University of Padova

Department of Mathematics "Tullio Levi-Civita"

Master Thesis in Computer Science

Enhancing Image Classification with Colorization Techniques and Ensemble Learning

Supervisor
Prof. Loris Nanni
University of Padova

Master Candidate
Hala Mohamed Mostafa Abousobh

Student ID
2041590

Academic Year
2023-2024

"I suppose therefore that all things I see are illusions; I believe that nothing has ever existed of everything my lying memory tells me. I think I have no senses. I believe that body, shape, extension, motion, location are functions. What is there then that can be taken as true? Perhaps only this one thing, that nothing at all is certain."
— Rene Descartes

Abstract

In recent decades, Artificial Intelligence systems have increasingly achieved and surpassed human-level performance in a variety of complex tasks. Despite their success, the intricate and non-linear structures of deep learning models often make them opaque and challenging to interpret. This thesis presents an innovative automated system for the classification of planktic foraminifera at the species level and extends this methodology to the classification of satellite images from the EuroSAT dataset. The system leverages advanced deep learning techniques, including Generative Adversarial Networks (GANs) and U-Net-based autoencoders.

Initially, the foraminifera dataset, comprising 1437 groups of sixteen grayscale images (one group for each specimen), is converted to RGB images through various processing methods. Similarly, the EuroSAT dataset, based on Sentinel-2 satellite images and including 13 spectral bands across 10 classes with a total of 27,000 labeled and geo-referenced images, is also converted to RGB images through diverse processing methods. These newly colored RGB images from both datasets are then classified using transfer learning.

The RGB images are fed into a set of Convolutional Neural Networks (CNNs) organized in an Ensemble Learning (EL) environment. The ensemble is built by training different networks using diverse approaches for creating the RGB images, supporting the classifiers to enhance performance. This study demonstrates that an ensemble of CNN models trained on the newly colored RGB images from both datasets improves the system's performance compared to other state-of-the-art approaches. The main focus of this thesis is to introduce multiple colorization methods that differ from current cutting-edge techniques

Keywords: Generative Adversarial Network, **Ensemble Learning**, **Transfer Learning**, **Plankton Classification**, **EuroSAT Classification**, **Sentinel-2 Satellite Images**.

Contents

Listing of figures

Listing of tables

Listing of acronyms

NN	Neural Network
ANN	Artificial Neural Network
EL	Ensembel Learning
GAN	Generative Adversarial Network
CNN	Convlutional Neural Network
SGDM	Stochastic Gradient Gescent with Momentum
DNN	Deep Neural Networks
DWT	Discrete Wavelet Transform
RMSE	Root Mean Square Error
DCT	Discrete Cosine Transform
ReLU	Rectified Linear Unit
TP	True Positives
TN	True Negatives
FP	False Positive
FN	False Negative

1 Introduction

1.1 OVERVIEW

The landscape of image classification has been revolutionized by rapid advancements in hardware and the deployment of Deep Learning techniques. These technologies, inherently repetitive and non-creative, are perfectly suited for automation. Convolutional Neural Networks (CNNs) stand out among deep learning models for their exceptional efficacy and efficiency in image classification, as demonstrated in numerous studies. However, the stochastic nature of neural networks (NNs) can introduce variability in their results[3]. Ensemble Learning (EL)[4] effectively addresses this challenge by combining outputs from different heuristic algorithms, thereby enhancing performance and consistency and reducing individual model variability. This thesis introduces an innovative application of EL for the classification of planktic foraminifera and extends this approach to classify images from the EuroSAT dataset. Planktic foraminifera serve as paleo-environmental bioindicators, with their radiocarbon measurements providing insights into historical environmental conditions such as global ice volume, temperature, salinity, pH, and nutrient content . Traditionally, the classification of foraminifera has been a labor-intensive and time-consuming task performed by large teams of experts. Since the early 1990s [5], efforts have been made to automate this process. Despite significant advancements, many methods still require considerable human oversight. The problem with classifying foraminifera images lies in their complexity and variability. Foraminifera come in numerous species with subtle morphological differences that can be challenging to distinguish even for trained experts. The images are grayscale and taken from different angles, adding to the difficulty. Accurately classifying these images is crucial for paleoceanography and climate research, as it helps scientists reconstruct past marine environments and understand historical climate changes. Similarly, the EuroSAT dataset poses its own set of challenges. Based on Sentinel-2 satellite images[6], this dataset includes 13 spectral bands representing various land cover types such as forests, urban areas, and water bodies. The images are geo-referenced and come in different resolutions and conditions, making the classification task complex. The multi-spectral nature of the data adds another layer of complexity, as each spectral band provides unique information that must be effectively integrated. Accurate classification of these images is essential for environmental monitoring, urban planning, and agricultural management. Understanding land cover changes over time can inform policy decisions and help address environmental issues.Ensemble Learning (EL) plays a crucial role in enhancing the performance of these classification tasks. EL involves combining multiple models to produce a more robust and accurate prediction than any single model alone. By aggregating the outputs of different models, EL can mitigate the weaknesses of individual models and exploit their strengths. This approach not only improves accuracy but also provides higher consistency and reliability in predictions. The RGB images from both datasets are fed into a set of CNNs organized within an Ensemble Learning framework. This ensemble is developed by training various networks with different methods for creating the RGB images. The findings indicate that an ensemble of CNNmodels trained on differently processed RGB images from both datasets significantly improves performance compared to current state-of-the-art methods. Additionally, the proposed system surpasses human experts in classification accuracy, showcasing the effectiveness of the employed techniques.

1.2 HISTORICAL BACKGROUND

Image colorization, the process of adding color to grayscale images, has evolved significantly since the inception of photography. This challenge has existed since the advent of photography, prompting numerous solutions over time. Initially, manual colorization involved physically inpainting images with watercolors, oils, or dyes and fixing them with heat, a method first popularized by Hippolyte Bayard between 1840-1845 [7]. Despite advancements in monochromatic photography, such as the introduction of Kodachrome color reversal film by Kodak in 1935 [8], hand-colorization remained prevalent even into the digital age. The 1970s saw the emergence of digital colorization techniques, with notable contributions by Wilson Markle, who colorized the original moon landing footage using a method of assigning and recalibrating color values to grayscale shades [9]. Modern tools like Adobe Photoshop allow for high-quality digital colorization, though it still requires considerable manual effort.

Hint-based colorization methods, such as those proposed by Levin et al. [10], involve adding scribbles to guide the colorization process, leveraging the idea that neighboring pixels with similar intensities should have similar colors. This approach allows users to see immediate results and adjust as needed, though it still demands careful color palette selection. Welsh et al. [11] advanced this concept by developing a method that transfers colors between similar images based on luminance and texture information. Example-based colorization [12], introduced by Irony et al., combines previous techniques, using a reference image segmented into common color areas, which then informs the colorization of the target image.

Automatic colorization, enabled by deep learning, represents a significant leap forward. Deep neural networks (DNNs) trained on vast datasets can automatically colorize images by recognizing and applying common color patterns. One prominent model [7], the Colorful Image Colorization model, uses a convolutional neural network and class rebalancing to ensure vibrant results, demonstrating the ability to colorize a wide variety of scenes and objects. This fully automatic approach, driven by advances in DNNs and substantial training data, highlights the progress in achieving realistic and aesthetically pleasing colorized images. Various models, from simple convolutional networks to complex adversarial networks, illustrate the diverse strategies employed in this field.

Image classification has seen tremendous advancements since its early days, evolving through various technological breakthroughs and methodologies. In the early 20th century[13], image classification was primarily a manual process, relying on human experts to analyze and categorize images based on visual inspection. This process was time-consuming and prone to human error, limiting its scalability and accuracy.

The advent of digital computers in the mid-20th century marked a significant shift in image classification[14]. Early computer-based methods focused on simple feature extraction techniques, such as edge detection and texture analysis, to automate parts of the classification process. These methods laid the foundation for more sophisticated algorithms that would emerge later.

In the 1980s, the development of artificial neural networks (ANNs) brought new possibilities to image classification. Researchers like Yann LeCun pioneered the use of convolutional neural networks (CNNs), which mimicked the visual processing system of the human brain [15]. These early CNNs demonstrated the potential of deep learning for image recognition tasks, although their widespread adoption was limited by computational constraints.

The 1990s and early 2000s saw incremental improvements in both hardware and algorithms[16], allowing for deeper and more complex neural networks. The introduction of the backpropagation algorithm and advances in GPU technology enabled the training of larger networks[17], leading to better performance on image classification benchmarks.

The advent of deep learning has revolutionized the field of image classification. Early developments in convolutional neural networks (CNNs) demonstrated the potential of deep learning for handling complex visual recognition tasks. The introduction of deeper and more sophisticated neural network architectures has significantly improved the performance and accuracy of image classification systems.

In particular, the use of deep learning has enabled the automatic extraction of high-level features from raw image data, which has proven to be more effective than traditional handcrafted features. The availability of large-scale annotated datasets and advancements in computational power, especially through GPUs[17], have facilitated the training of very deep networks, leading to substantial improvements in image recognition benchmarks.

Innovations such as batch normalization[18], residual learning[19], and attention mechanisms [20] have further enhanced the training process and efficiency of these deep networks. These techniques help to stabilize the training of deep models, mitigate issues such as the vanishing gradient problem, and allow the networks to focus on the most relevant parts of the input images.

Deep learning models have also benefited from transfer learning, where pre-trained models on large datasets are fine-tuned for specific tasks, significantly reducing the need for extensive labeled data and computational resources. This approach has enabled the application of deep learning models to a wide range of image classification tasks with impressive results.

The success of deep learning in image classification has spurred a surge of interest and research in the field, leading to continuous advancements and the development of new techniques that push the boundaries of what is possible in visual recognition.

1.3 Related Work

In recent years, significant advancements have been made in the field of Artificial Intelligence (AI) and deep learning, particularly in the domains of image classification and segmentation. Convolutional Neural Networks (CNNs) have become the cornerstone for image classification tasks. Krizhevsky et al. [8] demonstrated the groundbreaking performance of CNNs with their AlexNet model, which won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012. Following this, various architectures like VGGNet [21], GoogLeNet [22], and ResNet [19] have been proposed, each introducing novel ways to enhance the depth and efficiency of neural networks.

Introduced by Goodfellow et al. [23], Generative Adversarial Networks (GANs) have shown remarkable success in generating high-quality images and data augmentation. GANs consist of two neural networks, a generator and a discriminator, that are trained adversarially. Radford et al. [24] further improved GANs with the introduction of Deep Convolutional GANs (DCGANs), which leverage convolutional layers to produce more realistic images.

Autoencoders, particularly the U-Net architecture proposed by Ronneberger et al. [25], have been highly effective for image segmentation and reconstruction tasks. U-Net's symmetric encoder-decoder structure with skip connections allows for precise localization, making it suitable for tasks requiring high-resolution output.

The classification of planktic foraminifera has seen significant progress with the application of machine learning techniques. Hsiang et al. [26] developed a deep learning-based system for automated identification of foraminifera species, achieving high accuracy and efficiency. Similarly, Nguyen et al. [27] employed a combination of CNNs and traditional image processing techniques to classify microscopic images of foraminifera.

Satellite image classification has been an area of active research, particularly with the availability of large datasets like EuroSAT [28]. Basu et al. [29] introduced DeepSat, which utilized deep learning for land cover classification using satellite images. More recently, Zhou et al. [30] proposed a deep learning approach for remote sensing image scene classification, emphasizing the importance of multi-spectral data.

Ensemble learning techniques combine multiple models to improve overall performance. The success of ensemble methods in machine learning was highlighted by Breiman [31] with the introduction of Random Forests. In the context of deep learning, ensembles of CNNs have been shown to outperform single models in various tasks, as demonstrated by the works of Szegedy et al. [22] and He et al. [19].

Transfer learning has become a popular technique for improving the performance of deep learning models on specific tasks with limited data. Yosinski et al. [32] discussed the transferability of features in deep neural networks, showing that features learned by deep models on large datasets can be effectively transferred to other tasks. This technique has been particularly useful in satellite image classification, as demonstrated by Marmanis et al. [33].

Beyond these well-known techniques, recent advancements have explored the use of hybrid models that integrate GANs with CNNs for enhanced performance in image classification tasks. For example, Ledig et al. [34] proposed SRGAN, a GAN-based approach that achieves high-quality image super-resolution, which can be applied to enhance the resolution of satellite images before classification. Moreover, Liang et al. [35] explored the use of GANs for enhancing the quality of training data, demonstrating significant improvements in model accuracy.

In the realm of planktic foraminifera classification, automated systems leveraging deep learning have begun to integrate multi-view approaches. For instance, Liu et al. [36] proposed a multi-view convolutional network that utilizes images captured from different angles to improve classification accuracy. This approach aligns closely with our method of using diverse image processing techniques to create multiple RGB representations.

Furthermore, the adoption of advanced optimization algorithms has shown promise in refining deep learning models. Techniques such as Adam [37] and RMSprop [38] optimizers have been instrumental in achieving faster convergence and better performance in training deep neural networks.

These advancements collectively underscore the potential of integrating multiple deep learning techniques to enhance the accuracy and efficiency of image classification systems. Our proposed system builds upon these foundations, aiming to push the boundaries of what is achievable in the classification of both planktic foraminifera and satellite images.

2
DATASETS

2.1 Foraminifera dataset

The Foraminifera dataset represents a valuable resource in marine biology and environmental science, comprising grayscale images that capture the structural intricacies of foraminifera, microscopic marine organisms essential for paleoceanography and climate research. Foraminifera are particularly significant due to their role as sensitive indicators of environmental changes, making their detailed morphological study crucial for understanding past climates and predicting future environmental trends.

Each image in the dataset is meticulously labeled according to species, facilitating precise classification tasks and providing a foundation for biodiversity assessments in marine ecosystems. The imaging process utilizes a reflected light binocular microscope, specifically the AmScope SE305R-PZ, which operates at a magnification of 30×. This setup ensures high-resolution imaging of specimens, capturing fine details and variations in morphology across multiple angles of illumination.

To comprehensively depict each foraminifera specimen, images are captured at intervals of 22.5°, resulting in sixteen grayscale images per sample. This approach enables the documentation of morphological features from diverse perspectives, enhancing the dataset's utility in detailed morphometric analyses and species differentiation studies.

The images typically exhibit a resolution averaging approximately 520 × 480 pixels, although slight variations in resolution may be present due to differences in imaging conditions and specimen sizes. This variability in resolution enriches the dataset, reflecting the natural diversity observed in foraminifera specimens.

In total, the Foraminifera dataset comprises 1,437 samples categorized into seven distinct classes based on species taxonomy and morphological characteristics. This classification scheme facilitates targeted investigations into species-specific adaptations, ecological interactions, and responses to environmental stressors, supporting broader studies on marine biodiversity and ecosystem health.

By providing a comprehensive collection of labeled images with detailed metadata, the Foraminifera dataset not only serves as a cornerstone for species identification and taxonomic research but also contributes to advancing our understanding of marine ecosystems' resilience and adaptation in the face of global environmental changes. Its accessibility promotes collaborative research efforts and educational initiatives aimed at fostering marine conservation and sustainable management practices worldwide. The dataset comprises a total of 1,437 samples, which are categorized into the seven classes as it is provided in the following figure.

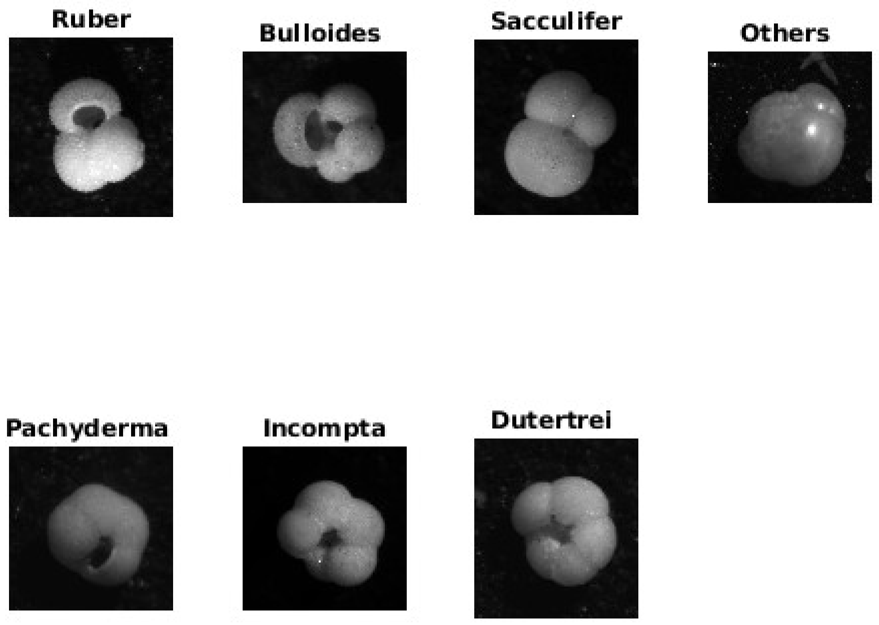

Figure 2.1: samples of grayscale foraminifera dataset

The following is the number of images for each class in the Foraminifera dataset:

1. 178 images of *G. bulloides*
2. 182 images of *G. ruber*
3. 150 images of *G. sacculifer*
4. 174 images of *N. incompta*
5. 152 images of *N. pachyderma*
6. 151 images of *N. dutertrei*
7. 450 images of "rest of the world," belonging to other species of planktic foraminifera

The diversity and variability in the morphological characteristics of foraminifera make this dataset particularly challenging for classification tasks. The subtle differences between species often require high precision in feature extraction and classification algorithms. Moreover, the use of different illumination angles adds another layer of complexity, as the same specimen may appear different under varying lighting conditions. A significant problem with this dataset is the imbalance in the number of images per category, particularly the large number of images in the "rest of the world" category compared to the other species. This imbalance, known as class imbalance, can lead to biased model predictions, where the classifier may be more inclined to predict the overrepresented class.

2.2 EUROSAT DATASET

The EuroSAT dataset is a multi-spectral dataset that captures a wide range of information across different wavelengths, revealing diverse features of the Earth's surface. This dataset is based on Sentinel-2 satellite images and is part of the Copernicus Earth observation program, which aims to provide accurate and timely information about the planet.

