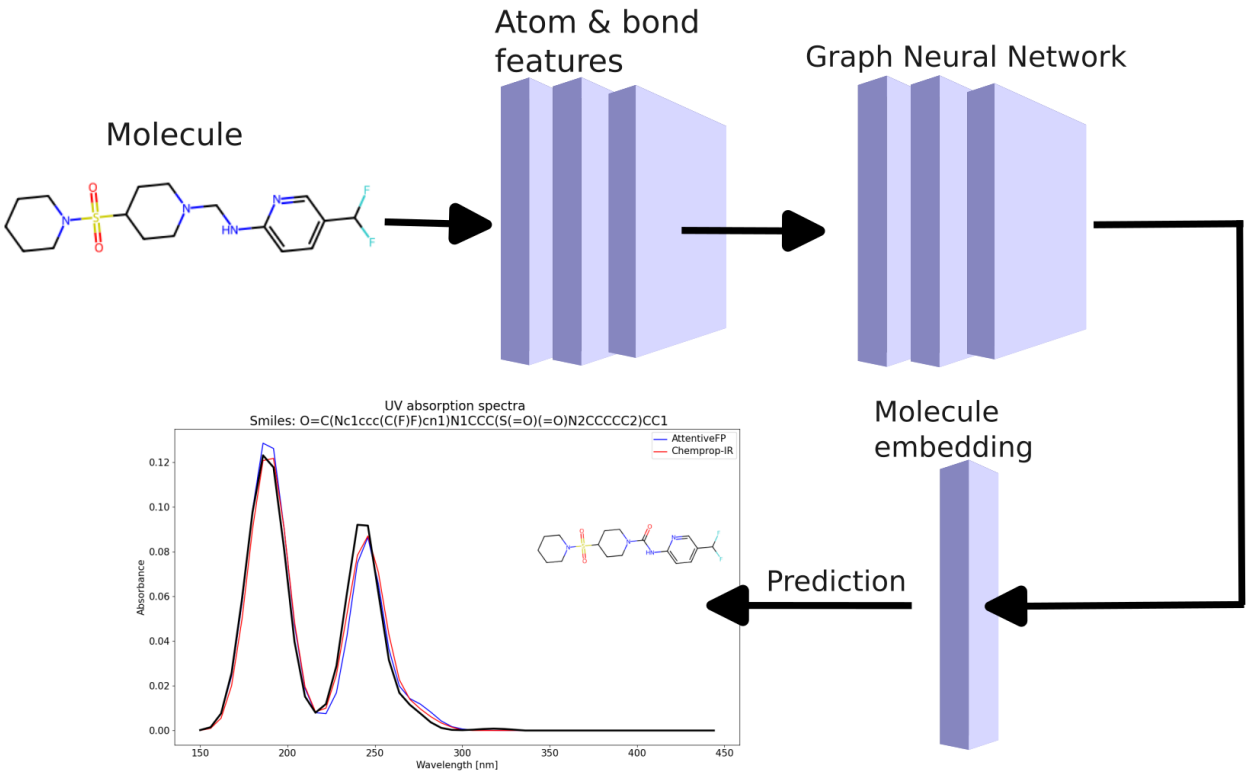
CHALMERS
UNIVERSITY OF TECHNOLOGY



Predicting UV-Vis absorption spectra by using graph neural network models

Implementing graph neural networks to predict UV-Vis absorption spectra of molecules

Master's thesis in Engineering Mathematics and Computational Science

William Nyrén & Ibrahim Taha

DEPARTMENT OF MATHEMATICAL SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2024
www.chalmers.se

Master's thesis 2024

Predicting UV-Vis absorption spectra by using graph neural network models

Implementing graph neural networks to predict UV-Vis absorption spectra of molecules

William Nyrén & Ibrahim Taha

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mathematical Sciences
Chalmers University of Technology
Gothenburg, Sweden 2024

Predicting UV-Vis absorption spectra by using graph neural network models
Implementing graph neural networks to predict UV-Vis absorption spectra of molecules
William Nyrén & Ibrahim Taha

Supervisor: Mats Josefson & Gustaf Hulthe, AstraZeneca
Examiner: Jan Gerken, Department of Mathematical Sciences

Master's Thesis 2024
Department of Mathematical Sciences
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: The general process of a molecule's spectra being predicted by a graph neural network.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2024

Predicting UV-Vis absorption spectra by using graph neural network models
Implementing graph neural networks to predict UV-Vis absorption spectra of molecules
William Nyrén & Ibrahim Taha
Department of Mathematical Sciences
Chalmers University of Technology

Abstract

In recent years, predicting absorption spectra by using different types of models utilizing deep neural networks has become an increasingly popular topic within spectroscopy. These models can be trained on datasets consisting of molecules represented as SMILES (Simplified Molecular Input Line Entry System), as well as intensities for a range of wavelengths. The resulting models can accurately predict excitation spectra for molecules. The capabilities of these models have proven useful in the drug industry, such as identifying harmful molecules or detecting substances.

Two popular models implementing graph neural networks (GNNs) and predict absorption spectra are AttentiveFP and Chemprop-IR. AttentiveFP uses a graph attention mechanism and message passing neural network to form a graph convolutional network. Chemprop-IR uses a directed message passing neural network, originally designed to predict IR spectra. These models were chosen due to both using the same type of data and molecular representation, outperforming regular regression models, and the ability of capturing complex patterns of diverse spectra as well as predicting these. The goal of this project was to set up, modify, and train the GNNs AttentiveFP and Chemprop-IR, which were constructed for other spectra prediction purposes, to predict UV-Vis absorption spectra within the range of 150 nm to 450 nm, with a 6 nm discretization, from the chemical structures of molecules.

Both models were trained on an identical dataset consisting of 10,502,904 molecules obtained from Oak Ridge National Laboratory with the same split for training, testing, and validation. The models used SMILES as the input of the molecules, as well as the intensities at each corresponding wavelength. AttentiveFP was trained with three different different implementations of the attention mechanism, GAT, GATv2, and DenseGAT. GAT is proven to compute static attention while GATv2 computes dynamic attention. DenseGAT is based on GAT but instead considers the molecule as a fully connected graph. Chemprop-IR was trained for three different choices of FFN-hidden size (feedforward neural network) and hidden size of 2200, 2800, and 3400.

The best-performing model of AttentiveFP used the GATv2 attention mechanism and 8 attentive layers. For Chemprop-IR, the best-performing model used an FFN hidden size of 2800. Both models show promise predicting UV-Vis absorption spectra. AttentiveFP proved to be the better-performing model of the two based on predictions and validation loss. The model was able to make accurate predictions on a large set of molecules, correctly identifying the number of peaks and their

positions. However, it failed for some molecules, where the predicted spectra were completely different from the true spectra. The exact reason is hard to determine. However, the coexistence of larger molecules with high-frequency components and low prediction error, along with small molecules with low-frequency components and high prediction error, suggests that the attention mechanism is working. Chemprop-IR resulted in an increased accuracy of predicting multiple peaks for larger sizes of FFN-hidden size and hidden size. Predicted spectra of lowest accuracies were mostly ones with multiple peaks. One theory is that predicting IR spectra differs greatly from UV.

Keywords: Attention-based mechanism, AttentiveFP, Backpropagation, Chemprop-IR, D-MPNN, GNN, MPNN, SMILES, UV-Vis absorption spectra

Acknowledgements

First and foremost, we would like to thank our supervisors at AstraZeneca Mats Josefson and Gustaf Hulthe for allowing us to extend this project into a master's thesis and being there for every step of the way, guiding us, and providing us with tons of resources and inspiration. To have had the chance of working with people of such high intelligence has been inspiring for the both of us. We would also like to thank our examiner Jan Gerken for his expert insights and knowledge of this project, as well as taking us during such a late stage compared to others. Finally, our gratitude lies among our friends and family, who showed love and support that helped us proceed with the master's thesis.

A big thanks to everyone mentioned, this would have not been possible without your support.

William Nyrén & Ibrahim Taha, Gothenburg, June 2024

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

CSV	Comma separated values
D-MPNN	Directed message passing neural network
DNN	Deep neural network
FFN	Feed forward neural network
GNN	Graph neural network
HDF5	Hierarchical Data Format Version 5
IR	Infrared
MPNN	Message passing neural network
RMSE	Root mean square error
ReLU	Rectified Linear Unit
SELFIES	Self-referencing embedded strings
SID	Spectral information divergence
SMILES	Simplified molecular input line entry system
SRMSE	Scaled root mean square error
UV-Vis	Ultraviolet-visible

Nomenclature

Below is the nomenclature of indices, sets, parameters, and variables that have been used throughout this thesis.

Parameters and variables

$\mathcal{G}$	An undirected graph
$\mathcal{V}$	Set of vertices
$\mathcal{E}$	Set of edges
$\mathcal{G}_{dir}$	A directed graph
N	Number of nodes
T	Number of iterations for GNNs
i	Vertex with index i
l_{ij}	Edge connecting vertex i to j
L	Loss function
$\mathbf{W}$	Weight matrix
$co[i]$	Set of edges connected to node i
$ne[i]$	Set of neighbors of node i
$\boldsymbol{A}$	Adjacency matrix
$\boldsymbol{D}$	Diagonal degree matrix
$d(v_i)$	Vertex with index i
m	Number of inputs that a neuron receives
x_k	Input k for a neuron
w_k	Weight for input x_k for a neuron
y	Weighted summation of inputs of a neuron
f	Activation function
z	Activation function
f_t	Local transition function

g	Local output activation function
$\mathbf{h}$	Hidden state
$\mathbf{o}$	Output state
x_v	Input feature for node v
$\mathbf{H}$	States matrix
$\mathbf{O}$	Outputs matrix
$\mathbf{X}$	Features matrix
$\mathbf{X}_N$	Node features matrix
F_t	Global transition function
G	Global output function
$\mathbf{H}^t$	t^{th} iteration of $\mathbf{H}$
$\mathbf{t}_v$	Output target for node v
$\mathcal{N}_i$	Neighboring set of indices
α_{ij}	Attention weight
e_{ij}	Attention score
σ	Non-linear activation function

Contents

List of Figures

List of Tables

1

Introduction

1.1 Background

In the early stages of pharmaceutical development, chemists and researchers gather and interpret information about the chemical structures of molecules of interest. This is typically done using spectroscopy, which offers various methods to study molecular structures and their properties. The four main spectroscopic methods are infrared (IR), ultraviolet-visible (UV-Vis), nuclear magnetic resonance (NMR), and mass spectrometry (MS). These methods are often combined to provide a more comprehensive understanding of a compound. IR spectroscopy examines vibrational modes and molecular structure, UV-Vis spectroscopy studies electronic transitions and identifies chromophores, NMR spectroscopy analyzes the magnetic properties of atoms to provide insight into their environment and connectivity, and MS characterizes mass and fragmentation patterns. Together, these methods are powerful tools for building a detailed picture of complex compounds [1].

Despite these powerful tools, researchers still face significant challenges in navigating the vast universe of molecules to find lead compounds and optimize them by making small structural changes to achieve desired properties. For instance, a small subset of organic molecules containing up to 30 carbon (C), nitrogen (N), oxygen (O), and sulfur (S) atoms may have more than 10^{60} possible members [2]. Furthermore, small structural changes in molecules can lead to significant changes in their spectral fingerprints and properties. A common example is Bromothymol Blue (BTB), a pH indicator that changes color based on the acidity of the solution. Figure 1.1 demonstrates that the BTB molecule is mostly similar whether it absorbs blue light, making the solution appear yellow, or absorbs yellow, green, and red light, making the solution appear blue. This example illustrates how small changes can have significant impacts [3].

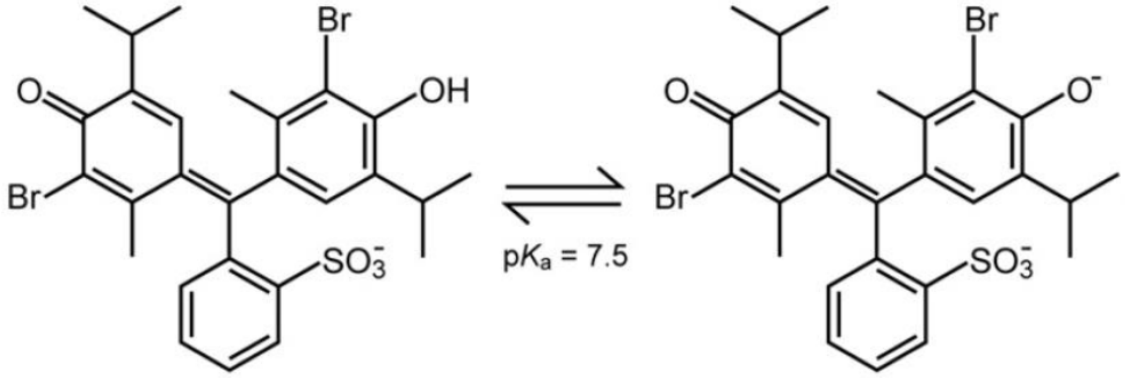


Figure 1.1: Color change of Bromothymol Blue based on pH: Absorbing blue light (left), making the solution appear yellow, and absorbing yellow, green, and red light (right), making the solution appear blue.

To aid the chemists in the exploration of the molecular universe, they have molecular spectroscopic databases at their disposal. These databases consist of either collected data from within the organization or bought from other suppliers. With a rich molecular spectroscopic database, a molecule and its properties of interest could be found within a database, which effectively could eliminate the steps of producing a compound, making sure that the sample is clean, making new measurements, and analyzing the results. If molecular spectroscopic databases do not provide the necessary information for the chemist, the laborious process of producing compounds and sampling new measurements could go through a slow, iterative process which ideally is avoided to speed up the process of drug discovery and thus making it more cost-effective.

In tandem with the use of molecular spectroscopic databases, other useful tools that could be of use are computational methods based on theory [4, 5]. Although these methods have a major role in drug discovery, they typically come with limitations such as the large amount of hardware resources needed and the fact that they rely on approximations [6]. Increasing the complexity is the fact that when real-world measurements are taken, it is often the case that the compound of interest is dissolved in a solution. This adds to the difficulty of using computational methods and potentially causes a disconnect between observed real-world data and the capabilities of computational tools.

With the limiting factors of using databases and computational tools, there should be no surprise that the increase of deep learning in recent years has caught the attention of the pharmaceutical community with its potential to add to the discovery of new pharmaceuticals [7, 8, 9, 10, 11]. Due to the graph-like structure of molecules, a sub-field of the deep learning paradigm called graph neural networks (GNNs) is of particular interest, which is also the case in the paper. New models are constantly created, each offering unique abilities, which specialize in certain fields, such as predicting spectra within certain wavelengths or for different categories of molecules. These new models are developed with different types of algorithms and can be used

in many real-world applications.

1.2 Goals and objectives

Despite there being an increased activity in designing state-of-the-art deep learning models to predict spectra, such as MS and IR in recent years, the same amount of resources and attention has not gone into the realm of UV-Vis spectroscopy. This could be due to several factors, such as a lack of necessary data for training models, the type of spectra that provide the most potential benefits of utilizing deep learning to researchers, and simply which problem is the most attractive to solve[9, 10, 11, 8]. Whatever the reason, there has recently been a major contribution to the former, with over 10.5 million simulated UV-Vis absorption spectra of organic molecules being released to the public. [5, 12]. With access to this dataset, there is an opportunity to aid in the research of the capabilities of deep learning models to predict UV-Vis absorption spectra based on chemical structures.

Therefore, the goal of this project is to implement and modify two GNNs, namely AttentiveFP and Chemprop-IR, in such a way that they are able to train on the entire dataset and produce predictions at the end of the stage. Firstly, the data is prepared in such that they contain information which is identical and acceptable for both of the models. Secondly, the models are to be trained on this dataset to thereafter predict the UV-Vis absorption spectra for unseen molecules within the range of 150 nm to 450 nm. Finally, the predictions produced by the models are evaluated and compared to each other in order to find where each model excels and where it is limited. Many evaluations are considered to determine which model generally offers the most optimal results.

Chemprop-IR is selected since it has demonstrated promising results in predicting IR spectra [10]. Since the IR-spectra shares some similarities to the UV-Vis absorption spectra, it makes the model mostly set up for our task. Our task is to predict UV-Vis absorption spectra, where the input and output structure of the underlying code is similar in both scenarios of using different types of spectra. Due to the flexibility of the code, it does not require too many modifications to work for UV spectra. Another reason is the benefit of the MPNN (message passing neural network) structure that offers the ability of capturing complex relationships between atoms, bonds, and their features, which appears for the UV-Vis range [10]. It is therefore out of interest to see how well the model adapts to this range. AttentiveFP is also a GNN model, but in contrast to Chemprop-IR, it incorporates an attention mechanism that enables the model to capture long-range interactions between atoms in a molecule [13]. This ability is a highly desirable property when dealing with UV-Vis spectra, since the spectra are influenced by long-range interactions in the molecules. Just like Chemprop-IR, AttentiveFP has also been used for IR prediction tasks with state-of-the-art performance, making the model interesting in this setting as well [9].

Since we are dealing with powerful GNNs, it is important to understand the building blocks of these models. One important topic that is further discussed in this thesis

is the utilization of the Weisfeiler-Lehman's (WL) graph isomorphism test, which analyzes whether if graphs are topologically identical or not. This usage of the WL test is talked more in the theorems 2.2.1 and 2.2.2. These theorems help us understand if there is loss of information when the models capture as much information as possible for the provided data [14].

1.3 Limitations and considerations

The limits set by us are that we will only work with Graph Neural Networks, in particular the two chosen models, AttentiveFP and Chemprop-IR. Secondly, the range of wavelengths of interest is the UV-Vis spectra, and thus we only consider intensities within the range of wavelengths that are 150 to 450 nm. The intensities will be discretized by 6 nm, rather than representing the data continuously, to comply with resource limitations and time restrictions. Thirdly, the dataset of molecules will be limited to 10,502,904 molecules and all come from Oak Ridge National Laboratory. This selection is made due to the dataset providing the excitation energies, which correspond to the absorption peak positions, relevant for the UV-Vis range. This dataset contains molecules of wide diversity, from the types of atoms to the lengths of molecules, and was therefore deemed as suitable for this project.

2

Theory

This section of the project focuses on presenting the necessary theoretical and mathematical background to proceed further with the project. The main topics discussed are graph theory, the concept of neural networks, graph neural networks, as well as the involved models which are based on graph neural network architectures. Finally, a method of representing molecules in a string, denoted by SMILES, is presented, which is helpful for graph neural networks.

2.1 Graph theory

The purpose of graph theory is to use a graph representation that illustrates a connection between different objects. The connections are the edges and the objects are the vertices.

2.1.1 A short introduction to graph theory

A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consists of a set of vertices $\mathcal{V}$ and a set of edges, $\mathcal{E}$. An edge $l_{ij} = i, j$ has the vertices i and j as endpoints, which are said to be joined by l_{ij}. Two vertices, i and j, are adjacent if they are joined by a common edge l_{ij}. This can be used to define the adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$, where N denotes the number of vertices, with the following elements

$$A_{ij} = \begin{cases} 1 & \text{if } l_{ij} \in \mathcal{E} \text{ and } i \neq j, \\ 0 & \text{else.} \end{cases}$$

The number of edges connected to vertex i is denoted by the degree of vertex i, which is $d(i)$. This can be used to create the diagonal degree matrix $\boldsymbol{D} \in \mathbb{R}^{N \times N}$ with elements $D_{ii} = d(i)$, where N denotes the number of vertices [15, 16].

There are two types of graphs assessed by directionality, directed and undirected. Figure 2.3 illustrates two basic examples of such graphs.

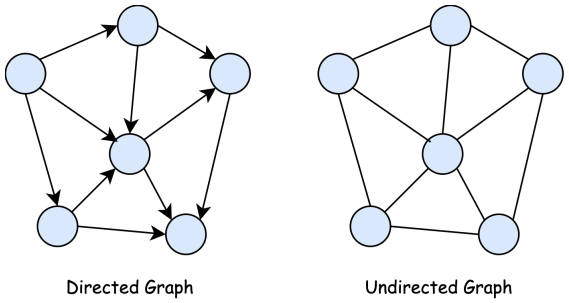


Figure 2.1: A basic example of a directed graph on the left and an undirected graph on the right.

The main differences between these are that information stored in the edge for an undirected graph can flow in either direction, while the information for a directed graph is only able to flow in the direction denoted by the arrow of the edge. Another difference is that for an undirected graph, the adjacency matrix $\boldsymbol{A}$ is a symmetric graph [15, 16].

2.1.2 Connection between molecule representation graph theory

Molecules, consisting of atoms and electrons in the 3D space, that are used in this project can be represented by graphs and strings [15]. This graph representation displays the vertices as the atoms, such as C, F, N, and O, and the edges as the different types of chemical bonds, i.e., single-bonds and double-bonds. Graph-structured data can therefore have edge-, vertex-, or graph-leveled tasks [15].

In an edge-level task, the goal is to predict the edges, which are relationships and connections, between the given vertices, which are objects, and their corresponding values. The case is opposite for a vertex-level task, where the goal is to predict and label the vertices given the edges. Lastly, we have graph-level tasks where the objective is to predict the property of the whole graph [15]. Graph-level tasks are the most relevant out of the three types for this project since the goal is to capture the behavior of the graph as an entirety. In other words, it means that we predict the UV-Vis absorption spectra given the molecules, where peaks in the spectra are achieved from the graph as a whole unit or subsets of graphs.

2.2 Theory behind neural networks

This project uses models that implement graph neural networks (GNNs) to initiate training procedures that ultimately lead to models being able to predict UV-spectra, given the molecules represented as graphs, which were constructed from a string representation of the molecules. It is therefore important to understand how GNNs operate in general and later on narrow it down to the specific models used. The theory enables a better understanding and visualization of how the models operate and take into considerations that we are dealing with molecules by using mathematical

principles.

