Hybrid CNN-GNN Models in Active Sonar Imagery: an Experimental Evaluation

Gabriel Arruda Evangelista[1], **João Baptista de Oliveira e Souza Filho**[1]

[1] Programa de Engenharia Elétrica / COPPE
Universidade Federal do Rio de Janeiro (UFRJ), Rio de Janeiro – RJ – Brazil

{gabrielevangelista7, jbfilho}@poli.ufrj.br

Abstract. *The development of sonar technologies, such as Multibeam Forward Looking Sonar (MFLS), has enabled detailed underwater imaging, which can be applied for tasks like identifying mine-like objects. However, obtaining large datasets to train image recognition models remains challenging, leading to the need for smaller yet equally accurate alternative models. Previous research proposed a hybrid model that combines Convolutional Neural Networks with Graph Neural Networks for MFLS image classification. This study refines the feature extractor of this model using Knowledge Distillation (KD) and evaluates the cost-effectiveness of this pipeline compared to alternative solutions. The proposed method achieved an error rate of 6.42%, a value comparable to that of other solutions but with less computational effort.*

1. Introduction

The development of sonar technologies, particularly Side Scan Sonar and Multibeam Forward-Looking Sonar (MFLS), has greatly advanced the detailed imaging and mapping of underwater environments. The widespread use of this technology for remote sensing of water bodies and their beds has facilitated various applications, including bathymetric surveys, submarine navigation systems, inspections, and the search for objects [Dos Santos et al. 2022] such as mine-like threats, which may pose risks to navigation [Sinai et al. 2016].

Leveraged by advances in computer vision (CV) techniques, a range of emerging applications explores the automatic processing of sonar-generated images [Steiniger et al. 2022], including the classification of underwater objects [Huo et al. 2020], semantic segmentation of the seabed [Yang et al. 2022], and applied object detection [Yu et al. 2021]. However, training these models can be challenging due to restrictions on access as well as the scarcity of open datasets, which are often expensive and complex to collect. Some available options include [Singh and Valdenegro-Toro 2021], which focuses on the semantic segmentation of sonar images, and [Xie et al. 2022], which targets object detection.

Convolutional Neural Networks (CNNs) are the most common approach for addressing CV problems [Steiniger et al. 2022]. However, graph-based model alternatives have gained attention in recent literature [Vasudevan et al. 2023, Avelar et al. 2020, Dwivedi et al. 2023, Knyazev et al. 2019]. In [Evangelista and Souza Filho 2023], a hybrid model is proposed where a CNN first extracts rich and complex features from image segments. These features are then used to build a Region Adjacency Graph (RAG).

A Graph Neural Network (GNN) subsequently enhances the features associated with each image region, which are finally aggregated for image classification.

The present work experimentally evaluates this hybrid architecture more comprehensively by focusing on the cost-effectiveness of the derived solutions in terms of both performance and computational effort. To this end, a simpler sonar image classification pipeline is discussed as an alternative processing method, aiming to identify the most suitable application domain for each approach using the MFLS sonar image dataset [Xie et al. 2022].

The paper is structured as follows. Section 2 briefly describes the general graph-based framework for image classification with GNNs adopted in the literature. Section 3 revisits the hybrid CNN-GCN model, highlighting its similarities and differences to the literature as well as the contributions of this work. Section 4 reports the experimental results. Finally, Section 5 presents the conclusions and possible future works.

2. GNN-based literature approaches for image classification

A typical pipeline of graph-based approaches in image recognition applications [Defferrard et al. 2016, Monti et al. 2017, Fey et al. 2018, Knyazev et al. 2019, Avelar et al. 2020, Dwivedi et al. 2023] is as follows: Initially, the image is segmented (**segmentation stage**), usually into superpixels, and a graph is constructed (**graph forming stage**), with one node assigned to each image segment. Node features are often defined by inferring general properties of these segments, such as average luminosity, centroid values, and the intensities of each color channel (**segment features**). Graph edges are defined based on a metric that measures the similarity between the nodes (**edges definition**), following a process similar to [Belkin and Niyogi 2001]. Subsequently, node features are enriched (**feature improvement**) using GNNs. Finally, a pooling process (**graph pooling**) is applied to produce a single embedding for the entire graph, which is then used by the final classifier.

