ICGNN REVIEW: A REVIEW OF INPUT CONVEX GRAPH NEURAL NETWORKS AND THEIR APPLICATIONS*

Michael E. Sperling & Siddharth Raipal Department of Computer Science
University of Southern California
Los Angeles, CA 90089, USA
{mesperli,raipal}@usc.edu

ABSTRACT

Input Convex Graph Neural Networks (ICGNNs) represent a novel class of neural architectures that leverage the principles of convex optimization to model graph-structured data. By enforcing input convexity, ICGNNs ensure global optimality in their predictions, making them particularly suitable for applications requiring robust and interpretable models. This paper provides a review and further experimentation on ICGNNs, highlighting their theoretical foundations, architectural innovations, and practical applications. We discuss their particular advantages when used on dynamic problems that can benefit from the convexity constraints that ICGNNs can impose as well as briefly discuss some of their limitations when applied to tasks which require less complexity.

1 INTRODUCTION

Graph-structured data arise in a wide range of applications, from social and biological networks to physical systems and recommendation engines. Graph Neural Networks (GNNs) have emerged as a powerful paradigm for learning on such data by propagating and aggregating information across nodes and edges. However, standard GNNs often rely on nonconvex objectives that can admit multiple local optima and lack guarantees of global consistency. In parallel, Input Convex Neural Networks (ICNNs) enforce convexity in their input–output mapping, ensuring that inference corresponds to a unique global optimum (Amos et al., 2016). In this work, we marry these two strands by reviewing Input Convex Graph Neural Networks (ICGNNs) and their recurrent variant (ICGRNN) for modeling dynamic, time-evolving graphs (Park et al., 2022).

Our main contributions are threefold. First, we provide a concise review of the theoretical foundations of ICGNNs. Second, we implement and share code on ICGRNN. Third, we conduct extensive experiments on synthetic heat diffusion networks and the standard CORA and PubMed datasets.

Empirically, our ICGRNN achieves a temperature trajectory MSE of 0.00724 and a control input MSE of 0.00010 on the heat diffusion task. Our non-recurrent ICGCN variant matches competitive baselines with accuracies of 0.78 (CORA) and 0.73 (PubMed). These results illustrate that convex graph models can offer robustness and interpretability where global optimality is critical, while revealing performance gaps in unguided representation tasks.

2 RELATED WORK + BACKGROUND

We will be architecturing and referencing formulas and equations found in Amos et al. (2016) and Park et al. (2022), please go to these papers for further information on the theory behind these structures and the reasoning behind the holding of convexity through these layered models.

*Code available at: https://github.com/Master2679/ICGNN

2.1 GNN

Graph Neural Networks (GNNs) learn representations of graph-structured data through message-passing between nodes. While successful in domains like social networks, molecular modeling, and recommendation systems, traditional GNNs face challenges including oversmoothing and limited expressiveness. A GNN layer will use the previous layer to create a new layer as follows:

$$e_{uv}^{(k)} = \phi_e(h_u^{(k-1)}, h_v^{(k-1)}, e_{uv}^{(k-1)}) \tag{1}$$

$$h_v^{(k)} = \phi_h(h_v^{(k-1)}, \rho_{u \in \mathcal{N}(v)} e_{uv}^{(k)}) \tag{2}$$

where $\mathbf{h}_v^{(k)}$ is the node feature vector at layer k for node v, ϕ_e and ϕ_h are functions that are typically modeled by neural networks, whose weight and bias vectors are learned through training, and ρ is the aggregation function that combines information from neighboring nodes. The process is repeated for multiple layers to capture higher-order relationships.

2.2 ICNN

Input Convex Neural Networks (ICNNs) are a class of neural networks that enforce convexity in their input space. This property allows ICNNs to guarantee global optimality in their predictions, making them particularly suitable for applications requiring robust and interpretable models. ICNNs have been successfully applied to various tasks, including regression, classification, and optimization problems. However, their application to graph-structured data has been limited, leading to the development of ICGNNs as a promising extension of the ICNN framework. The ICNN formulation is as follows. For i = 0, .., k-1, the function is defined as:

$$z_0 = x, z_{i+1} = \sigma_i \left(\mathbf{W}_i^{(z)} z_i + W_i^{(x)} x + \mathbf{b}_i \right), f_\theta(x) = z_k \tag{3}$$

where x is the input, z_i is the intermediate representation at layer i, σ_i is the activation function, $\mathbf{W}_i^{(z)}$ and $\mathbf{W}_i^{(x)}$ are the learnable weight matrices, and $\mathbf{b}_i$ is the bias term. The final output $f_\theta(x)$ is obtained after passing through k layers of the network. It maintains convexity if $W_i^{(z)}$ is a non-negative matrix and σ_i is convex and non-decreasing. The convexity of the function is preserved through the composition of convex functions, ensuring that the overall network remains convex.

