Deep Learning Tools for image classification in Cryo-electron microscopy

Author: Joshua Lorenzana Santuyo, jsantulo7@alumnes.ub.edu
Facultat de Física, Universitat de Barcelona, Diagonal 645, 08028 Barcelona, Spain.

Advisor: Dr. David Maluenda

Abstract: Cryo-electron microscopy is an imaging technique used for 3D reconstruction of biomolecules, enabling researchers to study their structures. However, due to low signal-to-noise ratios in captured images, 2D classification is a critical preprocessing step. This thesis explores the application of a deep learning approach, specifically a similarity network, to address this challenge. A Siamese model, trained with a Triplet Loss function, is used to differentiate between similar and dissimilar images. The model was trained on a dataset with known ground truth and tested on two types of unseen data: a similar dataset with ground truth and a different dataset without the ground truth. This study demonstrates the potential of deep learning to complement traditional 2D classification methods in cryo-EM.
Keywords: Deep Learning, Machine Learning, Similarity Neural Network, Siamese Model, Triplet Loss
SDGs: Industry, innovation, infrastructures. Earthly life.

I. INTRODUCTION

Since its discovery, cryo-electron microscopy (cryo-EM) has become an essential tool across numerous scientific disciplines [1]. It allows researchers to visualize molecular structures, particularly biomolecules, at atomic resolution [2]. Unlike traditional electron microscopy, cryo-EM uses low-energy electrons and preserves samples by freezing them, protecting delicate structures from damage. Although the technique offers high-resolution imaging, it yields a low signal-to-noise ratio, necessitating a critical step known as 2D classification [3]. By grouping and averaging similarly oriented samples, this process enhances the signal and minimizes noise in the combined images, ultimately clarifying their molecular structures.

Joachim Frank, a Nobel laureate, approached the 2D classification problem through classical statistical and computational techniques which is slow and computational heavy [2]. However, recent advances in deep learning present promising alternatives to these traditional methods [4]. Convolutional Neural Networks (CNNs) have shown remarkable success in tasks like object detection and recognition but are often designed for classification with explicit labels rather than similarity-based grouping.

This thesis explores the potential of similarity networks, specifically Siamese Networks trained with triplet loss [5] to tackle the unique challenges of cryo-EM 2D classification. Commonly used in fields like facial recognition and fingerprint analysis, these networks excel at learning similarity relationships. By evaluating their performance on noisy and clear cryo-EM datasets, this study aims to establish an embedding-based similarity framework that could complement or even replace conventional approaches.

The code for this thesis can be found in my repository [6].

II. SIAMESE MODEL

Standard CNNs address computer vision tasks by classifying objects into predefined categories. However, as mentioned earlier, this project adopts a different approach by employing similarity-based neural networks.

The idea is to train the Siamese Network using the Triplet Loss Function to generate feature vectors, known as embeddings. These embeddings are produced by passing input images through three identical neural networks with shared weights as depicted in **Figure 1**. The three input images are:

1. **Anchor**: The reference image.
2. **Positive**: An image from the same class as the anchor.
3. **Negative**: An image from a different class from the anchor.

When projected into the embedding space, similar images are clustered closely together, while dissimilar images are separated.

The basic architecture of the Siamese Network used in this project is illustrated in Figure 1. Its principal components are as follows:

- **Embedding Generator**: It consists of a convolutional base using the ResNet50 backbone pretrained on the ImageNet database which weights are frozen and excludes the top layer. Then, it is followed by a custom and trainable top layer consisting of 3 dense layers to finally obtain an embedding vector of 256 dimensions (default settings).
- **Distance Layer**: A custom layer that computes the Euclidean distance between the embeddings of anchor-positive and anchor-negative pairs.

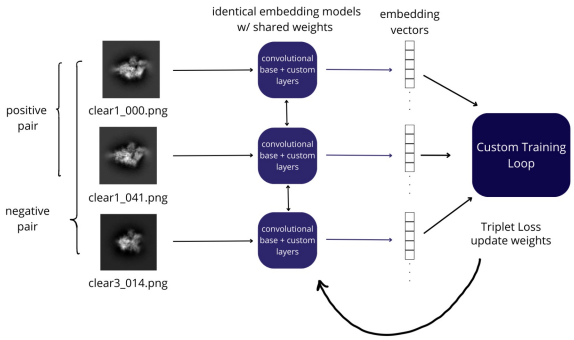


FIG. 1: A basic representation of the Siamese Model used in this project: The first image is the anchor (reference), the second is the positive (from the same class as the anchor), and the third is the negative (from a different class).