3. The hybrid method

Figure 1 illustrates the pipeline proposed in [Evangelista and Souza Filho 2023]and its relationship with the general framework adopted in the literature. The key difference is the use of a partial CNN structure as the backbone for generating an embedding for each image segment through its feature map. This approach is analogous to the region of interest (RoI) projection explored by Fast R-CNN [Girshick 2015]. The remaining processing stages - segmentation, feature improvement, and graph pooling - are similar in both approaches.

Despite the increased computational cost associated with using CNN-based embeddings compared to those in literature, the resulting node features retain more information. This is because CNNs incorporate higher-level problem abstractions that enhance feature representativeness. The subsequent GNN module also enables a more effective fusion of image information contained in non-contiguous image segments. As a result, it is possible to use fewer CNN layers and, consequently, fewer convolutional filters while still obtaining feature vectors that are representative of the image. Therefore, this approach may achieve better balance between performance and computational effort compared to standard methods, given that GNNs typically involve fewer parameters than CNNs.

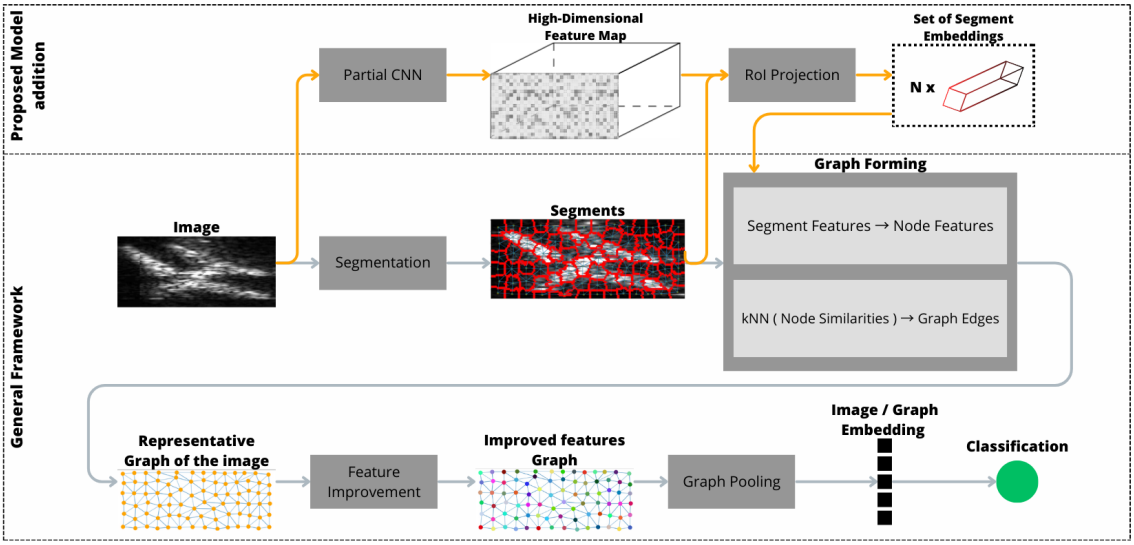


Figure 1. Traditional graph-based approach for image recognition tasks and the proposed extension (see text).

This work differs from [Evangelista and Souza Filho 2023] by evaluating an alternative classification pipeline that solely relies on segment embeddings, thus not modeling the sonar images with graphs. Specifically, the features generated by the CNN-based embedding model and segmented through RoI projection are aggregated into a single representative vector for each image, which is then used for classification, as discussed below. In addition, we propose to fine-tune this embedding module using Knowledge Distillation (KD) [Hinton et al. 2015] and explore alternative approaches for image segmentation and edge formation.

In summary, two approaches were evaluated: the model from [Evangelista and Souza Filho 2023], referred to as Graph-Based Enhanced Features (GBEF), and its simplified version, named Mean Segment Feature Aggregation (MSFA), which excludes the feature aggregation phase based on GNN. Both approaches use the same CNN processing and image segmentation methods to generate embeddings for the image segments. The following subsections discuss these models and the related pipeline steps in greater detail.

