CSP572 Final Report: To Graph or not to Graph

Introduction

In recent years, graph-based machine learning techniques have gained popularity due to their ability to represent complex relationships and dependencies between data points. In this project, we will explore the use of graph completion techniques to predict the internal configurations of buildings using only external information. By leveraging the power of graph-based models, we aim to achieve more accurate predictions compared to traditional modeling techniques.

Moreover, this project will contribute to the broader field of machine learning by evaluating the effectiveness of graph-based models for node classification tasks. Node classification refers to the problem of assigning labels to nodes in a graph based on their attributes and the relationships between them. By investigating the performance of graph-based models in comparison to traditional machine learning models, we can gain insights into the strengths and weaknesses of different techniques for node classification tasks.

Finally, the ability to predict the internal layout of buildings using only external information can have significant benefits for firefighters, special forces, and SWAT teams. One of the primary advantages is improved situational awareness, which can provide a better understanding of the building's layout, potential hazards, and areas of interest. This knowledge can help responders make better decisions and act more effectively in high-pressure situations, potentially improving the safety of everyone involved.

Literature Review

This literature review serves as the foundation for the research and helps to establish the significance, scope, and direction of the project. In this section, the existing literature on GNNs and floorplan classification will be reviewed, examining the major themes and findings, identifying gaps in knowledge, and discussing the implications for future research. The purpose of this literature review is to provide a comprehensive overview of the current state of research on GNNs and floorplan classification and to identify areas where further research is needed. By analyzing and synthesizing the existing literature, this review aims to contribute to the development of a more complete and nuanced understanding of the topic, informing the research question, methodology, and interpretation of findings.

CubiCasa5k

The Cubicasa 5k dataset is a widely used dataset in the field of computer vision and machine learning, especially for floor plan detection and room segmentation tasks. The dataset is a collection of 5,000 vector-based floor plans from real estate listings from the Finland region.

The Cubicasa 5k dataset's diverse collection of floor plans, which includes various types of buildings such as houses, apartments, and commercial buildings, makes it more representative of real-world scenarios. This diversity is essential in developing and evaluating machine learning models that can be applied to different types of buildings.

One of the significant advantages of the Cubicasa 5k dataset is the high quality and accuracy of its annotations. The floor plans in the dataset are accurately annotated for room labels, walls, doors, and windows. This high level of annotation is essential in training and evaluating machine learning models for floor plan detection and room segmentation tasks.

The dataset's quality and diversity make it a valuable resource for researchers and practitioners in the field of computer vision and machine learning. It is also valuable for creating benchmark datasets for different machine learning applications and evaluating other spatial analysis algorithms.

How Powerful are Graph Neural Networks?

Graph Neural Networks (GNNs) have gained significant attention in recent years as a powerful framework for learning representations of graph-structured data. The paper "How Powerful are Graph Neural Networks?" by Xu et al. (2019) provides a comprehensive analysis of the expressive power of GNNs, which has important implications for their use in various graph-based applications.

The paper begins with a formal definition of the expressive power of GNNs and proposes a theoretical framework for analyzing it. The authors show that GNNs are not universal approximators, which means that there exist graph functions that cannot be represented by GNNs. However, they also demonstrate that GNNs are powerful enough to represent a broad class of graph functions and can outperform other state-of-the-art methods on various graph-based tasks.

The authors also provide a thorough empirical evaluation of GNNs on several benchmark datasets and compare their performance with other graph-based methods. They show that GNNs can effectively capture the structural information of graphs and achieve state-of-the-art performance on tasks such as node classification, graph classification, and link prediction.

One of the key contributions of the paper is the proposed framework for analyzing the expressive power of GNNs. The framework is based on the notion of Weisfeiler-Lehman (WL) graph isomorphism test, which is a popular method for distinguishing non-isomorphic graphs.

The authors show that the number of iterations of the WL test is a lower bound on the expressive power of GNNs, and that increasing the number of iterations can increase the expressive power of GNNs.

Another significant contribution of the paper is the demonstration of the effectiveness of GNNs on several benchmark datasets. The authors show that GNNs can outperform other state-of-the-art graph-based methods, such as Graph Convolutional Networks (GCNs) and Graph Attention Networks (GATs), on several tasks.

Generalizing Floor Plans Using Graph Neural Networks

The paper proposes a GNN-based approach for generating diverse and realistic floor plans by learning from a dataset of existing floor plans. The authors encode each floor plan as a graph, where the nodes represent rooms and the edges represent the spatial relationships between rooms. The GNN is then trained on this graph representation to generate new floor plans that satisfy certain constraints, such as the number of rooms and the desired square footage.