- **Triplet Loss Function**: Used to optimize the embedding space, using the distance layer it ensures that anchor-positive pairs are closer than anchor-negative pairs by a margin.

$$L(A, P, N) = max(||f(A) - f(P)||^2 - ||f(A) - f(N)||^2 + m, 0), \quad (1)$$

where A represents the anchor image, P the positive image, N the negative image, and m the margin, which ensures that there is sufficient distance between similar and dissimilar images. The $f(\cdot)$ represents the embedding function that generates embedding vectors of dimensions 256 from input images of size 128 x 128 pixels.

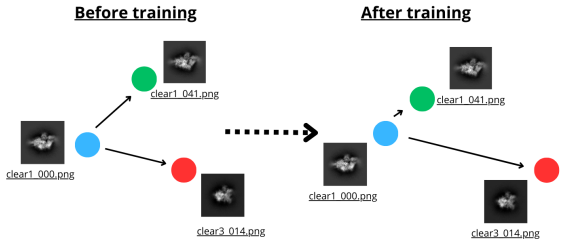


FIG. 2: The loss function optimizes this process by bringing positive pairs closer while pushing negative pairs further apart. Notice how the positive pair (blue and green) are brought closer together after training while the negative pair (blue and red) are brought away from each other. This figure is illustrative and does not represent the real points.

The Triplet Loss function plays a crucial role in optimizing the embedding space. It works by comparing an anchor image with a positive (similar) and a negative (dissimilar) sample, ensuring that the anchor-positive distance is minimized and the anchor-negative distance is maximized, as illustrated in **Figure 2**.

This kind of network is scalable as it does not require retraining to differentiate similar from dissimilar images in unseen data. However, a model trained on one dataset may not form distinct clusters when applied to an unseen dataset. This uncertainty is one of the key aspects to be explored later.

The entire Siamese model was implemented using Keras and TensorFlow API [7] and trained using a custom training loop based on the documentation provided by Keras official website [8].

III. METHODOLOGY

A. Experimental Setup

The experiments were conducted on a personal laptop with a GPU NVIDIA RTX 3050 running on Windows 11. To enable GPU, the system was configured with Windows Subsystem for Linux (WSL), running Ubuntu 24.04.1 LTS. The experiments were implemented in Python 3.12.3, with libraries including Keras, TensorFlow-GPU, Scikit-learn, NumPy, and Matplotlib.

B. Dataset preparation for training.

The dataset used consists of images of the COVID-19 virus's spike protein from [9], formatted as .png files. The use of computer-generated images enables us to acquire data in two conditions: a) free of noise (clear dataset) and b) with added Gaussian noise (noisy dataset) to simulate real low signal-to-noise ratio images. A traditional classifier is used to further divide each dataset into four distinct classes to establish a ground truth. Images within the same class are considered to be similar, forming a positive pair, while those from different classes are dissimilar, forming a negative pair.

Key pre-processing steps included:

1. **Dataset Splitting**: The samples were divided into training (80%) and validation (20%) subsets.
2. **Triplet Preparation**: Triplets consisting of (anchor, positive, negative) images are generated. Additionally, the class labels of the anchor image are also included, but solely for evaluation purposes. However, only the triplet is used for training.

C. Training parameters

The default parameters of the training are the following, most of which are chosen to fit the GPU's memory:

1. **Batch Size**: 8 triplets per batch.
2. **Early Stopping**: Training was monitored using validation loss with a patience of 5.
3. **Optimizer**: Adam optimizer with a learning rate of 0.0001.

4. **Margin**: In the training loop, 0.5 is defined for the margin.
5. **Custom Top Layer for the convolutional base**: The innermost layers produce outputs of 512, 256, and finally a 256-dimensional embedding vector.

D. Model evaluation analysis

After training the model, two evaluation methods were employed to ensure the model has effectively learned: (a) confusion matrix and (b) Dimensionality-reduction visualization.

