GRAPH CONVOLUTIONAL NETWORKS WITH AUTOENCODER-BASED COMPRESSION AND MULTI-LAYER GRAPH LEARNING

L. GIUSTI[1] C. BATTILORO[2] P. DI LORENZO[2] S. BARBAROSSA[2]
[1] DIAG DEPARTMENT, SAPIENZA UNIVERSITY OF ROME, VIA ARIOSTO 25, 00185, ROME, ITALY
[2]CDIET DEPARTMENT, SAPIENZA UNIVERSITY OF ROME, VIA EUDOSSIANA 18, 00184, ROME, ITALY

SAPIENZA UNIVERSITÀ DI ROMA
icassp 2022 Singapore

ABSTRACT

The aim of this work is to propose a novel architecture and training strategy for compress the convolutional features at multiple hidden layers, hinging on a novel end-to-end training procedure that learns different graph representations per each layer.

Contribution: We exploit autoencoders in each layer, before applying the pointwise non-linearity, so that the convolutional features can be tunably compressed in an information-rich embedding. Then, since compression calls for learning a new graph representation to be used in the following layer, we formulate a novel training strategy that *jointly* optimizes the GNN weight parameters *and* the graph representations at different layers.

SIGNALS ON GRAPHS

- Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a weighted undirected graph
- The sets $\mathcal{V} = \{1, 2, ..., N\}$ and $\mathcal{E} = \{a_{i,j}\}_{i,j\in\mathcal{V}}$ are the sets of vertices and edges, respectively
- The weights $a_{i,j} \geq 0$ if there is a relationship from vertex i to vertex j, or $a_{i,j} = 0$ otherwise.
- The adjacency matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$: a collection of all weights, i.e., $\mathbf{A} = \{a_{i,j}\}$, $i, j = 1, ..., N$
- The Laplacian matrix: $\mathbf{L} = \mathrm{diag}(\mathbf{1}^T\mathbf{A}) - \mathbf{A}$, where $\mathrm{diag}(\boldsymbol{x})$ is a matrix having $\boldsymbol{x}$ as main diagonal, and zeros elsewhere
- A graph signal (or data) is defined as a one-to-one mapping from the set $\mathcal{V}$ of vertices to the set of real numbers:

$$\boldsymbol{x} = \mathcal{V} \rightarrow \mathbb{R} \tag{1}$$

- An order K linear shift invariant graph filter (LSIGF) can be written as a K-degree polynomial of the shift operator $\mathbf{S}$, with coefficients $\boldsymbol{h} = [h_0, ..., h_{K-1}]^T$.
- Let $\boldsymbol{u}$ and $\boldsymbol{y}$ be the input and the filtered signals, respectively, we have:

$$\boldsymbol{y} = \sum_{k=0}^{K-1} [\boldsymbol{h}]_k \mathbf{S}^k \boldsymbol{u}. \tag{2}$$