3.1. Mean Segment Feature Aggregation (MSFA) Approach

To analyze the relevance of each processing stage in the hybrid method, we considered a simpler processing pipeline that excludes the feature enhancement phase provided by the GNN. In this case, the embedding vector generated for each segment, denoted as $\mathbf{x}_i$ (where $1 \leq i \leq N_s$ and N_s is the number of image segments), can be aggregated using the following schemes to produce a single representative vector $\mathbf{x}_c$ for classification:

- **Mean**: In this case, the image representative vector is defined by the average of all segment embeddings. Thus, $\mathbf{x}_c = \frac{1}{N_s} \sum_{i=1}^{N_s} \mathbf{x}_i$.
- **Flatten**: Here, the embeddings are concatenated by considering segments taken from the image from left to right and top to bottom. Thus, $\mathbf{x}_c^T = \begin{bmatrix} \mathbf{x}_1^T & \mathbf{x}_2^T & \cdots & \mathbf{x}_{N_s}^T \end{bmatrix}$.
- **Flatten+Position**: This alternative is similar to **Flatten**, but in this case, after each segment embedding, its normalized centroid coordinates

(c_x^i, c_y^i) are appended, The resulting vector is represented as: $\mathbf{x}_c^T = \begin{bmatrix}\mathbf{x}_1^T & c_x^1 & c_y^1 & \mathbf{x}_2^T & c_x^2 & c_y^2 & \cdots & \mathbf{x}_{N_s}^T & c_x^{N_s} & c_y^{N_s}\end{bmatrix}$. Here, the normalized centroid coordinates (c_x^i, c_y^i) are derived by normalizing the centroid coordinates by the image width and height, respectively.

3.2. Graph-Based Enhanced Features (GBEF) Approach

The hyperparameters for the GBEF approach include the graph edge formation criterion, the GNN model for feature enhancement, and the graph embedding process, which defines how node features are aggregated into a single graph embedding vector for classification. For simplicity, mean pooling (i.e., averaging the node features) was used as the graph embedding criterion.

Three proximity-based methods were evaluated for edge formation in this work:

- **Positional proximity**: This is a commonly used approach in the literature, where edges connect each node to its k-nearest neighbor nodes, with k typically set to 8. These neighbors are identified based on the distance between the centroids of their corresponding segments.
- **Feature proximity**: This approach is similar to **Positional proximity**, but it uses cosine distance instead of Euclidean distance as the similarity measure between centroids.
- **Feature+Positional proximity**: In this case, the k edges are determined by combining positional and feature proximity, with each type contributing half of the total edges.

GNN processing involves choosing a GNN architecture and related hyperparameters, such as the number of hidden layers and their dimensionality. The following architectures were considered here:

- **GCN**: Graph Convolutional Networks (GCN) [Kipf and Welling 2017].
- **GAT**: Graph Attention Networks (GAT) [Vaswani et al. 2017].
- **GIN**: Graph Isomorphism Networks (GIN) [Xu et al. 2019].

3.3. Generating embeddings for the image segments

Common to MSFA and GBEF, this processing stage requires defining both the segmentation approach and the CNN architecture to produce segment embeddings. For segmentation, we used the SLIC Zero algorithm [Achanta et al. 2012]. The algorithm begins with a target number of segments, initially dividing the image into a regular grid with uniform spatial resolution across both dimensions. Then, it iteratively refines these segments by grouping pixels to balance superpixel compactness with adherence to image edges. In this work, we considered two segmentation approaches:

- **Regular:** In this case, the algorithm runs with zero iterations, resulting in regular segments.
- **Content-based:** The algorithm is set with its default settings (ten iterations), resulting in compact regions that adhere to the salient image features.

Regarding the CNN model, our proposal uses a single feature map produced by a subset of the initial layers of ResNet-18 [He et al. 2016]. This approach requires

setting an architectural cut-point, i.e., selecting a set of consecutive layers, starting from the network inputs, to extract features from the image. It is also necessary to define a fine-tuning strategy for this feature extractor, considering the classification task at hand, especially since both MSFA and GBEF pipelines are not end-to-end differentiable.

Two design alternatives were considered here:

- **Only Pre-Training (PT)**: The backbone network uses its original parameters derived from ImageNet1K training [Russakovsky et al. 2015].
- **Knowledge Distillation (KD)**: In this process, the partial CNN architecture is fine-tuned using the Knowledge Distillation schema proposed in [Hinton et al. 2015]. The teacher model was a complete ResNet-18 architecture re-trained on the target dataset.

4. Results

This section presents the experimental results obtained with the MFLS classification dataset. The core implementation was carried out with PyTorch [Paszke et al. 2019], while the GNN modules were built on PyTorch Geometric [Fey and Lenssen 2019]. For the CNN, we employed a pretrained model from the TIMM library [Wightman 2019]. The SLIC algorithm and partition management were handled using the Scikit Image and Scikit Learn libraries, respectively [Van der Walt et al. 2014, Pedregosa et al. 2011]. Experiments were conducted on a workstation with an Intel i9-13900k processor, 64 GB of RAM, and a Nvidia GPU GeForce RTX 4090 with 24 GB of VRAM.