One of the key contributions of the paper is the use of a conditional GNN, which allows the model to generate floor plans that satisfy specific user-defined constraints. The authors also propose a novel objective function that encourages the model to generate diverse and realistic floor plans, by penalizing similar or repetitive floor plans.

The authors evaluate their proposed method on a large dataset of floor plans and show that it can generate diverse and realistic floor plans that satisfy user-defined constraints. They also demonstrate the effectiveness of their method in a user study, where participants rated the generated floor plans as more diverse and realistic compared to other state-of-the-art methods.

Overall, the paper provides a promising approach for generating diverse and realistic floor plans using GNNs, which can be customized to specific user needs and preferences. The proposed method has potential applications in architecture, urban planning, and interior design. The paper also highlights the potential of GNNs in generative modeling tasks, which is an important area of research in machine learning.

Methodology

The methodology of this project involves developing and evaluating machine learning models for predicting room types in floorplans. The overall approach consists of several key steps, including data preprocessing, feature extraction, model training, and evaluation.

The first step is to preprocess the data by removing any floor plans that contain undefined rooms. This is important to ensure that the models are trained on high-quality data and can

generalize well to new floor plans. The OpenCV and shapely libraries will be used to identify rooms as well as the icons in each room. Using this information, a classification model will be created for the sole purpose of identifying unidentified rooms, thereby cleaning the dataset and making more of the floorplans useful for training.

After preprocessing the data, the next step is to extract relevant features from the floor plans. For this project, node-level features such as area, and number of neighboring rooms will be used to create graph representations of each floorplan. These graph representations will be used as input to the machine learning models.

Two types of machine learning models will be developed and evaluated in this project: a non-graph based classification model and a graph neural network (GNN) based model. The non-graph based classification model will use traditional machine learning algorithms such as random forest classifier, logistic regression, and K-nearest neighbors, trained on the extracted features to predict room types. The GNN-based model, on the other hand, will use the graph structure of the floorplan to predict room types.

The performance of each model will be evaluated using metrics such as accuracy, F1 score, and precision-recall. To test the generalization ability of each model, a hold-out test set will be used to evaluate their performance on unseen data.

Overall, this methodology aims to develop and evaluate machine learning models for predicting room types in floorplans. By comparing the performance of non-graph based and GNN-based models, valuable insights into the strengths and limitations of each approach can be gained, which can inform future research in this area.

Graph Neural Network Architecture

The chosen GNN architecture for this task is a Graph Convolutional Network (GCN). The GCN is a popular GNN model that can capture the underlying structure of the graph data while considering the node features. The GCN model used in this project comprises two layers:

Input layer (GCNConv): The first layer is a graph convolutional layer that takes the input features (Relative Area and Number of neighboring rooms) and maps them to a 16-dimensional hidden representation space. The graph convolutional layer operates by aggregating feature information from each node's neighbors, enabling the model to learn spatial relationships between nodes.

Output layer (GCNConv): The second layer is another graph convolutional layer that takes the 16-dimensional hidden representation and maps it to the output logits with a dimension equal to the number of room classes. A log-softmax activation function is applied to the logits to generate probability distributions over the room classes for each node.

Model Implementation:

The model was implemented using the PyTorch and PyTorch Geometric (PyG) libraries. PyTorch Geometric provides efficient and convenient tools for working with graph data in deep learning, including GCNConv layers and DataLoader utility for batching graph samples. The following steps were followed during the model implementation:

Data preparation: The dataset was split into training, validation, and testing sets. The room type labels were converted to numerical indices using a predefined list of room classes. The floorplan data was processed and converted into PyTorch Geometric Data objects for compatibility with PyG's DataLoader.

Model definition: The GCN model was defined as a subclass of the torch.nn.Module class. The __init__ method initializes the graph convolutional layers, and the forward method defines the forward pass of the model, including the application of ReLU activation and dropout between the layers.

Training loop: The model was trained using a standard training loop, iterating over mini-batches of data using the PyG DataLoader. The loss function used for training was the negative log-likelihood loss (NLLLoss), which is suitable for multi-class classification tasks. The loss was backpropagated, and the model parameters were updated using the Adam optimizer.

Hyperparameter Choices:

Several hyperparameters were chosen during the model development process:

Hidden layer size: A 16-dimensional hidden layer was chosen as a trade-off between model complexity and computational efficiency. This choice provides the model with enough capacity to learn meaningful representations while limiting the risk of overfitting and reducing training time.