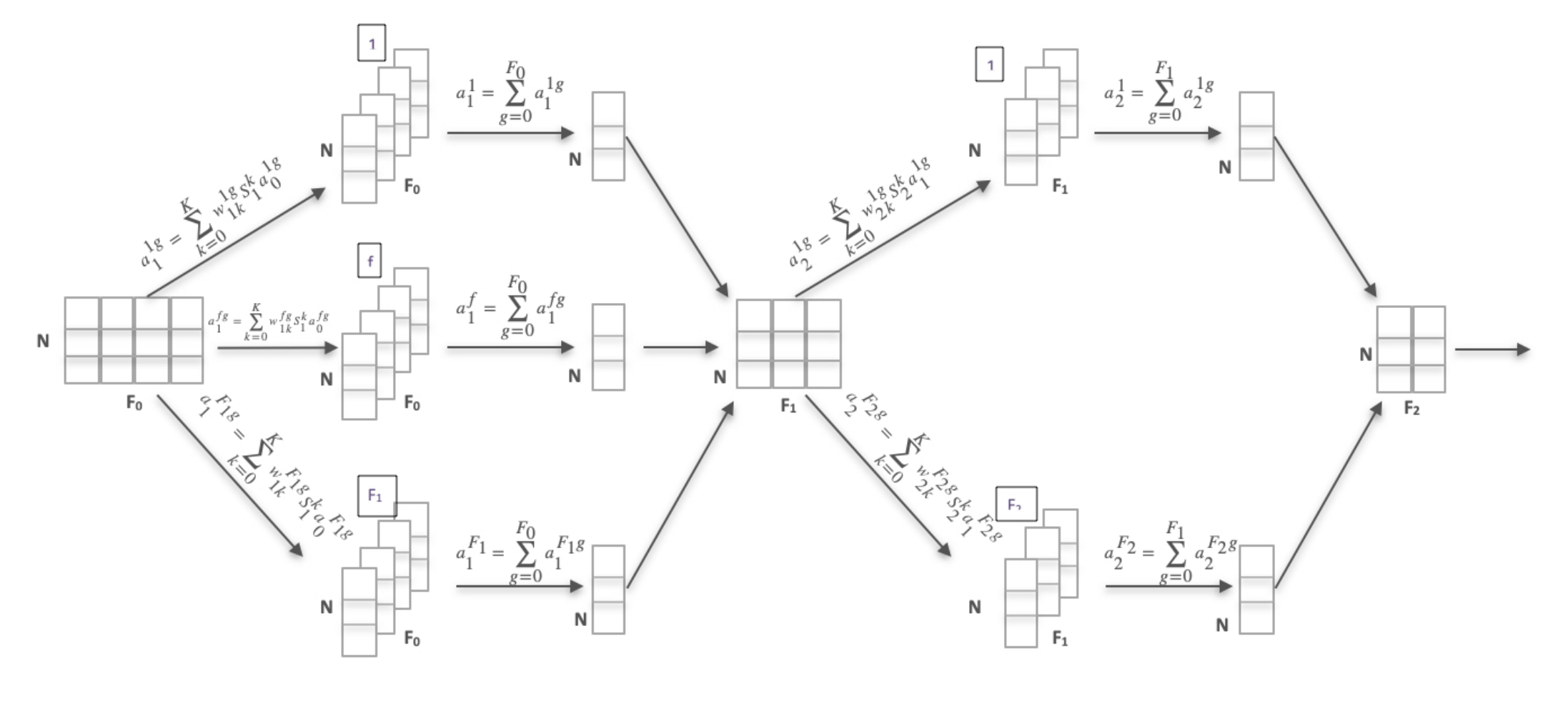
- The LSIGF are able to account for the local structure of the graph, requiring information only from the K-neighborhood of each node.
- Linear and shift invariant graph filters represent a legit generalization of the convolution operation for signals supported on graphs [1], and are the basic building block of GCNs.

GRAPH CONVOLUTIONAL NETWORKS

- The l-th layer of a GCN, taking as input $\widetilde{\mathbf{Z}}_{l-1} = \{\widetilde{\boldsymbol{z}}^f_{l-1}\}_{f=1}^{F_{l-1}}$ and yielding as output $\widetilde{\mathbf{Z}}_l = \{\widetilde{\boldsymbol{z}}^g_l\}_{g=1}^{F_l}$, with pointwise non-linearity $\sigma_l(\cdot)$, reads as: [2]:

$$\widetilde{\boldsymbol{z}}^g_l := \sigma_l\left(\sum_{f=1}^{F_{l-1}} \sum_{k=0}^{K_l-1} [\boldsymbol{h}_l^{fg}]_k \mathbf{S}^k \widetilde{\boldsymbol{z}}^f_{l-1}\right), \quad g = 1, ..., F_l. \tag{3}$$

- The order K_l of the filters, the number F_l of convolutional features of the output, and the non-linearity $\sigma_l(\cdot)$ are hyperparameters to be chosen at each layer
- A GCN of depth L with input data $\mathbf{X}$ is built as the stack of L layers defined as in (3), where $\widetilde{\mathbf{Z}}_0 = \mathbf{X}$
- Based on the learning task, an additional multi layer perceptron (MLP) can be inserted after the last layer