4.1. Experimental setup

The MSFA and GBEF models were evaluated using stratified 10-fold cross-validation. In this process, each fold defined a different testing set, while the remaining data were randomly split into training and validation sets with the following sample proportions: 10%, 80%, and 10%, respectively. To identify the best hyperparameters for each model while keeping the design space of MSFA and GBEF computationally feasible, we restricted the hyperparameter choices to a small subset and performed a grid search, as summarized in Table 1.

Table 1. Range of hyperparameters parameters considered in the grid-search procedure for both experiments.

Segments' embedding approach		MSFA approach		GBEF approach	
Cut-off point	{1, 5, 6, 7}	Aggregation strategy	{**Mean**, **Flatten**, **Flatten+Position**}	Edge forming	{**position**, **feature**, **feature+position**}
Segment embedding strategy	{**PT**, **KD**}			GNN architecture	{GCN, GAT, GIN}
Segmentation modality	{**Content-Based**, **Regular**}			Number of hidden layers	{2, 4, 8}
				Hidden dimensionality	{8, 16, 32, 64}

4.2. Dataset

The dataset comprises images of submerged objects captured with an MFLS [Xie et al. 2022], focusing on object detection. Each image may contain a varying number of objects from multiple classes. Images were cropped to contain a single object, as described in [Evangelista and Souza Filho 2023]. The resulting dataset comprises 14,649 images, each one associated with one of ten possible classes: cube, ball, cylinder, human body model, tyre, circle cage, square cage, metal bucket, plane model, and ROV. The number of samples per class ranges from 486 to 984, and image dimensions (height or width) vary from 20 to 278 pixels.

4.3. Generation of embeddings for image segments

ResNet-18 [He et al. 2016] was explored for generating the embeddings of each segment. Different cutoff points were tested to evaluate the trade-offs between classification efficiency and computational effort that could be achieved with MSFA and GBEF. The first cutoff point only includes its first convolutional layer, while the cutoff points 5, 6, and 7 progressively add one more basic-block-pair [He et al. 2016] to the network architecture. Table 2 summarizes, for each cutoff point, the number of parameters of the added layers, the dimensionality of the corresponding feature map, the dimensionality of the resulting embeddings, and the total number of parameters.

Table 2. Summary of the general characteristics of the CNN feature embedding backbone network for each evaluated cutoff point.

Cut-off point	Added modules	Feature embedding dimensionality	Feature map size	Number of parameters
1	Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3))	64	112, 112	10,058
5	BatchNorm2d(64, eps=1e-05, momentum=0.1) ReLU MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1) Basic-Block-Pair	64	56, 56	158,154
6	Basic-Block-Pair	128	28, 28	684,362
7	Basic-Block-Pair	256	14, 14	2,785,354

4.4. Training of networks

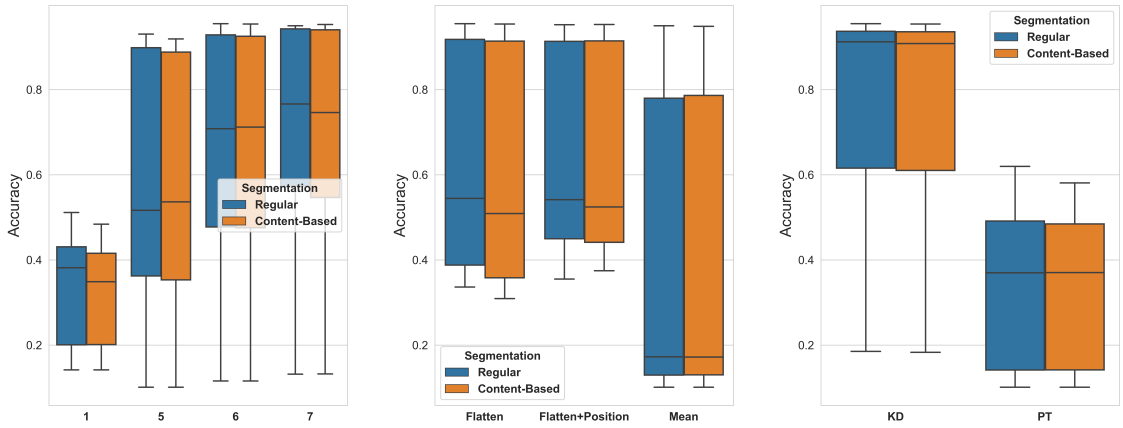
The training of all networks training used the Adam optimizer [Kingma and Ba 2014] and the cross-entropy loss, except for KD, which explored the Kullback-Leibler divergence [Hinton et al. 2015]. We adopted a learning rate and batch size of $1 \cdot 10^{-3}$ and 32, respectively, inferring the values of the balanced accuracy on the test set.