The confusion matrix evaluates the model by calculating the cosine similarity between embeddings between batches grouped by the ground truth classes. The cosine similarity of embeddings from the same class should approach or equal 1, while the cosine similarity of embeddings from different classes should be significantly less than 1. However, due to the high dimensionality of the vectors, specifically 256 dimensions, the cosine similarity for negative pairs can approach values near 1.

$$cosine\ similarity = \frac{\overrightarrow{f(X)} \cdot \overrightarrow{f(Y)}}{||\overrightarrow{f(X)}||||\overrightarrow{f(Y)}||} \tag{2}$$

Equation (2) represents the cosine similarity, the cross product between two vectors.

This matrix helps identify whether the embeddings display diagonal dominance and symmetry, which are key indicators of effective model training as presented in **Figure 3** .

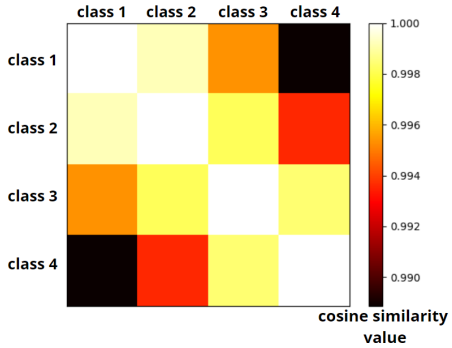


FIG. 3: An example of the confusion matrix using the cosine similarity to compare different classes by batches using the clear dataset. It is symmetric and shows a diagonal dominance. Note that it differs by little due to the embedding vectors' high dimensionality.

Given the high dimensionality of the embedding vectors, visualizing them directly is challenging. To address this, a dimensionality reduction is performed. In this case, the dimensionality reduction is carried out in two steps using techniques provided by SciKit-learn library.

The first step involves reducing the original embedding vector to 50 dimensions using Principal Component Analysis (PCA) which is frequently applied for data clusterization.

The next step involves further reducing the embedding vectors to two dimensions using t-distributed stochastic neighbor embedding or t-SNE. This technique is specifically chosen for its suitability in visualizing high-dimensional data.

In a well-trained model, embeddings of the same class should form distinct clusters as seen in **Figure 4 (a)**.

E. Hyper-parameter tuning

Tuning a model's parameters is a standard practice to ensure effective training, particularly for Siamese models, which are susceptible to over-fitting. To address this challenge, the following prioritized strategies were employed:

1. **Optimizer value**: Varying the optimizer value from 10^{-3} to 10^{-4}.
2. **Margin value**: Using the default 0.5 and changing it to 0.75 and 1.0.
3. **Lowering the number of parameters**: Changing the output Dense layers of the embedding generator from $512 - 256 - 256$ to $256 - 128 - 128$.
4. **Data Augmentation** Generating additional data by introducing Gaussian noise into the training set.

F. Model evaluation

After training a model and selecting the best-performing model, it is used to calculate the embeddings for the two types of datasets previously mentioned.

Evaluating the model on the similar dataset allows us to assess whether the Siamese model is capable of learning this type of data. This is determined by applying the k-means algorithm to our calculated embeddings using the trained model and verify whether they coincide with the samples knowing its ground truth. On the other hand, testing it on unseen data can help us explore its potential to cluster or group data meaningfully.

To determine the optimal number of k clusters to use, a visual guide called the elbow plot is used. The elbow plot represents the inertia against the number of clusters. The inertia is the sum of squared distances of each data point to its closest cluster center meaning the lower the inertia the better clusterization. However, this approach is subjective due to it being a visual interpretation. The optimal k is selected where the change in inertia becomes negligible compared to the previous changes. For the similar dataset, it is expected for $k \approx 4$, reflecting its four classes. For the dissimilar dataset, the absence of ground truth makes k more ambiguous.

IV. RESULTS AND DISCUSSION

A. Validation Dataset

For the the following prediction and evaluation, "training_008" is chosen for achieving the highest resolution in cosine similarity. Further information on its parameters can be found in the Supplementary Material.