The core of GNNs are neural networks, which are made of neurons that interconnect. Neurons are the units of the network, which receive inputs and return outputs. A basic example is that we consider that a neuron receives m inputs denoted by$\{x_k\}_{k=1}^m$ with the corresponding weights $\{w_k\}_{k=1}^m$. This results in the weighted summation $y = \sum_{k=1}^m w_k x_k$, which later on passes an activation function f and in turn returns the output $z = f(y)$. The purpose of the activation function is to map a real-valued number to a number between 0 and 1, where 0 denotes a deactivated neuron and 1 is a fully activated neuron. An example of such an activation function is

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \tag{2.1}$$

The choice of output activation function is dependent on the model chosen to know what types of values are handled [16].

The training procedure starts with randomized weights and values, which are as earlier denoted by $w_1, ..., w_m$ and result in an output z. The goal is to reach an optimized state where the performance is satisfied, which is done with the help of the backpropagation algorithm. The backpropagation algorithm is an important method for training neural networks. It is an iterative algorithm that minimizes the cost function, where the minimization is done by finding out which weights and parameters of the model are to be adjusted. Backpropagation runs for a set number of epochs, where for every epoch, the derivatives of the loss of a parameter set are computed. A gradient-descent optimization algorithm is used afterward for the derivatives, which moves down towards the gradient of the error. This in turn minimizes the loss and updates the weights and parameters. The computation of the gradient uses the chain rule, which navigates through a neural network consisting of many complex layers, as illustrated in the example figure below.

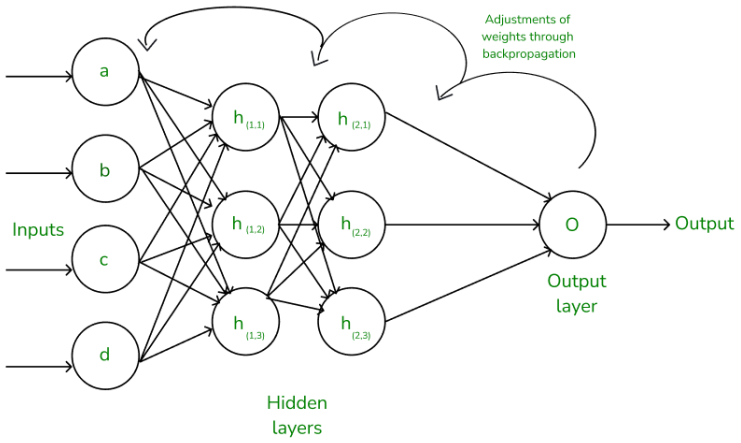


Figure 2.2: An example of how backpropagation operates and updates the weights.

There are two phases that make up the algorithm, which are forward pass and backward pass. For the forward pass, the input layer receives raw data as input, which

is used for training a neural network. This data, followed by their corresponding weights, are directed to the hidden layers, such as $h_{(i,j)}$ in figure 2.2, where an activation function is applied to the data. This activation data ensures that the data follows restrictions, such as returning positive data and zero if the data is negative. Signs of non-linearity start to appear as a result, which enables the model to learn complex relationships among the data. The weighted outputs from the last hidden layer are plugged into the output to produce a final prediction [16, 17].

The second pass is the backward pass, where the error is reused in the network. This causes the network to learn and update its weights, which in turn advances the performance of the network. This backward mannered propagation is done layerwise, where the gradient of each weight is computed by using the chain rule iteratively for efficient computations. The gradient reveals how the weights should be updated to minimize the error in the coming forward pass. These steps are repeated until the optimization target of performance has been satisfied [16, 17].

Backpropagation offers several benefits, such as being usable for a diverse selection of problems and networks. This is due to its flexibility and simplicity. Another benefit is the accelerated learning process from updating weights directly based on the derivatives of the errors from the parameters used. Finally, a benefit that is important for this project is the scalability of the algorithm, which allows large-scale machine learning tasks that appear for training datasets of large sizes [17].

There are several types of neural networks, each adapted to specialize in certain fields. The most important one that is to be focused on here is graph neural networks (GNNs) [16].

2.2.1 Graph neural networks

Graph neural networks (GNNs) have been constructed to deal with graph-structured data, such as molecules, which acts as an extension to already existing neural networks. The goal of GNNs is to learn a state embedding that encodes the information of the neighborhood for each node [16].

The model consists of the following additional functions and variables

- f_t which is the local transition function, which is mutual for all nodes
- g which is the local output function,
- $\mathbf{h}$ which is the hidden state,
- $\mathbf{o}$ which is the output state,
- $co[i]$ which is the set of edges connected to node i,
- $ne[i]$ which is the set of neighbors of node i,
- $\mathbf{x}_i$ which is the input feature for node v.

This setup returns the following

$$\mathbf{h}_i = f_t(\mathbf{x}_i, \mathbf{x}_{co[i]}, \mathbf{h}_{ne[i]}, \mathbf{x}_{ne[i]}) \tag{2.2}$$

$$\mathbf{o}_i = g(\mathbf{h}_i, \mathbf{x}_i), \tag{2.3}$$

where $\mathbf{x}_i$, $\mathbf{x}_{co[i]}$, $\mathbf{x}_{ne[i]}$, and $\mathbf{h}_{ne[i]}$ are the features of i, the features of the set of edges connected to i, the features of the set of neighbors connected to i, and the hidden states of i. The image below illustrates an example of a graph neural network.

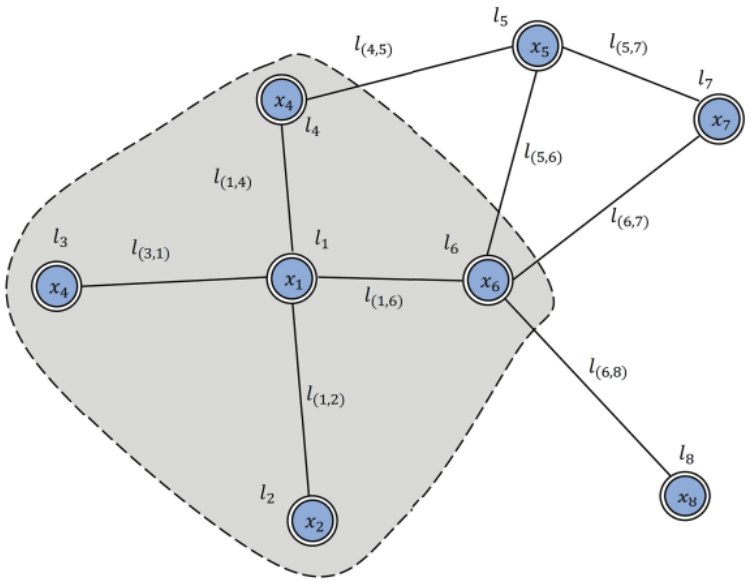


Figure 2.3: A basic example of a graph neural network.

The image contains the nodes l_i, edges $l_{i,j}$, and $\mathbf{x}_{l_i}$ which are the input features of node l_i.

Another way of representing the model is by using the compact form. Denote $\mathbf{H}$, $\mathbf{O}$, $\mathbf{X}$, and $\mathbf{X}_N$ as the matrices created from stacking all the states, outputs, features, and node features correspondingly. This yields the compact form below

$$\mathbf{H} = F_t(\mathbf{H}, \mathbf{X}), \tag{2.4}$$

$$\mathbf{O} = G(\mathbf{H}, \mathbf{X}_N), \tag{2.5}$$

with F_t being the global transition function and G the global output function. These functions are stacked variants of the previous local transition function f_t and local output function g. $\mathbf{H}$ has a fixed point value and is uniquely defined when assuming that F_t is a contraction map [16].

Another way of writing the compact form is by using Banach's fixed point theorem, which yields the following iterative alternative to compute the state

$$\mathbf{H} = F_t(\mathbf{H}, \mathbf{X}), \tag{2.6}$$

in which $\mathbf{H}^t$ is the t^{th} iteration of $\mathbf{H}$.

The final step of the model is to evaluate the accuracy of reaching the target $\mathbf{t_v}$ for node v. This is done with a loss function, which in general is described as

$$L = \sum_{i=1}^{N} (\mathbf{t}_i - \mathbf{o}_i), \tag{2.7}$$

where N denotes the number of nodes in the network.

The method used for the learning algorithm is gradient-descent, which consists of the steps below [16].

- $\mathbf{h}_i^t$ are iteratively updated until time step T has been reached, which results in the fixed point solution $\mathbf{H}(T) \approx \mathbf{H}$.
- The weights $\mathbf{W}$ have their gradients estimated from the loss. These weights are then changed corresponding to the gradient computed in the last step.

One relevant topic for GNNs is the building blocks of making more powerful models. These powerful GNNs have the ability to differentiate diverse graph structures by mapping different graphs to different embeddings. An implication of doing so is the graph isomorphism problem, which studies whether graphs are topologically identical. The goal is that isomorphic graphs are to be mapped to the same embedding and non-isomorphic ones to different embeddings. One way of exploring this representation capacity further is by using Weisfeiler-Lehman's (WL) graph isomorphism test [14].

The WL graph isomorphism test does the following iteratively. Firstly, the labels of nodes and their neighbors are aggregated. Secondly, the aggregated labels are hashed into new unique labels. Thereafter, two graphs are concluded to be non-isomorphic if for some iteration, these node labels between the two graphs in comparison differ. This injective aggregation update makes the test a powerful method to use. The WL test is deemed as a powerful tool and basis to be used in the following theorems, due to being proven to work for general cases of GNNs. The importance of the WL test is to prove that these types of GNNs are as powerful as the WL test in terms of expressiveness, flexibility, and classification as well as simply determining if two graphs are isomorphic [14]. We use the WL test to define such powerful GNNs in the theorems below, where the first one creates a connection between powerful GNNs and the WL test.

Theorem 2.2.1. *([14]) Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be two non-isomorphic graphs. If a GNN $\mathcal{A} : \mathcal{G} \to \mathbb{R}^d$ maps $\mathcal{G}_1$ and $\mathcal{G}_2$ to different embeddings, the Weisfeiler-Lehman graph isomorphism test also states that $\mathcal{G}_1$ and $\mathcal{G}_2$ are non-isomorphic.*

What this theorem states is that if a GNN can distinguish between two non-isomorphic graphs, then the WL test can also do that. This means that the ability the GNN has of finding differences between graphs is at most as powerful as the WL test. The idea of proving theorem 2.2.1 is proven by contradiction by using induction. The goal is to prove that $\mathcal{A}(\mathcal{G}_1) = \mathcal{A}(\mathcal{G}_2)$ by assuming from the start of induction that $\mathcal{A}(\mathcal{G}_1) \neq \mathcal{A}(\mathcal{G}_2)$. This brings up the question if we can define GNNs to be as powerful as the WL test, which takes us to the second theorem that talks about a case where they are equally powerful.

Theorem 2.2.2. *([14]) Let $\mathcal{A} : \mathcal{G} \to \mathbb{R}^d$ be a GNN. With a sufficient number of GNN layers, $\mathcal{A}$ maps to any graphs G_1 and G_2 that the Weisfeiler-Lehman test of isomorphism decides as non-isomorphic, to different embeddings if the following conditions hold*

1. *$\mathcal{A}$ aggregates and updates node features $h_v^{(k)}$ iteratively, where*

$$h_v^{(k)} = \phi\left(h_v^{(k-1)}, f\left(\left\{h_u^{(k)} : u \in \mathcal{N}(v)\right\}\right)\right), \tag{2.8}$$

   where $\mathcal{N}(v)$ is the set of nodes adjacent to node v, and functions f and ϕ, that operate on multisets, are injective.
2. *$\mathcal{A}$'s graph-level readout, which operates on the multiset of node features $\left\{h_v^{(k)}\right\}$, is injective.*

In essence, the theorem discusses that if a neighbor aggregation and graph-level readout functions are injective, then it implies that the GNN is as powerful as the WL test. The idea of proving theorem 2.2.2 is to use the property of injective functions and induction to prove that the multisets of the graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ must be different for each graph and thus the node embeddings for the GNN $\mathcal{A}$ must be different for both of the graphs. Combining theorems 2.2.1 and 2.2.2 ensures that if these are true, then it ensures the absence of loss of information when a GNN processes multiple graphs or subsets of graphs. It also implies that the GNN is able to capture as much information about graphs as the stable WL test.

In-depth proofs of the theorem and lemma are found in Appendix A.2.

2.2.1.1 Considerations and limitations of graph neural networks

Despite enabling models to efficiently handle and construct graph-represented data and including neural networks in graph theory, it has some limitations. Specifically, the topic has 4 main limitations, which are discussed further in this section.

The first limitation is the inefficiency of updating hidden states of nodes through iterations to achieve an approximation of a fixed point, which takes T time steps to reach. This fixed point is what is considered as a state where the GNN does not produce representations that differ for any additional updates. This can be partly resolved by relaxing the assumption of a fixed point and is done by using a multi-layer GNN, which in turn produces a stable representation of the node and the neighborhood. The second limitation is the constant usage of parameters for each iteration and all layers. Compared to other popular GNNs, such as graph convolutional networks, a different setup of parameters is used in different layers. An implication of doing so is the creation of a hierarchical feature extraction method. One solution is to incorporate the usage of different parameters in different layers. The third limitation is the lack of efficient modeling informative features on edges. Having messages propagate through edges can be beneficial to learn about hidden states of edges. The final problem is related to when a model needs a large number of steps, T, to approximate the fixed points. For such T, the model is deemed inappropriate to use for the fixed points if the representation of nodes is focused on rather than the representation of graphs since the distribution of representation in fixed points being smoother in value and therefore less informative to distinguish each node [15, 16].

The main problems regarding graph neural networks are solved with specialized

variants, each dealing with a specific problem [16]. Relevant examples of such variants are attention-based graph neural networks and directed message passing neural networks (D-MPNN).

2.2.2 Attention-based mechanism for graph neural networks

The idea of treating neighbors of a node differently in a GNN comes naturally from, e.g., neighboring nodes having different levels of influence on the node of interest. A network design called graph attention networks for this type of situation was proposed in recent studies [18]. Graph attention networks (GATs) treat neighboring nodes differently by comparing their importance using a scoring function and softmax, in contrast to graph convolutional networks, which interpret the importance of all neighboring nodes equally.

Before the attentive functionality is utilized in a GAT network, at least one learnable linear transformation is recommended to increase the expressive power and ability to transform input features into higher-level features [18]. In this project, the learnable linear transformation is performed by a shared weight matrix $\mathbf{W}$ to obtain an initial node embedding. The initial atom embeddings are then passed through a convolutional layer, GATEConv. GATEConv aggregates neighboring (initial) atom embeddings with the corresponding bond features to produce embeddings that also include bond information. This is done such that the final embeddings have a unified length. A detailed description of the procedure is provided by the original authors [13]. With the final embeddings in place, subsequent layers are used to propagate messages between nodes using the graph attention mechanism, which is described next.

We denote $\mathbf{W} \in \mathbb{R}^{F' \times F}$ as a learnable weights matrix together with a *self-attention* a. The dimension of $\mathbf{W}$ is determined by the size of the node embedding $\mathbf{h} \in \mathbb{F}$, and the transformed node embedding $\mathbf{h}' \in \mathbb{F}'$. A scoring function e computes a score for all edges $(i,j)_{i \neq j}$ in the graph where $i, j \in \{1, \ldots, N\}$ and N is the number of nodes. The scoring function e can be expressed as

$$e_{ij} = a(\mathbf{W}\mathbf{h}_i, \mathbf{W}\mathbf{h}_j), \quad \begin{cases} a : \mathbb{R}^{F' \times F} \to \mathbb{R} \\ i, j \in \{1, \ldots, N\} \end{cases} . \tag{2.9}$$

The scores computed using equation 2.9 are then normalized using softmax to compute the attention weights

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})} \tag{2.10}$$

where $\mathcal{N}_i$ is the set of neighboring nodes to node i. Only considering nodes in the set $\mathcal{N}_i$ is called *masked attention*. If all nodes in the graph are considered when computing the sum in equation 2.10, attention weights are computed as if the underlying graph were fully connected. This method is referred to as DenseGAT going forward.

The attention mechanism $a(\cdot)$ in this work consists of a single-layer feedforward neural network, parameterized by a weight vector $\mathbf{a} \in \mathbb{R}^{2F'}$ to which the LeakyReLU nonlinearity is applied. This means that the attention weights can be expressed as

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}^T\left[\mathbf{W}\mathbf{h}_i \parallel \mathbf{W}\mathbf{h}_j\right]\right)\right)}{\sum_{k\in N_i}\exp\left(\text{LeakyReLU}\left(\mathbf{a}^T\left[\mathbf{W}\mathbf{h}_i \parallel \mathbf{W}\mathbf{h}_k\right]\right)\right)} \tag{2.11}$$

where $\mathbf{a}^T$ is the transpose of $\mathbf{a}$ and $\parallel$ is the concatenation operation, hence the dimension $\mathbb{R}^{2F'}$ of the weight vector $\mathbf{a}$.

The last step is to use the normalized attention weights α_{ij} to compute the new node embeddings for each node as a linear combination of the neighboring nodes as follows:

$$\mathbf{h}_i' = \sigma\left(\sum_{j\in\mathcal{N}_i}\alpha_{ij}\mathbf{W}\mathbf{h}_j\right), \tag{2.12}$$

where σ is an applied nonlinearity.

The process of applying graph attention layers can be repeated multiple times to perform *multi-head attention*, which recent studies argue to be beneficial [18, 19]. Multi-head attention is performed by computing K independent node embeddings as in equation 2.12 to obtain one node representation $\mathbf{h}_i^1, \ldots, \mathbf{h}_i^K$ for each *attention head*. The final node feature embedding is then retrieved by concatenation ($\|$) or averaging:

$$\mathbf{h}_i' = \Big\|_{k=1}^{K}\sigma\left(\sum_{j\in\mathcal{N}_i}\alpha_{ij}^k\mathbf{W}^k\mathbf{h}_j\right), \quad \mathbf{h}_i' \in KF', \quad \text{(concatenation)}$$

$$\mathbf{h}_i' = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in\mathcal{N}_i}\alpha_{ij}^k\mathbf{W}^k\mathbf{h}_j\right), \quad \mathbf{h}_i' \in F', \quad \text{(averaging)}.$$

It has recently been pointed out that the expressiveness of GAT can be increased with a small modification known as GATv2 [20]. The modification involves reordering the computations made in GAT by changing the self-attention operation $a(\cdot)$, which in the case of GAT is

$$e_{ij} = \text{LeakyReLU}\left(\mathbf{a}^T \cdot \left[\mathbf{W}\mathbf{h}_i \parallel \mathbf{W}\mathbf{h}_j\right]\right) \tag{2.13}$$

and was changed to GATv2 as

$$e_{ij} = \mathbf{a}^T\text{LeakyReLU}\left(\mathbf{W} \cdot \left[\mathbf{h}_i \parallel \mathbf{h}_j\right]\right). \tag{2.14}$$

With the use of a self-attention operation as in equation 2.14, it has been demonstrated that GATv2 outperforms GAT significantly in the *dictionary-lookup* problem, which was used to demonstrate the improved expressiveness. The main take-home message is that GAT suffers from *static attention*, which means that when computing the attention function α, there is always one neighboring node $\mathbf{h}_k$ that gets weighted at least as much as any other neighboring node $\mathbf{h}_j$, regardless of which

node $\mathbf{h}_i$ is being considered.

In actuality, the weight matrix $\mathbf{W}$ does not need to be shared across transformations described above, and bias terms could also be included. But this was omitted in the notation here for brevity.

For clarity going forward, the neighboring nodes are referred to as key vectors, and the current node as a query vector in the definitions that follow. Also, a collection of states of the function e in equation 2.9 constructs a set of scoring functions $\mathcal{F}$. That is, each state of the trainable weights in $\mathbf{W}$ and $\mathbf{a}^T$ represents a unique scoring function $e \in \mathcal{F}$.

Definition 2.2.1. *(Static attention). A (possibly infinite) family of scoring functions $\mathcal{F} \subseteq \mathbb{R}^{F \times F} \to \mathbb{R}$ computes static scoring for a given set of key vectors $\mathbb{K} = \{\mathbf{k}_1, \ldots, \mathbf{k}_n\} \subset \mathbb{R}^{\mathbb{F}}$ and query vectors $\mathbb{Q} = \{\mathbf{q}_1, \ldots, \mathbf{q}_n\} \subset \mathbb{R}^{\mathbb{F}}$, if for every $f \in \mathcal{F}$ there exists a "highest scoring" key $j_f \in \{1, \ldots, n\}$ such that for every query $i \in \{1, \ldots, m\}$ and key $j \in \{1, \ldots, n\}$ it holds that $f(\mathbf{q}_i, \mathbf{k}_{j_f}) \geq f(\mathbf{q}_i, \mathbf{k}_j)$. We say that a family of attention functions computes static attention given $\mathbb{K}$ and $\mathbb{Q}$, if its scoring function computes static scoring, possibly followed by monotonic normalization such as softmax.*

That is to say that for every function $f \in \mathbb{F}$ there is a highest scoring key j_f regardless of the query node.

