Conference on Engineering and Technology (ICET)*, Antalya, Turkey, pp. 1-6, 2017. [CrossRef] [Publisher Link]

[20] Sheldon Mascarenhas, and Mukul Agarwal, "A Comparison between VGG16, VGG19 and ResNet50 Architecture Frameworks for Image Classification," *2021 International Conference on Disruptive Technologies for Multi-Disciplinary Research and Applications (CENTCON)*, Bengaluru, India, pp. 96-99, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[21] Armand Joulin et al., "Bag of Tricks for Efficient Text Classification," *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistic*, Valencia, Spain, vol. 2, pp. 427-431, 2017. [Google Scholar] [Publisher Link]

[22] Lecture 41: Space and Computational Complexity in DNN, Deep Learning For Visual Computing - IITKGP, YouTube, 2018. [Online] Available at: https://www.youtube.com/@deeplearningforvisualcompu3823

[23] Kaiming He et al., "Deep Residual Learning for Image Recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, pp. 770-778, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[24] Elyor Kodirov, Tao Xiang, and Shaogang Gong, "Semantic Autoencoder for Zero-Shot Learning," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, pp. 4447-4456, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[25] Bernardino Romera-Paredes, and Philip Torr, "An Embarrassingly Simple Approach to Zero-Shot Learning," *Proceedings of the 32nd International Conference on Machine Learning, PMLR*, vol. 37, pp. 2152-2161, 2015. [Google Scholar] [Publisher Link]

[26] Yongqin Xian et al., "Latent Embeddings for Zero-Shot Classification," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, pp. 69-77, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[27] Yongqin Xian et al., "Zero-Shot Learning-A Comprehensive Evaluation of the Good, the Bad and the Ugly," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 9, pp. 2251-2265, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[28] Soravit Changpinyo et al., "Synthesized Classifiers for Zero-Shot Learning," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, pp. 5327-5336, 2016. [CrossRef] [Google Scholar] [Publisher Link]