

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

PEDRO HENRIQUE DA COSTA AVELAR

Graph Representation Learning for Images

Thesis presented in partial fulfillment of the
requirements for the degree of Doctor of
Computer Science

Advisor: Prof. Dr. Luis Lamb
Co-advisor: Prof. Dr. Anderson Rocha Tavares

Porto Alegre
Jan 2025

CIP — CATALOGING-IN-PUBLICATION

da Costa Avelar, Pedro Henrique

Graph Representation Learning for Images / Pedro Henrique da Costa Avelar. – Porto Alegre: PPGC da UFRGS, 2025.

90 f.: il.

Thesis (Ph.D.) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2025. Advisor: Luis Lamb; Co-advisor: Anderson Rocha Tavares.

1. Graph neural networks. 2. Image classification. 3. Image representation. 4. Superpixels. 5. Attention. I. Lamb, Luis. II. Tavares, Anderson Rocha. III. Título.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Reitora: Prof^ª. Dr^ª. Marcia Cristina Bernardes Barbosa

Vice-Reitor: Prof. Dr. Pedro de Almeida Costa

Pró-Reitora de Pós-Graduação: Prof^ª. Dr^ª. Claudia Wasserman

Diretor do Instituto de Informática: Prof. Dr. Luciano Paschoal Gaspary

Coordenador do PPGC: Prof. Dr. Gabriel Luca Nazar

Bibliotecário-chefe do Instituto de Informática: Dr. Alexsander Borges Ribeiro

“Once men turned their thinking over to machines in the hope that this would set them free. But that only permitted other men with machines to enslave them.”

— FRANK HERBERT, DUNE

ACKNOWLEDGEMENTS

I would like to thank my supervisors Luís da Cunha **Lamb** and Anderson Rocha **Tavares** for all their help and support during my PhD: without you I wouldn't be able to complete this and I will forever be grateful. I'd like to thank all my friends and colleagues which helped develop my work and ideas, and helped me along the way. I'd like to also give a special thanks to: Jessica Reinheimer de **Lima** for her support during the first years of this PhD; Cláudio R. **Jung** for the initial spark for Superpixel image classification; Arthur **Bizzi** for helping me proofread the Superpixel Transformer Work; Syed Hesham **Syed Ariff** for helping me get a grounding on the current Computer Vision literature; Dylan Alexander **Slavin Hillier** for help with setting up some of the HPC used here and for providing helpful discussions on transformers; Wai Yee (Nicola) **Wong** for her support for me during the nights working on this thesis; and Henrique José **Avelar** for helping proofread this thesis.

I'd also like to thank the members of my thesis evaluation panel, Ricardo Matsumura de **Araujo**, Álvaro Freitas **Moreira**, Joel Luís **Carbonera** and Leandro Resende **Mattioli**, both for agreeing to be members with such short notice, as well as for their discussion and suggestions during the defense. I tried, to the best of my resources, to follow all the suggestions I could, and they greatly enriched this document and my understanding of the knowledge I generated through this project.

This work was partially funded by CNPQ, CAPES and FAPERGS, which I thank greatly for their financial support. Some experiments in this work used the PCAD infrastructure, <<http://gppd-hpc.inf.ufrgs.br>>, at INF/UFRGS (finance codes FAPERGS 16/2551-0000348-8 and Petrobras TC 5900.0111175.19.9, 2020/00182-5), and I'd like to thank both the funding that allowed UFRGS to build PCAD infrastructure as well as Lucas Mello **Schnorr** for providing the computational resources from these finance codes to PCAD for the UFRGS community to use.

ABSTRACT

While a plurality of developments in AI still focus largely on Convolutional Neural Networks and Transformers, there has been frequent calls in the community towards building models with symbolic and relational processing capabilities. In our study we first reframe image classification as graph classification and then we attempt to realign the graph-based image classification back into the main image classification literature through the use of attentional models.

To use graph-based attention models for image classification we first use a superpixel algorithm to segment images, which we then use to build a graph whose nodes' information is based on the superpixel positions and average colours, and whose edges are based on the superpixel adjacencies, forming a Superpixel Image Classification (SIC) instance, to which then Graph Neural Networks can be applied for classification, for which we use a Graph Attention Network (GAT). Our Superpixel Image Classification with Graph Attention Networks (SICGAT) model was tested in several common image-based benchmarks and achieved a higher classification accuracy than the baselines available at the time, even though we restricted our model to have access to less information and, thus, a harder problem. This was done by both reducing the number of superpixels available to the model as well as by using the superpixel adjacencies, which provides less edges per node than previous benchmarks, which both reduces information and constrains information flow within the model.

We then propose a novel framework – SuperPixel Transformer (SPT) – which generalises both our SICGAT model and the Vision Transformer (ViT) model, and allows us to keep the full superpixel information while still working with superpixel graphs. Our results show that using superpixel patches in our SPT framework is comparable to using regular square grid patches normally used in ViTs, and that different connectivity patterns than the fully connected one used in the ViT might actually improve results from using square-grid-based superpixels.

We conclude by providing discussion on possible future applications of our work as well as some directions for future improvements, some of which have already been explored in the mainstream ViT literature, but might not have been explored in the SIC literature.

Keywords: Graph neural networks. Image classification. Image representation. Superpixels. Attention.

Aprendizado de Representações em Grafos aplicado a Imagens

RESUMO

Com uma variedade de desenvolvimentos em IA ainda focando em Redes Neurais Convolucionais e *Transformers*, existem vozes persistentes na comunidade para que se construam modelos com capacidades relacionais e simbólicas. Neste estudo primeiramente repensamos classificação de imagens como classificação de grafos para então tentarmos realinhar classificação de imagens baseadas em grafos de volta com a literatura de classificação de imagens através do uso de modelos com atenção.

A fim de utilizar modelos com atenção baseados em grafos para classificação de imagens, primeiramente utilizamos um algoritmo de *superpixel* para segmentar a imagem, que é transformada em um grafo onde a informação de cada nodo contém a média das posições e cores de cada *superpixel*, e cujas arestas são baseadas na adjacência entre cada par de *superpixel*, formando uma instância de *Superpixel Image Classification (SIC)*, na qual então uma rede grafo-neural pode ser aplicada para realizar a classificação, dentre as quais escolhemos a rede grafo-atencional – *Graph Attention Network (GAT)*. Nosso modelo *Superpixel Image Classification with Graph Attention Networks (SICGAT)* foi testado em vários *benchmarks* de imagens comuns, onde conseguiu uma acurácia de classificação maior que *baselines* disponíveis na época, a despeito do fato de que restringimos a informação disponível para o nosso modelo e, portanto, resolvemos um problema mais difícil. Isto foi feito tanto reduzindo o número de *superpixels* disponíveis para o modelo quanto usando as adjacências entre os *superpixels* para construir o grafo, o que produz menos arestas por nodos que *benchmarks* anteriores, assim não só reduzindo a informação disponível para o modelo quanto restringindo o fluxo de informação dentro do modelo.

Então, propomos uma nova *framework* – *SuperPixel Transformer (SPT)* – que generaliza tanto o nosso modelo SICGAT quanto *Vision Transformers (ViT)*, e permite que mantenhamos toda a informação disponível em um *superpixel* enquanto o modelo ainda usa grafos de *superpixels*. Nossos resultados mostram que usar fragmentos baseados em *superpixel* tem performance comparável com quando se utiliza fragmentos quadrados regulares, comumente utilizados nos ViTs. Também mostramos que padrões de conectividade além de fragmentos completamente conectados, também comumente utilizados nos ViTs, podem melhorar os resultados quando comparados quando utilizados até com *superpixels* de fragmentos quadrados regulares.

Por fim, concluímos a tese discutindo possíveis usos do nosso trabalho assim como direções para melhorias e trabalhos futuros, algumas das quais já foram exploradas na literatura de ViTs convencionais, mas talvez não foram ainda exploradas na literatura de SIC.

Palavras-chave: Redes grafo-neurais. Classificação de imagens. Representação de imagens. Superpixels. Atenção.

LIST OF FIGURES

Figure 2.1 An artificial neuron which receives inputs x_j and outputs h_j depending on the inputs, their associated weights $w_{i,j}$ and its activation function σ	22
Figure 2.2 A diagram comparing a fully connected layer to a small convolutional kernel	24
Figure 2.3 Example of how a GNN layer might be computed using gather and scatter operations	28
Figure 2.4 A diagram showing the transformer architecture	32
Figure 4.1 The procedure for converting an image into a RAG.....	39
Figure 4.2 Training Curve for the Street View House Numbers dataset.....	44
Figure 4.3 Examples for the information loss in the RAG generation procedure.....	45
Figure 5.1 The ViT and SICGAT architectures compared.....	50
Figure 5.2 The ViT and GAT encoders compared.	50
Figure 5.3 Our proposed Superpixel Transformer (SPT) architecture.....	53
Figure 5.4 Plots highlighting the Validation Accuracy for the ablation on the FashionMNIST dataset.	62
Figure 5.5 Plots highlighting the Validation Accuracy for the ablation on the CIFAR10 dataset.	63
Figure B.1 Regression plots between number of layers and training+testing time.	86
Figura C.1 Resumo Expandido – Exemplos da perda de informação no procedimento de geração do grafo de entrada do SICGAT.....	89
Figura C.2 Resumo Expandido: A arquitetura proposta para o nosso framework <i>Superpixel Transformer (SPT)</i>	90

LIST OF TABLES

Table 4.1 Test accuracy for the tested models on MNIST and FashionMNIST, processed as RAGs with approximately 75 nodes.....	43
Table 5.1 Model configurations used for ViT and SPT models defined by Steiner et al. (2022).	57
Table 5.2 The description of the datasets used for the Superpixel Transformer model. .	57
Table 5.3 Results of the Superpixel Transformer Model	58
Table 5.4 Ablation Study Results for the Superpixel Transformer Model.....	59
Table B.1 Linear model intercept for the time taken in the SPT ablation study.	85

LIST OF ABBREVIATIONS AND ACRONYMS

BERT	Bidirectional Encoder Representations from Transformers
BP	BackPropagation
CNN	Convolutional Neural Network
FC	Fully Connected (layer)
FCG	Fully Connected Graph
GAT	Graph Attention Network
GCN	Graph Convolutional Network
GNN	Graph Neural Network
KNN	K-Nearest Neighbours
MHA	Multi-Head Attention
MLP	MultiLayer Perceptron
MPNN	Message-Passing Neural Network
RAG	Region-Adjacency Graph
RGB	Red Green and Blue
SAT	boolean SATisfiability (problem)
SIC	Superpixel Image Classification
SICGAT	Superpixel Image Classification with Graph Attention Networks
SLIC	Simple Linear Iterative Clustering
SPT	SuperPixel Transformer
ViT	Vision Transformer

CONTENTS

1 INTRODUCTION	13
1.1 Objectives	14
1.1.1 Main Objective	14
1.1.2 Specific Objectives	14
1.2 Contributions	15
1.2.1 Graph Neural Networks Meet Neural-Symbolic Computing: A Survey and Perspective	16
1.2.2 Neural-Symbolic Relational Reasoning on Graph Models: Effective Link Inference and Computation from Knowledge Bases	16
1.2.3 Superpixel Image Classification with Graph Attention Networks*	17
1.2.4 Discrete and Continuous Deep Residual Learning over Graphs	17
1.2.5 Measuring Ethics in AI with AI: A Methodology and Dataset Construction	18
1.2.6 A Bayesian predictive analytics model for improving long range epidemic forecasting during an infection wave	18
1.2.7 Solving the kidney exchange problem via graph neural networks with no supervision	19
1.2.8 On the Evolution of A.I. and Machine Learning: Towards a Meta-level Measuring and Understanding Impact, Influence, and Leadership at Premier A.I. Conferences	19
1.2.9 Is an Image Also Worth $16 \times 16 = 256$ Superpixels? A framework for attentional image classification*	20
2 BACKGROUND	21
2.1 Deep Learning	21
2.1.1 Fully Connected Neural Networks and Backpropagation	21
2.1.2 Convolutional Neural Networks	23
2.1.3 Graph Neural Networks	24
2.1.3.1 Graph Batching and Dense Adjacency Matrices	26
2.1.3.2 Node-Edge and Edge-Node Adjacency Matrices	26
2.1.3.3 Applications of Graph Neural Networks	27
2.1.3.4 Modern approaches	28
2.1.3.5 The Message-Passing view of Graph Neural Networks	28
2.1.4 Transformers and Attention	29
2.1.5 Graph Attention Networks	33
2.2 Superpixel Algorithms	34
3 RELATED WORK	37
4 SUPERPIXEL IMAGE CLASSIFICATION WITH GRAPH ATTENTION NETWORKS	39
4.1 Superpixel images as Region-Adjacency Graphs	39
4.2 Graph Attention Networks	40
4.3 Benchmark Datasets	41
4.4 Results	42
4.4.1 MNIST and FashionMNIST	42
4.4.2 Street View House Numbers	43
4.4.3 CIFAR-10	44
4.5 Discussion	45
5 FROM GATS TO VITS AND BACK: SUPERPIXEL TRANSFORMERS	47
5.1 Related Work	47

5.2 SuperPixel Transformers (SPT): A General Framework for Attentional Learning in Images	49
5.2.1 Key differences between Superpixel Image Classification and Vision Transformers	49
5.2.2 The Superpixel Transformer Model.....	52
5.3 Experimental Details	56
5.4 Results	57
5.5 Discussion	60
6 CONCLUSION	64
REFERENCES.....	66
APPENDIX A — THE SLICO ALGORITHM	80
APPENDIX B — EXTRA MATERIAL FROM THE SUPERPIXEL TRANSFORMER MODEL EXPERIMENTS	83
APÊNDICE C — RESUMO EXPANDIDO	87
C.1 Introdução.....	87
C.2 Motivação e Contribuições	87
C.3 Resultados Principais.....	88

1 INTRODUCTION

In the past decade, deep learning quickly became the leading machine learning techniques, replacing hand-engineered solutions with data-driven ones in many areas (Lecun; Bengio; Hinton, 2015). It has been argued that this success is at least partly due to what are called “relational inductive biases” in the models (Battaglia et al., 2018), which are biases that impose constraints on relationships and interactions among entities in a learning process, such as the choice of prior in bayesian learning, a regularization term in an optimisation problem, etc. Many modern models still using building blocks based on much earlier precepts: modern Convolutional Neural Networks (CNNs) still work using convolutional layers very similar to those proposed in before the 2000s (Fukushima, 1980; Lecun et al., 1998), with adjustments to allow for scalability and deeper networks (Krizhevsky; Sutskever; Hinton, 2012; Simonyan; Zisserman, 2015; Liu et al., 2022). Modern graph-based models are also very much akin to their earlier iterations (Scarselli et al., 2009), and instead focused on applying techniques from different areas into graph neural networks themselves, such as adding attention to graph aggregation, different pooling strategies, loss strategies and regularisations (Monti et al., 2017; Kipf; Welling, 2017; Gilmer et al., 2017; Velickovic et al., 2018; Veličković et al., 2019).

Despite earlier calls for a focus on symbolic and relational reasoning capabilities in AI (Battaglia et al., 2018; Lamb et al., 2020), recent models have still followed similar trends of increasing the number of parameters and increasing training data towards achieving “zero-shot” or “few-shot” performance (Brown et al., 2020; Ramesh et al., 2021; Radford et al., 2021; Kirillov et al., 2023). However, given recent predictions that the end of zero-shot scalability of data cannot be achieved without exponentially increasing data (Udandaraao et al., 2024), there is value in building models that can handle symbolic computation well, of which Graph Neural Networks seem to be a prime example. Due to the intrinsically symbolic and relational nature of their input data, graph neural networks have the potential to be extremely flexible and help drive further improvements in AI that do not require scaling parameters or training data.

In this thesis we explore the application of Attentional Graph Neural Networks to image processing and the application of other techniques in the Graph Domain. **In Chapter 1** (this chapter) we introduce and contextualise this thesis. **In Chapter 2** we will provide the necessary background for the thesis. **In Chapter 3** we provide a background and overview of related work for Superpixel Image Classification. **In Chapter 4**, we

analyse the performance the Graph Attention Network (GAT) architecture for Superpixel Image Classification (SIC) (Avelar et al., 2020), we do so by extracting the superpixels from an image, then building a graph using the superpixels' neighbourhoods and colour/position information, to which then we apply our model for Superpixel Image Classification with Graph Attention Networks (SICGAT). **In Chapter 5**, we realign the work shown in Chapter 4 with current literature by providing a framework that generalises both Superpixel Image Classification with Graph Attention Networks (SICGAT) and Vision Transformers (ViT) (Avelar; Tavares; Lamb, 2024): SuperPixel Transformers (SPT). We show how this framework allows Graph-based models to close the gap with a strong vision baseline. Finally, we'll provide concluding remarks and comment on future work **in Chapter 6**. We believe that the advancement of superpixel image classification could be a stepping-stone to leveraging the high availability of planar images into transferring image models to spherical images, which have much more scarce resources.

1.1 Objectives

The next sections present the main and specific objectives of our research.

1.1.1 Main Objective

Our main objective is to investigate attentional Graph Neural Networks as a representation learning method for image classification – with a specific focus on representing images through superpixels – and to transfer some of the techniques applied in computer vision to the domain of graph processing.

1.1.2 Specific Objectives

1. To provide a clear methodology for implementing attentional Graph Neural Networks in images, using superpixels as a medium of representation of images as graphs
2. To evaluate attentional Graph Neural Network architectures and categorise them with respects to their representational power using classification accuracy as a sur-

rogate for representational power.

3. To re-align graph-based image processing with current computer vision techniques such as Vision Transformers.

1.2 Contributions

The main contributions of this thesis are, to the best of our knowledge: (1) the first application of a purely attentional (and thus, transformer-like) neural network to image classification, which we performed by transforming the applying a Graph Attention Network (GAT) to a Superpixel-based graph, which we called Superpixel Image Classification with Graph Attention Networks (SICGAT); and (2) the first framework which bridges the gap between superpixel-graph based image classification (SIC) and the mainstream image classification, which we do by constructing our SuperPixel Transformer (SPT) framework. It is important to note, however, that this thesis has been written with a 4-year gap between the initial development of its fundamental basis in 2020, where we first proposed SICGAT (Avelar et al., 2020), and its conclusion in 2024 with the SPT framework (Avelar; Tavares; Lamb, 2024). During these four year, significant advances happened in the literature, including the development of Vision Transformers (Dosovitskiy et al., 2021), which was one of the motivating factors for us to generalise SICGAT into our SPT framework, which we do by showing the similarities between the SICGAT and the ViT architectures, unifying both fields.

The student's contributions, if measured in terms of scientific papers, are described below in order of publication. We include the abstracts of these papers to make it easier to understand the context of the contributions, and mark the two abovementioned papers which are considered to be the main topic of this thesis with a star. The choice of not including these other contributions into the thesis was to provide a more cohesive thesis with a definite contribution, since the projects that the PhD student took part in were extremely diverse in topics, such as Neural-Symbolic perspectives in AI (Lamb et al., 2020), Link Prediction (Lemos et al., 2020), Superpixel Image Classification* (Avelar et al., 2020), the application of continuous residual layers in the context of graphs (Avelar et al., 2021), Ethics in AI (Avelar; Audibert; Lamb, 2022), Epidemic Forecasting (Avelar et al., 2022), using graph neural networks for NP-complete problems (Pimenta, 2023), meta-research (Audibert et al., 2023), and finally circling back to Image Classification*

(Avelar; Tavares; Lamb, 2024) once more.

1.2.1 Graph Neural Networks Meet Neural-Symbolic Computing: A Survey and Perspective

Lamb et al.

Neural-symbolic computing has now become the subject of interest of both academic and industry research laboratories. Graph Neural Networks (GNNs) have been widely used in relational and symbolic domains, with widespread application of GNNs in combinatorial optimization, constraint satisfaction, relational reasoning and other scientific domains. The need for improved explainability, interpretability and trust of AI systems in general demands principled methodologies, as suggested by neural-symbolic computing. In this paper, we review the state-of-the-art on the use of GNNs as a model of neural-symbolic computing. This includes the application of GNNs in several domains as well as their relationship to current developments in neural-symbolic computing.

(Lamb et al., 2020)

The student was the co-author of the paper, published at the IJCAI-PRICAI-20 (A1, Qualis 2017-2020) (Lamb et al., 2020).

1.2.2 Neural-Symbolic Relational Reasoning on Graph Models: Effective Link Inference and Computation from Knowledge Bases

The recent developments and growing interest in neural-symbolic models has shown that hybrid approaches can offer richer models for Artificial Intelligence. The integration of effective relational learning and reasoning methods is one of the key challenges in this direction, as neural learning and symbolic reasoning offer complementary characteristics that can benefit the development of AI systems. Relational labelling or link prediction on knowledge graphs has become one of the main problems in deep learning-based natural language processing research. Moreover, other fields which make use of neural-symbolic techniques may also benefit from such research endeavours. There have been several efforts towards the identification of missing facts from existing ones in knowledge graphs. Two lines of research try and predict knowledge relations between two entities by considering all known facts connecting them or several paths of facts connecting them. We propose a neural-symbolic graph neural network which applies learning over all the paths by feeding the model with the embedding of the minimal subset of the knowledge graph containing such paths. By learning to produce representations for entities and facts corresponding to word embeddings, we show how the model can be trained end-to-end to decode these representations and infer relations between entities in a multitask approach. Our contribution is two-fold: a neural-symbolic methodology leverages the resolution of relational inference in large graphs, and we also demonstrate that such neural-symbolic model is shown more effective than path-based approaches.

(Lemos et al., 2020)

The student was the co-author of the paper published at the ICANN 2020 (A2, Qualis 2017-2020) (Lemos et al., 2020).

1.2.3 Superpixel Image Classification with Graph Attention Networks*

This paper presents a methodology for image classification using Graph Neural Network (GNN) models. We transform the input images into region adjacency graphs (RAGs), in which regions are superpixels and edges connect neighboring superpixels. Our experiments suggest that Graph Attention Networks (GATs), which combine graph convolutions with self-attention mechanisms, outperforms other GNN models. Although raw image classifiers perform better than GATs due to information loss during the RAG generation, our methodology opens an interesting avenue of research on deep learning beyond rectangular-gridded images, such as 360-degree field of view panoramas. Traditional convolutional kernels of current state-of-the-art methods cannot handle panoramas, whereas the adapted superpixel algorithms and the resulting region adjacency graphs can naturally feed a GNN, without topology issues.

(Avelar et al., 2020)

The student was the 1st author of the paper published at the SIBGRAPI 2020 (A3, Qualis 2017-2020) (Avelar et al., 2020).

1.2.4 Discrete and Continuous Deep Residual Learning over Graphs

Avelar et al.

We propose the use of continuous residual modules for graph kernels in Graph Neural Networks. We show how both discrete and continuous residual layers allow for more robust training, being that continuous residual layers are applied by integrating through an Ordinary Differential Equation (ODE) solver to produce their output. We experimentally show that these modules achieve better results than the ones with non-residual modules when multiple layers are used, thus mitigating the low-pass filtering effect of Graph Convolutional Network-based models. Finally, we discuss the behaviour of discrete and continuous residual layers, pointing out possible domains where they could be useful by allowing more predictable behaviour under dynamic times of computation.

(Avelar et al., 2021)

The student was the 1st author of the paper accepted at the ICAART 2021 (A4, Qualis 2017-2020) (Avelar et al., 2021).

1.2.5 Measuring Ethics in AI with AI: A Methodology and Dataset Construction

Avelar; Audibert; Lamb

Recently, sound measures and metrics in Artificial Intelligence have become a focus of research and development in academia, government, and industry. Efforts towards measuring different phenomena have gained traction in the AI community, as illustrated by several influential field reports and policy documents. These metrics are designed to help decision-makers inform themselves about the fast-moving and impacting influences of key advances in Artificial Intelligence in general and Machine Learning in particular. In this paper, we propose to use such newfound capabilities of AI technologies to augment our AI measuring capabilities. We do so by training a model to classify publications related to ethical issues and concerns. Our methodology uses an expert, manually curated dataset as the training set and then evaluates an extensive collection of research papers. Finally, we highlight the implications of AI metrics, particularly their contribution towards developing trustful and fair AI-based tools and technologies.

(Avelar; Audibert; Lamb, 2022)

The student was the 1st author of the paper accepted at the BRACIS 2022 (A4, Qualis 2017-2020) and also published as a book chapter in Lecture Notes in Computer Science (B1, Qualis 2017-2020) (Avelar; Audibert; Lamb, 2022).

1.2.6 A Bayesian predictive analytics model for improving long range epidemic forecasting during an infection wave

Avelar et al.

Following the outbreak of the coronavirus epidemic in early 2020, municipalities, regional governments and policymakers worldwide had to plan their Non-Pharmaceutical Interventions (NPIs) amidst a scenario of great uncertainty. At this early stage of an epidemic, where no vaccine or medical treatment is in sight, algorithmic prediction can become a powerful tool to inform local policymaking. However, when we replicated one prominent epidemiological model to inform health authorities in a region in the south of Brazil, we found that this model relied too heavily on manually predetermined covariates and was too reactive to changes in data trends. Our four proposed models access data of both daily reported deaths and infections as well as take into account missing data (e.g., the under-reporting of cases) more explicitly, with two of the proposed versions also attempting to model the delay in test reporting. We simulated weekly forecasting of deaths from the period from 31/05/2020 until 31/01/2021, with first week data being used as a cold-start to the algorithm, after which we use a lighter variant of the model for faster forecasting. Because our models are significantly more proactive in identifying trend changes, this has improved forecasting, especially in long-range predictions and after the peak of an infection wave, as they were quicker to adapt to scenarios after these peaks in reported deaths. Assuming reported cases were under-reported greatly benefited the model in its stability, and modelling retroactively-added data (due to the “hot” nature of the data used) had a negligible impact on performance.