To show that GAT computes static attention, we consider the family of scoring functions as in equation 2.13. It is then possible to demonstrate that the attention is static by changing the place of where the concatenation takes place:
Theorem 2.2.3. *A GAT layer computes only static attention, for any set of node representations $\mathbb{K} = \mathbb{Q} = \{\mathbf{h_1}, \ldots, \mathbf{h_n}\}$. In particular, for $n > 1$, a GAT layer does not compute dynamic attention.*

Proof. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph modeled by a GAT layer with some $\mathbf{a}$ and $\mathbf{W}$ values (Equation 2.13), and having node representations $\{\mathbf{h}_1, \ldots, \mathbf{h}_n\}$. By expressing the learned parameter $\mathbf{a}$ as the concatenation $\mathbf{a} = [\mathbf{a}_1 || \mathbf{a}_2] \in \mathbb{R}^{2F'}$ were $\mathbf{a}_{1,2} \in \mathbb{R}^{\mathbb{F}'}$, equation 2.13 can be rewritten as

$$e_{ij} = \text{LeakyReLU}\left(\mathbf{a}_1^T \mathbf{W}\mathbf{h}_i + \mathbf{a}_2^T \mathbf{W}\mathbf{h}_j\right). \tag{2.15}$$

With a fixed query node, we have:

$$\max(e_{ij}) = \text{LeakyReLU}\left(\mathbf{a}_1^T \mathbf{W}\mathbf{h}_i + \max(\mathbf{a}_2^T \mathbf{W}\mathbf{h}_j)\right) \tag{2.16}$$

and since there exists the highest scoring key node $j_{\max} \in \mathcal{V}$ which leads to a maximal value of its attention distribution $\{\alpha_{ij} | j \in \mathcal{V}\}$ due to the monotonicity of LeakyReLU and softmax, the attention α will only be static by definition 2.2.1. Definition 2.2.2 holds only for the constant mapping $\varphi(i) = j_{\max} \in \mathcal{V}$, thus GAT does not compute dynamic attention. □

To combat the issues with static attention the highest scoring key node needs to be dependent of the query node. This is the solved by dynamic attention which is the general and more powerful form of attention:

Definition 2.2.2. *(Dynamic attention). A (possibly infinite) family of scoring functions* $\mathcal{F} \subseteq \mathbb{R}^{F\times F} \to \mathbb{R}$ *computes dynamic scoring for a given set of key vectors* $\mathbb{K} = \{\mathbf{k}_1, \ldots, \mathbf{k}_n\} \subset \mathbb{R}^{\mathbb{F}}$ *and query vectors* $\mathbb{Q} = \{\mathbf{q}_1, \ldots, \mathbf{q}_n\} \subset \mathbb{R}^{\mathbb{F}}$*, if for any mapping* $\varphi : \{1, \ldots, m\} \to \{1, \ldots, n\}$ *there exists* $f \in \mathcal{F}$ *such that for any query* $i \in \{1, \ldots, m\}$ *and any key* $j_{\neq\varphi(i)} \in \{1, \ldots, n\} : f(\mathbf{q}_i, \mathbf{k}_{\varphi(i)}) > f(\mathbf{q}_i, \mathbf{k}_j)$*. We say that a family of attention functions computes dynamic attention for* $\mathbb{K}$ *and* $\mathbb{Q}$*, if its scoring function computes dynamic scoring, possibly followed by monotonic normalization such as softmax.*

By making $f(\mathbf{q}_i, \mathbf{k}_{\varphi(i)})$ the maximal in $\{f(\mathbf{q}_i, \mathbf{k}_j) | j \in \{1, \ldots, n\}\}$ dynamic attention can select every key $\varphi(i)$ with the query i. Before proving that a GATv2 layer computes dynamic attention, we demonstrate an important feature of non-linear activation functions, such as LeakyReLU:

$$\begin{aligned} e_{ij} &= \mathbf{a}^T \text{LeakyReLU}\left(\mathbf{W} \cdot [\mathbf{h}_i || \mathbf{h}_j]\right) = \\ &[\mathbf{a}_1^T || \mathbf{a}_2^T] \text{LeakyReLU}\left(\mathbf{W}_1\mathbf{h}_i + \mathbf{W}_2\mathbf{h}_j\right) \neq \\ &\mathbf{a}_1^T \text{LeakyReLU}\left(\mathbf{W}_1\mathbf{h}_i\right) + \mathbf{a}_2^T \text{LeakyReLU}\left(\mathbf{W}_2\mathbf{h}_j\right) \end{aligned}$$

Thus the highest scoring e_{ij} is now dependent on both the query node $\mathbf{h_i}$ and key node $\mathbf{h}_j$ since we can no longer have these variables in two separate terms.

Theorem 2.2.4. *A GATv2 layer computes dynamic attention for any set of node representations* $\mathbb{Q} = \mathbb{K} = \{\mathbf{h}_1, \ldots, \mathbf{h}_n\}$.

Proof. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph modeled by a GATv2 layer, having node representations $\{\mathbf{h}_1, \ldots, \mathbf{h}_n\}$, and let $\varphi : \{1, \ldots, m\} \to \{1, \ldots, n\}$. We define $g : \mathbb{R}^{2F} \to \mathbb{R}$ as follows:

$$g = \begin{cases} a_i & \exists i : \mathbf{x} = [\mathbf{h}_i || \mathbf{h}_{\varphi(i)}] \\ b_i & \text{otherwise} \end{cases} \qquad \text{where } a_i, b_i \in \mathbb{R} : a_i > b_i \tag{2.17}$$

Then, we define the continuous function $\bar{g} : \mathbb{R}^{2F} \to \mathbb{R}$:

$$\bar{g}([\mathbf{h}_i || \mathbf{h}_j]) = g([\mathbf{h}_i || \mathbf{h}_j]), \forall i, j \in \{1, \ldots, n\} \tag{2.18}$$

For all other inputs $\mathbf{x} \in \mathbb{R}^{2F}$, $\bar{g}$ realizes any values such that continuity of $\bar{g}$ is maintained. This is possible since $\bar{g}$ is only fixed to a finite set of points.
Thus, for every query node $i \in \mathcal{V}$ and key node $j_{\neq\varphi(i)} \in \mathcal{V}$:

$$a_i = \bar{g}([\mathbf{h}_i || \mathbf{h}_{\varphi(i)}]) > \bar{g}([\mathbf{h}_i || \mathbf{h}_j]) = b_i \tag{2.19}$$

If we define a scoring function e as in equation 2.14, parameterized by its trainable weights:

$$e_{\mathbf{W},\mathbf{a}} = \mathbf{a}^T \text{LeakyReLU}\left(\mathbf{W} \cdot [\mathbf{h}_i || \mathbf{h}_j]\right) \tag{2.20}$$

By then utilizing the universal approximation theorem which is defined for continuous functions, e can approximate $\bar{g}$ for any compact subset of $\mathbb{R}^{2F}$ [21]. This also means that e approximates g in the finite set of points were equation 2.18 holds. Thus, for any sufficiently small $\epsilon \in (0, {}^{a_i}/_2 - {}^{b_i}/_2)$ there exists parameters $\mathbf{W}$ and $\mathbf{a}$ such that for every query node $i \in \mathcal{V}$ and key node $j_{\neq\varphi(i)} \in \mathcal{V}$:

$$e_{\mathbf{W},\mathbf{a}}(\mathbf{h}_i, \mathbf{h}_{\varphi(i)}) > a_i - \epsilon > b_i + \epsilon > e_{\mathbf{W},\mathbf{a}}(\mathbf{h}_i, \mathbf{h}_j) \tag{2.21}$$

and due to the monotonicity of softmax

$$\alpha_{i,\varphi(i)} > \alpha_{i,j}$$

Thus given a query node i, any key node $j \in \{1, \ldots, n\}$ can recive the higest attention score. □

2.2.3 Message passing neural networks

Message passing neural networks operate on the idea of iteratively transferring information between nodes that are neighbors through the neighboring edges for a general undirected graph $\mathcal{G}$. The model is made of 2 phases, firstly a message passing phase and secondly a readout phase, which runs for T time steps [16, 22].

The message passing phase uses the message function M_t, which is the message sent from one node to its neighbors. This function depends on the hidden state of the nodes and neighbors, as well as the edge features between the nodes. The message function enables the following equation

$$\mathbf{m}_v^{t+1} = \sum_{\omega \in N_v} M_t(\mathbf{h}_v^t, \mathbf{h}_w^t, \mathbf{e}_{vw}), \tag{2.22}$$

where $\mathbf{m}_v^{t+1}$ are the messages at the node v during the time step $t+1$, $\mathbf{h}_v^t$ is the hidden state of node v, $\mathbf{h}_w^t$ is the hidden state of node w, $\mathbf{e}_{vw}$ are edge features from node v to w, and N_v are the neighbors of node v . This phase also uses a node update function U_t, which accumulates all incoming messages and updates the node's own hidden state. This is illustrated in the equation below [16, 22]

$$\mathbf{h}_v^{t+1} = U_t(\mathbf{h}_v^t, \mathbf{m}_v^{t+1}). \tag{2.23}$$

The readout phase adapts a readout function R which outputs a graph-level representation of $\mathcal{G}$. This representation is denoted by $\hat{\mathbf{y}}$ and written as [16, 22]

$$\hat{\mathbf{y}} = R(\{\mathbf{h}_v^t | v \in \mathcal{G}\}) \tag{2.24}$$

We have that the message functions M_t, the readout function R, as well as the node update functions U_t are learned differentiable functions. The readout function R has functions on the set of node states and thus has to be invariant to all permutations of those states for the entire neural network to be invariant to graph isomorphism,

which was talked about earlier on. Another way of learning the edge features in an MPN is by initiating hidden states for all the edges in a graph, denoted by $h^t_{l_{vw}}$, which are updated analogously for equations 2.22 and 2.23. A learned differentiable edge function, denoted by E has to be used to compute

$$\mathbf{l}^{t+1}_{vw} = E(\mathbf{h}^t_v, \mathbf{h}^t_w, \mathbf{l}^t_{vw}). \tag{2.25}$$

However, this idea of learning the edge features has only been experimented with in very few works [22]. There are several variants of MPNN, where the directed message passing neural network (D-MPNN) is one that is focused on further since it is implemented in one of the GNNs used in this project.

2.2.3.1 Directed message passing neural networks

Directed message passing neural networks (D-MPNN) can be seen as an extension of Message passing neural networks (MPNN) where the general undirected graph $\mathcal{G}$ is replaced by the directed counterpart $\mathcal{G}_{dir}$. The consideration of direction for the flow of information is crucial for the representation of information flow during the message passing phase of a molecule [23]. This is due to MPNN architecture's messages $\mathbf{m}^t_v$ focusing on atoms, rather than the bonds, which D-MPNN does. Usually, the idea has been to view molecules as undirected molecular graphs. D-MPNNs set forth a method to use directed bonds, which in turn averts complex or redundant loops included in the message passing phase of the MPNN algorithm. A consequence of directed bonds is the mitigation of over-mixing of information and thus reducing over-smoothing [24]. An example of such an illustration of a molecule is shown below

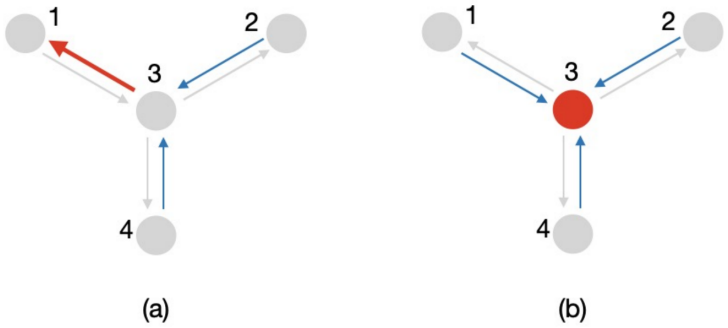


Figure 2.4: An example of a molecular graph with directed bonds, where the left figure shows an update of bond states, while the right figure shows an update of atom states.

D-MPNNs conserve representations of messages from either direction of two nodes i and j, which in this case are $\mathbf{m}^t_{ij}$, denoting the messages from node v to j and $\mathbf{m}^t_{ij}$, denoting the messages from node j to i, at iteration t. Instead of collecting the messages from the neighboring atoms, the model collects the messages from the neighboring bonds, where the messages from each bond are updated depending on the cumulative bond messages $\mathbf{m}^t_{ki}$, where $k \in \{N(i) \setminus j\}$. As a result, D-MPNN has

a greater influence over the flow of information across the entire molecule, which in turn results in a far more detailed molecular representation [23].

2.3 Models selected that use a graph neural network architecture

This project has selected AttentiveFP and Chemprop-IR as the implemented models, which use a graph neural network as an architecture, to train on our dataset. In the following section, a deeper analysis of the models is presented.

2.3.1 AttentiveFP model architecture

AttentiveFP is a model that incorporates both message passing and graph attention mechanisms in a graph convolution. When the model was introduced in 2019, it was demonstrated that AttentiveFP achieved state-of-the-art performance on a variety of prediction tasks[13]. The core concept of AttentiveFP is to make use of multiple attention and message passing layers with intermediate cells of gated recurrent units (GRU), which yields the model the potential to learn intramolecular interactions based on prediction tasks and data. This is a highly desirable property when predicting UV absorption spectra of a molecule since excitations in a molecule are dependent on both short and long-range interactions[25].

Figure 2.5 depicts the design and architecture of AttentiveFP in detail, Where (a) is a demonstration of AttentiveFP's ability to capture the importance of distant atoms compared to other graph-based neural networks. Subfigure (b) represents a high-level overview of the architecture of AttentiveFP, where the initial molecule representations to the left pass through a fully connected layer to generate initial node embeddings with equal dimensions. This is achieved with a concatenation of all neighboring node embeddings and connecting edge features for each node, this ultimately yields embedding vectors of different lengths. To get the embedding vectors to consistent length, a linear transformation and a nonlinear activation function are applied. These transformed embeddings are then passed on to perform graph attention. Subfigure (c) illustrates the graph attention mechanism acting on atom number 3 within the aspirin molecule. *Alignment* corresponds to equation 2.9, *weighting* corresponds to equation 2.10, and *context* corresponds to the computation of embeddings in equation 2.12. Subfigure (d) consists of k GAT layers described in (c) with intermediate GRU cells. The GRU cells receive the new context embedding C_ν^i and the current embedding h_ν^i as input and ideally filters out a mix of important information from the two embeddings to generate the new output embedding. Subfigure (e) A supervirtual node (super-node) S is constructed by concatenation of all final node embeddings to combine individual state vectors and generate a state vector that represents the entire molecule. The node S is then connected to all nodes in the molecule, and the same type of graph attention layers are performed for t time-steps. When the final embedding in section (e) in figure 2.5 is reached, a linear layer is used to perform the prediction with the desired number of prediction

tasks.

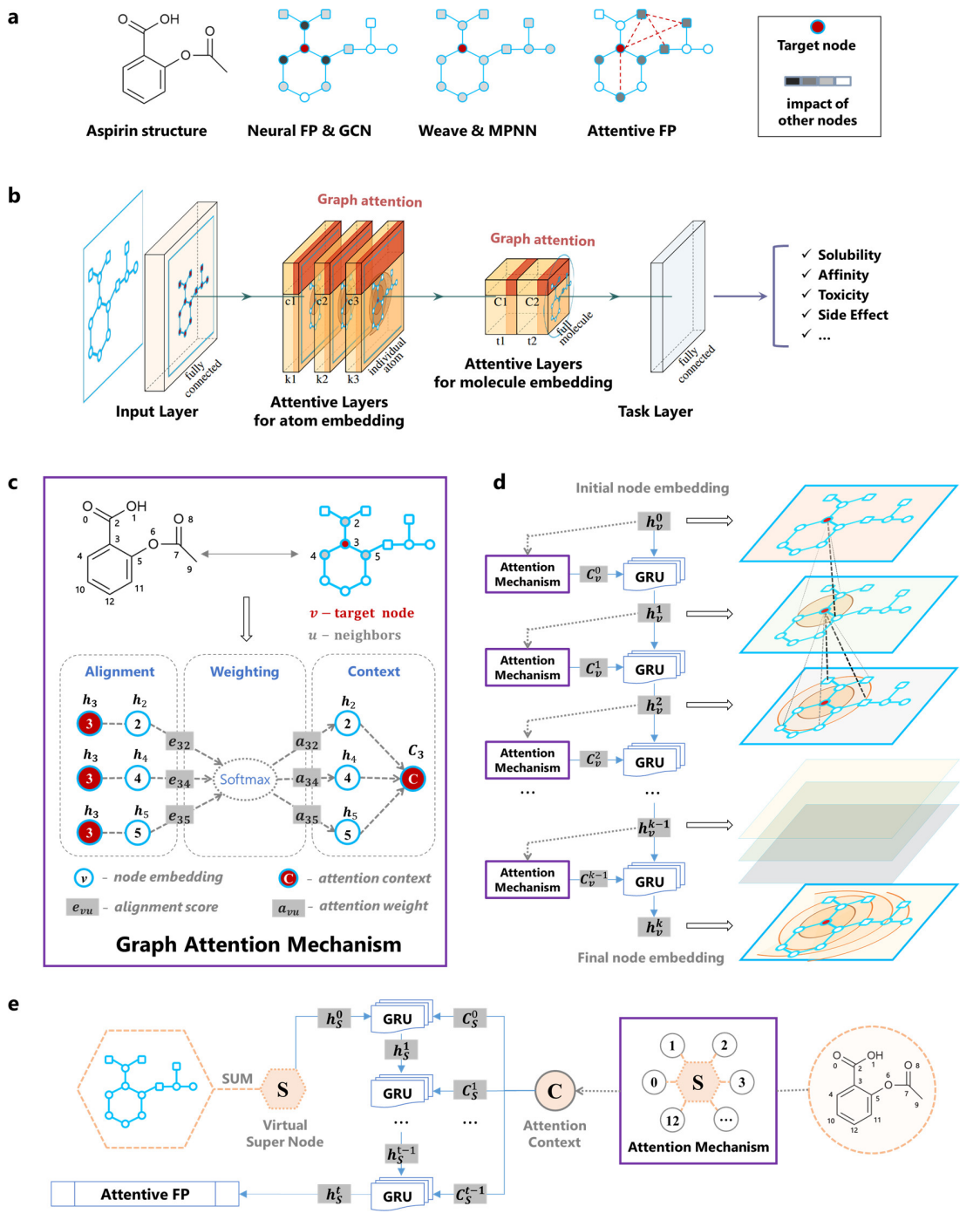


Figure 2.5: An example of the model structure with Aspirin used as an example molecule. Reprinted (adapted) with permission from Journal of Medicinal Chemistry. Copyright 2020 American Chemical Society.

The initial molecule embedding used in the first step of section (b) in figure 2.5 is constructed in a preprocessing procedure that generates all molecular graphs. A molecular graph is represented by the node (atom) features using either scalar or

one encoding and edge features (bond) also using scalar or one hot encoding. Connectivity between nodes and the corresponding edges is then set up by a coordinate list (COO) to represent the molecules as isomorphic graphs. All features used are collected using the RDKit where SMILES representation of a molecule is translated to the isomorphic graphs with node and edge feature encoding.

An easy-to-use and modular implementation of AttentiveFP is provided in the open-source PyTorch Geometric library where the model design of AttentiveFP has been isolated from all other functionalities such as the construction of dataset, data loading, and definition of loss function [26]. The modality provides the freedom to use any loss functions and metrics of accuracy. Since the source code is in the public domain it is also possible to make adjustments to the model, one such adjustment is to incorporate *multi-order graph attention* similar to MoGAT [27]. The main idea is to use one super-node in between each layer when performing graph attention. Referring to Figure 2.5, it corresponds to inserting one instance of section (e) in between each layer in section (d), thus capturing multiple graph embeddings at different stages of the message passing process.

2.3.2 Chemprop-IR

Chemprop-IR is a directed message passing neural network (D-MPNN) that is an extension of Chemprop, which is a message passing neural network (MPNN). The purpose of Chemprop is to predict solubility or energies within the bonds for molecules. Chemprop-IR shifts its focus from this area to predicting IR spectra with the help of selected suitable metrics and spectra loss functions optimized to enhance the process. The model can train on molecules of different phases, such as gas, liquid, CCl_4 solution, and mineral oil suspension. Chemprop-IR has mainly been tested with molecules containing the atoms C, H, O, N, Si, I, Br, Cl, S, F, and P [10].

In order for the model to work with diverse spectra, certain pre-training tasks have to be performed. Firstly, the resolution for all the spectra has to be constant rather than continuous, which in this case means that the discretization of the wavelengths is set constant for all spectra. Secondly, including a diverse range of spectra means that the peaks will have a wide range of intensities, which are seen as amplitudes. Therefore, it is necessary to include a normalization step [10]. This is to achieve uniform data that eases the model's training procedure and therefore increases accuracy. The normalization is described with the equation below

$$y_{norm,i} = \frac{y_i}{\sum_{j=1} y_j}, \tag{2.26}$$

where $y_{norm,i}$ is the normalized intensity for index i (where the index corresponds to a wavelength), and y_i is the original intensity for index i, which is divided by the sum of all intensities for that spectra. This normalization ensures that the intensities $y_{norm,i}$ are between 0 and 1 while preserving the features of the spectra between the stages. A benefit of the normalization is that the model learns during training that the intensities are within the interval of 0 and 1, thus lowering the chance of

the model to be outside this range, which in turn increases its accuracy of prediction.

There is a wide variety of metrics and spectra loss functions to choose from, each specializing in predicting different tasks. The most relevant metrics to predict spectra for Chemprop-IR are scaled mean square root error (SRMSE) and scaled mean square error (SMSE). In this case, we focus on SRMSE since it is just the square root of SMSE. SRMSE can be described as the following

$$\text{SRMSE} = \sqrt{\epsilon + \sum_{i=1}^{n} \frac{1}{n} \left(\frac{y_{pred,i}}{\sum_{j=1}^{m} y_{pred,j}} - y_{norm,i} \right)^2}, \tag{2.27}$$

where $y_{pred,i}$ is the predicted intensity which the model produces for position i (which corresponds to a wavelength), and ϵ is a small positive constant, often 10^{-8}, which provides numerical stability for SRMSE [10]. Equally important is the choice of spectra loss function, which as the metric, is designed based on different tasks. A valuable spectra loss function is the spectral information divergence (SID). The SID for a spectra prediction vector $\mathbf{y_{pred}}$ and a spectra target vector $\mathbf{y_{target}}$ is

$$\text{SID}(\mathbf{y_{pred}}, \mathbf{y_{target}}) = \sum_{i=1}^{n} y_{pred,i} \ln \left(\frac{y_{pred,i}}{y_{target,i}} \right) + y_{target,i} \ln \left(\frac{y_{target,i}}{y_{pred,i}} \right). \tag{2.28}$$

The idea behind SRMSE and SID is that the lower the values are, the better they fit to the data which they are training and validating on. According to the creators of Chemprop-IR, choosing SID as the spectral loss function is a reasonable decision since SID has shown promise in predicting spectra thanks to its target-weighting feature. A benefit of this procedure is the reduced impact of noise or extreme data for the finalized predicted spectra and the possibility of selecting regions of spectra of more importance. This is relevant since we are mostly interested in regions of the spectra where peaks appear. The reasoning behind SRMSE is that a scaling factor is included which improves the peak resolution and quantitative performance compared to metrics such as the mean absolute error (MAE) [10].