N
F₀
F₁
F₂
1
f
$a_1^1 = \sum_{g=0}^{F_0} a_1^{1g}$
$a_1^{1g} = \sum_{k=0}^{K} w_{1k}^{1g} S_1^k a_0^{1g}$
$a_2^1 = \sum_{g=0}^{F_1} a_2^{1g}$
$a_2^{F_2} = \sum_{g=0}^{F_1} a_2^{F_2 g}$

AUTOENCODER-AIDED GRAPH CONV NETS

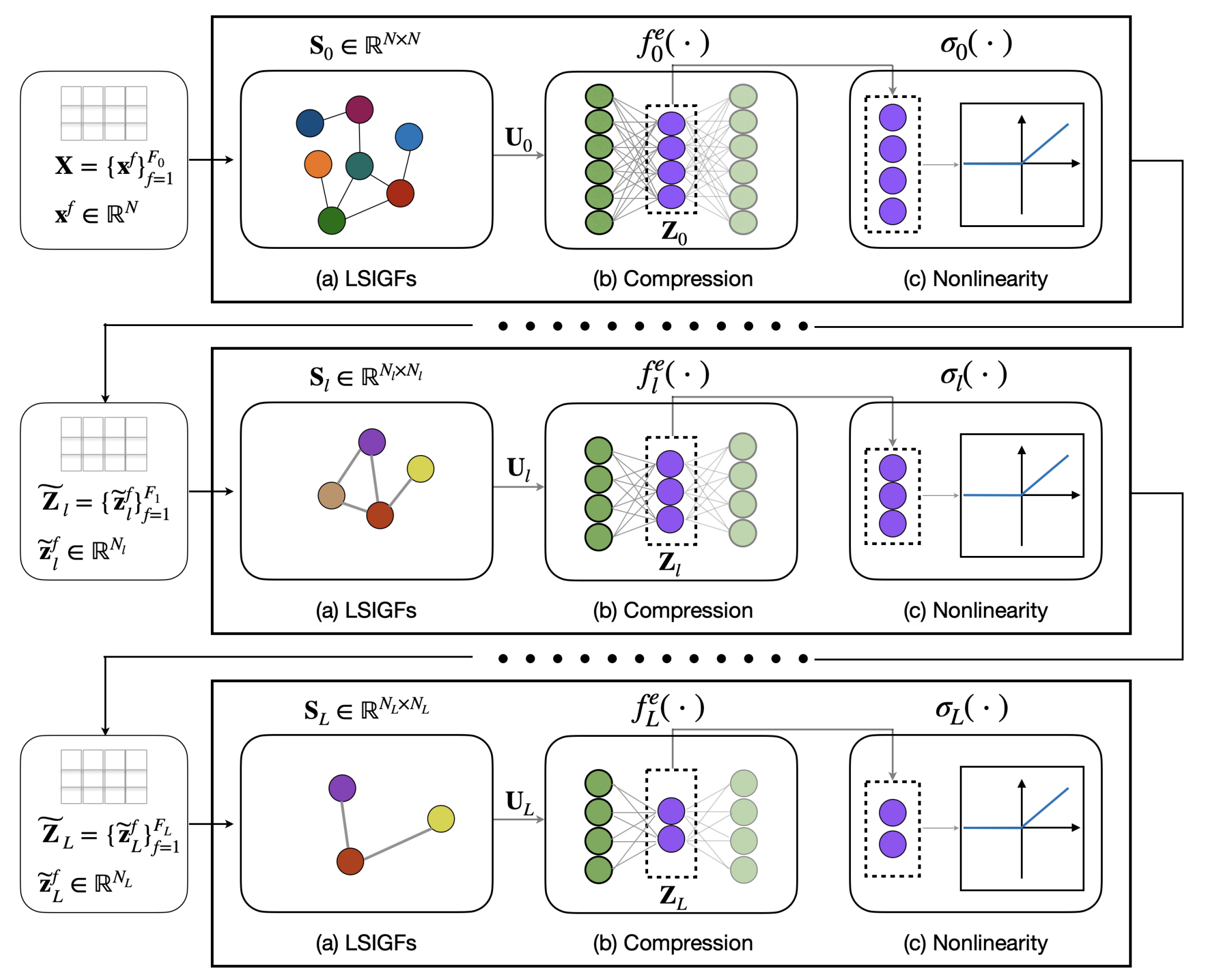
- The idea is to exploit autoencoders to perform representation learning and dimensionality reduction in the context of GCNs [3].
- Each layer is composed of three main stages: (i) A linear shift invariant graph filtering stage; (ii) Autoencoder-based compression; (iii) Pointwise non-linearity
- Autoencoders are able to reduce the dimension of the hidden layers' convolutional features, thus learning powerful and task-oriented low-dimensional representations in a data-driven fashion.
- To this aim, the layer in (3) is modified in the following way:

$$\widetilde{\boldsymbol{z}}^g_l = \sigma_l\big(\underbrace{f^e_l(\boldsymbol{u}^g_l)}_{\boldsymbol{z}^g_l}\big), \quad g = 1, ..., F_l, \tag{4}$$

Where $\widetilde{\boldsymbol{z}}^g_l \in \mathbb{R}^{N_l}$, $N_l \in \mathbb{N}$ is (generally) smaller than N_{l-1}, $f^e_l : \mathbb{R}^{N_{l-1}} \rightarrow \mathbb{R}^{N_l}$ is the encoder function of an autoencoder $f^d_l \circ f^e_l : \mathbb{R}^{N_{l-1}} \rightarrow \mathbb{R}^{N_{l-1}}$ associated with the l-th layer of the GCN, and

$$\boldsymbol{u}^g_l = \sum_{f=1}^{F_{l-1}} \sum_{k=0}^{K_l-1} [\boldsymbol{h}_l^{fg}]_k \mathbf{S}_l^k \widetilde{\boldsymbol{z}}^f_{l-1}, \quad g = 1, ..., F_l, \tag{5}$$

where $\mathbf{S}_l$ denotes the shift operator associated with the l-th layer. The compressed features at layer l are denoted as $\mathbf{Z}_l = \{\boldsymbol{z}^g_l\}_{g=1}^{F_l}$, and the final output of the layer is $\widetilde{\mathbf{Z}}_l \in \mathbb{R}^{N_l\times F_l}$. We call the stack of L layers as in (4) an Autoencoder-Aided Graph Convolutional Network (AA-GCN).


$\mathbf{X} = \{\mathbf{x}^f\}_{f=1}^{F_0}$
$\mathbf{x}^f \in \mathbb{R}^N$
$\mathbf{S}_0 \in \mathbb{R}^{N\times N}$
$f_0^e(\cdot)$
$\sigma_0(\cdot)$
$\mathbf{U}_0$
$\mathbf{Z}_0$
(a) LSIGFs
(b) Compression
(c) Nonlinearity
$\widetilde{\mathbf{Z}}_l = \{\widetilde{\mathbf{z}}_l^f\}_{f=1}^{F_l}$
$\widetilde{\mathbf{z}}_l^f \in \mathbb{R}^{N_l}$
$\mathbf{S}_l \in \mathbb{R}^{N_l\times N_l}$
$f_l^e(\cdot)$
$\sigma_l(\cdot)$
$\mathbf{U}_l$
$\mathbf{Z}_l$
$\widetilde{\mathbf{Z}}_L = \{\widetilde{\mathbf{z}}_L^f\}_{f=1}^{F_L}$
$\widetilde{\mathbf{z}}_L^f \in \mathbb{R}^{N_L}$
$\mathbf{S}_L \in \mathbb{R}^{N_L\times N_L}$
$f_L^e(\cdot)$
$\sigma_L(\cdot)$
$\mathbf{U}_L$
$\mathbf{Z}_L$