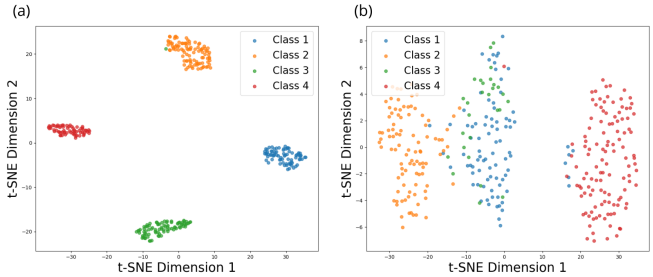


FIG. 4: (a) Dimension reduction done with the validation dataset of the clear images. (b) Dimension reduction done with the validation dataset of the noisy images.

Although the clear dataset did not require extensive hyperparameter tuning, as the model learned with the default settings, hyperparameter tuning was performed nonetheless.

The visualization of the embeddings in **Figure 4 (a)** after the dimension reduction confirms the presence of four distinct classes, aligning with the real dataset used in this study.

The elbow plot in **Figure 5 (a)** supports the use of four clusters, consistent with the dataset's structure.

On the other hand, the embedding space of the noisy dataset represented in **Figure 4 (b)** reveals some confusion between certain classes, particularly classes 1 and 3. This highlights the challenging nature of the dataset,.

For this reason, hyper-parameter tuning was required. Unfortunately, none of the fine-tuning made the model learn to distinguish said classes.

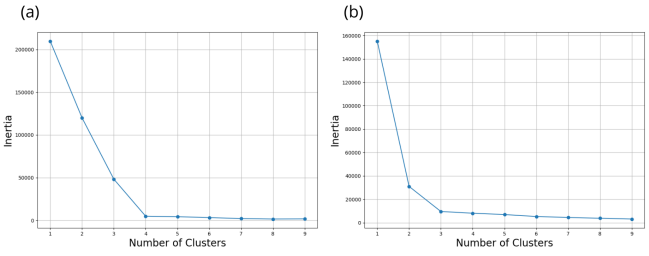


FIG. 5: (a) Elbow plot for the clear dataset (b) Elbow plot for the noisy dataset.

The elbow plot in **Figure 5 (b)** suggests that there are at least 3-5 clusters.

Finally, to conclude our findings, the selected model is used to calculate the embeddings which are then passed through the k-means algorithm to cluster them into $k = 4$ clusters to coincide with the ground truth. Then, we compare random average of different classes with the average of the samples with ground truth and the average of the samples belonging to the same group cluster found using k-means.