As previously mentioned, Chemprop-IR offers multiple tasks and can therefore accept both positive and negative inputs and outputs. Considering that this project handles spectra, which has non-negative intensities, the model must be set to only be allowed to accept positive input values. Therefore, it is important to implement an activation function that handles the inputs to ensure non-negative inputs. One common example of such an activation function is the rectified linear unit (ReLU) function, which is defined as

$$\text{ReLU}(x) = \max(0, x), \tag{2.29}$$

where x is the input value. This activation function has been proven to be a solid choice and suitable in the case of predicting spectra for different models [10].

The setup of parameters which the model chooses to save is the one resulting in the lowest validation loss. Therefore, Chemprop-IR has the goal of reducing the

validation loss in any way possible, and could thus result in negative predicted intensities to do so. One way to avoid this scenario for predicting spectra is to apply an output activation function, which handles the output of the model to follow set rules. One output activation function that ensures positive outputs is the exponential function, which is

$$f(y) = e^y, \tag{2.30}$$

where y is the value produced after all layers and sent to the output to output value $f(y)$. This finalized value is then ready to be stored and considered as the model's predicted intensity. This procedure is done for several epochs in order for the model to learn and detect patterns among the molecules and further reduce the validation loss to gain more accurate predictions. The most optimized model is determined to be the one with predictions that are the most similar to the validation dataset, which is denoted by the model with the lowest SRMSE throughout all the epochs.

In summary, the entire process of training Chemprop-IR to a dataset to predict spectra also has the following visual illustration.

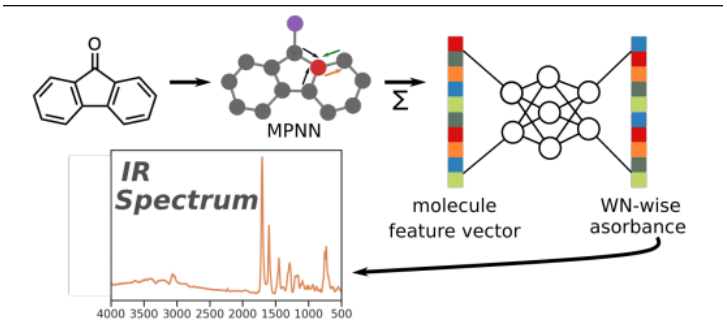
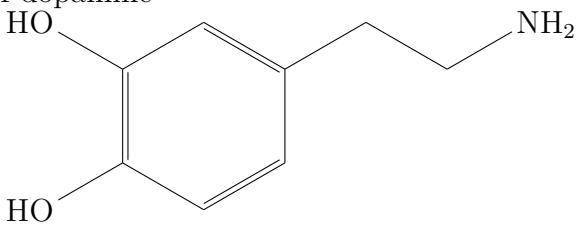


Figure 2.6: An illustration of the process of Chemprop-IR for IR spectra. The SMILES represent a molecule, which is transformed into its molecular representation of a D-MPNN. This D-MPNN propagates the messages between the atoms to learn about the molecule's features and structure, illustrated by Σ. The information is stored in a molecule feature vector, which contains information about the atoms and types of bonds of the molecule and is used to plug into the model. The model trains on the dataset to make predictions, which are stored in a vector by the name WN-wise absorbance. This in turn produces the final spectrum.

2.4 Simplified Molecular Input Line Entry System (SMILES)

An alternative way of representing molecules, which has gained popularity among GNNs and molecules is SMILES. SMILES stands for simplified molecular input line entry system and is used by computer software as a way to interpret a molecule's 3D structure. SMILES does this by using a string notation consisting of atomic symbols and bonds. Some examples of atomic symbols are C, F, O, S, N, P, and Cl, while bonds are represented by the symbols. The bonds are "-" which stands for

a single bond, "=" which stands for a double bond, "#" which stands for a triple bond, and finally ":" which represents aromaticity. There are atomic symbols that are necessary to write within brackets, such as [Au] [28]. Below is an example of the chemical structure of dopamine



dopamine

Dopamine has the corresponding SMILES representation C1=CC(=C(C=C1CCN)O)O [29].

However, there are a few drawbacks to consider when dealing with SMILES. There is a risk of generating invalid SMILES sequences for molecules. These errors appear in the form of incorrect chemical representations and do not encode the correct molecules or no existing molecules at all. Another drawback is the ambiguity of the notation, which means that one molecule can have multiple valid SMILES notations. A consequence of this is that it can confuse the model that is interpreting the notation about what molecule is being handled without some sort of canonicalization [30]. Therefore, it is important to include a validation step in translating molecules into SMILES and exclude those with invalid representations rather than trying to adjust these. It is equally important to try to differentiate for which molecular structure is being dealt with to reduce potential confusions and errors which the models could encounter.

3
Methods

The purpose of this section is to discuss the different necessary procedures to implement the models. Firstly, the topic of achieving and formatting the dataset is discussed. Secondly, the setups for the two different selected models, AttentiveFP and Chemprop-IR, are discussed and how these will be adjusted and modified to accept the modified dataset. Since there are two different models involved, they have to be set so that they perform the same tasks during the training procedure in order to guarantee as close of a common ground as possible. Finally, the technical aspects, such as programming language, software, hardware, and limitations are touched upon.

3.1 Background and handling of the molecules and datasets

This project used a dataset of molecules named ORNL_AISD-Ex, which was supplied by Oak Ridge National Laboratory. The dataset consists of 10,502,904 molecules, which provides information about the molecules, such as HOMO-LUMO gaps and the string notations for all molecules in SMILES [12]. The information of interest in our case was the SMILES and the UV-Vis absorption spectra, which contained excitation energies that corresponded to absorption peak positions. These energies were used to measure the probability of the molecules emitting UV-Vis light when excitation between resting and excited states occurs. SMILES were used to represent the molecules in the form of a string notation by presenting the atom and bond types. There are 5 possible non-hydrogen elements that appear in the SMILES of the molecules which are carbon (C), nitrogen (N), oxygen (O), sulfur (S), and fluorine (F). Each molecule can range in length from 5 to 71 non-hydrogen atoms [12].

Despite containing the necessary data needed to further complete the goals of implementing models to train on these, the data was not in a format that was ready for our case. There were some modifications needed in order for the data to be ready for use. The first important step implemented was the conversion from excitation energies, which were in the unit eV, to represent in terms of wavelength, which is in the unit nm. This was done with the formula

$$\lambda_m = \frac{hc}{E}, \tag{3.1}$$

where h is the Planck constant, c is the speed of light, E is the excitation energy in eV that is obtained from the HOMO-LUMO gaps, and λ_m is the wavelength in unit

m [31]. Since we were interested in the unit nm, we had to convert λ_m to the unit nm, which is the same as

$$\lambda = \lambda_m \cdot 10^9, \tag{3.2}$$

where 10^9 is a conversion factor corresponding to converting from meter to nanometer, and λ is the converted wavelength in nm [31].

Since there were over 10.5 million molecules, it became computationally expensive to include all the data in a continuous manner. An alternative, that was implemented, was to discretize the data with a constant resolution of 6 nm [31]. An implication was fewer data points and parameters involved in the training of the models, which in turn reduced the complexity and the necessary resources required of the models. It also enabled the training to be held in a shorter time that was possible to fit within the time limits.

Another important aspect was determining the range of UV-Vis, which was important for the data. These considerations were made since we wanted to limit ourselves to the most important range. Too wide of a range meant an increase in unnecessary complexities and computations, while too short of a range risked excluding important data. The range of wavelengths for UV-Vis are between 10 and 400 nm, where UVA is between 315 and 400 nm, UVB is between 280 and 315 nm, and finally, UVC is between 100 and 280 nm. Most UV-Vis spectra below 200 nm are considered VUV (Vacuum-UV) and require specialized equipment to capture, due to being easily absorbed by air. Wavelengths above 400 nm start to enter the visible light range outside the range of interest. Therefore, to avoid cutouts and ensure safe margins of range, we considered the UV-Vis spectra of interest as between 150 and 450 nm [32].

With the data within interesting wavelengths extracted and converted to the necessary units, the next step was to smoothen the data with a filter function and normalized, since the maximum intensities of all spectra resulted in a diverse range of values. A Gaussian filter was used, which used a Gaussian function according to the following

$$f(x, a) = a \exp\left(-\ln(2)\left(\frac{(x-m)}{w}\right)^2\right), \tag{3.3}$$

where a is the intensity of the unfiltered data, x is the wavelength, m is the mean of the range of wavelengths, and w is the standard deviation of the range of wavelengths. The set value for m was 300 and for w it was 10 [31]. This filtered data was then normalized across all molecules to ensure for both of the models that the intensities only ranged between 0 and 1. This in turn simplified the training procedure for both of the models by including less diversity of maximum peaks. The normalization was done as follows

$$y_{i,norm} = \frac{y_i}{\sum_{j=1} y_j}, \tag{3.4}$$

where y_i is the intensity at position i, where the position corresponded to some wavelength, and $y_{i,norm}$ is the normalized intensity for the same position i. This was

done to preserve the feature

$$\sum_{j=1} y_{j,norm} = 1, \tag{3.5}$$

for all the molecules. To reduce the noise among the data, intensities less than 0.0001 were set to 0 so that there were no small bumps in the spectra contributing to excessive noise that affected the finalized predictions.

3.2 Choice and setups of the models

Since two different models were implemented, they each needed their own setups and modifications to work on the provided dataset. The implementations of the models were set to be as similar as possible while being able to perform the task of predicting UV-Vis spectra for molecules. This included hyperparameters, such as an identical validation as well as an identical data split into equal ratios of training, testing, and validation. The reason being was to minimize the risk of unfair advantages as well as achieving comparable final results. These models were chosen since they are GNNs that show similar promises in predicting UV-Vis spectra for molecules represented as SMILES. Since both models only require the SMILES of the molecules and the corresponding intensities at each wavelength within the UV-Vis range, we could guarantee that the models would work for our task. It was therefore within our interest to implement these and compare the end results.

3.2.1 AttentiveFP

As briefly mentioned in section 2.3.1, the raw data provided by Oak Ridge National Laboratory undergoes some initial preprocessing steps using RDKit to construct isomorphic graphs to represent molecules. In the conversion from SMILES to graphs, a unique index becomes associated with each node (atom) in the molecule. These indices are used to construct the coordinate list to keep track of the nodes connected with a bond. The encoding of node and bond attributes are listed in table 3.1 and table 3.2 respectively.

Node (Atom)		
Attribute	Encoding	Vector length
Atom number	Scalar	1
Chirality	One-hot	9
Degree	Scalar	1
Formal charge	Scalar	1
# *H* atoms	Scalar	1
# radical electrons	Scalar	1
Hybridization	One-hot	8
Is aromatic	One-hot	2
Is in ring	One-hot	2
Total vector length		26

Table 3.1: Attribute and method of encoding of nodes (atoms) with the corresponding vector length for each embedding.

Edge (Bond)		
Attribute	Encoding	Vector length
Bond type	One-hot	7
Stereo	Scalar	1
Is conjugated	One-hot	2
Is aromatic	One-hot	2
Is in ring	One-hot	2
Total vector length		14

Table 3.2: Attribute and method of encoding of edges (bonds) with the corresponding vector length for each embedding.

Some encoding types listed in tables 3.1 and 3.2 could arguably be using scalar encoding instead, such as *Is in ring*, but one-hot encoding was used for these attributes to increase the number of input parameters to the model which has the potential to increase the expressivity of the initial embedding.

An example of an important step for most deep learning models is the search for good-performing combinations of hyperparameters. In the case of AttentiveFP, there are multiple parameters and design choices to consider, such as the number of hidden channels, GAT layers, attention heads, time steps, and dropout to name a few. Ex-

tending the search space is the possibility of using different graph attention schemes, such as GAT, GATv2, and whether or not to use masked attention. There are also the parameters related to the training of the model, such as batch size, learning rate, choice of the optimizer, and parameters related to the optimizer, also the loss function used can impact the outcome significantly. Needless to say, testing all combinations of parameters and settings is unfeasible and would require an obscene amount of computational resources. To combat this, the set of hyperparameters is reduced, where the most essential ones are focused on initially.

Other important considerations are the potential problems of over-smoothing and over-squashing. The effects of over-smoothing typically occur in GNNs when messages are passed and aggregated in multiple layers. This means that a larger network with an increasing number of layers does not necessarily increase prediction performance. The cause of this degradation of performance, as the number of layers increases, is typically caused by unwanted noise in node embeddings getting amplified in each layer, which causes the embeddings to mostly consist of noise and thus be uninformative. With the effects of over-smoothing in mind, a small number of GAT layers are initially used and gradually increased.

The effects of over-squashing are less of an issue in small graphs, and thus less of a concern when dealing with small molecules, as in this case. Over-squashing is mostly an issue when bottlenecks occur in the graph. Bottlenecks can be thought of as a single edge or a small number of edges that are responsible for the aggregation and message passing between two larger clusters of nodes in a graph. In these cases, the compression in the aggregation step can be too aggressive, and thus information is lost in the process, leading to over-squashing. Two ways to combat this potential issue with AttentiveFP are, (*i*) to include artificial edges in the molecule graphs that can be used to pass messages between nodes, (*ii*) to increase the number of attention heads and use concatenation as the aggregation function. The effects of over-squashing are assumed to be a non-issue as a starting point due to the relatively small size of molecule graphs. An increasing number of attention heads would also increase the training time significantly which is undesirable, especially in the early stage when in search of well-performing hyperparameters.

To gain some insight into the model quickly, small networks are initially trained and then gradually increased to find a suitable search domain, where further increasing the size of the model no longer yields significant improvements to the performance. When a suitable range of hyperparameters is found, sweeps over several combinations can be performed with the use of tools provided by Weights & Biases (W&B) in an automated fashion [33].

3.2.2 Chemprop-IR

The setup of hyperparameters for Chemprop-IR is essential for the training procedure of the model on the dataset. Therefore, in this section, we discuss the necessary

pre-training steps needed to be performed to initiate the training procedures, as well as the possible combinations of hyperparameters.

Firstly, before the model could proceed with the training, the format of the dataset could only be accepted in a CSV file. This file had to contain the header "smiles, 150, 156, ...,444,450", due to the 6 nm discretization and the UV-Vis absorption range of 150 nm to 450 nm. This step was especially important to shift the focus of the model from the IR spectra to the UV-Vis spectra. The first text of the header is "smiles" since each row of the CSV file represents a molecule, where all rows start with the SMILES representation of the selected molecules. The SMILES are followed up by the intensities for each of the corresponding wavelengths in that selected column, where a comma separates each intensity. A hyperparameter that is related to the dataset, which needed to be specified before training, was the dataset type. The dataset type had to be set to spectra to determine the possible metrics, spectral loss functions, and remaining hyperparameters. This was due to not all combinations of hyperparameters being suitable for the task of predicting spectra. Thereafter, the dataset was, just as for AttentiveFP, split into training, testing, and validation sets with the corresponding rations set to 94% of the data as a training set, 1% as a testing set, and 4% as a validation set.

Furthermore, the next step was to set the activation function to ReLU, as described in equation 2.29, which ensured that the intensities/ inputs of the model were positive. Secondly, an exponential output activation function, mentioned in equation 2.30, was used since negative intensities, which are impossible to achieve in the real world, had to be avoided at all costs. The cause of negative intensities potentially being included is to reduce the loss function. The choice of metric and spectral loss function was essential for the training step of the model. The metric was set to SRMSE according to equation 2.27 and the spectral loss function was set to SID according to equation 2.28. The choices of metric and spectral loss function were set due to showing more accurate predictions compared to other choices, such as MSE and MAE for the metrics and loss function for spectral data.

The training procedure was conducted for up to 100 epochs as well as 2 additional warm-up epochs. Warm-up epochs provided the benefit of increasing performance and thus a better optimization of the finalized hyperparameters. The only drawback was a longer initiation of training. However, an optimal solution was reached for fewer number of epochs. Since an epoch could take up to several hours, a constant learning rate of $7.5 \cdot 10^{-5}$ was set. This was due to the need of reaching the optimized model in as few number of epochs as possible without the possibility of overshooting the solution or ending up with an overfitted model. Another important aspect was the importance of finding a correct balance of the batch size. An increased batch size reduces the training time for each epoch. However, this comes at a cost of reduced accuracy of predictions and increased demand for hardware resources. Therefore, it was important to balance the choice of batch size, which in this case was set to 64 due to recommendations from the creators. Depth was another argument available for the model that stands for the number of message passing layers. It resulted in

deeper and more complex models that captured more details of the predicted model. A model with a higher number of message passing steps has a higher accuracy and ability to train on larger molecules with far apart atoms. Unfortunately, there are drawbacks of increasing the depth, for instance, longer training sessions due to more computational cost per epoch and overfitting risks. Therefore, the depth was set to 3 message passing layers.

One way to combat the drawbacks of increasing depth was to include dropout, which randomly dropped out a certain fraction of the neurons in each layer. This in turn led to reduced dependencies between the remaining neurons, which this reduces the risk of overfitting. However, it was important not to have too high of a fraction, since it prevented the model from learning patterns due to excluding too much important data. This fraction was therefore set to 0.1. Finally, there were other hyperparameters that needed to be taken into consideration that were less obvious as to what to set them as in the beginning. Since it was impossible to try every possibility of these, they had to be experimented with. By including the fact that some combinations did not work for our goal, it reduced the number of available hyperparameter combinations. Therefore, tests on smaller datasets were performed to experiment which choices of hyperparameters were possible, and then experimented further with on the entire dataset. Such examples of hyperparameter were the size of the hidden layers of the main MPNN, denoted by hidden size, as well as the size of the hidden layers in the feed-forward network of the model, denoted by FFN hidden size. The size of these hyperparameters determined the complexity of the resulting model, where an increased size captured more complex patterns at the cost of an increased risk of overfitting. Therefore, it was important to experiment with these different values to see where the most efficient trade-off lied. In this project, both the FFN hidden size and the hidden size were tested for 2200, 2800, and 3400, and evaluated where the most optimal one was selected for further analysis.

3.3 Programming language and additional software

The entirety of this project was carried out in Python due to several benefits. Firstly, the code availability and creations of both models were provided in Python by the creators of the models. Thus, there were many dependencies on libraries, packages, etc. that relied on the usage of Python. Secondly, additional modifications needed to be done to predict UV-Vis absorption spectra were most efficiently done via Python due to the diverse selection of packages and toolkits within the topic of cheminformatics. One example of such a vital toolkit used throughout this project is RDKit, which is an open-source toolkit for cheminformatics and machine learning that offers interfaces for Python and C++. The most relevant area of RDKit is the methods of 2D and 3D molecular operations that it offers, which were implemented when handling the dataset of molecules [34].

There are also libraries for Python that specialize in machine learning and sim-

plifying, as well as enabling the prediction of UV-Vis spectra. One library that both models used was PyTorch, which is an open-source machine learning library. The models benefitted from the library due to the incorporation of GPUs to increase the efficiency of training and thus speed up the process. PyTorch also provides a more efficient way to handle molecules of different lengths [35]. A package that simplified the handling of the necessary additional packages for the models was Conda. Conda provides an efficient way of installing packages and solving dependencies while doing so in isolated environments, where a user can have multiple environments for multiple models [36]. This ensures no conflicting dependencies or inflicting package versions, as well as the possibility of storing both models on the same computing unit without interferences.

3.3.1 Dealing with large datasets in Python

When working with large datasets consisting of millions of molecules, the topic of high RAM consumption and computational efficiency becomes an important topic among graph neural networks. Graph neural networks consume more RAM for larger datasets due to loading the structures onto the RAM, which often is linearly correlated to the size of the dataset. After loading the data, there were several pre-processing steps done before the training to prepare them, which consumed additional memory. The original intention of the usage of these models is for up to 100,000 molecules, according to the creators of the models [10]. Therefore, they did not take into account these limitations, since the models were not intended to be trained for millions of molecules. However, there were several methods to work around the performance-related areas that enable the training of 10.5 million molecules.

One example used for Chemprop-IR was the usage of HDF5 file formats. HDF5 is the latest version of the hierarchical data format (HDF), which allows the user to use data of large size and easily manipulate it from NumPy by splitting a large dataset into smaller datasets that are stored in one file and categorized. HDF5 is accessed from the h5py package, and the files implementing this data format are denoted by .h5py at the end of the file name. By replacing the CSV format with the HDF5 format, we were taking part in several performance-related benefits of training on all the molecules. One example is HDF5 outperforming CSV in terms of writing and reading speeds, as well as a reduced storage size. One drawback of HDF5 compared to CSV is its less user-friendly interaction options. HDF5 uses a complex hierarchical format, which means that it is more intricate to handle since it is not possible to read or open such a file directly. Rather, the file has to be opened in specific software to view the contents. This requires a more monitored and careful interaction [37, 38].

Another example is the implementation of OnDiskDataset for AttentiveFP, which is a functionality provided by PyTorch Geometric [26]. OnDiskDataset circumvents the restrictions of storing all data in to RAM by storing data in a *sqlite* database on disk [39]. OnDiskDataset is then compatible with the standard dataloader in

PyTorch, which enables the user to sample batches as desired. Depending on the structure of the data, batch size and computational demands posed by the size and design of a model, the usage of OnDiskDataset has the potential to reach performance comparable with implementations when all data is stored in RAM.

3.4 Computational and additional resources

The computational resources determined how complex and computationally intense we could implement our models and to what depths the models could be taken to. At our disposal were two computers, where the first one had the relevant specifications

- RTX 4090,
- 128 GB RAM,
- 4 TB NVME SSD,
- AMD Ryzen 7 7700X,

and the second one has the specifications

- RTX 3090,
- 128 GB RAM,
- 2 TB NVME SSD,
- Intel Core i5-11600K.