The EuroSAT dataset utilizes data from 13 spectral bands captured by the Sentinel-2 satellite, which operates within the Copernicus Earth observation program. These bands cover a range of wavelengths including visible light (RGB bands), near-infrared (NIR), and short-wave infrared (SWIR). Each of these spectral bands captures unique information about the Earth's surface, which collectively enable a detailed analysis of various land cover and land use characteristics.

- Visible Bands (Red, Green, Blue):These bands capture the visible spectrum of light that humans can perceive. They provide information about surface properties such as vegetation density, water bodies, and urban areas based on their reflectance characteristics in these wavelengths.

- Near-Infrared (NIR) Bands:NIR bands are sensitive to vegetation health and structure. Healthy vegetation reflects NIR light strongly, while stressed or sparse vegetation absorbs more NIR light. This band is crucial for distinguishing between different types of vegetation and assessing their health.

- Short-Wave Infrared (SWIR) Bands: SWIR bands penetrate through thin clouds and haze, allowing observation of surface features regardless of atmospheric conditions. These bands are particularly useful for identifying geological features, soil moisture content, and distinguishing between different materials based on their unique SWIR reflectance properties.

By combining information from these 13 spectral bands, the EuroSAT dataset provides a comprehensive view of Earth's surface. Researchers and practitioners can leverage this dataset for tasks such as:

- Land Cover Classification:Using machine learning algorithms to classify different types of land cover (e.g., forests, urban areas, water bodies) based on the distinctive spectral signatures captured across multiple bands.
- Land Use Analysis:

  Studying how land is utilized based on its spectral characteristics, such as agricultural practices, urban expansion, and natural resource management.
- Land Use Analysis: Studying how land is utilized based on its spectral characteristics, such as agricultural practices, urban expansion, and natural resource management.

The classes in the EuroSAT dataset are chosen based on their visibility at a resolution of 10 meters per pixel and their frequent coverage by the European Urban Atlas. The dataset consists of 27,000 labeled images divided into the following classes:

1. 3,000 images of annual crop
2. 3,000 images of forest
3. 3,000 images of herbaceous vegetation
4. 2,500 images of highway
5. 2,500 images of industrial
6. 2,000 images of pasture
7. 2,500 images of permanent crop
8. 3,000 images of residential
9. 2,500 images of river
10. 3,000 images of sea and lake

The multi-spectral nature of the EuroSAT dataset presents unique challenges for classification. Integrating information from 13 spectral bands requires sophisticated preprocessing steps to extract meaningful features for both grayscale and colored image representations. One primary challenge is normalizing the spectral data to ensure consistency and comparability across different bands.the figure below illustrate samples of grayscale images of Eurosat dataset. Grayscale image extraction is a common approach based on select specific spectral bands that are most informative for the target classification task.These selected bands are then normalized to a common scale to create a grayscale representation. Creating RGB images from the 13 spectral bands involves selecting and combining three specific bands that correspond to the red, green, and blue channels. This selection process must consider the unique information each band provides and how it contributes to the overall image representation. Normalization is crucial here to ensure that the combined RGB image has balanced color intensity and accurately represents the spectral information.

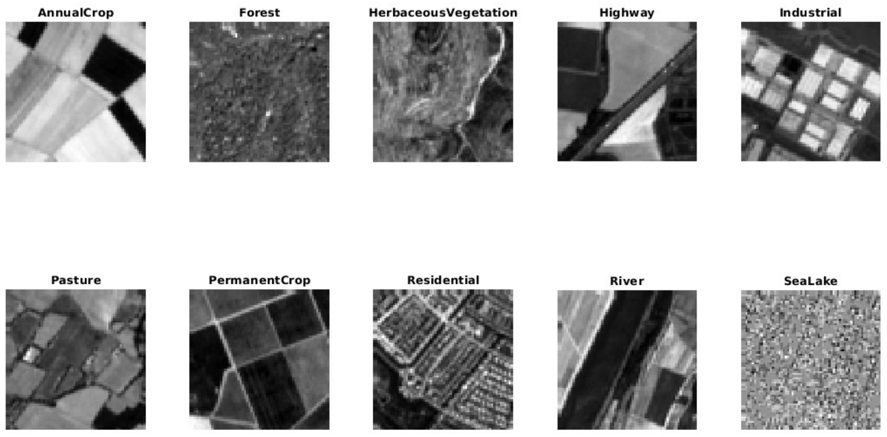

Figure 2.2: samples of grayscale EuroSat dataset

3

PRPOSED ALGORITHMS

As we have previously discussed, the primary objective of this research is to develop a method for image colorization, transforming gray-scale images into their colored counterparts. This colorized output will subsequently be utilized for classification tasks, aiming to categorize the images into their respective categories. The process of image colorization is crucial as it enriches the gray-scale images with additional information, potentially enhancing the performance of downstream tasks such as classification.Our proposed supervised learning approach for image colorization not only aims to achieve realistic and accurate colorization but also enhances the subsequent classification tasks by providing enriched input data. This dual benefit underscores the significance of effective image colorization in the broader scope of image analysis and classification in deep learning.

3.1 Supervised Learning for Image Colorization

In the context of supervised learning, the model is trained to map an input to a known output. For image colorization, this involves transforming a grayscale image into its colorized version. Colored images provide the "ground truth" or target outputs against which the generator's outputs are compared. This direct comparison helps in fine-tuning the generator to produce colorizations that are not only realistic but also accurate with respect to natural colors as seen in the real world.

A significant challenge arises when the available datasets lack the necessary color information. For instance, the Foraminifera dataset consists of single-channel grayscale images, capturing only the structural and morphological details of the marine organisms without any color information. This limitation poses a challenge for applying supervised learning techniques such as Generative Adversarial Networks (GANs) and U-Net, which require both grayscale inputs and their corresponding colored outputs to train the model effectively.

Similarly, the EuroSAT dataset, based on Sentinel-2 satellite images, comprises multi-spectral images stored in 13 bands in TIFF format. These bands capture a wide range of information across different wavelengths but do not directly provide RGB images. The multi-spectral bands offer comprehensive data on various surface conditions and materials but translating this information into a format suitable for RGB-based colorization presents an additional challenge. The necessity to convert these multi-spectral bands into RGB images that accurately represent the scene's true colors is critical for subsequent classification tasks.

To address these challenges, we must either source additional color information or employ techniques that can infer color from context and structure. The ultimate aim is to generate colorized images that are sufficiently accurate to be used in classification tasks. According to deep learning principles and supported by numerous studies indexed in Google Scholar, enhancing grayscale images through colorization can provide additional features that improve the accuracy of classification algorithms. By leveraging the enriched information from colorized images, the classification models can better capture and differentiate the nuanced features of different categories, leading to improved performance.

3.1.1 Colorization Using Supervised Generative Adversarial Networks

Generative Adversarial Networks (GANs)are a sophisticated neural network architecture designed for generating synthetic data. GANs were introduced by Ian Goodfellow and his colleagues in 2014[23], revolutionizing the field of generative models. The basic idea of GANs involves two main elements: the generator and the discriminator, engaging in an adversarial process.

The need to obtain colored images for both datasets is crucial for the successful application of supervised learning in colorization tasks. In the case of the Foraminifera dataset, to produce colored target images for Foraminifera, the process begins by normalizing the gray-scale images to ensure consistent processing. Each image I is normalized using the equation $I_{\text{normalized}} = \frac{I - \min(I)}{\max(I) - \min(I)}$. Following normalization, each image undergoes a Discrete Wavelet Transform (DWT)[39] to decompose it into frequency components, capturing essential details at multiple scales. This decomposition is represented as

$$\text{DWT}(I) = \{(c_A, c_H, c_V, c_D)\} = \text{dwt2}(I, \psi)$$

where c_A, c_H, c_V, and c_D are the approximation, horizontal, vertical, and diagonal coefficients, respectively.

The images are then grouped into batches and reordered based on the mean of their 2D Discrete Cosine Transform (DCT), given by

$$\text{mean\_dct}(I) = \text{mean}(\text{dct2}(I))$$

After reordering, the images are organized into three sets, each representing one of the RGB channels. For each group of images $G = \{I_1, I_2, \ldots, I_5\}$, DWT is applied to each image in the group, resulting in

$$\mathrm{DWT}(G) = \{\mathrm{DWT}(I_1), \mathrm{DWT}(I_2), \ldots, \mathrm{DWT}(I_5)\}$$

The DWT coefficients are then merged to form a single channel for each group. This merging is performed by averaging the pixel values across the group, expressed as

$$R(i,j) = \frac{1}{5}\sum_{k=1}^{5} G_R(k)(i,j)$$

$$G(i,j) = \frac{1}{5}\sum_{k=1}^{5} G_G(k)(i,j)$$

$$B(i,j) = \frac{1}{5}\sum_{k=1}^{5} G_B(k)(i,j)$$

where G_R, G_G, and G_B are the groups for the red, green, and blue channels, respectively.

Each merged channel then undergoes inverse DWT to reconstruct the spatial domain images, represented by

$$R_{\text{reconstructed}} = \text{idwt2}(R, \psi)$$

$$G_{\text{reconstructed}} = \text{idwt2}(G, \psi)$$

$$B_{\text{reconstructed}} = \text{idwt2}(B, \psi)$$

The reconstructed R, G, and B channels are combined to form a composite RGB image,

$$\text{RGB}_{\text{image}} = [R_{\text{reconstructed}}, G_{\text{reconstructed}}, B_{\text{reconstructed}}]$$

Finally, an additional image from each batch is used to enhance the final RGB image by averaging its details with the RGB image, as expressed by

$$\text{Final\_Image}(i,j) = \frac{\text{RGB}_{\text{image}}(i,j) + \text{Additional\_Image}(i,j)}{2}$$

This process results in colored images necessary for supervised learning in colorization tasks for Foraminifera dataset, while in the case of colorization EuroSAT dataset, normalization is crucial to manage the 13 spectral bands effectively. Each band is normalized to the range [0, 255], allowing them to be treated as grayscale images as it is shown in figure 2. This normalization is accomplished by scaling the pixel values of each band.

$$I_{\text{normalized}} = \left(\frac{I_{\text{original}} - \min(I_{\text{original}})}{\max(I_{\text{original}}) - \min(I_{\text{original}})} \right) \times 255$$

Once normalized, each band can be treated as a grayscale image. To obtain an RGB image, three normalized bands corresponding to the red, green, and blue wavelengths are selected and combined as it is shown in figure 2:

$$I_{\text{RGB}} = (I_{\text{normalized, band1}}, I_{\text{normalized, band2}}, I_{\text{normalized, band3}})$$

By treating each band as a separate grayscale image or combining specific bands for RGB images, the EuroSAT dataset's spectral richness can be fully exploited for various image processing and analysis tasks. The following figure illustrates a sample of colored images that will be used as target image for both dataset.

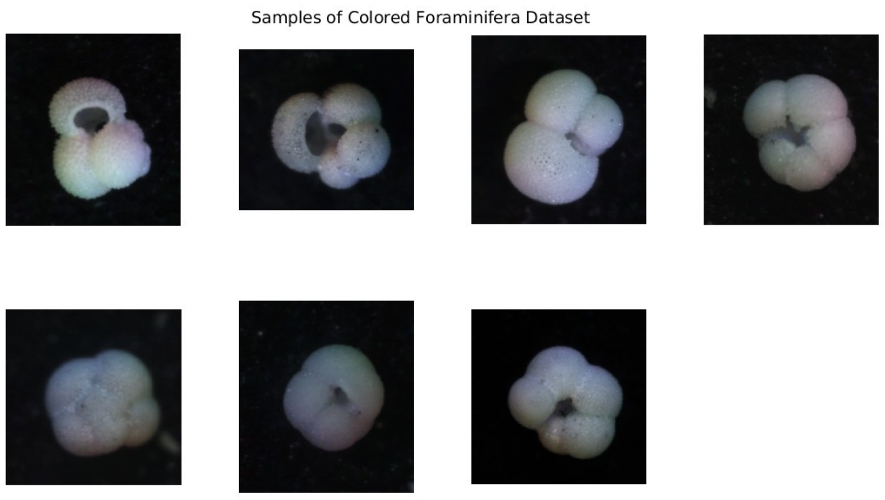

Figure 3.1: Sample of Target Images of Foraminfera dataset using DWT techinque

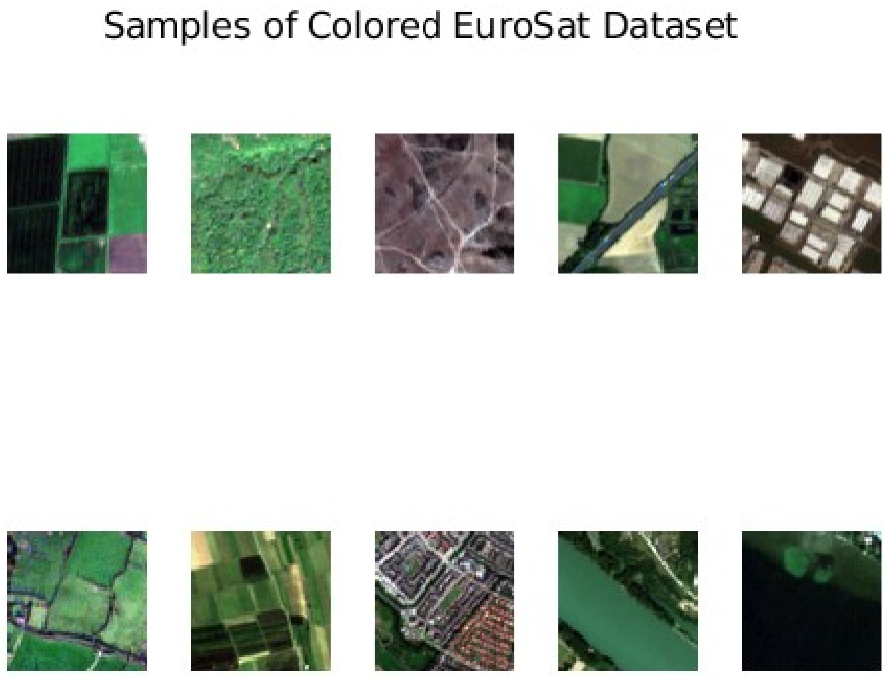

Figure 3.2: Sample of Target Images of EuroSat dataset using Normalization techinque

In generative adversarial network, the generator's goal is to produce data that are indistinguishable from genuine data, while the discriminator aims to accurately distinguish between real and generated data. This process is akin to a min-max game, where the generator tries to minimize its loss by producing realistic data, and the discriminator tries to maximize its accuracy in distinguishing real from fake data. Mathematically, this can be formulated as:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \tag{3.1}$$

As mentioned before, in a supervised GAN framework the model uses paired datasets—specifically, grayscale images alongside their colorized counterparts—providing clear target outcomes for the generator, crucial for tasks requiring high fidelity like image colorization. The generator receives a grayscale image and predicts its color channels, transforming a low-dimensional input into a high-dimensional output. It aims to produce an output that closely matches the target color image provided during training. The discriminator assesses the authenticity of the generated images against real, colored images from a training set. It evaluates both the realism of the generated images and how closely these images match the target color images.

GANs typically employ convolutional neural networks (CNNs) for both the generator and the discriminator. CNNs are particularly well-suited for image processing tasks due to their ability to capture spatial hierarchies in images through the use of convolutional layers. Convolutional layers apply a series of filters to the input image to detect features such as edges, textures, and patterns. These features are then used to construct a detailed representation of the image.

The generator in a GAN is designed as a deep convolutional network that takes a low-dimensional input (such as a noise vector or a grayscale image) and upsamples it through a series of convolutional layers to produce a high-dimensional output (such as a color image). The discriminator, on the other hand, is a convolutional neural network that takes an image as input and classifies it as real or fake by downsampling the image through a series of convolutional layers, ultimately outputting a probability score indicating the likelihood that the image is real.

The versatility of GANs makes them applicable to various tasks beyond image colorization. These include:

- **Image Inpainting:** Filling in missing parts of an image.
- **Style Transfer:** Applying the style of one image to the content of another.
- **Image Super-Resolution:** Enhancing the resolution of an image.
- **Semantic Segmentation:** Converting an image to a segmentation map and vice versa.
- **Object Transfiguration:** Changing objects in an image to other types, such as turning horses into zebras.

The ability of GANs to learn the mapping from input to output images, along with a loss function to train this mapping, makes them a powerful tool for a wide range of applications in computer vision and beyond. The combination of convolutional neural networks and adversarial training in the GAN framework facilitates the generation of highly realistic and contextually accurate images, significantly advancing the capabilities of machine learning in visual data processing.

Training involves minimizing adversarial and content losses:

- **Generator's Loss:** The generator's loss combines adversarial loss, aimed at fooling the discriminator, and content loss[40], which minimizes the difference between the generated and the target images.

$$\mathcal{L}_{\text{GAN}}(G) = -\mathbb{E}[\log(1 - D(G(I_{\text{gray}})))] \tag{3.2}$$

$$\mathcal{L}_{\text{content}} = \mathbb{E}\left[|G(I_{\text{gray}}) - I_{\text{real}}|^2\right] \tag{3.3}$$

- **Discriminator's Loss:** The discriminator's loss is designed to effectively distinguish real images from the synthetically generated ones.

$$\mathcal{L}_{\text{GAN}}(D) = -\mathbb{E}[\log D(I_{\text{real}}) + \log(1 - D(G(I_{\text{gray}})))] \tag{3.4}$$

Explanation of Parameters

- G: The generator network function, which outputs the colorized image from the input grayscale image I_{gray}.
- D: The discriminator network function, which evaluates whether images are real or generated by the generator.
- I_{gray}: The input grayscale image to the generator.
- I_{real}: The real colored image corresponding to the grayscale input, serving as the target for the generator's output.
- $\mathbb{E}$: The expectation operator, used here to denote the expected value over all samples in the dataset, integral to computing the average loss across batches of images.

The feedback from the discriminator's evaluations is used to adjust the generator's parameters, refining the output through successive training iterations.The following diagram illustrates the core operation of a Generative Adversarial Network (GAN), where the Generator starts with a random input and attempts to create images that mimic real data. These generated images, along with actual images from a dataset, are fed into the Discriminator, which evaluates each to determine whether they are real or synthetic. The Generator aims to fool the Discriminator into misclassifying the fake images as real, while the Discriminator learns to better distinguish between the two. This adversarial process dynamically improves the capabilities of both networks, enhancing the Generator's ability to produce realistic images and the Discriminator's accuracy in classification.

