Work on Image Recognition

Introduction

The purpose of this work is to implement an image recognition system, combining conventional and unconventional machine learning techniques. The main approach is based on the K-Nearest Neighbors (KNN) algorithm for classifying images into 10 categories. Additionally, the impact of data preprocessing is evaluated through the use of an autoencoder and a pre-trained neural network model, CLIP embedding from OpenAI, on the classifier's performance.

Description of the Dataset

The CIFAR-100 dataset is used, which includes 60,000 images categorized into 20 and 100 parent classes and subclasses, respectively. Due to limited computational power, 10 subclasses were randomly selected, which belong to 9 parent categories. Each subclass contains 600 images, with 500 used for training and 100 for evaluation. Thus, the application of the image classification algorithm is performed on a dataset that has already been divided into training and evaluation sets. It is important to emphasize that each class is equally represented, as each contains the same number of images in the dataset.
Finally, it should be noted that for the remainder of this work, we will use the term "class" to refer to the subclasses of images, while we will always use the term "parent" to distinguish subclasses from the parent class.

Regarding the quality of the images, it should be noted that we are working with images of dimensions 32x32, which use three color channels (red, green, and blue). The pixels of each image in the dataset are stored in a single row, consisting of 3072 columns. The first 1024 columns contain values for the red color, the next 1024 for the green color, and the last 1024 for the blue color. Finally, the images are stored in a row-major layout, meaning that the first 32 values of the layout correspond to the red color of the first row of the image.

Data Processing Before Feeding the 3-NN Classifier

The main approach is implemented using the K-Nearest Neighbors (KNN) algorithm. KNN classifies each new image based on its similarity to K neighboring images in the training set. A similarity measure, such as the Euclidean distance, is used to determine the K nearest neighbors, and the category of the new image is defined as the most common category among

the K nearest neighbors. However, our primary concern in this work is to evaluate the effect of data preprocessing, as two additional approaches were implemented:

1. **Image Representation with Autoencoder**:
    - An autoencoder is used to represent each image in a lower-dimensional space (128 dimensions).
    - KNN is applied to the representations of the images instead of the original images.
2. **Image Representation with CLIP Embedding**:
    - A pre-trained neural network model from OpenAI is used to extract features from each image.
    - KNN is applied to the extracted features instead of the original images.

At this point, it should be noted that besides implementing classification with KNN, we leveraged some of the capabilities offered by the pre-trained model from OpenAI and also applied the Zero-Shot method. A brief description of the key mechanisms behind this approach will be provided in the relevant section.

Image Representation in a 128-Dimensional Space Using Autoencoder

Each image in the dataset describes an object, such as a tiger, leopard, bicycle, and others. However, the main object is depicted differently and against different backgrounds, which, in this case, can also be characterized as noise, due to the goal of image categorization, as they can confuse the primary classification algorithm. The main function of an autoencoder is to describe the same object while retaining only the essential elements. Specifically, the original image, in an abstract sense, is a point in a 3072-dimensional space. The goal of the autoencoder is initially to represent the same image in a lower-dimensional space, retaining as much information as possible so that the original image can be recovered. The similarity of the image after applying the autoencoder in relation to the original photograph will determine the effectiveness of the autoencoder, as well as the extent to which it is possible to retain all significant information during the dimension compression process.

In an ideal case, where the distortion of the image after processing is minimal, it means that during the dimension reduction process, the model successfully retains the significant information, removing only the noise. Therefore, we could feed the classifier with the representation of the image in a lower-dimensional space, aiding the task of image

classification, as we have removed the noise while still retaining all the significant information.