These computers were used to run smaller training sessions and experiment with different setups of hyperparameters and modifications of the models. However, due to heavy resource demands, the models struggled for larger datasets or more parameters that were included in the models, which were caused by the choices of hyperparameters. Therefore, whenever a foundation of possible hyperparameters and setups was found, they were tested further and with higher complexity and more parameters in a software environment AiQu, Aixia Sweden. This software environment was able to include all the molecules in the models as well as perform training sessions at a much faster rate due to the higher specifications of resources compared to the two main computers. The software environment had the following specifications

- 6 A100-SXM4-40GB
- 1 TB of RAM
- 2 AMD EPYC 7742 64-Core Processors

which was accessed remotely. It also had multiple GPUs, which resulted in the possibility of running multiple training sessions at once. However, these sessions could extend the duration of completion for multiple runs due to restrictions of other resources, such as CPU power. One example is Chemprop-IR, which is a very CPU-intensive model. Despite the high specifications, multiple runs would take longer since all the resources were almost always used up for a single run.

4

Results

In this section, we present the predicted spectra produced by AttentiveFP and Chemprop-IR for the training procedures on the dataset consisting of over 10.5 million molecules. The split of the dataset has been identical for both of the models and structured in the following way, where 94% of the data was used for training, 1% for testing, and 4% for validation. Additionally, we highlight aspects such as the performance, accuracy, and spectral loss for both of the models. Visual examples of predictions of spectra are also provided, where a selection of randomly selected spectra are further analyzed to present a further insight to the comparisons between the models. This comparison talks about what areas each of the models excel or fall behind in.

4.1 Results of AttentiveFP

The first result demonstrates how different modes of attention performed relative to each other, a mode refers to whether the attention mechanism is based on GAT, GATv2, or DenseGAT (GAT without masked attention) discussed in section 2.3.1. Multiple runs of 10 epochs (0 to 9) were performed for each attention mode, DenseGAT was configured with 2 layers and GAT and GATv2 with 6 layers, these configurations were chosen since DenseGAT can consider all atoms in the molecule when computing attention scores, while GAT and GATv2 need multiple layers to pass information between distant atoms. Thus making a more fair comparison. The learning rate was set to 0.001 in most cases which was a known stable learning rate from initial testing. In the few exceptions, a lower learning rate of 0.0001 was tested in the case of DenseGAT.

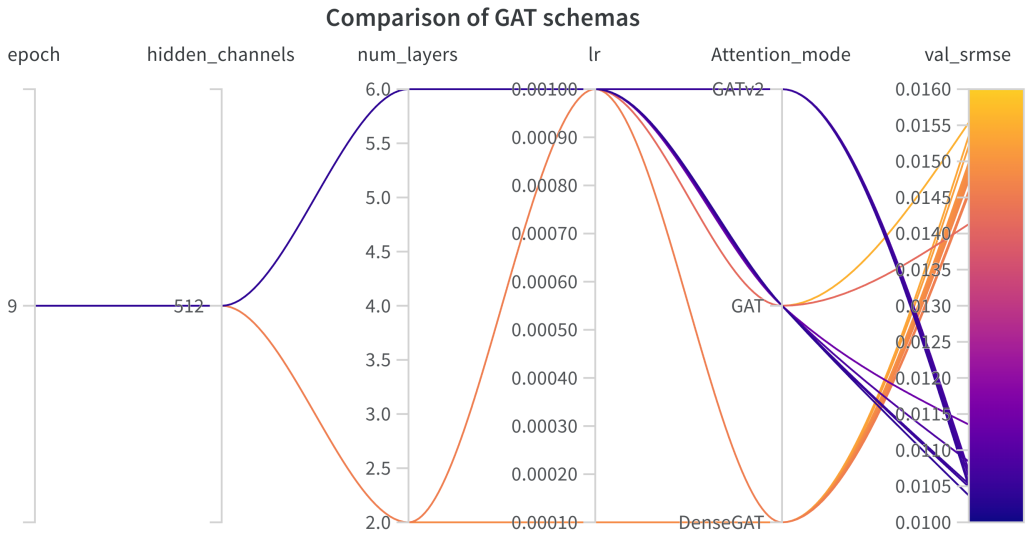


Figure 4.1: Parallel Coordinates plot to track the performance between different attention modes. Based on the SRMSE validation score to the right, GATv2 was the top-performing mode based on the consistently low validation score. GAT had some outliers and even though it performed better than DenseGAT in most cases, GAT received the lowest performance in one case.

The epochs were set to 10 because it was considered sufficient to reach convergence. This is demonstrated in Figure 4.2, where the validation score is plotted against the total training time. In total, it amounted to 257 hours distributed between 23 runs.

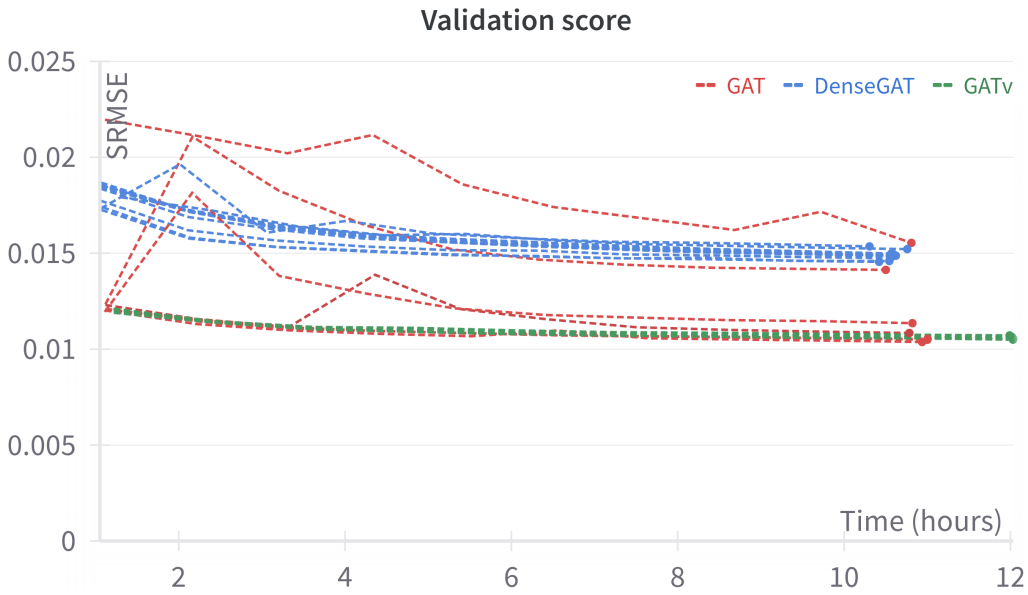


Figure 4.2: Validation score SRMSE during end-to-end training of the different attention modes. Grouped with color we have GAT in red, DenseGAT in blue, and GATv2 in green. Each epoch in GATv2 took about 9% longer than GAT and DenseGAT.

As GATv2 was deemed to be the top-performing attention mode in this prediction task, it received the most attention and computational resources. To get a sense of

the characteristics of the model, a sweep of multiple runs with a mixture of model parameters was performed using the sweep functionality provided by W&B.

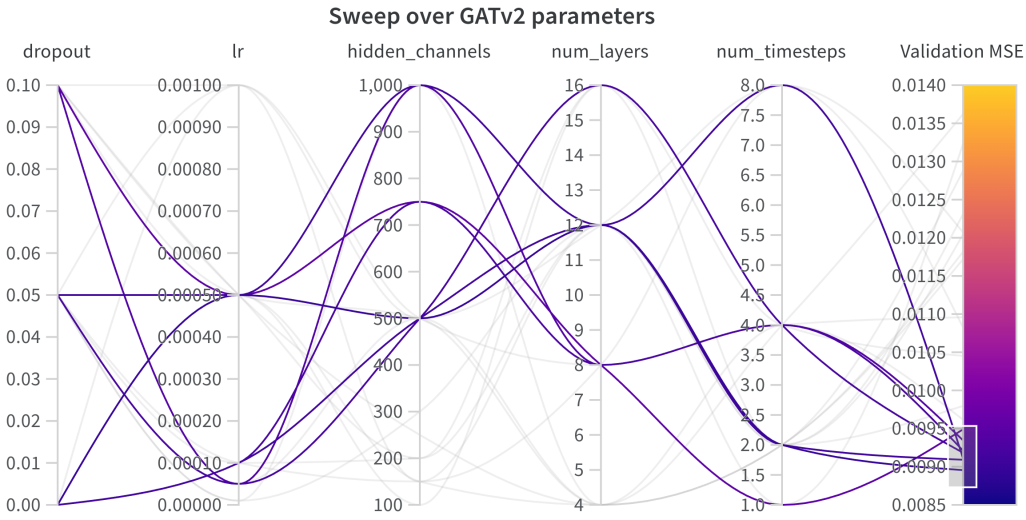


Figure 4.3: Sweep over a mixture of AttentiveFP models with GATv2 attention. Top-performing hyperparameters are highlighted according to the color bar on the right, while less performant models are grayed out.

The sweep in Figure 4.3 indicates that *num_timesteps* has low importance since no significant difference is observed between a low and high number of time steps. The parameters *hidden_channel* and *num_layers* seem to have a threshold where the performance gets a significant increase and once the threshold is reached, no significant improvement is observed by increasing the model size. Notice that the number of attention heads is not included among the parameters considered in the sweep, the number of attention heads was set to one throughout the sweep. Based on the results from models using graph attention mechanisms with multiple attention heads, they seem to consistently improve the prediction performance [20]. Thus assuming this is the case in this implementation as well, excluding the number of attention heads as a variable in the sweep reduces the search space significantly.

To explore the potential of AttentiveFP, the model configuration in Table 4.1 was chosen to perform training with a large number of epochs, this is also the best AttentiveFP model found in this project. The model in Table 4.1 is based on the MoGAT design discussed in section 2.3.1 with a last MLP network before predicting the spectra.

Table 4.1: Model configurations of the best performing AttentiveFP model. It uses the MoGAT super-node structure where all full-molecule embeddings from each super-node get concatenated and passed into an MLP before prediction.

Model architecture	MoGAT + MLP
Attention mode	GATv2
Hidden channels	300
Number of layers	8
Attention heads	3
Number of time steps	5
Dropout	0.025
Initial learning rate	0.0005
Loss function	RMSE
Number of epochs	63
Training duration	96 h 40 min
Final SRMSE	0.009503

Randomly selected molecules with predictions made by the model in Table 4.1 are displayed in Figure 4.5 and 4.4

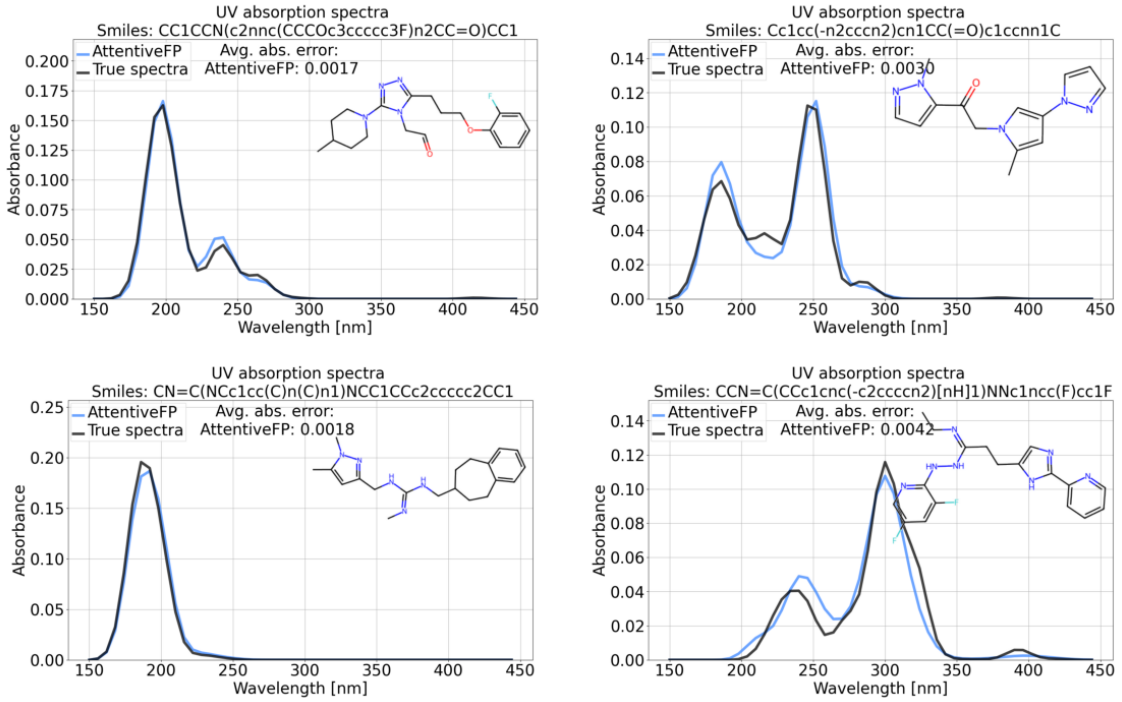


Figure 4.4: Prediction and true spectra of four randomly selected molecules.

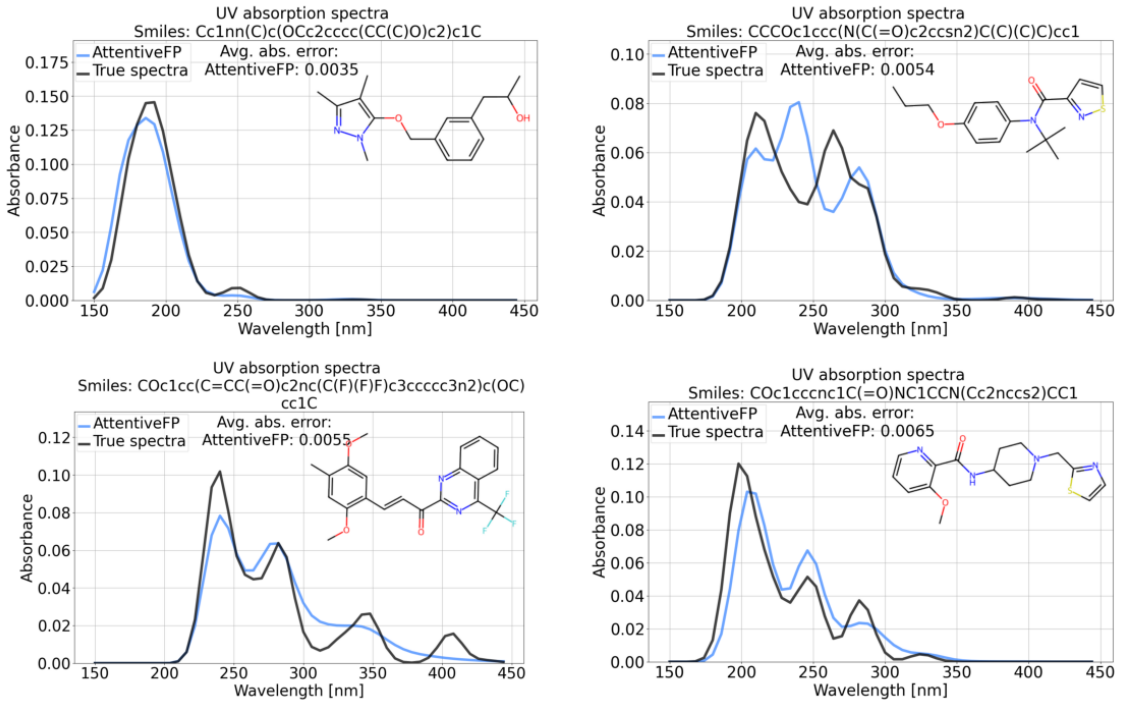


Figure 4.5: Prediction and true spectra of four randomly selected molecules.

A trend noticed when observing predictions as in Figure 4.5 and 4.4 is that predictions are less accurate when there are more than two prominent peaks in the true spectra. A second observation is that the model is most often able to predict the spectra at wavelengths with absorbance close to zero correctly.

4.2 Results of Chemprop-IR

Chemprop-IR was implemented in two different ways. The first implementation followed a path of loading all the data onto the RAM for faster data access, which was what the original authors of the model did. This was done by keeping the CSV implementation. Since it relied heavily on large RAM resources, it was only able to train on 3 million molecules with the following results, for 100 epochs and a batch size of 64.

Table 4.2: Loss values of validation SRMSE and training SID as well as the training duration for 100 epochs.

FFN hidden size and hidden size	SRMSE	SID	Training duration
2200	0.01667	0.543	219 h 58 min

The change in the validation loss metric SRMSE is illustrated in the figure below.

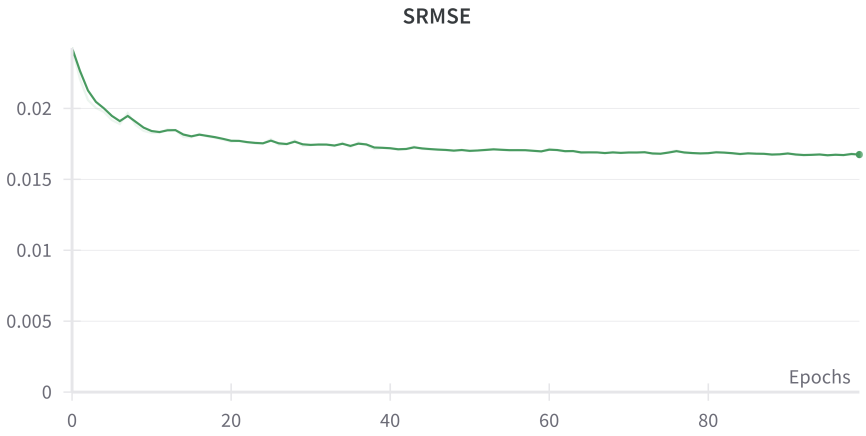


Figure 4.6: Validation SRMSE of the training on 3 million molecules for the first version of Chemprop-IR.

Something to note from figure 4.6 is that around epoch 50, the SRMSE starts to converge to 0.016. Increasing the number of epochs any more leads to an improvement of a 0.77% lower SRMSE than for epoch 50, which appears at epoch 96. However, such a small improvement required 100 additional hours of training, which was deemed to not be a reason enough to keep on going for more epochs. Another remark is that by only using around a third of the available molecules, the predictions of this version of the trained model are illustrated in the following.

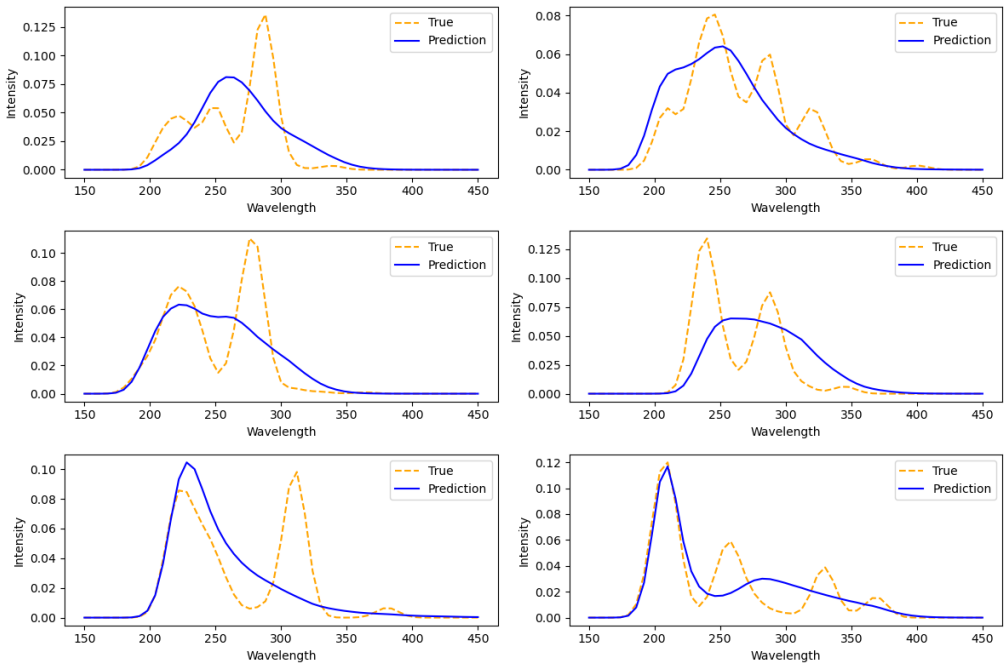


Figure 4.7: Predictions produced of the first version of Chemprop-IR for an FFN hidden size and a hidden size of 2200. The yellow curves represent the true spectra and the blue curves represent the predicted spectra.

Figure 4.7 shows that the first version of the model struggles to find multiple peaks for spectra. The model handles such situations is either by only approximating one peak, usually the one with the highest intensity, and missing the remaining ones, or by combining peaks close to each other into one peak positioned between the originating peaks. An explanation behind this behavior is the exclusion of over 70% of all the available data, thus restricting the model from reaching its full potential. The second version of Chemprop-IR uses instead an HDF5 implementation, as discussed earlier, which enables the training of the entire dataset. The following results have been produced for 25 epochs, a batch size of 64, and the combination of 2200, 2800, and 3400 as the hidden size and FFN hidden size. As a result, the following tables contain the validation loss, training loss, and training duration of the model for all 10.5 million molecules which was achieved.

Table 4.3: Validation SRMSE and training SID for different setups of hidden size and FFN hidden size, as well as the training duration for 25 epochs.

FFN hidden size and hidden size	SRMSE	SID	Training duration
2200	0.01325	0.5409	123 h 26 min
2800	0.01291	0.5766	122 h 53 min
3400	0.01325	0.5722	123 h 26 min

The progress of achieving these results for each epoch is illustrated in figures 4.8 and 4.9, which show the change in the SRMSEs and SIDs for the hidden sizes and FFN hidden sizes of 2200, 2800, and 3400 for 25 epochs. Due to the increased size of data, which had a wider range in diversity of molecules, convergence occurred quicker for the second version of Chemprop-IR compared to the first version. Therefore, 25 epochs were assumed to be suitable as the maximum number of epochs.