- **Generator Architecture (U-Net):**

The U-Net generator[41], designed for image colorization tasks, features a symmetric architecture with distinct encoder and decoder paths connected by a bridge, facilitating precise colorization of grayscale images. The encoder consists of a series of convolutional blocks, each composed of a 3x3 convolutional layer with stride 1, ensuring spatial dimensions are preserved while doubling the number of feature channels at each step. These convolutional layers are followed by ReLU activation functions which introduce non-linearity, essential for learning complex features in images. After each convolutional operation, max pooling with a 2x2 filter and stride 2 is applied, successively reducing the spatial dimensions by half and thereby increasing the receptive field, allowing the network to capture broader image contexts at deeper layers.

At the lowest resolution, the bridge consists of a further 3x3 convolutional layer and ReLU activation, processing the most abstract features which carry the highest-level image representations. The decoder mirrors the encoder's structure but in reverse, utilizing transposed convolutional layers with a stride of 2 to progressively upsample feature maps to higher spatial resolutions. Each upsampling step is followed by a 3x3 convolutional layer and ReLU activation, refining the upsampled features to recover spatial details. Crucially, the decoder employs skip connections from corresponding encoder layers, which concatenate feature maps from the encoder to the decoder, aiding in the restoration of fine details lost during downsampling. This ensures that local and global information is effectively combined to produce a detailed and accurately colorized output image. The final output is generated through a 3x3 convolutional layer that adjusts the channel dimension to three, corresponding to the RGB channels of the output color image, followed by a tanh activation function which normalizes the pixel values to the range [-1,1], suitable for image processing tasks.

The architecture of the U-Net generator used for colorizing grayscale images is illustrated in the following figure, showcasing the detailed configuration of each encoder and decoder layer, along with their respective roles in the image colorization process.

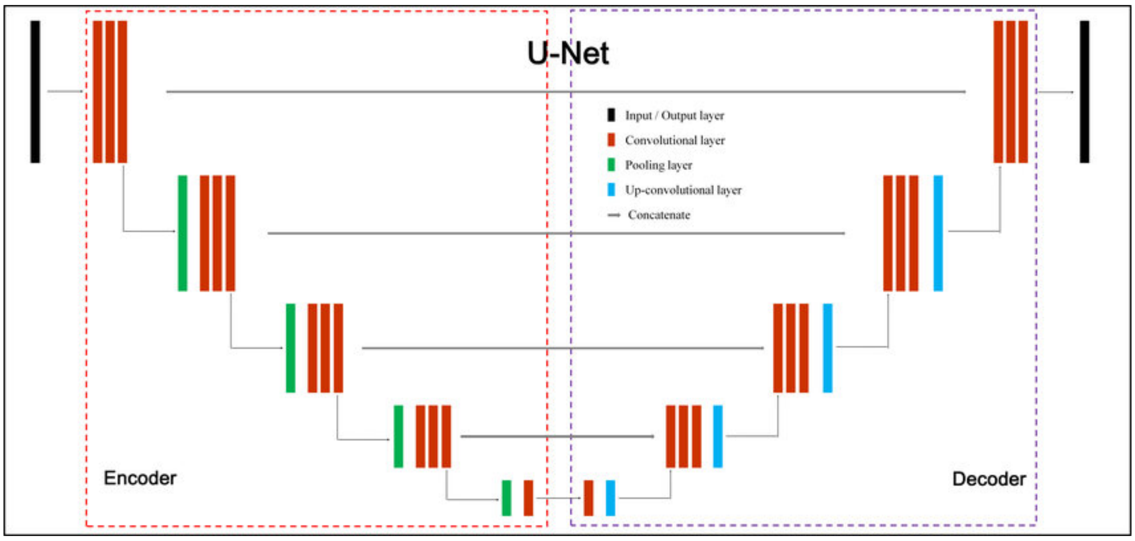

Figure 3.3: The architecture of U-Net

- **Discriminator Architecture (PatchGAN):**

The discriminator employs a PatchGAN architecture[41], designed to classify patches of the image as real or fake, which allows it to focus on fine-grained, local image statistics and is computationally efficient. The input layer accepts patches of size $[256 \times 256 \times 3]$ from the Foraminifera dataset and $[128 \times 128 \times 3]$ from the EuroSAT dataset, from both real and generated images. As shown in the following figure, the discriminator comprises multiple downsampling blocks. The first block consists of a 4x4 convolutional layer with 64 filters, using symmetric padding to include edge information. This block is followed by batch normalization and a leaky ReLU activation. The second block doubles the number of filters to 128 and includes a 4x4 convolutional layer, batch normalization, and a leaky ReLU activation. The third block further doubles the number of filters to 256, with similar components: a 4x4 convolutional layer, batch normalization, and a leaky ReLU activation. The fourth block doubles the filters again to 512, also containing a 4x4 convolutional layer, batch normalization, and a leaky ReLU activation. For the Foraminifera dataset, an additional block with 512 filters is used, bringing the total number of blocks to five. This block includes a 4x4 convolutional layer, batch normalization, and a leaky ReLU activation. The final block is followed by a 4x4 convolutional layer that reduces the number of filters back to 1, using a sigmoid activation function to output a probability between 0 and 1 for each patch, classifying it as real or fake. The number of filters in these layers starts at 64 and doubles with each block to capture a broader range of features at different scales. Batch normalization is applied to stabilize training by normalizing the outputs, while a leaky ReLU activation allows for a small, non-zero gradient when the unit is less active, preventing the dying ReLU problem and maintaining gradient flow. The final output is produced through a sigmoid activation layer, which classifies each image patch as real or fake by outputting a probability between 0 and 1, providing a measure of the generator's success at fooling the discriminator.

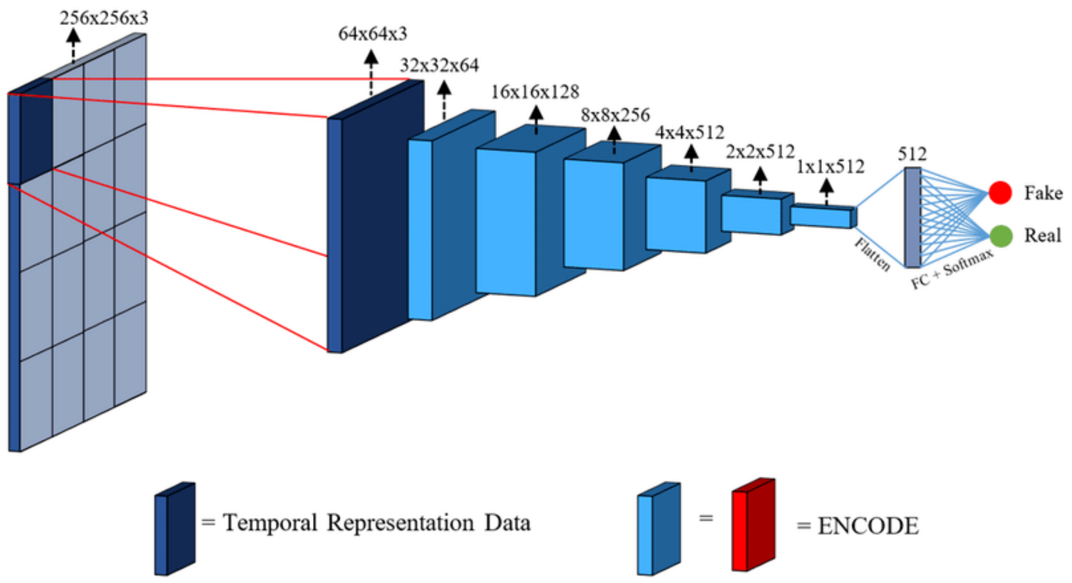

Figure 3.4: Discriminator Framework

- **TRAINING STRATEGIES:**

In the training of the Generative Adversarial Network (GAN), several strategies are implemented to optimize the performance and ensure stability. The Adam optimizer is utilized for both the generator and the discriminator with a learning rate set at 2×10^{-5} and beta coefficients $\beta_1 = 0.6$ and $\beta_2 = 0.8$, $\beta_1 = 0.5$ and $\beta_2 = 0.999$facilitating effective adaptation of the learning process.

The learning rate (α) in the Adam optimizer controls the step size at each iteration while moving toward a minimum of the loss function. A smaller learning rate like 2×10^{-5} ensures that the model learns slowly and steadily, reducing the risk of overshooting the minimum and helping to converge to an optimal solution.

The beta coefficients (β_1 and β_2) are parameters of the Adam optimizer that control the exponential decay rates of the moving averages of the gradient and its square, respectively. Specifically, β_1 (set to 0.5 and 0.6) is the decay rate for the first moment estimate, which essentially averages the gradient values to smooth out the updates. β_2 (set to 0.8 and 0.999) is the decay rate for the second moment estimate, which averages the squared gradient values to keep track of the variability. These coefficients help in reducing the variance of the parameter updates, leading to more stable and efficient training.

A key component in training GANs is the patch size used for the discriminator. Patch size refers to the size of the image patches that the discriminator processes to determine whether they are real or generated. Smaller patch sizes can help the discriminator focus on local features, while larger patch sizes can provide more global context. In this study, several patch sizes were used: 70x70 for both datasets, 256x256 for Foraminifera (matching the input size), 128x128 for EuroSAT (matching the input size), and 50x50 for both datasets. These varying patch sizes allow the discriminator to learn both fine-grained details and broader contextual features.

The model is trained over 10, 25 and 30 epochs, with a batch size of 64 for Foraminifera dataset and 32 for EuroSat. This setup strikes a balance between efficient computation and memory usage, allowing for detailed gradient updates over an adequate number of iterations. Both EuroSAT and Foraminifera datasets are divided into three parts: 80% is used for training, and 10% each is allocated for validation and testing. This distribution ensures extensive training on a significant volume of data while reserving sufficient examples for validation and unbiased testing.

During each epoch, images are processed in mini-batches, formatted into arrays, and, when applicable, transferred to a GPU to enhance computational speed. Losses for both the generator and discriminator are calculated using tailored loss functions that assess the discriminator's ability to differentiate real from generated images and the generator's success in deceiving the discriminator. These losses guide the updates to the parameters of both networks using the Adam optimization technique, which adjusts learning rates based on the statistical properties of the gradients.

Training progress is continuously monitored, with losses averaged after each epoch to evaluate the model's performance and detect potential overfitting. Following the training phase, the GAN undergoes rigorous testing against a separate set of data to assess its generalization capabilities across new, unseen data. The tables below present the discriminator and generator losses obtained from tests conducted on unseen data, showcasing the final performance of the model across various datasets with differing epochs and patch sizes.

Table 3.1: Losses for Discriminator and Generator using 70*70 patch sizewith $\beta_1 = 0.6$ and $\beta_2 = 0.85$

Dataset	Epochs	Patch Size	Discriminator Loss	Generator Loss	β_1	β_2
EuroSat	10	70x70	0.239	0.922	0.6	0.85
EuroSat	25	70x70	0.089	2.163	0.6	0.85
EuroSat	30	70x70	0.117	1.981	0.6	0.85
Foraminifera	10	70x70	0.358	0.886	0.6	0.85
Foraminifera	25	70x70	0.112	1.832	0.6	0.85
Foraminifera	30	70x70	0.222	1.689	0.6	0.85

Table 3.2: Losses for Discriminator and Generator using 100*100 patch sizewith $\beta_1 = 0.6$ and $\beta_2 = 0.85$

Dataset	Epochs	Patch Size	Discriminator Loss	Generator Loss	β_1	β_2
EuroSat	10	100x100	0.416	0.853	0.6	0.85
EuroSat	25	100x100	0.113	1.003	0.6	0.85
EuroSat	30	100x100	0.455	1.122	0.6	0.85
Foraminifera	10	100x100	0.566	0.886	0.6	0.85
Foraminifera	25	100x100	0.098	1.912	0.6	0.85
Foraminifera	30	100x100	0.369	1.772	0.6	0.85

Table 3.3: Losses for Discriminator and Generator losses matching the input size for both dataset with $\beta_1 = 0.5$ and $\beta_2 = 0.999$

Dataset	Epochs	Patch Size	Discriminator Loss	Generator Loss	β_1	β_2
EuroSat	10	128x128	0.369	0.935	0.6	0.85
EuroSat	25	128x128	0.272	1.871	0.6	0.85
EuroSat	30	128x128	0.358	2.052	0.6	0.85
Foraminifera	10	256x256	0.328	0.933	0.6	0.85
Foraminifera	25	256x256	0.0875	2.008	0.6	0.85
Foraminifera	30	256x256	0.0964	1.828	0.6	0.85

Table 3.4: Losses for Discriminator and Generator using 70x70 Patch Size with $\beta_1 = 0.5$ and $\beta_2 = 0.999$

Dataset	Epochs	Patch Size	Discriminator Loss	Generator Loss	β_1	β_2
EuroSat	10	70x70	0.466	0.771	0.5	0.999
EuroSat	25	70x70	0.302	1.023	0.5	0.999
EuroSat	30	70x70	0.383	0.892	0.5	0.999
Foraminifera	10	70x70	0.364	0.779	0.5	0.999
Foraminifera	25	70x70	0.322	0.871	0.5	0.999
Foraminifera	30	70x70	0.355	0.764	0.5	0.999

Table 3.5: Losses for Discriminator and Generator using 100x100 Patch Size with $\beta_1 = 0.5$ and $\beta_2 = 0.999$

Dataset	Epochs	Patch Size	Discriminator Loss	Generator Loss	β_1	β_2
EuroSat	10	100x100	0.523	0.652	0.5	0.999
EuroSat	25	100x100	0.447	0.886	0.5	0.999
EuroSat	30	100x100	0.422	0.611	0.5	0.999
Foraminifera	10	100x100	0.501	0.680	0.5	0.999
Foraminifera	25	100x100	0.432	0.772	0.5	0.999
Foraminifera	30	100x100	0.558	0.718	0.5	0.999

Table 3.6: Losses for Discriminator and Generator using 70x70 Patch Size with $\beta_1 = 0.5$ and $\beta_2 = 0.999$

Dataset	Epochs	Patch Size	Discriminator Loss	Generator Loss	β_1	β_2
EuroSat	10	128x128	0.479	0.733	0.5	0.999
EuroSat	25	128x128	0.411	0.955	0.5	0.999
EuroSat	30	128x128	0.502	0.877	0.5	0.999
Foraminifera	10	256x256	0.283	0.893	0.5	0.999
Foraminifera	25	256x256	0.258	0.933	0.5	0.999
Foraminifera	30	256x256	0.277	0.904	0.5	0.999

The following figures illustrate sample colorized images generated by the Generative Adversarial Network, showcasing the network's ability to produce visually compelling results from grayscale inputs. After experiments, we found that β_1 and $\beta_2 = 0.6$ and 0.85 perform better than 0.5 and 0.999. The lowest loss function for both datasets was achieved at 25 epochs. Moreover, for patch size, 70x70 is optimal for the EuroSAT dataset, while the entire input size for the Foraminifera dataset works best. The lowest discriminator loss for the Foraminifera dataset was 0.0875 with a generator loss of 2.008 using the entire input size, and the lowest discriminator loss for the EuroSAT dataset was 0.089 with a generator loss of 2.163 using the 70x70 patch size.

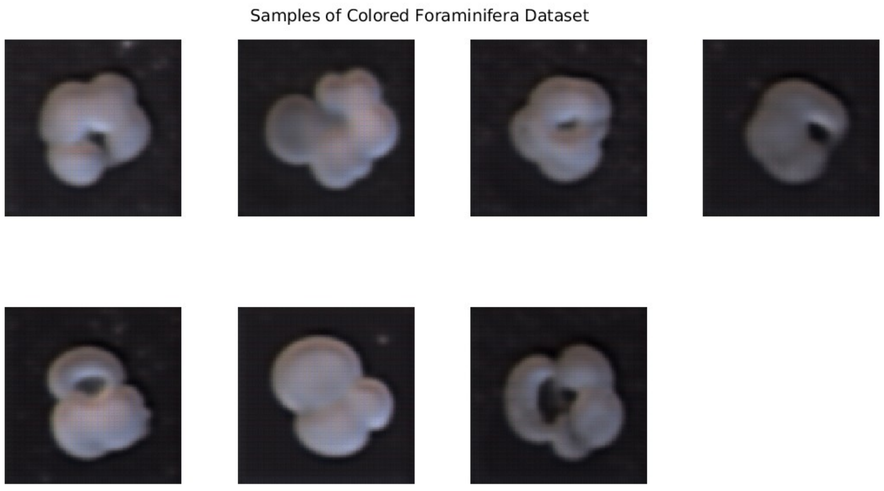

Figure 3.5: Samples of colored Foraminifera images using GAN

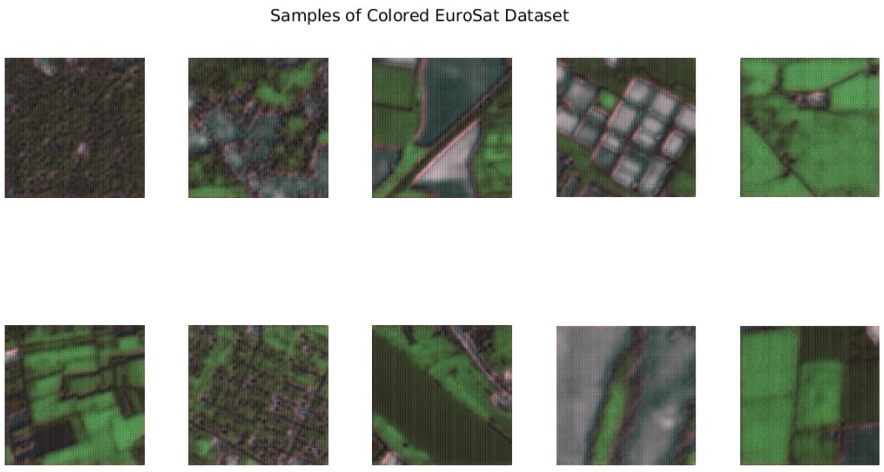

Figure 3.6: Samples of colored EuroSat images using GAN

3.1.2 Auto-encoder with Multi-skip Connections for Image Colorization

Auto-encoder colorization[7] is a supervised learning process that leverages paired datasets of grayscale and corresponding colored images. In this approach illustrated in the below figure, the model learns to map grayscale images to their colored counterparts by minimizing the difference between the predicted and actual color values. Supervised learning in image colorization is critical because it provides the model with explicit examples of how grayscale information translates into color, enabling the auto-encoder to develop a robust understanding of colorization patterns and nuances.For both Foraminifera and EuroSAT datasets, I employed the same pervious methods to colorize gray scale images in order to obtain colored images that will be used as target images.