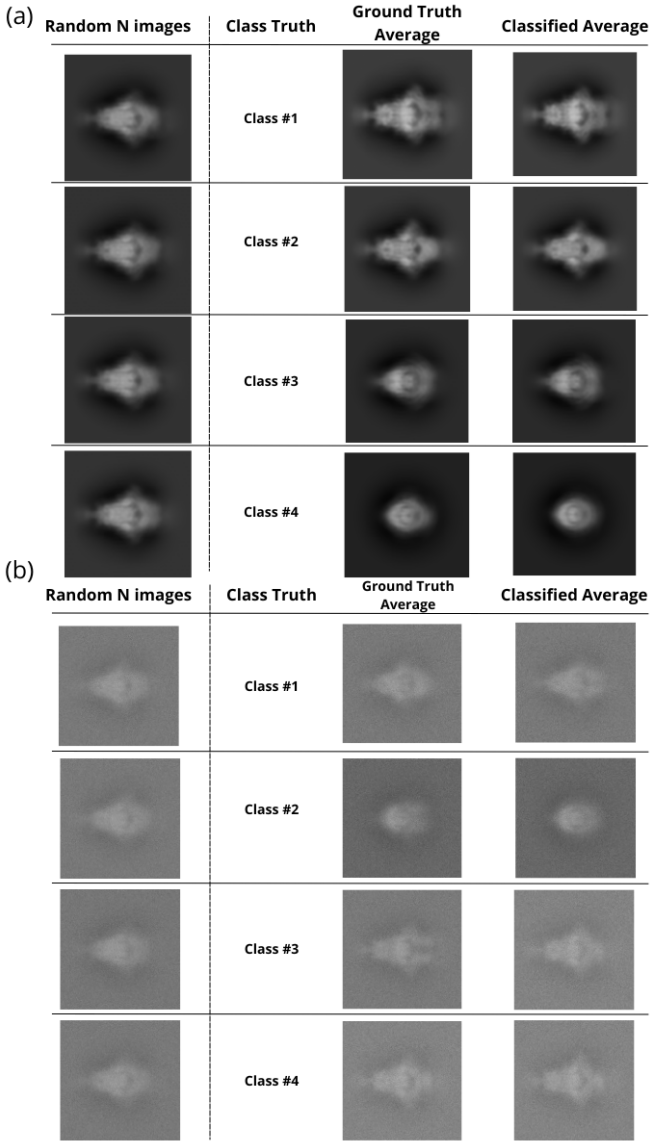


FIG. 6: The first column represents the average of N samples from the whole clear (a) and noisy (b) dataset. The second column indicates the class to which the next two columns belong to. The third represents the ground truth average of N samples of each class while the last column represents the average for N samples from each class by applying kmeans algorithm to the embeddings calculcated using the selected model, the result of this work. Note that model trained on clear images has definitely learned while for the noisy dataset, classes 1 and 3 are more ambiguous.

B. New dataset application

Using the same model, the embeddings calculation and visualization is also performed to an unseen and different type of dataset corresponding to the S. cerevisiae dimer [10]. Compared to the COVID-19 virus, this is a completely different molecule but having a similar shape.

The embeddings space of the clear dataset in **Figure 7 (a)** show that the embeddings are continuous. Nevertheless, k-means was performed, with the optimal k cluster selected using the elbow plot. However in **Figure 7 (b)** it reveals a disorganized clusterization.

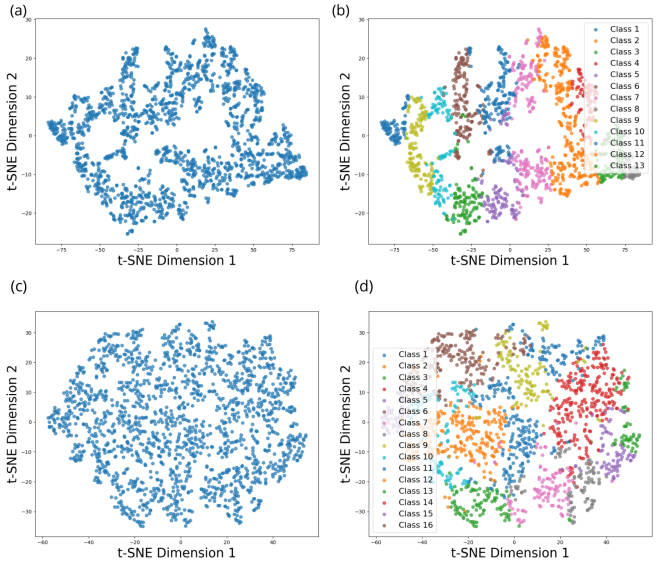


FIG. 7: (a) Embeddings of the clear dataset visualized without labels. (b) Embeddings of the clear dataset visualized with labels. (c) Embeddings of the noisy dataset visualized without labels. (d) Embeddings of the noisy dataset visualized with labels.

The same conclusions can be said for the noisy dataset in **Figure 7 (c)** and **Figure 7 (d)**.

V. CONCLUSIONS

For the similar dataset, the model trained on the clear dataset learned better than the one trained on the noisy dataset. Notably, the noisy dataset struggled to distinguish between classes 1 and 3.

While similarity networks do not necessarily form distinct clusters on unseen data, the model is nevertheless applied for its embeddings visualization. As anticipated, it failed to distinguish between classes based on their embeddings.

Despite these challenges, the Siamese Network is a viable alternative for the 2D classification when trained on data with ground truth. However, further training on a bigger dataset is required to determine whether the model can ultimately succeed in this task.

On the contrary, the lack of clear clustering might suggest additional investigation is needed to verify whether a Siamese model trained on a one type of data can reliably to differentiate similar and dissimilar images of another type of data.

Acknowledgments

I would like to express my heartfelt gratitude to my advisor, Dr. David Maluenda, for granting me the opportunity to explore this topic and for his ceaseless guidance throughout the project. I am also deeply thankful to my parents and friends for their constant encouragement during this journey.

[1] Joachim Frank, *Generalized single-particle cryo-EM – a historical perspective, Microscopy, Volume 65, Issue 1, Pages 3–8* (February 2016).

[2] Joachim Frank and Christian M T Spahn *Rep. Prog. Phys. 69 1383, DOI 10.1088/0034-4885/69/5/R03*, (2006).