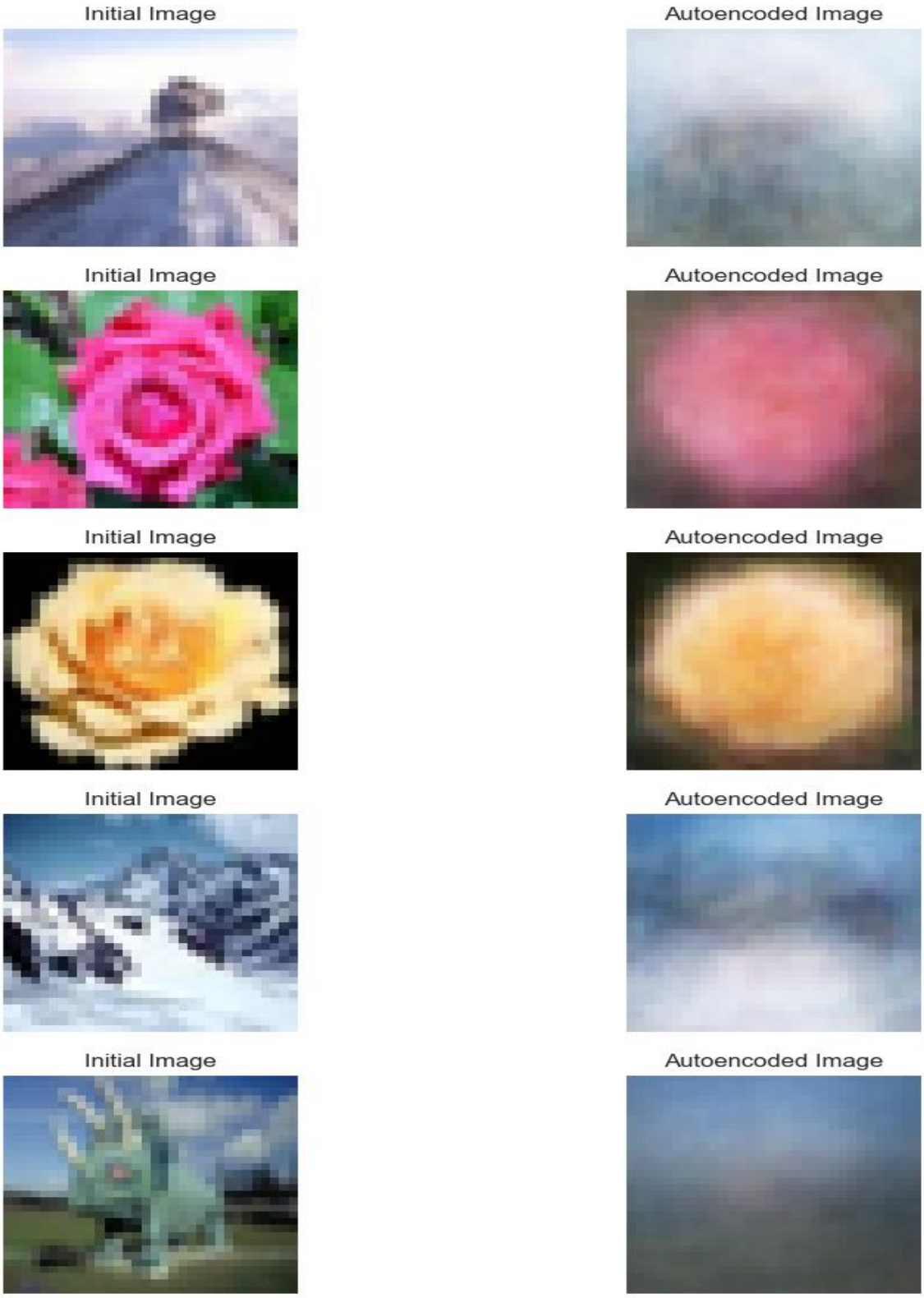
For the development of the autoencoder, a symmetrical sequential structure of neural networks was used, where the output of each neuron feeds all the neurons of the next layer. This structure is also characterized as symmetrical, as an equal number of layers with equal neuron density is used during the compression phase as is used during the decompression phase of the representation space dimensions.

Specifically, during the compression phase, 4 layers of neurons with 1024, 512, 256, and 128 neurons respectively were used, while, as indicated by the symmetrical structure, the decompression process includes another four layers with 256, 512, 1024, and 3072 neurons respectively.

Regarding the activation functions, ReLU is used in all intermediate layers, while Sigmoid is used in the final layer. The reasons for using ReLU are that it introduces non-linearity into the network, allowing it to learn more complex relationships, it is a computationally efficient function ($f(x) = \max(0, x)$) that simultaneously deactivates neurons that have negative input values, a feature particularly useful in image recognition issues where the input values are pixels. Finally, it should be mentioned that the above function has shown very good performance in empirical studies related to image recognition.

As for the sigmoid activation function used in the final layer, the main reason is that it produces outputs between 0 and 1, and the pixel values after normalization fall within the same range. Continuing with the description of the autoencoder, it should be noted that the Adam optimizer was employed, primarily due to the dynamically adjustable learning rate that allows during the training process, while the mean squared error was used as the cost function, a quite popular choice in continuous value prediction issues. Finally, for training the model, we experimented with various batch sizes from 1 to 16. Although smaller batch sizes generally lead to improved accuracy, in this case, through trial and comparison, it was determined to be much more effective to use a batch size equal to 8. This choice was confirmed both by the model's performance metrics such as MSE and R-squared and by visually comparing the original images with those that resulted after applying the autoencoder. Specifically, with a batch size of 1, the coefficient of determination was 30%,

while with a batch size of 8, the value of the aforementioned metric rose to just over 60%. Below are some sample photos from the best performance of this autoencoder.



Representation of the Image and Description in a 512-Dimensional Space via the CLIP Model.

CLIP (Contrastive Language-Image Pretraining) is a pre-trained model by OpenAI designed to associate images with their corresponding text descriptions. Its training process is based on identifying the differences between images and text descriptions, then embedding them in a shared representation space where similar images and their descriptions are very close to each other. In simpler terms, the goal is for the image of a tiger and its corresponding description, "the image of a tiger," to represent the same point in the shared representation space. Additionally, photos depicting similar objects will be closer together than those of completely different objects. The model then uses contrastive learning, pushing common images and descriptions to be close together while the opposite happens for dissimilar images and descriptions. For instance, the point depicting a tiger will have a smaller distance to a point representing a leopard compared to a point representing a bicycle.

Continuing in the process of image categorization, the CLIP model provides us with the opportunity to follow two approaches:

1. **Few-shot approach:** CLIP is capable of understanding new categories of images based on a few sample images. In simpler terms, using the photos we have in the training set, the model learns the new categories of images, allowing it to develop representations for these new categories. Once the model is trained on these new categories, we can use it to create representations for the images in our evaluation set. Subsequently, we can feed any of the classic machine learning algorithms with the image representations instead of the original images.
2. **Zero-shot approach:** In this case, we leverage the model's ability to combine the representations of images with their corresponding descriptions, enabling it to understand the meaning of new image categories through comparison with already understood language. Specifically, for the application of this approach, CLIP first projects each photo into the shared representation space. Then, similarly, we create a representation vector in the shared representation space for each new text label. Finally, for image categorization, we calculate the similarities between the representation vector of each photo and all the representation vectors of the text labels, keeping the description that is closest to each image. As a measure of similarity, we use the cosine distance.

Comparison and Commentary on Results After Applying KNN

In the context of the impact of data preprocessing as well as the performance of the k-nearest neighbors algorithm on categorization, we focused on key metrics referred to in English literature as accuracy, precision, and recall. Before proceeding with the presentation of the results for the different datasets, it is worth providing a brief description of each of these:

- **Accuracy:** Perhaps the most understandable metric, measuring the percentage of correctly assigned images relative to the total. Its main weakness is that it focuses on the model rather than each class separately, sometimes leading to misleading results, especially when there is class imbalance. For example, consider a model that categorizes all images as depicting a dog, and our dataset consists of 95 dogs and 5 cats. In this case, we would have a performance based on this metric of 95%, completely overlooking that it failed to recognize even one example from the other category.
- **Recall:** Largely complements the weakness of the previous metric as it focuses on each class and the corresponding percentage decreases in the class under examination for each instance that belonged to that class but was categorized differently. Based on the above example, the recall for the "cat" class would be 0%, and for the dog, it would be 100%. Special attention is given to this metric, especially when the class in question is more significant than the others.
- **Precision:** Focuses on each class separately and essentially measures how much we can trust that the examples of this class actually belong to that class rather than another. It works in opposition to recall, "penalizing" the presence of images from other categories that were misclassified in the examined category.

Below are detailed results for each class and the aforementioned metrics for the different datasets:

1. Original Images without Any Processing

Εκτιμήσεις αλγορίθμου	chimpanzee	dinosaur	girl	hamster	leopard	motorcycle	mountain	rose	shark	tiger
Πραγματικές κλάσεις										
chimpanzee	72	7	5	1	6	8	1	0	0	0
dinosaur	16	46	6	7	7	7	4	2	0	5
girl	12	17	33	10	8	3	1	4	1	11
hamster	6	9	20	37	16	1	3	4	0	4
leopard	15	19	11	6	32	6	1	0	1	9
motorcycle	26	8	8	4	10	36	2	1	4	1
mountain	7	3	2	2	1	11	62	3	9	0
rose	7	21	11	7	5	1	1	41	2	4
shark	10	7	7	3	6	5	13	0	48	1
tiger	13	18	16	10	14	7	1	1	1	19

Accuracy: 46%

	Precision	Recall
chimpanzee	54%	64%
dinosaur	31%	49%
girl	39%	20%
hamster	34%	47%
leopard	26%	44%
motorcycle	90%	26%
mountain	71%	80%
rose	72%	41%
shark	62%	72%
tiger	33%	16%

2. Representations Using Autoencoder

Εκτιμήσεις αλγορίθμου	chimpanzee	dinosaur	girl	hamster	leopard	motorcycle	mountain	rose	shark	tiger
Πραγματικές κλάσεις										
chimpanzee	72	7	5	1	6	8	1	0	0	0
dinosaur	16	46	6	7	7	7	4	2	0	5
girl	12	17	33	10	8	3	1	4	1	11
hamster	6	9	20	37	16	1	3	4	0	4
leopard	15	19	11	6	32	6	1	0	1	9
motorcycle	26	8	8	4	10	36	2	1	4	1
mountain	7	3	2	2	1	11	62	3	9	0
rose	7	21	11	7	5	1	1	41	2	4
shark	10	7	7	3	6	5	13	0	48	1
tiger	13	18	16	10	14	7	1	1	1	19

Accuracy: 43%

	Precision	Recall
chimpanzee	0.39	0.72
dinosaur	0.3	0.46
girl	0.28	0.33
hamster	0.43	0.37
leopard	0.3	0.32
motorcycle	0.42	0.36
mountain	0.7	0.62
rose	0.73	0.41
shark	0.73	0.48
tiger	0.35	0.19

3. Categorization with the Zero-Shot Approach

Accuracy: 89%

(It should be noted that all images from both the training and evaluation sets were used)

Εκτιμήσεις αλγορίθμου	chimpanzee	dinosaur	girl	hamster	leopard	motorcycle	mountain	rose	shark	tiger
Πραγματικές κλάσεις										
chimpanzee	98	1	0	1	0	0	0	0	0	0
dinosaur	2	82	0	1	10	0	0	0	0	5
girl	4	0	96	0	0	0	0	0	0	0
hamster	3	1	0	92	2	0	0	1	0	1
leopard	1	5	0	3	78	0	0	0	0	13
motorcycle	0	2	0	0	0	97	0	0	1	0
mountain	1	0	0	0	0	0	99	0	0	0
rose	0	1	0	0	1	0	0	98	0	0
shark	0	1	0	1	1	0	0	1	94	2
tiger	3	3	0	3	18	0	0	0	0	73

Accuracy: 91%

	Precision	Recall
chimpanzee	0.88	0.98
dinosaur	0.85	0.82
girl	1	0.96
hamster	0.91	0.92
leopard	0.71	0.78
motorcycle	1	0.97
mountain	1	0.99
rose	0.98	0.98
shark	0.99	0.94
tiger	0.78	0.73

Based on the results, we observe that processing the images with the CLIP model achieved the highest performance. In both the few-shot and zero-shot approaches, the accuracy metric reached approximately 90%, compared to the application of KNN on unprocessed images. In contrast, the autoencoder approach resulted in a decline in KNN performance.

Another point worth noting, primarily from the results after using CLIP embedding, is that similarities between different classes significantly affected the algorithm's performance. In this work, we categorized photos into 10 different classes, two of which belong to the same parent category. These are the classes of leopard and tiger, where we observe the algorithm showing the greatest confusion, likely due to their high similarity. Conversely, for categories that are completely dissimilar to the others, such as rose, mountain, and motorcycle, the algorithm's performance was nearly perfect.