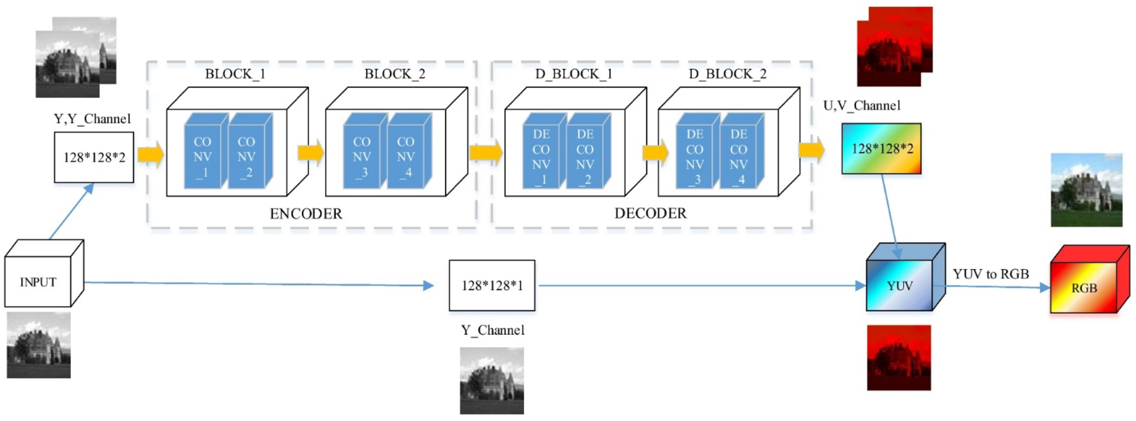

Figure 3.7: Auto-Encoder with Skip Connections Framework

After obtaining colored images for both the EuroSAT and Foraminifera datasets to be used as target images, we apply an advanced image colorization method utilizing a deep convolutional auto-encoder that exploits the YUV color space, in contrast to the traditional RGB space.

The methodology begins by transforming RGB images into YUV, then effectively separating luminance (Y) from chrominance (U and V) as it is shown in the following figure. This decomposition simplifies the learning process as the model directly utilizes the Y channel, a gray-scale representation, and focuses on accurately predicting the U and V channels, which embody the color information.

The YUV color space is designed to separate image luminance from color information, which can be beneficial for various image processing tasks. The Y channel represents the luminance, or brightness, of the image, essentially

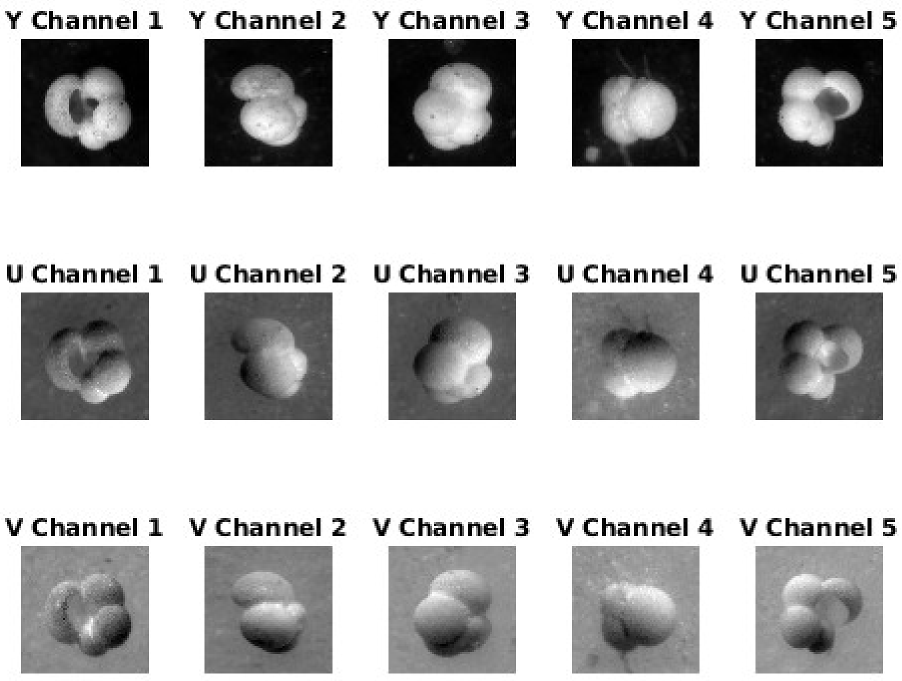

Figure 3.8: Foraminifera Example of Separating Y,U And V Channels

capturing all the gray-scale information. This channel is crucial because the human eye is more sensitive to changes in brightness than to changes in color.

The U and V channels, on the other hand, represent chrominance information. The U channel (also known as Cb) captures the blue-difference chroma component, while the V channel (also known as Cr) captures the red-difference chroma component. These channels contain the color information by describing how much blue or red should be added or subtracted from the luminance to achieve the desired color. The steps steps in our methodology are as follows:

1. **Transform RGB to YUV**:

   Each RGB image is converted into the YUV color space. This step decomposes the image into one luminance channel (Y) and two chrominance channels (U and V).

2. **Utilize Y Channel**: The Y channel, which contains the grayscale representation of the image, is directly used by the model. This simplifies the learning process because the model only needs to predict the chrominance components to generate a color image.

3. **Predict U and V Channels**: The model focuses on accurately predicting the U and V channels. These channels are responsible for the color information in the image. By learning the patterns and correlations between the Y channel and the corresponding U and V channels, the model can effectively generate realistic colorizations.

4. **Reconstruct RGB Image**: After predicting the U and V channels, the Y, U, and V channels are combined and transformed back into the RGB color space to produce the final colorized image.

This approach[42] leverages the separation of luminance and chrominance to simplify the learning task for the auto-encoder, enabling it to produce high-quality colorization by focusing on the color information in the U and V channels while relying on the Y channel for structural details.

The encoder is a critical component of the auto-encoder model. It consists of multiple convolutional neural network (CNN) layers designed to extract high-level features from the input gray-scale image. Each convolutional layer applies filters to the input image, capturing spatial hierarchies of patterns[8]. The encoder works by progressively reducing the spatial dimensions of the input image (down-sampling) while increasing the depth of the feature maps. This process, often involving convolutional and max-pooling layers [43] , compresses the information into a lower-dimensional representation, making it easier for the network to learn the essential features necessary for colorization [44]

The decoder in the auto-encoder model is responsible for reconstructing the color information (U and V channels) from the encoded feature representation. It consists of up-sampling layers, typically implemented using transposed convolutional layers, which increase the spatial dimensions of the feature maps. The decoder works by gradually restoring the spatial dimensions of the image (up-sampling) while reducing the depth of the feature maps, reversing the compression performed by the encoder. This up-sampling process restores the image to its original resolution, synthesizing fine details and enabling the generation of detailed and accurate color images. The decoder combines these upsampled features with the corresponding high-resolution features from the encoder (via skip connections in U-Net), enhancing the accuracy and detail of the colorization.

The U-Net model[7], initially developed for biomedical image segmentation, has proven to be highly versatile and effective for various image processing tasks, including image colorization.

In the context of image colorization, the U-Net architecture's unique U-shape plays a crucial role in its performance. The symmetric structure, comprising an encoder (contracting path) and a decoder (expanding path) with skip connections, helps in effectively capturing and preserving both global and local features of the image.

The U-Net model is named after its distinctive U-shaped architecture, which is designed to perform well on image-to-image tasks. This architecture includes an encoder (contracting path) that reduces the spatial dimensions of the input image through convolutional and max-pooling layers, capturing the context and essential features at multiple scales. The decoder (expanding path) increases the spatial dimensions back to the original size, using transposed convolutions (or up-sampling) and convolutional layers to reconstruct the image. Skip connections are the direct links between corresponding layers of the encoder and decoder, which help in preserving spatial information and fine details by allowing the model to reuse features from earlier layers. These connections are vital for tasks like colorization, where maintaining the structure and details of the original image is crucial.

For image colorization, the U-Net architecture is particularly beneficial because the skip connections ensure that the fine details and spatial information of the input gray-scale image are retained, which is essential for accurate colorization. The bottleneck captures abstract features, which helps the model understand the high-level content of the image, aiding in applying the correct colors. The decoder reconstructs the image by combining high-level abstract features with fine details, resulting in a colorized image that is both accurate and visually pleasing.

The idea is to replace the traditional U-Net encoder (the contracting path) with the convolutional layers of VGG19. This means using the pre-trained VGG19 model up to a certain depth, typically excluding the fully connected layers, as the feature extractor. The output from these convolutional layers is then passed through the U-Net decoder (the expanding path) to reconstruct the image and predict the U and V color channels.

To replace the traditional U-Net encoder with the VGG19 model, we load the pre-trained VGG19 model, specifically its convolutional layers, which will serve as the encoder. The convolutional layers of VGG19 are known for their ability to capture intricate details and hierarchical features from images, making them well-suited for the task of feature extraction in image colorization.

Incorporating VGG19 into the U-Net architecture involves using the output from the VGG19 convolutional layers as input to the U-Net decoder. The decoder, responsible for upsampling and reconstructing the image, uses transposed convolutions and convolutional layers to predict the U and V color channels.

A key component of the U-Net architecture is the skip connections, which help preserve spatial information that might be lost during the downsampling process in the encoder. In the modified architecture, these skip connections are maintained by linking the output of intermediate convolutional blocks from the VGG19 encoder to the corresponding layers in the U-Net decoder. This approach ensures that fine-grained details and spatial context from the input image are retained and effectively utilized during the upsampling process.

Architecture Overview :

-

U-Net Auto-Encoder Architecture:

This encoder comprises four distinct blocks, each equipped with specific convolutional layer arrangements:

- Each block contains three convolutional layers with 3x3 filters, applying strides of 1 and 2 alternately to reduce spatial dimensions while increasing feature depth. The first layer typically has 64 filters, doubling in each subsequent block, culminating at 512 filters in the fourth block.
- Batch normalization is employed post each convolution to standardize inputs, helping to accelerate the training process and stabilize the network.
- ReLU activations follow each convolutional layer, introducing non-linearities that enable the network to learn more complex patterns.

The output of each block is formulated as:

$$Y^{(i)}_{encoded} = \sigma(BN(Conv(Y^{(i-1)}_{encoded})))$$

where $Y^{(0)}_{encoded} = Y_{gray}$, and σ denotes the ReLU activation.

VGG19 Auto-Encoder Architecture:

I experiment with the VGG19 model up to different depths, to analyze the impact of feature abstraction on the performance of the image colorization task. Using shallower layers might preserve more local details, while deeper layers could provide more abstract and semantic features. Maintaining skip connections ensures that critical spatial information is retained, enhancing the overall quality of the colorized images.

*Depth 3 Configuration

- The first three blocks of VGG19 are adapted, each consisting of two to three convolutional layers with increasingly complex structures. The number of filters starts at 64 in the first block and increases to 256 by the third block.
- Each convolutional layer uses 3x3 filters with a stride of 1, and padding is included to maintain the size of the feature maps.
- Following each block, a max-pooling layer with a 2x2 filter and a stride of 2 reduces the spatial dimensions, focusing on the most significant features.

*Depth 5 Configuration

- The first five blocks of VGG19 are adapted, each consisting of two to four convolutional layers with increasingly complex structures. The number of filters starts at 64 in the first block and increases to 512 by the fifth block.
- Each convolutional layer uses 3x3 filters with a stride of 1, and padding is included to maintain the size of the feature maps.
- Following each block, a max-pooling layer with a 2x2 filter and a stride of 2 reduces the spatial dimensions, focusing on the most significant features.

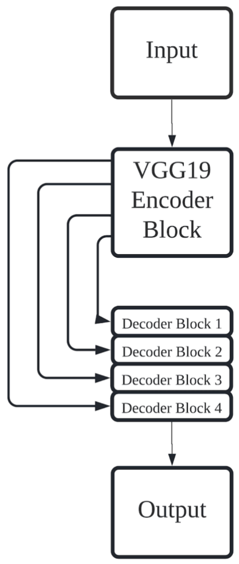

Figure 3.9: VGG-19 Architecture

Mathematically, the operations in each block of the encoder can be represented as:

$$Y_{encoded}^{(i)} = \sigma(BN(Conv(Y_{encoded}^{(i-1)})))$$

where $Y_{encoded}^{(0)} = Y_{gray}$, σ denotes the ReLU activation, BN represents batch normalization, and $Conv$ denotes convolution operations.

Decoder: The decoder mirrors the encoder's structure with four corresponding blocks, utilizing transposed convolutional layers to progressively upscale the encoded features and reconstruct the U and V channels. The decoder:

- Uses transposed convolutions to increase spatial dimensions,
- Applies batch normalization and ReLU activation after each transposed convolution,
- Ensures detailed chrominance information is restored by the final layer.

$$UV_{predicted}^{(i)} = \sigma(BN(TransConv(UV_{predicted}^{(i-1)})))$$

with $UV_{predicted}^{(0)} = Y_{encoded}^{(4)}$.

Multi-Skip Connections: Critical for preserving high-frequency details, multi-skip connections link each block of the encoder to the corresponding block of the decoder. These connections directly convey gradients and detailed information, enhancing the fidelity of the reconstructed colors. The network employs an L1 loss function, focusing on the chrominance channels:

$$L = \frac{1}{N}\sum_{i=1}^{N} \|UV_{predicted}^{(i)} - UV_{real}^{(i)}\|$$

Post-training, the predicted U and V channels are combined with the Y channel to form a complete YUV image. This image is then converted back to RGB using the following transformation matrix:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1.13983 \\ 1 & -0.39465 & -0.58060 \\ 1 & 2.03211 & 0 \end{bmatrix} \begin{bmatrix} Y \\ U \\ V \end{bmatrix}$$

The following figure displays our proposed framework illustrating the encoder and decoder blocks with multi-skip connections.

TRAINING STRATEGIES :

This section details the training strategy adopted for two deep learning models designed for image colorization: a custom autoencoder and a modified VGG19-based encoder. Both models utilize the YUV color space and share a consistent training approach to ensure comparability and optimal performance.

When training neural networks, evaluating the model's performance and understanding its learning process involves using various metrics. Two critical metrics in this context are the mini-batch loss and Root Mean Squared Error (RMSE).

Mini-batch Loss :

The mini-batch loss represents the average error computed over a subset of the training data, known as a mini-batch. This subset is used during each iteration of the training process. The primary goal of using mini-batches is to make the training process more efficient and to provide a regularizing effect, which can help the model generalize better.

Definition: Mini-batch loss is typically computed using a loss function such as Mean Squared Error (MSE). For a given mini-batch $\{(x_i, y_i)\}_{i=1}^{m}$, where x_i are the input samples and y_i are the corresponding ground truth values, the MSE loss is calculated as:

$$\text{MSE} = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2 \tag{3.5}$$

where $\hat{y}_i$ are the predicted values, and m is the number of samples in the mini-batch.

Root Mean Squared Error (RMSE)

RMSE is a standard performance metric for regression tasks, providing an aggregate measure of the model's prediction error.

Definition: RMSE is the square root of the average of the squared differences between predicted values and actual values. For a set of predictions $\hat{y}$ and true values y, the RMSE is calculated as:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2} \tag{3.6}$$

where n is the total number of predictions.

Training Parameters:

- **Data Splitting:** The datasets for both models are divided with 80% allocated for training, 10% for validation, and 10% for testing. This distribution ensures that a substantial portion of the data is used for model training while still providing adequate data for validation and testing to gauge model generalization and performance effectively.
- **Optimizer:** The training of both models is conducted using both stochastic gradient descent and Adam optimizer, known for their effectiveness in handling sparse gradients and adapting the learning rates during training.
- **Learning Rate:** An initial learning rate of 1.1×10^{-4} is set for both architectures, providing a balanced approach that allows for precise adjustments in weight updates necessary for the models' convergence.
- **Epochs:** Each model undergoes training for 50 ,100 and finally 200 epochs which turns out that 200 is the best duration chosen to sufficiently expose the models to the training enough.
- **Batch Size:** The custom autoe-ncoder processes mini-batches of 16, while the VGG19-based model handles smaller batches of 4.
- **Image Size:** The autoencoder is trained on images resized to 128×128 pixels, suitable for its network structure, whereas the VGG19 model uses larger 224×224 pixel images, aligning with its requirement for a higher resolution to effectively leverage pre-trained weights.

By standardizing the training parameters across both models, including data splitting, loss function, and specifically tailored batch sizes and image dimensions, the strategy ensures that any observed differences in performance are attributable to the models' architectural variations rather than disparities in their training configurations. This unified approach facilitates a fair assessment of each model's effectiveness in image colorization tasks, paving the way for further optimizations based on empirical performance data. The final performance of the network was measured using the test dataset.The tables below describe the results after training using test data.

After training the dataset using both VGG19 and the U-Net auto-encoder,we compared the performance of different auto-encoder models (U-Net, VGG19 with depths 3 and 5) on two datasets (Foraminifera and EuroSat) using two different optimizers (Adam and SGDM). The key metrics used for comparison were the loss function and RMSE (Root Mean Squared Error).

Table 3.7: Results After Training Auto-Encoders on Test Dataset using Adam Optimizer

Dataset	Encoder	Loss Function	RMSE
Foraminifera	U-Net	1.4	1.6
Foraminifera	VGG19 depth(3)	1.1	1.3
Foraminifera	VGG19 depth(5)	2.3	4.2
EuroSat	U-Net	3.3	2.4
EuroSat	VGG19 dpeth(3)	2.5	3.01
EuroSat	VGG19 depth(5)	4.8	5.2

Table 3.8: Results After Training Auto-Encoders on Test Dataset using SGDM Optimizer

Dataset	Encoder	Loss Function	RMSE
Foraminifera	U-Net	3.22	5.3
Foraminifera	VGG19 depth(3)	2.8	4.8
Foraminifera	VGG19 depth(5)	3.3	6.6
EuroSat	U-Net	3.4	5.6
EuroSat	VGG19 depth(3)	5.6	4.2
EuroSat	VGG19 depth(5)	6.1	5.3

For both datasets, the Adam optimizer consistently provides better performance in terms of lower loss function and RMSE compared to the SGDM optimizer for both U-Net and VGG19 depth(3). This suggests that Adam's adaptive learning rate capabilities help achieve more effective training and better generalization for these models.