[3] C. Sorzano, et al. *Structural Proteomics, High-Throughput Methods: Image Processing in Cryo-Electron Microscopy of Single Particles: The Power of Combining Methods*, 3rd. ed. (2021).

[4] Sanchez-Garcia, R., Segura, J., Maluenda, D., Carazo, J. M. and Sorzano, IUCrJ, 5, 854-865. *Deep Consensus, a deep learning-based approach for particle pruning in cryo-electron microscopy*, C. O. S. (2018).

[5] S. Das, *Image similarity using Triplet Loss*, (2019).

[6] J. Lorenzana, *Deep Learning Cryo, https://github.com/AmFruitY/DeepLearningCryo.*

[7] F. Chollet, *Deep Learning with Python*, 2nd edition (2021).

[8] H. Essam and S. Valdarrama, *Image similarity estimation using a Siamese Network with a triplet loss, https://keras.io/examples/vision/siamese_network/*, (2021).

[9] M Berlinguer, et. al., *SARS-CoV-2 Spike protein in complex with the single chain fragment scFv76-77, https://www.ebi.ac.uk/emdb/EMD-50417* (2024).

[10] N Noskova, et. al. *Cryo-EM structure of S. cerevisiae Rai1-Rat1 dimer. https://www.ebi.ac.uk/emdb/EMD-18199* (2024).

Deep Learning tools for Cryo-electron microscopy

Author: Joshua Lorenzana Santuyo, jsantulo7@alumnes.ub.edu
Facultat de Física, Universitat de Barcelona, Diagonal 645, 08028 Barcelona, Spain.

Advisor: Dr. David Maluenda

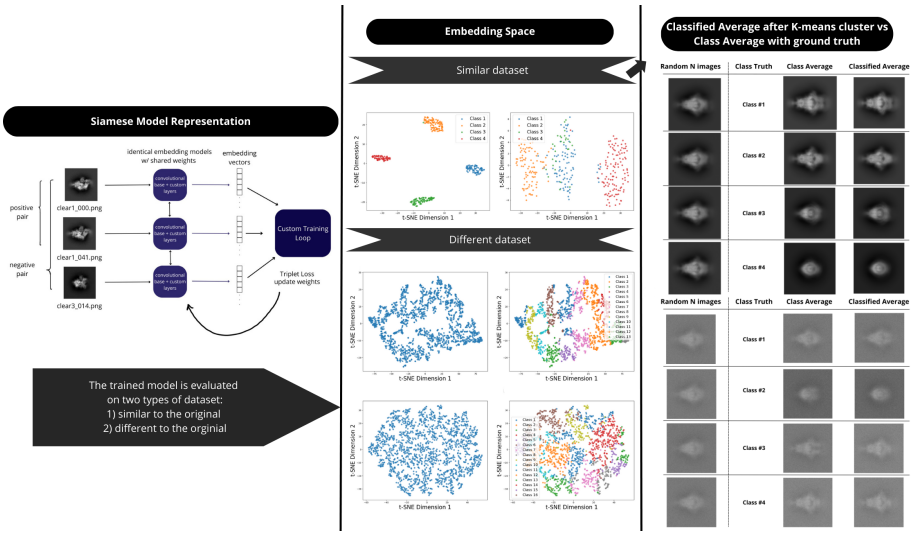
Resum: La microscòpia crioelectrònica és una tècnica utilitzada per a la reconstrucció 3D de biomolècules, la qual permet l'estudi detallat de les seves estructures. A causa de la baixa relació senyal-soroll de les imatges capturades, és imprescindible una classificació 2D d'aquestes. Aquest treball examina l'aplicació d'un model de *Deep Learning*, concretament una xarxa de similitud, per afrontar aquest repte. Un model siamès utilitza una funció de pèrdua aplicada a triplets per distingir entre imatges similars i diferents. És entrenat amb una base de dades que inclou el ground truth i es prova amb dos tipus de dades: una base de dades similar amb el seu *ground truth* i una altra d'una naturalesa diferent sense el *ground truth*. Aquest estudi posa de manifest la possibilitat d'utilitzar eines de *Deep Learning* com a complement o fins i tot com una alternativa als mètodes tradicionals de classificació 2D en microscòpia crioelectrònica.
Paraules clau: Deep Learning, Machine Learning, Xarxa Neuronal siamès, model siamès, Triplet Loss.
ODSs: Indústria, innovació, infraestructures. Vida terrestre.