2.3 ICGNN

Input Convex Graph Neural Networks (ICGNNs) integrate the principles of Input Convex Neural Networks (ICNNs) and Graph Neural Networks (GNNs) to form a robust framework for learning on graph-structured data.

The formulation of ICGNNs is as follows: ICGNNs achieve their convexity properties by employing Input Convex Neural Networks (ICNNs) for the functions $\phi_\theta(\cdot)$ and $\psi_\theta(\cdot)$, alongside commonly used aggregation functions (e.g., sum, mean, max) for $\rho(\cdot)$. A generalized ICGNN operates on a graph $\mathcal{G}$, where $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ The updated graph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ is computed as follows:

$$\epsilon'_{ij} = \phi_\theta(\nu_i, \nu_j, \epsilon_{ij}) \quad \forall \epsilon_{ij} \in \mathcal{E}^c \tag{4}$$

$$\nu'_j = \psi_\theta \left(\nu_j, \rho \left(\{ \epsilon'_{ij} \mid i \in \mathcal{N}_j \} \right) \right) \quad \forall \nu_j \in \mathcal{V} \tag{5}$$

2.4 ICGRNN

The Input Convex Graph Recurrent Neural Network (ICGRNN) is an advanced architecture that integrates the principles of Input Convex Graph Neural Networks (ICGNNs) with recurrent neural networks (RNNs) to model dynamic, time-evolving graph-structured data. By enforcing input convexity, it ensures global optimality in predictions, enhancing robustness and interpretability—critical advantages for control and optimization applications requiring reliable solutions.

Consider a sequence of graphs:

$$\{G^{(t)} = (\mathcal{V}^{(t)}, \mathcal{E}^{(t)}, X^{(t)}, E^{(t)})\}_{t=0}^{T-1},$$

where $\mathcal{V}^{(t)}$ is the set of nodes, $\mathcal{E}^{(t)}$ is the set of edges, $X^{(t)}$ represents node features, and $E^{(t)}$ represents edge features at time t.

The ICGRNN evolves over T timesteps as follows:

$$H^{(0)} = X^{(0)}, \tag{6}$$

$$H^{(t+1)} = \Big(\underbrace{\mathrm{ICGNN}_t\big(\ldots \mathrm{ICGNN}_1\big(H^{(0)}, G^{(0)}\big)\ldots, G^{(t)}\big)}_{\text{stacked } t \text{ ICGNN modules}}\Big), \quad t = 0, 1, \ldots, T-1, \tag{7}$$

$$O^{(T)} = \mathrm{Decoder}\big(H^{(T)}\big) \tag{8}$$

where here we stack t ICGNNs as defined in the above subsection and pass their Hidden States as a matrix of their node feature vectors, which are represented by H^t. Each ICGNN has a message passing and aggregation and can include multiple layers. Finally, to be able to see the output at the end of the T time steps, by passing our Hidden States at time t through the Decoder function, which is represented by a non-decreasing ICGNN to preserve convexity.

3 METHODOLOGY

Our research focuses on developing Input Convex Graph Recurrent Neural Networks (ICGRNNs) for optimal control in heat diffusion systems. The implementation combines graph neural networks, input convexity constraints, and recurrent structure to model spatiotemporal dynamics on graph-structured data. Our implementation, including all training scripts, model architectures, and evaluation pipelines, is available at:
https://github.com/Master2679/ICGNN.

3.1 ARCHITECTURE DESIGN

Our ICGRNN implementation uses PyTorch and PyTorch Geometric (Fey & Lenssen, 2019)libraries with four key components: (1) Convex Layer as the fundamental building block, ensuring input convexity through non-negative weights; (2) ConvexLayerICNN, a multi-layer network with skip connections from input to each hidden layer; (3) Convex Message Passing layer for information flow across the graph while preserving convexity; and (4) a recurrent framework incorporating input-to-hidden, hidden-to-hidden transformations, and dedicated output layers for temperature and control prediction. This design maintains convexity constraints throughout, providing theoretical guarantees for finding global optima in control problems.

3.2 DATASET GENERATION AND EXPERIMENTAL SETUP

We generated synthetic heat diffusion data with varied graph structures (50 nodes), different connectivity patterns, and 500 diverse samples containing initial/target temperatures and optimal trajectories. For broader evaluation, we tested our model on Cora and PubMed citation networks for node classification, assessing performance beyond controlled synthetic environments.

3.3 TRAINING PROCEDURE

Our training methodology comprised several components working in concert. We developed a composite loss function that balanced temperature prediction accuracy and control efficiency, formulated as

$$\mathcal{L} = \mathcal{L}_{\text{temp}} + \lambda \cdot \mathcal{L}_{\text{control}}$$

controls the trade-off between temperature matching and control effort. We employed the Adam optimizer with learning rate scheduling and early stopping to ensure efficient convergence while preventing overfitting. Weight decay regularization was applied to further improve generalization.