Additionally, VGG19 depth(5) was excluded from the colorization task due to its relatively poorer performance compared to VGG19 depth(3) for both datasets. This indicates that increasing the depth beyond a certain point does not necessarily improve performance and may even hinder it due to overfitting or increased complexity.

The following Figures showing the resulted colored image performed by VGG19 and U-Net Autoencoder for both datasets.

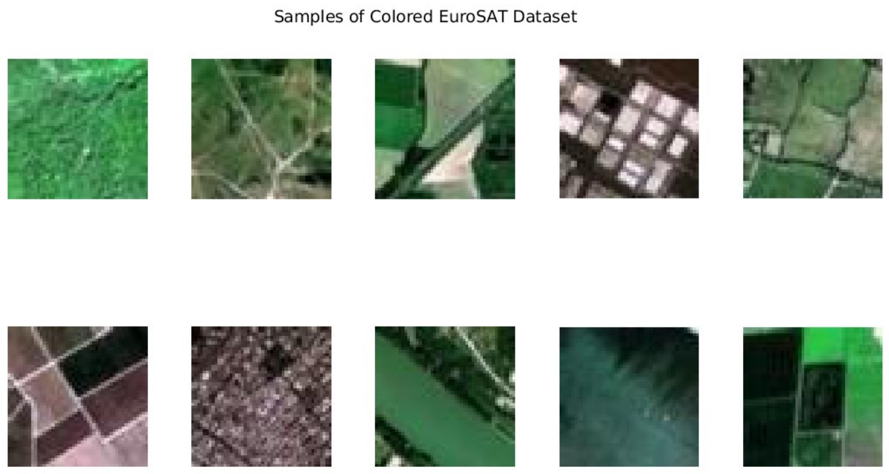

Figure 3.10: The resulted colored images for EuroSat dataset after applying VGG-19 Auto-Encoder

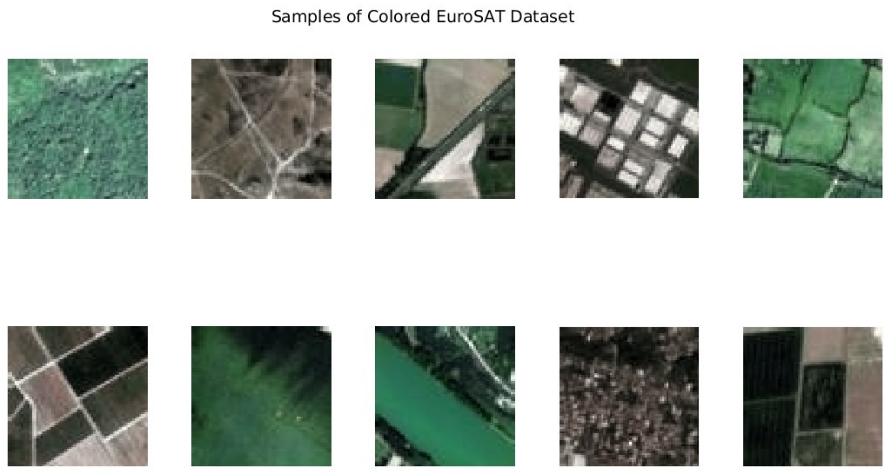

Figure 3.11: The resulted colored images for EuroSat dataset after applying U-Net Auto-Encoder

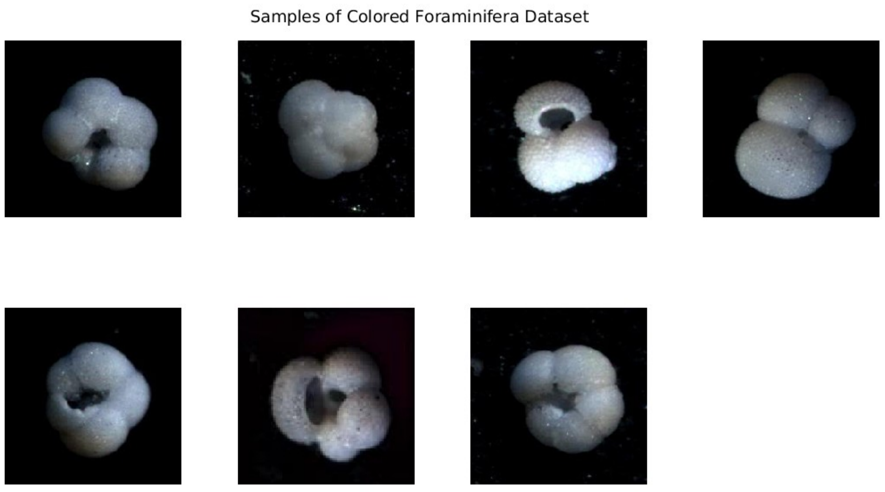

Figure 3.13: The resulted colored images for Foraminifera after applying VGG-19 Auto-Encoder

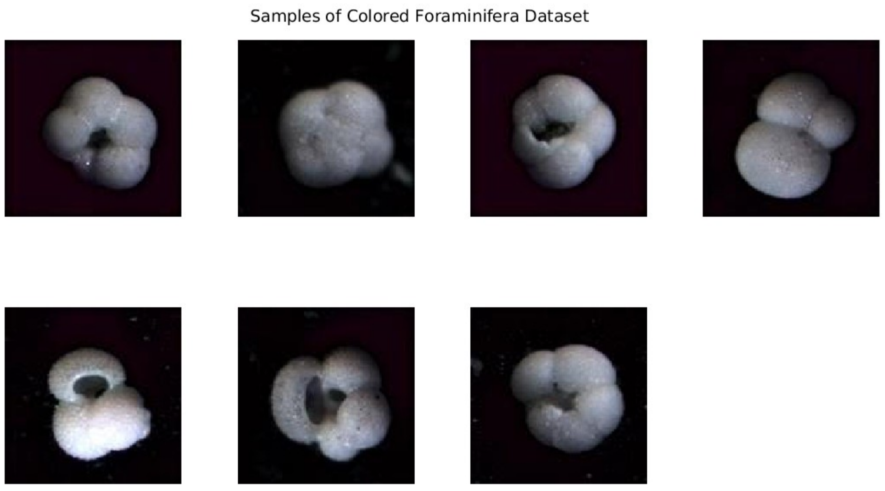

Figure 3.12: The resulted colored images for Foraminifera after applying U-Net Auto-Encoder

3.2 IMAGE CLASSIFICATION USING TRANSFER LEARNING

Convolutional Neural Networks (CNNs) are designed for image classification tasks. They consist of several key layers that are essential for learning and extracting features from images.These layers apply filters to the input image, detecting features such as edges, textures, and patterns. The convolution operation is defined as:

$$(I * K)(i,j) = \sum_{m} \sum_{n} I(i-m, j-n) \cdot K(m,n)$$

where I is the input image, K is the kernel, and (i,j) are the coordinates of the output feature map.

The ReLU (Rectified Linear Unit) function introduces non-linearity into the network, which allows it to learn from complex data:

$$f(x) = \max(0, x)$$

Pooling layers reduce the spatial dimensions of the feature maps, retaining the most important information. This process helps to control overfitting and reduces computational load. The most common pooling operation is max pooling, defined as:

$$P(i,j) = \max_{m,n} \{I(i+m, j+n)\}$$

where $P(i,j)$ is the pooled feature map.

These layers are used at the end of the network to make predictions based on the extracted features. Each neuron in a fully connected layer is connected to all neurons in the previous layer. The softmax function is often used in the output layer for classification tasks:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

where z is the input vector and $\sigma(z)_j$ is the$j$-th component of the output probability distribution.

Transfer learning involves using a pre-trained model and fine-tuning it for a new, often smaller, task-specific dataset. This approach showed in the belwo figure leverages the knowledge gained from the initial training task, making it possible to achieve better performance with less data and computational resources.Pre-trained Models such as VGG, ResNet, and Inception are commonly used for transfer learning due to their robust architectures and extensive training on large datasets like ImageNet.Fine-tuning involves adjusting the pre-trained model's

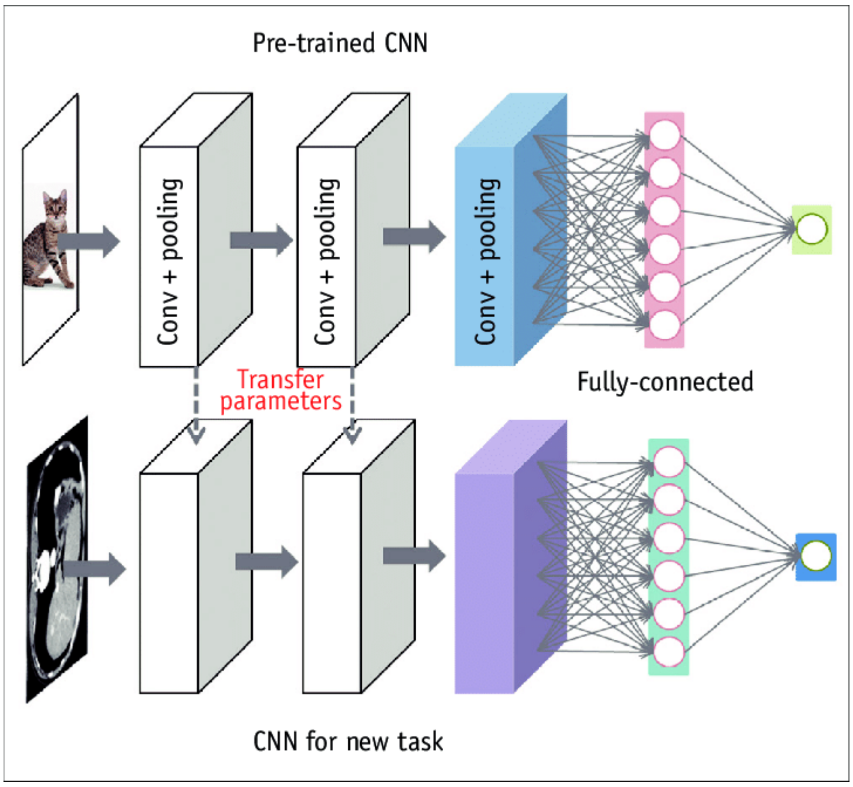

Figure 3.14: Transfer Learning Architecture

weights on the new dataset. Typically, the initial layers (which capture general features) are frozen, and only the final layers (task-specific features) are retrained. The process can be mathematically represented by updating the weights using gradient descent:

$$\theta = \theta - \eta \nabla_\theta J(\theta)$$

where θ represents the model parameters, η is the learning rate, and $J(\theta)$ is the loss function.

The softmax function is often used in the output layer for classification tasks:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

where z is the input vector and $\sigma(z)_j$ is the j-th component of the output probability distribution.

Advantages of Transfer Learning

- **Reduced Training Time:** Since the model is already partially trained, it requires less time to converge.

- **Improved Performance:** Leveraging learned features from large datasets can enhance performance, especially when the new dataset is small.
- **Less Data Required:** Effective even with limited data, making it suitable for tasks where data collection is challenging.

3.2.1 TRANSFER LEARNING CLASSIFIERS :

This section provides an in-depth overview of several prominent CNN architectures used for classifying the resulted colored images for both Foraminifera and EuroSat datasets.

- ResNet50

ResNet50 is a deep residual network architecture designed to address the vanishing gradient problem in deep neural networks by introducing residual connections. As it is shown in the below figure the architecture is composed of 50 layers, including 49 convolutional layers and a single fully connected layer. The primary innovation in ResNet50 lies in its use of residual blocks, which incorporate identity and convolution shortcuts to facilitate more efficient training of deeper networks. In traditional deep networks, as the number of layers increases, the gradient of the loss function can become extremely small, causing the network to stop learning or learn very slowly—a phenomenon known as the vanishing gradient problem. ResNet50 mitigates this issue through residual connections. These connections allow the gradient to bypass certain layers, ensuring that it remains strong and the network can continue to learn effectively even as it becomes deeper. The architecture's fundamental building block is the residual block, which comes in two varieties: identity blocks and convolutional blocks. Identity blocks preserve the original dimensions of the input, allowing the input to be added directly to the output. Convolutional blocks, on the other hand, alter the dimensions of the input to match the dimensions of the output, using convolutional layers to perform this transformation before addition.

Mathematically, the operation of a residual block can be described by the following equations. For an identity block:

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x} \tag{3.7}$$

Here, $\mathbf{x}$ is the input, $\mathcal{F}(\mathbf{x}, \{W_i\})$ represents the residual mapping to be learned, and $\mathbf{y}$ is the output of the residual block. The function $\mathcal{F}$ typically consists of a series of convolutional, batch normalization, and ReLU layers.

For a convolutional block, the equation is slightly modified to include a convolutional layer in the shortcut path to match the dimensions:

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + W_s\mathbf{x} \tag{3.8}$$

where W_s is a convolutional weight matrix that aligns the dimensions of the input $\mathbf{x}$ with the output of the residual function $\mathcal{F}$.

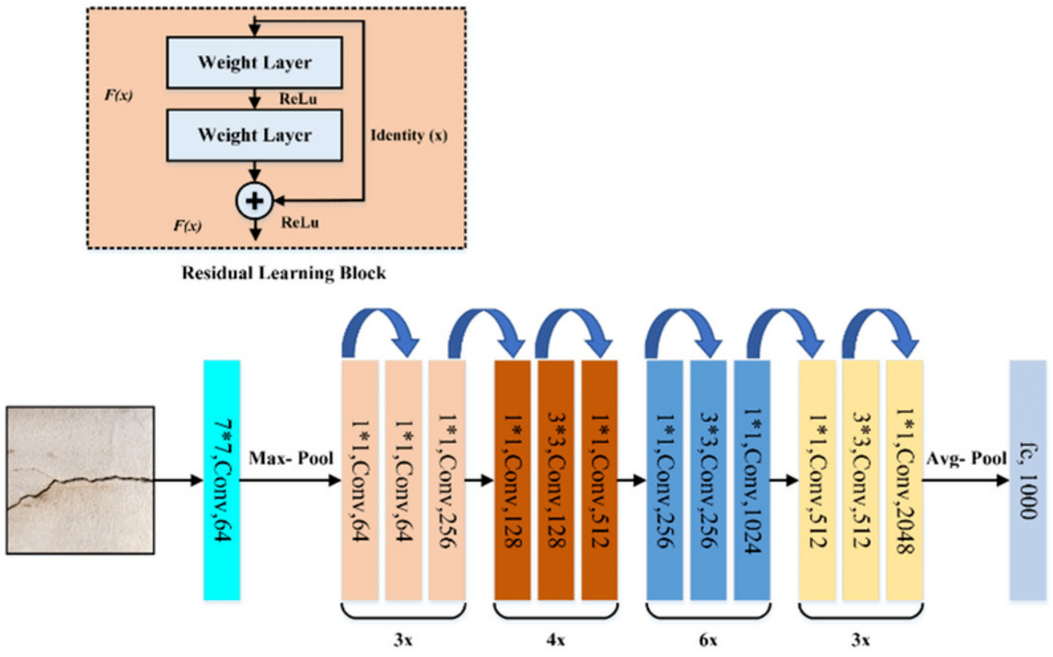

Figure 3.15: ResNet-50-model Architecture

- **ResNet18**

ResNet18 architecture provided in the following fi is a smaller version of the ResNet architecture, suitable for tasks with less complexity or limited computational resources. The architecture of ResNet18 consists of 18 layers, including 17 convolutional layers and a single fully connected layer. It uses basic residual blocks, which are simpler and contain fewer layers than the residual blocks in ResNet50.

The basic residual block in ResNet18 can be described by the following equation:

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x} \tag{3.9}$$

where the structure of $\mathcal{F}(\mathbf{x}, \{W_i\})$ is simpler compared to ResNet50, typically containing fewer convolutional layers.

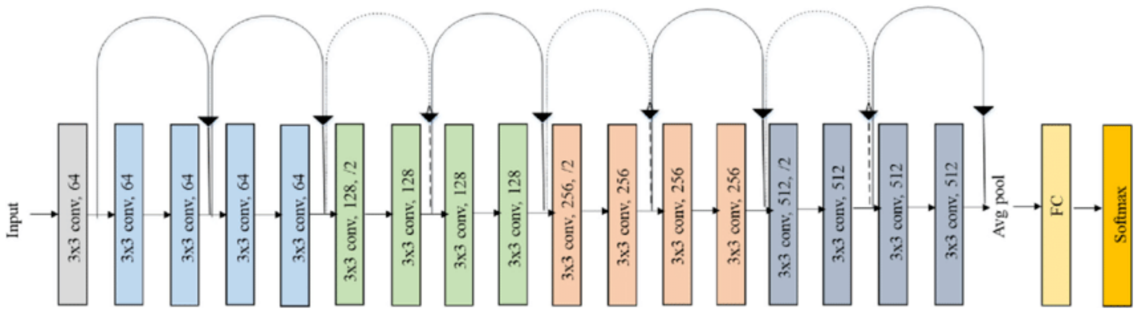

Figure 3.16: ResNet-18 Architecture

- GoogLeNet

GoogLeNet, also known as Inception v1, introduces the Inception module as it is illustrated in the below figure,this inceptio allows the network to capture features at multiple scales. The architecture of GoogLeNet consists of 22 layers; however, the complexity is managed using Inception modules. These modules are the core innovation of GoogLeNet, enabling the network to perform multiple operations, such as convolutions and pooling, in parallel.

An Inception module performs 1x1 convolutions to reduce the dimensionality of feature maps, thereby enhancing computational efficiency. It then applies 3x3 and 5x5 convolutions to extract features at different scales, providing a diverse array of spatial information. Pooling operations within the module aid in down-sampling and capturing dominant features. The parallel pathways within the Inception module execute these operations concurrently, and their outputs are concatenated to form the final output of the module.