Objectius de Desenvolupament Sostenible (ODSs o SDGs)

1. Fi de la es desigualtats		10. Reducció de les desigualtats	
2. Fam zero		11. Ciutats i comunitats sostenibles	
3. Salut i benestar		12. Consum i producció responsables	
4. Educació de qualitat		13. Acció climàtica	
5. Igualtat de gènere		14. Vida submarina	
6. Aigua neta i sanejament		15. Vida terrestre	X
7. Energia neta i sostenible		16. Pau, justícia i institucions sòlides	
8. Treball digne i creixement econòmic		17. Aliança pels objectius	
9. Indústria, innovació, infraestructures	X		

El contingut d'aquest TFG es relaciona amb l'ODS 9, fent èmfasi sobretot amb la fita 9.3 ja que investiga una alternativa a la solució actual de classificació 2D de la microscopia crioelectrònica. El TFG proposa una tècnica computacional més eficient i per tant, energèticament més sostenible. A més a més, també es podria relacionar amb l'ODS 15, concretament aquelles fites relacionades amb la protecció d'animals com la fita 15.7. La microscopia crioelèctronica s'utilitza molt en la fabricació de medicaments i el contingut d'aquest TFG dona una proposta de solució ràpida que podria contribuir a evitar la necesitat de fer assajos amb animals.

GRAPHICAL ABSTRACT



SUPPLEMENTARY MATERIAL

Training Number	Dataset	Optimizer	Optimizer Value	Margin	Dense1	Dense2	Dense3	EarlyStopping	Patience	Distance	Data Augmentation?
training_000_clear	Clear	Adam	0.0001	0.5	512	256	256	Val_loss	5	Euclidean	FALSE
training_000_noisy	Noisy	Adam	0.0001	0.5	512	256	256	Val_loss	5	Euclidean	FALSE
training_001_clear	Clear	Adam	0.00001	0.5	512	256	256	Val_loss	5	Euclidean	FALSE
training_001_noisy	Noisy	Adam	0.00001	0.5	512	256	256	Val_loss	5	Euclidean	FALSE
training_002_clear	Clear	Adam	0.00001	0.75	512	256	256	Val_loss	5	Euclidean	FALSE
training_002_noisy	Noisy	Adam	0.00001	0.75	512	256	256	Val_loss	5	Euclidean	FALSE
training_003_clear	Clear	Adam	0.00001	0.75	512	256	256	Val_loss	5	Euclidean	TRUE
training_003_noisy	Noisy	Adam	0.00001	0.75	512	256	256	Val_loss	5	Euclidean	TRUE
training_004_clear	Clear	Adam	0.00001	1	512	256	256	Val_loss	5	Euclidean	TRUE
training_004_noisy	Noisy	Adam	0.00001	1	512	256	256	Val_loss	5	Euclidean	TRUE
training_005_clear	Clear	Adam	0.0001	0.5	256	128	128	Val_loss	5	Euclidean	FALSE
training_005_noisy	Noisy	Adam	0.0001	0.5	256	128	128	Val_loss	5	Euclidean	FALSE
training_006_clear	Clear	Adam	0.00001	0.5	256	128	128	Val_loss	5	Euclidean	FALSE
training_006_noisy	Noisy	Adam	0.00001	0.5	256	128	128	Val_loss	5	Euclidean	FALSE
training_007_clear	Clear	Adam	0.00001	0.75	256	128	128	Val_loss	5	Euclidean	FALSE
training_007_noisy	Noisy	Adam	0.00001	0.75	256	128	128	Val_loss	5	Euclidean	FALSE
training_008_clear	Clear	Adam	0.00001	0.75	256	128	128	Val_loss	5	Euclidean	TRUE
training_008_noisy	Noisy	Adam	0.00001	0.75	256	128	128	Val_loss	5	Euclidean	TRUE
training_009_clear	Clear	Adam	0.00001	1	256	128	128	Val_loss	5	Euclidean	TRUE
training_009_noisy	Noisy	Adam	0.00001	1	256	128	128	Val_loss	5	Euclidean	TRUE

FIG. 8: Training parameters modified to see with which parameters, the model learned best.