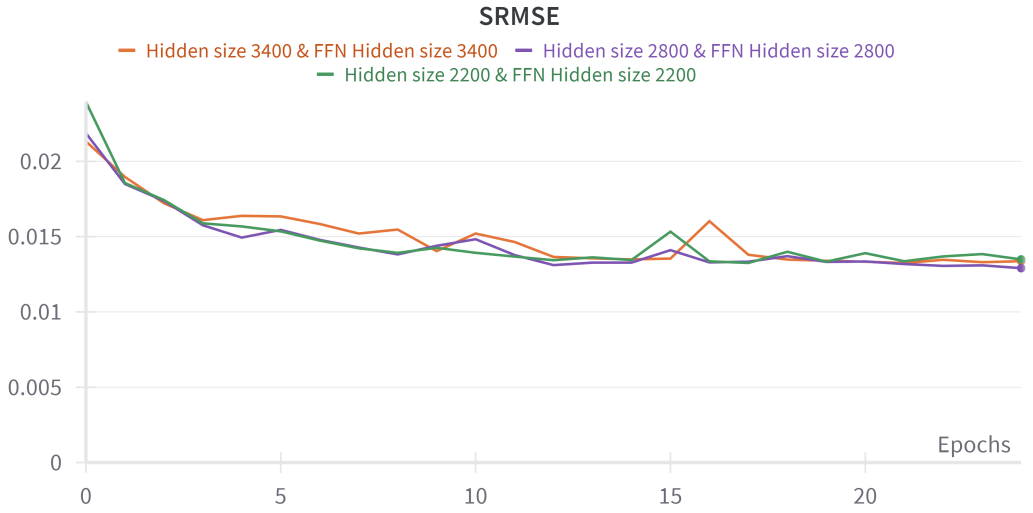


Figure 4.8: Validation SRMSE of the training on 10.5 million molecules for the second version of Chemprop-IR. The colors correspond to the hidden size and FFN hidden size, where green is for the size 2200, purple is for size 2800, and orange is for 3400.

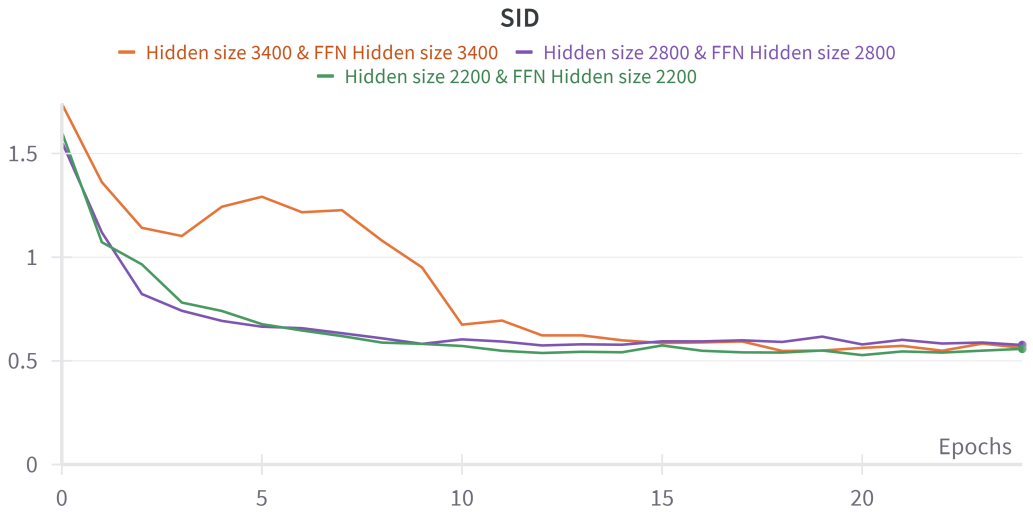


Figure 4.9: Training SID of the training on 10.5 million molecules for the second version of Chemprop-IR. The colors correspond to the hidden size and FFN hidden size, where green is for the size 2200, purple is for size 2800, and orange is for 3400.

Figure 4.8 tells us that around epoch 14, the SRMSE starts to converge to 0.013. Increasing the number of epochs does not cause any major improvements, since the SRMSE starts to fluctuate around 0.013, but not to the point where it reaches a new minimum that is noticeably lower. By including all 10.5 million molecules, we can instantly see faster convergence at a lower value of SRMSE. By using all the available molecules for the second version of Chemprop-IR, the predictions of this version of the trained model are illustrated as follows.

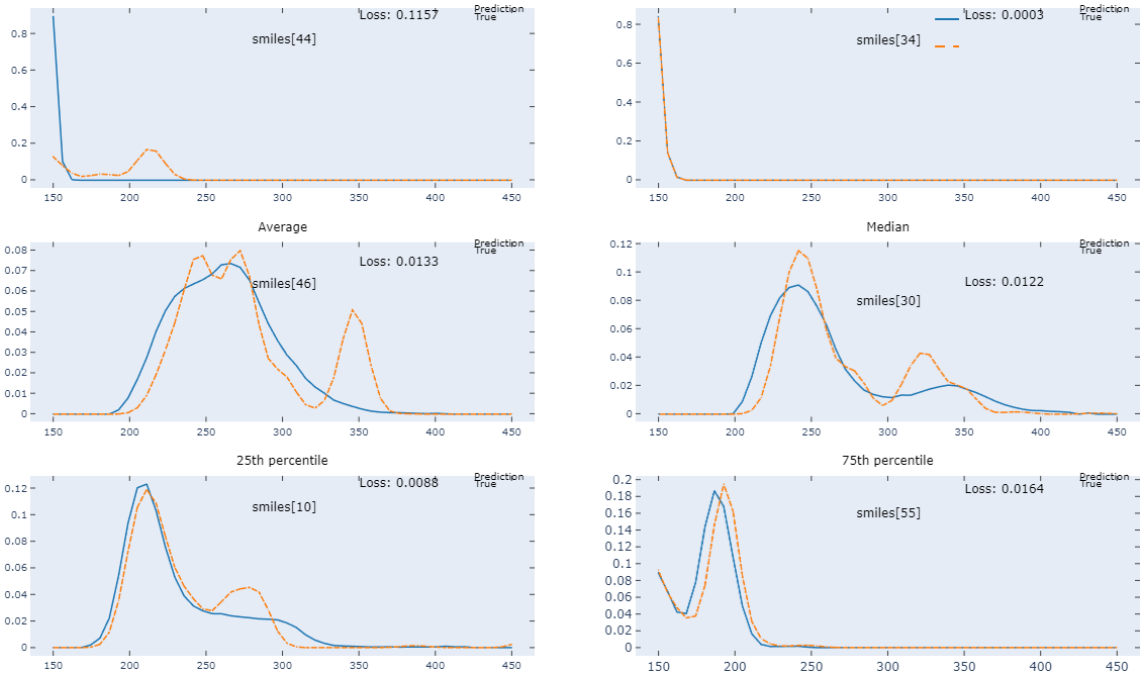


Figure 4.10: Predictions produced of the second version of Chemprop-IR for FFN hidden size 2200 and hidden size 2200. The orange curves represent the true spectra and the blue curves represent the predicted spectra.

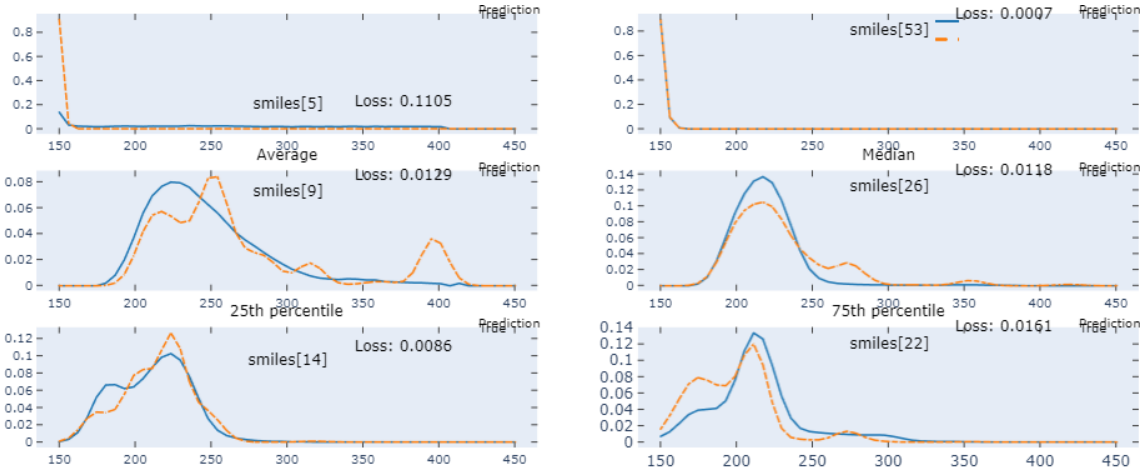


Figure 4.11: Predictions produced of the second version of Chemprop-IR for FFN hidden size 2800 and hidden size 2800. The orange curves represent the true spectra and the blue curves represent the predicted spectra.

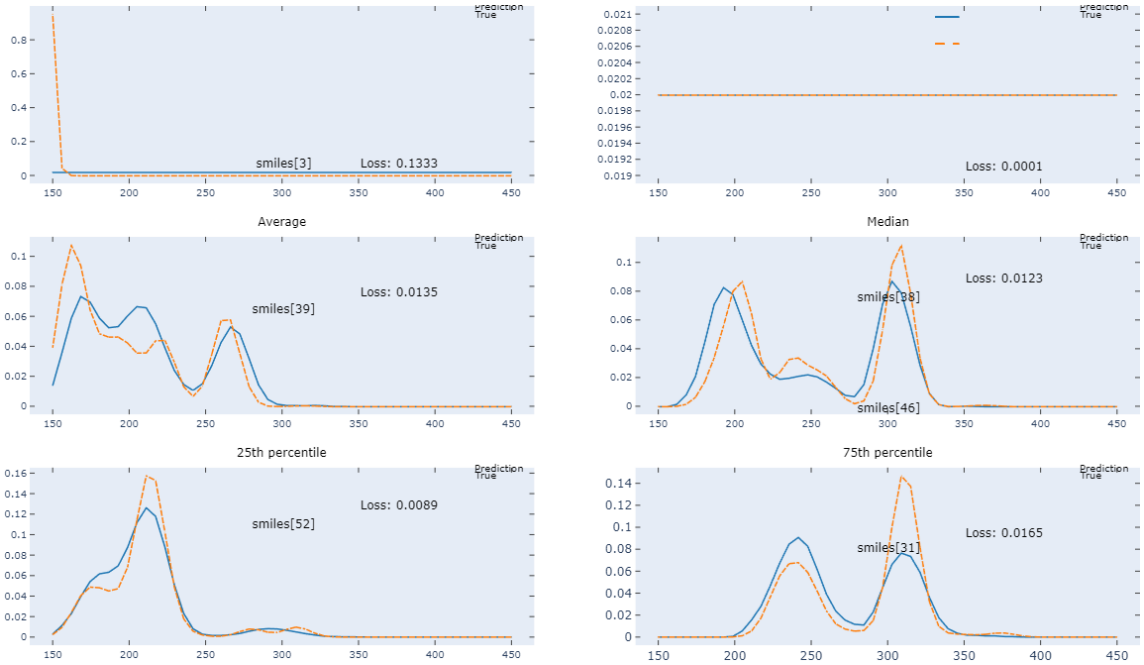


Figure 4.12: Predictions produced of the second version of Chemprop-IR for FFN hidden size 3400 and hidden size 3400. The orange curves represent the true spectra and the blue curves represent the predicted spectra.

Based on the SRMSE figures, we can see that the most optimal epochs are 17 for an FFN hidden size and hidden size of 2200, 24 for the size 2800, and 21 for the size 3400, assuming that the first epoch is denoted as 0. After these epochs, there are no signs of overfitting or major improvements, since the SRMSE and SID appear stable and neither starts to increase nor decrease drastically. The training time was similar for all three setups, indicating that a more complex model with more parameters is not an issue due to the available resources. From the prediction figures, we can

see that a larger FFN hidden size and a hidden size create a model that has a higher accuracy of predicting UV-Vis absorption spectra that consist of multiple peaks. This is visible due to the model more commonly detecting the presence of multiple peaks and therefore predicting these with a higher accuracy of wavelength and intensity, compared to the models with lower FFN hidden sizes and hidden sizes. Despite this, not all the peaks were detected for the models with larger FFN hidden sizes and hidden sizes. However, the model with the lowest achieved SRMSE is the one out of most interest in this case, which is Chemprop-IR with an FFN hidden size and a hidden size of 2800, with an SRMSE of 0.1291.

4.3 Comparisons between the two different models and performance of selected molecules

By comparing the results of the two different models, it becomes clearer how they each perform by training, testing, and validating on the same dataset. The most basic comparison of performance is by evaluating the models' setups that yielded the lowest SRMSE. The lowest SRMSE that AttentiveFP achieved is 0.009503 for the mode GATv2, while Chemprop-IR got the lowest SRMSE of 0.01291 for the second version, which had a setup of 2800 as FFN hidden size and hidden size. By only comparing these two values to each other limits any deeper conclusions or comparisons to be made. Therefore, further analysis has to be made.

An example of such a way that takes more considerations into aspect and compares the accuracy of the two models is by randomly selecting 104,926 molecules that range in diversity, and compare the actual spectra to the predicted spectra of these. The predicted spectra produced by both of the models are plotted in two different colors, one for each model on the y-axis, accompanied by the actual values of intensities for the spectra on the x-axis. A way of judging the most accurate model is by seeing how well the data points fit the line $y = x$. This comparison is shown in the figures below for the wavelengths 180 nm, 312 nm, and 420 nm.

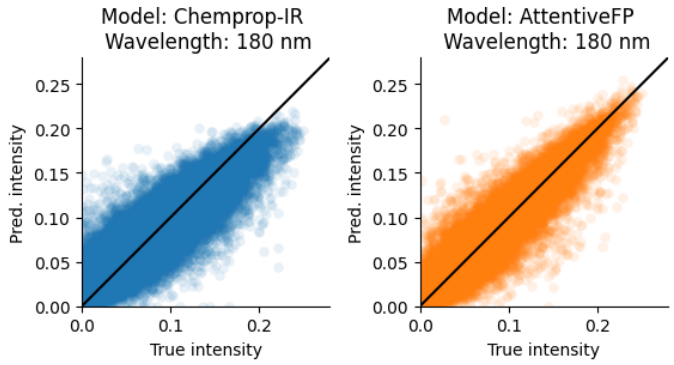


Figure 4.13: A comparison between the predicted spectra of 104,926 molecules on the y-axis for the wavelength 180 nm, where the orange points are produced by AttentiveFP and the blue points are produced by Chemprop-IR. The x-axis represents the actual values of the spectra. The black line represents $y = x$.

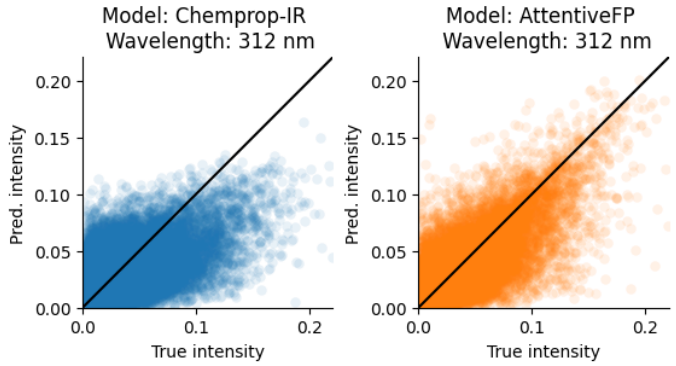


Figure 4.14: A comparison between the predicted spectra of 104,926 molecules on the y-axis for the wavelength 312 nm, where the orange points are produced by AttentiveFP and the blue points are produced by Chemprop-IR. The x-axis represents the actual values of the spectra. The black line represents $y = x$.

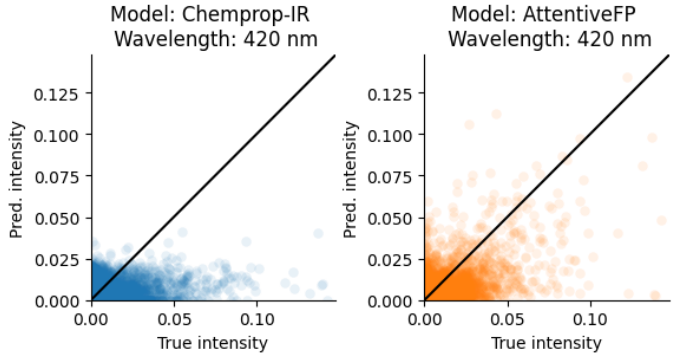


Figure 4.15: A comparison between the predicted spectra of 104,926 molecules on the y-axis for the wavelength 420 nm, where the orange points are produced by AttentiveFP and the blue points are produced by Chemprop-IR. The x-axis represents the actual values of the spectra. The black line represents $y = x$.

The figures 4.13, 4.14, and 4.15 show that for the selected wavelengths, AttentiveFP produced predictions that fit the line $y = x$ closer than Chemprop-IR. The predictions are also closer to the line $y = x$ while being more evenly spaced out on both sides of the line. Additional plots for more wavelengths of accuracy comparisons between the two different models, that back up this statement, can be found in Appendix A.4.

Since including figures for all wavelengths is impossible, a density plot of the selected 104,926 molecules for both of the models can be created, which yields the following figure.

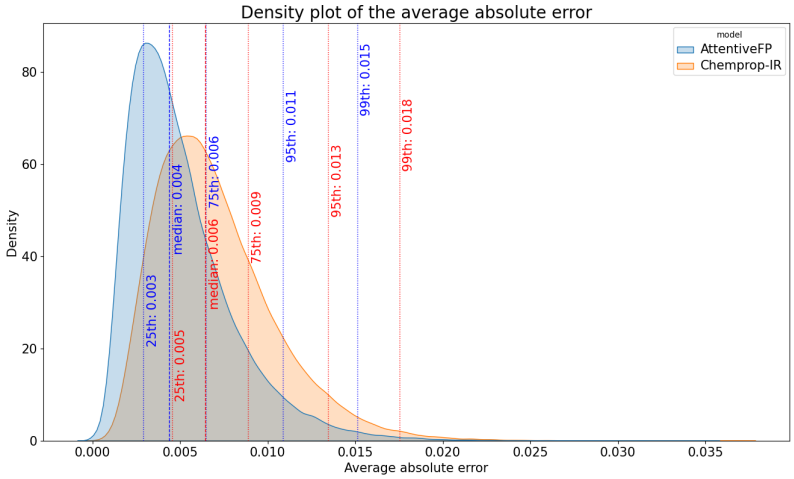


Figure 4.16: Density plot comparing the average absolute error of the models when predicting the spectra of the selected 104,926 molecules. The blue curve corresponds to AttentiveFP and the orange curve corresponds to Chemprop-IR.

The density plot shows that AttentiveFP has lower average absolute errors that occur more frequently compared to Chemprop-IR. This means that the model is more likely to better fit the actual spectra of the randomly selected 104,926 molecules. One way to further prove this point is to randomly select predictions of both models to visually confirm these results, which is shown below.

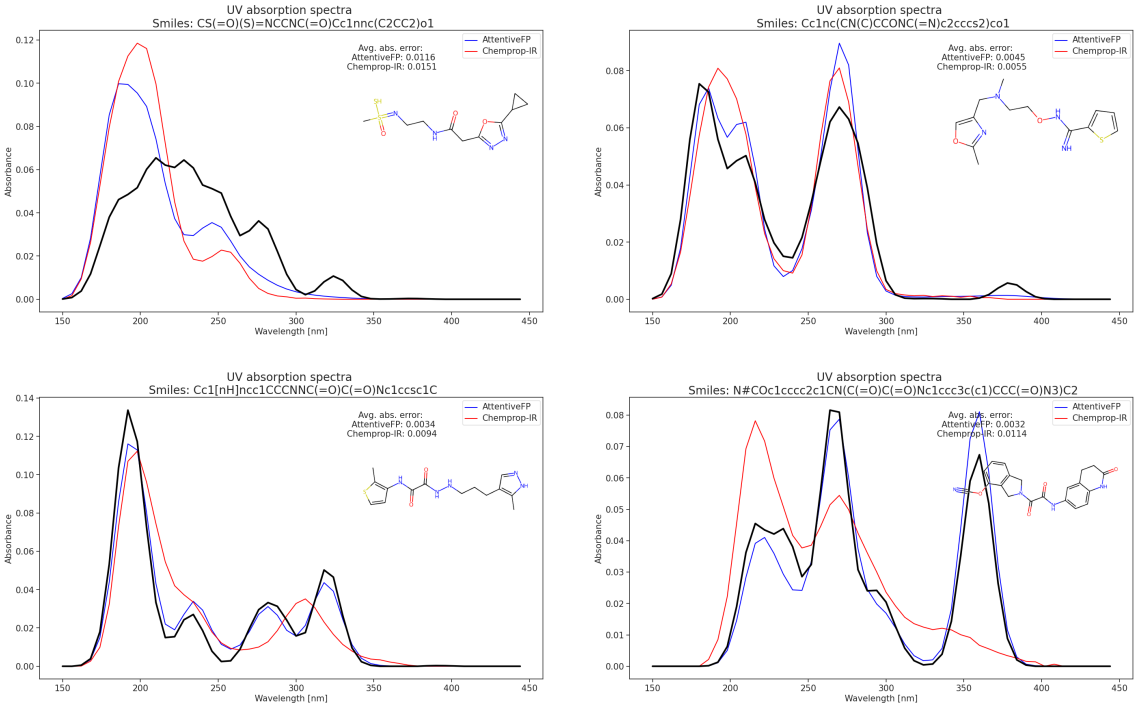


Figure 4.17: Comparison between the actual spectra and predicted spectra of both of the models for randomly selected molecules. The black spectra are the actual spectra, the blue spectra are the predicted spectra produced by AttentiveFP, and the red spectra are the predicted spectra produced by Chemprop-IR.