Mathematically, the operations within an Inception module can be described as follows:

$$\text{Output} = \text{Concat}\left(\text{Conv}_{1\times1}(\mathbf{x}), \text{Conv}_{3\times3}(\mathbf{x}), \text{Conv}_{5\times5}(\mathbf{x}), \text{Pooling}(\mathbf{x})\right) \tag{3.10}$$

$$\mathbf{y} = \text{Concat}\left(\mathcal{F}_{1\times1}(\mathbf{x}, W_{1\times1}), \mathcal{F}_{3\times3}(\mathbf{x}, W_{3\times3}), \mathcal{F}_{5\times5}(\mathbf{x}, W_{5\times5}), \mathcal{P}(\mathbf{x})\right) \tag{3.11}$$

Here, **x** represents the input to the Inception module. $\mathcal{F}_{1\times1}(\mathbf{x}, W_{1\times1})$, $\mathcal{F}_{3\times3}(\mathbf{x}, W_{3\times3})$, and $\mathcal{F}_{5\times5}(\mathbf{x}, W_{5\times5})$ represent the 1x1, 3x3, and 5x5 convolutions respectively, with their corresponding weight matrices $W_{1\times1}$, $W_{3\times3}$, and $W_{5\times5}$. $\mathcal{P}(\mathbf{x})$ represents the pooling operation. The outputs of these operations are concatenated to form the final output **y** of the Inception module.

The use of 1x1 convolutions before larger convolutions is particularly significant. These smaller convolutions reduce the number of parameters and the computational cost, which increases efficiency and helps in reducing overfitting. By combining these various operations within a single module, GoogLeNet can effectively capture and process information at multiple scales while maintaining computational efficiency.

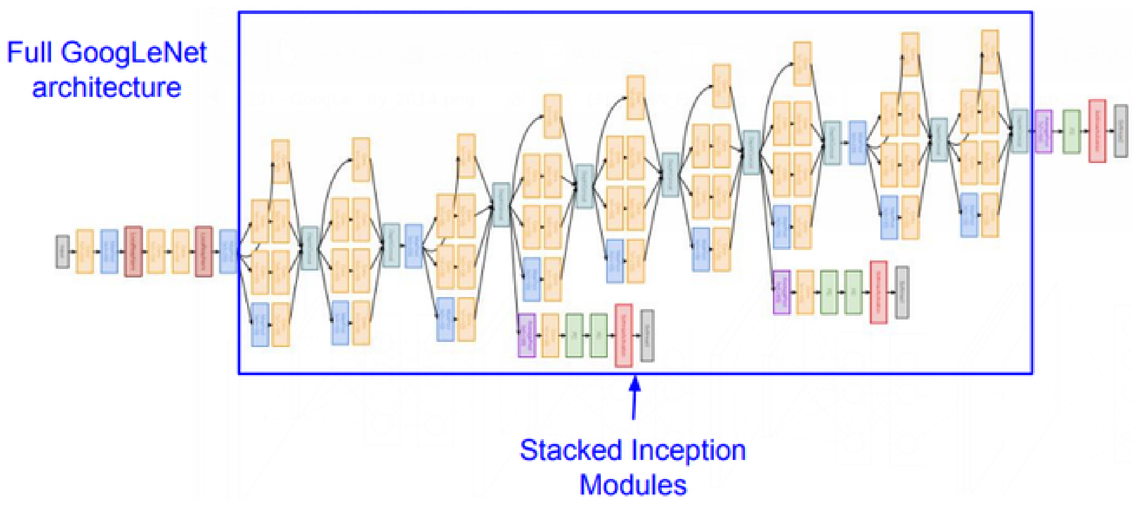

Figure 3.17: GoogleNet Architecture

- **MobileNetV2**

MobileNetV2 is designed for mobile and embedded vision applications, emphasizing efficiency and speed without significant loss in accuracy. The architecture of MobileNetV2 provided in the below figure primarily consists of depthwise separable convolutions followed by pointwise convolutions. A key innovation in MobileNetV2 is the use of inverted residuals with linear bottlenecks to enhance computational efficiency.

Depthwise separable convolutions split the convolution operation into two parts: depthwise convolutions and pointwise convolutions. The depthwise convolution applies a single filter per input channel, and the pointwise convolution applies a 1x1 filter to combine the outputs. Mathematically, these operations are represented as follows:

$$\mathrm{Conv}_{\mathrm{depthwise}}(\mathbf{x}) = \mathbf{x} * \mathbf{K}_{\mathrm{depthwise}} \tag{3.12}$$

$$\mathrm{Conv}_{\mathrm{pointwise}}(\mathbf{x}) = \mathbf{x} * \mathbf{K}_{\mathrm{pointwise}} \tag{3.13}$$

where $\mathbf{x}$ is the input, $*$ denotes the convolution operation, $\mathbf{K}_{\mathrm{depthwise}}$ is the depthwise convolution filter, and $\mathbf{K}_{\mathrm{pointwise}}$ is the pointwise convolution filter.

Inverted residuals in MobileNetV2 use linear bottlenecks to reduce the number of channels before applying depthwise separable convolutions, thus enhancing computational efficiency. This structure inverts the traditional residual block by first expanding the number of channels with a pointwise convolution, then applying the depthwise convolution, and finally reducing the number of channels back with another pointwise convolution. This method maintains rich feature representation while minimizing computational cost. The mathematical representation of the inverted residual block is given by:

$$\mathbf{y} = \mathbf{x} + \mathrm{Conv}_{\mathrm{inverted\ residual}}(\mathbf{x}) \tag{3.14}$$

where the inverted residual block expands the input $\mathbf{x}$ to a higher dimension, performs depthwise convolution, and then projects it back to a lower dimension.

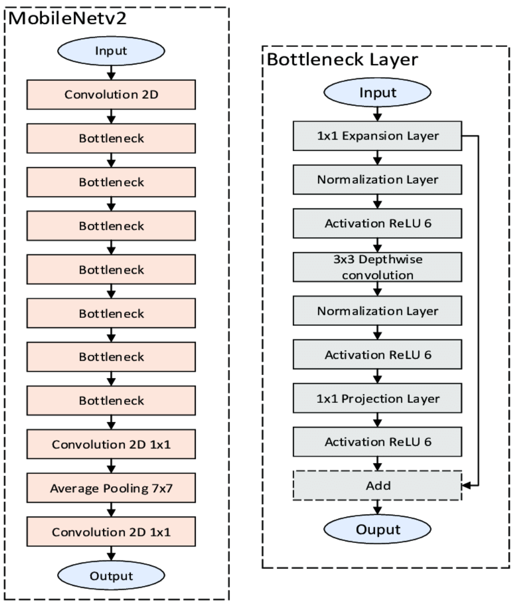

Figure 3.18: MobileNetV2

3.2.2 Ensemble Learning

After training each classifier individually, we can use ensemble learning to combine their predictions, enhancing the overall model performance. Ensemble learning is a machine learning paradigm where multiple models, often referred to as ”weak learners,” are trained to solve the same problem and then combined to obtain better performance than any of the individual models. The underlying principle is that a group of models can make more robust and accurate predictions than a single model due to the diversity and complementarity of the models' predictions.

Motivation for Using Ensemble Learning

- **Improved Accuracy**: By combining the outputs of several models, ensemble methods can achieve higher predictive accuracy. Different models may capture different aspects of the data or make different errors, and these differences can be averaged out, leading to better performance.
- **Reduced Overfitting**: Ensembles can reduce the risk of overfitting compared to individual models, especially if the individual models are diverse. Overfitting occurs when a model learns the noise in the training data rather than the underlying pattern.
- **Increased Robustness**: The combined output of multiple models is typically more robust to variations in the training data and to the peculiarities of individual models. This is particularly valuable in complex classification tasks where a single model might be prone to specific weaknesses.

Types of Ensemble Methods

There are various methods for combining the predictions of multiple models, including:

- **Boosting**: Sequentially trains models, each attempting to correct the errors of the previous one. Examples include AdaBoost and Gradient Boosting.
- **Bagging (Bootstrap Aggregating)**: Trains models on different subsets of the training data and averages their predictions. Random Forests are a common example.
- **Voting**: Aggregates the predictions of multiple models by majority vote (for classification) or averaging (for regression).
- **Stacking**: Combines the outputs of several base models using a meta-model, which learns to make the final prediction based on the outputs of the base models.

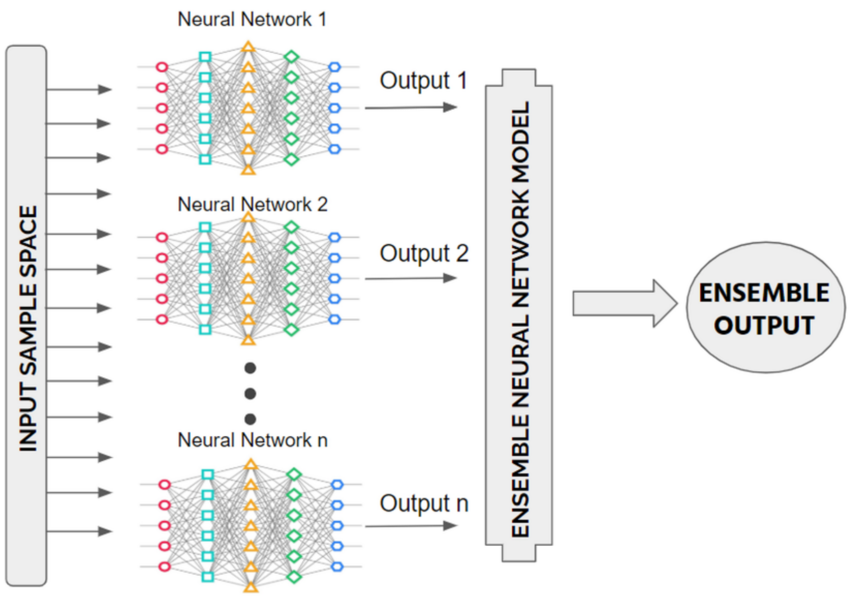

Figure 3.19: Ensemble Learning Framework

Weighted Averaging Ensemble Model.

Ensemble methods are powerful techniques in machine learning that combine the predictions of multiple models to improve accuracy and robustness. One such method is the **weighted averaging ensemble model**, a type of voting mechanism where different models contribute to the final prediction based on their individual performances.

Mathematical Formulation:

In a weighted averaging ensemble, each model M_i is assigned a weight w_i reflecting its reliability or accuracy. Given N models, the final prediction $\hat{y}$ for a data point is computed as a weighted sum of the individual predictions $\hat{y}_i$:

$$\hat{y} = \sum_{i=1}^{N} w_i \hat{y}_i, \tag{3.15}$$

where:

- $\hat{y}$ is the final prediction,
- $\hat{y}_i$ is the prediction from the i-th model,
- w_i is the weight assigned to the i-th model, with $w_i \geq 0$ and $\sum_{i=1}^{N} w_i = 1$.

For classification problems where models output probability distributions over classes, the weighted average of probabilities p_{ij} (the probability of class j from model i) is used:

$$p_j = \sum_{i=1}^{N} w_i p_{ij}. \tag{3.16}$$

The final class prediction $\hat{y}$ is the class with the highest combined probability:

$$\hat{y} = \arg\max_j \left(\sum_{i=1}^{N} w_i p_{ij} \right). \tag{3.17}$$

Determining Weights

The weights w_i can be determined based on various criteria, such as:

- **Validation Accuracy**: Using the accuracy of each model on a validation set as its weight.
- **Inverse Error Rate**: Assigning weights inversely proportional to the error rates of the models.
- **Other Performance Metrics**: Using metrics like precision, recall, or F1 score to determine weights.

For example, if the accuracy of model i on a validation set is acc_i, the weights can be normalized as:

$$w_i = \frac{\text{acc}_i}{\sum_{i=1}^{N} \text{acc}_i}. \tag{3.18}$$

Suitability of Weighted Averaging

Weighted averaging is particularly suitable when:

- **Model Diversity**: The individual models are diverse and capture different aspects of the data.

- **Varying Model Performance**: Models have different levels of accuracy, and it is beneficial to give more weight to better-performing models.
- **Robustness Requirement**: There is a need to improve the robustness and generalization of the final prediction by leveraging multiple models.

This method can mitigate the risk of overfitting by ensuring that no single model overly influences the final prediction, especially if that model performs poorly on certain subsets of the data.

3.2.3 K-Fold Cross-Validation

K-fold cross-validation is a robust method used for evaluating the performance of machine learning models, particularly in classification tasks. It helps in assessing how well the model generalizes to an independent dataset and is essential for preventing overfitting.In k-fold cross-validation, the dataset is randomly partitioned into k equal-sized folds. The model is trained and validated k times, each time using a different fold as the validation set and the remaining $k-1$ folds as the training set. The performance metric is averaged over the k iterations to provide a comprehensive evaluation.

Formally, let D be the dataset with n samples. The dataset is divided into k folds, $D_1, D_2, \ldots, D_k$, each containing $\frac{n}{k}$ samples. For each iteration i from 1 to k, the model is trained on $D \setminus D_i$ (all data except D_i) and validated on D_i.

The general procedure is as follows:

1. Divide the dataset D into k folds.
2. For each fold $i = 1$ to k:
    (a) Train the model on $D \setminus D_i$.
    (b) Validate the model on D_i.
    (c) Compute the performance metric (e.g., accuracy, precision, recall).
3. Average the performance metrics across all k folds.

Mathematical Formulation

Let M be the performance metric (e.g., accuracy). The metric for the i-th fold is denoted as M_i. The overall performance metric $\bar{M}$ is computed as:

$$\bar{M} = \frac{1}{k} \sum_{i=1}^{k} M_i \tag{3.19}$$

Where:

$$M_i = \text{Performance metric for the } i\text{-th fold}$$

Importance in Classification Tasks:

K-fold cross-validation is crucial for several reasons:

- **Generalization**: It provides a reliable estimate of the model's performance on unseen data by ensuring that every data point is used for both training and validation.
- **Bias-Variance Tradeoff**: By training and validating the model multiple times, k-fold cross-validation helps in finding a balance between bias and variance, thus improving the model's ability to generalize.
- **Efficient Use of Data**: Especially in scenarios with limited data, k-fold cross-validation makes efficient use of the available dataset, as every data point is used for both training and validation.
- **Robust Performance Metrics**: Averaging the performance metric over *k* folds provides a more stable and robust estimate than a single train-test split, reducing the impact of data variability.

4

EXPERIMENTS AND ANALYSIS :

In this chapter, we present experiments conducted using newly generated colored images obtained from supervised learning algorithms. The primary objective of these experiments is to utilize these colored images to perform classification tasks via transfer learning.

To compare the performance of the models, we will use metrics such as F1 score, which is the weighted average of Precision and Recall, providing a balance between them:

$$\text{F1 Score} = \frac{2 \times \text{TP}}{2 \times \text{TP} + \text{FP} + \text{FN}} \tag{4.1}$$

In these equations:

- **TP** (True Positives) are instances correctly predicted as positive.
- **TN** (True Negatives) are instances correctly predicted as negative.
- **FP** (False Positives) are instances incorrectly predicted as positive.
- **FN** (False Negatives) are instances incorrectly predicted as negative.

In addition to the F1 Score, we also use other performance metrics such as Recall, Precision, and Accuracy to evaluate the models:

- **Recall** (Sensitivity) is the ratio of correctly predicted positive observations to all the actual positives. It is defined as:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{4.1}$$

- **Precision** (Positive Predictive Value) is the ratio of correctly predicted positive observations to the total predicted positives. It is defined as:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{4.2}$$

- **Accuracy** is the ratio of correctly predicted observations to the total observations. It is defined as:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{4.3}$$

- **Confusion Matrix** allows visualization of the performance of classifier. Each row of the matrix represents the instances in an actual class, while each column represents the instances in a predicted class. For a multi-class classification problem, the Confusion Matrix C is defined as:

$$C = \begin{bmatrix} C_{1,1} & C_{1,2} & \cdots & C_{1,N} \\ C_{2,1} & C_{2,2} & \cdots & C_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ C_{N,1} & C_{N,2} & \cdots & C_{N,N} \end{bmatrix}$$

where $C_{i,j}$ is the number of instances of class i that were predicted as class j. The diagonal elements $C_{i,i}$ represent the counts of correctly classified instances for each class.

The Confusion Matrix provides insight into the types of errors being made by the classifier and can be used to calculate other performance metrics such as Precision, Recall, and Accuracy for each class, as well as overall performance metrics.

4.1 CLASSIFICATION EXPERIMENTS AND ANALYSIS OF FORAMINIFERA DATASET

All networks were pre-trained with the open-source image dataset ImageNet, which contains millions of labelled images. Hyperparameters for all networks were set as follows:

- Mini Batch Size: 30
- Max Epochs: 15
- Learning Rate: 10^{-4}
- Optimizer: adam

"The table below presents the F1 scores for the test data after training the pre-trained models using augmented data, which involved random reflections of the images top-bottom and left-right to create two additional images, and a third transformation that linearly scaled the original image along both axes with factors randomly extracted from the uniform distribution.

Table 4.1: Results After Training New Colored Images on Test Dataset Using Adam Optimizer

Colorization Model	**ResNet-50**	**ResNet-18**	**GoogleNet**	**MobileNetV2**
GAN	94%	96%	86%	94%
U-Net Auto-Encoder	83%	77%	75%	66%
VGG19 Auto-Encoder	87%	79%	79%	74%
VGG19+U-Net EL Model	88%	87%	85%	86%
U-Net+GAN EL Model	94%	93%	90%	91%
VGG19+GAN EL Model	96%	95%	92%	93%
VGG19+U-Net+GAN EL Model	95%	91%	84%	90%

The Foraminifera dataset was split into 80% for training, 10% for validation, and 10% for testing. As shown in Table 4.10, the GAN model is the best among individual models, achieving accuracies of 94%, 96%, 86%, and 94% on ResNet-50, ResNet-18, GoogleNet, and MobileNetV2, respectively.

For ensemble models, the VGG19+GAN EL model outperformed others, achieving the highest accuracies of 96%, 95%, 92%, and 93% on the respective network architectures. Thus, the VGG19+GAN EL model is the most effective overall, providing superior classification performance across the test dataset.