KD experiments involved training the teacher network for 20 epochs with Early Stop [Goodfellow 2016]. The network architectures with different cutoff points were trained using the soft outputs produced by the teacher network.

4.5. MSFA results

The first experiment aimed to assess the impact of the backbone cutoff point, segmentation approach, and knowledge distillation procedure on MSFA performance. In each test, the parameter under analysis was fixed at predefined values, while the remaining parameters varied according to Table 1, similar to [You et al. 2020]. The results are depicted in Figure 2.

Figure 2a shows an expected increase in accuracy as deeper feature-embedding networks are explored. Figure 2b indicates that **Flatten** aggregation outperforms **Mean**, while adding positional information (**Flatten+Position**) does not lead to significant performance improvement. Figure 2c also shows that **KD** significantly improves accuracy, raising the median from 37% to 91%. In all cases, the segmentation process had a mild impact on models' performance. The best-performing model had the following hyperparameters: cutoff point of 7, **KD**, **Flatten** for embedding, and content-based segmentation, achieving an accuracy of 93.92 % ($\pm 1.10\%$).



(a) Cut-point (b) Aggregation (c) Fine-tuning

Figure 2. Boxplot graphs analyzing each MSFA design hyperparameter (see text).

4.5.1. MSFA cost-effectiveness analysis

Complex neural networks typically achieve lower error rates at the cost of increased computational requirements and memory usage. Multiple design alternatives can more effectively analyzed when framed as a multi-objective optimization problem, where the design goal is to minimize both the error rate and the number of parameters. In this context, solutions that cannot improve one objective without compromising the other are denoted as nondominated, forming what is known as the Pareto frontier [Eiben and Smith 2015].

Figure 3 depicts the average error rate versus the number of parameters for each nondominated MSFA model tested using the MSFA approach, all adopting KD, as other PT and KD design alternatives were dominated. This figure also includes the performance of the teacher model (ResNet 18) and the different embedding networks (i.e., with different cutoff points) followed by a classification layer, which are referred here to as reference classifiers.

The models using segment embeddings based on cutoff points 5 and 6 showed the most attractive trade-off between the number of parameters and the error rate. Compared to reference classifiers, MSFA allows for some reduction in the error rate with only a mild increase in the total number of parameters for cutoff points 5 and 6. The most cost-effective model (5, **KD**, **Flatten**, **Content-Based**) achieved an error rate of 9.08% ($\pm$ 1.30%), surpassing the error rate of 13.93% ($\pm$ 1.43%) obtained with the corresponding reference classifier, marked by a blue star in the graph.

4.6. GBEF results

Similarly to MSFA, the impact of each GBEF design hyperparameter is analyzed in Figure 4. Figure 4a shows an increasing trend in accuracy with the number of layers in the embedding extractor, consistent with the results observed for MSFA. Figure 4b indicates that, the difference between using regular segments and content-based segments is not significant. Figure 4c shows that forming edges based solely on the geometric distance between segment centroids produced slightly better results. Figure 4d demonstrates that model accuracy increased with higher hidden layer dimensionality. Figure 4e reveals that