Dropout rate: A dropout rate of 0.5 was applied between the graph convolutional layers to prevent overfitting and encourage the model to learn more robust representations. The dropout rate is a common regularization technique in deep learning and helps improve generalization performance.

Learning rate and weight decay: The learning rate (0.01) and weight decay (5e-4) were chosen for the Adam optimizer. These hyperparameters control the step size during optimization and the amount of L2 regularization applied to the model weights, respectively. The chosen values are within typical ranges used in deep learning and were found to provide stable convergence during training.

Training epochs: The model was trained for 200 epochs, which provided sufficient time for the model to converge to a low training loss while avoiding overfitting.

Results

Precision, recall, and F1-score are essential performance metrics used to evaluate classification models. Here's a brief explanation of each:

Precision: Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. It indicates the proportion of positive identifications that were actually correct. A high precision indicates a low false-positive rate.

Recall: Recall (also known as sensitivity or true positive rate) is the ratio of correctly predicted positive observations to all the observations in the actual class. It indicates the proportion of actual positives that were identified correctly. A high recall indicates a low false-negative rate.

F1-score: The F1-score is the harmonic mean of precision and recall. It ranges between 0 (worst) and 1 (best), providing a single metric that balances both precision and recall. The F1-score is particularly useful when dealing with imbalanced datasets, as it considers both false positives and false negatives.

Comparing the three models based on these metrics:

Graph Model:
This model demonstrated the best overall performance among the three models. It had the highest accuracy (40.74%), precision (weighted avg: 0.38), recall (weighted avg: 0.41), and F1-score (weighted avg: 0.38). This suggests that the Graph Model was better at correctly classifying room types and had a better balance between false positives and false negatives compared to the other two models.

Logistic Regression Model:
The Logistic Regression Model performed moderately, with an accuracy of 31.28%, weighted avg precision of 0.25, weighted avg recall of 0.31, and weighted avg F1-score of 0.26. While it performed better than the Random Forest Classifier, it was less accurate and had lower precision, recall, and F1-scores compared to the Graph Model. This indicates that the Logistic Regression Model had a higher rate of both false positives and false negatives compared to the Graph Model.

Random Forest Classifier:
This model had the lowest performance among the three models, with an accuracy of 25.51%, weighted avg precision of 0.26, weighted avg recall of 0.26, and weighted avg F1-score of 0.25. Its low precision, recall, and F1-scores suggest that the Random Forest Classifier was more prone to false positives and false negatives than the other two models.

In conclusion, the Graph Model outperformed the Logistic Regression Model and the Random Forest Classifier in terms of precision, recall, and F1-score. It was better at classifying room types and had a better balance between false positives and false negatives. However, it is essential to note that all models could benefit from improvements in data collection, feature engineering, model selection, and hyperparameter tuning to enhance their performance.

Discussion

There could be several reasons why the Graph Model outperformed the Logistic Regression Model and the Random Forest Classifier in this particular classification task. Some of these reasons may include:

1. Suitability for the task: The Graph Model might be more suitable for this specific problem due to its ability to capture complex relationships and dependencies between features. This could enable the Graph Model to better differentiate between room types compared to the Logistic Regression Model, which primarily assumes a linear relationship between features and class labels, and the Random Forest Classifier, which uses decision trees to make predictions.
2. Implicit feature extraction: Graph-based models can automatically discover relevant features by learning the underlying structure of the data. This can lead to better feature representations and, in turn, improved classification performance. In contrast, Logistic Regression and Random Forest Classifier models rely on the given features and might not be as effective in identifying discriminative features.
3. Robustness to noise: Graph Models can be more robust to noise and outliers in the data, as they model the relationships between data points in a more holistic manner. In contrast, Logistic Regression and Random Forest Classifier models might be more sensitive to noise and outliers, which could affect their performance.
4. Regularization: Graph Models can inherently include regularization, preventing overfitting and improving generalization to new data points. Regularization helps the model strike a balance between fitting the training data too closely and finding a more generalized solution. In contrast, Logistic Regression and Random Forest Classifier models may require additional tuning or regularization techniques to achieve similar effects.
5. Hyperparameters: It is possible that the Graph Model's specific hyperparameters were better for this specific classification problem, leading to improved performance. In contrast, the Logistic Regression and Random Forest Classifier models might not have had their hyperparameters tuned optimally, potentially limiting their performance.

It is essential to note that these reasons are not mutually exclusive, and it is possible that a combination of these factors contributed to the Graph Model's superior performance in this specific classification task. To further understand and validate the reasons for this performance difference, future research consider conducting additional experiments, such as testing different model architectures, performing feature engineering, or exploring alternative data representations.