By using best ensemble model (VGG19)+(GAN),the following table presents a comparative analysis of performance measurements with using for the test data across different categories of Foraminifera. The performance metrics include precision, recall, F1 score, and accuracy for each class. These metrics provide a comprehensive evaluation of the classification model's effectiveness in accurately identifying each Foraminifera category.

Table 4.2: Comparison of Performance Measurements for Test Data Across Foraminifera Categories

Class	Precision	Recall	F1 Score	Accuracy
Bulloides	%94	%94	%94	%98
Dutertrei	1	%73	%84	%97
Incompta	1	1	1	1
Others	%97	1	%99	%99
Pachyderma	1	1	1	1
Ruber	%94	%94	%94	%98
Sacculifer	%83	1	%90	%97

The table compares the precision, recall, F1 score, and accuracy for various Foraminifera categories. Incompta, Others, and Pachyderma achieve perfect scores across all metrics, indicating excellent model performance in identifying these categories. Bulloides and Ruber also show high performance with precision, recall, F1 score, and accuracy all above 94%. Sacculifer has a high recall and accuracy (100% and 97% respectively) but lower precision (83%) and F1 score (90%). Dutertrei, however, presents a unique case with a high accuracy of 97% despite a lower recall of 73%, which suggests that while the model correctly identifies a large portion of the overall instances, it struggles to identify all true Dutertrei instances, likely due to a higher rate of false negatives. This discrepancy highlights the importance of considering multiple metrics to fully understand model performance across different categories.

The confusion matrix below provides a detailed breakdown of the performance of the ensemble model that combines Model 1 and Model 3 in classifying various Foraminifera species.

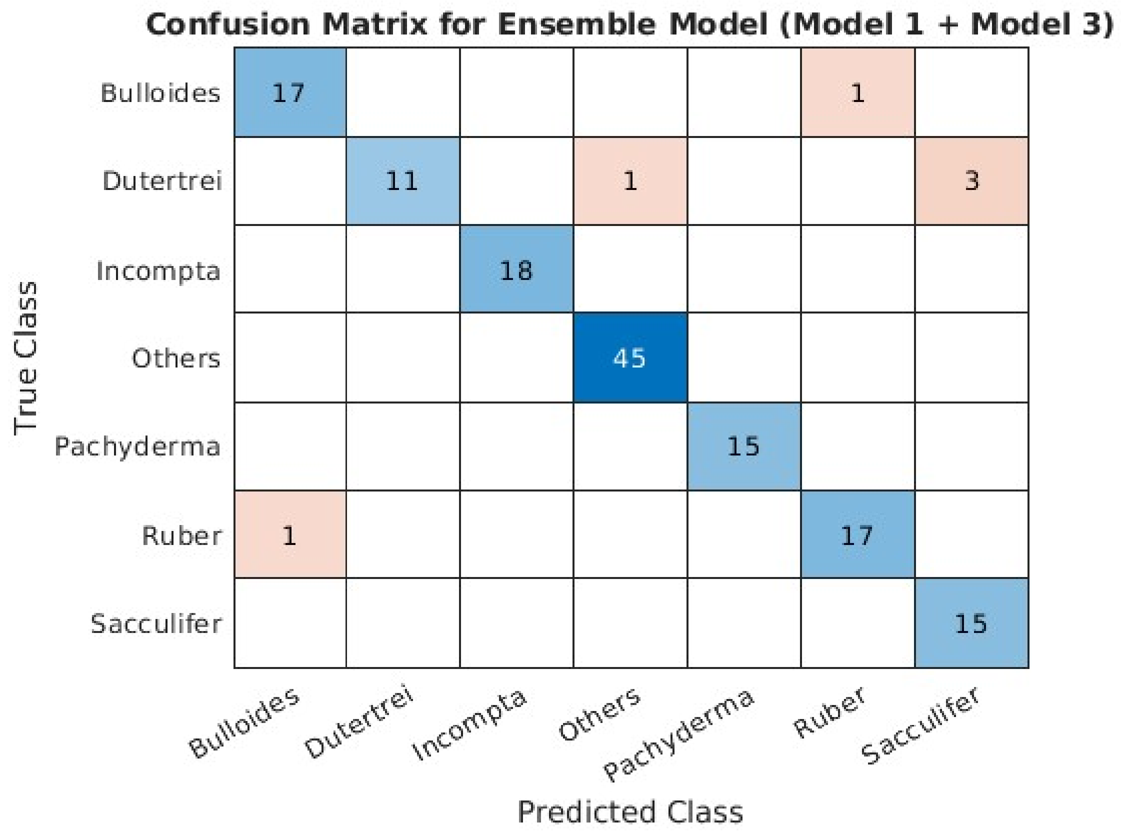

Figure 4.1: Confusion Matrix for Ensemble Model (Model 1 + Model 3)

The model correctly classified 17 instances of Bulloides, but misclassified 1 instance as Dutertrei and 1 as Ruber. For Dutertrei, out of 16 instances, the model correctly identified 11, misclassifying 1 as Incompta and 3 as Ruber. This lower recall (73%) for Dutertrei indicates difficulty in distinguishing Dutertrei from other species, particularly Ruber, likely due to similarities in their features causing confusion. In contrast, the model accurately classified all 18 instances of Incompta, demonstrating high accuracy. It also showed high performance for the 'Others' category, correctly classifying all 45 instances. For Pachyderma, the model achieved perfect classification with all 15 instances correctly identified. The Ruber category had 17 correctly classified instances, with 1 misclassified as Bulloides, indicating minor confusion between these species. Lastly, Sacculifer was perfectly classified with all 15 instances correctly identified.

To enhance the comparison of our work with methods discussed in the literature on foraminifera classification [1], [2], we adopt a rigorous scientific approach. Initially, we employ a 4-fold cross-validation technique to ensure robust evaluation of our model. Subsequently, we extend our analysis by using 5-fold cross-validation to assess the impact of different k-fold values on model performance.

After training and evaluating our model using both k-fold strategies, we compare the results of ensemble models from our study with those presented in the referenced papers. This comprehensive comparison highlights the effectiveness of our approach relative to existing studies in the field.

The table below illustrates the results of our study using 4-fold cross validation.

Table 4.3: Results of Forminifera Test Dataset using 4-Fold Cross Validation

Colorization Model	**ResNet-50**	**ResNet-18**	**GoogleNet**	**MobileNetV2**
GAN	90%	85%	76%	76%
U-Net Auto-Encoder	80%	78%	70%	68%
VGG19 Auto-Encoder	81%	79%	74%	70%
VGG19+U-Net EL model	85%	84%	79%	78%
U-Net+GAN EL model	91%	90%	81%	83%
VGG19+GAN EL model	90%	90%	84%	85%
VGG19+U-Net+GAN EL model	95%	94%	90%	89%

It can be shown that the GAN model shows solid performance with ResNet-50 (90%) and ResNet-18 (85%) but performs less well with GoogleNet (76%) and MobileNetV2 (76%). The U-Net Auto-Encoder and VGG19 Auto-Encoder models also show moderate performance across all architectures, with U-Net Auto-Encoder achieving a notable 80% with ResNet-50 and VGG19 Auto-Encoder reaching 81% with ResNet-50.Comparing the ensemble learning (EL) models, the VGG19+U-Net EL model and the U-Net+GAN EL model both demonstrate enhanced performance, outperforming the single models. The U-Net+GAN EL model achieves high scores with ResNet-50 (91%) and ResNet-18 (90%), and also performs strongly with GoogleNet (81%) and MobileNetV2 (83%). The VGG19+GAN EL model similarly shows improved performance, especially with ResNet-50 (90%) and ResNet-18 (90%).However, the VGG19+U-Net+GAN EL model emerges as the best performing ensemble learning model, achieving the highest scores across all architectures: 95% with ResNet-50, 94% with ResNet-18, 90% with GoogleNet, and 89% with MobileNetV2. This indicates that combining the strengths of VGG19, U-Net, and GAN into a single ensemble model provides the most robust and consistent performance across various neural network architectures.

Table 4.4: Results of our proposed work compared to other ensemble models reported here [1]

Colorization Model	ResNet-50	ResNet-18	GoogleNet	MobileNetV2
Percentile(3)+Luma Scaling(3)+Means Reconstruction(3)	87.7%	86.8%	82.1%	88.1%
Gaussian(3)+Luma Scaling(3)+Means Reconstruction(3)	87.9%	85.9%	79.8%	87.2%
Percentile(2)+Gaussian(2)+Luma Scaling(2)+Means Reconstruction(2)+HSVPP(2)	88.5%	88.0%	84.0%	88.8%
Percentile(1)+Gaussian(1)+Luma Scaling(1)+Means Reconstruction(1)+HSVPP(1)+GraySet(5)	90.6%	86.5%	84.1%	89.7%
VGG19+U-Net+GAN	95.0%	94.0%	90.0%	89.0%

The table above presents a performance comparison of various colorization models using different neural network architectures: ResNet-50, ResNet-18, GoogleNet, and MobileNetV2. The models include several configurations involving Percentile, Gaussian, Luma Scaling, Means Reconstruction, and HSVPP methods, compared against our ensemble learning model, VGG19+U-Net+GAN.

The Percentile(3)+Luma Scaling(3)+Means Reconstruction(3) model shows good performance across the board, with notable scores such as 87.7% for ResNet-50 and 88.1% for MobileNetV2. Similarly, the Gaussian(3)+Luma Scaling(3)+Means Reconstruction(3) model achieves comparable results, with its highest score of 87.9% on ResNet-50.

A combined approach of Percentile(2)+Gaussian(2)+Luma Scaling(2)+Means Reconstruction(2)+HSVPP(2) improves the performance, particularly for ResNet-50 (88.5%) and MobileNetV2 (88.8%). The most advanced configuration among the non-ensemble models, Percentile(1)+Gaussian(1)+Luma Scaling(1)+Means Reconstruction(1)+HSVPP(1)+GraySet(5), achieves the highest performance among these models with a score of 90.6% on ResNet-50, although it shows variability across other architectures, such as 86.5% for ResNet-18 and 89.7% for MobileNetV2.

In contrast, our ensemble learning model, VGG19+U-Net+GAN, demonstrates superior and consistent performance across all architectures, outperforming all other models. It achieves remarkable scores of 95.0% for ResNet-50, 94.0% for ResNet-18, 90.0% for GoogleNet, and 89.0% for MobileNetV2. These results indicate the robustness and effectiveness of our ensemble model, leveraging the combined strengths of VGG19, U-Net, and GAN to provide the highest accuracy and reliability in colorization tasks.

Model	**Precision (%)**	**Recall (%)**	**F1 Score (%)**	**Accuracy (%)**
Novices (max)	65%	64%	63%	63%
Experts (max)	83%	83%	83%	83%
ResNet50 + Vgg16 [2]	84%	86%	85%	85%
Vgg16 [2]	80%	82%	81%	81%
Best Ensemble model [1]	90.9%	90.6%	90.6%	90.7%
Proposed Ensemble	96%	95%	95%	95%

Table 4.5: Precision, recall, accuracy and F1 score comparison between models reported in [1], [2], and the best ensemble presented in the paper.

The above table shows that our proposed ensemble model significantly outperforms novices, experts, and previous ensemble models. Novices exhibit the lowest performance, with precision, recall, F1 score, and accuracy around 63-65(%). Experts perform better, achieving 83(%) across these metrics. The ResNet50 + Vgg16 ensemble model further improves these scores to 84-86(%), while Vgg16 alone scores slightly lower at 80-82(%). The best ensemble model prior to the proposed one achieves around 90.6-90.9(%). In contrast, the proposed ensemble model achieves the highest scores, with precision, recall, F1 score, and accuracy all reaching 95-96(%).

Class	Precision (%)	Recall (%)	F_1 Score (%)	Accuracy (%)
Bulloides	96%	93%	94%	98%
Ruber	96%	94%	95%	98%
Sacculifer	100%	95%	97%	99%
Dutertrei	96%	94%	95%	99%
Incompta	95%	94%	95%	98%
Pachyderma	96%	97%	97%	99%
Other	93%	98%	95%	97%

Table 4.6: Precision, recall, F_1 score, and accuracy of the best ensemble across all classes of the dataset.

The performance measurements presented in the table above highlight that G. sacculifer exhibits perfect precision (100%), indicating no false positives, followed closely by other classes with precision values ranging from 95% to 96%, and the "Other" category at 93%. In terms of recall, the "Other" category and N. pachyderma stand out with the highest values at 98% and 97%, respectively, suggesting they effectively identify nearly all true positive instances. The remaining classes demonstrate recall values between 93% and 95%. The F_1 scores, which balance precision and recall, range from 94% to 97% across all classes, signifying a well-rounded performance. Accuracy metrics are uniformly high, ranging from 97% to 99%, indicating consistent overall effectiveness of the ensemble model.

Overall, G. sacculifer and N. pachyderma show the strongest performance across the metrics, while the "Other" category excels in recall despite having slightly lower precision. All classes exhibit high overall performance, suggesting that the ensemble model is effective across different categories.

The confusion matrix below reveals the best ensemble model's performance across Foraminifera species using 4-fold cross-validation. The model correctly classified most instances, particularly for Others (88/91), but had some misclassifications. Specifically, Bulloides was misclassified as Incompta (2 instances) and as Others and Ruber (1 instance each). Dutertrei was misclassified as Bulloides, Others, and Sacculifer (1 instance each). Incompta had high accuracy with only 1 misclassification as Others. Pachyderma and Ruber also had a few misclassifications. The model demonstrates robustness but shows some confusion between closely related classes such as Bulloides and Incompta, and Ruber and Sacculifer.

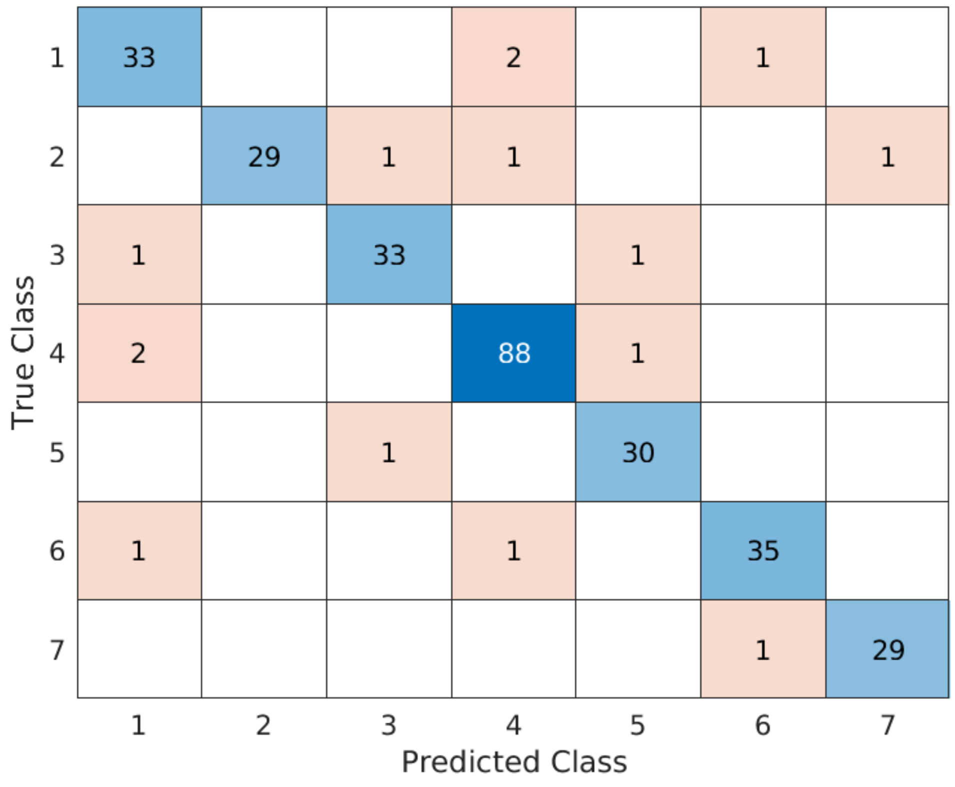

Figure 4.2: Confusion Matrix for the Best Ensemble Model Using 4-Fold Cross Validation

The confusion matrix above demonstrates that the ensemble model performs well across most classes, particularly for Others, which had the highest number of correct classifications. For Bulloides, the model correctly classified 33 instances, but misclassified 2 instances as Incompta and 1 instance each as Others and Ruber. Dutertrei had 32 true instances, with 29 correctly identified. Misclassifications for Dutertrei included 1 instance each as

Bulloides, Others, and Sacculifer. Incompta showed high accuracy, with 33 correct classifications and only 1 misclassified as Others. The Others category had the highest number of correct classifications, with 88 out of 91 instances correctly identified, but 2 instances were misclassified as Bulloides and 1 as Incompta. For Pachyderma, the model correctly classified 30 instances, with 1 misclassification as Dutertrei. Ruber, with 38 true instances, had 35 correctly identified, and 1 instance each misclassified as Bulloides, Others, and Sacculifer. Lastly, Sacculifer was accurately classified in 29 instances, with only 1 misclassification as Ruber. This analysis indicates that while the model demonstrates robust performance overall, there is still some confusion between closely related classes such as Bulloides and Incompta, and Ruber and Sacculifer.

4.2 CLASSIFICATION EXPERIMENTS AND ANALYSIS OF EUROSAT DATASE

All networks were pre-trained with the open-source image dataset ImageNet, which contains millions of labeled images. Additionally, the EuroSat dataset was split into 60% for training, 20% for validation, and 20% for testing.

- Mini Batch Size: 80
- Max Epochs: 15
- Learning Rate: 10^{-4}
- Optimizer: adam

Firstly, we conducted a comparative analysis using two sets of F1 score performance data. The first set, presented in Table 1, displays the F1 scores for image classification tasks that utilized colored images generated through supervised learning colorization in combination with RGB images. The second set, shown in Table 2, contains the F1 scores for image classification tasks that employed only the colored images without integrating them with RGB images. This comparison aims to determine which method yields better performance, thereby informing the selection of the most effective approach for ensemble methods in image classification.