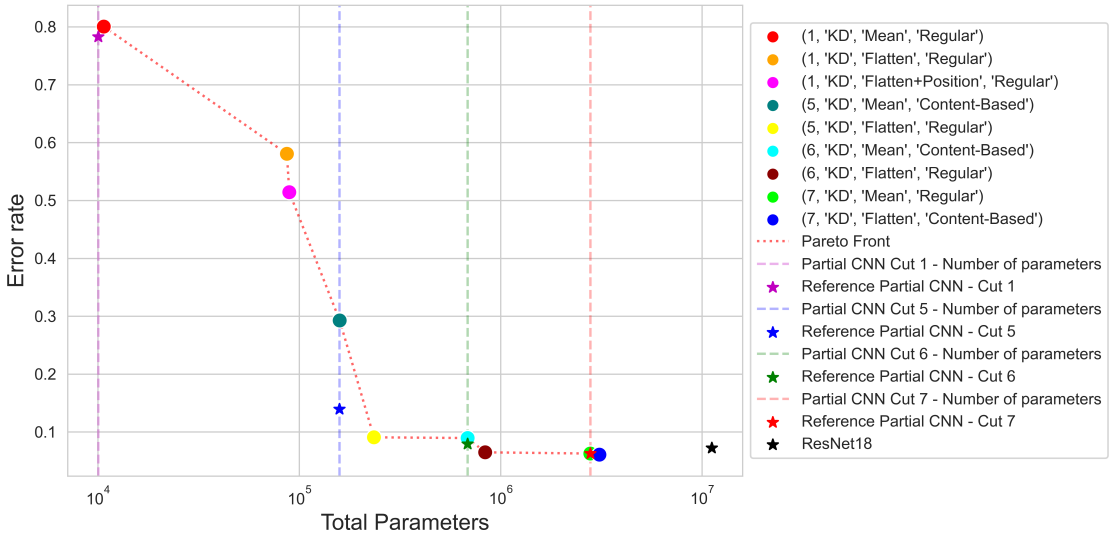


Figure 3. Average error rate and the associated number of parameters for non-dominated models. The legends for MSFA Models are ("Cutoff point," "segment embedding strategy," "aggregation strategy," and "segmentation modality").

increasing the number of GNN layers reduced the accuracy of KD models while improving it for PT models, suggesting that fine-tuned embeddings benefited less from GNN feature enrichment. Among the GNN architectures, GAT performed slightly better for both KD and PT, as shown in Figure 4f. In all cases, the use of KD proved beneficial.

4.7. GBEF Cost-Effectiveness Analysis

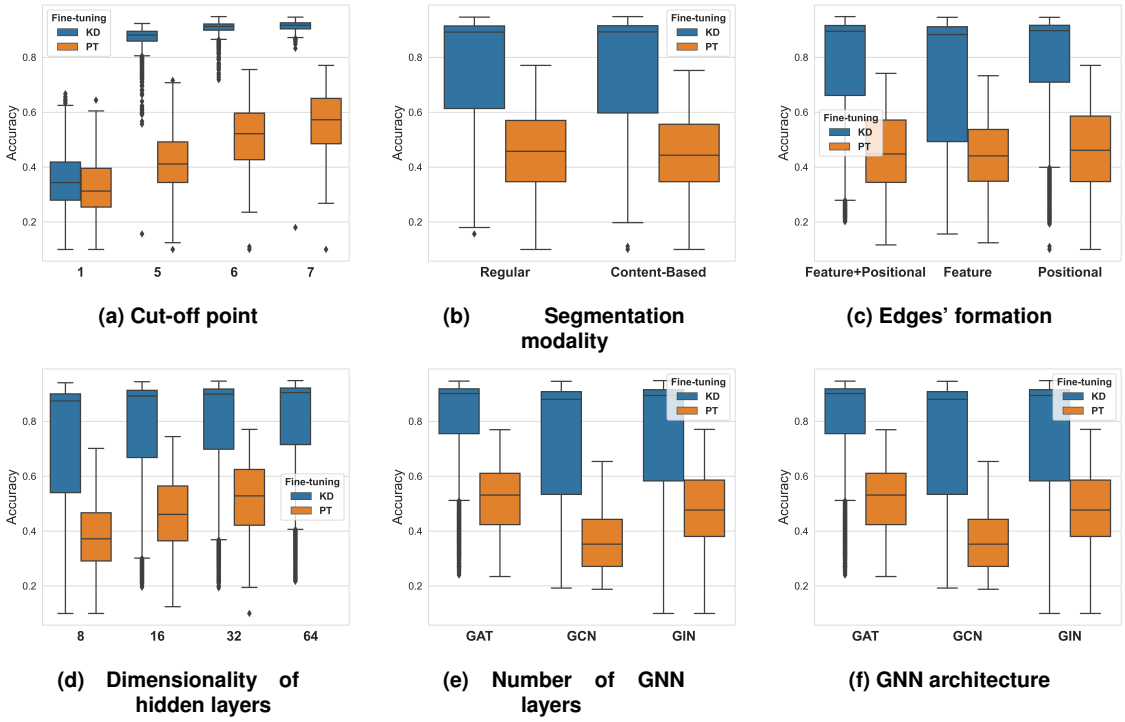
Figure 5 depicts the Pareto front for both GBEF and MSFA. Models closer to the origin exhibit lower error rates with fewer parameters, making them more cost-effective. For models with fewer parameters, particularly those with complexity comparable to or lower than the cutoff 5 reference classifier, the GBEF approach has proven more cost-effective than MSFA. For more complex segment embedding networks, GBEF performed similarly to MSFA.