This further strengthens the theory that AttentiveFP detects peaks in their corresponding positions more frequently and accurately compared to Chemprop-IR. Further examples of such visual comparisons can be found in Appendix A.4.

5
Conclusion

The conclusion part of the project reflects further on the comparison between each models' performance, such as which areas each model performs the best, and other performance-related aspects. We also reflect on the difficulties and restrictions which the models revealed along the way of implementing them, as well as potential sources of error and future considerations in improving this experiment or things to consider.

5.1 Discussion of the different models

The results of the models are further interpreted, such that conclusions are drawn upon these in this section. We start this by comparing the two models to each other and narrow it down to deeper conclusions for each model separately.

5.1.1 Analysis of comparisons between the different models

The results of the comparisons between the different models have shown that the models are viable candidates to predict the UV-Vis absorption spectra of molecules. This is thanks to the low value of SRMSE achieved and the fact that it worked well in the UV-Vis range to detect the behaviors of the spectra with no clear signs of overfitting. However, there are some important considerations to take into account when using and comparing these models.

Firstly, based on the figures 4.13, 4.14, 4.15, and 4.17, AttentiveFP seems to be the model with signs of better fit compared to Chemprop-IR. The easiest sign of the better fit is the way AttentiveFP closely fits the line $y = x$ in the figures mentioned. Another sign was there being predictions that were more evenly spread out on both sides of the line $y = x$ while remaining closer to the line than what Chemprop-IR was. This is partly because of the SRMSE being lower for AttentiveFP, compared to Chemprop-IR. An implication of this lower value is that the model favors the finalized spectra resembling the actual spectra in shape and outline, such as capturing the number of peaks, with the potential tradeoff of amplitudes for the intensities that might be slightly more incorrect. The benefit of doing so is that the process of identifying molecules becomes less complex due to the structure of the spectra resembling the actual spectra, with the difference in amplitudes, while the opposite would mean that there would be missing peaks. The missing peaks could be detrimental since they are characteristic for each molecule, where two different molecules can have similar spectra but be distinguished by one or more additional peaks. The

conclusion of a better fit is carried over to the analysis of randomly selecting 104,926 molecules, where the same properties and theories were visible.

Something that stood out for both of the models was the reduced accuracy in predicting the intensities for larger wavelengths. This became somewhat apparent in the 300 nm range, while at the end of the spectra around 430 to 450 nm, the accuracy was at its lowest. The common pattern among the models for the end of the range was that they underestimated the intensities. This behavior was especially true for Chemprop-IR, which assumed that the majority of the intensities being 0 for the range 430 to 450 nm. An explanation to this behavior is that most molecules of the entire dataset have intensities of 0 at the end of the range. This affected both of the models, but Chemprop-IR the most, where the predictions at this range most often ended up lower than the actual spectra. The sample of 104,926 molecules on the other hand did not seem to contain the same proportion of molecules containing intensities that end around 0 as for the entire dataset.

Another example of a difference between the models was the number of peaks detected. In the example figure 4.17, we can see that AttentiveFP has a tendency to detect multiple peaks, when there are multiple peaks, more accurately and frequently than Chemprop-IR. We also notice that more present peaks appear in the predictions of AttentiveFP than for Chemprop-IR, whereas Chemprop-IR has an easier time predicting single peaks, such as for the first prediction in 4.17. One idea behind this behavior is that Chemprop-IR might need data of a larger size that range much more in diversity, whereas AttentiveFP is more satisfied with the current dataset. Another idea is that the setup of Chemprop-IR was not complex enough, due to the better predictions being for single peak spectra. In order to strengthen this reason, the number of layers, FFN hidden size, and hidden size could be increased. This in turn would capture more complex behaviors and long range relationships between molecules. However, a tradeoff of doing so is an increased training time depending on the available resources as well as the risk of overfitting.

5.1.2 Discussion of AttentiveFP

From the results in Figure 4.17 it is demonstrated that AttentiveFP can predict the UV absorption spectra quite well for some molecules in the test set, and based on Figure 4.16 AttentiveFP outperforms Chemprop-IR significantly. However, there are instances where the AttentiveFP fails by making a completely wrong prediction. This is demonstrated by the prediction in the top-left in Figure 4.17 where the intensities of the predicted spectra have a large error and the relative shape of the prediction is also completely wrong. Since this molecule is not particularly big, the model's ability to propagate messages between atom embeddings in the GATv2 network is probably not the cause of the bad fit. As a first guess, the reason for the low performance is probably caused by structures in the molecule that do not occur as much in the training set. As a comparison, we have the molecule in Figure 5.1 which is significantly larger and AttentiveFP can still capture the positions of the peaks.

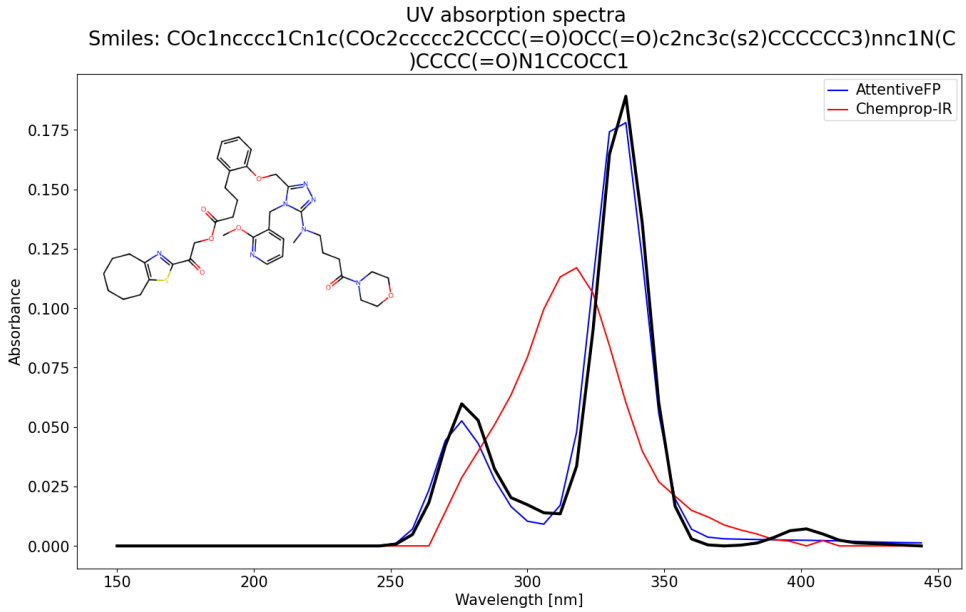


Figure 5.1: Spectra of a large molecule with true spectra in black, prediction from AttentiveFP in blue and Chemprop-IR in red.

As mentioned, the significantly improved performance in Figure 5.1 could be attributed to the molecule's composition, which primarily consists of frequently occurring components such as carbon atoms, single and double bonds, and small ring structures. This suggests that the challenge may not lie in the ability to propagate messages in the graph, but rather in constructing the necessary embeddings to propagate the "correct" message. One potential solution could be to devise a method to balance the training set by reducing the number of molecules with similar structures. This might result in embeddings less likely to favor common structures when propagating messages in the graph. However, the actual outcome of such a method needs to be tested, and it is unlikely to be a complete solution. As demonstrated by the spectra in Figure 4.5 and 4.4, AttentiveFP can often predict single and double peaks correctly, though performance degrades as the number of peaks and "complexity" of the spectra increase. This could suggest that the embeddings may not have sufficient dimensions to encode all the necessary information. However, it is difficult to conclusively argue that this is the sole reason for the degrading performance with an increasing number of peaks. It could also be due to an imbalanced dataset, as most molecules have spectra with one or two peaks in the lower half of the spectral interval considered in this project.

Given that alternating single and double bonds (a conjugated system) significantly influence the characteristics of a molecule's UV-absorption spectra, there might be an opportunity to enhance performance by distinguishing the message passing between atoms and bonds, as implemented in the ComABAN model[40]. Briefly, ComABAN conducts attention-based message passing similar to AttentiveFP. However, unlike AttentiveFP, which solely constructs its embeddings using the attention-based mechanism between atoms, ComABAN expands on this concept and also forms a separate embedding with the attention-based mechanism between bonds.

Additionally, it employs the concept of super-nodes and *super-edges* to construct the final atom and bond embeddings with attentive layers. These two embedding vectors are subsequently combined with a final attentive layer, resulting in a comprehensive full-molecule state vector that can be passed to a final linear layer or MLP for prediction tasks. With these enhancements, ComABAN was able to surpass AttentiveFP in several prediction benchmarks [40]. Given the promising results we have demonstrated with AttentiveFP, this represents a logical next step to finding a more effective model.

A third opportunity for improvements would be to use an ensemble of models, where each model is trained using a unique weight initialization. This is analogous to sampling a distribution with replacement multiple times to improve the estimate of the expected value of the population mean. However, this method comes with the obvious downside of increased energy expenditure.

5.1.3 Discussion of Chemprop-IR

Based on the results of Chemprop-IR, several conclusions can be made of the model. Firstly, the differences in results between the two versions of the model show that convergence of SRMSE and SID emerges quicker the more molecules that are included in the dataset. For instance, half as many epochs are required to converge when having 10.5 million molecules as compared to 3 million molecules. Secondly, the model is able to converge to a lower SRMSE and SID, which is due to the inclusion of more diverse molecules which the model is able to utilize for training. Generally, more diverse molecules result in a more robust finished model that is able to more accurately predict molecules of different types. The conclusion that more molecules result in a better model is true for Chemprop-IR.

One topic that became active was the limitation of choices of spectral loss functions and metrics. By attempting several training runs of different combinations, only SID was the possible spectral loss function, while SRMSE and SMSE only worked as the possible metrics. This could be determined due to the increasing training and validation loss for each epoch that passed for the setups of other hyperparameters. During the inclusion of an exponential output activation function, the choices of hyperparameters that also fulfilled our task of predicting spectra became increasingly more limited. One reason behind this is that several choices of hyperparameters enabled negative outputs, which was not allowed in our case and was therefore displayed by an increasing SRMSE. Another reason was that some setups caused unstable training, where the SRMSE kept increasing at a much faster rate for each epoch.

Another interesting discovery was the model's adaptability in predicting UV-Vis absorption spectra. The original intention of the model was to predict IR-spectra. Chemprop-IR has shown promises in predicting UV-spectra, especially for single-peak spectra. However, the model seemed to struggle somewhat for molecules with multiple peaks. In spectra with multiple peaks of different intensities, the peak

with the highest intensity was almost always predicted, while the smaller ones were sometimes predicted. In some cases, the smaller peaks were overlapped by a slope, or neglected. However, if there were peaks close to each other, for instance, 20 nm apart, they often end up as one wider peak in the predicted spectra. One figure that illustrates this is the following.

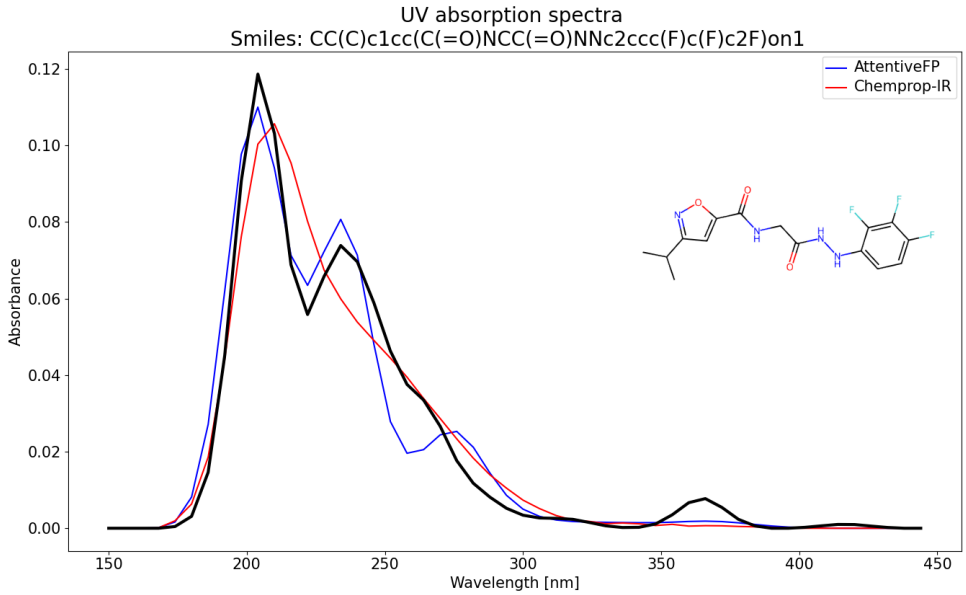


Figure 5.2: An example of a predicted spectrum produced by Chemprop-IR where two peaks are combined into one peak and the remaining peaks are averaged into a slope rather than multiple peaks.

An additional pattern that appeared throughout the prediction was that the model was sometimes off with the values of the predicted intensities by a margin, by either being too low or too high in some cases. A reason for this behavior is that the maximum peak for each spectrum can vary greatly between 0 and 1 for all the molecules, due to the sum of the spectra always being 1, Therefore, the model will detect this pattern more clearly the more molecules that are contributed to it. One way to minimize this pattern from appearing is to divide all intensities in the spectra by their corresponding maximum peaks, rather than the sum of all intensities. As a result, all maximum intensities will be 1 and the shape of the spectra will be preserved due to all molecules having 0 as their minimum intensities, which does not cause the spectra to move vertically.

Finally, the most optimal choice of FFN hidden size and hidden size was analyzed. As mentioned in the results, setting the size to 2800 was deemed to be the most optimal choice due to the ability of such a model to predict multiple peaks and the low value for its SRMSE. However, when this was increased to 3400, the size enabled the model to work with more complex spectra that consisted of two or more peaks. The model did not always detect the correct number of peaks at the exact positions, but it performed better than the remaining sizes, since they had difficulties predicting multiple peaks. When using a higher size, the initial pattern was that the SRMSE

reduced while the SID increased. As the training procedure went on for a number of epochs, these values converges closely to each other, where a larger FFN hidden size and hidden size had a slightly lower SRMSE and a higher SID. Despite enabling such a complex model with many parameter, no signs of overfitting were detected, which means that our data in nature requires a more complex model. Another explanation to the lack of overfitting could be due to the inclusion of a dropout and it being set being 0.1.

One theory of why Chemprop-IR did not perform as well as AttentiveFP is that the model was intended to predict IR absorption spectra for molecules rather than UV-Vis absorptions spectra. The nature of IR spectroscopy is that information about molecular vibrations are provided, where the present functional groups in a molecule are revealed. In turn, this gives a view about the structure of the molecule. On the other hand, UV spectroscopy contains information about the excitation energies of electrons moving between different energy levels. Therefore, Chemprop-IR is not designed to take inputs of excitation energies and could struggle, which in turn produces less accurate predictions.

5.2 Technical, physical, and mathematical challenges

By working with the different models, that were designed for a molecule count of up to 100,000, several limitations were encountered, which were unique for each model. In this section, every encounter of such limitations and possible solutions that were implemented throughout the project is discussed.

5.2.1 AttentiveFP

With the large dataset provided by the Oak Ridge National Laboratory, the first challenge to overcome was finding a solution to store the majority of the data on disk while training the model. As mentioned, this was initially solved with the OnDiskDataset class provided by PyG. However, managing data storage does not solve the issue of the requirement for manual labor when training multiple models. Without a solution for automating the process of initiating the training of new models, the hardware will most likely spend a significant amount of time in an idle state between model initialization, which is undesirable. To mitigate this issue, a solution provided by Weights & Biases (W&B) was used. Their Python package made it possible to have multiple isolated computers contribute to the same sweep, meaning that all connected computers are training models in a unified predefined search space of hyperparameters. Once a training instance is terminated, the use of W&B also makes it possible to automatically initiate the next combination of hyperparameters and contribute to the search performed by the sweep.

Another limiting factor that required some attention is the potentially long time needed for training a model. The time spent on one epoch can easily reach over 4

hours depending on the model size and batch size. This problem could be mitigated when using systems that have multiple GPUs accessible. Using the *Distributed Data Parallel* (DDP) solution in PyTorch enables us to train each epoch in a parallel fashion, up to one parallel thread per GPU in the system. The time required to run one epoch in training effectively scales inversely proportional to the number of parallel threads (number of GPUs). This scaling is, of course, ideal, and the true speedup is case-specific.

5.2.2 Chemprop-IR

The technical limitations of Chemprop-IR revolve more around the first version of the model. Chemprop-IR had some technical limitations when handling more than one hundred thousand molecules. Only 500,000 molecules could be used for the two standard computers due to the model using up all the 128 GB of RAM. Extending the model to the server and including more molecules, the model's limitations became clearer, where 3,000,000 molecules could be included in the training procedure, which almost used up all the 1 TB of RAM that the AiQu server offered. This could be avoided by turning off the cache option, although this resulted in a significant increase in training time. Turning off caching more than doubled the training time for the same number of molecules. The training time seemed to increase linearly, which theoretically means that the duration of an epoch for all 10.5 million molecules would take over is 24 hours, which was not a possibility within the limited timeframe of this project, since a single run for testing a choice of FFN hidden size and hidden size would take close to a month.

By including the HDF5 structure in the second version of Chemprop-IR, the constraint of working around the RAM issue was resolved. In turn, the training time per epoch was reduced from 24 hours to 4.5 hours. This was thanks to the more efficient data accessing, storage, memory management, and compression techniques that HDF5 offered. Therefore, the possibility of running multiple runs on different GPUs and from different storage devices was explored. This turned out to be another limitation of the model, where it appeared to rely on all the CPU resources to be available. It turns out that Chemprop-IRs CPU usage was extremely high, often utilizing all available cores at a constant rate of 100% for a single run. Therefore, when initiating multiple runs, the training time increased greater than linearly for each run, where a single run takes 4.5 hours per epoch normally would cause two runs running in parallel to have a duration 10 to 11 hours per epoch to complete. Therefore, having multiple runs was deemed as the less optimal route as compared to initiating new runs after the old ones had finished.

5.3 Outlook

Throughout this project, there appeared some potential sources of error. In the early stages of handling the data, we saw that there were some invalid SMILES according to RDKit. This raised the question of there being any more invalid SMILES or SMILES that represented other molecular structures than intended, which RDKit

did not detect. One example of an unconventional appearance of SMILES which appeared was the inclusion of ".". This dot stands for non-bonded atoms, where for instance C and O are not bonded to each other in C1.O2. To avoid this data affecting the remaining valid data, they were immediately sorted out.

For future recreations or improvements of this project, several paths are available to resolve or minimize the risk of facing similar issues that have appeared throughout the project. Firstly, the data-related issues can be minimized by modifying Chemprop-IR and AttentiveFP to utilize DeepSMILES or SELFIES as representations of the molecules, rather than using SMILES. SELFIES (Self-Referencing Embedded Strings) has been shown to resolve the issue of validity when representing a model, which SMILES sometimes faces. Every SELFIES corresponds to a valid molecule and can also represent any valid molecule without notation errors arising. According to the creators of SELFIES, the implementation of it does not require major changes to the models in use, if sometimes any [41]. The inclusion of SELFIES contributes to a wider diversity of molecules, which is a sought-after characteristic since it creates trained models that can predict the UV-Vis spectra of more types of molecules.

Another way of trying a different alternative to the models is by training an ensemble of runs. This means that multiple models are run at the same and the best predictions of each model are selected to reduce the errors and bias for certain types of molecules. An ensemble of models can eliminate the fault of one submodel. However, the drawback of this is the increased risk of overfitting, due to the increased complexity, as well as being more computationally expensive. Training an ensemble of runs also means that it will take longer to train, which in our case was not possible with the time constraint. Finally, we have the option to train all models for different spectral loss functions. This would in turn show which of the spectral loss functions is best suited for predicting spectra by comparing the different loss functions. However, not every model can comply with all spectral loss functions, since some models have a risk of an increasing loss metric for each epoch for certain spectral loss functions. For instance, in the case of Chemprop-IR, only SID and SRMSE were the valid ones that caused the metric loss to decrease for each epoch. Therefore, this has to be carefully tested.

Bibliography

[1] Bansal M. Organic spectroscopy: Principles of organic spectroscopy. 2019.

[2] Bohacek R. S., Guida W., and McMartin C. The art and practice of structure-based drug design: A molecular modeling perspective. *Medicinal Research Reviews*, 16(1):3–50, 1996.

[3] Alina Bachmann and Cosimo De Caro. *Competence Guide - Automated acid dissociation constant determination of organic compounds by coupling of titration and UV/Vis instruments*. 06 2024.

[4] Brehm M., Fligg R., Kirchner B., Thomas M., and Vöhringer P. Computing vibrational spectra from ab initio molecular dynamics. *Phys. Chem. Chem. Phys.*, 15:6608–6622, 2013.

[5] Irle S., Mehta K., Paisini ML., and Yoo P. Two excited-state datasets for quantum chemical uv-vis spectra of organic molecules. *Scientific Data*, 10(1):546, Aug 2023.

[6] Chen K., Jiang H., Li X., Li Z., Liu X., Luo X., Wan X., Wang D., Xiong Z., Zheng M., and Zhong F. Pushing the boundaries of molecular representation for drug discovery with the graph attention mechanism. *J Med Chem*, 63(16):8749–8760, August 2019.

[7] Aboul H., Askr H., Elgeldawi E., Elshaier Y., Gomaa M., and Hassanien A. Deep learning in drug discovery: an integrative review and future challenges. *Artificial Intelligence Review*, 56(7):5975–6037, Jul 2023.

[8] Hong Y., Li S., Tang H., Tichy S., Welch C., and Ye Y. 3DMolMS: prediction of tandem mass spectra from 3D molecular conformations. *Bioinformatics*, 39(6), June 2023.

[9] Ellis J., Iqbal R., Saquer N., and Yoshimatsu K. Infrared spectra prediction using attention-based graph neural networks. *Digital Discovery*, 3:602–609, 2024.

[10] Furselo M., Green H. W., Guan Y., and McGill C. Predicting Infrared Spectra with Message Passing Neural Networks. *Journal of Chemical Information and Modeling*, 2021.