PROBLEM FORMULATION

$$\begin{aligned}
&\min_{\{\mathbf{A}_l\}_{l=1}^L,\mathbf{H},\mathbf{W}} \quad \mathcal{L}(\{\mathbf{A}_l\}_{l=1}^L, \mathbf{H}, \mathbf{W}; \{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i\in\mathcal{T}})\\
&+ \eta\sum_{l=1}^{L}\sum_{g=1}^{F_l} ||f^d_l \circ f^e_l(\boldsymbol{w}_l; \boldsymbol{u}^g_l) - \boldsymbol{u}^g_l||_2^2; \rightarrow \text{Autoencoders' Loss}\\
&+ \beta\sum_{l=1}^{L} \mathrm{Tr}\{\widetilde{\mathbf{Z}}_l^T \mathbf{L}_l \widetilde{\mathbf{Z}}_l\}; \rightarrow \text{Promote signal smoothness}\\
&- \gamma\sum_{l=1}^{L} \mathbf{1}^T\log(\mathbf{A}_l\mathbf{1}); \rightarrow \text{Penalize disconnected components}\\
&+ \lambda\sum_{l=1}^{L} ||\mathbf{A}_l||_F^2; \rightarrow \text{Weights' regularizer}\\
&\text{subject to}\\
&\qquad [\mathbf{A}_l]_{i,i} = 0 \rightarrow \text{no self-loops in the learnt graphs} \qquad (6)\\
&\qquad [\mathbf{A}_l]_{i,j} = [\mathbf{A}_l]_{j,i} \geq 0, \qquad \forall\, i, j, l \rightarrow \text{edge weights must be positive and symmetric}\\
&\qquad \mathrm{Tr}\{\mathbf{L}_l\} = d_l, \quad \forall\, l \rightarrow \text{avoid null solutions}
\end{aligned}$$

where λ, β, γ, η, and d_l are non-negative parameters to be tuned.

HALF-VECTOR

- Since the adjacency matrices are symmetric, the number of variables of the optimization problem can be greatly reduced (approximatively by a factor of two) solving for the lower triangular parts of $\mathbf{A}_l$ for all $l = 1, \ldots, L$
- Let $\boldsymbol{\alpha}_l := \mathrm{vech}(\mathbf{A}_l) \in \mathbb{R}^{\frac{N(N+1)}{2}}$ be the half-vectorization of $\mathbf{A}_l$, obtained by vectorizing only the lower triangular part of $\{\mathbf{A}_l\}_l$
- Then, the following relations hold:

$$\mathrm{vec}(\mathbf{A}_l) = \mathbf{M}_d\boldsymbol{\alpha}_l \quad \Longleftrightarrow \quad \mathbf{A}_l = \mathrm{vec}^{-1}(\mathbf{M}_d\boldsymbol{\alpha}_l), \tag{7}$$

- $\mathrm{vec}(\cdot)$ and $\mathrm{vec}^{-1}(\cdot)$ are the vectorization and the inverse vectorization operators, respectively
- $\mathbf{M}_d \in \mathbb{R}^{N^2\times\frac{N(N+1)}{2}}$ is the (highly sparse) duplication matrix

All the objective terms and the constraints can be easily recast in terms of the variables $\boldsymbol{\alpha}_l$, for $l = 1, \ldots, L$