Among all the experiments performed, MSFA (7, KD, Flatten, Content-Based) achieved the lowest average error rate of 6.08% ($\pm$1.10%). In comparision, the best GBEF model with the same cutoff factor obtained an error rate of 6.42% ($\pm$0.57%) but has 5.56% fewer parameters, utilizing content-based segmentation, two GAT layers, and hidden layers with 64 dimensions. Notably, the reference classifier with the same cutoff factor exhibited an error rate of 7.95% ($\pm$1.72%).

5. Conclusion

This study revisits the model proposed by [Evangelista and Souza Filho 2023], referred here to as GBEF, to more comprehensively analyze its cost-effectiveness by evaluating alternative processing approaches and design choices. In particular, we propose fine-tuning the segment embedding network using Knowledge Distillation.

The experiments utilized a dataset of images captured by an MFLS sonar and compared GBEF with two additional approaches: MSFA and reference classifiers. MSFA



(a) Cut-off point (b) Segmentation modality (c) Edges' formation

(d) Dimensionality of hidden layers (e) Number of GNN layers (f) GNN architecture

Figure 4. Boxplot graphs for analyzing the impact of each design hyperparameter in GBEF performance (see text).

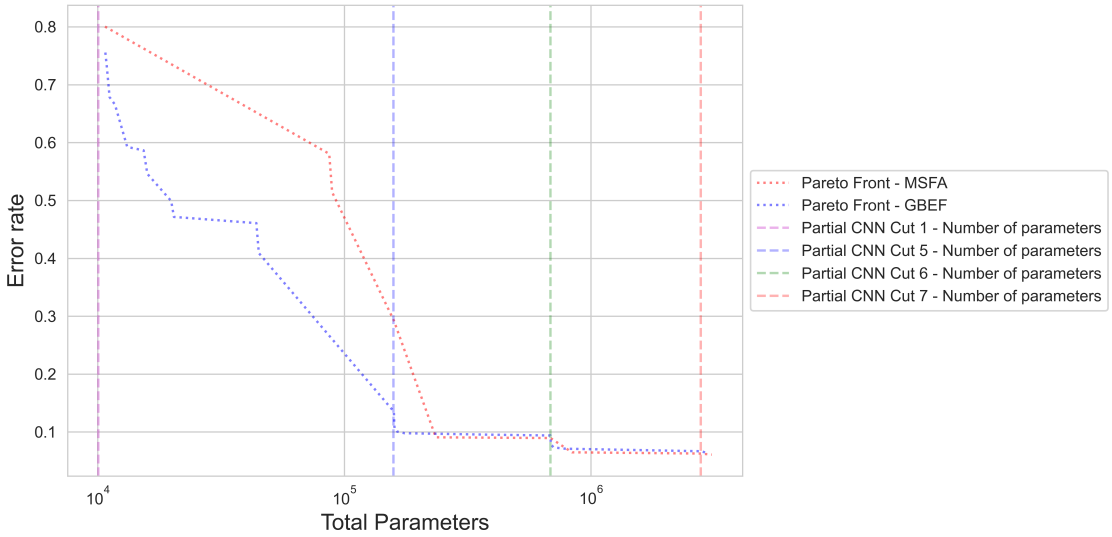


Figure 5. Comparision of the Pareto front of GBEF and MSFA approaches (see text).

employs a similar image processing pipeline to GBEF but does not include the graph-based image segment feature enhancement. Reference classifiers represent a conventional CNN approach, consisting of the initial layers of a pre-trained ResNet network followed by a softmax classification layer.

Results indicate that fine-tuning the segment embedding network is crucial for classification performance. In terms of cost-effectiveness, the processing pipelines performed similarly with more complex, computationally heavier segment embedding models, showing equivalent error rates for a similar number of parameters. However, GBEF exhibited better error rates relative to computational effort for less complex embedding solutions, outperforming simpler approaches like the reference classifiers.

6. Acknowledgments

To CNPq, FAPERJ, and CAPES. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001

References

Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., and Süsstrunk, S. (2012). SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282.