Model Maintenance, Updates, and Future Work

Model Maintenance and Updates

To ensure the continued effectiveness of the GCN-based room type classification model, it is essential to establish a plan for regular maintenance and updates. The following steps will help maintain the model's performance and adapt it to any changes in the data distribution:

- Monitoring performance: Continuously monitor the model's performance on a held-out validation set to detect any decline in accuracy or other performance metrics. This monitoring will enable the identification of potential issues that may require model updates or retraining.
- Updating the dataset: As new floorplan data becomes available or if the characteristics of the data change, update the dataset used to train the model. This process will ensure that the model remains up-to-date and can generalize well to new data.
- Retraining the model: Periodically retrain the model using the updated dataset to maintain optimal performance. Retraining should be done using the same model architecture and hyperparameters, adjusting them only if necessary based on observed performance changes.
- Version control: Keep track of different versions of the model, their performance metrics, and associated datasets. This practice will enable easy rollback to previous versions if required and allow for easier identification of the best-performing model.

Improvements and Future Work:

To enhance the model's accuracy and usefulness, consider the following improvements and future work:

- Hyperparameter optimization: Conduct a more comprehensive hyperparameter search using techniques such as grid search, random search, or Bayesian optimization to identify the best combination of hyperparameters that maximize the model's performance.
- Alternative GNN architectures: Experiment with different GNN architectures, such as GraphSAGE, GAT, or Graph Isomorphism Networks, to determine if any of these models can yield better performance on the room type classification task.
- Incorporate additional features: Explore the inclusion of additional node features or edge features to provide the model with more information that may help improve classification accuracy.
- Multi-task learning: Investigate the possibility of training the model to perform multiple related tasks simultaneously, such as predicting room function and room size. This approach may improve the model's overall performance by leveraging shared information between tasks.

- Transfer learning: If similar datasets are available, consider using transfer learning techniques to fine-tune the model on the target dataset. This approach can help improve performance by leveraging pre-trained models that have already learned useful features from related data.
- Ensemble methods: Experiment with ensemble techniques such as stacking, bagging, or boosting to combine the predictions of multiple models and potentially improve overall classification accuracy.

By following the outlined maintenance and update plan and considering the suggested improvements and future work, the GCN-based room type classification model will remain effective and relevant in addressing the problem at hand.

Conclusion

In conclusion, the Graph Convolutional Network (GCN) based room type classification model has showcased its potential in accurately identifying room types from floorplan data. This report has provided a detailed account of the model's architecture, hyperparameter choices, and implementation, offering a comprehensive understanding of the current work and serving as a foundation for future research and development.

As the field of architectural design and analysis continues to evolve, it is crucial to maintain and update the model to ensure its relevance and effectiveness. This report has presented a systematic plan for model maintenance, updates, and future work that encompasses essential aspects such as performance monitoring, dataset updates, model retraining, and version control. By adhering to this plan, the model can be fine-tuned and enhanced over time to address the room type classification problem more effectively.

In addition to maintenance and updates, the report has outlined a variety of potential improvements and areas for future work, including hyperparameter optimization, alternative GNN architectures, additional feature incorporation, multi-task learning, transfer learning, and ensemble methods. These avenues for improvement offer exciting prospects for advancing the model's performance and expanding its scope of applicability.

In summary, the GCN-based room type classification model represents a powerful and versatile tool for understanding and analyzing floorplan data. The model's performance demonstrates its potential to contribute significantly to the field of architectural design and analysis, providing valuable insights into room type distributions and relationships within floorplans. By committing to regular maintenance, updates, and exploring potential improvements, the model can continue to be a robust and reliable solution for room type classification tasks.

The successful development and implementation of the GCN-based room type classification model is a testament to the power of graph-based machine learning techniques in tackling complex problems with structured data.

Sources

C. P. Simonsen, F. M. Thiesson, M. P. Philipsen and T. B. Moeslund, "Generalizing Floor Plans Using Graph Neural Networks," 2021 IEEE International Conference on Image Processing (ICIP), Anchorage, AK, USA, 2021, pp. 654-658, doi: 10.1109/ICIP42928.2021.9506514.

Kalervo, Ahti, et al. “Cubicasa5k: A Dataset and an Improved Multi-Task Model for Floorplan Image Analysis.” ArXiv.org, 3 Apr. 2019, https://arxiv.org/abs/1904.01920.

Xu, Keyulu, et al. HOW POWERFUL ARE GRAPH NEURAL NETWORKS? 2019, https://cs.stanford.edu/people/jure/pubs/gin-iclr19.pdf.