ALGORITHMIC SOLUTION

Algorithm 1 : AA-GCN TRAINING

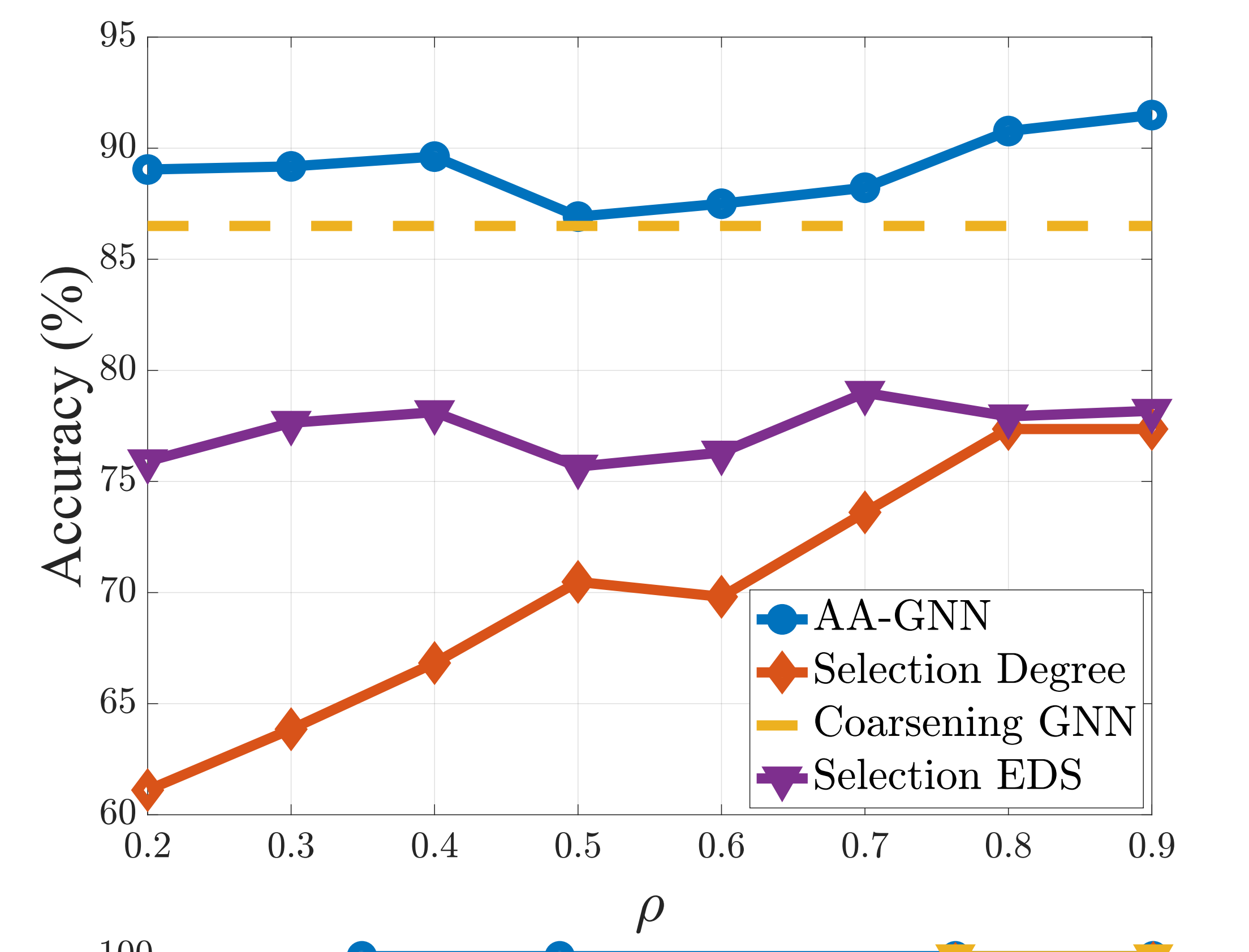
```
    Inputs:
        $\mu \in \mathbb{R}$: Learning rate.
        $\Delta_\mu(\cdot)$: Optimizer-dependent backpropagation step.
        $\Pi(\cdot)$: Projection operator on the feasible set.
        $E \in \mathbb{N}_+$: Maximum number of training iterations.
        $\{\mathcal{B}_t\}_{t=1}^E$: Training dataset batches
        Estimates initializations $\widehat{\mathbf{H}}_0$, $\widehat{\mathbf{W}}_0$ and $\{\widehat{\boldsymbol{\alpha}}_{l,0}\}_l$.
        Loss $\mathcal{L}(\cdot)$.
    Outputs:
        $\{\widehat{\boldsymbol{\alpha}}_l\}_l$: Learned graph encodings.
        $\widehat{\mathbf{H}}$: Learned graph filters weights.
        $\widehat{\mathbf{W}}$: Learned autoencoders weights.
1: function AA-GCN TRAINING(Inputs)
2:     for $t \in [1, E]$ do
3:         $\widehat{\mathbf{H}}_{t+1} = \Delta_\mu\Big(\nabla_{\mathbf{H}}\mathcal{L}\,(\widehat{\mathbf{H}}_t; \mathcal{B}_t, \{\widehat{\boldsymbol{\alpha}}_{l,t}\}_l, \widehat{\mathbf{W}}_t)\Big)$
4:         $\widehat{\mathbf{W}}_{t+1} = \Delta_\mu\Big(\nabla_{\mathbf{W}}\mathcal{L}\,(\widehat{\mathbf{W}}_t; \mathcal{B}_t, \{\widehat{\boldsymbol{\alpha}}_{l,t}\}_l, \widehat{\mathbf{H}}_t)\Big)$
5:         $\widehat{\boldsymbol{\alpha}}_{l,t+1} = \Pi\Big(\Delta_\mu\Big(\nabla_{\boldsymbol{\alpha}_l}\mathcal{L}\,(\{\widehat{\boldsymbol{\alpha}}_{l,t}\}_l, \mathcal{B}_t, \widehat{\mathbf{W}}_t, \widehat{\mathbf{H}}_t)\Big)\Big),\ \forall l$
6:     return $\{\widehat{\boldsymbol{\alpha}}_l\}_l = \{\widehat{\boldsymbol{\alpha}}_{l,E}\}_l,\ \widehat{\mathbf{W}} = \widehat{\mathbf{W}}_E,\ \widehat{\mathbf{H}} = \widehat{\mathbf{H}}_E$
```

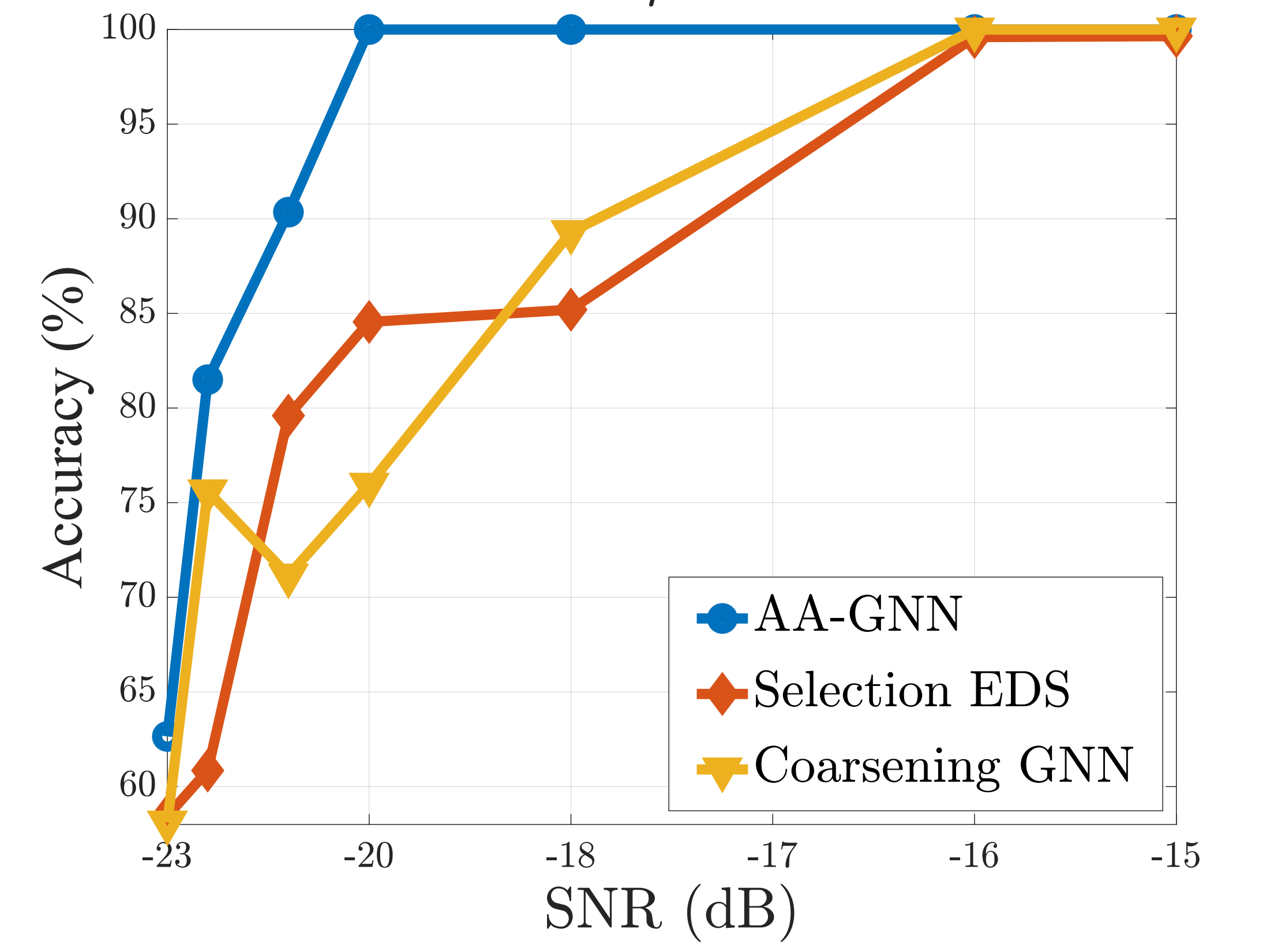
PROJECTION ALGORITHM

Algorithm 2 Euclidean projection of $\widetilde{\boldsymbol{\alpha}}_l$ onto $\mathcal{X}$ [4]

```
    Input:
        $\{\widetilde{\boldsymbol{\alpha}}_l\}_l$: Updated graphs parameters.
        $\{N_l\}_l$: Number of nodes of $\{\mathcal{G}_l\}_l$.
        $\{d_l\}_l$: Expected degrees of nodes at layer $l$.
    Output:
        $\{\widehat{\boldsymbol{\alpha}}_l\}_l$: Projected updated graphs estimates.
    Operator $\Pi_{\mathcal{X}}(\{\widetilde{\boldsymbol{\alpha}}_{l,t+1}\}_l)$:
        for $l = 1, ..., L$ do:
            Set $[\boldsymbol{A}_l]_{i,i} = 0$;
            Sort $\widetilde{\boldsymbol{\alpha}}_l$ in increasing order;
            $\rho := \max_{1\leq j\leq dim(\widetilde{\boldsymbol{\alpha}}_l)} \underbrace{[\widetilde{\boldsymbol{\alpha}}_l]_j + \frac{1}{j}\left(\frac{d_l}{2} - \sum_{i=1}^{j}[\widetilde{\boldsymbol{\alpha}}_l]_i\right)}_{[\tilde{\boldsymbol{\alpha}}_l^*]_j}$ s.t. $[\tilde{\boldsymbol{\alpha}}_l^*]_j > 0$
            $\lambda := \frac{1}{\rho}\left(\frac{d_l}{2} - \sum_{i=1}^{\rho}[\widetilde{\boldsymbol{\alpha}}_l]_i\right)$
            Set $[\hat{\boldsymbol{\alpha}}_l]_i = \max\,[\widetilde{\boldsymbol{\alpha}}_\mathrm{l}]_\mathrm{i} + \lambda, 0$
```

NUMERICAL RESULTS





- The first results show the accuracy score compared to the compression ratio: $\rho = N_1/N$