Table 4.7: F1 Score Performance for Image Classification Using Combined Various Colorization Methods and RGB Images

Colorization Model	**ResNet-50**	**ResNet-18**	**GoogleNet**	**MobileNetV2**
GAN	98.6%	96.1 %	%92.4	%95.3
U-Net Auto-Encoder	%95.0	%95.5	%94.4	%95.7
VGG19 Auto-Encoder	%98.1	%97.7	%96.1	%96.9

Table 4.8: F1 Score Performance for Image Classification Using Only Various Colorization Methods

Colorization Model	**ResNet-50**	**ResNet-18**	**GoogleNet**	**MobileNetV2**
GAN	95.1%	94.2%	%92.0	%94.0
U-Net Auto-Encoder	93.1%	%92.5	%90.0	%91.3
VGG19 Auto-Encoder	%95.2	%94.1	%93.4	%93.2

The comparison between the two models indicates that the image classification performance is generally higher when using a combination of colorized and RGB images, as shown in Table 4.6. Specifically, the F1 scores for all models (GAN, U-Net Auto-Encoder, and VGG19 Auto-Encoder) are consistently higher across all classifiers (ResNet-50, ResNet-18, GoogleNet, and MobileNetV2) when using the combined approach. In contrast, Table 4.7 shows lower F1 scores when only colorized images are used. Therefore, the best approach for performing an ensemble model is to use a combination of colorized and RGB images, as this yields superior classification performance.

Table 4.9: F1 Score Performance of Individual and Ensemble Models Using Various Colorization Methods Combined with RGB Images

Colorization Model	ResNet-50 (%)	ResNet-18 (%)	GoogleNet (%)	MobileNetV2 (%)
GAN	98.6%	96.1%	92.4%	95.3%
U-Net Auto-Encoder	95.0%	95.5%	94.4%	95.7%
VGG19 Auto-Encoder	98.1%	97.7%	96.1%	96.9%
VGG19+U-Net EL model	98.6%	97.2%	95.8%	96.1%
U-Net+GAN EL model	98.8%	96.2%	94.4%	95.7%
VGG19+GAN EL model	99.5%	98.5%	96.0%	97.3%
VGG19+U-Net+GAN EL model	99.1%	98.6%	96.6%	97.2%

The table above compares several individual and ensemble colorization models across four different evaluation metrics: ResNet-50, ResNet-18, GoogleNet, and MobileNetV2. Among the individual models, the GAN model shows strong performance on ResNet-50 (98.6%) and ResNet-18 (96.1%) but lower scores on GoogleNet (92.4%). The U-Net Auto-Encoder performs consistently across all metrics, with scores like 95.0% on ResNet-50 and 95.5% on ResNet-18, but it does not achieve the highest scores overall. The VGG19 Auto-Encoder model exhibits high performance across all metrics, particularly on GoogleNet (96.1%) and MobileNetV2 (96.9%).

The ensemble models show improved performance over individual models. The VGG19+U-Net EL model performs well, especially on ResNet-50 (98.6%) and ResNet-18 (97.2%), but does not significantly outperform the best individual models. The U-Net+GAN EL model shows a slight improvement, particularly on ResNet-50 (98.8%). The VGG19+GAN EL model demonstrates significant improvements across all metrics, achieving the highest scores on ResNet-50 (99.5%) and ResNet-18 (98.5%). The VGG19+U-Net+GAN EL model exhibits the best overall performance, achieving the highest or near-highest scores across all metrics, with scores like 99.1% on ResNet-50, 98.6% on ResNet-18, 96.6% on GoogleNet, and 97.2% on MobileNetV2.

In conclusion, the VGG19+U-Net+GAN EL model is the best ensemble model, achieving the highest or near-highest scores across all evaluation metrics, making it the most robust and effective colorization model in this comparison.

Table 4.10: Accuracy Metrics of Individual and Ensemble Models for EuroSAT Dataset Classification Using ResNet-50 Architecture

Colorization Model	**ResNet-50 (%)**
GAN	98.7%
U-Net Auto-Encoder	95.0%
VGG19 Auto-Encoder	98.1%
VGG19+U-Net EL model	98.6%
U-Net+GAN EL model	98.8%
VGG19+GAN EL model	99.5%
VGG19+U-Net+GAN EL model	99.1%
Best Model[45]	98.5%

To rigorously compare our work with similar EuroSAT dataset classification methods proposed in previous studies[45].In the following table, we computed the accuracy metrics for both individual models and ensemble models. By conducting this comparison, we aim to contextualize our findings within the existing body of research and highlight the performance improvements achieved through our methodologies. Specifically, we evaluate the accuracies of various models, including Generative Adversarial Network, U-Net Auto-Encoder, and VGG19

Auto-Encoder, as well as ensemble models such as VGG19 combined with U-Net Encoder-Decoder Learning model, U-Net combined with Generative Adversarial Network Encoder-Decoder Learning model, and VGG19 combined with U-Net and Generative Adversarial Network Encoder-Decoder Learning model. These comparisons are crucial for validating the robustness and efficacy of our proposed models against established benchmarks.

Among the models, the VGG19+GAN EL model achieves the highest accuracy at 99.5%, surpassing the VGG19+U-Net+GAN EL model, which scores 99.1%. The previous best model cited in the last row has an accuracy of 98.5%. Therefore, the VGG19+GAN EL model outperforms the previous best model[45] by a significant margin, demonstrating an improvement of 1.0% in accuracy.

Class	**Precision**	**Recall**	**F_1 Score**	**Accuracy**
AnnualCrop	99.6%	99.8%	99.7%	99.9%
Forest	99.3%	99.8%	99.5%	99.9%
HerbaceousVegtation	96.4%	99.3%	97.8%	99.5%
Highway	99.3%	98.7%	99.0%	99.8
Industrial	100%	98.6%	99.2%	99.8%
Pasture	100%	96.9%	98.4%	99.7%
PermanentCrop	98.6%	97.9%	98.3%	99.6%
Residential	98.5%	99.6%	99.1%	99.8%
River	99.8%	99.6%	99.7%	99.9%
SeaLake	99.8%	99.8%	99.8%	99.9%

Table 4.11: Precision, recall, F_1 score, and accuracy of the best ensemble across all classes of the dataset.

The performance of the model across various classes has been evaluated based on precision, recall, F1 score, and accuracy, revealing strong predictive capabilities with some variability between classes using best ensembel model. The table above show that the model demonstrates high performance across most classes, with precision, recall, F1 score, and accuracy generally exceeding 97%. The Industrial and Pasture classes achieve perfect precision (100%), while AnnualCrop, Forest, River, and SeaLake exhibit the highest recall (99.8%). AnnualCrop and River also have the top F1 scores (99.7%). Accuracy is uniformly high, around 99.8%-99.9%, with AnnualCrop, Forest, River, and

SeaLake reaching 99.9%. Conversely, HerbaceousVegetation shows the lowest metrics, with a precision of 96.4%, recall of 98.7%, F1 score of 97.6%, and accuracy of 99.3%, indicating it is the most challenging class for the model. Overall, the model performs exceptionally well, with slight room for improvement in HerbaceousVegetation and Pasture classes.

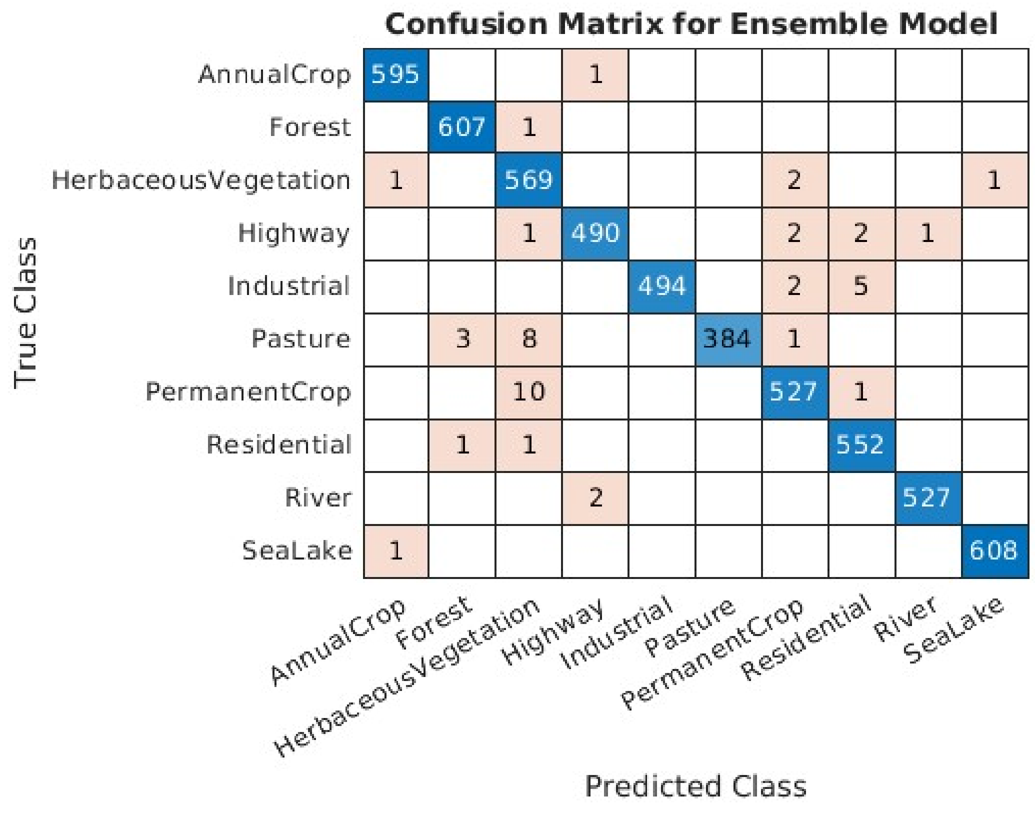

Figure 4.3: Confusion Matrix for Best Ensemble Model Performance on EuroSAT Dataset

The confusion matrix illustrates the performance of the best ensemble model, combining two pre-trained neural networks, on the EuroSAT dataset, which classifies various land use and land cover types.From the matrix, we observe high classification accuracy for most classes, with the diagonal values (correct classifications) being significantly higher than the off-diagonal values (misclassifications). For example, the classes ”Forest,” ”AnnualCrop,” and ”SeaLake” show very high accuracy, with 607, 595, and 608 correctly classified instances, respectively, and very few misclassifications. The ”PermanentCrop” and ”HerbaceousVegetation” classes also demonstrate strong performance, with 527 and 569 correct classifications.

However, there are some notable misclassifications. The ”Pasture” class has a few instances misclassified as

”Forest” (8 instances) and ”PermanentCrop” (3 instances), indicating some confusion between these classes, potentially due to similar spectral or textural features. Similarly, ”Industrial” and ”Highway” classes show minor misclassifications, with ”Highway” being confused with ”Industrial” (2 instances).

5

Conclusion

The study demonstrates the efficacy of integrating multiple colorization techniques with Convolutional Neural Networks (CNNs) within an Ensemble Learning (EL) framework to enhance image classification tasks. Specifically, the innovative use of Generative Adversarial Networks (GANs) and U-Net-based autoencoders for the colorization of grayscale images has significantly improved classification performance for both the foraminifera and EuroSAT datasets.

5.1 Foraminifera Classification

The classification of planktic foraminifera, microscopic marine organisms critical for paleoceanographic studies, was notably enhanced through the use of advanced deep learning techniques. The application of various colorization methods to grayscale images, combined with the power of ensemble CNNs, resulted in superior accuracy and reliability. This approach not only facilitated the detailed morphological analysis of foraminifera but also demonstrated potential for automating traditionally labor-intensive classification tasks, thus accelerating research in marine biology and environmental science.

5.2 EuroSAT Dataset Classification

Similarly, the classification of the EuroSAT dataset, which comprises Sentinel-2 satellite images across 13 spectral bands, benefited greatly from the proposed methodologies. The conversion of multi-spectral data into RGB images using sophisticated colorization techniques allowed for effective utilization of CNN-based models. The ensemble models, trained on these enhanced RGB images, consistently outperformed state-of-the-art methods, highlighting the robustness and versatility of the proposed system. This has significant implications for environmental monitoring, urban planning, and agricultural management, where accurate land cover classification is essential.

5.3 Ensemble Learning Effectiveness

The ensemble learning strategy proved to be particularly effective in mitigating the limitations of individual models, resulting in improved consistency and accuracy. By combining outputs from multiple models trained on differently processed images, the ensemble approach leveraged the strengths of each model while compensating for their weaknesses. This was evident in the high precision, recall, and F1 scores achieved across various models and datasets.

5.4 Future Work

Future research could explore the integration of additional colorization techniques and the application of this framework to other domains requiring detailed image analysis. Additionally, enhancing the training process with larger and more diverse datasets could further improve model performance and generalization capabilities.

References

[1] L. Nanni, G. Faldani, S. Brahnam, R. Bravin, and E. Feltrin, "Improving foraminifera classification using convolutional neural networks with ensemble learning," 2023.

[2] R. Mitra, T. Marchitto, Q. Ge, B. Zhong, B. Kanakiya, M. Cook, J. Fehrenbacher, J. Ortiz, A. Tripati, and E. Lobaton, "Automated species-level identification of planktic foraminifera using convolutional neural networks, with comparison to human performance," *Marine Micropaleontology*, vol. 147, pp. 16–24, 2019.

[3] F. Y. W. P. S. Z. J. Li, Z.; Liu, "A survey of convolutional neural networks: Analysis, applications, and prospects," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 12, pp. 6999–7019, Dec 2022.

[4] Z.-H. Zhou, *Ensemble Methods: Foundations and Algorithms*. Chapman and Hall/CRC, 2012.

[5] J. A. Cushman, *Foraminifera: Their Classification and Economic Use, 4th Revised and Enlarged Edition*. Harvard University Press, 1948.

[6] K. Segl, L. Guanter, F. Gascon, T. Kuester, C. Rogass, and C. Mielke, "S2etes: An end-to-end modeling tool for the simulation of sentinel-2 image products," *Remote Sensing of Environment*, vol. 255, p. 112300, 2022.

[7] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," https://arxiv.org/abs/1603.08511, 2016, accessed: 2024-06-23.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 25, 2012, pp. 1097–1105, [Google Scholar] [CrossRef].

[9] "How (and why) are black and white films colorized?" https://www.mentalfloss.com/article/26956/how-and-why-are-black-and-white-films-colorized, 2011, accessed: 2024-06-23.

[10] A. Levin, D. Lischinski, and Y. Weiss, "Colorization using optimization," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 689–694, 2004.

[11] T. Welsh, M. Ashikhmin, and K. Mueller, "Transferring color to greyscale images," *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 277–280, 2002.

[12] R. Ironi, D. Cohen-Or, and D. Lischinski, "Colorization by example," *Computer Graphics Forum*, pp. 201–210, 2005.

[13] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. John Wiley & Sons, 1973.

[14] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. Prentice Hall, 2002.

[15] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.

[16] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[17] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips, "Gpu computing," *Proceedings of the IEEE*, vol. 96, no. 5, pp. 879–899, 2008.

[18] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*. PMLR, 2015, pp. 448–456.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[21] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014, [Google Scholar] [CrossRef].

[22] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[23] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, vol. 27, 2014, pp. 2672–2680.

[24] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[25] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2015, pp. 234–241.

[26] A. Y. Hsiang, A. Brombacher, A. L. Johnson, P. W. Wilson, M. C. Rillo, and P. M. Hull, "The digital evolution of macroevolutionary theory: developing, deploying, and interpreting machine-learning approaches for studying the fossil record," *Paleobiology*, vol. 45, no. 4, pp. 520–536, 2019.

[27] F. Nguyen and Others, "A novel approach for foraminifera classification using a combination of convolutional neural networks and traditional image processing techniques," *Journal of Micropaleontology*, vol. XX, no. X, pp. XX–XX, 2020.

[28] P. Helber, B. Bischke, A. Dengel, and D. Borth, "Eurosat: Land use and land cover classification with sentinel-2," available at https://github.com/phelber/eurosat.

[29] S. Basu, S. Ganguly, S. Mukhopadhyay, R. DiBiano, M. Karki, and R. Nemani, "Deepsat: A learning framework for satellite imagery," in *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2015, pp. 1–10.

[30] W. Zhou, S. Newsam, C. Li, and Z. Shao, "Learning deep features for remote sensing image scene classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 8, pp. 4489–4501, 2017.

[31] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[32] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Advances in Neural Information Processing Systems*, vol. 27, 2014, pp. 3320–3328.

[33] D. Marmanis, K. Schindler, J. D. Wegner, S. Galliani, M. Datcu, and U. Stilla, "Classification and segmentation of urban areas using remote sensing data and pretrained convolutional neural networks," in *International Geoscience and Remote Sensing Symposium (IGARSS)*. IEEE, 2016, pp. 5073–5077.

[34] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4681–4690.

[35] Z. Liang, L. Zheng, M. Tan, S. Venkatesh, and S. Zhang, "Gan-based synthetic data generation for small sample size classification," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2804–2814, 2018.

[36] W. Liu, C. Gao, Z. Han, S. Gong, J. Chen, and H. Xu, "Multi-view convolutional neural networks for 3d shape recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3650–3659.

[37] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[38] T. Tieleman and G. Hinton, "Lecture 6.5 - rmsprop, coursera: Neural networks for machine learning," 2012, available at https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.

[39] S. Mallat, "A theory for multiresolution signal decomposition: The wavelet representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 7, pp. 674–693, 1989.

[40] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *European Conference on Computer Vision*. Springer, 2016, pp. 694–711.

[41] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 5967–5976.

[42] A. Name, "Image colorization using deep convolutional auto-encoder with multi-skip connections," *Application of Soft Computing*, vol. 27, pp. 3037–3052, 2023.

[43] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[44] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[45] P. Helber, B. Bischke, A. Dengel, and D. Borth, "Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 12, no. 7, pp. 2217–2226, 2019.

Acknowledgments

I wish to express my deepest gratitude to Professor Loris Nanni, my supervisor, for his invaluable guidance, scholarly insights, and unwavering support throughout the entirety of this thesis. His expertise has greatly enriched this research endeavor and shaped its scholarly rigor.

To my family, especially my mother and my husband , I owe an immense debt of gratitude. Their unconditional love, encouragement, and understanding have sustained me through the challenges of academic pursuit. Their unwavering belief in my abilities has been a constant source of strength.

I am also indebted to my friends, Farah, Zinab, and Lama, whose friendship and support have been invaluable. Their encouragement, thoughtful discussions, and shared academic interests have provided much-needed inspiration and camaraderie during this journey.