4 EXPERIMENTS - RESULTS

Our research evaluated the Input Convex Graph Recurrent Neural Network (ICGRNN) architecture and Input Convex Graph Convolutional Network(ICGCN) on two distinct types of tasks: optimal

control for heat diffusion and node classification on citation networks. The results demonstrate both the capabilities and limitations of our approach.

4.1 HEAT DIFFUSION OPTIMAL CONTROL

For the heat diffusion problem, our trained ICGRNN model achieved the following error metrics:

Table 1: Results of our experiment-1

Objective function	**Loss(Mean Squared Error)**
Temperature Trajectory	0.007244
Control Input	0.000101

These metrics indicate that our model achieved reasonable accuracy in predicting both temperature evolution and control strategies, with notably higher precision for control inputs than temperature trajectories. Figure 1 illustrates temperature trajectories for a representative node (Node 7), comparing target values (solid blue lines) with our model's predictions (dashed red lines).

As Figure 1 shows, our model successfully captures temperature evolution patterns, accurately

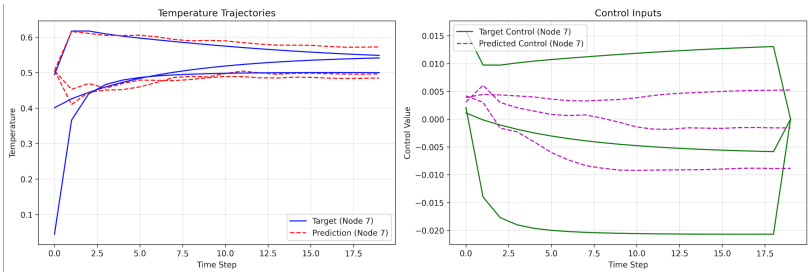

Figure 1: Comparison of target and predicted values for the heat diffusion optimal control problem.

predicting initial rises and stabilization despite minor overestimations in later timesteps. Figure 2 reveals more significant differences in control strategies: while target controls exhibit sharp transitions with larger magnitude variations (+0.014 to -0.020), our model adopts a more conservative approach with smaller changes (+0.005 to -0.010), yet still achieves effective temperature management.

This discrepancy in control strategy is noteworthy but not unexpected. The model appears to have developed a more conservative control approach that still achieves reasonable temperature tracking. This behavior may be influenced by our loss function weighting

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{temperature}} + \lambda \cdot \mathcal{L}_{\text{control}}$$

with $\lambda = 0.1$, which places greater emphasis on temperature accuracy than control precision.

4.2 NODE CLASSIFICATION BENCHMARKS

To assess our architecture's broader applicability, we evaluated the ICGCN variant (without the recurrent component) on standard graph benchmarks which utilize the standard Graph Convolutional Network(GCN)(Zhang et al., 2019) as illustrated in Table 2:

Table 2: Experimental results on CORA and PubMed

Dataset	**CORA**		**PubMed**	
Metric	Accuracy	Cross-entropy Loss	Accuracy	Cross-entropy Loss
ICGCN	0.78	0.68	0.73	0.77
Benchmark	0.81	0.57	0.79	0.63
Difference	–0.03	+0.11	–0.06	+0.14

5 CONCLUSION

Our work successfully integrates input convexity constraints with graph recurrent neural networks, creating a model particularly well-suited for spatiotemporal control problems on graphs. The ICGRNN architecture maintains theoretical guarantees while effectively modeling complex dynamics, as demonstrated by our heat diffusion experiments (Temperature MSE: 0.007244, Control MSE: 0.000101). While the model develops control strategies that differ from targets, it achieves the primary objective of temperature management across the graph.

Application to node classification reveals both versatility and limitations of the convexity approach. Our ICGCN variant achieves competitive results on CORA and PubMed (accuracies of 0.78 and 0.73), but the small performance gap compared to unconstrained GNNs suggests convexity constraints may limit representational capacity for tasks where convexity offers no inherent advantage.

REFERENCES

Brandon Amos, Lei Xu, and J. Zico Kolter. Input convex neural networks. *CoRR*, abs/1609.07152, 2016. URL http://arxiv.org/abs/1609.07152.

Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric, 2019. URL https://arxiv.org/abs/1903.02428.

Junyoung Park, Chihyeon Song, and Jinkyoo Park. Input convex graph neural networks: An application to optimal control and design optimization, 2022. URL https://openreview.net/forum?id=S2pNPZM-w-f.

Si Zhang, Hanghang Tong, Jiejun Xu, and Ross Maciejewski. Graph convolutional networks: a comprehensive review. *Computational Social Networks*, 6, 11 2019. doi: 10.1186/s40649-019-0069-y.