(Avelar et al., 2022)

The student was the 1st author of the paper published in Healthcare Analytics (Qualis unavailable) (Avelar et al., 2022).

1.2.7 Solving the kidney exchange problem via graph neural networks with no supervision

Pimenta; Avelar; Lamb

This paper introduces a new learning-based approach for approximately solving the Kidney-Exchange Problem (KEP), an NP-hard problem on graphs. The KEP consists of, given a pool of kidney donors and patients waiting for kidney donations, optimally selecting a set of donations to optimize the quantity and quality of transplants performed while respecting a set of constraints about the arrangement of these donations. The proposed technique consists of two major steps: the first is a Graph Neural Network (GNN) trained without supervision; the second is a deterministic non-learned search heuristic that uses the output of the GNN to find a valid solution. To allow for comparisons, we also implemented and tested an exact solution method using integer programming, two greedy search heuristics without the machine learning module, and the GNN alone without a heuristic. We analyze and compare the methods and conclude that the learning-based two-stage approach is the best solution quality, outputting approximate solutions on average 1.1 times more valuable than the ones from the deterministic heuristic alone.

(Pimenta; Avelar; Lamb, 2024)

The student (PHCA) was co-advisor to Pedro Foletto Pimenta's final year undergraduate project (Pimenta, 2023). The resulting work was published both as a conference (Pimenta; Avelar; Lamb, 2023) and journal paper (Pimenta; Avelar; Lamb, 2024).

The student (PHCA) was the co-author of the paper accepted at the 35th International Conference on Tools with Artificial Intelligence, Atlanta, USA (A3, Qualis 2017-2020) (Pimenta; Avelar; Lamb, 2023). The student (PHCA) was the co-author of the paper published in Neural Computing and Applications (A1, Qualis 2017-2020) (Pimenta; Avelar; Lamb, 2024).

1.2.8 On the Evolution of A.I. and Machine Learning: Towards a Meta-level Measuring and Understanding Impact, Influence, and Leadership at Premier A.I. Conferences

Artificial Intelligence is now recognized as a general-purpose technology with ample impact on human life. This work aims at understanding the evolution of AI and, in particular Machine learning, from the perspective of researchers'

contributions to the field. In order to do so, we present several measures allowing the analyses of AI and machine learning researchers' impact, influence, and leadership over the last decades. This work also contributes, to a certain extent, to shed new light on the history and evolution of AI by exploring the dynamics involved in the field's evolution by looking at papers published at the flagships AI and machine learning conferences since the first International Joint Conference on Artificial Intelligence (IJCAI) held in 1969. AI development and evolution have led to increasing research output, reflected in the number of articles published over the last sixty years. We construct comprehensive citation-collaboration and paper-author datasets and compute corresponding centrality measures to carry out our analyses. These analyses allow a better understanding of how AI has reached its current state of affairs in research. Throughout the process, we correlate these datasets with the work of the ACM Turing Award winners and the so-called two AI winters the field has gone through. We also look at self-citation trends and new authors' behaviors. Finally, we present a novel way to infer the country of affiliation of a paper from its organization. Therefore, this work provides a deep analysis of Artificial Intelligence history from information gathered and analysed from large technical venues datasets and suggests novel insights that can contribute to understanding and measuring AI's evolution.

The student was the co-author of the paper published at the Journal of Applied Logics (B1, Qualis 2017-2020) (Audibert et al., 2023).

1.2.9 Is an Image Also Worth $16 \times 16 = 256$ Superpixels? A framework for attentional image classification*

Superpixel-based image classification has traditionally leveraged graph neural networks (GNNs) for processing irregular image representations. Recent advancements in computer vision, driven by Vision Transformers (ViTs), have introduced new paradigms in self-attentional models, surpassing convolutional neural networks (CNNs) in various tasks. However, a synergistic connection between GNNs, superpixels, and transformers remains unexplored. In this work, we propose Superpixel Transformers (SPT), a novel framework that unifies superpixel-based image classification and ViTs. SPT generalizes the Superpixel Image Classification with Graph Attention Networks (SICGAT) model and ViT to support arbitrary superpixel-based chunking strategies, connectivity graphs, and positional encodings. We introduce refinements including a multidimensional sine-cosine positional encoding and an enriched patch data structure that fully incorporates superpixel shape and color information. By testing SPT across datasets such as CIFAR10, FashionMNIST, and Imagenette, with various superpixel generation and graph connectivity strategies, we demonstrate that SPT achieves superior performance compared to previous superpixel-based GNN methods and remains competitive with ViTs. Notably, our approach addresses the limitations of SICGAT, such as information loss during pixel aggregation, and shows how constrained graph connectivity can enhance ViT performance. SPT bridges the gap between superpixel-based and transformer models, opening avenues for cross-domain generalization and future innovations in hybrid attentional frameworks.

The student is the first author of the paper submitted to the Neural Computing and Applications Journal (A2, Qualis 2017-2020) (Avelar; Tavares; Lamb, 2024).

2 BACKGROUND

In this Chapter, we will provide the basic ideas necessary to understand how graph neural networks can be used as an alternative to convolutional neural networks in image processing, and also introduce the Transformer network which will be used later in the thesis. We already assume that the reader is familiar with neural networks/deep learning and is proficient with a modern deep learning framework and we will be using notation similar to what is used in them. Furthermore, we highly recommend the interested reader to be exposed to Schmidhuber (2015) for a good historical overview of deep learning, Goodfellow, Bengio and Courville (2016) for a more comprehensive introduction to deep learning, and Sanchez-Lengeling et al. (2021) for a good introduction to Graph Neural Networks in particular. The author has previously also covered many of the topics covered here in Avelar (2019), which might be useful to understanding some of the underlying ideas here, and from which we attempted to not repeat much information from here.

We will first give an introduction to the basic block of neural networks and the idea of backpropagation, and then we will pivot to an overview of convolutional neural networks, which will then lead into an explanation of graph neural networks through a convolutional lens. We will finish by explaining transformers also as an expansion of a convolutional neural network and briefly link it to graph neural networks. After that we will provide an overview of superpixel algorithms as they are a crucial part of our methodology for using graph neural networks for images, focusing on a particular superpixel algorithm (SLIC/SLICO, proposed in Achanta et al. (2010)) as it is the one used in our work, but also touching on other superpixel algorithms.

2.1 Deep Learning

2.1.1 Fully Connected Neural Networks and Backpropagation

Usually the first example of a neural network is traced back to McCulloch and Pitts (1943) and the first learning algorithm for such a neural network is often attributed to be the Hebbian rule (Hebb, 1949)¹. The early neural network were modelled under the assumptions of neurons receiving information from other neurons through weights that mediated those, and afterwards applying a non-linear transformation, usually threshold-

¹ Although originally published in 1949 a more recent edition was made available (Hebb, 2005).

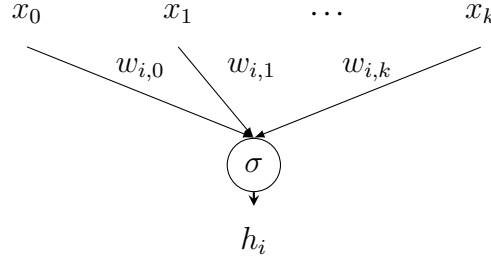


Figure 2.1 – An artificial neuron which receives inputs x_j and outputs h_j depending on the inputs, their associated weights $w_{i,j}$ and its activation function σ

ing, to activate or deactivate a neuron, as shown in Fig 2.1 and Eq 2.1². However, it was only between 1970 and 1990 that what could be recognised as one of the earliest forms of our current deep neural networks emerged using error backpropagation (BP) for optimising neural weights (Linnainmaa, 1970; Dreyfus, 1973; Werbos, 1974; LeCun, 1985; Rumelhart; Hinton; Williams, 1986; LeCun, 1988; Schmidhuber, 2015).

$$h_i = \sigma\left(\sum_j x_j w_{i,j}\right) \quad (2.1)$$

The understanding of a deep neural network as one that passes forward information through its neural connections from early layers allows for a more constrained formalisation of neural networks to emerge where a layer is typically only connected to the previous one, and allows us to transform our view from being connection-by-connection directed-graph to being a layer-by-layer and more modular view. That is, if before we'd think of a neuron forward pass as being as in Eq. 2.2, where the i -th neuron in layer l h_i^l aggregates information from the previous layer $h_j^{l-1} \in \mathbb{R}^{1 \times d_{l-1}}$ through its learned connection parameter $w_{i,j}^l$, now one can realise that such operation can be equivalently represented through a learned matrix $W^l \in \mathbb{R}^{d_{l-1} \times d_l}$, with d_l being the number of neurons on, or dimensionality of, layer l . This gives Eq. 2.3.

$$h_i^l = \sigma\left(\sum_j h_j^{l-1} w_{j,i}^l\right) \quad (2.2)$$

$$h^l = \sigma(h^{l-1} \times W^l) \quad (2.3)$$

These neural networks, composed of these Fully-Connected (FC) layers, are known either as Fully-Connected Networks or as Multi-Layer Perceptrons (MLPs), in reference

²Note that we do not include a bias term as it can equivalently be considered to be an input x_b that is always 1.

to their multiple layers and the historical Perceptron model. These became the building block for many other neural networks, and are usually represented either as individual layers or as blocks which represent an entire MLP.

For the specific case of multi-class classification³, the last layer L of an MLP will then have k neurons, each of which represent a *score* for its representative class. Then, we can define a probability distribution $\hat{p}$ from these scores by applying a *softmax* function, giving us the probability $\hat{p}_i$ for each class i as described in Eq 2.4.

$$\hat{p}_i = \frac{\exp(h_i^L)}{\sum_{j=1}^k \exp(h_j^L)} \quad (2.4)$$

A common error function for classification is the Cross Entropy between the predicted probability distribution and the target distribution which, in our case, can be seen as being the true label c for a classification instance, where $p_c = 1$ and all other $p_i = 0$. This gives us loss $\mathcal{L}$ in Equation 2.5 that can be backpropagated through the network to optimise the network's weights.

$$\mathcal{L} = p_c \log \hat{p}_c \quad (2.5)$$

2.1.2 Convolutional Neural Networks

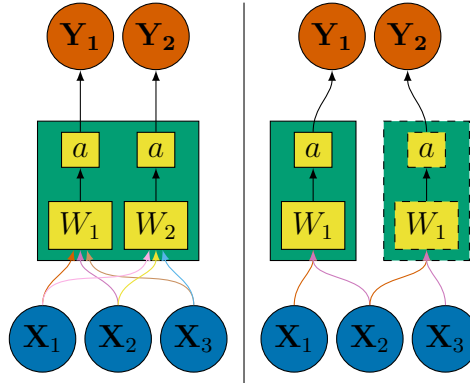
Convolutional Neural Networks (Fukushima, 1980; Lecun et al., 1998) differ from traditional fully-connected neural networks in that they preserve two *relational inductive bias* that are present in many different situations: (Locality) that nearby points in a space are often related to each other; and (translation-invariance) that the same points, if they were translated in space, would still hold similar or equal meaning. This is achieved through the concept of *parameter sharing*. That is: while a fully connected layer has each output node with its own parameters, a convolutional layer will have each output node share the same parameters, but applied to different parts of the input.

This parameter sharing allows for a much greater parameter efficiency in the models. For example, assuming a d_i -dimensional input with i points, a fully connected layer would need $(d_i \times i) \times (d_o \times o)$ weights to produce an output with o points and an output dimension of d_o , whereas a convolutional kernel that is applied to every k points at a time

³We only describe multi-class classification optimisation here as it is the topic of this thesis, but any sensible error function can be applied to any point of a neural network, as seen from the literature.

would produce $o = i - (k - 1)$ outputs using only $(d_i \times k) \times d_o$ parameters. This can be visualised in Figure 2.2 for the case where $i = 3$, $o = 2$ (and therefore $k = 2$).

Figure 2.2 – A diagram comparing a fully connected layer (left) to a small convolutional kernel (right), with the input edges coloured according to and their respective linear neuron’s weight. Note that on the right the convolutional *kernel* is shared between Y_1 and Y_2 , denoted by dashed outlines on the repeated kernel. Yellow squares are neural network layers, dark blue circles are inputs, red circles are outputs and green backgrounds are to represent the whole neural network block. Source: Author, based on Olah (2014) and Battaglia et al. (2018)



Convolutional neural networks have since their inception achieved great performance in diverse fields such as audio processing, where 1-d convolutions are used (Abdel-Hamid et al., 2012; Abdel-Hamid; Deng; Yu, 2013; Abdel-Hamid et al., 2014), image processing, where 2-d convolutions are used (Krizhevsky; Sutskever; Hinton, 2012; Simonyan; Zisserman, 2015; Li et al., 2015; He et al., 2016; Krizhevsky; Sutskever; Hinton, 2017), and video processing, where either 2-d or 3-d convolutions are used (Tran et al., 2015; Dosovitskiy et al., 2015; Simonyan; Zisserman, 2014; Karpathy et al., 2014), and they have remain one of the most popular methods in many fields to this day (Liu et al., 2022; Woo et al., 2023). This section’s main objective, however, is to show that sharing parameters over a certain data structure (in this case, n-dimensional arrays) can help construct neural networks that use less parameters and also, to realise that, due to how they exploit these data structures’ internal organisation, they provide better performance than when assuming that all inputs are unrelated to each other. This is the same idea we will be exploring below with Graph Neural Networks.

2.1.3 Graph Neural Networks

The idea of using neural networks for processing information contained in Graphs traces back to 2001 (Bianchini; Gori; Scarselli, 2001; Bianchini; Gori; Scarselli, 2002), which was formalised into something more similar to modern graph neural networks

(Scarselli et al., 2009), more specifically those based in message-passing (Gilmer et al., 2017). The idea follows a similar principle of parameter-sharing that allows an arbitrary number of nodes to be processed by the same network kernel (Bianchini; Gori; Scarselli, 2001), similar to what is done to a neighbourhood of points in a CNN.

However, not only do most graphs have nodes with varying number of connections, due to the scale-free properties of most graphs, the number of connections often varies drastically (Albert; Jeong; Barabási, 1999). This requires that a “graph convolution” operation be invariant to a node’s neighbourhood size. Although some earlier models had some success in using recurrent neural networks to process neighbourhoods, the use of recurrent neural networks mean that the input *order* of the neighbours matter to the operation.

Therefore, when applying convolutions in graphs, it is often common to individually transform each node, and then aggregate the node’s immediate neighbourhood $\mathcal{N}$ in an order-invariant way. Given a Graph $G = (V, E)$ composed of nodes $v \in V$ and edges $e \in E$; $s, t \in V$; $e_j = (s, t)$, with node i ’s embedding (i.e., its dense vectorial representation) on the next layer h_i^{l+1} given as the application of a learned update function g on its previous representation, together with the aggregation of a learned transformation f of each neighbour:

$$h_i^{l+1} = g(h_i^l, \text{AGGREGATE}(\{f^l(h_j^l) \forall j \in \mathcal{N}(i)\})) \quad (2.6)$$

More often than not, this computation is simplified by treating the node as being part of its own neighbourhood. Many order-invariant aggregation functions have been proposed in the past, such as feature-wise maximum (Hamilton; Ying; Leskovec, 2017), average (Kipf; Welling, 2017), among others. Despite this, it has been found that a simple sum usually produces a theoretically sound way to provide an unordered convolution on neighbours (Xu et al., 2019), which was already the original aggregation proposed in Scarselli et al. (2009), and naturally can be translated as a multiplication of the adjacency matrix M with the transformed node’s feature vector:

$$h^{l+1} = M \times f^l(h^l) \quad (2.7)$$

Drawing a comparison with convolutional neural networks, this would be equivalent to using a 1×1 convolutional kernel on an image and sum-pooling (i.e., reducing all values to a single embedding) the resulting values. In fact, modern convolutional net-

works (e.g. ConvNext from Liu et al. (2022)) have 1x1 convolution layers, which are equivalent to the application of a function node-wise – the major difference here is that they still perform (depthwise) convolutions to aggregate pixel neighbourhoods, which will learn a weighted sum instead of an unweighted sum. However this is mostly due to why CNN inputs are intrinsically ordered by their spatial configuration.

2.1.3.1 Graph Batching and Dense Adjacency Matrices

Most of the early examples of modern GNNs used a dense adjacency matrix for its neighbourhood aggregation, one of which was Kipf and Welling (2017). However, Kipf; Welling’s application was to perform transductive node classification, that is: to classify a node based through the relationships it has with other nodes based on the classes these other nodes are a part of. This means that there is no concept of a batch (i.e., feeding multiple inputs – graphs, in this case – to the algorithm in a single pass), and that each network is treated separately. When dealing with graph-based problems (i.e., classifying a graph that represents an image), there comes the need to pack all graphs into a single batch-graph tensor.

Graph batching that kept matrix multiplication aggregation was usually done in one of two ways: 1. Using another dimension as the batch dimension and performing batch matrix multiplication; 2. or batching graphs as a block-diagonal matrix, where all the graphs are converted into a single batch-graph, where each graph is a subgraph disconnected from the others.. However, both of these involve a significant increase in the time complexity for the aggregation operation, with $\mathcal{O}((bn)^2)$ space complexity for both, and $\mathcal{O}(b(n)^2d)$ time complexity for using another batch dimension and $\mathcal{O}((bn)^2d)$ for using block-diagonal matrices⁴.

2.1.3.2 Node-Edge and Edge-Node Adjacency Matrices

Another way to perform neighbourhood aggregation would be to split the adjacency matrix M into two adjacency matrices: M_s and M_t , of size $n \times m$ and $m \times n$, respectively. These matrices are a one-hot representation of which node is the source and target of an edge, respectively for M_s and M_t . That is, for a node $i \in V$ and an edge e_j , $M_s[i, j] = 1 \iff \exists e_j = (i, k), k \in V$ and equivalently for $M_t[j, i] = 1$. Having these two matrices one can then multiply $h^\top \times M_s$ in $\mathcal{O}(bnmd)$ time and using only $\mathcal{O}(bnm)$

⁴While using another dimension initially has an advantage in time complexity, this is further complicated if graphs differ significantly in size, as the batch would need to take the shape of the biggest graph.

space for the adjacency matrices. Since block-adjacency matrix are highly sparse, having only connections inside each graph, this means that $bnm \ll (bn)^2$, making this operation more efficient:

$$h^{l+1} = M_t \times (M_s^\top \times f^l(h^l)) \quad (2.8)$$

In practice, however, most deep learning frameworks also allow for index-expansion, which allows one to substitute M_s with an index I_s , which allows the first operation to be performed in $\mathcal{O}(md)$ time and requiring only $\mathcal{O}(m)$ space for the index:

$$h^{l+1} = M_t \times f^l(h^l[I_s]) \quad (2.9)$$

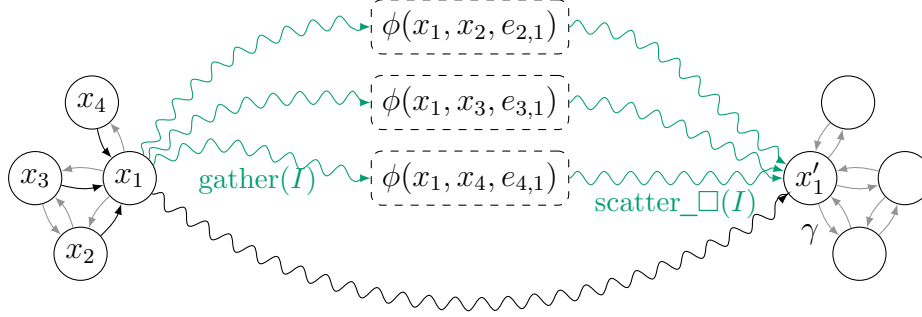
This technique was by the student to implement a generic Graph Neural Network library (Avelar, 2018), which was used to reimplement the model developed in Selsam et al. (2019) and later on used in GNN-based publications from the student’s group (Avelar, 2019; Prates et al., 2019; Lemos et al., 2019; Lemos et al., 2020). It was also used by the student when implementing a GAT for superpixel image classification (Avelar et al., 2020).

2.1.3.3 Applications of Graph Neural Networks

Graph Neural Networks (GNNs) have had considerable impact on Machine Learning (Battaglia et al., 2018). This is due, in part, to the many demonstrations of the power of GNN to model inductive biases of a problem by carefully crafting graphs that matched the structure of the problem (Gilmer et al., 2017; Battaglia et al., 2018), of which a notable example is the NeuroSAT model (Selsam et al., 2019): where a boolean satisfiability (SAT) instance is modelled into a bipartite graph between literals and clauses, allowing communication to flow between these iteratively until the neural “solver” reaches equilibrium.

The generalisability of GNNs and the many advancements in the area culminated in the development of a myriad of deep graph learning frameworks, such as Pytorch Geometric (Fey; Lenssen, 2019) and DGL (Wang et al., 2020) being popular choices (Wu et al., 2021), and the use of such frameworks being expanded on through area-specific extensions and resources, such as DGL-LifeSci (Li et al., 2021) for Life Sciences, and MatDeepLearn (Fung et al., 2021) and DMol (White, 2022) for Chemical/Materials engineering being notable examples.

Figure 2.3 – Example of how a GNN layer might be computed using gather and scatter operations, where $\square$ is a permutation-invariant function (e.g. sum, mean or max), and ϕ and γ are differentiable functions. Source: Author, based on Fey and Lenssen (2019)



2.1.3.4 Modern approaches

Sparse Adjacency Matrices are widely used in scientific computing and are the main implementation strategy that allows adjacency matrices to scale on larger graphs, being used throughout the `scipy` (Virtanen et al., 2020) library stack, such as for KNN graphs in `scikit-learn` (Pedregosa et al., 2011). Sparse Adjacency Matrices have been slowly incorporated in deep learning frameworks such as `pytorch` (Paszke et al., 2019; Ansel et al., 2024). However, their implementation still remains in beta to this day: “The PyTorch API of sparse tensors is in beta and may change in the near future. We highly welcome feature requests, bug reports and general suggestions as GitHub issues.” (Torch.Sparse Contributors, 2024).

Modern GNN-specific libraries such as Pytorch Geometric (Fey; Lenssen, 2019) also implement many of the functionality by providing differentiable and GPU-accelerated *gather* and *scatter* operations. These gather and scatter operations are similar in principle to Node-to-Edge and Edge-to-Node matrices, as they similarly spread edge information from source nodes into edges, and then from edges into a target node, which can be visualised in Figure 2.3. The major difference between this and the source-index variant discussed above is that since both are done through indices, the final time complexity of the operations is $\mathcal{O}(md)$ and it requires only $\mathcal{O}(m)$ space for the indices.

2.1.3.5 The Message-Passing view of Graph Neural Networks

GNNs can be interpreted through many different lenses, of which the most common are either a generalization of convolutions and convolutional neural networks (Monti et al., 2017; Kipf; Welling, 2017; Fey et al., 2018) described above. Another interpretation of GNNs is that of GNNs as being “Message-Passing Neural Networks” (MPNN)

(Gilmer et al., 2017; Selsam et al., 2019). In summary, under the MPNN framework, a simple graph neural network will receive a graph $G = \{V, E\}$ composed of a set of vertices $v \in V$ and edges $e \in E$, with edges connecting a source $s \in V$ and a target node $t \in V$, such that the edge can be identified by the pair $e = (s, t)$. The neural network will first embed the attributes of each of the nodes through a usually learned function so that we have $h_v^0 = f(v)$ and then propagate the information between the nodes using the adjacencies defined through the edges. Under the MPNN framework, this is seen as each of the nodes u being a neural computer g with state h_u which then exchanges messages $m_u^t = g^t(h_u^t)$ at a discrete time step t (which can also be interpreted as a layer). These messages then are aggregated through some, usually order-invariant, function a on the node’s neighbourhood $\mathcal{N}(u)$ to form the basis of the next time steps’ state $h_u^{t+1} = a(\{m_v^{t+1}, v \in \mathcal{N}(u)\})$. This view is relevant when comparing GNNs with other architectures, as we will do below.

2.1.4 Transformers and Attention

In this section we will give a brief constructive overview of the Transformer model, mostly based on the interpretation given in Algorithmic Simplicity (2023), which we highly recommend the interested reader to go through. Although recurrent neural networks were the state-of-the-art for language-based tasks in 2017 and had surpassed convolutional neural networks in this specific domain, many works still attempted incorporating convolutional layers for sequence processing due with the goal of reducing sequential computation (Kaiser; Bengio, 2016; Gehring et al., 2017; Kalchbrenner et al., 2017), these works all had to circumvent the main issues with using CNNs for sequence data: the depth of convolutions needed to be able to relate two far-apart items in a sequence.

Transformers attempt to fix CNN and Recursive Neural Networks’ issues with long-range relations (Schuster; Paliwal, 1997; Graves; Fernández; Schmidhuber, 2005; Schmidhuber, 2015) by having “pairwise convolutional layers” which would take two values i, j , which are not necessarily immediate neighbours, from the input sentence x and perform an operation that combines the information from these two pairs into a hidden representation $c_{i,j} = f(x_i, x_j)$, therefore, relating all values in a sequence to all other values with fully parallelisable operations. However, such an operation not only creates n^2 outputs for a sequence with n entries, but also combining each pair of values would be

equivalent to using a bag-of-words model, which does not consider the order in which the words appear.

The proposed solution for dealing with the word ordering issue is to simply add information about the position of a word into the word’s input embedding, thus performing the operation as some $c_{i,j} = f(x_i + p_i, x_j + p_j)$ where p is an embedding containing the positional information for the sequence. In particular, in Vaswani et al. (2017), they use sine and cosine functions of different frequencies to encode a node’s positional information. This is done so that, for a positional embedding of dimension d_{model} – that is, $p_i \in \mathbb{R}^{d_{model}}$ – the value of the embedding in odd positions ($2j + 1$) is equivalent to a cosine and in even positions ($2j$) it is equivalent to a sine function as in Eq. 2.10. Other common options include also to have one learned embedding per position in a sequence, which became quite popular after its use in Devlin et al. (2019), and will be used later in this work.

$$\begin{aligned} p_i[2j] &= \sin(i/10000^{2j/d_{model}}) \\ p_i[2j + 1] &= \cos(i/10000^{2j/d_{model}}) \end{aligned} \quad (2.10)$$

In any case, even though this allows our pairwise operation to take into account positional information, it still leaves such an operation with an exploding concept size, where on each layer the number of values would be squared in size. To address this issue, it is necessary to aggregate the n^2 output back into n vectors. For this, the Transformer model uses a weighted average where each weight is marked by how related the two inputs are: Self-Attention. This uses a set of n query, key and value vectors, which are respectively represented as the matrices $q \in \mathbb{R}^{n \times d_k}$, $k \in \mathbb{R}^{n \times d_k}$ and $v \in \mathbb{R}^{n \times d_v}$, and performs the operation in Equation 2.11, using a scaling factor before the softmax to avoid regions of extremely small gradients (Vaswani et al., 2017).

$$\text{attention}(q, k, v) = \text{softmax}\left(\frac{qk^\top}{\sqrt{d_k}}\right)v \quad (2.11)$$

Performing self-attention by both using the same vector to generate both the key and value vectors, and applying a different transformation for each of them, allows us to define our previous pairwise convolution as $c_i = \text{attention } W_k(x + p), W_q(x + p), W_v(x + p)_i$, which maps the n^2 . Other options for attention exist, which include using a single linear layer or a bilinear layer between each key-query pair. While this would allow the neural network to effectively transform the information for each pair of vectors, it might be beneficial to have information from multiple pairs into the next layer. To do so, the model

uses Multi-Head Attention (MHA) as in Eq. 2.12, where each attention head has separate weights and operates only on d_k dimensions and then are concatenated ($\parallel$) back together, keeping the overall computational cost similar to single-head attention while allowing information from multiple other values in the sequence to be selected.

$$\text{MultiHead}(q, k, v) = (\parallel (\text{head}_1, \dots, \text{head}_h))W^O \quad (2.12)$$

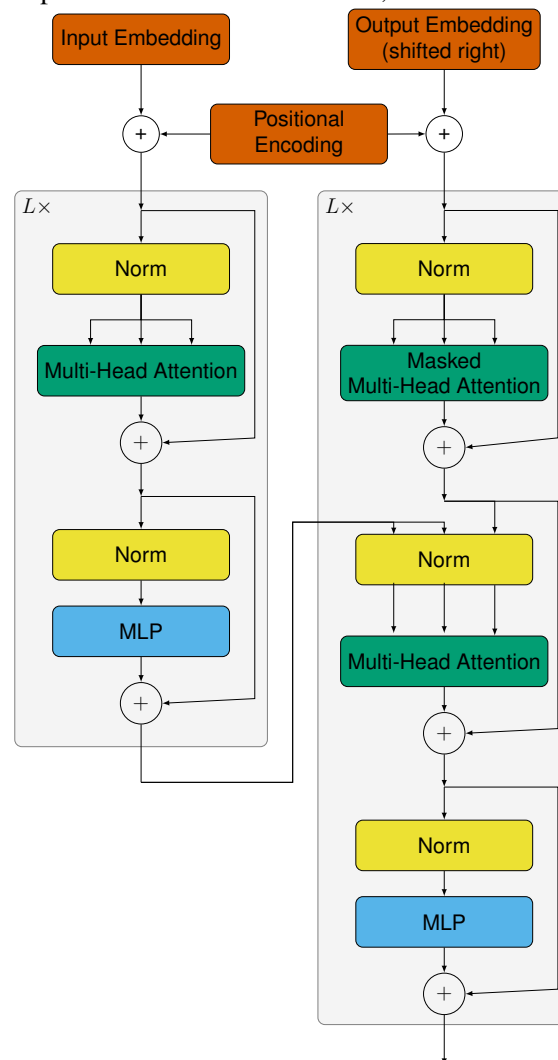
$$\text{where head}_i = \text{attention}(q \times W^{q_i}, k \times W^{k_i}, v \times W^{v_i})$$

Finally, the transformer also employs both regularising layers such as layer/batch norm (Ioffe; Szegedy, 2015; Ba; Kiros; Hinton, 2016) and residual/skip connections (He et al., 2016) in between each MHA module, and furthermore applies an MLP module to update each item’s internal representation from those transformed through multi-head attention, much like how a MPNN’s node updates its representations. Finally, for the decoder architecture, Vaswani et al. (2017) also defined a masked attention, shown in Eq 2.13, and has as its only difference that the attention values are point-wise multiplied (Hadamard product $\circ$) with a binary mask m . The final architecture can be visualised in Fig 2.4. An interactive visualisation of a text-generative transformer can also be seen in (Cho et al., 2024).

$$\text{masked-attention}(q, k, v, m) = (\text{softmax}(\frac{q \times k^\top}{\sqrt{d_k}}) \circ m) \times v \quad (2.13)$$

While the transformer architecture was already a breakthrough for machine translation, it soon became apparent that it was suitable for many other tasks in natural language processing, including word/sentence embedding, for which an Encoder-only architecture is used (Devlin et al., 2019), or language modelling for which a decoder-only architecture is often used (OpenAI et al., 2023). While other small architectural changes and optimisations have been proposed over time, the overall idea of the Transformer model to exclusively use self-attention to process inputs has remained the same, and has expanded even into other domains, such as Computational Chemistry (Wu; Radev; Li, 2023; Honda; Shi; Ueda, 2019; Wang et al., 2019), Computational Biology (Chen et al., 2023; Cui et al., 2023), and has recently also been proposed for image processing with the development of the Vision Transformer (Dosovitskiy et al., 2021). Vision Transformers have since become a staple in Computer Vision and, despite being notoriously data-hungry, they have surpassed CNNs in many applications, including image classification, object detection and semantic image segmentation (Steiner et al., 2022). We will cover ViTs in

Figure 2.4 – A diagram showing the transformer architecture (Vaswani et al., 2017), slightly modified for consistency with the variant we use in our work. The transformer architecture, initially envisioned for Machine Translation, would be composed of a Transformer Encoder tower (left) and a Transformer Decoder tower (right). The encoder tower would have full attention enabled on the input sentence, where every word would be allowed to attend to every other word, whereas the decoder tower would use *masked* attention which disallowed the model from using tokens which had not been generated yet, while also having unmasked attention with respects to the input sentence. Source: Author, based on Vaswani et al. (2017)



more depth in Chap. 5.

2.1.5 Graph Attention Networks

Graph Attention Networks (GATs) (Velickovic et al., 2018) were originally developed in 2017⁵ after the surge of interest in Graph-based neural networks (Battaglia et al., 2018), due, in part, to the Graph Convolutional Network (GCN) model (Kipf; Welling, 2017) making graph neural networks accessible in modern deep learning frameworks, in tandem with the surge of interest in attentional mechanisms due to the Transformer model (Vaswani et al., 2017). Graph attention networks function in a very similar way to other GNN architectures, except that they mediate the message-passing between neighbours through a self-attention mechanism very similar to Eq. 2.11, allowing the network to learn how to prioritise information flow depending on the each node's information content. That is, given the representation h_i of node i , we define an update rule that is based on an attentional mechanism but that performs masked self-attention by only computing the softmax over i 's neighbours $\mathcal{N}(i)$, as in Eq. 2.14 below, where each layer has three learned parameter matrices: $W^{\leftarrow}$, $W^{\rightarrow}$, and W^{repr} which could be seen as the query, key, and value representation transformations, with a being a function to measure the similarity between the key and query vectors such as the abovementioned dot product:

$$\begin{aligned}
 h'_i &= \sigma\left(\sum_{j \in \mathcal{N}(i)} \alpha_{i,j}(h_j \times W^{\text{repr}})\right) \\
 &= \sigma\left(\sum_{j \in \mathcal{N}(i)} \text{softmax}_j(e_{ij})(h_j \times W^{\text{repr}})\right) \\
 &= \sigma\left(\sum_{j \in \mathcal{N}(i)} \left(\frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}(i)} \exp(e_{ik})}\right)(h_j \times W^{\text{repr}})\right) \tag{2.14} \\
 &= \sigma\left(\sum_{j \in \mathcal{N}(i)} \left(\frac{\exp(a(h_i \times W^{a\leftarrow}, h_j \times W^{a\rightarrow}))}{\sum_{k \in \mathcal{N}(i)} \exp(a(h_i \times W^{a\leftarrow}, h_k \times W^{a\rightarrow}))}\right)(h_j \times W^{\text{repr}})\right) \\
 &\approx \sigma(\text{masked-attention}(h \times W^{a\leftarrow}, h \times W^{a\rightarrow}, h \times W^{\text{repr}}, M))
 \end{aligned}$$

It is possible, however, to draw a relationship between GATs and Transformers. In fact, if we assume that a Transformer's attention mask can be treated as the adjacency matrix in a GAT, it is clear that both architectures work in a similar way, with a layer

⁵Arxiv version released in 2017: <<https://arxiv.org/abs/1710.10903>>

transforming each input/node, and then having self-attention mediating the information between non-masked inputs/neighbours. GATs differs from Transformers almost only through the lack of specific layers in the update block, such as residual/skip connections and layer/batch normalisation of the input. This similarity has been exploited to develop graph-specific transformer architectures through adopting a graph-specific positional embedding using Laplacian eigenvectors (Dwivedi; Bresson, 2021), but which failed to see the connection between masking and graph-masking. We will discuss more about the similarities and differences between GATs and Transformers in Chap. 5.

2.2 Superpixel Algorithms

Superpixels are groups of pixels in semantically similar areas of an image. A number of techniques exist to generate superpixels from images, such as SLIC (Achanta et al., 2012), SNIC (Achanta; Susstrunk, 2017), SEEDS (Bergh et al., 2012), ETPS (Yao et al., 2015), and SH (Wei et al., 2018). Out of these techniques, SLIC is widely thought of as one of the most stable and is still recommended over other state-of-the-art algorithms (Stutz; Hermans; Leibe, 2018).

SLIC is a modified version of the K-means algorithm which only attempts to assign points to cluster centers if these points are at most at a within a $2S$ -sized square from the cluster center, where $S = \sqrt{N/k}$, with N the number of pixels in the image and k the desired number of superpixels. It works by initialising k cluster centers C_j that are composed of a colour component c_{C_j} and a spatial component s_{C_j} , and then iteratively assigning pixels to clusters, and updating the cluster means based on their component pixels. The algorithm is laid out in detail in Algorithm 1. Distances between cluster centers and pixels are calculated as the l2 norm of the two normalised euclidean distance vectors: one of the colour component, and one for the spatial component, normalised by the expected maximum spatial distance (S) and the maximum expected colour distance, which is usually represented by a manually-selected constant m ⁶.

Some variations of the SLIC algorithm were proposed by the authors, which involve the same algorithm, but modifying the distance calculation in line 11 of Algorithm 1. This includes Adaptive SLIC (ASLIC) and SLICO, both of which make use of adaptive distance normalisation. ASLIC modifies the distance calculation by normalising

⁶lower values of m give more emphasis to colour homogeneity but produce superpixels with less regular size and shape.

Algorithm 1 SLIC superpixel segmentation (Achanta et al., 2012)

```

1: procedure SLIC(Image  $I$  of width  $w$ , height  $h$ ,  $c$  channels, desired number of super-
   pixels  $k$ , maximum expected colour distance  $m$ )
2:   Init.  $k$  cluster centers  $C_j = [c_{C_j}, s_{C_j}]^\top$  by sampling pixels at regular grid steps  $S$ .
3:   Move  $C_k$  to the lowest gradient position in a  $3 \times 3$  neighbourhood
4:   Set label  $l(i) = -1$  for each pixel  $i$ 
5:   Set distance  $d(i) = \inf$  for each pixel  $i$ 
6:   repeat
7:     for each cluster center  $C_j$  do ▷ Assign pixels to Clusters
8:       for each pixel  $i$  in a  $2S \times 2S$  region around  $C_j$  do
9:          $d_c \leftarrow \|c_i - c_{C_j}\|_2$ 
10:         $d_p \leftarrow \|s_i - s_{C_j}\|_2$ 
11:         $D \leftarrow \sqrt{\left(\frac{d_c}{m}\right)^2 + \left(\frac{d_p}{S}\right)^2}$ 
12:        if  $D < d(i)$  then
13:           $d(i) \leftarrow D$ 
14:           $l(i) \leftarrow j$ 
15:        end if
16:      end for
17:    end for
18:    for each new cluster center  $C'_j$  do ▷ Compute new cluster centers
19:       $c_{C'_j} = \frac{1}{\sum_{i:l(i)=j} 1} \sum_{i:l(i)=j} c_i$ 
20:       $s_{C'_j} = \frac{1}{\sum_{i:l(i)=j} 1} \sum_{i:l(i)=j} s_i$ 
21:    end for
22:    Compute residual error  $E$  with the l2 norm between the new  $C'_j$  and old  $C_j$ 
23:     $C_j \leftarrow C'_j$ 
24:  until  $E \leq \text{threshold}$ 
25:  return pixel labels  $l(i)$  and cluster centers  $C_k$ 
26: end procedure

```

the proximities of each cluster j using the maximum observed spatial ($m_s(j)$) and colour ($m_c(j)$) distances from the previous iteration, using Equation 2.15 for all iterations after the first.

$$D = \sqrt{\left(\frac{d_c}{m_c(j)}\right)^2 + \left(\frac{d_s}{m_s(j)}\right)^2} \quad (2.15)$$

Even though ASLIC achieves a more consistent compactness, that isn't necessarily always set to m , it was shown to cause a reduced boundary recall (Achanta et al., 2012). SLICO instead keeps the distance component unchanged from SLIC but allows for the compactness value to be chosen adaptively for each superpixel, which allows for the algorithm to work better in images that contain both highly textured and non-textured regions, and doesn't require the user to set the compactness parameter themselves while still keeping the step size constant and user-defined. This means that line 11 of Algorithm 1 will be replaced with Eq. 2.16, where $m_c(j)$ is the maximum colour distance in the previous iteration. The full modified algorithm can be seen in Algorithm 5.

$$D = \sqrt{\left(\frac{d_c}{m_c(j)}\right)^2 + \left(\frac{d_s}{S}\right)^2} \quad (2.16)$$

While the usual representation for images – a grid/tensor of pixels – can already be seen as a regular grid graph (a lattice), superpixels induce a graph representation of an image more naturally (Monti et al., 2017; Avelar et al., 2020). This is due to the fact that, since superpixels are usually irregular, the connectivity between superpixels requires a more abstract structure than a simple grid or hypercube. Thus, superpixels which have bordering pixels become connected between themselves. Therefore, a superpixel-based representation is more suitable for processing with Graph Neural Networks.

When an image can be represented as a grid of pixels, the convolution window could be seen as a pixel's neighbourhood. However, the fact that GNN-based methods often aggregate a node's neighbourhood with a single set of weights would imply that simple GNNs would have a lower representational power. In fact, this is exactly what is shown in Monti et al. (2017). We believe that an idea similar to that of using Multi-Head Attention (Vaswani et al., 2017) can remedy this by allowing the neighbourhood aggregation to contain information from multiple neighbours through different learned weights.

3 RELATED WORK

During the development of our work, we encountered many different applications of GNNs as images classifiers. Although superpixel-like segmentation algorithms had been used together with Neural Networks for processing images since 2005 by Bianchini et al., it was only in 2017 that modern iterations of Graph Neural Networks started using superpixel algorithms to generate image graphs (Monti et al., 2017). We limited ourselves to comparing our developments with those models which were purely GNN-based, of which three were prominent: MoNET (Monti et al., 2017), SplineCNN (Fey et al., 2018), and GeoGCN (Danel et al., 2020), where the latter two use the same dataset provided by Monti et al. (2017): a KNN graph of the superpixels.

MoNET (Monti et al., 2017) was the first application of a modern GNN model to image classification. They defined a graph convolution based on a previous definition of a *patch* operator where, given a point¹ x and points in its neighbourhood $y \in \mathcal{N}(x)$, they define a d -dimensional vector of pseudo-coordinates $u(x, y)$ and an associated weighting function/kernel $w_\Theta(u) = (w_1(u), \dots, w_J(u))$, where J represents the dimensionality of the extracted patch. Therefore, the patch operator $D_j(x)f$ is defined as:

$$D_j(x)f = \sum_{y \in \mathcal{N}(x)} w_j(u(x, y))f(y), \quad j = 1, \dots, J \quad (3.1)$$

Then, a convolution is defined as:

$$(f \star g)(x) = \sum_{j=1}^J g_j D_j(x)f \quad (3.2)$$

Within this framework, Monti et al. generalised many previous graph neural networks (such as Kipf and Welling (2017)), with these having handcrafted weight functions w_j . For example, GCN (Kipf; Welling, 2017) would have $u(x, y) = \deg(x), \deg(y)$ and $w(u) = (1 - \left|1 - \frac{1}{\sqrt{u_1}}\right|)(1 - \left|1 - \frac{1}{\sqrt{u_2}}\right|)$. The main contribution of Monti et al., however, was that they used parametric gaussian kernels with learnable parameters, allowing for the neural network to discover weights which optimised convolutions towards each specific problem. SplineCNN (Fey et al., 2018) similarly uses a trainable kernel, but instead uses B-splines as basis functions for the patch kernels.

GeoGCN (Danel et al., 2020) builds up on simpler Graph Convolutions (Duvenaud et al., 2015; Defferrard; Bresson; Vandergheynst, 2016), where

¹In their definition x can mean both a node in a graph as well as a point in a manifold.

$$h_x^{l+1} = \sum_{y \in \mathcal{N}(x)} \sigma(W^l \times h_y^l) \quad (3.3)$$

However, they borrow from Monti et al. (2017) the concept of kernels that learn based on the pseudo-coordinates u , building instead layers as

$$h_x^{l+1} = \sum_{y \in \mathcal{N}(x)} \sigma(U^{l\top} \times (p_y - p_x) + b^l) \odot h_y^l \quad (3.4)$$

where $U \in \mathbb{R}^{t \times d}$, and each node x is associated with a t -dimensional coordinate $p_x \in \mathcal{R}^t$.

However, although all of the abovementioned methods allow for a learned convolutional operator that is based on point-wise distance, they all fail to address that points (or patches) in an image might not necessarily depend exclusively on their positions, and that a neural network might be able to learn to weight each node's aggregation using not only its spatial information, but also the information contained within the node itself. This is especially salient in the case of superpixels, where separating the positional information from other information such as colour, texture and shape could improve performance.

4 SUPERPIXEL IMAGE CLASSIFICATION WITH GRAPH ATTENTION NETWORKS

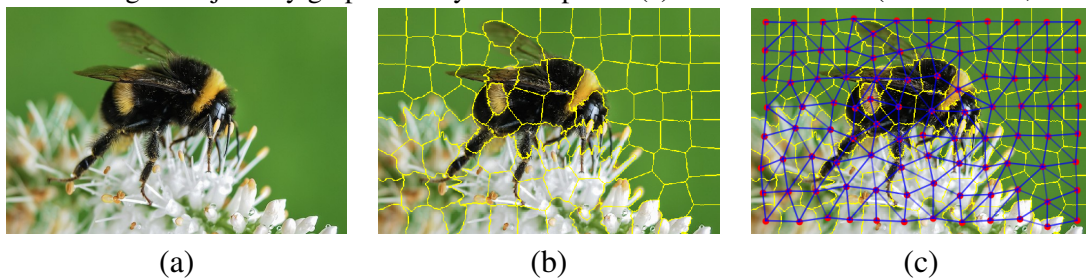
Here we briefly provide the results from our work with Superpixel Image Classification (SIC) (Avelar et al., 2020) where we used Graph Attention Networks (GATs). This model is also referred to as SICGAT in this thesis.

4.1 Superpixel images as Region-Adjacency Graphs

Previous work has also provided superpixel image classification, but they dealt with these images using a KNN graph with either 300, 150, or 75 superpixels (Monti et al., 2017). We build a similar dataset for each of the abovementioned datasets, choosing the harder problem of having only 75 superpixels. For our experiments, we chose to use SLICO (Achanta et al., 2010), a zero-parameter variant the SLIC algorithm (Achanta et al., 2012), for building superpixels, as it is readily available, doesn't require parameter tuning, and has a spatial component for its segmentation. Nonetheless, we believe that other segmentation techniques with similar characteristics could be used. Although we set the target number of superpixels as 75, the generated RAGs are not guaranteed to have exactly 75 nodes due to how the SLIC algorithm works.

After applying SLICO, we generate a Region Adjacency Graph (RAG), by treating each superpixel as a node and adding edges between all directly adjacent superpixels (1-neighborhood connection). This is different from Monti et al. (2017), which builds a KNN graph, spanning a further neighbourhood than our graph. This means that for our problem information should be harder to spread. Algorithm 2 describes the out procedure, whereas Figure 4.1 depicts the generation of a RAG from an image.

Figure 4.1 – From left to right, the image to be converted into a RAG (a), the image with the superpixel segmentation being shown (b), and the image with the superpixel segmentation and the generated region adjacency graph overlayed on top of it (c). Source: Author (Avelar et al., 2020).



Algorithm 2 Region Adjacency Graph (RAG) generation (Avelar et al., 2020)

```

1: procedure SUPERPIXEL2GRAPH(Image  $I$  of width  $w$ , height  $h$  and  $k$  channels, Superpixel segmentation technique  $S$ , and node feature builder  $F$ )
2:    $s, \mathcal{N} \leftarrow S(I)$   $\triangleright s$  returns the superpixel  $n \in \mathcal{N}$  of a (x,y) pixel
3:    $x(n) \leftarrow F(I, s, n) \forall n \in \mathcal{N}$   $\triangleright x(n)$  is the feature vector of node  $n$ 
4:    $\mathcal{E} \leftarrow \{\}$ 
5:   for  $1 \leq x \leq w$  do
6:     for  $1 \leq y \leq h$  do
7:       if  $s(x, y) \neq s(x + 1, y)$  then
8:          $\mathcal{E} \leftarrow \mathcal{E} \cup \{(s(x, y), s(x + 1, y))\}$ 
9:       end if
10:      if  $s(x, y) \neq s(x, y + 1)$  then
11:         $\mathcal{E} \leftarrow \mathcal{E} \cup \{(s(x, y), s(x, y + 1))\}$ 
12:      end if
13:    end for
14:  end for
15:  return  $\mathcal{G} = (\mathcal{N}, \mathcal{E}), x$ 
16: end procedure

```

For grayscale images we build each superpixel’s feature vector as the concatenation of the average luminosity of the pixels in a superpixel and the geometric (x,y) centroid. For RGB images we naturally extend the feature vector to use the concatenation of the average value for each colour channel alongside the geometric centroid. We do not build features for the edges, since we use an attention-based technique, and believe that the edge feature will be learned accordingly from the attention mechanism using both nodes’ features. There exist other possibilities for building the superpixel features such as using statistics (mean, standard deviation, and correlation matrices) of the pixels in the superpixel, but these were left to be explored in future work.

4.2 Graph Attention Networks

Our model is a version of the GAT model by Velickovic et al. (2018), roughly based on the implementation in Nathani et al. (2019). We transform the Undirected Graph produced from the segmented image’s RAG into a Directed Graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, with edges $e = (s_e, t_e) \in \mathcal{E}$ and node features $x(n) \forall n \in \mathcal{N}$. Let $\mathcal{T}(t)$ be the set of nodes with an edge towards t , we scatter the source and target nodes’ input features (each with dimension d_i) into their respective edges. Given the scattered input features, we run a transition function, $f : \mathbb{R}^{2d_i} \rightarrow \mathbb{R}^{d_o}$, to produce an output with dimension d_o , and an attention function, $a : \mathbb{R}^{2d_i} \rightarrow \mathbb{R}$. The values produced by the attention function are

activated using softmax over the target nodes. The attention model can be summarised by Equations (4.1) and (4.2) below, where $\parallel$ denotes vector/tensor concatenation.

$$\alpha(s, t) = \frac{\exp(a(x(s) \parallel x(t)))}{\sum_{s' \in \mathcal{T}(t)} \exp(a(x(s') \parallel x(t)))} \quad (4.1)$$

$$o(t) = \sum_{s \in \mathcal{T}(t)} \alpha(s, t) f(x(s) \parallel x(t)) \quad (4.2)$$

The detailed algorithm showing the optimisation can be seen in Algorithm 3. Each of these layers can be arranged in a multi-head model by concatenating their outputs after the forward pass of each layer. That is, given k heads, the joint output of the k -headed layer, where each head has its own transition and attention functions f_i and a_i (as well as the intermediary α_i), would be as in Equation (4.3), where $\parallel_{i=1}^k a_k$ is the concatenation of all vectors/tensors a_i :

$$o(t) = \sum_{s \in \mathcal{T}(t)} \parallel_{i=1}^k \alpha_i(s, t) f_i(x(s) \parallel x(t)) \quad (4.3)$$

We sum-pool the output of the final GAT layer (i.e., $\sum_{n \in \mathcal{N}} o(n)$) and apply a Multi-Layer Perceptron (MLP) for the final prediction. We train two versions of the GAT model: A single-headed GAT with 3 layers, with the layers having 32, 64 and 64 neurons, and a two-headed GAT, where each head has the same amount of neurons as the single-headed model. The last MLP block has two layers of 32 and d_o neurons for the final classification, where d_o is the number of classes in the target dataset. The Python/Pytorch implementation in its fullest can also be seen in the provided GitHub repository¹.

4.3 Benchmark Datasets

We tested our models in four datasets: MNIST (Lecun et al., 1998), FashionMNIST (Xiao; Rasul; Vollgraf, 2017), SVHN (Yuval Netzer et al., 2011), and CIFAR-10 (Krizhevsky, 2009). All four datasets are image classification datasets with 10 classes. MNIST is a handwritten digit-classification dataset, with each class representing a number from 0-9 (Lecun et al., 1998). Similarly, SVHN is a digit classification dataset, but the images are from house number taken from google street view (Yuval Netzer et al., 2011). CIFAR-10 is a collection of tiny images which have 10 classes, with 6000 ex-

¹<<https://github.com/machine-reasoning-ufrgs/spixel-gat>>

Algorithm 3 Implemented GAT Layer. Source: Author (Avelar et al., 2020)

```

1: procedure GAT-FORWARD(Directed graph  $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ , Node Features  $x(n) \forall n \in \mathcal{N}$ , learnable transition function  $f$  and learnable attention function  $a$ )
2:    $M_{tgt}(t_e, e) \leftarrow 1\{e = (s_e, t_e)\} \forall e \in \mathcal{E}$ 
3:    $h_{src}(s_e) \leftarrow x(s_e) \forall e \in \mathcal{E}$ 
4:    $h_{tgt}(t_e) \leftarrow x(t_e) \forall e \in \mathcal{E}$ 
5:    $h(e) \leftarrow h_{src}(s_e) \parallel h_{tgt}(t_e) \forall e = (s_e, t_e) \in \mathcal{E}$ 
6:    $y(e) \leftarrow f(h(e)) \forall e \in \mathcal{E}$ 
7:    $\alpha(e) \leftarrow a(h(e)) \forall e \in \mathcal{E}$ 
8:    $\alpha_{base}(e) \leftarrow \max_{e \in \mathcal{E}} \alpha(e)$ 
9:    $\alpha_{norm}(e) \leftarrow \alpha(e) - \alpha_{base}(e) \forall e \in \mathcal{E}$ 
10:   $\alpha_{exp}(e) \leftarrow \exp(\alpha_{norm}(e)) \forall e \in \mathcal{E}$ 
11:   $\alpha_{sum} \leftarrow (M_{tgt} \times \alpha_{exp}(e)) + \epsilon$ 
12:   $y_\alpha(e) \leftarrow y(e) \alpha_{exp}(e) \forall e \in \mathcal{E}$ 
13:  return  $o = (M_{tgt}, y_\alpha) / \alpha_{sum}$ 
14: end procedure

```

amples for each class: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck (Krizhevsky, 2009). FashionMNIST is a dataset that follow the data standards of MNIST with 28×28 pixels and a single channel, but with more complex images that are less trivial to classify than MNIST (Xiao; Rasul; Vollgraf, 2017). For all datasets we use their original training and testing splits. If a validation set is provided, we use it as our validation set, if not, we randomly split the training dataset into 90%/10% for training/validation. We use classification accuracy as our evaluation metric.

4.4 Results

4.4.1 MNIST and FashionMNIST

Table 4.1 shows that both GAT models performed better than the MoNET model (Monti et al., 2017), showing that a learned representation of the geometric distance can lead to better performance than the fixed one in Monti et al. (2017). Note that our model has also to deal with graphs that are sparser than the ones used in the baselines (Monti et al., 2017; Fey et al., 2018; Danel et al., 2020), since in their graph edges are formed through K-nearest neighbours with $k=15$ and ours use only directly adjacent nodes which, due to SLIC’s compact superpixels, should average around 9 edges per superpixel. Although one could expect worse accuracy, the results suggest that our approach is able to learn relevant geometric information relating the features from all neighbouring super-

Table 4.1 – Test accuracy for the tested models on MNIST and FashionMNIST, processed as RAGs with approximately 75 nodes (called MNIST-75 and FashionMNIST-75), compared to the baseline models. Bold values show the best of the Graph-based models. We also present the mean accuracies of the two best classifiers for the non-segmented MNIST and FashionMNIST datasets, available in the FashionMNIST benchmark. Source: Author (Avelar et al., 2020)

	MNIST-75	FashionMNIST-75
MoNET (Monti et al., 2017)	91.11%	-
SplineCNN (Fey et al., 2018)	95.22%	-
GeoGCN (Danel et al., 2020)	95.95%	-
GAT-1Head	95.83%	83.07%
GAT-2Head	96.19%	81.40%
	MNIST	FashionMNIST
GTNC (Sun et al., 2020)	97.6%	88.2%
SVC(10,poly)	97.6%*	89.7%*
SVC(100,poly)	97.8%*	89.6%*

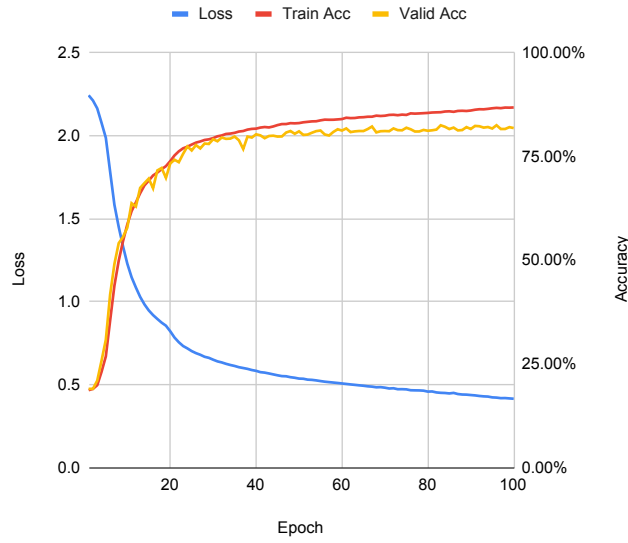
pixels. We also report the performance of the graph-based approaches SplineCNN (Fey et al., 2018) and Geo-GCN (Danel et al., 2020), as well as recent alternative approaches such as the Generative Tensor Network Classifier (GTNC) (Sun et al., 2020) and the best performing method based on Support Vector Classification (SVC) as reported in the dataset homepage². Although our approach does not reach state-of-the-art results, it is competitive and the best among graph-based methods.

None of the graph-based models report their performance on Fashion-MNIST, but we can see that the performance gap between the traditional ML models that used the full features of the dataset and our models, is higher on FashionMNIST. This shows how much harder the FashionMNIST dataset is for segmented images, where the information loss is greater with the aggregation of the features of the pixels in the superpixel. It is also interesting to note that the single-headed model performed better than multi-headed model. We tried learning with a 4-headed model, which performed slightly worse than the 2-headed one.

4.4.2 Street View House Numbers

To check the performance of the model with multi-channel data, we trained and tested the two-headed model on the Street View House Numbers (SVHN) dataset (Yuval Netzer et al., 2011), using the 32×32 cropped version, and training for 100 epochs. We

²<<https://github.com/zalandoresearch/fashion-mnist>>

Figure 4.2 – Training Curve for the Street View House Numbers 32×32 dataset

achieved a similar performance with the two-headed GAT model on the FashionMNIST dataset, with 80.72% test accuracy (with training loss and training/validation accuracy available in Fig. 4.2). This comes to show that our model works even with full colour data, as well as the confounders present in the SVHN dataset.

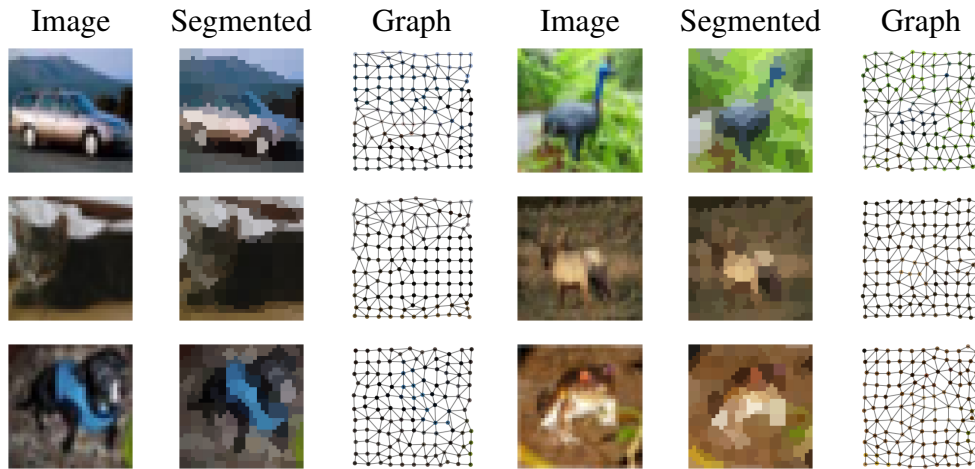
The classifier learned to prioritise the centermost superpixels (where the digits are) and to compare changes in colour tones instead of simple changes in luminosity, which allowed it to identify the internal structure of the digit. This proves that, although our model does not reach state-of-the-art performance, it can achieve relatively good accuracy with less expressive data.

4.4.3 CIFAR-10

A network with the same architecture as GAT-2Head on the CIFAR-10 dataset achieved only 45.93% accuracy on the test set – very distant from what we achieved on the MNIST and FashionMNIST datasets. The training and validation accuracies were not impressive either, being 58.61% and 53.40% respectively. As a baseline, we used the VGG11 (Simonyan; Zisserman, 2015) architecture, with and without batch normalisation, but fed the model with an image where every pixel’s colour is set to be that of its superpixel’s. We were unable to train the model without batch normalisation, whereas with batch normalisation we achieved 62.86% validation accuracy, which shows the heavy information loss during image segmentation, which is even worse during the RAG proce-

Figure 4.3 – Examples showing the loss of information in the RAG procedure when using only the average of each channel, which negatively affects the performance of the network. Source:

Author (Avelar et al., 2020)



dure.

This comparison, however, is still unfair. As the VGG model has access to the segmented image: this also approaches *geometry* information – that is, the VGG model will not only know the colour of each superpixel, but also its shape. Our GAT model only has access to the average colour and the centroid position, and does not have access to a superpixel’s shape. This can be visualised in Figure 4.3, where the VGG model has access to the middle images and GNN model only has access to the graphs on the right.

As for the model size, the VGG11 network has 132,868,840 parameters, while the GAT has only 55,364. The VGG network also consumed almost twice as much VRAM as the GAT-2Head architecture on the CIFAR images. VGG11 consumed 4,109MiB for training and 1,055MiB for testing, while the GAT model expended 1,067MiB for training and 485MiB for testing. For the sake of illustration, one of the previous state-of-the-art models on CIFAR-10 was AmoebaNet-B (Huang et al., 2019), which has 550M parameters, and was pre-trained on ImageNet before being fine-tuned on CIFAR-10.

4.5 Discussion

We provided in this chapter, to the best of our knowledge, the first use of a purely attention-based model for image classification, by interpreting an image as a collection of superpixels and designing a neural network that operates on those superpixels. Although originally we had planned as future steps for this work to incorporate the superpixel’s fully shape and colour information into the model, since we believed that to be one of the

largest drawbacks our model had when compared to traditional image classification (see Figure 4.3), there are other concerns that could be raised towards our model.

First of all, there are efficiency concerns with regards to our model architecture: while convolutional architectures often incorporate some sort of pooling to help reduce complexity as model depth increases, our models do not have any pooling layers, even though pooling layers already existed when our work was being developed (Avelar et al., 2020), such as DiffPool (Ying et al., 2018), Top-K pooling (Gao; Ji, 2019), SortPool (Zhang et al., 2018), and SAGpool (Lee; Lee; Kang, 2019). Although we believe that graph-based pooling can be used in attentional-based models for image classification, at the time of our work we did not see much necessity for it, and we believed that handling the abovementioned information loss (Fig. 4.3) was more important. In the next chapter’s Section 5.5 it will also be possible to see that pooling might not be entirely necessary, although more experiments would be warranted to decide whether it could be beneficial.

Secondly, there might be some worries that, since our model is a graph-based model whose neighbourhood aggregation sums to 1 (due to the softmax in the attention), as some authors had previously raised concerns that Graph Neural Networks worked only as a low-pass filter (NT; Maehara, 2019), suffered from over-smoothing (Li; Han; Wu, 2018), and that GCNs (another model whose neighbourhood aggregation sums to 1) are not as powerful as a Weissfeiler-Lehman test and might not be able to tell apart between two different graphs due to structural limitations of the model (Xu et al., 2019). We’ll hopefully settle these questions on this on the next Chapter’s Section 5.5, by discussing the representational power of Graph Attention Networks in specific, and attentional-image classification models in general.

Finally, due to the gap between the original publication and this thesis, our original plans of fully incorporating the fully superpixel colour and shape information in a fully-attentional model has, in a sense, already been done through the development of the Vision Transformer framework (Dosovitskiy et al., 2021). Therefore, in the following chapter we will building upon this modern architecture to fold it into our original graph-based superpixel framework.

5 FROM GATS TO VITS AND BACK: SUPERPIXEL TRANSFORMERS

To the best of our knowledge, no previous work has yet linked Vision Transformers with GNN- and Superpixel-based image classification. In this chapter we will bridge the gap between Superpixel Image Classification and Vision Transformers and put them into a framework of attentional image classification. We attempt to ablate which components are crucial for better performance and generalisability in an attempt to challenge the notion that Graph Neural Networks cannot be deep and unify both fields.

We do so by proposing a framework that generalises both the Superpixel Image Classification with Graph Attention Networks (SICGAT) model proposed in Avelar et al. (2020) and discussed in Chapter 4, as well as the Vision Transformer (ViT) model (Dosovitskiy et al., 2021). Our framework can work with multiple patch types and allows for arbitrary graph connectivities. We include tests with both grid-patches like in ViT and also extend SICGAT’s superpixels into a patch-like data structure that fully utilises the information from both the superpixel’s shape as well as the colour information of each component superpixel. We test our models with three common graph strategies from the literature (Monti et al., 2017; Avelar et al., 2020; Dosovitskiy et al., 2021; Rodrigues; Carbonera, 2024) and show that Vision Transformers might benefit from a more constrained connectivity graph between its patches. Finally, we propose an extension of the Sine-Cosine positional embedding from Vaswani et al. (2017) to an arbitrary number of dimensions, which we show to be highly beneficial for working in a superpixel environment.

We hope that these developments will allow the areas of Superpixel Image Classification to be unified with those of mainstream Computer Vision and that these models spur further investigation on different superpixel formats for future ViT work.

5.1 Related Work

While an image is usually represented as a grid of pixels, as mentioned above, if one were to work using superpixels instead, a graph representation becomes required due to the irregular connectivity patterns between superpixels, and has been widely used together with Graph Neural Networks for Superpixel Image Classification (Monti et al., 2017; Fey et al., 2018; Avelar et al., 2020; Rodrigues; Carbonera, 2024). However, to allow for such general connectivity patterns, GNN-based methods often aggregate a

node’s neighbourhood using the same weight for every neighbour. This implies that simple GNNs would have a lower representational power than convolutional neural networks, since CNNs allow for a different weight to be used for each value in the “neighbourhood” (the convolution window). In fact, a pixel-grid-based GNN was tested in Monti et al. (2017) and had a worse performance than a CNN.

The model presented in Superpixel Image Classification with Graph Attention Networks (Avelar et al., 2020), here onwards referred to as SICGAT, expanded on previous work to allow for a GNN that allows for a pseudo-convolutional representational power. This was achieved through using multiple attention heads to mediate the information transfer between nodes in a graph, which allows different “soft”-weights to be learned by the network when aggregating between neighbourhoods. Surprisingly, however, the performance of these models reached its limit with a much smaller amount of attention heads than the medium connectivity in the superpixel graph they built (which is a Region Adjacency Graph). In this chapter we will provide a generalisation of their method and show that our model has better results even than their “hard” baseline of using a VGG architecture that only has access to a superpixel-segmented image where each superpixel’s pixels have their RGB values set to their common average. Our model also overcomes the information loss they show in Figure 4.3.

Recently there has also been a comparative study on the best graph construction method for GCNs (Rodrigues; Carbonera, 2024), which showed an improved performance on Avelar et al.’s work even when using GCN: a GNN previously shown to have a lower representational power (Xu et al., 2019). They provide a thorough study showing an ablation of various parameters used to build a GNN for SIC and argue that, similar to Avelar et al. (2020), performance was capped at very shallow layer depths. In this chapter, we will also perform a thorough exploration of an attention-based GNN model which, thus, has a higher representational power than GCNs and can be at least as powerful as a Weissfeiler-Lehman test (Xu et al., 2019; Tao; Li; Wu, 2022). Through our ablation, we show that, although more layers did not always cause better performance, the layer depth of our best-performing model often reached our cap for the number of layers (12), which allows for networks four times as deep as the ones used in Avelar et al. (2020) and Rodrigues and Carbonera (2024).

Another branch that could be seen related to our work is that of using ViTs for creating superpixel segmentations from images. While ViT have recently been extensively used for image segmentation (Thisanke et al., 2023), these work are different from the

work presented here since a segmentation is the outcome of using a ViT, whereas we use it as the input for our work, using a low-resource segmentation technique in hopes to provide a better and more meaningful input for image classification. In particular, Zhu et al. (2023) propose to use a ViT for generating superpixels and call their model a Superpixel transformer, which should not be confused with our SuperPixel Transformer (SPT) framework and the contribution we provide here.

5.2 SuperPixel Transformers (SPT): A General Framework for Attentional Learning in Images

In this section, we provide the definition of our SuperPixel Transformer (SPT) framework that generalises the SICGAT model and the ViT architecture, meaning that both can be implemented within the SPT framework by simply changing some of SPT’s hyperparameters. We first start by providing a step-by-step description of the differences between SICGAT and the ViT architectures, and then we follow with a description of a concrete formalisation of our model.

5.2.1 Key differences between Superpixel Image Classification and Vision Transformers

Figure 5.1 shows a comparison of the SICGAT and ViT architectures. From this, we identify the main differences between these two approaches: (1) chunking strategy, (2) pixel information aggregation, (3) positional embeddings, (4) architectural differences between a GAT and a Transformer block, (5) differences in chunk connectivity, and (6) pooling vs virtual classification node. We discuss in depth each of these differences next.

Chunking Strategy The first and most obvious difference between the two models is the fact that while SICGAT uses a superpixel algorithm to chunk the image into superpixels, ViT chunks images into equally-sized rectangles. However, this scheme had also previously been used in other superpixel image classification algorithms, such as in MoNET (Monti et al., 2017), and the approach followed by SICGAT could easily be adapted to also use equally sized chunks. The main reason, however, why SICGAT performs better when chunking the image into superpixels instead of regular patches is due to the pixel information aggregation scheme.

Figure 5.1 – The ViT architecture (Dosovitskiy et al., 2021) (left) and the SICGAT architecture (Avelar et al., 2020) (right). Note that, apart from the main self-attention encoding layer, the adjacency graph masking and the fact that SICGAT compresses all input through averaging the channel information of the superpixel instead of having access to all pixels, both architectures are remarkably similar. Source: Author, ViT diagram based on Dosovitskiy et al. (2021)

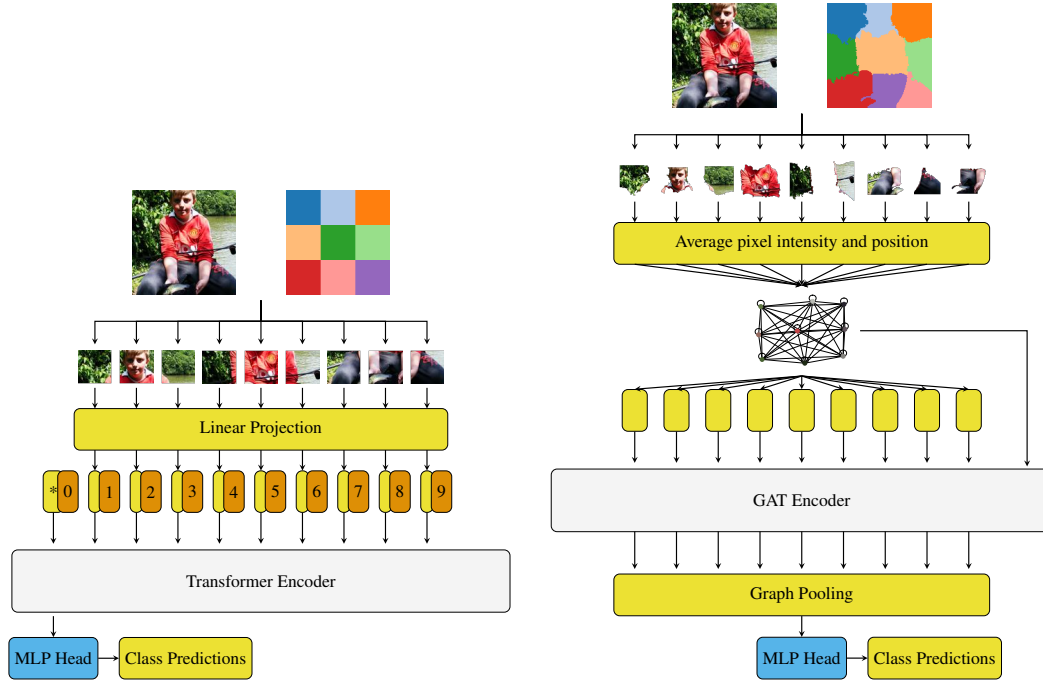
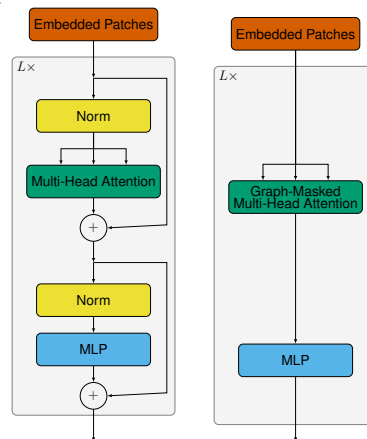


Figure 5.2 – The Transformer Encoder (Vaswani et al., 2017) used on the ViT model (Dosovitskiy et al., 2021) (left) and an equivalent view of a GAT encoder (Velickovic et al., 2018) used on the SICGAT model (Avelar et al., 2020) (right). Using an attention mask on a transformer encoder would then be equivalent to using an adjacency matrix on a GAT encoder, with the only exceptions being the inner normalisation layers, residual connections, and the specifics of the attention mechanism.



Pixel Information Aggregation Most superpixel image classification methods aggregated the information from all of a chunk’s pixels into a single value, such as taking the average value of pixel intensities around the pixel. This was also followed by SICGAT, and it was shown that it caused heavy information loss, as can be seen in Figure 4.3. ViTs avoid this by making a linear projection of the flattened pixel inputs for each chunk, which allows the network to learn to use the information from every pixel in a chunk, mitigating the information loss present in SICGAT and other methods such as MoNet.

Positional Embeddings Another difference between these approaches is how positional information is treated. Earlier superpixel image classification models such as MoNet used positional information only as a weighting factor when performing neighbourhood aggregation, simply using this information as an inductive bias for the network. SICGAT extended this by appending each node’s positional information to the node’s features, treating the positional information in the same way as its pixel information. ViTs follow similarly but with two major differences: first, instead of appending the positional information to the pixel information, ViTs follow the previous standard used by Transformers and sums the positional information with the pixel information; lastly, due to the fact that ViTs uses a fixed number of equidistant patches, it encodes the positional information directly through a patch’s index instead of as a transformation of its xy coordinates.

Architectural differences A GAT layer and a Transformer layer have significant differences in their architectural components, with GATs being, in a sense, a simpler form of a Transformer Layer, which can be seen in Figure 5.2. Firstly, The GAT layer used in SICGAT only performs multi-head attention and immediately follows it with a linear/MLP block to update a node’s information. The transformer layer instead first norms information before either the multi-head attention or MLP blocks, and then uses a residual approach to update the information instead of directly editing it, which has been shown to be an easier problem to solve and, thus, more efficient parameter-wise. Finally, the attention mechanisms of the original GAT layer and the Transformer encoder are different in their representational power (Brody; Alon; Yahav, 2022), which we will discuss more at the end of this chapter.

Chunk Connectivity Another difference is how connectivity is handled. Many of the earlier works in Superpixel Image Classification followed the same connectivity scheme as done by MoNet: a KNN graph based on chunk position. SICGAT tested their architecture only on a more restricted version of the problem, using instead a Region Ad-

jacency Graph (RAG), arguing that it was both a harder problem as well as being closed conceptually to a convolution block, since only chunks that were physically connected would be also connected in this scheme. ViTs, having originated from traditional Transformers where there is no concept of adjacency, eschew this altogether and treat the image as a fully connected graph, leaving for the attentional mechanism to route the information flow between chunks.

Pooling vs virtual classification node Finally, the last major difference between these two models is that, while SICGAT average-pools the information in all nodes before feeding it into a classification head, ViTs again followed the traditional Transformer approach of adding a virtual classification node, which is then passed to the classification head. This would be equivalent to adding a virtual graph node, which is also another commonly used way to perform graph classification in Graph Neural Networks. In this chapter we will follow the original ViT paper and work mainly with the concept of the classification, however it is worthy of note that more recent papers claim better performance by classifying and pooling over every single patch instead (Beyer; Zhai; Kolesnikov, 2022).

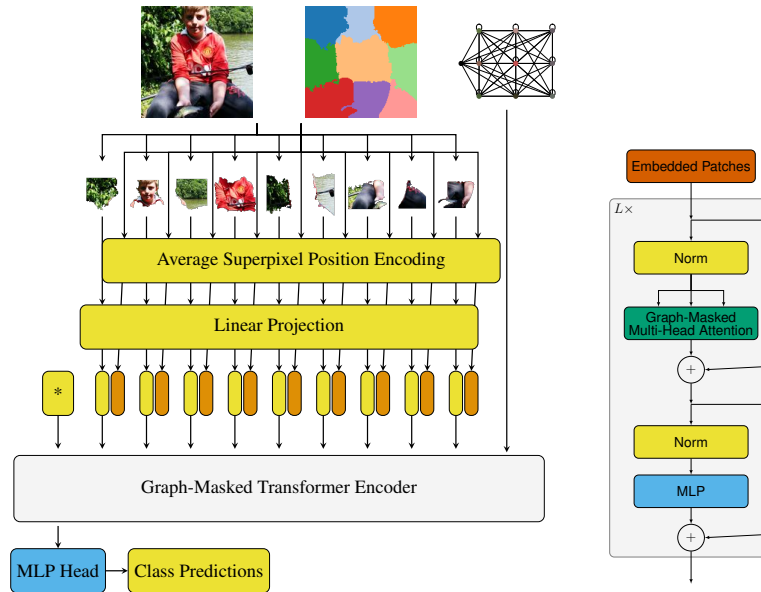
With these differences in mind, we can then build a generalisation that allows for both models to be represented under the framework, which we'll describe in the following sections.

5.2.2 The Superpixel Transformer Model

We define here our framework called Superpixel Transformers (SPT), depicted in Figure 5.3, which is an attempt at generalising (mostly attentional) SIC models and bridging the gap between these models and Vision Transformers, which see use in mainstream computer vision. Our framework is composed of 6 steps divided into preprocessing and on how to use the input for the network architecture. In the preprocessing stage, given an image I , we use a chunking strategy f_S to divide the image into its chunks/superpixels S that contain information about both the superpixel's content S_c and its position S_p on the image, then we use a connectivity strategy f_G to build a connectivity graph G between the superpixels. Then, the superpixels S and input graph G are passed first encoded through a positional encoding strategy P and a content encoding strategy C , to be passed through a self-attention module (e.g. Transformer Encoder) that will finally be used to produce a classification. We'll discuss the particular choices for each step below.

In the preprocessing part we choose two specific chunking strategies: A grid-

Figure 5.3 – Our proposed Superpixel Transformer (SPT) architecture (left) and the Graph-Masked Transformer Encoder with Graph-Masked Multi-head Attention (right). Differently from SICGAT (Avelar et al., 2020) we use the full superpixel information through the patch data structure built in Algorithm 4 and use a variety of superpixel positional encodings to build the input for our encoder network (instead of encoding the position together with the patch information in the first GAT layer). Different from ViT (Dosovitskiy et al., 2021) we allow for arbitrary superpixel algorithms and also use our model with a variety of adjacency graphs which constrain and help information flow better through the model. Also, due to the superpixel patches possibly having different shapes and positions, we extend and allow for encodings that use the xy coordinates of the superpixel/patch. Source: Author



based chunking (similar to what was done in Dosovitskiy et al. (2021)) that we define precisely in Algorithm 7 and which returns a $\mathbb{Z}^{h \times w}$ integer mask showing which superpixel each pixel belongs to, and SLICO (Achanta et al., 2010; Achanta et al., 2012) (here onwards also referred to as SLIC), which provides a similar output but follows a colour- and position-wise k-means variation to divide pixels into superpixels.

Specifically for SLICO, we used the implementation in `scikit-image` version 0.24 (Walt et al., 2014). After obtaining the mask from a superpixel strategy, such as one of the two above, we use the segment mask to construct a neighbourhood graph G and use both the image and the segment mask into an algorithm that builds out superpixel patch data structure. The data structure we use is composed of three major components: An $\mathbb{R}^{k \times 2}$ tensor containing the xy positions of each superpixel, an $\mathbb{R}^{k \times 3 \times h_{chunk} \times w_{chunk}}$ tensor that contains the superpixel’s RGB content, and an $\{0, 1\}^{k \times 1 \times h_{chunk} \times w_{chunk}}$ binary mask that contains the superpixel’s shape content. We originally used a $h_{chunk} = w_{chunk} = k_{chunk} \times S$ with $k_{chunk} = 2$ since that is the search limit on the SLIC algorithm (Achanta et al., 2010), but the SLIC implementation we used allowed superpixels with dimensions larger than $2S$, therefore we set $k_{chunk} = 3$. To allow for batching we provide a fixed-size superpixel chunk tensor, and thus we also provide a $\{0, 1\}^k$ binary mask to inform the transformer which chunks contain information. The full procedure to build this data structure can be seen in Algorithm 4. Note that while we define our model here for 2-dimensional data with 3 channels, it can easily be adapted to more dimensions and a different number of channels. For simplicity’s sake we transform any image with only a single channel to a 3-channel grayscale image.

For the graph-generation strategies we tested three strategies: 1. Using a fully connected graph FCG, similar to what a Vision Transformer uses (Dosovitskiy et al., 2021); 2. Using a graph build through a positional k-nearest neighbours KNN (Monti et al., 2017); 3. And using a Region-Adjacency graph RAG, as defined in Avelar et al. (2020). These strategies encompass most of the strategies used in previous models.

After producing our data structures it is then fed to our end-to-end model, which encode the chunk’s with single fully connected layer as is done with the original ViT (Dosovitskiy et al., 2021). Although our model can also be trained using convolutional stems (Xiao et al., 2021), we do not do so for this chapter. We test three different strategies for encoding a node’s position: 1. The original learned positional embedding used in the ViT (Dosovitskiy et al., 2021) and proposed by Devlin et al. (2019); 2. a simple Linear layer that maps the xy coordinates to the size of the embedding; 3. and an adaptation of the

Algorithm 4 Get Superpixel Patches

Require: $\text{image} \in \mathbb{R}^{C \times h \times w}$, $\text{segments} \in \mathbb{Z}^{h \times w}$, $\text{max_patches} \in \mathbb{Z}$, $\text{patch_shape} \in \mathbb{Z}^2$
Require: $\text{search_patch_size} \in \mathbb{Z}$, background_fill strategy

- 1: Initialize spixpresent , spixcenters , spixpatches , and spixpatchpresent if not provided
- 2: Fill spixpatches according to a background_fill strategy $\triangleright$ Default: all zeros
- 3: Mark segments with data: $\text{spixpresent}[:, \text{segments.max}() + 1] \leftarrow 1$
- 4: **for** $i \leftarrow 0$ to $\text{segments.max}()$ **do**
- 5: $\text{spixpos} \leftarrow$ positions where $\text{segments} = i$
- 6: $\text{spixmins}, \text{spixmaxs} \leftarrow \min(\text{spixpos}), \max(\text{spixpos})$
- 7: $\text{spixcenters}[i] \leftarrow \text{spixmins} + \frac{\text{spixmaxs} - \text{spixmins}}{2}$
- 8: $y_0, x_0 \leftarrow \lfloor (\text{spixcenters}[i] - \frac{\text{search_patch_size}}{2}) \rfloor$
- 9: $y_1, x_1 \leftarrow \lfloor (\text{spixcenters}[i] + \frac{\text{search_patch_size}}{2}) \rfloor$
- 10: $\text{spixpos} \leftarrow$ positions within box $[y_0, y_1, x_0, x_1]$
- 11: Update spixpatches by copying colour information from the i -th superpixel's component pixels
- 12: Update spixpatchpresent as a binary mask showing which pixels in spixpatches are contained in i -th the superpixel
- 13: **end for**

return $\text{spixpresent}, \text{spixcenters}, \text{spixpatches}, \text{spixpatchpresent}$

Sine-Cosine positional embedding originally used in Vaswani et al. (2017), but expanded to work with multiple dimensions, the algorithm for which can be seen in Algorithm 8, with a linear mixing layer after it. With which we then apply the a transformer encoder network using the adjacency graph to define the attention mask, with an extra classification token added which connects to all other tokens. We chose the transformer encoder architecture instead of a GAT encoder due to both more mature Transformer implementations as well as due to the fact that the Transformer Encoder includes normalisation layers which are known to help stabilise training. Finally, we extract the classification token to pass on to a classification MLP.

Our SPT framework generalises both the ViT (Dosovitskiy et al., 2021) and the SICGAT (Avelar et al., 2020) architecture. To implement a ViT on our framework, one can simply use a **square grid chunking strategy**, flattening the patch and applying a linear layer as the pixel information embedding, then using a **Fully Connected Graph (FCG) chunk connectivity** in the graph, and using **learned positional embeddings** (Devlin et al., 2019; Dosovitskiy et al., 2021), which gives us the exact configuration of the original ViT (Dosovitskiy et al., 2021).

To approximate the SICGAT architecture (Avelar et al., 2020), however, some untested additions would need to be done. Firstly, to appropriately reproduce the superpixel pixel information aggregation, we'd incorporate that as part of the **chunking**

strategy: by both performing SLIC and, instead of providing the whole chunk as input to the network, we’d aggregate the superpixels into a single average value, which would then be mapped into an embedding via a linear layer. We can use the abovementioned **Region Adjacency Graph** (RAG) for the **chunk connectivity**, and finally we’d use a **linear layer mapping the xy coordinates** of the superpixel as the **positional embedding**. The only difference, then would be the encoder type, which we believe is an easy modification to perform: by removing the normalisation layers and residual connections.

5.3 Experimental Details

We train all of our models from scratch, taking most of the hyperparameters from Steiner et al. (2022), with different values marked by *:

- Learning Rate: $1e-3$
- Learning Rate Scheduler*: We Reduce the LR on a plateau by a factor of 10 with a patience of 20% the number of epochs
- Dropout = 0.1
- Global Gradient Clipping = 1.0 for FashionMNIST and Imagenette
- Weight Decay: $1e-4$
- Optimiser: Adam (Kingma; Ba, 2015), using $\beta_1 = 0.9$ and $\beta_2 = 0.999$
- Due to instabilities on FashionMNIST, we do not backpropagate any batches that cause invalid gradients.

We evaluate our models on 4 datasets: FashionMNIST (Xiao; Rasul; Vollgraf, 2017), CIFAR10 (Krizhevsky, 2009), Imagenette (Howard; Imagenette Contributors, 2024), and Resisc45 (Cheng; Han; Lu, 2017), which are described in Table 5.2. These datasets were chosen to allow us to compare directly with our previous work and other superpixel-based baselines (FashionMNIST, CIFAR10), and to provide proof that our model can work on larger images (Imagenette, Resisc45), while keeping the compute required manageable. We trained models whose widths follow the naming convention defined by Steiner et al. (2022), which can be seen in Table 5.1.

Due to compute constraints, we have to limit the batch size, setting it as 512 for the FashionMNIST and CIFAR10 datasets, 64 for the Imagenette dataset and 32 for the

Table 5.1 – Model configurations used for ViT and SPT models defined by Steiner et al. (2022).

Size	Encoder		MLP	12-layer
	Width	Heads	Width	ViT Params
Ti	192	3	768	$\approx 6\text{M}$
S	384	6	1536	$\approx 22\text{M}$
B	768	12	3072	$\approx 86\text{M}$

Resisc dataset. We train Ti-sized models on all datasets except on Resisc45, where we train a B-sized model. We use $k=15$ for all KNN graphs. Whenever a superpixel would violate a patch’s bounds, we clip it to fit the patch by keeping the centermost pixels, and we used patch holder of size $3 \times S$.

Table 5.2 – The description of the datasets used for the Superpixel Transformer model. C,W,H stand for the number of Channels, Width, and Height of the images. * Original development split was split randomly into train and validation, † No test set provided, results are shown for validation

Dataset	Training	Validation	Test	Total	C,W,H	Classes	S
FashionMNIST (Xiao; Rasul; Vollgraf, 2017)	*54,000	*6,000	10,000	70,000	1, 28, 28	10	4
CIFAR10 (Krizhevsky, 2009)	*47,500	*2,500	10,000	60,000	3, 32, 32	10	4
Imagenette (Deng et al., 2009; Howard; Imagenette Contributors, 2024)	*8,996	*473	3,925	13,394	3, 160, 160	10	10
Resisc45 (Cheng; Han; Lu, 2017)	28350	3150	†	31500	3, 256, 256	45	10

5.4 Results

We trained Ti-sized models (Table 5.1) using all combinations of graph connectivity strategies, patch building strategies, and superpixel building strategy, for 7 possible layer sizes: 1, 2, 4, 6, 8, 10, and 12, with 12 being the original network depth defined in Steiner et al. (2022). The layer size which provided the best validation accuracy out of the 7 test is shown for each of the combinations in Tables 5.4 and 5.4. The best result of

the SLIC-based model from this table is shown in Table 5.3 as our model, even though we had a better performing model using the grid superpixel generation strategy. The ViT model (Grid-BERT-FCG combination) achieved a slightly higher but comparable validation performance to our best SLIC model on the CIFAR10 dataset and a slightly lower but comparable performance in the FashionMNIST dataset.

Our proposed graph connectivity strategies also boosted the validation performance of the Grid-based superpixel models when compared to the fully connected graphs, and all of our trained models (including the ViT) outperformed the Graph Superpixel Image Classification baselines in all datasets. We trained an SPT-Ti (12 layers, Sine-Cosine, FCG) and a ViT-Ti (12 layers, BERT, FCG) model on the Imagenette dataset to provide a result for which the image was not too small, and both models were comparable and, while no previous SIC work has provided results on higher-resolution images, we believe that due the information loss shown in Avelar et al. (2020) would make it very hard for these models to perform anywhere close to our models.

Table 5.3 – Results for the three main datasets we tested our models in. Due to compute limitations, we did not re-train models outside the SPT framework and instead use previous works’ claimed results. For our models, we limit ourselves to using only SPT models using SLIC superpixels and we use the test results of whichever model provided the best valid accuracy in our ablation study, these models are marked with *. Best and second-best values shown in **bold** and underscored, respectively. For the ViT model, we only consider its original implementation’s hyperparameters: Fully-connected graph with a BERT embedding. For the Imagenette result we use fully connected graphs with the Sine-Cosine positional embedding. For the VGG-16 architecture we use results reported in Tang et al. (2022), Mandal (2023), Kasen (2024).

Model	FashionMNIST		CIFAR10		Imagenette	
	Valid	Test	Valid	Test	Valid	Test
VGG-16 (Tang et al., 2022)	-	92.0	-	77.2	83.3	83.5
VGG-16 SLIC (Avelar et al., 2020)	-	-	-	62.9	-	-
SICGAT (Avelar et al., 2020)	-	83.1	53.4	45.9	-	-
gSLICr-GCN (l=12) (Tang et al., 2022)	-	81.9	-	59.5	-	-
bestGCN (Rodrigues; Carbonera, 2024)	-	84.2	-	58.5	-	-
ViT (Dosovitskiy et al., 2021)*	86.5	<u>85.6</u>	70.6	69.0	-	67.9
SPT-slic*	87.9	87.3	67.0	<u>63.3</u>	-	<u>67.7</u>

We trained our SPT-B model with patch size of $S \approx 10$ for about 150 on the Resisc45 dataset using a NVIDIA GTX 1070m, after which it achieved a validation accuracy of 53.5%, which only slightly lower performance to that reported for the much larger ViT-B with a patch size of 16 trained from scratch for a similar amount of hours

Table 5.4 – Ablation Study for each superpixel generation strategy S , each positional embedding P , and each graph connectivity strategy G for the FashionMNIST and CIFAR10 datasets. We mark the **best** and second best validation accuracies for each dataset. The best SLIC-based model and non-ViT grid-based is marked with *, and the original ViT model is marked with †. For the CIFAR10 dataset Linear embeddings were failing and thus were cancelled to save compute time.

S	P	G	Valid Acc (%)		Epochs		Layers	
			Best	Last	Best	Last		
FashionMNIST								
SLIC	Linear	FCG	80.0	79.2	30	32	1	
		KNN	77.2	76.5	28	32	12	
		RAG	80.2	79.8	31	32	12	
	SinCos	FCG	86.3	86.3	32	32	6	
		KNN	87.7	87.4	31	32	4	
		RAG	87.0	87.0	32	32	12	
	BERT	FCG	87.4	87.3	25	32	12	
		KNN	87.5	87.5	32	32	12	
		RAG	*87.9	87.8	31	32	4	
	Grid	Linear	FCG	10.9	9.5	3	32	6
			KNN	80.2	79.6	31	32	6
			RAG	70.5	69.5	27	32	1
SinCos		FCG	86.0	85.9	24	32	8	
		KNN	87.2	87.2	32	32	4	
		RAG	87.0	86.8	23	32	10	
BERT		FCG	†86.5	86.5	32	32	2	
		KNN	<u>*87.9</u>	87.9	32	32	8	
		RAG	87.2	87.2	31	32	2	
		CIFAR10						
SLIC	SinCos	FCG	*67.0	64.4	221	279	10	
		KNN	67.0	65.7	185	187	2	
		RAG	66.4	65.0	279	308	4	
	BERT	FCG	60.8	59.9	162	271	6	
		KNN	61.9	59.6	172	198	10	
		RAG	65.2	64.2	504	506	4	
		Grid	SinCos	FCG	70.8	69.4	388	418
KNN	72.2			71.8	450	475	12	
RAG	<u>74.3</u>			73.1	272	420	8	
BERT	FCG		†70.6	69.5	479	512	6	
	KNN		71.7	69.5	253	260	6	
	RAG		*75.5	72.2	492	512	8	

on a higher-end GPU. When testing our model on the Imagenette dataset a SLIC-based Ti model with Sine-Cosine positional embedding achieved 67.7% test accuracy while a grid-based Sine-Cosine model achieved 67.3%.

5.5 Discussion

In this chapter we provided, to the best of our knowledge, the first framework that bridges the gap between Graph-Based Image Classification and Vision Transformers by expanding on the ideas of the SICGAT (Avelar et al., 2020) and ViT (Dosovitskiy et al., 2021) models. However, we still need to address some of the issues discussed in Section 4.5 of the previous chapter.

First of all is the discussion on the feasibility of performing graph pooling on one of our architectures. Although we have not provided experiments doing so ourselves, we believe that graph-based pooling can be applied to Vision Transformers both due to there being multiple works reporting some sort of pooling for ViTs (Fan et al., 2021; Marin et al., 2023), as well as the parallels we draw between SICGAT and ViT in this chapter. However, this was left for future work as in this chapter we wanted to compare our architecture with the original version of the ViT, and also since pooling still does not seem to be as widespread in ViTs¹ as they are in CNNs, which we posit could be due to the low number of patches when compared to the number of pixels in a convolutional image, and the fact that a higher number of pixels per patch seem to correlated well with better performance (Steiner et al., 2022).

This leaves us with the point about the representational power of a GATs and Transformer encoders (Figure 5.2) with regards to the points raised before in Section 4.5 of the previous chapter and the discussion about the “power” of the attentional mechanism itself, raised in Subsection 5.2.1. Starting with the latter point of how the attentional mechanisms differ, Brody, Alon and Yahav (2022) showed that attentional mechanisms fall into two categories: **static** attentional mechanisms and **dynamic** attentional mechanisms. **Static** mechanisms are weaker than dynamic attention due to the fact that, if a key vector produces a high attentional score for a certain query vector, it would do so for all other query vectors, limiting the extent to which attention can point to different inputs depending on its queries. They also showed that the original GAT attention mechanism

¹While Fan et al. (2021) had 1501 citations at time of writing, Marin et al. (2023) only had 35, for a total of 1536, which would comprise only 3% of the 51738 citations for Dosovitskiy et al. (2021).

developed by Velickovic et al. (2018) is a **static** attentional mechanism, while their proposed GATv2 mechanism and the dot-product attention mechanism commonly used in the transformer encoders are both **dynamic** – that is, the GAT attention is less capable of attending to different keys given different queries, while the dot-product attention used in our Transformer Encoder does not have such a flaw. If one still wishes to use a GAT-like attention mechanism, we suggest that the GATv2 mechanism proposed by Brody, Alon and Yahav (2022) is used instead.

Regarding the points raised before in Section 4.5 of the previous chapter, the first thing to notice is that most of the previous work focused on the GCN architecture (Li; Han; Wu, 2018; NT; Maehara, 2019; Xu et al., 2019) which had exploded in popularity at the time. While Xu et al. (2019) acknowledges that their work does not apply directly to GATs, more recent works discuss the effects of many of these issues with respects to GATs.

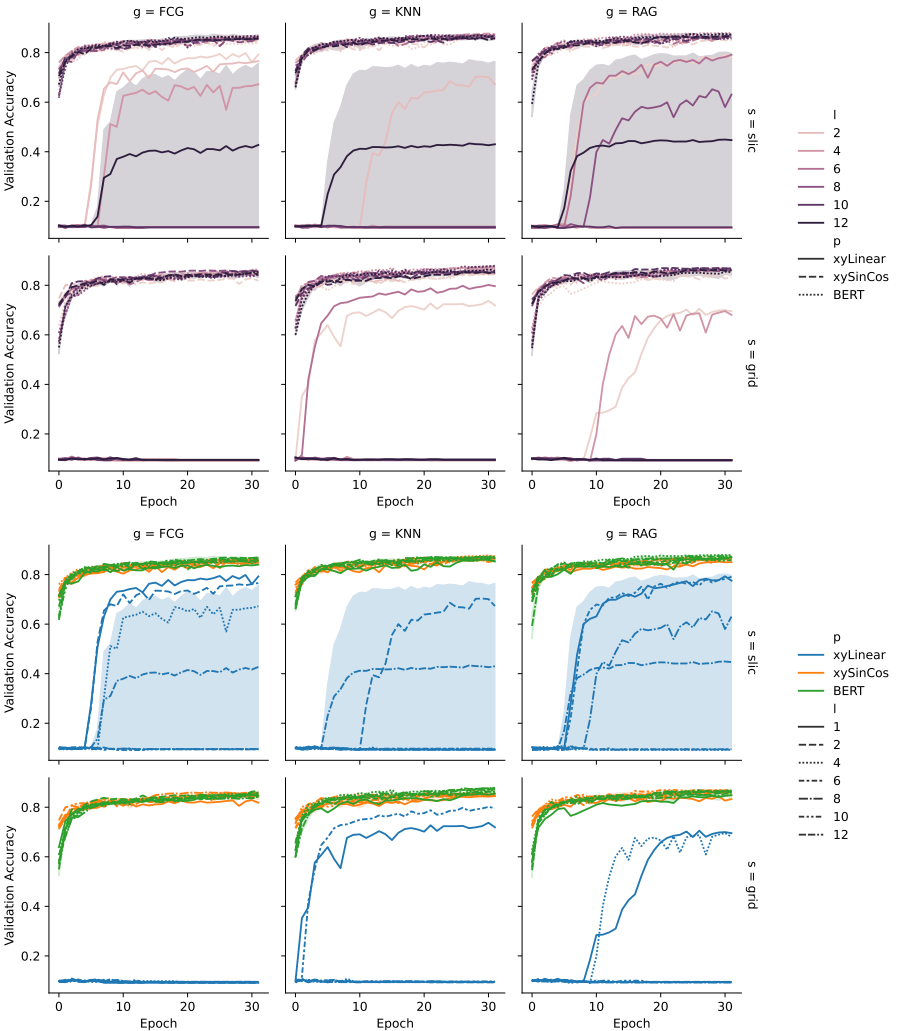
Firstly, Tao, Li and Wu (2022) express concerns that the GAT architecture might not be equivalent to a generalised Weissfeiled-Lehman test. However, while they raise this concern, they also show that by simply including the degree of each node in a GAT’s input, it would already prove equivalent to their proposed EGAT model. We believe that this might be equivalent to using graphs with a regular number of degrees such as KNN and Fully Connected Graphs, since the degree of each node is implicit.

Additionally, Zhao et al. (2023) show that, while GATs might suffer from over-smoothing, they do not necessarily do so, and we believe that a Transformer-based architecture would possibly avoid over-smoothing through other means, of which we hypothesise layer normalisation (Ba; Kiros; Hinton, 2016) and residual layers (He et al., 2016) to be prime candidates for this, as these two blocks have been a staple in the Computer Vision literature for allowing deeper convolutional networks.

Finally, while image classification can be framed as a graph classification problem, and thus be constrained by the issues described above, there is reason to believe that Transformer encoders might not suffer from the same problems, as shown by their massive adoption in the Computer Vision literature². In fact, when analysing how differently is the learning that occurs in a ViT versus a CNN, Raghu et al. (2021) discuss how residual layers/skip-connections seem to be paramount for ViTs, and that if removed from even a single point in the architecture, might cause a performance drop of 4% and partitions the representations used by the transformer before and after the block without

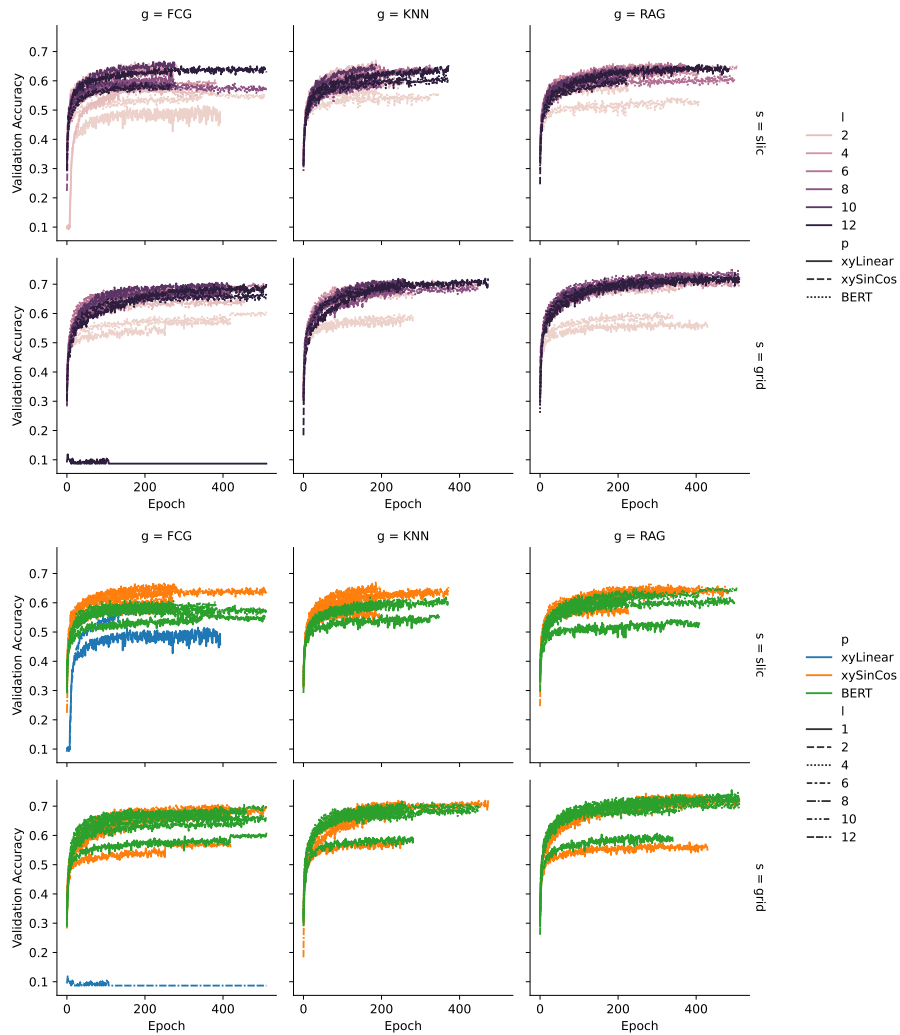
²51738 citations for Dosovitskiy et al. (2021) at the time of writing.

Figure 5.4 – Plots highlighting the Validation Accuracy throughout the epochs for all the configurations we trained on the FashionMNIST dataset. One can see from graph on the top that the number of layers seemed to have little to no impact for the FashionMNIST dataset, except for the Linear positional embedding based on the xy coordinates. From the graph on the bottom it's possible to see that the Linear embedding caused an extremely unstable training and sometimes even failed to reach similar accuracy levels.



skip connections.

Figure 5.5 – Plots highlighting the Validation Accuracy throughout the epochs for all the configurations we trained on the CIFAR10 dataset. One can see from graph on the left that the number of layers seemed to be more relevant for this dataset than the FashionMNIST dataset, possibly due to this dataset requiring a higher model capacity. Higher numbers of layers, however, often had a diminishing returns effect and sometime even negatively affected performance. For the slic model, the BERT encoding often fared worse than the Sine-Cosine embedding based on the xy coordinates. This is likely due to the fact that the superpixel embeddings have variable xy coordinates and thus the fixed learned positional encoding cannot work properly. Due to resource-sharing on HPC many models ran for fewer epochs, an analysis of time complexity and time per iteration can be see in Table B.1 and Figure B.1 where we analyse based on the FashionMNIST dataset, which ran on a more controlled environment. The linear xy positional embedding exhibited inconsistent performance and was dropped from further experiments early on.



6 CONCLUSION

In this thesis we started by providing a self-attention-, superpixel-, and graph-based model to perform image classification, showing that it was better than the baselines available at the time and, to the best of our knowledge, the first use of a purely-attentional model for image classification. Then, we proposed SuperPixel Transformer (SPT): a generalisation of the Vision Transformer model to accept arbitrary superpixel formats and graph connectivities, bridging the gap between using Vision Transformers and the Superpixel-based Image Classification literature. We've tested our models with two different Superpixel generation strategies, three graph connectivity strategies, and three positional embedding strategies, showing through this ablation study that our proposed multidimensional SinCos worked for both grid-based and slic-based superpixels, and that graph connectivity strategies produce different results in different datasets.

Furthermore, we validated our SPT models against three previous works on Superpixel Image Classification and show better performance than these baselines, showing that even when constrained to working under a superpixel framework (e.g. if one wants to transfer between euclidean and non-euclidean images as was proposed in Avelar et al. (2020), one could use a similar approach to ours and would need only to adapt the initial embedding head, being able to keep the transformer core of the model. We test our model on datasets with different image sizes and sample contents, showing that our model can adapt to these different circumstances.

Our SPT model, however, does not come without compromises. To be able to fully utilise the superpixel information an extra channel was needed during the patch embedding and the lack of search distance limitations on the SLIC implementation used (Walt et al., 2014) meant that our patches could have sizes above the $2S$ limit defined in the original SLIC/SLICO algorithm (Achanta et al., 2010). Furthermore, since the number of superpixels is dynamic and might be different than the value specified, it was necessary to provide padding on the input sequences. These issues, however, did not constitute a major resource consumption increase in our model.

Future work can address some of the abovementioned limitations of our models and can also attempt to use convolutional cores in the patch embedding heads (Xiao et al., 2021), as these would allow the model to better identify the superpixel's component parts and take full advantage of the superpixel's shape and colour information. Finally, future work could also focus on pre-training a Superpixel-based model, since pre-training is

known to be a much better strategy for training transformer-based image models (Steiner et al., 2022), this would also allow one to attempt transferring models between euclidean and non-euclidean domains as mentioned above, which our model is naturally more suitable for due to its flexible learning pattern, even though our results are only comparable to a standard ViT model.

These graph-based approaches to image classification could also be a stepping stone to the improvement of classification on non-euclidean images, such as omnidirectional images (Lee et al., 2019). While spherical convolutions are categorically different from planar convolutions and cannot be transferred without mapping images to/from the spherical/planar domains, training an image-graph classifier might allow for pre-training on planar images and transfer to spherical images. This is due to the flexibility of a graph-based approach, since the network only expects a graph and it would not matter, in theory, if the source graph was a planar or a spherical image. Thus, a model trained on a source image domain (e.g. planar images) could be fine-tuned on the target application domain (e.g. spherical images), with very little modification of the overall model (e.g., changing the embedding head from a planar embedding head to a spherical embedding head, and possibly changing the positional embeddings).

REFERENCES

ABDEL-HAMID, O.; DENG, L.; YU, D. Exploring convolutional neural network structures and optimization techniques for speech recognition. In: **Interspeech 2013**. ISCA, 2013. p. 3366–3370. Available from Internet: <https://www.isca-archive.org/interspeech_2013/abdelhamid13c_interspeech.html>.

ABDEL-HAMID, O. et al. Applying Convolutional Neural Networks concepts to hybrid NN-HMM model for speech recognition. In: **2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)**. Kyoto, Japan: IEEE, 2012. p. 4277–4280. ISBN 978-1-4673-0046-9 978-1-4673-0045-2 978-1-4673-0044-5. Available from Internet: <<http://ieeexplore.ieee.org/document/6288864/>>.

ABDEL-HAMID, O. et al. Convolutional Neural Networks for Speech Recognition. **IEEE/ACM Transactions on Audio, Speech, and Language Processing**, v. 22, n. 10, p. 1533–1545, oct. 2014. ISSN 2329-9290, 2329-9304. Available from Internet: <<http://ieeexplore.ieee.org/document/6857341/>>.

ACHANTA, R. et al. **SLIC Superpixels: SLICO**. 2010. Available from Internet: <<https://www.epfl.ch/labs/ivrl/research/slic-superpixels/#SLICO>>.

ACHANTA, R. et al. SLIC Superpixels Compared to State-of-the-Art Superpixel Methods. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 34, n. 11, p. 2274–2282, nov. 2012. ISSN 0162-8828, 2160-9292. Available from Internet: <<http://ieeexplore.ieee.org/document/6205760/>>.

ACHANTA, R.; SUSSTRUNK, S. Superpixels and Polygons Using Simple Non-iterative Clustering. In: **2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. Honolulu, HI: IEEE, 2017. p. 4895–4904. ISBN 978-1-5386-0457-1. Available from Internet: <<http://ieeexplore.ieee.org/document/8100003/>>.

ALBERT, R.; JEONG, H.; BARABÁSI, A.-L. Diameter of the World-Wide Web. **Nature**, v. 401, n. 6749, p. 130–131, sep. 1999. ISSN 0028-0836, 1476-4687. Available from Internet: <<https://www.nature.com/articles/43601>>.

Algorithmic Simplicity. **Transformer Neural Networks Derived from Scratch**. 2023. Available from Internet: <<https://www.youtube.com/watch?v=kWLed8o5M2Y>>.

ANSEL, J. et al. PyTorch 2: Faster Machine Learning Through Dynamic Python Byte-code Transformation and Graph Compilation. In: **Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2**. La Jolla CA USA: ACM, 2024. p. 929–947. ISBN 979-8-4007-0385-0. Available from Internet: <<https://dl.acm.org/doi/10.1145/3620665.3640366>>.

AUDIBERT, R. B. et al. On the Evolution of AI and Machine Learning: Towards a Meta-level Measuring and understanding Impact, Influence and Leadership at Premier AI Conferences. **Journal of Applied Logics - IfCoLog Journal of Logics and their Applications**, v. 10, n. 5, p. 693–817, nov. 2023. ISSN 2631-9810. Available from Internet: <<https://www.collegepublications.co.uk/downloads/ifcolog00061.pdf>>.

AVELAR, P. et al. Discrete and Continuous Deep Residual Learning over Graphs:. In: **Proceedings of the 13th International Conference on Agents and Artificial Intelligence**. Online Streaming, — Select a Country —: SCITEPRESS - Science and Technology Publications, 2021. p. 119–131. ISBN 978-989-758-484-8. Available from Internet: <<https://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0010231501190131>>.

AVELAR, P. H. C.; AUDIBERT, R. B.; LAMB, L. C. Measuring Ethics in AI with AI: A Methodology and Dataset Construction. In: XAVIER-JUNIOR, J. C.; RIOS, R. A. (Ed.). **Intelligent Systems**. Cham: Springer International Publishing, 2022. v. 13653, p. 370–384. ISBN 978-3-031-21685-5 978-3-031-21686-2. Series Title: Lecture Notes in Computer Science. Available from Internet: <https://link.springer.com/10.1007/978-3-031-21686-2_26>.

AVELAR, P. H. C. et al. Superpixel Image Classification with Graph Attention Networks. In: **2020 33rd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)**. Recife/Porto de Galinhas, Brazil: IEEE, 2020. p. 203–209. ISBN 978-1-7281-9274-1. Available from Internet: <<https://ieeexplore.ieee.org/document/9265983/>>.

AVELAR, P. H. D. C. et al. A Bayesian predictive analytics model for improving long range epidemic forecasting during an infection wave. **Healthcare Analytics**, v. 2, p. 100115, nov. 2022. ISSN 27724425. Available from Internet: <<https://linkinghub.elsevier.com/retrieve/pii/S2772442522000557>>.

AVELAR, P. H. d. C. da C. **Learning centrality measures with graph neural networks**. Thesis (Master's Thesis) — Universidade Federal do Rio Grande do Sul, Porto Alegre, RS, Brasil, 2019. Accepted: 2020-01-21T04:15:16Z Alternative Title: Aprendendo medidas de centralidade com redes grafo-neurais. Available from Internet: <<https://lume.ufrgs.br/handle/10183/204663>>.

AVELAR, P. H. da C. **phcavelar/graph-neural-networks**. 2018. Original-date: 2019-03-27T17:00:22Z. Available from Internet: <<https://github.com/phcavelar/graph-neural-networks>>.

AVELAR, P. H. da C.; TAVARES, A. R.; LAMB, L. C. Is an Image Also Worth 16x16=256 Superpixels? A framework for attentional image classification. **Neural Computing and Applications**, in review, 2024. ISSN 0941-0643, 1433-3058.

BA, J. L.; KIROS, J. R.; HINTON, G. E. **Layer Normalization**. arXiv, 2016. ArXiv:1607.06450 [stat]. Available from Internet: <<http://arxiv.org/abs/1607.06450>>.

BATTAGLIA, P. W. et al. **Relational inductive biases, deep learning, and graph networks**. arXiv, 2018. ArXiv:1806.01261 [cs, stat]. Available from Internet: <<http://arxiv.org/abs/1806.01261>>.

BERGH, M. V. D. et al. SEEDS: Superpixels Extracted via Energy-Driven Sampling. In: HUTCHISON, D. et al. (Ed.). **Computer Vision – ECCV 2012**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. v. 7578, p. 13–26. ISBN 978-3-642-33785-7 978-3-642-33786-4. Series Title: Lecture Notes in Computer Science. Available from Internet: <http://link.springer.com/10.1007/978-3-642-33786-4_2>.

BEYER, L.; ZHAI, X.; KOLESNIKOV, A. **Better plain ViT baselines for ImageNet-1k**. arXiv, 2022. ArXiv:2205.01580 [cs]. Available from Internet: <<http://arxiv.org/abs/2205.01580>>.

BIANCHINI, M.; GORI, M.; SCARSELLI, F. Processing directed acyclic graphs with recursive neural networks. **IEEE Transactions on Neural Networks**, v. 12, n. 6, p. 1464–1470, nov. 2001. ISSN 10459227. Available from Internet: <<http://ieeexplore.ieee.org/document/963781/>>.

BIANCHINI, M.; GORI, M.; SCARSELLI, F. Recursive Processing of Directed Acyclic Graphs. In: TAYLOR, J. G.; TAGLIAFERRI, R.; MARINARO, M. (Ed.). **Neural Nets WIRN Vietri-01**. London: Springer London, 2002. p. 96–101. ISBN 978-1-85233-505-2 978-1-4471-0219-9. Series Title: Perspectives in Neural Computing. Available from Internet: <http://link.springer.com/10.1007/978-1-4471-0219-9_6>.

BIANCHINI, M. et al. Recursive neural networks learn to localize faces. **Pattern Recognition Letters**, v. 26, n. 12, p. 1885–1895, sep. 2005. ISSN 01678655. Available from Internet: <<https://linkinghub.elsevier.com/retrieve/pii/S0167865505000656>>.

BRODY, S.; ALON, U.; YAHAV, E. How Attentive are Graph Attention Networks? In: **The Tenth International Conference on Learning Representations, ICLR 2022**. Virtual Event: OpenReview.net, 2022. Available from Internet: <<https://openreview.net/forum?id=F72ximsx7C1>>.

BROWN, T. B. et al. Language Models are Few-Shot Learners. In: LAROCHELLE, H. et al. (Ed.). **Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual**. [s.n.], 2020. Available from Internet: <<https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html>>.

CHEN, J. et al. Transformer for one stop interpretable cell type annotation. **Nature Communications**, v. 14, n. 1, p. 223, jan. 2023. ISSN 2041-1723. Available from Internet: <<https://www.nature.com/articles/s41467-023-35923-4>>.

CHENG, G.; HAN, J.; LU, X. Remote Sensing Image Scene Classification: Benchmark and State of the Art. **Proceedings of the IEEE**, v. 105, n. 10, p. 1865–1883, oct. 2017. ISSN 0018-9219, 1558-2256. Available from Internet: <<http://ieeexplore.ieee.org/document/7891544/>>.

CHO, A. et al. **Transformer Explainer: LLM Transformer Model Visually Explained**. 2024. Available from Internet: <<https://poloclub.github.io/transformer-explainer/>>.

CUI, H. et al. **scGPT: Towards Building a Foundation Model for Single-Cell Multi-omics Using Generative AI**. [S.l.], 2023. Available from Internet: <<http://biorxiv.org/lookup/doi/10.1101/2023.04.30.538439>>.

DANEL, T. et al. **Spatial Graph Convolutional Networks**. arXiv, 2020. ArXiv:1909.05310 [cs, stat]. Available from Internet: <<http://arxiv.org/abs/1909.05310>>.

DEFFERRARD, M.; BRESSON, X.; VANDERGHEYNST, P. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In: LEE, D. D. et al. (Ed.). **Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain**.

[s.n.], 2016. p. 3837–3845. Available from Internet: <<https://proceedings.neurips.cc/paper/2016/hash/04df4d434d481c5bb723be1b6df1ee65-Abstract.html>>.

DENG, J. et al. ImageNet: A large-scale hierarchical image database. In: **2009 IEEE Conference on Computer Vision and Pattern Recognition**. Miami, FL: IEEE, 2009. p. 248–255. ISBN 978-1-4244-3992-8. Available from Internet: <<https://ieeexplore.ieee.org/document/5206848/>>.

DEVLIN, J. et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: BURSTEIN, J.; DORAN, C.; SOLORIO, T. (Ed.). **Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019**. Minneapolis, MN, USA: Association for Computational Linguistics, 2019. v. 1, p. 4171–4186. Available from Internet: <<http://aclweb.org/anthology/N19-1423>>.

DOSOVITSKIY, A. et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In: **9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021**. OpenReview.net, 2021. Available from Internet: <<https://openreview.net/forum?id=YicbFdNTTy>>.

DOSOVITSKIY, A. et al. FlowNet: Learning Optical Flow with Convolutional Networks. In: **2015 IEEE International Conference on Computer Vision (ICCV)**. Santiago: IEEE, 2015. p. 2758–2766. ISBN 978-1-4673-8391-2. Available from Internet: <<https://ieeexplore.ieee.org/document/7410673/>>.

DREYFUS, S. The computational solution of optimal control problems with time lag. **IEEE Transactions on Automatic Control**, v. 18, n. 4, p. 383–385, aug. 1973. ISSN 0018-9286. Available from Internet: <<http://ieeexplore.ieee.org/document/1100330/>>.

DUVENAUD, D. et al. Convolutional Networks on Graphs for Learning Molecular Fingerprints. In: CORTES, C. et al. (Ed.). **Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada**. [s.n.], 2015. p. 2224–2232. Available from Internet: <<https://proceedings.neurips.cc/paper/2015/hash/f9be311e65d81a9ad8150a60844bb94c-Abstract.html>>.

DWIVEDI, V. P.; BRESSON, X. **A Generalization of Transformer Networks to Graphs**. arXiv, 2021. ArXiv:2012.09699 [cs]. Available from Internet: <<http://arxiv.org/abs/2012.09699>>.

FAN, H. et al. Multiscale Vision Transformers. In: **2021 IEEE/CVF International Conference on Computer Vision (ICCV)**. Montreal, QC, Canada: IEEE, 2021. p. 6804–6815. ISBN 978-1-6654-2812-5. Available from Internet: <<https://ieeexplore.ieee.org/document/9710800/>>.

FEY, M.; LENSSEN, J. E. Fast Graph Representation Learning with PyTorch Geometric. In: **Representation Learning on Graphs and Manifolds, ICLR 2019 Workshop**. New Orleans, Louisiana, United States: OpenReview.net, 2019. Available from Internet: <<https://rllgm.github.io/papers/2.pdf>>.

FEY, M. et al. SplineCNN: Fast Geometric Deep Learning with Continuous B-Spline Kernels. In: **2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition**. Salt Lake City, UT: IEEE, 2018. p. 869–877. ISBN 978-1-5386-6420-9. Available from Internet: <<https://ieeexplore.ieee.org/document/8578195/>>.

FUKUSHIMA, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. **Biological Cybernetics**, v. 36, n. 4, p. 193–202, abr. 1980. ISSN 0340-1200, 1432-0770. Available from Internet: <<http://link.springer.com/10.1007/BF00344251>>.

FUNG, V. et al. Benchmarking graph neural networks for materials chemistry. **npj Computational Materials**, v. 7, n. 1, p. 84, jun. 2021. ISSN 2057-3960. Available from Internet: <<https://www.nature.com/articles/s41524-021-00554-0>>.

GAO, H.; JI, S. Graph U-Nets. In: CHAUDHURI, K.; SALAKHUTDINOV, R. (Ed.). **Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA**. PMLR, 2019. (Proceedings of Machine Learning Research, v. 97), p. 2083–2092. Available from Internet: <<http://proceedings.mlr.press/v97/gao19a.html>>.

GEHRING, J. et al. Convolutional Sequence to Sequence Learning. In: PRECUP, D.; TEH, Y. W. (Ed.). **Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017**. PMLR, 2017. (Proceedings of Machine Learning Research, v. 70), p. 1243–1252. Available from Internet: <<http://proceedings.mlr.press/v70/gehring17a.html>>.

GILMER, J. et al. Neural Message Passing for Quantum Chemistry. In: PRECUP, D.; TEH, Y. W. (Ed.). **Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017**. PMLR, 2017. (Proceedings of Machine Learning Research, v. 70), p. 1263–1272. Available from Internet: <<http://proceedings.mlr.press/v70/gilmer17a.html>>.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep learning**. Cambridge, Mass: The MIT press, 2016. (Adaptive computation and machine learning). ISBN 978-0-262-03561-3. Available from Internet: <<https://www.deeplearningbook.org/>>.

GRAVES, A.; FERNÁNDEZ, S.; SCHMIDHUBER, J. Bidirectional LSTM Networks for Improved Phoneme Classification and Recognition. In: DUCH, W. et al. (Ed.). **Artificial Neural Networks: Formal Models and Their Applications - ICANN 2005, 15th International Conference, Warsaw, Poland, September 11-15, 2005, Proceedings, Part II**. Warsaw, Poland: Springer, 2005. (Lecture Notes in Computer Science, v. 3697), p. 799–804. Available from Internet: <https://doi.org/10.1007/11550907_126>.

HAMILTON, W. L.; YING, Z.; LESKOVEC, J. Inductive Representation Learning on Large Graphs. In: GUYON, I. et al. (Ed.). **Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA**. [s.n.], 2017. p. 1024–1034. Available from Internet: <<https://proceedings.neurips.cc/paper/2017/hash/5dd9db5e033da9c6fb5ba83c7a7e9a9-Abstract.html>>.

HE, K. et al. Deep Residual Learning for Image Recognition. In: **2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. Las Vegas, NV, USA: IEEE, 2016. p. 770–778. ISBN 978-1-4673-8851-1. Available from Internet: <<http://ieeexplore.ieee.org/document/7780459/>>.

HEBB, D. **The Organization of Behavior**. 0. ed. New York: Wiley, 1949.

HEBB, D. **The Organization of Behavior**. 0. ed. Psychology Press, 2005. ISBN 978-1-4106-1240-3. Available from Internet: <<https://www.taylorfrancis.com/books/9781135631918>>.

HONDA, S.; SHI, S.; UEDA, H. R. SMILES Transformer: Pre-trained Molecular Fingerprint for Low Data Drug Discovery. 2019. Publisher: arXiv Version Number: 1. Available from Internet: <<https://arxiv.org/abs/1911.04738>>.

HOWARD, J.; Imagenette Contributors. **fastai/imagenette**. fast.ai, 2024. Original-date: 2019-03-06T01:58:45Z. Available from Internet: <<https://github.com/fastai/imagenette>>.

HUANG, Y. et al. GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism. In: WALLACH, H. M. et al. (Ed.). **Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada**. [s.n.], 2019. p. 103–112. Available from Internet: <<https://proceedings.neurips.cc/paper/2019/hash/093f65e080a295f8076b1c5722a46aa2-Abstract.html>>.

IOFFE, S.; SZEGEDY, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In: BACH, F. R.; BLEI, D. M. (Ed.). **Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015**. JMLR.org, 2015. (JMLR Workshop and Conference Proceedings, v. 37), p. 448–456. Available from Internet: <<http://proceedings.mlr.press/v37/ioffe15.html>>.

KAISER, L.; BENGIO, S. Can Active Memory Replace Attention? In: LEE, D. D. et al. (Ed.). **Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain**. [s.n.], 2016. p. 3774–3782. Available from Internet: <<https://proceedings.neurips.cc/paper/2016/hash/fb8feff253bb6c834deb61ec76baa893-Abstract.html>>.

KALCHBRENNER, N. et al. **Neural Machine Translation in Linear Time**. arXiv, 2017. ArXiv:1610.10099 [cs]. Available from Internet: <<http://arxiv.org/abs/1610.10099>>.

KARPATY, A. et al. Large-Scale Video Classification with Convolutional Neural Networks. In: **2014 IEEE Conference on Computer Vision and Pattern Recognition**. Columbus, OH, USA: IEEE, 2014. p. 1725–1732. ISBN 978-1-4799-5118-5. Available from Internet: <<http://ieeexplore.ieee.org/document/6909619>>.

Kasen. **VGG model construction and Imagenette image recognition based on PyTorch (in Mandarin)**. 2024. Available from Internet: <<https://blog.imkasen.com/pytorch-vgg16/>>.

KINGMA, D. P.; BA, J. Adam: A Method for Stochastic Optimization. In: BENGIO, Y.; LECUN, Y. (Ed.). **3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings**. [s.n.], 2015. Available from Internet: <<http://arxiv.org/abs/1412.6980>>.

KIPF, T. N.; WELLING, M. Semi-Supervised Classification with Graph Convolutional Networks. In: **5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings**. OpenReview.net, 2017. Available from Internet: <<https://openreview.net/forum?id=SJU4ayYgl>>.

KIRILLOV, A. et al. Segment Anything. In: **2023 IEEE/CVF International Conference on Computer Vision (ICCV)**. Paris, France: IEEE, 2023. p. 3992–4003. ISBN 979-8-3503-0718-4. Available from Internet: <<https://ieeexplore.ieee.org/document/10378323/>>.

KRIZHEVSKY, A. **Learning multiple layers of features from tiny images**. Dissertation (Master) — University of Toronto, Toronto, Canada, abr. 2009. Available from Internet: <<http://www.cs.utoronto.ca/~kriz/learning-features-2009-TR.pdf>>.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. ImageNet Classification with Deep Convolutional Neural Networks. In: BARTLETT, P. L. et al. (Ed.). **Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States**. [s.n.], 2012. p. 1106–1114. Available from Internet: <<https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html>>.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. ImageNet classification with deep convolutional neural networks. **Communications of the ACM**, v. 60, n. 6, p. 84–90, may 2017. ISSN 0001-0782, 1557-7317. Available from Internet: <<https://dl.acm.org/doi/10.1145/3065386>>.

LAMB, L. C. et al. Graph Neural Networks Meet Neural-Symbolic Computing: A Survey and Perspective. In: **Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence**. Yokohama, Japan: International Joint Conferences on Artificial Intelligence Organization, 2020. p. 4877–4884. ISBN 978-0-9992411-6-5. Available from Internet: <<https://www.ijcai.org/proceedings/2020/679>>.

LECUN, Y. Une procédure d'apprentissage pour réseau à seuil asymétrique. **Proceedings of Cognitiva** 85, p. 599–604, 1985.

LECUN, Y. A Theoretical Framework for Back-Propagation. In: TOURETZKY, D.; HINTON, G. E. (Ed.). **Proceedings of the 1988 Connectionist Models Summer School**. CMU, Pittsburgh, Pa: Morgan Kaufmann, 1988. p. 21–28.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **Nature**, v. 521, n. 7553, p. 436–444, may 2015. ISSN 0028-0836, 1476-4687. Available from Internet: <<https://www.nature.com/articles/nature14539>>.

LECUN, Y. et al. Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, v. 86, n. 11, p. 2278–2324, nov. 1998. ISSN 00189219. Available from Internet: <<http://ieeexplore.ieee.org/document/726791/>>.

LEE, J.; LEE, I.; KANG, J. Self-Attention Graph Pooling. In: CHAUDHURI, K.; SALAKHUTDINOV, R. (Ed.). **Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA**. PMLR, 2019. (Proceedings of Machine Learning Research, v. 97), p. 3734–3743. Available from Internet: <<http://proceedings.mlr.press/v97/lee19c.html>>.

LEE, Y. et al. SpherePHD: Applying CNNs on a Spherical PolyHeDron Representation of 360° Images. In: **2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)**. Long Beach, CA, USA: IEEE, 2019. p. 9173–9181. ISBN 978-1-7281-3293-8. Available from Internet: <<https://ieeexplore.ieee.org/document/8954175/>>.

LEMOS, H. et al. Neural-Symbolic Relational Reasoning on Graph Models: Effective Link Inference and Computation from Knowledge Bases. In: FARKAŠ, I.; MASULLI, P.; WERMTER, S. (Ed.). **Artificial Neural Networks and Machine Learning – ICANN 2020**. Cham: Springer International Publishing, 2020. v. 12396, p. 647–659. ISBN 978-3-030-61608-3 978-3-030-61609-0. Series Title: Lecture Notes in Computer Science. Available from Internet: <https://link.springer.com/10.1007/978-3-030-61609-0_51>.

LEMOS, H. et al. Graph Colouring Meets Deep Learning: Effective Graph Neural Network Models for Combinatorial Problems. In: **2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)**. Portland, OR, USA: IEEE, 2019. p. 879–885. ISBN 978-1-7281-3798-8. Available from Internet: <<https://ieeexplore.ieee.org/document/8995250/>>.

LI, H. et al. A convolutional neural network cascade for face detection. In: **2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. Boston, MA, USA: IEEE, 2015. p. 5325–5334. ISBN 978-1-4673-6964-0. Available from Internet: <<http://ieeexplore.ieee.org/document/7299170/>>.

LI, M. et al. DGL-LifeSci: An Open-Source Toolkit for Deep Learning on Graphs in Life Science. **ACS Omega**, v. 6, n. 41, p. 27233–27238, oct. 2021. ISSN 2470-1343, 2470-1343. Available from Internet: <<https://pubs.acs.org/doi/10.1021/acsomega.1c04017>>.

LI, Q.; HAN, Z.; WU, X.-m. Deeper Insights Into Graph Convolutional Networks for Semi-Supervised Learning. **Proceedings of the AAAI Conference on Artificial Intelligence**, v. 32, n. 1, abr. 2018. ISSN 2374-3468, 2159-5399. Available from Internet: <<https://ojs.aaai.org/index.php/AAAI/article/view/11604>>.

LINNAINMAA, S. **The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors**. Thesis (Master's Thesis) — Univ. Helsinki, Helsinki, Finland, 1970.

LIU, Z. et al. A ConvNet for the 2020s. In: **2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)**. New Orleans, LA, USA: IEEE, 2022. p. 11966–11976. ISBN 978-1-6654-6946-3. Available from Internet: <<https://ieeexplore.ieee.org/document/9879745/>>.

MANDAL, S. **VGG16 on MNIST and Fashion_MNIST**. 2023. Available from Internet: <<https://kaggle.com/code/shreyasi2002/vgg16-on-mnist-and-fashion-mnist>>.

MARIN, D. et al. Token Pooling in Vision Transformers for Image Classification. In: **2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)**. Waikoloa, HI, USA: IEEE, 2023. p. 12–21. ISBN 978-1-6654-9346-8. Available from Internet: <<https://ieeexplore.ieee.org/document/10030157/>>.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. **The Bulletin of Mathematical Biophysics**, v. 5, n. 4, p. 115–133, dec. 1943. ISSN 0007-4985, 1522-9602. Available from Internet: <<http://link.springer.com/10.1007/BF02478259>>.

MONTI, F. et al. Geometric Deep Learning on Graphs and Manifolds Using Mixture Model CNNs. In: **2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. Honolulu, HI: IEEE, 2017. p. 5425–5434. ISBN 978-1-5386-0457-1. Available from Internet: <<http://ieeexplore.ieee.org/document/8100059/>>.

NATHANI, D. et al. Learning Attention-based Embeddings for Relation Prediction in Knowledge Graphs. In: **Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics**. Florence, Italy: Association for Computational Linguistics, 2019. p. 4710–4723. Available from Internet: <<https://www.aclweb.org/anthology/P19-1466>>.

NT, H.; MAEHARA, T. **Revisiting Graph Neural Networks: All We Have is Low-Pass Filters**. arXiv, 2019. ArXiv:1905.09550 [stat]. Available from Internet: <<http://arxiv.org/abs/1905.09550>>.

OLAH, C. **Conv Nets: A Modular Perspective - colah's blog**. 2014. Available from Internet: <<https://colah.github.io/posts/2014-07-Conv-Nets-Modular/>>.

OPENAI et al. **GPT-4 Technical Report**. arXiv, 2023. ArXiv:2303.08774 [cs]. Available from Internet: <<http://arxiv.org/abs/2303.08774>>.

PASZKE, A. et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: WALLACH, H. M. et al. (Ed.). **Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada**. [s.n.], 2019. p. 8024–8035. Available from Internet: <<https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html>>.

PEDREGOSA, F. et al. Scikit-learn: Machine Learning in Python. **J. Mach. Learn. Res.**, v. 12, p. 2825–2830, 2011. Available from Internet: <<https://dl.acm.org/doi/10.5555/1953048.2078195>>.

PIMENTA, P. F. **Solving the Kidney Exchange Problem using Graph Neural Networks Trained with No Supervision**. Thesis (BSc Dissertation) — Universidade Federal do Rio Grande do Sul, Porto Alegre, RS, Brasil, 2023. Accepted:. Available from Internet: <<https://lume.ufrgs.br/handle/10183/204663>>.

PIMENTA, P. F.; AVELAR, P. H. C.; LAMB, L. C. Graph Neural Networks with No Supervision and Heuristics for the Kidney-Exchange Problem. In: **2023 IEEE 35th International Conference on Tools with Artificial Intelligence (ICTAI)**. Atlanta, GA, USA: IEEE, 2023. p. 258–262. ISBN 979-8-3503-4273-4. Available from Internet: <<https://ieeexplore.ieee.org/document/10356509/>>.

PIMENTA, P. F.; AVELAR, P. H. C.; LAMB, L. C. Solving the kidney exchange problem via graph neural networks with no supervision. **Neural Computing and Applications**, may 2024. ISSN 0941-0643, 1433-3058. Available from Internet: <<https://link.springer.com/10.1007/s00521-024-09887-5>>.

PRATES, M. et al. Learning to Solve NP-Complete Problems: A Graph Neural Network for Decision TSP. **Proceedings of the AAAI Conference on Artificial Intelligence**, v. 33, p. 4731–4738, jul. 2019. ISSN 2374-3468, 2159-5399. Available from Internet: <<https://aaai.org/ojs/index.php/AAAI/article/view/4399>>.

RADFORD, A. et al. Learning Transferable Visual Models From Natural Language Supervision. In: MEILA, M.; ZHANG, T. (Ed.). **Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event**. PMLR, 2021. (Proceedings of Machine Learning Research, v. 139), p. 8748–8763. Available from Internet: <<http://proceedings.mlr.press/v139/radford21a.html>>.

RAGHU, M. et al. Do Vision Transformers See Like Convolutional Neural Networks? In: RANZATO, M. et al. (Ed.). **Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual**. [s.n.], 2021. p. 12116–12128. Available from Internet: <<https://proceedings.neurips.cc/paper/2021/hash/652cf38361a209088302ba2b8b7f51e0-Abstract.html>>.

RAMESH, A. et al. Zero-Shot Text-to-Image Generation. In: MEILA, M.; ZHANG, T. (Ed.). **Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event**. PMLR, 2021. (Proceedings of Machine Learning Research, v. 139), p. 8821–8831. Available from Internet: <<http://proceedings.mlr.press/v139/ramesh21a.html>>.

RODRIGUES, J.; CARBONERA, J. Graph Convolutional Networks for Image Classification: Comparing Approaches for Building Graphs from Images. In: **Proceedings of the 26th International Conference on Enterprise Information Systems**. Angers, France: SCITEPRESS - Science and Technology Publications, 2024. v. 1, p. 437–446. ISBN 978-989-758-692-7. Available from Internet: <<https://www.scitepress.org/Papers/2024/122632/122632.pdf>>.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. **Nature**, v. 323, n. 6088, p. 533–536, oct. 1986. ISSN 0028-0836, 1476-4687. Available from Internet: <<https://www.nature.com/articles/323533a0>>.

SANCHEZ-LENGELING, B. et al. A Gentle Introduction to Graph Neural Networks. **Distill**, v. 6, n. 8, p. 10.23915/distill.00033, aug. 2021. ISSN 2476-0757. Available from Internet: <<https://distill.pub/2021/gnn-intro>>.

SCARSELLI, F. et al. The Graph Neural Network Model. **IEEE Transactions on Neural Networks**, v. 20, n. 1, p. 61–80, jan. 2009. ISSN 1045-9227, 1941-0093. Available from Internet: <<http://ieeexplore.ieee.org/document/4700287/>>.

SCHMIDHUBER, J. Deep learning in neural networks: An overview. **Neural Networks**, v. 61, p. 85–117, jan. 2015. ISSN 08936080. Available from Internet: <<https://linkinghub.elsevier.com/retrieve/pii/S0893608014002135>>.

SCHUSTER, M.; PALIWAL, K. Bidirectional recurrent neural networks. **IEEE Transactions on Signal Processing**, v. 45, n. 11, p. 2673–2681, nov. 1997. ISSN 1053587X. Available from Internet: <<http://ieeexplore.ieee.org/document/650093/>>.

SELSAM, D. et al. Learning a SAT Solver from Single-Bit Supervision. In: **7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019**. OpenReview.net, 2019. Available from Internet: <https://openreview.net/forum?id=HJMC_iA5tm>.

SIMONYAN, K.; ZISSERMAN, A. Two-Stream Convolutional Networks for Action Recognition in Videos. In: GHAHRAMANI, Z. et al. (Ed.). **Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada**. [s.n.], 2014. p. 568–576. Available from Internet: <<https://proceedings.neurips.cc/paper/2014/hash/00ec53c4682d36f5c4359f4ae7bd7ba1-Abstract.html>>.

SIMONYAN, K.; ZISSERMAN, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. In: BENGIO, Y.; LECUN, Y. (Ed.). **3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings**. [s.n.], 2015. Available from Internet: <<http://arxiv.org/abs/1409.1556>>.

STEINER, A. P. et al. How to train your ViT? Data, Augmentation, and Regularization in Vision Transformers. **Transactions on Machine Learning Research**, 2022. ISSN 2835-8856. Available from Internet: <<https://openreview.net/forum?id=4nPswr1KcP>>.

STUTZ, D.; HERMANS, A.; LEIBE, B. Superpixels: An evaluation of the state-of-the-art. **Computer Vision and Image Understanding**, v. 166, p. 1–27, jan. 2018. ISSN 10773142. Available from Internet: <<https://linkinghub.elsevier.com/retrieve/pii/S1077314217300589>>.

SUN, Z.-Z. et al. Generative tensor network classification model for supervised machine learning. **Physical Review B**, v. 101, n. 7, p. 075135, feb. 2020. ISSN 2469-9950, 2469-9969. Available from Internet: <<https://link.aps.org/doi/10.1103/PhysRevB.101.075135>>.

TANG, T. et al. Image Classification Based on Deep Graph Convolutional Networks. In: **2022 IEEE 9th International Conference on Data Science and Advanced Analytics (DSAA)**. Shenzhen, China: IEEE, 2022. p. 1–6. ISBN 978-1-6654-7330-9. Available from Internet: <<https://ieeexplore.ieee.org/document/10032329/>>.

TAO, Y.; LI, Y.; WU, Z. Revisiting Attention-Based Graph Neural Networks for Graph Classification. In: RUDOLPH, G. et al. (Ed.). **Parallel Problem Solving from Nature – PPSN XVII**. Cham: Springer International Publishing, 2022. v. 13398, p. 442–458. ISBN 978-3-031-14713-5 978-3-031-14714-2. Series Title: Lecture Notes in Computer Science. Available from Internet: <https://link.springer.com/10.1007/978-3-031-14714-2_31>.

THISANKE, H. et al. Semantic segmentation using Vision Transformers: A survey. **Engineering Applications of Artificial Intelligence**, v. 126, p. 106669, nov. 2023. ISSN 09521976. Available from Internet: <<https://linkinghub.elsevier.com/retrieve/pii/S0952197623008539>>.

Torch.Sparse Contributors. **torch.sparse — PyTorch 2.3 documentation**. 2024. Available from Internet: <<https://pytorch.org/docs/2.3/sparse.html>>.

TRAN, D. et al. Learning Spatiotemporal Features with 3D Convolutional Networks. In: **2015 IEEE International Conference on Computer Vision (ICCV)**. Santiago, Chile: IEEE, 2015. p. 4489–4497. ISBN 978-1-4673-8391-2. Available from Internet: <<http://ieeexplore.ieee.org/document/7410867/>>.

UDANDARAO, V. et al. **No "Zero-Shot" Without Exponential Data: Pretraining Concept Frequency Determines Multimodal Model Performance**. arXiv, 2024. ArXiv:2404.04125 [cs]. Available from Internet: <<http://arxiv.org/abs/2404.04125>>.

VASWANI, A. et al. Attention is All you Need. In: GUYON, I. et al. (Ed.). **Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA**. Long Beach, CA, USA: [s.n.], 2017. p. 5998–6008. Available from Internet: <<https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html>>.

VELICKOVIC, P. et al. Graph Attention Networks. In: **6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings**. OpenReview.net, 2018. Available from Internet: <<https://openreview.net/forum?id=rJXMpikCZ>>.

VELIČKOVIĆ, P. et al. **Graph Attention Networks**. arXiv, 2017. ArXiv:1710.10903 [stat]. Available from Internet: <<http://arxiv.org/abs/1710.10903>>.

VELIČKOVIĆ, P. et al. Deep Graph Infomax. In: **International Conference on Learning Representations**. [s.n.], 2019. Available from Internet: <<https://openreview.net/forum?id=rklz9iAcKQ>>.

VIRTANEN, P. et al. SciPy 1.0: fundamental algorithms for scientific computing in Python. **Nature Methods**, v. 17, n. 3, p. 261–272, mar. 2020. ISSN 1548-7091, 1548-7105. Available from Internet: <<https://www.nature.com/articles/s41592-019-0686-2>>.

WALT, S. V. D. et al. scikit-image: image processing in Python. **PeerJ**, v. 2, p. e453, jun. 2014. ISSN 2167-8359. Available from Internet: <<https://peerj.com/articles/453>>.

WANG, M. et al. **Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks**. arXiv, 2020. ArXiv:1909.01315 [cs]. Available from Internet: <<http://arxiv.org/abs/1909.01315>>.

WANG, S. et al. SMILES-BERT: Large Scale Unsupervised Pre-Training for Molecular Property Prediction. In: **Proceedings of the 10th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics**. Niagara Falls NY USA: ACM, 2019. p. 429–436. ISBN 978-1-4503-6666-3. Available from Internet: <<https://dl.acm.org/doi/10.1145/3307339.3342186>>.

WEI, X. et al. Superpixel Hierarchy. **IEEE Transactions on Image Processing**, v. 27, n. 10, p. 4838–4849, oct. 2018. ISSN 1057-7149, 1941-0042. Available from Internet: <<https://ieeexplore.ieee.org/document/8360136/>>.

WERBOS, P. **Beyond regression: New tools for prediction and analysis in the behavioral sciences**. Thesis (PhD thesis) — Harvard University, Cambridge, MA, 1974.

WHITE, A. D. Deep learning for molecules and materials. **Living Journal of Computational Molecular Science**, v. 3, n. 1, p. 1499, 2022. ISSN 25756524. Available from Internet: <<https://dmol.pub>>.

WOO, S. et al. ConvNeXt V2: Co-designing and Scaling ConvNets with Masked Autoencoders. In: **2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)**. Vancouver, BC, Canada: IEEE, 2023. p. 16133–16142. ISBN 979-8-3503-0129-8. Available from Internet: <<https://ieeexplore.ieee.org/document/10205236/>>.

WU, F.; RADEV, D.; LI, S. Z. Molformer: Motif-Based Transformer on 3D Heterogeneous Molecular Graphs. **Proceedings of the AAAI Conference on Artificial Intelligence**, v. 37, n. 4, p. 5312–5320, jun. 2023. ISSN 2374-3468, 2159-5399. Available from Internet: <<https://ojs.aaai.org/index.php/AAAI/article/view/25662>>.

WU, J. et al. Performance Analysis of Graph Neural Network Frameworks. In: **2021 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)**. Stony Brook, NY, USA: IEEE, 2021. p. 118–127. ISBN 978-1-7281-8643-6. Available from Internet: <<https://ieeexplore.ieee.org/document/9408211/>>.

XIAO, H.; RASUL, K.; VOLLGRAF, R. **Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms**. arXiv, 2017. ArXiv:1708.07747 [cs, stat]. Available from Internet: <<http://arxiv.org/abs/1708.07747>>.

XIAO, T. et al. Early Convolutions Help Transformers See Better. In: RANZATO, M. et al. (Ed.). **Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual**. [s.n.], 2021. p. 30392–30400. Available from Internet: <<https://proceedings.neurips.cc/paper/2021/hash/ff1418e8cc993fe8abcfe3ce2003e5c5-Abstract.html>>.

XU, K. et al. How Powerful are Graph Neural Networks? In: **7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019**. OpenReview.net, 2019. Available from Internet: <<https://openreview.net/forum?id=ryGs6iA5Km>>.

YAO, J. et al. Real-time coarse-to-fine topologically preserving segmentation. In: **2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. Boston, MA, USA: IEEE, 2015. p. 2947–2955. ISBN 978-1-4673-6964-0. Available from Internet: <<http://ieeexplore.ieee.org/document/7298913/>>.

YING, Z. et al. Hierarchical Graph Representation Learning with Differentiable Pooling. In: BENGIO, S. et al. (Ed.). **Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada**. [s.n.], 2018. p. 4805–4815. Available from Internet: <<https://proceedings.neurips.cc/paper/2018/hash/e77dbaf6759253c7c6d0efc5690369c7-Abstract.html>>.

Yuval Netzer et al. Reading digits in natural images with unsupervised feature learning. In: **NIPS workshop on Deep Learning and Unsupervised Feature Learning**. Sierra Nevada, Spain: [s.n.], 2011. Available from Internet: <<https://deeplearningworkshopnips2011.wordpress.com/wp-content/uploads/2011/12/12.pdf>>.

ZHANG, M. et al. An End-to-End Deep Learning Architecture for Graph Classification. **Proceedings of the AAAI Conference on Artificial Intelligence**, v. 32, n. 1, abr. 2018. ISSN 2374-3468, 2159-5399. Available from Internet: <<https://ojs.aaai.org/index.php/AAAI/article/view/11782>>.

ZHAO, W. et al. Exploring Over-smoothing in Graph Attention Networks from the Markov Chain Perspective. In: **Proceedings of the 2023 International Conference on Frontiers of Artificial Intelligence and Machine Learning**. Beijing China: ACM, 2023. p. 282–288. ISBN 979-8-4007-0754-4. Available from Internet: <<https://dl.acm.org/doi/10.1145/3616901.3617022>>.

ZHU, A. Z. et al. Superpixel Transformers for Efficient Semantic Segmentation. In: **2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. Detroit, MI, USA: IEEE, 2023. p. 7651–7658. ISBN 978-1-6654-9190-7. Available from Internet: <<https://ieeexplore.ieee.org/document/10341519/>>.

APPENDIX A — THE SLICO ALGORITHM

One can see the full SLICO algorithm described in the Chapter 2 and used in our models in Algorithm 5 as well as an algorithm describing the original implementation in Algorithm 6, which contained a small bug causing the distance values to be considered as the last cluster’s distance values instead of the member cluster’s distance value, which we believed to be unfaithful to the algorithm’s original description. Furthermore, our choice of implementation for SLICO, `scikit-image` (Walt et al., 2014), follows Algorithm 5 more closely, but instead recalculates the distances for each point, which is equivalent to the description here. Both implementations seem to be unfaithful to the original description of using the “last cluster center’s maximum distance” (Achanta et al., 2010; Achanta et al., 2012) and instead use the maximum distance ever encountered in a cluster center, which we keep in our Algorithms even though it is not as faithful to the description given in Achanta et al. (2012). `scikit-image` also allows the user to set the initial compactness parameter, but we omit that here since we use the default value of 10 (Walt et al., 2014), which is the one used in the original SLIC/SLICO implementation.

Algorithm 5 SLICO superpixel segmentation (Achanta et al., 2010; Achanta et al., 2012)

```

1: procedure SLICO(Image  $I$  of width  $w$ , height  $h$ ,  $c$  channels, desired number of
   superpixels  $k$ )
2:   Init.  $k$  cluster centers  $C_j = [c_{C_j}, s_{C_j}]^\top$  by sampling pixels at regular grid steps  $S$ .
3:   Init.  $k$  maximum color distances  $m_c(j) = 10$ .
4:   Move  $C_k$  to the lowest gradient position in a  $3 \times 3$  neighbourhood
5:   Set label  $l(i) = -1$  for each pixel  $i$ 
6:   Set distance  $d(i) = \inf$  for each pixel  $i$ 
7:   Set color distance  $d_c(i) = \inf$  for each pixel  $i$ 
8:   repeat
9:     for each cluster center  $C_j$  do ▷ Assign pixels to Clusters
10:      for each pixel  $i$  in a  $2S \times 2S$  region around  $C_j$  do
11:         $d_c \leftarrow \|c_i - c_{C_j}\|_2$ 
12:         $d_p \leftarrow \|s_i - s_{C_j}\|_2$ 
13:         $D \leftarrow \sqrt{\left(\frac{d_c}{m_c(j)}\right)^2 + \left(\frac{d_p}{S}\right)^2}$ 
14:        if  $D < d(i)$  then
15:           $d(i) \leftarrow D$ 
16:           $d_c(i) \leftarrow d_c$ 
17:           $l(i) \leftarrow j$ 
18:        end if
19:      end for
20:      if first iteration then ▷ Set all maximum cluster distances to a low value
21:         $m_c(j) = 1$ 
22:      end if
23:    end for
24:    for each pixel  $i$  do ▷ Update the maximum color distance of its cluster center
25:       $m_c(l(i)) = \max(m_c(l(i)), d_c(i))$ 
26:    end for
27:    for each new cluster center  $C'_j$  do ▷ Compute new cluster centers
28:       $c_{C'_j} = \frac{1}{\sum_{\forall i: l(i)=j} 1} \sum_{\forall i: l(i)=j} c_i$ 
29:       $s_{C'_j} = \frac{1}{\sum_{\forall i: l(i)=j} 1} \sum_{\forall i: l(i)=j} s_i$ 
30:    end for
31:    Compute residual error  $E$  with the l2 norm between the new  $C'_j$  and old  $C_j$ 
32:     $C_j \leftarrow C'_j$ 
33:  until  $E \leq \text{threshold}$ 
34:  return pixel labels  $l(i)$  and cluster centers  $C_k$ 
35: end procedure

```

Algorithm 6 SLICO superpixel segmentation as per the source code of (Achanta et al., 2010)

```

1: procedure SLICO(Image  $I$  of width  $w$ , height  $h$ ,  $c$  channels, desired number of
   superpixels  $k$ )
2:   Init.  $k$  cluster centers  $C_j = [c_{C_j}, s_{C_j}]^\top$  by sampling pixels at regular grid steps  $S$ .
3:   Init.  $k$  maximum color distances  $m_c(j) = 10$ .
4:   Move  $C_k$  to the lowest gradient position in a  $3 \times 3$  neighbourhood
5:   Set label  $l(i) = -1$  for each pixel  $i$ 
6:   Set distance  $d(i) = \inf$  for each pixel  $i$ 
7:   Set color distance  $d_c(i) = \inf$  for each pixel  $i$ 
8:   repeat
9:     for each cluster center  $C_j$  do ▷ Assign pixels to Clusters
10:      for each pixel  $i$  in a  $2S \times 2S$  region around  $C_j$  do
11:         $d_c(i) \leftarrow \|c_i - c_{C_j}\|_2$ 
12:         $d_p \leftarrow \|s_i - s_{C_j}\|_2$ 
13:         $D \leftarrow \sqrt{\left(\frac{d_c(i)}{m_c(j)}\right)^2 + \left(\frac{d_p}{S}\right)^2}$ 
14:        if  $D < d(i)$  then
15:           $d(i) \leftarrow D$ 
16:           $l(i) \leftarrow j$ 
17:        end if
18:      end for
19:      if first iteration then ▷ Set all maximum cluster distances to a low value
20:         $m_c(j) = 1$ 
21:      end if
22:    end for
23:    for each pixel  $i$  do ▷ Update the maximum color distance of its cluster center
24:       $m_c(l(i)) = \max(m_c(l(i)), d_c(i))$ 
25:    end for
26:    for each new cluster center  $C'_j$  do ▷ Compute new cluster centers
27:       $c_{C'_j} = \frac{1}{\sum_{i:l(i)=j} 1} \sum_{i:l(i)=j} c_i$ 
28:       $s_{C'_j} = \frac{1}{\sum_{i:l(i)=j} 1} \sum_{i:l(i)=j} s_i$ 
29:    end for
30:    Compute residual error  $E$  with the l2 norm between the new  $C'_j$  and old  $C_j$ 
31:     $C_j \leftarrow C'_j$ 
32:  until  $E \leq \text{threshold}$ 
33:  return pixel labels  $l(i)$  and cluster centers  $C_k$ 
34: end procedure

```

APPENDIX B — EXTRA MATERIAL FROM THE SUPERPIXEL TRANSFORMER MODEL EXPERIMENTS

Here we provide extra algorithms that are not entirely novel, such as the square grid superpixel/chunking strategy in Algorithm 7, or that are, to the best of our knowledge, novel but somehow still might be considered as incremental/derivative, such as the multidimensional sine-cosine embedding in Algorithm 8.

Algorithm 7 Square Grid Superpixel

Require: $\text{image} \in \mathbb{R}^{C \times h \times w}$, $\text{patch_size} \in \mathbb{Z}$
Ensure: $\text{segments} \in \mathbb{Z}^{h \times w}$
Ensure: $\sqrt{n_segments} \in \mathbb{Z}$
Ensure: $H \bmod \text{patch_size} = 0$
Ensure: $H = W$

- 1: $\text{num_patches_per_line} \leftarrow \sqrt{n_segments}$
- 2: Initialize $\text{segments} \leftarrow \text{zeros}(H, W)$
- 3: **for** $i \leftarrow 0$ to $n_segments - 1$ **do**
- 4: $i_y \leftarrow \lfloor i / \text{num_patches_per_line} \rfloor$
- 5: $i_x \leftarrow i \bmod \text{num_patches_per_line}$
- 6: Assign segment index i to region in segments
- 7: $\text{segments}[i_y \times \text{patch_size} \dots (i_y + 1) \times \text{patch_size}, i_x \times \text{patch_size} \dots (i_x + 1) \times \text{patch_size}] \leftarrow i$
- 8: **end for**
- 9: **return** segments

Algorithm 8 Sine-Cosine Coordinate Encoding

Require: $\text{hidden_dim} \in \mathbb{Z}$, $n_axes \in \mathbb{Z}$
Ensure: $\text{pos_emb} \in \mathbb{R}^{b \times s \times h}$, where b is batch size, s is sequence length, h is hidden dimension

- 1: Initialize $\text{div_term} = \exp([0 \dots h] \cdot (-\log(10000)/h))$
- 2: Initialize $\text{mixer} = W \in \mathbb{R}^{h \times h}$
- 3: **procedure** ENCODE(pos, x) $\triangleright \text{pos is } \mathbb{R}^{b \times s \times d}, x \text{ is } \mathbb{R}^{b \times s \times h}$
- 4: $d \leftarrow \min(n_axes, \text{pos.shape}[2])$
- 5: Initialize $\text{pos_emb} \leftarrow \text{zeros_like}(x)$
- 6: **for** $i \leftarrow 0$ to $d - 1$ **do** $\triangleright$ Build sin-cos for each dimension
- 7: Select all indices for cosine as $j_{\cos} = \{i, i + 2d, i + 4d, \dots\}$ up to h
- 8: Select all indices for sine as $j_{\sin} = \{i + 1, i + 1 + 2d, i + 1 + 4d, \dots\}$ up to h
- 9: $\text{pos_emb}[:, :, j_{\cos}] \leftarrow \cos(\text{pos}[:, :, i] \cdot \text{div_term}[j_{\cos}])$
- 10: $\text{pos_emb}[:, :, j_{\sin}] \leftarrow \sin(\text{pos}[:, :, i] \cdot \text{div_term}[j_{\sin}])$
- 11: **end for**
- 12: **return** $\text{mixer}(\text{pos_emb})$
- 13: **end procedure**

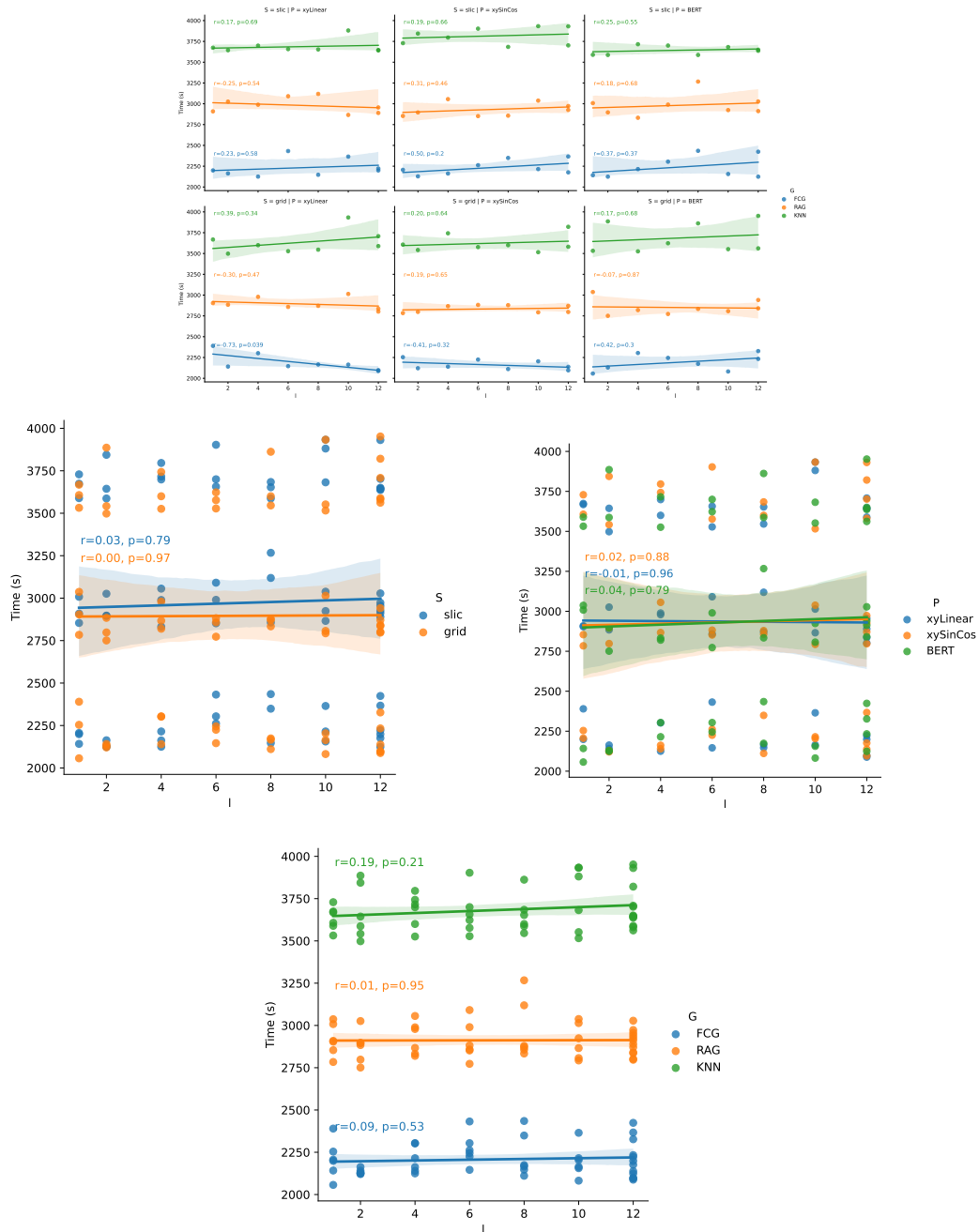
We also provide values about the time taken to complete training for models

trained on the FashionMNIST dataset with each variation of the superpixel strategy, positional embedding, and graph adjacency method in Table B.1 and Figure B.1, where it is possible to see that the biggest impact on performance was due to the graph building strategy.

Table B.1 – The average and standard deviation of the intercepts for linear regression models build on a certain combination of Superpixel, Positional Embedding and Graph Adjacency method. We can see that the Graph Adjacency method has the biggest impact on performance. Values for slopes are not included since most slopes had a non-significant p-value, and the comparison between intercepts yielded more significant p-values. Results are for the FashionMNIST dataset.

S	P	G	$\mu(b)$	$\sigma(b)$
slic	xyLinear	FCG	2191.445	79.985
	xySinCos	FCG	2163.883	56.462
	BERT	FCG	2163.673	91.721
	xyLinear	RAG	3018.270	67.750
	xySinCos	RAG	2890.867	59.136
	BERT	RAG	2945.360	97.905
	xyLinear	KNN	3665.725	59.916
	xySinCos	KNN	3784.874	75.899
	BERT	KNN	3622.777	38.381
grid	xyLinear	FCG	2308.859	53.799
	xySinCos	FCG	2198.941	40.184
	BERT	FCG	2127.548	68.252
	xyLinear	RAG	2927.251	51.175
	xySinCos	RAG	2820.486	32.625
	BERT	RAG	2860.717	70.748
	xyLinear	KNN	3547.717	97.029
	xySinCos	KNN	3590.628	77.070
	BERT	KNN	3636.872	133.942
slic	xyLinear	*	2958.480	252.643
	xySinCos	*	2946.541	274.293
	BERT	*	2910.603	244.672
grid	xyLinear	*	2927.942	251.979
	xySinCos	*	2870.018	252.923
	BERT	*	2875.046	261.837
slic	*	FCG	2173.000	40.673
		RAG	2951.499	42.239
		KNN	3691.125	43.873
grid	*	FCG	2211.783	35.102
		RAG	2869.485	30.399
		KNN	3591.739	56.513
*	xyLinear	FCG	2250.152	52.541
	xySinCos		2181.412	40.801
	BERT		2145.611	54.348
	xyLinear	RAG	2972.761	45.539
	xySinCos		2855.677	40.709
	BERT		2903.039	66.223
	xyLinear	KNN	3606.721	55.538
	xySinCos		3687.751	71.164
	BERT		3629.824	65.661
slic	*	*	2938.542	144.431
grid	*	*	2891.002	143.395
*	xyLinear	*	2943.211	174.740
	xySinCos	*	2908.280	183.329
	BERT	*	2892.825	175.366
	*	FCG	2192.391	28.569
		RAG	2910.492	29.785
		KNN	3641.432	36.430

Figure B.1 – Figures showing regression plots (with person r and p values) between the number of layers and the time taken for the process to finish training and testing. The top figure shows results separated by each of the variables, and most results show a positive r but a not-significant p , with the only significant p having a negative r , which we believe to be spurious. Only when we look at each variable individually on the three bottom figures that we are able to find significant differences between variables, and the graph building algorithm is has the biggest difference.



Aprendizado de Representações em Grafos aplicado a Imagens

APÊNDICE C — RESUMO EXPANDIDO

Este apêndice apresenta de forma resumida esta tese de doutorado.

C.1 Introdução

Com uma variedade de desenvolvimentos em IA ainda focando em Redes Neurais Convolucionais e *Transformers*, existem vozes persistentes na comunidade para que se construam modelos com capacidades relacionais e simbólicas, e as redes grafo-neurais – *Graph Neural Networks (GNNs)* – são frequentemente utilizadas como exemplo de tais modelos. Neste estudo primeiramente o problema de classificação de imagens é repensada como um problema classificação de grafos. Então a classificação de imagens baseadas em grafos é realinhada novamente com a literatura de classificação de imagens através do uso de modelos com atenção.

C.2 Motivação e Contribuições

Nesta tese é proposta a utilização de redes grafo-neurais para a classificação de imagens, a fim de explorar a capacidade de tais redes como método de aprendizado de representação para imagens. A fim de utilizar modelos com atenção baseados em grafos para classificação de imagens, o conceito de *superpixels* é utilizado como uma representação intermediária de uma imagem a partir da qual um grafo é construído: Primeiramente é utilizado um algoritmo de *superpixel* para segmentar a imagem, que é então transformada em um grafo onde a informação de cada nodo contém informação sobre a posição e o conteúdo de cada *superpixel*, e cujas arestas são baseadas na adjacência ou proximidade entre cada par de *superpixel*, formando uma instância de *Superpixel Image Classification (SIC)*, na qual então uma rede grafo-neural pode ser aplicada para realizar a classificação. Dentre as várias redes grafo-neurais, foi escolhida a rede grafo-atencional – *Graph Attention Network (GAT)* – com o qual propomos primeira aplicação de uma rede puramente atencional para classificação de imagens: nosso modelo intitulado *Superpixel Image Classification with Graph Attention Networks (SICGAT)*. Para este modelo, cada nodo contém a média das posições e cores de cada *superpixel*, e cujas arestas são baseadas na adjacência entre cada par de *superpixel*.

Posteriormente à proposta do nosso modelo, um novo modelo puramente atencional foi independentemente proposto para a classificação de imagens: os *Vision Transformers* (ViTs). Os ViTs foram altamente influentes na literatura, criando toda uma classe de modelos utilizadas frequentemente na literatura de visão computacional atual. Então, um *framework* metodológico é proposto nesta tese que generaliza tanto os ViTs quanto o nosso modelo SICGAT precedente, ao permitir várias combinações diferentes das partes do *pipeline* de SIC que, dependendo das escolhas, produz tanto o modelo SICGAT quanto o modelo ViT. Neste *framework* um estudo de ablação é realizado para mostrar quais componentes são importantes para a utilização correta de uma rede grafo-atencional para classificação de imagens.

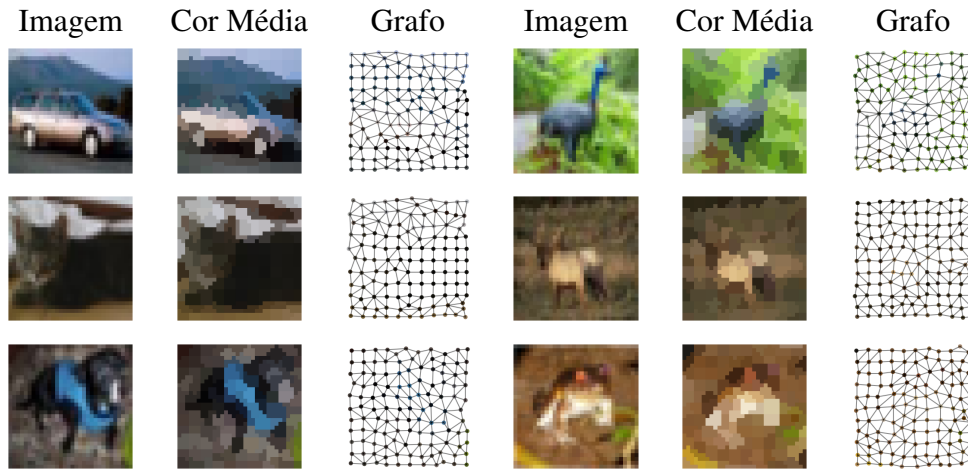
C.3 Resultados Principais

Nosso modelo *Supapixel Image Classification with Graph Attention Networks* (SICGAT) foi testado em vários *benchmarks* de imagens comuns, onde conseguiu uma acurácia de classificação maior que *baselines* disponíveis na época, a despeito do fato de que restringimos a informação disponível para o nosso modelo e, portanto, resolvemos um problema mais difícil. Isto foi feito tanto reduzindo o número de *superpixels* disponíveis para o modelo quanto usando as adjacências entre os *superpixels* para construir o grafo, o que produz menos arestas por nodos que *benchmarks* anteriores, assim não só reduzindo a informação disponível para o modelo quanto restringindo o fluxo de informação dentro do modelo. Foi discutido, entretanto, que sumarização das informações dos pixels que compunham o superpixel possivelmente causa demasiada perda de informação sobre a imagem, como pode ser visto na Figura C.1

Em vista da publicação do modelo *Vision Transformer* e dos problemas identificados no nosso modelo SICGAT, foram identificadas 6 diferenças fundamentais entre os modelos tradicionais de SIC, o modelo SICGAT e os ViTs, das quais um resumo é dado a seguir:

1. **(Estratégia de partição)** Embora um esquema de partição em retângulos – utilizado pelo ViT – já houvesse sido aplicado anteriormente na literatura, o modelo SICGAT (e a maioria dos modelos de SIC) utilizam um algoritmo de segmentação em *superpixels* para particionar a imagem em *chunks/superpixels*;
2. **(Agregação de informações dos pixels)** Enquanto no SICGAT e em outras redes de SIC da literatura as informações dos pixels que compõem o superpixel são nor-

Figura C.1 – Exemplos mostrando a perda de informação ao utilizar a a cor média dos *pixels* de um *superpixel* para se transformar uma imagem no grafo utilizado do modelo SICGAT. Isto afeta negativamente a performance possível de se atingir com esta entrada. Fonte: Autor (Avelar et al., 2020)

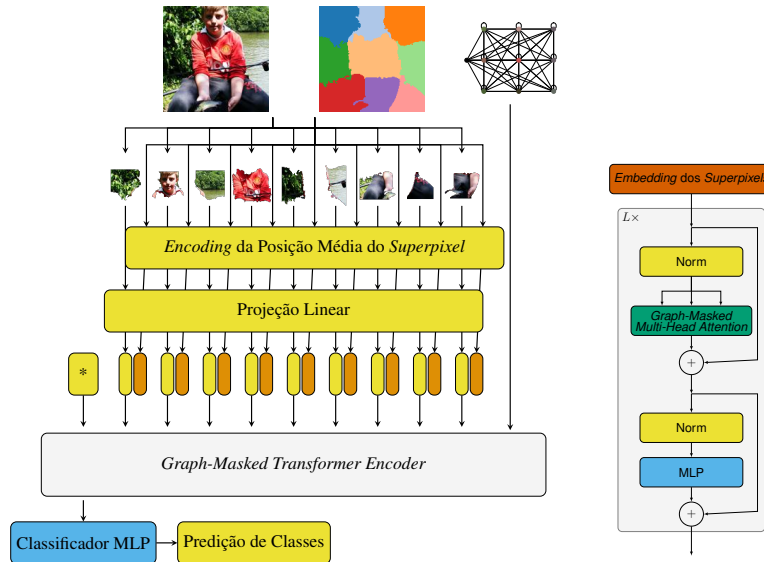


malmente resumidas em estatísticas, os ViTs utilizam uma projeção linear de todos os pixels num *chunk* antes de aplicar a parte principal da sua rede;

3. (**Embedding posicional**) Enquanto os modelos de SIC normalmente utilizam a posição média dos superpixels como informação posicional, os ViTs utilizam um *embedding* fixo, visto que seus *chunks* estão sempre localizados nas mesmas posições independente da imagem;
4. (**Blocos neurais utilizados**) o bloco *transformer* tem dois componentes regularizadores adicionais ao bloco GAT, são normalmente aplicados com conexões residuais, e têm uma função de atenção mais poderosa;
5. (**Conectividade do grafo**) Os ViTs tratam a imagem como um grafo completamente conectado, enquanto modelos de SIC normalmente utilizam um grafo onde cada nodo é conectado com seus k vizinhos mais próximos (*k-nearest neighbours*) ou, no caso específico do SICGAT, somente os vizinhos adjacentes são conectados;
6. (**Graph-pooling vs nodo virtual de classificação**) O ViT original utiliza um nodo virtual de classificação, enquanto no SICGAT a informação de todos os nodos é agregada no final a fim de se realizar a classificação, algo que também já foi explorado na literatura ViT e que demonstrou resultados promissores.

Com estas diferenças em mente, propomos uma nova *framework* – *SuperPixel Transformer (SPT)* – que generalisa tanto o nosso modelo SICGAT quanto *Vision Transformers*

Figura C.2 – A arquitetura proposta para o nosso framework (na esquerda) e o bloco neural *Graph-Masked Transformer Encoder* que utiliza *Graph-Masked Multi-head Attention* (na direita). Diferente do modelo SICGAT, toda a informação dos *superpixels* é utilizada através de uma estrutura de dados, definida no Algoritmo 4, e se permite várias maneiras de se representar a posição de um *superpixel* para uso na nossa rede. Diferente do ViT, permitimos algoritmos de *superpixel* arbitrários e usamos vários modelos de conectividade para o grafo, que ajuda a restringir o fluxo de informação no modelo. Visto que os *superpixels* podem ter formatos e posições diferentes, extendemos também o encoding seno-coseno tradicional para permitir que as coordenadas x e y de cada *superpixel* sejam utilizadas. Fonte: Autor



(ViT), e permite que mantenhamos toda a informação disponível em um *superpixel* enquanto o modelo ainda usa grafos de *superpixels*. Nosso framework SPT pode ser visualizado na Figura C.2 e, simplesmente alterando-se alguns componentes do nosso modelo, é possível recuperar tanto o modelo SICGAT quanto o modelo ViT.

Nossos resultados mostram que usar fragmentos baseados em *superpixel* tem performance comparável com quando se utiliza fragmentos quadrados regulares, comumente utilizados nos ViTs. Também foi demonstrado que padrões de conectividade além de fragmentos completamente conectados, comumente utilizados nos ViTs, podem melhorar os resultados quando comparados quando utilizados até com *superpixels* de fragmentos quadrados regulares. Por fim, foi demonstrado que um modelo que utiliza toda a informação sobre os *pixels* dentro de um *superpixel* pode superar a perda de informação demonstrada na Figura C.1.