- The results show the accuracy score compared to the SNR of the training data: $\mathrm{SNR} = 10\log_{10}(\sigma^2_{\mathcal{T}}/\sigma^2_\epsilon)$, $\sigma^2_{\mathcal{T}}$ and σ^2_ε are the variance of the data used for training our model and the variance of the AWGN, respectively.
- For a fair comparison, the second hidden layer does not provide a coarser version of the first one

CONCLUSIONS

- We have enabled tunable compression of the convolutional features, while learning different graph representations jointly with the GNN parameters
- The architecture scales well with the number of nodes of the input graph, extracting higher level representations of the convolutional features
- Experiments illustrate the competitive performance of our architecture with respect to state of the art methods
- Future developments of this research trend include: Topological Neural Networks, Explainability, include additional regularisations to the autoencoders' loss

REFERENCES

[1] S. Segarra, A. G. Marques, and A. Ribeiro. Optimal graph-filter design and applications to distributed linear network operators. *IEEE Transactions on Signal Processing*, 65(15):4117–4131, 2017.

[2] F. Gama, A. G. Marques, G. Leus, and A. Ribeiro. Convolutional neural network architectures for signals supported on graphs. *IEEE Transactions on Signal Processing*, 67(4):1034–1049, 2019.

[3] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*, pages 318–362. MIT Press, Cambridge, MA, 1986.

[4] W. Wang and M. A. Carreira-Perpinán. Projection onto the probability simplex: An efficient algorithm with a simple proof and an application. *arXiv:1309.1541*, 2013.