Avelar, P. H., Tavares, A. R., da Silveira, T. L., Jung, C. R., and Lamb, L. C. (2020). Superpixel image classification with graph attention networks. In *2020 33rd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 203–209. IEEE.

Belkin, M. and Niyogi, P. (2001). Laplacian eigenmaps and spectral techniques for embedding and clustering. *Advances in Neural Information Processing Systems*, 14.

Defferrard, M., Bresson, X., and Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in Neural Information Processing Systems*, 29.

Dos Santos, M. M., De Giacomo, G. G., Drews-Jr, P. L., and Botelho, S. S. (2022). Cross-view and cross-domain underwater localization based on optical aerial and acoustic underwater images. *IEEE Robotics and Automation Letters*, 7(2):4969–4974.

Dwivedi, V. P., Joshi, C. K., Luu, A. T., Laurent, T., Bengio, Y., and Bresson, X. (2023). Benchmarking graph neural networks. *Journal of Machine Learning Research*, 24(43):1–48.

Eiben, A. E. and Smith, J. E. (2015). *Introduction to evolutionary computing*. Springer.

Evangelista, G. A. and Souza Filho, J. B. O. (2023). Graph-based multibeam forward looking acoustic image classification. In *Anais do XX Encontro Nacional de Inteligência Artificial e Computacional*, pages 756–770, Porto Alegre, RS, Brasil. SBC.

Fey, M., Lenssen, J., Weichert, F., and Muller, H. (2018). SplineCNN: Fast geometric deep learning with continuous B-Spline kernels. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 869–877. IEEE Computer Society.

Fey, M. and Lenssen, J. E. (2019). Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

Girshick, R. (2015). Fast R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448.

Goodfellow, I. (2016). *Deep Learning*. MIT Press.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Huo, G., Wu, Z., and Li, J. (2020). Underwater object classification in sidescan sonar images using deep transfer learning and semisynthetic training data. *IEEE Access*, 8:47407–47418.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kipf, T. N. and Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.

Knyazev, B., Taylor, G. W., and Amer, M. (2019). Understanding attention and generalization in graph neural networks. *Advances in Neural Information Processing Systems*, 32.

Monti, F., Boscaini, D., Masci, J., Rodola, E., Svoboda, J., and Bronstein, M. M. (2017). Geometric deep learning on graphs and manifolds using mixture model CNNs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5115–5124.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Fei-Fei, L., et al. (2015). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115:211–252.

Sinai, A., Amar, A., and Gilboa, G. (2016). Mine-like objects detection in side-scan sonar images using a shadows-highlights geometrical features space. In *OCEANS 2016 MTS/IEEE Monterey*, pages 1–6. IEEE.

Singh, D. and Valdenegro-Toro, M. (2021). The marine debris dataset for forward-looking sonar semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3741–3749.

Steiniger, Y., Kraus, D., and Meisen, T. (2022). Survey on deep learning based computer vision for sonar imagery. *Engineering Applications of Artificial Intelligence*, 114:105157.

Van der Walt, S., Schönberger, J. L., Nunez-Iglesias, J., Boulogne, F., Warner, J. D., Yager, N., Gouillart, E., and Yu, T. (2014). scikit-image: Image Processing in Python. *PeerJ*, 2:e453.

Vasudevan, V., Bassenne, M., Islam, M. T., and Xing, L. (2023). Image classification using graph neural network and multiscale wavelet superpixels. *Pattern Recognition Letters*, 166:89–96.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 31, page 6000–6010. Curran Associates Inc.

Wightman, R. (2019). PyTorch image models. `https://github.com/rwightman/pytorch-image-models`.

Xie, K., Yang, J., and Qiu, K. (2022). A dataset with multibeam forward-looking sonar for underwater object detection. *Scientific Data*, 9:739.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2019). How powerful are graph neural networks? In *International Conference on Learning Representations (ICLR)*.

Yang, D., Cheng, C., Wang, C., Pan, G., and Zhang, F. (2022). Side-scan sonar image segmentation based on multi-channel CNN for AUV navigation. *Frontiers in Neurorobotics*, 16.

You, J., Ying, Z., and Leskovec, J. (2020). Design space for graph neural networks. *Advances in Neural Information Processing Systems*, 33:17009–17021.

Yu, Y., Zhao, J., Gong, Q., Huang, C., Zheng, G., and Ma, J. (2021). Real-time underwater maritime object detection in side-scan sonar images based on Transformer-YOLOv5. *Remote Sensing*, 13:3555.