[11] Röst H., Wang V., and Young A. Massformer: Tandem mass spectrum prediction for small molecules using graph transformers, 2023.

[12] Irle S., Mehta K., Paisini ML., and Yoo P. ORNL_AISD-Ex: Quantum chemical prediction of UV/Vis absorption spectra for over 10 million organic molecules. *Oak Ridge National Laboratory*, 2023.

[13] Chen K., Jiang H., Li X., Li Z., Liu X., Luo X., Wan X., Wang D., Xiong Z., Zheng M., and Zhong F. Pushing the Boundaries of Molecular Representation for Drug Discovery with the Graph Attention Mechanism. *Distill*, 2021.

[14] Hu W., Jegelka S., Leskovec J., and Xu K. How powerful are graph neural networks? 02 2019.
[15] Pearce A., Reif E., Sanchez-Lengeling B., and Wiltschko A. A Gentle Introduction to Graph Neural Networks. *Distill*, 2021.
[16] Liu Z. and Zhou J. *Introduction to Graph Neural Networks*. Morgan & Claypool, 2020.
[17] Ganesan T. Backpropagation in Neural Network. *Geeks for geeks*, 05 2024.
[18] Casanova A., Cucurull G., Liò P., Romero A., Veličković P., and Bengio Y. Graph Attention Networks. 2018.
[19] Gomez A., Jones L., Kaiser L., Parmar N., Polosukhin I., Shazeer N., Uszkoreit J., and Vaswani A. Attention Is All You Need. *Journal of Chemical Information and Modeling*, 08 2020.
[20] Alon U, Brody S., and Yahav E. How Attentive are Graph Attention Networks? 2022.
[21] Hornik K., Stinchcombe M., and White H. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.
[22] Dahl E. G., Gilmer J., Riley F. P., Schoenholz S. Samuel, and Vinyals O. Neural Message Passing for Quantum Chemistry. 06 2017.
[23] Swanson K. Message Passing Neural Networks for Molecular Property Prediction. 06 2019.
[24] Gong C. and Maday Y. Directed Message Passing Based on Attention for Prediction of Molecular Properties. 05 2023.
[25] Basu M. and Mishra P. *Absorption Spectroscopy: What Can We Learn About Conformational Changes of Biomolecules?*, pages 1–18. Springer Nature Singapore, Singapore, 2022.
[26] Fey M. and Lenssen J. Fast graph representation learning with pytorch geometric, 5 2019.
[27] Choi C., Han Y., Kang C., Kang J., Kim K., Kim W., Lee S., Park H., and Son Y. Multi-order graph attention network for water solubility prediction and interpretation. *Scientific Reports*, 13(1):957, Mar 2023.
[28] Office of Chemical Safety and Pollution Prevention (7403M). Sustainable Futures / P2 Framework P2 Framework Manual. *United States Environmental Protection Agency*, 2012.
[29] National Center for Biotechnology Information. PubChem Compound Summary for CID 681, Dopamine. *PubChem*, 04 2024.
[30] Béquignon O., Jespers W., Schoenmaker L., and et al. UnCorrupt SMILES: a novel approach to de novo design. *Journal of Chemical Information and Modeling*, 2023.
[31] Irle S., Mehta K., Paisini ML., and Yoo P. Large scale molecular dataset analysis with python. `https://github.com/ORNL/Analysis-of-Large-Scale-Molecular-Datasets-with-Python/tree/main`, 2023.
[32] Altmeyer P. UV rays. *Altmeyer encyclopedia*, 2020.
[33] Biewald L. Experiment tracking with weights and biases, 2020. Software available from wandb.com.
[34] Rdkit: Open-source cheminformatics.

[35] Meta AI. Pytorch, 2024.
[36] Inc Anaconda. Conda documentation, 2024.
[37] Hamilton A. Intro to hdf5/h5py and comparison to csv for speed & compression. 2023.
[38] The HDF Group (THG). Hdf5 for python, 2024.
[39] Hipp R. SQLite, 2020.
[40] Liu Y., Sheng R., Xie Y., Yan H., and Yuan L. ComABAN: refining molecular representation with the graph attention mechanism to accelerate drug discovery. *Briefings in Bioinformatics*, 23(5):bbac350, 08 2022.
[41] Krenn M. et al. Self-referencing embedded strings (selfies): A 100 2020.

A

Appendix 1

A.1 Code availability

The code for AttentiveFP can be found at `https://github.com/williamnyren/AttentiveFP-UV`.

The code for the first version of Chemprop-IR can be found at `https://github.com/itaha01/Chemprop_IR_V1_UV`.

The code for the second version of Chemprop-IR, which implements HDF5, can be found at `https://github.com/williamnyren/Chemprop-UV`.

A.2 Proof of theorems

A.2.1 Proof of theorem 2.2.1

Proof. Assume that after k iterations, the GNN $\mathcal{A}$ satisfies $\mathcal{A}(G_1) \neq \mathcal{A}(G_2)$, for the two graphs G_1 and G_2, while the WL test cannot decide if the two graphs are non-isomorphic. Implicitly, the two graphs G_1 and G_2 have the same WL node labels for iteration i and $i+1$ for $i = 0, ..., k-1$ and the graphs have the same collection, for instance multiset of WL node labels $\left\{l_v^{(i)}\right\}$ and the same collection of node neighborhoods denoted by $\left\{\left(l_v^{(i)}, \left\{l_u^{(i)} : u \in \mathcal{N}(v)\right\}\right)\right\}$. This is true since otherwise, the WL test results in different collections of node labels at iteration $i+1$ for both of the graphs due to the different multisets getting new unique labels.

It is known that the WL test relabels different multisets of neighboring nodes into new different labels. Therefore, it is necessary to show that on the same graph, $G = G_1$ or $G = G_2$m if the WL node labels $l_v^{(i)} = l_u^{(i)}$, then the GNN node features $h_v^{(i)} = h_u^{(i)}$ is true for any iteration i. This idea of this proof is proof by contradiction by using induction.

For $i = 0$, this is true since WL and GNN start with the same iterations. Assume that it is true for iteration j, if for any u, v, that $l_v^{(j+1)} = l_u^{(j+1)}$, then it must be the case that

$$\left\{\left(l_v^{(j)}, \left\{l_w^{(j)} : w \in \mathcal{N}(v)\right\}\right)\right\} = \left\{\left(l_u^{(j)}, \left\{l_w^{(i)} : w \in \mathcal{N}(v)\right\}\right)\right\}. \tag{A.1}$$

From the assumption for iteration j, this results in

$$\left\{\left(h_v^{(j)}, \left\{h_w^{(j)} : w \in \mathcal{N}(v)\right\}\right)\right\} = \left\{\left(h_u^{(j)}, \left\{h_w^{(i)} : w \in \mathcal{N}(v)\right\}\right)\right\}. \tag{A.2}$$

In the aggregation process of the GNN, the same input, for instance, neighborhood features, generates the same output. This means that $h_v^{(j+1)} = h_u^{(j+1)}$ and by induction, if the WL node labels $l_v^{(i)} = l_u^{(i}$, the GNN node features $h_v^{(i)} = h_u^{(i)}$ holds true for any iteration i. A valid mapping ϕ such that $h_v^{(i)} = \phi(l_v^{(i)})$ for any $v \in G$ is created. The fact that the two graphs G_1 and G_2 have the same multiset of WL neighborhood labels causes the graphs to also have the same collection of GNN neighborhood features, which is the same as

$$\left\{\left(h_v^{(i)}, \left\{h_u^{(i)} : u \in \mathcal{N}(v)\right\}\right)\right\} = \left\{\left(\phi(l_v^{(i)}), \left\{\phi(l_u^{(i)}) : u \in \mathcal{N}(v)\right\}\right)\right\}. \tag{A.3}$$

Therefore, the $\{h_v^{i+1}\}$ are identical. The same collection of GNN node features $\left\{h_v^k\right\}$ for the graphs G_1 and G_2 is obtained. The graph level readout function is permutation invariant with respect to the collection features, which means that $\mathcal{A}(G_1) \neq \mathcal{A}(G_2)$, which in turn is a contradiction. □

A.2.2 Proof of theorem 2.2.2

Proof. Assume that the GNN $\mathcal{A}$ holds the condition of the theorem to be true, and G_1 and G_2 to be two graphs for which the WL test tells non-isomorphic at iteration K. It is enough to show that the aggregation process of the neighborhood of $\mathcal{A}$ embeds G_1 and G_2 into different multisets of node features since the graph-level readout function is injective. The readout function maps distant multisets of node features into unique embeddings. Assume that $\mathcal{A}$ updates node representations in the manner of

$$h_v^{(k)} = \phi\left(h_v^{(k-1)}, f\left(\left\{h_u^{(k-1)} : u \in \mathcal{N}(v)\right\}\right)\right), \tag{A.4}$$

where f and ϕ are injective functions. Using the WL test, which applies a predetermined injective has function g to update the WL node labels $l_v^{(k)}$, results in

$$l_v^{(k)} = g\left(l_v^{(k-1)}, \left\{l_u^{(k-1)} : u \in \mathcal{N}(v)\right\}\right). \tag{A.5}$$

Proof by induction is used to show that for any iteration k, there exists an injective function φ such that $h_v^{(k)} = \varphi(l_v^{(k)})$.

For the base case of $k = 0$, this is true since the initial node features are identical for WL and GNN which is $l_v^{(0)} = h_v^{(0)}\ \forall v \in G_1, G_2$. The injective function φ has the possibility of being the identity function for the base case. The next step is to assume that the theorem is true for iteration $k-1$ and thus show that it is true for iteration k. By substituting $h_v^{(k-1)}$ with $\varphi(l_v^{(k-1)})$, we have that

$$h_v^{(k)} = \phi\left(\varphi(l_v^{(k-1)}), f\left(\left\{\varphi(l_u^{(k-1)}) : u \in \mathcal{N}(v)\right\}\right)\right). \tag{A.6}$$

The composition of injective functions is injective, which means that there exists an injective function ψ such that

$$h_v^{(k)} = \psi\left(l_v^{(k-1)}, \left\{l_u^{(k-1)} : u \in \mathcal{N}(v)\right\}\right), \tag{A.7}$$

which can be re-written as

$$h_v^{(k)} = \psi \circ g^{-1} g \left(l_v^{(k-1)}, \left\{ l_u^{(k-1)} : u \in \mathcal{N}(v) \right\} \right) = \psi \circ g^{-1}(l_v^{(k)}), \tag{A.8}$$

where $\varphi = \psi \circ g^{-1}$ is an injective function since the composition of injective functions is also injective. Therefore, for any iteration k, there exists an injective function φ such that $h_v^{(k)} = \varphi(l_v^{(k)})$. For iteration K, the WL test determines that G_1 and G_2 are non-isomorphic, which means that the multisets $\left\{ l_v^{(K)} \right\}$ are different for the two graphs. The node embeddings of the GNN $\mathcal{A}$, which are $\left\{ h_v^{(K)} \right\} = \left\{ \varphi(l_v^{(K)}) \right\}$, are therefore different for the two graphs due to the injectivity property of φ. □

A.3 Additional predicted spectra produced by the models

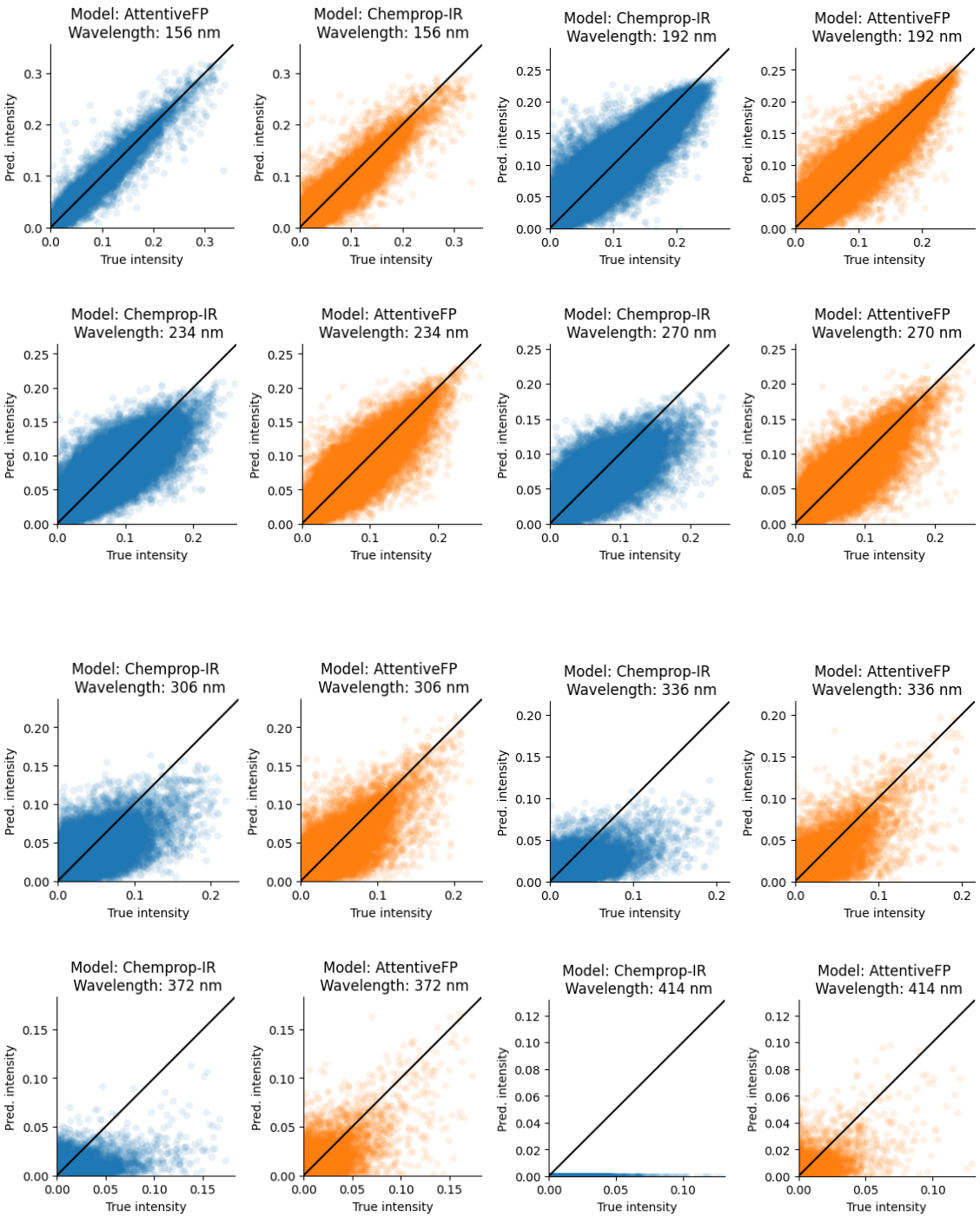
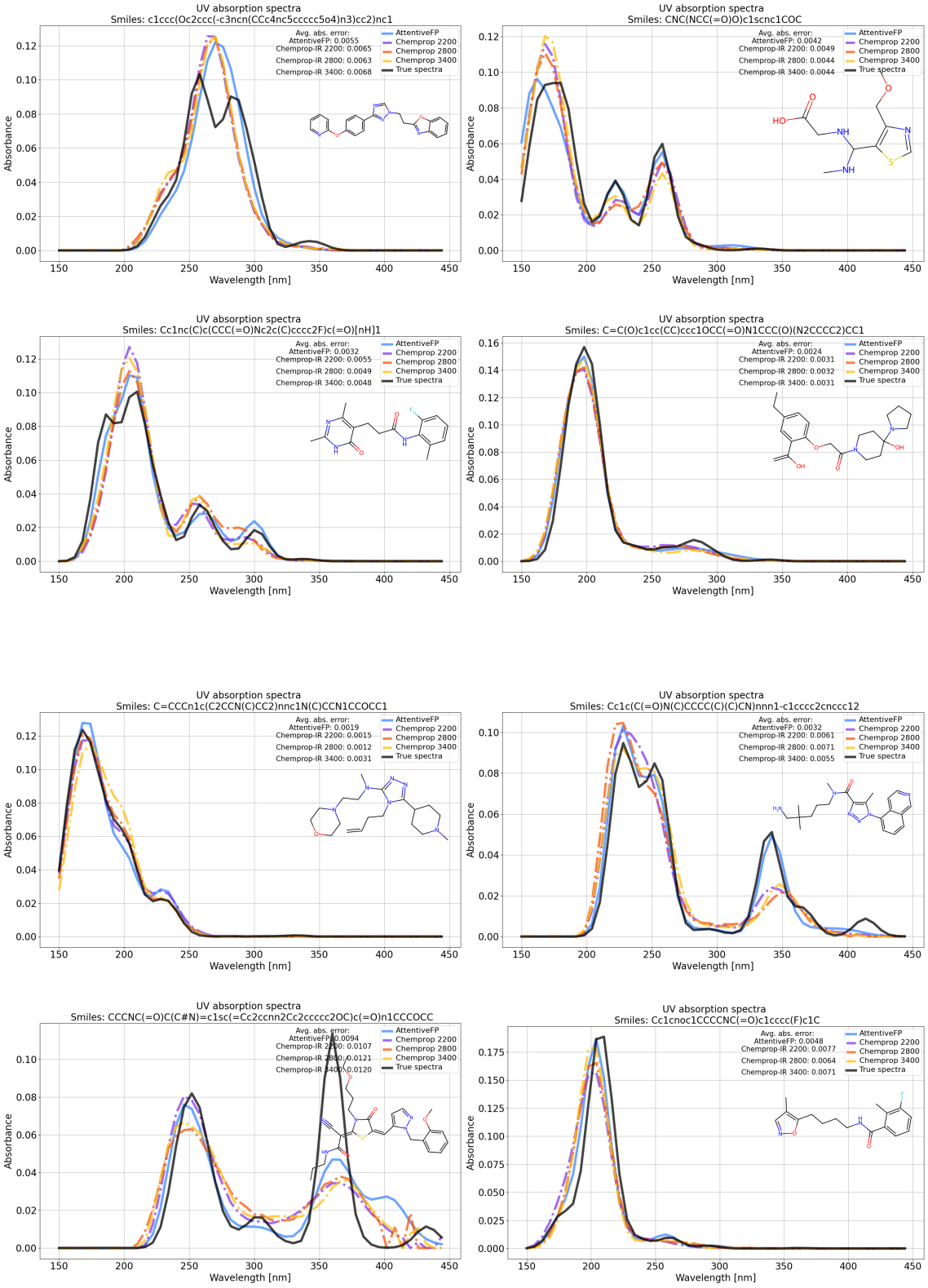


Figure A.1: Scatter plots of predictions on the y-axis and true spectra on the x-axis for 8 different wavelengths. Chemprop-IR (2800) is in blue and AttentiveFP is in orange. The black line ($y = x$) outlines how a perfect model would perform.

A.4 Additional predicted spectra produced by the models



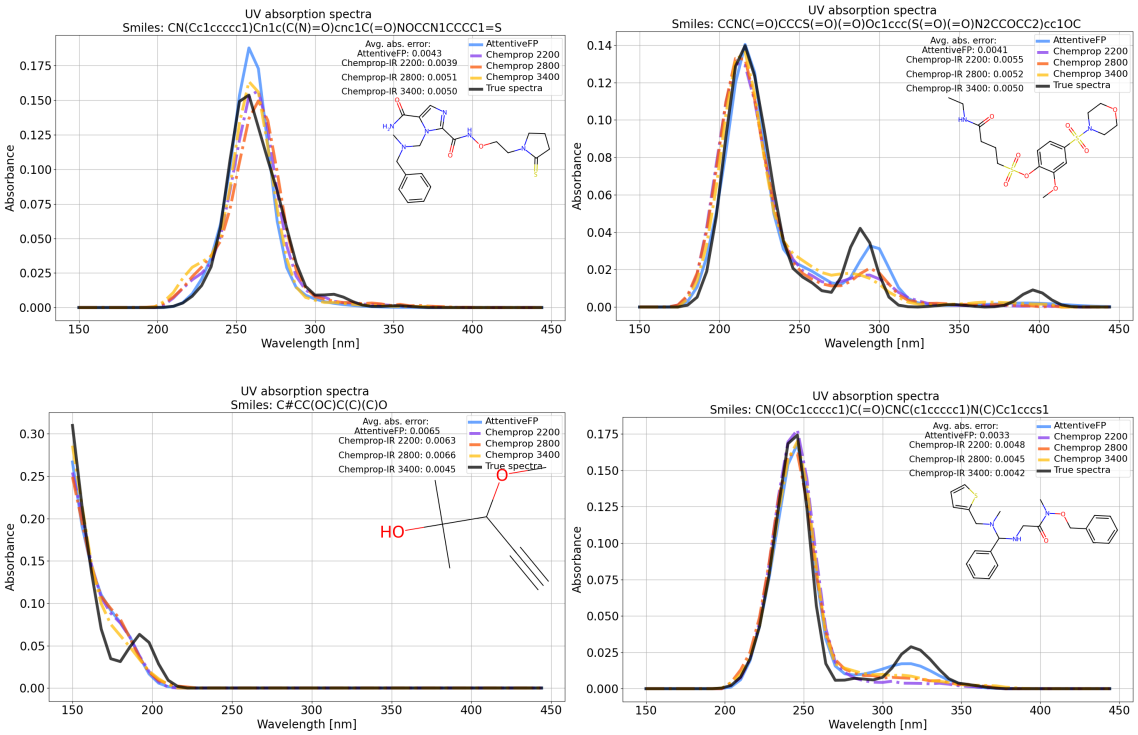


Figure A.2: Comparison between the actual spectra and predicted spectra of both of the models for randomly selected molecules. The black spectra are the actual spectra, the blue spectra are the predicted spectra produced by AttentiveFP. The purple, orange and yellow spectra are the predicted spectra produced by Chemprop-IR with an FFN hidden size and a hidden size of 2200, 2800, and 3400 correspondingly.

DEPARTMENT OF MATHEMATICAL SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY