DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI

SAPIENZA
UNIVERSITÀ DI ROMA

Control of Communication and Energy Networks

Report

Implementation of a Siamese Neural Network on Beach Litter Dataset

Ludovica Cartolano
cartolano.1796046@studenti.uniroma1.it

Supervised by
PhD Danilo Menegatti

Master in Control Engineering

Department of Computer, Control and Management Engineering
"Antonio Ruberti" (DIAG)
University of Rome "La Sapienza"
AY 2023/2024

Contents

List of Figures

Chapter 1

Introduction

Siamese neural networks are an innovative architecture tailored for comparing and distinguishing between various inputs, making them ideal for applications like face recognition, signature verification, and anomaly detection. These networks consist of two or more identical subnetworks that share the same parameters and weights but receive different inputs. During training, they learn to project these inputs into a lower-dimensional space, known as embeddings.

The effectiveness of Siamese networks lays on their use of distance metrics, such as Euclidean distance or cosine similarity, to evaluate the proximity between embeddings. The training process focuses on minimizing this distance for similar input pairs and maximizing it for dissimilar ones, enabling the networks to extract relevant features that underscore the differences and similarities between the inputs.

Siamese networks are generally trained using pairs of inputs labeled as similar or dissimilar. This method teaches the network to draw closer the embeddings of similar items while distancing those of dissimilar items. Post-training, the network can assess the similarity between new input pairs based on their embeddings.

The Beach Litter dataset [Shin'ichiro 2022] is essential for environmental research, offering insights into the types and volumes of litter on beaches. It includes detailed categorizations of various litter items, such as plastic bottles and cigarette butts, along with their locations and the times they were collected. This data is decisive for analyzing pollution patterns and advocating for environmental policies.

This dataset not only aids in environmental monitoring and cleanup efforts but also serves educational purposes by increasing awareness of pollution impacts and the importance of proper waste management.

The following discussion will include how the Siamese Neural Network can be applied to the Beach Litter Dataset, as referenced in the cited code [Cartolano 2024].

Chapter 2

Implementation

This project uses a Siamese Neural Network to classify beach litter from images taken from the dataset [Shin'ichiro 2022], aiming to benefit environmental studies and pollution monitoring by making the identification of litter items automated. Such networks are particularly effective for tasks with limited data points per class, utilizing the similarities or differences between pairs of inputs.

2.1 Environment Setup and Library Installation

Essential Python libraries such as `TensorFlow`, `matplotlib`, and `numpy` are installed to facilitate data manipulation and model building.

Google Drive is mounted to access datasets stored there, likely containing the beach litter images necessary for training.

2.2 Functions

`load_image(image_path)`: Loads images from a specified directory, processes them by resizing and normalizing, and then converts them into an array format suitable for processing. Each image is converted to RGB color space, resized to 105x105 pixels, and normalized by dividing pixel values by 255.0, which scales the pixel values to a range of [0,1] for model training.

`create_pairs(images, num_pairs=1000, positive_ratio=0.5)`: Generates pairs of images for training the model. The pairs include both similar (positive) and different (negative) images based on the specified ratio.

`initialize_base_network(input_shape)`: This function defines the architecture of the base network which processes the input data. The network uses several convolutional layers (`Conv2D`), followed by batch normalization (`BatchNormalization`), pooling (`MaxPooling2D`), and dropout (`Dropout`) for regularization. This sequence of layers extracts features from the input images, and the resulting feature vectors are then used to measure similarity.

`euclidean_distance(vects)`: This function computes the Euclidean distance between the feature vectors of two images. The distance is a direct measure of similarity, where a smaller distance indicates higher similarity.

`create_siamese_model(optimizer='adam', dropout_rate=0.5)`: This function builds the Siamese Neural Network using the defined base network. It takes a

pair of input images (`input_a` and `input_b`), processes them through the same `base_network` to maintain weight sharing, and computes the similarity using the `euclidean_distance`. The output is then the Euclidean distance between the processed inputs. The network is then compiled with the Adam optimizer and uses `contrastive_loss` which is suitable for learning fine-grained differences and similarities between the inputs.

`contrastive_loss(y_true, y_pred)`: This function calculates the contrastive loss which is designed to help the model learn from pairs of examples. The function computes the loss based on the Euclidean distance between the outputs from the two inputs of the network:

- Positive pairs (similar): For pairs labeled as similar (`y_true=1`), the loss is the square of the Euclidean distance (`square_pred`), encouraging the model to drive the distance towards zero.
- Negative pairs (dissimilar): For pairs labeled as dissimilar (`y_true=0`), the loss is the square of the margin minus the predicted distance, but only when the predicted distance is less than the margin (`margin_square`). This pushes the distance to be at least the margin, penalizing the model if the distance is less.

2.3 Data Preparation

Images are loaded and paired by using the functions that have been already described.

2.3.1 Data Exploration and Visualization

Follows some data exploration and visualization within the context Beach Litter Dataset [Shin'ichiro 2022].

First, one has created pairs for visualization by generating ten pairs of images. This allows to visually verify the kind of pairs the function is creating, which is crucial for understanding how well your pairing logic might perform in training scenarios.

Visualize by displaying the image pairs as in Fig.(2.1). Visualization is a critical step in understanding the data and ensuring that the pairing logic (similar vs. dissimilar) aligns with human judgment. This helps in spotting any anomalies or errors in the data preprocessing or pairing logic early in the project life-cycle.
In Fig.(2.1), the top two pairs are labeled as "Similar," indicating that has been identified a level of similarity between the images in each pair. This is based on factors like the presence of litter, beach terrain, or overall image composition.
In the same figure the subsequent three pairs are labeled as "Dissimilar," suggesting that the algorithm has detected significant differences between the images in each pair.

Plotting class distribution is important for ensuring that there is balance between the classes in the sample data. An unbalanced distribution could lead to a model that is biased towards the majority class, potentially degrading performance when faced with a balanced real-world dataset.
In Fig.(2.2) the bar chart indicates an equal number of similar and dissimilar pairs. This suggests a balanced dataset.

Figure 2.1: Visualize the Pairs

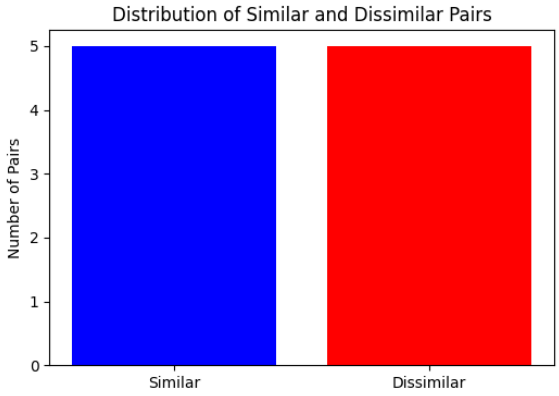

Figure 2.2: Visualize the Class Distribution

2.3.2 Set Up for Model Training

Data is split into training and testing sets. Pairs for both training and testing are created using the `create_pairs` function. The split is done by using the "2/3 for training-1/3 for testing" rule.

2.4 The Model

Follows a discussion on the model's structure, evaluation and performance.

2.4.1 Build the Model

Define a Siamese Neural Network by using the function `create_siamese_model`. The architecture consists of two identical subnetworks that share the same weights and parameters, indicating they are mirror images of each other. This function uses `initialize_base_network` to initialize each subnetwork with several convolutional layers followed by max pooling, dropout for regularization, and batch normalization layers to ensure that the network generalizes well to new, unseen data.

2.4.2 Evaluation

The model uses a custom loss function called contrastive loss which computes the loss based on Euclidean distance.

The model is trained on the training dataset and evaluated on the testing dataset.

- Test loss: 0.6373.
  This value indicates how well the model is doing in terms of making predictions on new data, with lower values generally being better. This value is not particularly low.

- Test accuracy: 0.505.
  An accuracy of 50.5% implies that the model is only slightly better than flipping a coin to determine the class, which is not effective for most practical purposes. For a binary classification, you'd expect a well-performing model to have an accuracy significantly higher than this.

These results could indicate several potential issues. The model might be under-fit the data, meaning it is too simple to capture the underlying pattern or the model's parameters may not be optimized. This is all true for the considered code [Cartolano 2024], especially due to RAM constraints on the free version of Google Colab.

2.4.3 Prediction

One makes predictions on the Beach Litter Dataset [Shin'ichiro 2022] by using the trained model.

Model performance is evaluated using accuracy, precision, recall, and F1-score. Remember their meaning:

- Accuracy is the proportion of true results (both true positives and true negatives) among the total number of cases examined. It measures how often the classifier makes the correct prediction.

$$\text{Accuracy} = \frac{\text{True Positives (TP)} + \text{True Negatives (TN)}}{\text{TP} + \text{TN} + \text{False Positives (FP)} + \text{False Negatives (FN)}}$$

- Precision is the ratio of true positives to the sum of true and false positives. Precision is a measure of a classifier's exactness. The higher the precision, the more confident you can be in the classifier when it predicts a positive result.

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

- Recall is the ratio of true positives to the sum of true positives and false negatives. Recall measures a classifier's completeness. The higher the recall, the more it captures all positive instances.

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

- F1-Score is the harmonic mean of precision and recall, taking both false positives and false negatives into account. It is a good way to show that a classifier has a good balance between precision and recall. The F1 score reaches its best value at 1 (perfect precision and recall) and worst at 0.

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

where,

- True Positives (TP): The number of positive instances correctly predicted by the model.
- True Negatives (TN): The number of negative instances correctly predicted by the model.
- False Positives (FP): The number of negative instances incorrectly labeled as positive by the model.
- False Negatives (FN): The number of positive instances incorrectly labeled as negative by the model.

In this case,

- Precision: 0.533. While better than a coin flip, this level of precision is typically considered low for most practical applications, suggesting that the model generates a considerable number of false positives.
- Recall: 0.08. The recall value is very low. This means that out of all the actual positive cases, the model is only correctly identifying 8% of them. This suggests the model is missing a large number of positives (high false negatives), which is especially critical if the cost of missing true positives is high.

- F1 Score: 0.139. It is quite low, indicating that the model is not performing well in terms of both precision and recall. The model is neither precise nor robust in identifying the positive class effectively.

Additionally, one plots learning curves for accuracy and loss, confusion matrices, ROC curves, and Precision-Recall curves to provide comprehensive insights into model performance.
Remember that

- Confusion Matrix is a table that summarizes how well a classification model's predictions match the actual labels. It shows true positives (top-left position), false positives (bottom-left position), true negatives (bottom-right position), and false negatives (top-right position), providing clear insight into the performance of the model, especially with regard to its errors.

- ROC Curve is a graphical plot that shows the performance of a binary classifier by plotting the true positive rate against the false positive rate at various threshold levels. The area under this curve (AUC) indicates how well the classifier can distinguish between classes.

- Precision-Recall Curve illustrates the trade-off between precision (the accuracy of positive predictions) and recall (the ability to find all positive instances) across different thresholds. It's especially useful when dealing with imbalanced datasets.

Let us discuss the following graphs.

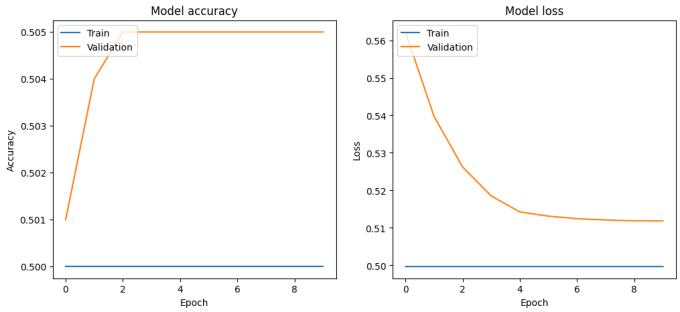

Figure 2.3: Plot of Model Accuracy-Model Loss

Fig.(2.3) shows model accuracy and loss graphs. In the accuracy graph the model's accuracy over epochs shows that both training and validation accuracies are very close, hovering around 50.5%. This indicates that the model isn't learning effectively, as the accuracy is close to what you would expect from random guessing. Loss graph shows the loss for both training and validation decreases sharply after the first epoch and then decreases, which is typical, but the loss values are still relatively high, suggesting that the model could be improved.

Fig.(2.4) shows a large number of false negatives (460) and false positives (35), with relatively few true positives (40) and a large number of true negatives (465). This indicates the model is biased towards predicting the negative class and is often incorrect when the actual class is positive.

Fig.(2.5) shows ROC Curve. It is almost a diagonal line with an area under the curve (AUC) of approximately 0.51. This is just slightly better than a completely

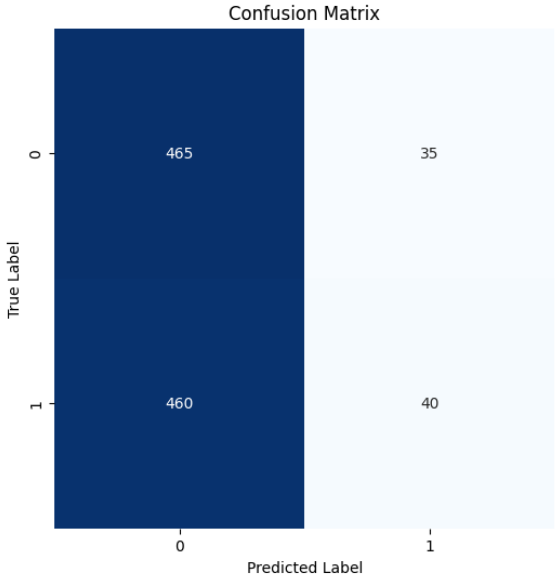

Figure 2.4: Confusion Matrix

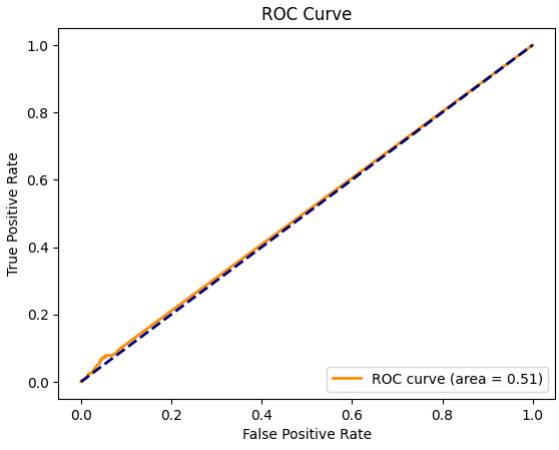

Figure 2.5: ROC Curve

random model (which would have an AUC of 0.5). It indicates that the model has no discriminatory power to distinguish between the positive and negative classes.

Fig.(2.6) is the Precision-Recall Curve which is irregular, showing an unstable model when it comes to the trade-off between precision and recall. Typically, one wants the curve to be as close to the top-right corner as possible, indicating high precision and high recall.

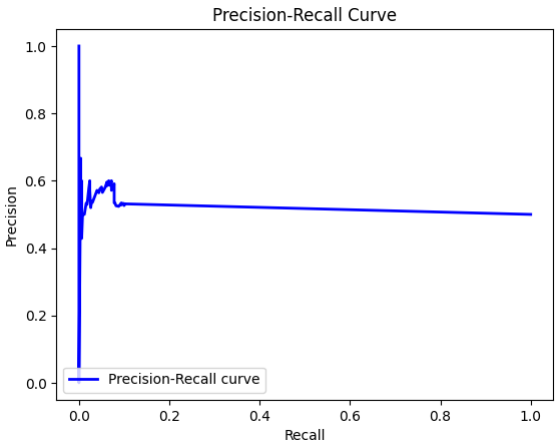

Figure 2.6: Precision-Recall Curve

In summary, these images suggest that the model is performing poorly and is not much better than random at classifying the instances correctly. The model struggles particularly with identifying the positive class, leading to a high number of false negatives. There might be issues with the data quality, feature selection, model architecture, or the learning process that need to be addressed to improve performance.

Chapter 3

Conclusions

This report has detailed the application of a Siamese Neural Network (SNN) for the classification of beach litter from images within the Beach Litter Dataset. The methodology incorporates environment setup, data preprocessing, and a thorough explanation of the functions developed for image pairing and model construction.

Throughout the report, it is evident that the innovative architecture of SNNs holds great potential for distinguishing between various inputs, making them suitable for tasks such as identifying similarities or differences within image data. However, the performance metrics such as precision, recall, and F1 score, along with the generated ROC and Precision-Recall curves, suggest that the current model is performing only marginally better than random chance.

In particular the precision of 0.533 indicates a moderate ability to predict true positives, but the recall of 0.08 and an F1 score of 0.139 reveal that the model struggled to identify most of the positive instances. These outcomes highlight the challenges faced in model training, perhaps due to the inherent complexity of the dataset, limitations within the model architecture or data processing steps. Moreover, the confusion matrix revealed a substantial imbalance in the model's ability to classify the positive class, with an inclination towards predicting the negative class. Such a bias may arise from an inadequate feature extraction process that fails to capture the defining characteristics of the litter within the images.

The model's tendency to over-fit the negative class could potentially be mitigated through more advanced regularization techniques, a richer feature set, or by employing different neural network architectures that may better capture the nuances of the dataset.

Future improvements could include expanding the dataset (using data augmentation techniques), experimenting with different network architectures, or employing newer methods of contrastive learning.

To achieving a more accurate and reliable model could make substantial contributions to environmental conservation efforts, informing policy and guiding beach cleanup initiatives with data-driven precision.

Bibliography

Shin'ichiro, Sugiyama Daisuke Hidaka Mitsuko Matsuoka Daisuke Murakami Koshiro Kako (2022). *Beach Litter Dataset v2022*. Digital Database. URL: https://doi.org/10.17882/85472.

Cartolano, Ludovica (2024). *SiameseNN_{B}eachLitter.py*. Python code for CCEN Implementation task.