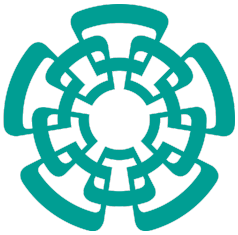


CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL INSTITUTO POLITÉCNICO NACIONAL

Unidad Mérida

DEPARTAMENTO DE FÍSICA APLICADA

"Machine Learning for Molecular Structure Elucidation"

a thesis submitted by:

Saúl Hazel Martínez Treviño

In partial fulfillment of the requirements for the degree of

Doctor of Science

in

Physical Chemistry

Thesis Advisors:

Dr. José Gabriel Merino Hernández **Dr. Víctor Uc Cetina**

Mérida, Yucatán, México July 2021

Abstract

Molecular structure elucidation of organic molecules is a complex and challenging activity that requires expertise and well-suited tools. To assign the molecular structure of a given compound, NMR spectroscopy and mass spectrometry are the most widely used techniques. In practice, these methods are coupled to software tools for helping in the elucidation task and library peak matching of existing reported compounds. Nevertheless, for *de novo* molecular structure determination, or simply in case one wants to obtain a complementary aid, another kind of assistance is required. The contributions of the present work consist of two different approaches for helping in the elucidation task, depending on the analytical tool and the algorithms employed. Firstly, taking into account that molecules found in nature can be grouped into natural product (NP) classes because of structural similarities, we explore the possibility of NP class prediction via ^{13}C NMR data and Machine Learning (ML) algorithms. Employing freely available ^{13}C NMR data of NPs, we trained four classifiers for the prediction of eight common NP classes. The best performance was obtained with the XGBoost classifier reaching f1-scores above 0.82. We also performed experiments with different percentages of positive samples, that is, samples belonging to the NP class to be predicted, including the glycoside presence. Furthermore, we tested cases outside the data set, yielding performances above 80% for most classes. For the chromans case, we restricted the test examples to the coumarin subclass, and the prediction accuracy increased to 100%. The second approach relies on mass spectrometry data and is based on a neural architecture for solving the molecular structure of an organic compound. This proposal consists of an encoder-decoder scheme where an attention-based representation of the input spectra acts as the encoder and a stacked LSTM acts as the decoder. Percentages of correctly predicted structures, f1-scores for different functional groups, and Tanimoto similarities, were used as indicators of model performance. On the test set, the percentage of correct structures achieves a value of 50 %. The f1-scores for selected functional groups result in above 0.8 and mean Tanimoto similarities for two types

of fingerprints result above 0.6.

Resumen

La elucidación de moléculas orgánicas es una actividad compleja y desafiante que requiere experiencia y herramientas adecuadas. Para asignar la estructura molecular de un compuesto determinado, la espectroscopía de NMR y la espectrometría de masas son las técnicas más ampliamente utilizadas. Para ayudar a la tarea de elucidación en la práctica, estos métodos están acoplados a programas informáticos y a la comparación de los picos de espectros obtenidos con los de compuestos reportados en librerías espectrales. Sin embargo, para la determinación de la estructura molecular *de novo*, o simplemente en caso de que se quiera obtener asistencia complementaria, se requieren otro tipo de herramientas. Las contribuciones de la presente tesis consisten en dos enfoques para ayudar en la tarea de elucidación, dependiendo de la herramienta analítica y los algoritmos empleados. En primer lugar, teniendo en cuenta que debido a similitudes estructurales, las moléculas que se encuentran en la naturaleza pueden agruparse en clases de productos naturales (NP), exploramos la posibilidad de predecir la clase de NP presente a través de ^{13}C NMR y algoritmos de aprendizaje automático (ML). Empleando datos disponibles libremente de ^{13}C NMR de NP, entrenamos cuatro clasificadores para la predicción de ocho clases comunes de NP. El mejor rendimiento se obtuvo con el clasificador XGBoost alcanzando puntuaciones f1 arriba de 0.82. También realizamos experimentos con diferentes porcentajes de muestras positivas, muestras que pertenecen a la clase a predecir, incluyendo la presencia de glicósidos. Además, probamos la predicción con casos fuera de los datos de entrenamiento, produciendo resultados superiores al 80% de exactitud en la mayoría de las clases. Para el caso de los cromanos, limitamos los ejemplos de prueba a la subclase cumarina, incrementando la precisión de predicción al 100%. El segundo enfoque se basa en los datos de la espectrometría de masas y consiste en una arquitectura neuronal para resolver la estructura molecular de compuestos orgánicos. Esta propuesta consiste en un esquema

neuronal de codificador-decodificador en el que la representación de los espectros de entrada basada en mecanismos de atención actúa como el codificador y un LSTM apilado actúa como el decodificador. Los porcentajes de estructuras correctamente pronosticadas, puntuaciones f1 para diferentes grupos funcionales y las similitudes de Tanimoto se utilizaron como indicadores del desempeño del modelo. En el conjunto de pruebas, el porcentaje de estructuras correctamente predichas alcanza un valor del 50 %. Las puntuaciones f1 para los grupos funcionales seleccionados resultan por encima de 0.8 y la media de las similitudes de Tanimoto para dos tipos de huellas dactilares moleculares resultan superiores a 0.6.

Acknowledgements

I want to acknowledge and give credits to Conacyt.

Contents

Chapter 1

Introduction

Synthesis and discovery of molecules in nature require expertise and sophisticated tools to characterize their structure and properties. For elucidating the molecular arrangement, key methodologies mainly consist of generating spectral data from spectroscopy and spectrometry. The information obtained is then used by spectroscopists to find the most probable molecular structure. This is done basically in two ways, the first one is via dereplication, that is, matching previously reported molecules; and the second one, performing the unknown new molecules structural determination.[1] For the latter case where de novo elucidation of the molecular morphology is carried on, the main related complication is the challenging combinatorial problem of the total number of possible arrangements, where the complexity increases as the molecular size grows. As a consequence, most of the time, experience and chemical intuition is not enough for the elucidation task and spectroscopists have to rely on computational assistance tools. The use of algorithms for solving combinatorial problems appearing in the elucidation of molecules goes back to the 60's with the Dendral project.[2–4] It is one of the first Artificial Intelligence (AI) systems that is an example of a knowledge-based system, a kind of AI approach that incorporates intensive knowledge rules to systematically reason about domain-specific problems.[2] For the particular case of the Dendral program, a set of empirical rules from Mass Spectrometry (MS) analysis is included in the algorithm in order to reduce the number of structural candidates that result from the

combinatorial explosion of possible isomers. After the Dendral development, several computational expert systems have emerged with different methodologies.[1,5–8] Most of them include the use of spectral data from sources other than MS, such as Nuclear Magnetic Resonance (NMR), Infrared Spectroscopy (IR) for finding the molecular structure. Moreover, contemporary expert systems present a strong dependence on databases for peak matching and also estimate spectra predictions to iteratively obtain the most probable set of structures. Nowadays computational and algorithmic resources are employed as a mandatory complement in the everyday spectroscopy analysis. In fact, similar to the Dendral program, software assistance tools employed now in practice are also considered AI-based systems in the sense that they substitute human expertise using logic rules in an autonomous fashion. However, it is important to mention that those methodologies related to the knowledge-based scheme originated in the 60's differ from modern approaches that have appeared in today's AI renaissance. Last decade has experienced an explosive AI development mainly linked to the Artificial Neural Networks (ANN) third wave emergence. The so-called third wave in ANN corresponds, in part, to the increase in complexity of AI learning models that yielded the Deep Learning (DL) term as a Machine Learning (ML) subfield, and to the increasing data and computational resources.[6,9] DL and ML are part of the learning paradigm in AI, that has been one of the first attempts to provide machines a human-like capability in analogy with a gathering of knowledge from experience. Most recent AI advances are based on ML and DL establishing modern breakthroughs in problems like searching, planning, language understanding, perception, and other AI subfields. Some of the mentioned milestones to which the world has recently paid attention to are AlphaGo,[10] based on Reinforcement Learning (RL) that masters the game of Go, the GPT-3 language model[11] that creates human-like simple essays, and the popular Generative Adversarial Networks (GANs),[12] generative models from which samples generation could be compared with artistic creation. It is important to consider that, given their state-of-the-art quality, some of the elements involved in the AI mentioned models

are usually employed in different application areas. For example, searching problems are also highly related with the optimization of properties in diverse systems.[13] One of those important cases is the chemical space exploration that is performed mostly with RL and generative models applied to inverse design new options of drugs and materials.[14–16] Moreover, the search for new molecular candidates for different applications is usually complemented with the prediction of desired properties using ML algorithms or DL complex architectures.[16] In the case of elucidation related tasks, prediction of spectral properties has been explored recently using DL principally.[17,18] Now, regarding the inverse problem of predicting the structure starting from spectral information, it is relatively recent the interest in solving this challenge leveraging ML capabilities.[19,20] Thus, the fact that the prediction of both spectral properties and molecular structures are being treated with ML raises the question of whether traditional computational approaches can be improved or modified given the more recent advances in AI. Like most of the knowledge reached by science and technology, it is important to get deep into not explored detailed questions in order to obtain a better understanding and motivations for an expansion in the theoretical and technical points of view. In this sense, a detailed analysis for specific organic systems prediction as well as specific types of spectra and ML models could be relevant for both elucidation and applied AI fields of study. With this in hand, this work results after the proposal of ML algorithms and DL architectures for the task of organic molecules elucidation, focusing on two types of spectra information and two different organic systems. The predictive models of those approaches implemented and deployed in end-to-end systems could be similar to and related with modern expert systems but with a renewed perspective based on ML recent developments. Nevertheless, as we will see, they are more limited than expert systems used in practice because, in part, of the available spectral data.

This thesis is mainly divided into two chapters established in accordance with particular predictive challenges that differ principally in the prediction scope. That is, the two schemes depicted in each chapter employ different spectral data and in-

tend to predict different molecular systems. After a gentle introduction to the ML and DL theory in Chapter 2, Chapter 3 describes the classification task of Natural Products (NP) classes comparing classic ML classifiers from ^{13}C NMR data. NPs are molecules with pharmaceutical importance and are generally obtained from biological species.[21,22] Given the size and complexity of NPs, a trade off between the elucidation capacity and the molecule size emerges. In this sense, the elucidation of NPs structures is explored performing a classification only, achieving in this way a clue for the possible complete structure determination at ulterior technical spectroscopic steps. The classifiers selected cover a range that depends on the complexity and time of appearance. In the unlikely case, a linear system is involved in the classification and starting with a simple algorithm, Logistic Regression is the first classifier selected. In addition, regarding the epoch and context, Support Vector Machine (SVM) that is also tested was once considered the king of the classifiers because of its degree of sophistication and accuracy reached. Of course, Multilayer Perceptron (MLP, a. k. a. Neural Network) is added considering the recent reemergence of the Neural Networks. The resultant classifier that performed best in the NPs classification is XGBoost. The selection of XGBoost with decision trees as weak estimators is motivated mainly by two factors. First, the decision trees as core estimators employ an approach that is different from the other three classifiers, yielding with this an extended diversity in the classifiers artifacts. The second motivation is its popularity among industry problem challenges. One important part of this analysis is the use of open source algorithms, that, given their supporting community, provides ML its increasing popularity. Another fact is the harnessing of a free available NMR dataset, alleviating in this way, a little of the current problems in ML inclusion and diversity. Finally, this study provides an awareness of model limitations exemplified with the observation of prediction failures given the lack of specific patterns in data.

Chapter 4 consists in an attempt to predict not a classification of the molecule, but the complete structure from LC-ESI (liquid chromatography - electrospray ion-

ization), a common type of MS technique. To achieve this, a more complex model than the ML approaches presented in Chapter 3 is proposed. For the description of this model, two different contextual meanings of representation play an important role. The first one is about the way the input and output are represented and the second is about the learned representations properly of a DL model. That is, being aware of the objective of predicting a molecular structure starting from spectral data, it is important to take into account the ways input and output are represented and that this selection is intrinsically related with the model architecture and the intrinsic learned representations within it. For the particular case of MS as input, the data is a two dimensional vector of varying size. Further, although data from MS is generally arranged, strictly speaking the order must not be relevant. It is important to mention that dealing with input data with varying size and independent of the elements order is different from DL schemes related with image or audio processing, where input data is scaled or partitioned into vectors of equal size. Thus, a different approach must be established for learning representations from MS information. The DL framework suited for problems where the input contains sets of numbers with varying size and invariant to an specific order comprises the so called set-input models.[23–26] Now, regarding the output side, the predicted molecules could be represented among different options such as graphs, matrices or string sequences. However, as stated previously, the selection of the molecular representation depends on the architecture designed to output these entities and the architecture must deal with the input representation. Two examples of this interdependence are models that predict molecular properties and models conceived for molecular structure generation. The former regards molecules as graphs harnessing representations from the Graph Neural Networks (GNN) framework.[27–29] The latter type of design outputs molecular structures represented as sequences leveraging this way sequence prediction architectures like translation mechanisms or speech recognition. For the case of string sequence representation of molecules, it has their utility in the virtual screening and drug design fields where it is important to reduce

the information as much as possible while preserving structural description.[30–32] One of the most remarkable examples of string sequence representation that have been employed widely in generative models is SMILES (Simplified Molecular Input Line Entry System).[33] Nevertheless, given the random factors involved in the generation, most models fail to output grammatically valid SMILES in a considerable percentage.[32,34] As a consequence, different representations like DeepSMILES[35] and SELFIES[34] have been proposed for generating valid strings from sequence related models partially alleviating the randomness issues. After considering the dependence of the output and input representations on the model design, an architecture for the problem of molecular elucidation from MS is proposed. The architecture takes the form of an encoder-decoder,[9] broadly used as a basis of several AI models, where the learned representations in the encoder are regarded as a *set-input* model and the decoder generates DeepSMILES predictions. Furthermore, the *set-input* model implemented in the present design, the *Set Transformer*[23], relies on self-attention[36] mechanisms that are almost ubiquitous in today's AI schemes. The learned representations from the encoder incorporates then attention mechanisms for modeling high order interdependencies between the input elements. The decoder is composed by a Recurrent Neural Network (RNN) using LSTM[37] (Long-Short Term Memory) units for outputting string sequences symbol by symbol. RNNs based on LSTM cells had shown remarkable performance in translation mechanisms and language models tasks achieving state-of-the-art before the Transformers[36,38,39] language models arrival, which are based on the previously mentioned attention mechanisms.

Being aware of the representational power and mathematical conceptualizations developed within the modern AI paraphernalia context, ML and DL approaches for solving molecular structures are a methodological complement for the logic-based systematizations employed in current computational assistant tools. Learned representations from a predictive model are expected to recognize important patterns in a similar way to an experienced spectroscopist. In this sense, the current work is to present another way a molecular structure is solved, contributing as a starting point

for future development from the point of view of artificial intelligence. Summarizing, the main contribution of the present thesis is the proposal of two predictive models for elucidating molecular structures employing Machine Learning algorithms. The particular objectives of the thesis can be stated as follows:

- Train a ML model for the classification of NP classes using ^{13}C NMR data.

- Train a neural architecture for the prediction of organic molecules from LC-ESI data.

Chapters 3 and 4 detail the motivations, specific methods, and results regarding these two objectives. But before the specific methodologies description, Chapter 2 introduces a general background of ML and DL for a proper understanding of the two proposed predictive methods.

Chapter 2

Machine Learning and Deep Learning Methods

2.1 Machine Learning Classifiers

This section briefly describes ML classification algorithms that will be employed in Chapter 3 for predicting a NP class. The Logistic regression, Support Vector Machine, and XGBoost are covered in the next lines and the Multilayer Perceptron will be explained in the next section. Logistic regression is a classification method that outputs a binary value from the logistic sigmoid function. The sigmoid function is applied to a function of the input vector. A simple representation of a linear discriminant function of the input is shown in Equation 2.1.

$$y(X) = W^\top X + w_0. \tag{2.1}$$

The input vector X, corresponding to a certain feature or dimension, are multiplied by a parameter vector W, from the same feature, and a bias w_0 is added. Eq. 2.2 shows the sigmoid function applied to the input function.

$$y(X) = \frac{1}{1 + e^{-(W^\top X + w_0)}}. \tag{2.2}$$

Then, the optimization problem consists of learning the parameters that best classify the samples measured by a score. In order to avoid overfitting, a regularization term E_w is added to the loss function (Sum-of-squares error function), where the parameters λ, and the penalties q must be specified (Eq 2.3).

$$E_w = \frac{\lambda}{2}|W|_q \tag{2.3}$$

The basic idea of the Support Vector Machine is not to be so rigid in the decision boundary for the classification task. That is, an optimum margin is established after the maximization of the distance between the decision boundary and the closest correctly classified point. This maximum margin is obtained by finding W and b that maximizes:

$$\frac{1}{||w||}[t_n(W^\top \phi(x_n) + b)], \tag{2.4}$$

where x_n are the points closest to the decision boundary, $\phi(x_n)$ the feature-space transformation of the input variable, and t_n the target value of point n. For overlapping class distributions, the goal is to maximize the margin while penalizing points that lie on the wrong side of the boundary. This is done by minimizing

$$C\Sigma_{n=1}^{N}\xi_n + \frac{1}{2}||W||^2, \tag{2.5}$$

where the hyperparameter C controls the trade-off between the margin and the variable ξ, which represents a measure of misclassification. Another attractive property of SVM is the use of kernel functions to convert the input from its original space to a new higher-dimensional space. So, in case that the categories are not linearly separable in the original input space, the feature space could yield a linear separation. Furthermore, the kernel functions favor a faster computation of the inner products involved in the optimization problem. The most common kernels used in practice are the linear, polynomial, sigmoid, and Gaussian kernels. XGBoost[40]

has recently shown increasing popularity due to its performance and accuracy on machine learning challenges. The boosting idea is based on a decision established by several weak learners that evolve over time. This evolution is referred to the continuous updates of the weight distribution of the samples and the weight or importance of each classifier. The AdaBoost algorithm is the example *par excellence* of this methodology. Another boosting method that incorporates the derivative of the loss function instead of using weighted examples is the Gradient Boosting Machine. This algorithm mitigates drawbacks like overlapping and mislabeling data from AdaBoost.[41] XGBoost, like the Gradient Boosting Machine, contains decision trees as base weak learners. It has a more efficient implementation providing the most powerful use of computational resources, i. e., parallelization and memory use. Additionally, sparse data handling is a novelty in the XGBoost idea, allowing high scalability providing efficiency with billions of examples without increasing the use of resources.

2.2 Deep Learning

One of the origins of the 'deep' qualification in the Deep Learning term regards the extension of an Artificial Neural Network (ANN). For example, in a MLP (see below) the adjective 'deep' is assigned for the presence of multiple hidden layers and units achieving a 'deep' learning because of this 'deep' architecture. A similar fact occurs with other kinds of neural architectures such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). But, for the multiple layers from a theoretically conceived model makes practical sense, the 'deep' term as an abstract concept is extended to also include a concrete definition referring to material resources such as computational power and data availability. One example of what is now considered an archetypical DL model which includes several layers in the architecture as well as GPUs parallelization among other complex characteristics is the AlexNet,[42] proposed as a solution to the ImageNet competition. This challenge consisted of a classification of thousands of categories using millions of samples for

training the models.[43]

Besides model size and complexity, one of the most prominent characteristics of DL is the capacity of learning representations for a classifier or regressor component in the supervised learning case. Traditional ML for computer vision problems engineer descriptors adapted to specific tasks while DL, instead, relies on an end-to-end approach for learning the parameters in the hidden layers that correspond to an appropriate representation for the final prediction layer. Deep architectures learn representations required to perform not only supervised tasks, but also for problems related with unsupervised learning such as transfer learning, disentanglement of factors, model compression, and dimensionality reduction.[44] Important examples of unsupervised learning are the autoencoders that change the original dimensions for obtaining good representations for downstream tasks.[45,46] One remarkable example is the unsupervised pretraining[47] by Hinton and Salakhutdinov consisting of a fine-tuning of a deep autoencoder weights for image reconstruction.

DL capabilities have motivated today's commercial interest, like face recognition,[48] object segmentation and tracking,[49,50] self-driving cars,[51,52] and many others. Deep Reinforcement Learning is a more recent success history. Architectures for navigating, solving search tasks, games,[53–55] among others are possible in part for deep representations for policy functions approximations. Moreover, awesome performances are also obtained using deep CNNs for translating raw image data in atari games.[56] Deep learning has also been involved in linguistics tasks, such as chatbots and language models for text generation.[11,38,57] For example, in problems that require continuous representations for language entities like words and sentences, the word embedding scheme coupled with Recurrent Neural Networks (RNNs), that include internal memory states and are designed for sequences, reach appropriate semantic modeling for different tasks.[58,59]

2.2.1 Multilayer Perceptron (MLP)

Within common ANNs architectures, the MLP has played an important role in classification or regression tasks. In fact, the Neural Network term has been commonly employed as a synonym of the MLP. For a brief description of the MLP, the starting mathematical representation of a linear or nonlinear combination of basis functions is considered. This is commonly expressed as:

$$y(\mathbf{x}) = f(\Sigma_i^M w_i \phi_i(\mathbf{x})), \tag{2.6}$$

where the predicted value y is the result of applying an activation function f, that can be the identity function or the nonlinear sigmoid in the classification task, to a linear combination of M basis functions ϕ with the respective parameters w_i and the input vector $\mathbf{x}$. Then, a MLP or a two layer network is established if the basis function ϕ is substituted with a nonlinear combination of the input x. This nonlinear combination is represented as a nonlinear activation function g applied to the sum of the vector x elements x_j and their adjustable parameters θ_j:

$$y(X) = f(\Sigma_i^M w_i (g(\Sigma_j^D \theta_j x_j))). \tag{2.7}$$

A representative diagram of these functions for a classification of k classes is shown in the Figure 2.1. In this scheme the number hidden units of the corresponding neural network is now specified in each layer as the number of functions M.

The optimization problem of finding the parameters that minimize an error function,

$$E(\theta) = f(\hat{y}, y), \tag{2.8}$$

can be viewed as an approximation to the global minimum of the parameters space surface. A common method to explore the parameters surface is based on the

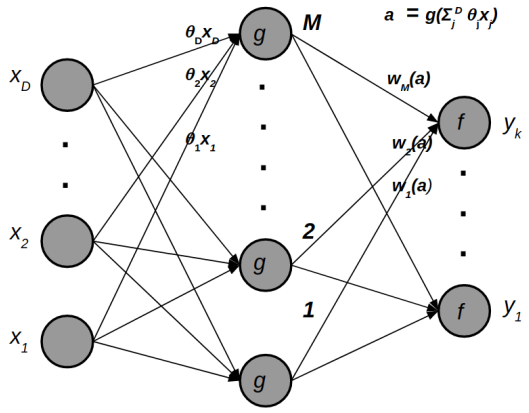


Figure 2.1: A MLP or neural network schematic representation. For simplicity, the summatory of x_j has been absorbed by the term a.

gradient information of the error function. The gradient ∇E is obtained via backpropagation, a differentiation scheme that computes the error function derivatives with respect to the parameters using the chain rule for a subsequent updating of the parameters.

A basic way to update the parameters from θ^t to θ^{t+1} uses the gradient information and the learning rate adjustable parameter η:

$$\theta^{t+1} = \theta^t - \eta \nabla E(\theta^t). \tag{2.9}$$

The origin of the MLP term is due to the resemblance of the Rosenblatt's perceptron. Although Rosenblatt's perceptron employs a heavystep discontinuous function for classifying linearly separable regions, the MLP uses continuous functions and nonlinearities in more than one layer. In addition, because of the noncyclic way the flow of information is performed forwards, the feed forward NN term is adjudicated to the MLP. In backpropagation instead, the information is sent backwards for the parameters updating. Indeed, the backpropagation algorithm is in part responsible for the reemerged interest in using MLP in the 80s, although the algorithm was first conceived back in the 60's.[60]

2.2.2 Convolutional Neural Networks

In an attempt to achieve a similarity with human intelligence, one of the main AI problems is to perceive objects from an environment. In the specific domain of computer vision, perception integrates different tasks such as detection, recognition, and classification. The way information from images raw pixels is transformed and employed to perform those duties is possible thanks principally to the Convolutional Neural Networks (CNNs). For the image case, an input conformed by a 3D tensor corresponding to the width, height and color values is partitioned into patches of small size. Then, the patch size defines the number of shared weights for learning contained in the feature or kernel. For example if the patch size is 3x3, a total number of 9 weights forms a feature and the same weights are employed in each feature of the feature map. Also each feature has a number of filters giving a total of 3 dimensions in the feature. Now, the convolution is the 'scanning' or 'slicing' that is performed all along the image patches with a defined step size ('stride') and transforms the patches to an output feature map of 3 dimensions: weight, height, and filter dimensions. Main causes for doing this are three-fold. Firstly, to provoke sparsity in the input weights, given that only small building patches are employed to learn patterns. Secondly, parameter sharing is obtained from this scheme. And finally, the learning of only local patterns relating points within the selected patch area allows a translation invariance property.[9,61,62] This is exemplified in the scheme of Figure 2.2.

The translation invariance property has useful consequences for learning the same object of interest located at different image positions from diverse samples. It is important to notice that convolution in the computer vision context is not exactly the same as the convolution in the signal processing context and that the mathematical functions employed in CNNs are called cross-correlations. Further details are provided by Goodfellow[9] and Venkatesan[62]. Now then, a CNN is basically an architecture conformed by layers that perform the convolution operation generally coupled with pooling and fully connected layers with different order vari-

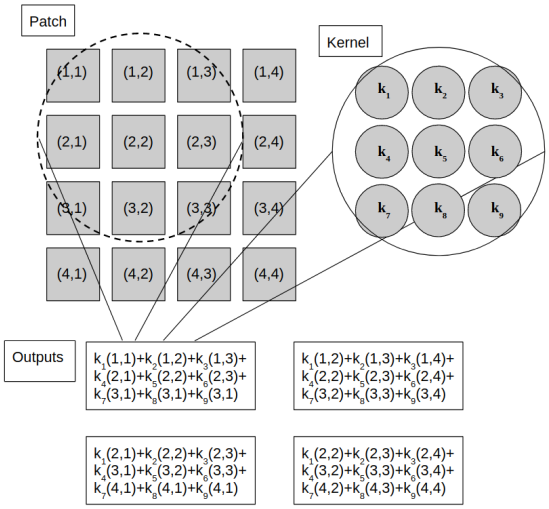


Figure 2.2: Illustration of the convolution operation.

ations. After obtaining the feature mapping, pooling operations such as mean or max as well as fully connected and non-linearities passing are carried on in different arrangements. Modern CNNs reach powerful hierarchical representations that differ from traditional designed filters for specific tasks. For example, explicit features with certain shapes that are the input of a ML classifier. CNN architectures designed for solving computer vision problems have established landmarks in the last decade.[63–65] These training required tons of data and computational resources for achieving state-of-the-art results. Thus, another important research field for the CNNs learned representations is transfer learning for diminishing the required training data.[66]

2.2.3 Recurrent NNs

Different predictive problems involve sequence entities such as sentences for language tasks or time-series data. An important property of these kinds of prediction problems is that the information in a particular state for a specific task depends on previous states, where each word or symbol in sentences or each time step in the time-series can be considered as states. This situation has been modeled via Hid-

den Markov Models (HMMs),[67] but they have the disadvantage of ignoring previous history and only consider a dependency on contiguous last states. A consequence of this limitation is that when treating long states dependencies HMMs fail to properly make suitable predictions. Moreover, given that predictive tasks in sequences depend on previous states prediction and that sequences themselves vary in sizes, standard NNs or CNNs do not provide a suitable solution. Instead, RNNs are designed with this purpose. RNNs date back to the 1990s as an attempt to explicitly model memory and time in a dynamical systems context.[68] Basically, a recurrent network is a cyclic architecture meaning that the output of the networks depends on the input and the output from the previous time step. In other words, the input of a particular state enters the RNN and an output, called hidden state, enters again to the same RNN with the input of the next state and so on. Equation 2.10 is the representation of this input-output relationship where the input x_t is a variable of interest, h_t the hidden state at the state t, and f the nonlinear transformation.

$$h(t) = f(h_{t-1}, x_t). \tag{2.10}$$

Because the hidden state h_t refers to the hidden state h_{t-1} in a feedback fashion, the function represented in eq. 2.10 is called recurrent network. A formal point of view of RNNs is to consider it as a folded computational graph where the information from a numerical matrix corresponding to different states enters to the computation node (RNN unit) and the respective hidden states flow recurrently to the nonlinear mapping function. Then, when unfolding the equation, the expression can be expressed as a Directed Acyclic Graph (DAG). A scheme from this is illustrated in Figure 2.3.

In the context of discrete values prediction, such as words or tokens, the output o_t is passed over a $softmax$ function for obtaining the prediction y_t. In addition, as we will see in the Encoder-Decoder section (Section 2.2.5), the output y_{t-1} could be passed as the next input x_t for every time steps. In this case, special tokens are placed at the starting and ending steps, as seen in the Figure 2.4. Some training

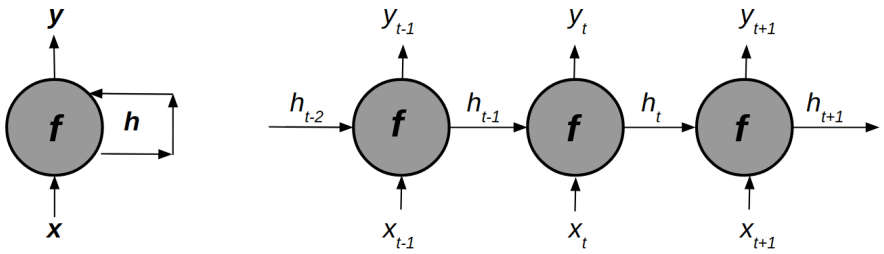


Figure 2.3: Representation of a RNN in a folded (left) and unfolded (right) fashion.

models employ the ground truth $\hat{y}$ instead of the predicted y for a faster learning. This scheme is called teacher forcing and in practice, the training step combines samples with and without teacher forcing at a certain predetermined ratio value. Next equations show the computation for each time step t:

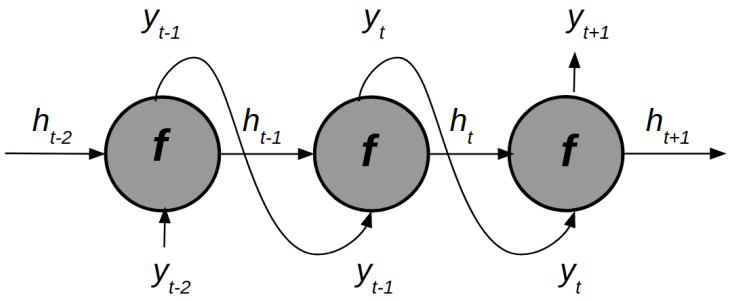


Figure 2.4: RNN that predicts an output y_t used to the input at the next step y_{t+1}.

$$h_t = f(wh_{t-1} + ux_t + b) \tag{2.11}$$

$$o_t = vh_t + c \tag{2.12}$$

$$y_t = softmax(o_t) \tag{2.13}$$

The w, u, and v are learnable parameters and the b and c entities, the biases terms. A common issue in training RNNs with long sequences is that the involved numerical dependencies such as the error gradient either explodes or vanishes. This problem is known as the vanishing or exploding gradient problem and several impor-

tant RNNs have been designed for solving it. The most famous RNN solution to the vanishing gradient problem is the highly cited Long-Short Term Memory (LSTM),[37] invented by Hochreiter and Schmidhuber in 1997. In order to control the flow of information the basic components of the LSTM are the gate functions and the memory cell. Moreover, the memory cell is employed as an extension of the hidden state for preserving the memory and gradient information over long sequences. The memory cell, c_t depends on the input and forget gates, i_t, and f_t, respectively, as well as the previous memory cell state c_{t-1} and the update cell gate g_t:

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t, \tag{2.14}$$

where $\odot$ is the elementwise (Hadamard) product and the update cell gate, g_t depends on the input, x_t, previous hidden state, h_{t-1}, and the biases terms b_{ig} and b_{hg} activated with the $tanh$ function:

$$g_t = tanh(w_{ig}x_t + b_{ig} + w_{hg}h_{t-1} + b_{hg}) \tag{2.15}$$

The input, i_t, forget, f_t, and output, o_t, gates are differentiable soft functions depending on the input, previous hidden states, the respective adjustable parameters w_{ii}, w_{hi}, w_{if}, w_{hf}, w_{io}, and w_{ho}, and the respective biases terms, b_{ii}, b_{hi}, b_{ii}, b_{hi}, b_{io}, and b_{ho} activated with the sigmoid function, σ:

$$i_t = \sigma(w_{ii}x_t + b_{ii} + w_{hi}h_{t-1} + b_{hi}) \tag{2.16}$$

$$f_t = \sigma(w_{if}x_t + b_{if} + w_{hf}h_{t-1} + b_{hf}) \tag{2.17}$$

$$o_t = \sigma(w_{io}x_t + b_{io} + w_{ho}h_{t-1} + b_{ho}). \tag{2.18}$$

Given that the gates are in the range (0,1) they can allow a complete pass of information or delete all the information that is passed through the memory cell. Then, for obtaining the new hidden state, h_t the elementwise product $\odot$ is computed

with the *tanh* of the new memory cell, c_t, and the output gate, o_t:

$$h_t = o_t \odot tanh(c_t). \tag{2.19}$$

Similar to the simple RNN (eq. 2.13) the output is obtained via a softmax function applied over the hidden state h_t in a discrete value prediction context like symbol sequences. A brief depiction of a LSTM unit is illustrated in the figure 2.5.

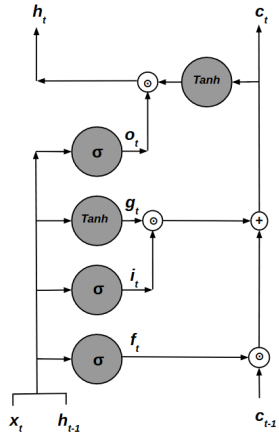


Figure 2.5: A LSTM unit representation.

This architecture yields powerful representations well suited for strong memory dependencies, required in sentences from language tasks, but also in speech recognition[69], music generation[70] among other applications.[71,72] Another important RNN that is based on the use of gates but simpler than the LSTM, is the Gated Recurrent Unit (GRU).[59,73] With the aim to study the effect of the gates in the LSTM and GRU networks for a proper prediction of sequences, in 2014 Chung *et al.* proposed a way to get similar performance but in a simpler architecture. The main findings were that decent results can be achieved with fewer gates and without the memory cell as in LSTM. Another important type of LSTM is the bidirectional-LSTM, emerged as a necessity of knowledge of future information in tasks related with speech recognition and linguistics.[74]

2.2.4 NLP

Natural Language Processing (NLP) is one of the most important areas within Artificial Intelligence. It is mainly conformed by computational approaches to linguistics in order to solve difficult problems. One of the main challenges is that, different from a formal language, i,e, computer languages, natural language is not purely based on logical structures.[75] In this sense, successful approximations to semantics, syntax, and meaning from processed information is one of the NLP main objectives. Most NLP applications are the design of algorithms that try to reach human-like capabilities in the performance of tasks related to language via text data. Some of those important mentioned tasks are question-answering systems, information extraction, text generation, speech recognition, text summarization, machine translation, among others. The resulting success is summarized in everyday typical applications such as chatbots and translators. Most recent advances are sentiment analysis applied in important fields such as economics and politics, where the general perception of an audience about a topic is analyzed. Also, other important NLP usages are analysis of propaganda and fake news,[76] and even identification of personal traits.[77] The number of examples of this type is huge and we don't have enough space to cover all of the NLP implications. It is important to note that NLP challenges and opportunities are intrinsically related to the fact that most data nowadays is composed in the form of text. Moreover, an important active research area is the way text is processed for passing from discrete quantities to another form of representation. Thus, the publications of new algorithms for specific tasks and representation proposals for text are increasing every year. As we will see in next sections, the mentioned algorithms harness the proposals of continuous differentiable spaces that emerged thanks to the backpropagation algorithms employing an always increasing computational power. Also, the proposed algorithms are mainly NN architectures that try to achieve and outperform state-of-the-art in some typical evaluations. Different scores and benchmarks have been the standard for different NLP tasks. For example in translation mechanisms BLEU[78] is the most widely used

score. Moreover, for language modeling, understanding, and other NLP problems, other evaluations such as SSA[79] (Sensibleness and Specificity Average), datasets of questions (SQuAD[80], SWAG[81]), word prediction datasets (LAMBADA[82]), and multilingual benchmarks (XTREME[83]) have been designed. Nevertheless, given that natural language is so complex -in fact, it has been said that NLP tasks are among the most relevant capabilities to achieve what could be an Artificial General Intelligence (AGI)- there are some reservations of the common evaluations and metrics implemented. Several discussions have emerged about the capabilities of a machine to comprehend and understand the sentence meaning themselves. One example is the analysis and critics of the BLEU score issues related to its failures about meaning and sentence structures.[84,85] In addition, another attempt to evaluate language understanding capabilities is the creation of the GLUE[86] and superGLUE[87] scores. They are designed to perform different designed tasks that are supposedly needed for a language understanding. Moving beyond, the philosopher John Searle has stated a problem on weak and strong AI, depending on the imitation intended by an algorithm.[88] That is, a weak AI is an algorithm designed for human imitation and does not necessarily hold a genuine intelligence, referred as strong AI. Finally, we must state that NLP is advancing in giant steps, and the methods described here as state-of-the-art could be overpassed in the few months after the publishing of this work.

2.2.5 Attention mechanisms

This section covers important works and architectures related with some NLP tasks such as NMT, image caption, and others, that employ attention mechanisms as the main core. We first describe the antecedents of one of the most cited architecture that introduces a solution for the alignment problem. After this model publication, a lot of schemes were based on this alignment idea for different tasks within many different DL areas. The idea is to obtain a background and perspective for attention in general to later employ some elements for our own attention-based proposal.

Previous work on NMT

Encoder-decoder

Neural Machine Translation (NMT) emerges after conceiving a continuous representation of language words. This representation is a differentiable vector space from the input embedding and allows a learning of word and phrase joint probability distributions (Neural Language Model). The way probabilities of words and phrases is treated in NMT is one of the main differences from phrase-based SMT (Statistical Machine Translation), where the words or phrases in the target language are conditioned on single words or phrases from the source language. Moreover, for the translation task, there is also a dependence on the context, and one advantage of NMT is the capability to represent context and semantic relationships in the continuous space. This space of continuous representations from non-numerical objects such as words and sentences is mostly achieved with an encoder-decoder. The encoder-decoder framework is a general abstraction term that describes the conversion of input variables, typically with a NN (Neural Network), into a fixed-length vector and then the conversion of this vector back to an output variable. In the context of NMT it would mean that the information of a source sentence is encoded in another representation, z, and then this vector is decoded to a target sentence. This could be better exemplified in the next picture (Fig. 2.6).

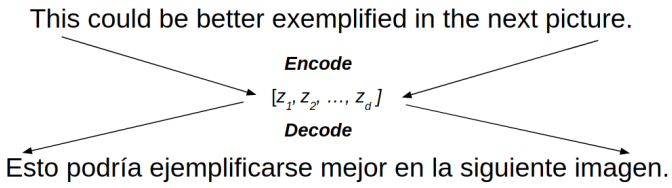


Figure 2.6: Schematic representation of encoding a sentence to a continuous space and decoding back for NMT.

We could mention at least two general models that encode and decode sentences depending on how the encoding is performed. Firstly, the work[89] by Kalchbrenner *et al.*, which propose a CNN based on n-grams (sequence of n words) to the encoding

part coupled with a RNN to decode the sentence generation. The second important NN architecture that learns phrase representations is the encoder-decoder by Cho *et al.*[59] In this encoder-decoder model, the encoding is not carried on by a CNN, but with a RNN. The Figure 2.7 shows a schematic representation of the encoder-decoder model of Cho *et al.*

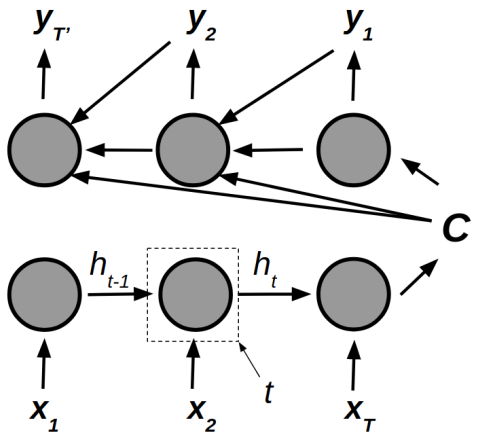


Figure 2.7: The encoder-decoder model from Cho *et al.* x_T and $y_{T'}$ are the final inputs and outputs and the circles represent a RNN function.

For the translation mechanism, each input word continuous representation at time t, x_t enters the RNN. Then, the hidden state h_t in the encoder part are obtained from the RNN (eq 2.10, see RNN section) given the previous hidden state h_{t-1} and the input x_t. The resulting hidden state after the processing of all the sequences can be thought of as the encoded input, c, also known as the summary vector. After encoding the input in a fixed-length vector, a decoding must be done to retrieve a string sequence, but depending now on the summary vector, c, that is equal to the final hidden state. Then the hidden state h_t in the decoder for the prediction of word y_t is then:

$$h_t = f(h_{t-1}, y_{t-1}, c). \tag{2.20}$$

The RNN chosen by the authors was a proposal of a LSTM (Long Short-Term Unit) variation, with only two gated units. This model was tested on English-

French translation and performed quantitative and qualitative analysis. This former resulted in BLEU scores comparable with the SOTA (state-of-the-art) at that time and the latter showed consistent semantic embeddings between learned phrase representations. After this encoder-decoder model, Cho *et al.* published another architecture[73] consisting of an encoder via a CNN and a RNN as the encoder, similar to the Kalchebrenner work. They compared both performances on English-French translation resulting in difficulties to translate satisfactorily large sentences.

Seq2seq

A similar work to the encoder-decoder model for problems related with sequences and their representations is the seq2seq[71] (sequence to sequence) by Sutskever *et al.* This model is basically an encoder-decoder with a LSTM with 3 main differences from the work of Cho *et al.*: 1) One LSTM for encoding and other for decoding, 2) Employed 4 Stacked LSTMs, 3) Used different input orders. Figure 2.8 shows a scheme of the seq2seq model. Notice the similarity with the encoder-decoder by Cho *et al.*[59]

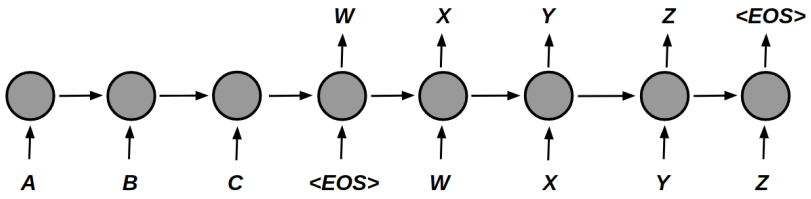


Figure 2.8: Representation of the seq2seq model. For indicating termination, the symbol <EOS> (End of Sentence) is added to the symbols of the input sentence 'ABC' and the output sentence 'WXYZ'.

The results on BLEU for English-French translation resulted in an important effect of the LSTM stackings. This also had remarkable results when dealing with long sentences. Another important task of NLP is to relate word embeddings with semantic representations. In fact there are algorithms designed specifically for carrying on this task such as Word2Vec[58] and GloVe.[90] As we will see later, the encoder-decoder architecture is not restricted to a machine translation task and this model could learn semantically consistent embeddings. As in the case of Cho's encoder-decoder, the

model from Sutskever yields phrasal representations with semantic consistency that could be projected into a 2D dimension to analyze the semantic meanings.

2.2.6 Attention

Translation

One of the drawbacks in the encoder-decoder based models is the difficulty to align the input and output words. This is partially due to the management of fixed-length vectors (summary vector, c in the encoder-decoder model) from which the decoding step is performed. In 2015, Bahdanau *et al.* reported[91] a new way to overpass this bottleneck by allowing the model to focus on different parts of the source words in order to predict a particular output word. That is, instead of relying on just one fixed-length vector, the model makes use of different input vectors for decoding and performing the translation. Attention is a general term for the mechanism in which different elements of an output are associated with input elements via a vector of importance weights. So, in the translation case the input vectors are weighted for attending them with different importances with respect to a certain output. Thus, this achieves a consistent alignment and alleviates the limitation of the summary vector c fixed length. In the Bahdanau's attention proposal, the vector of importance weights contains the encoded inputs and is called context vector. The context vector is then a weighted sum of elements called annotations:

$$c_i = \Sigma_j^{T_x} \alpha_{ij} h_j. \tag{2.21}$$

The h_j terms are the annotations and correspond to the hidden state for the input j. The annotations are where the encoder maps the input sentence using a biRNN. They result after the concatenation of the forward and backward hidden states for each word x_j exemplified as:

$$h_j = [h_j^{forward}; h_j^{backward}]. \tag{2.22}$$

This bidirectionality allows not to just focus on the word itself but in the sentential context around it. The context vector can also be considered the expected annotation over possible alignments with probabilities α_{ij}. Then, the decoder uses the context vector in order to decide on which parts of the encoding input it should focus. And for each coefficient α_{ij} in eq. 2.21:

$$\alpha_{ij} = softmax(e_{ij}), \tag{2.23}$$

where

$$e_{ij} = a(s_{i-1}, h_j). \tag{2.24}$$

Bahdanau calls the term e_{ij} energy, that is, the energy associated with the input j and output i with probabilities α_{ij}. The function a is the alignment model that is also considered a score function for measuring how well the inputs near j, represented with its encoder hidden state h_j, and the outputs near i, represented with its decoder hidden state s_{i-1}, correspond. In the Bahdanau's model the alignment a is obtained via a MLP:

$$a(s_{i-1}, h_j) = v_a^\top tanh(W_a s_{i-1} + U_a h_j). \tag{2.25}$$

v_a , W_a, and U_a are weight matrices. The hidden state of the decoder, s_i is estimated via RNN:

$$s_i = f(s_{i-1}, y_{i-1}, c_i), \tag{2.26}$$

where, the RNN f is the GRU architecture. The final step then is a multilayer network with a maxout layer for obtaining the probabilities of the words predicted from the hidden state s_i. The Figure 2.9 is a scheme summarizing the encoder-decoder with attention mechanism as proposed by Bahdanau.

In the performed experiments, the achieved BLEU scores overpassed the encoder-

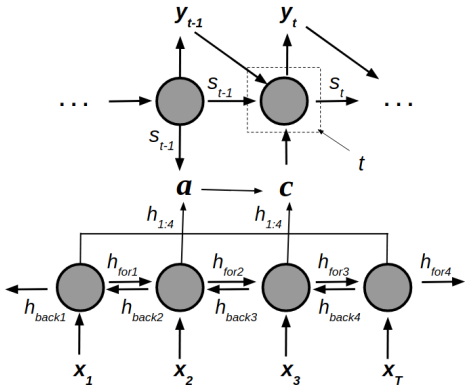


Figure 2.9: Schematic representation of the attention model of Bahdanau *et al.* The hidden states for the bidirectional RNN are the variables h_{for} and h_{back} for the forward and backward hidden states respectively.

decoder by Cho *et al.*[59] The authors tested the capacity of the model for treating long sentences. They employed two sets, one with sentences length up to 30 words and other with lengths up to 50. The attentional scheme addresses the issue of treating long sentences as results in a better performance than the conventional encoder-decoder. The main reason is to accurately focus on particular words while not employing a fixed-length vector as in the encoder-decoder.

After the publication of Bahdanau's seminal paper, Luong *et al.* explored different adaptations from Bahdanau's model.[92] They tested alternatives for the alignment model besides the one corresponding to the Bahdanau's model (Eq. 2.25). The considered alignment functions score(h_t , h_s) are called *concat*, *dot*, and *general* and are represented as $h_t^\top h_s$, $h_t W_a^\top h_s$ and $W_a[h_t; h_s]$, respectively. Where the hidden state of the target is ht and the source h_s. The W_a is a learned parameter. Although this classification assigns the *concat* term to Bahdanau's model, other authors[36,93] call Bahdanau's alignment function *additive* referring to the addition in Eq 2.25. In addition to the comparison of the mentioned alignment versions, another contribution from this study is the proposal of the local attention model. It is conceived for the drawback of needing to attend to all input elements for each target word as in the global case. Thus, the local approach only selects a subset of the source per output word. This idea of how to perform the selection of this subset was taken

from Gregor *et al.* '*SelectiveAttention*', which selects fragments of images to focus on and generate other images.[94] Basically, the local model proposes a 'window' for each alignment position, p_t, for each target at time t. This window is a space covered by $[p_t - D, p_t + D]$, where D is empirically selected. Thus for giving importance to points near p_t a Gaussian distribution is centered around p_t. The alignment position is obtained via :

$$p_t = S\sigma(v_p^\top tanh(W_p h_t)), \tag{2.27}$$

W_p and v_p are model parameters which are learned to predict positions and S is the source sentence length. And the alignment weights:

$$a_t(s) = align(h_t, h_s) exp^{(-\frac{(s-p_t)^2}{2\sigma^2})} \tag{2.28}$$

$\sigma = D/2$ and the align method are one of the *concat*, *dot* and *general* versions. An scheme for the local attention model is shown in Fig. 2.10

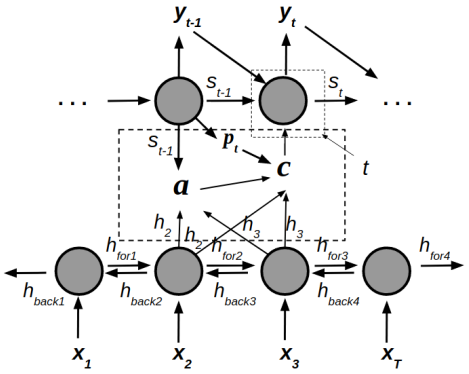


Figure 2.10: Architecture of the local model of attention.

The proposed local model is also compared against a global model that is very similar to the Bahdanau's scheme. This global approach differs from the Bahdanau's approach in the encoder, that doesn't employ a BiRNN and the way the computation steps are done. The main findings were in the obtained BLEU, and perplexity scores, higher with local-attention schemes than non-attention mechanisms. Also, in per-

plexity, the *concat* models in global and local obtained poor results. Subsequently to the mentioned models studies, Kim *et al*, performed an study[95] of attention architectures that consider relevant structural relationships in the input elements. That is, instead of delegating the network the implicit learning of the structural dependencies in the source, the authors propose an explicit modelling for those entities. Examples of some structural dependencies in input sentences for different tasks are the input segmentation, selection of chunks, and attending to latent syntactic subtrees. The authors called their approach 'Structured Attention Networks' and two main methodologies are: linear-chain conditional random fields[67] (CRF) and a graph-based parsing model. The reason for using a treatment different from the conventional attention mechanisms is the way the selection of input elements are given. In the attention mechanisms the eq. 2.22 consider each input element, but if they are replaced by all possible structural dependencies a computation intractability emerges. In order to model explicitly the dependencies between elements in a sequence, such as those involved in the mentioned structural relationships, graphical models are employed in Kim's proposal. Specifically, CRF plays a key role in the 'Structured Attention Networks' given that conditional independence assumptions aren't considered as other graphical models (for example Hidden Markov Models). Experiments were done in three NLP tasks: Natural Language Inference, Question Answering, and NMT. The obtained results indicated equal or better performances than SOTA models at that time. Another modification to the attention model of Bahdanau is the work by Tu *et al.*[96] They state that the conventional attention architecture has the problem of over and under translation due to the lack of coverage. The 'coverage' term is employed in SMT for taking into account which source words have been translated. Thus, Tu's proposal named coverage-based NMT is intended to consider the mentioned counting history information. The included coverage vector $C_{i,j}$ at time step i that depends on the hidden state h_j is obtained via:

$$c_{i,j} = f(C_{i-1,j}, \alpha_{i,j}, h_j, t_{i-1}), \tag{2.29}$$

where f is a RNN (GRU) and t_{i-1} is the vector of information about past translation and α is similar to eq 2.23. Specifically the eq. 2.25 is modified resulting as :

$$a(t_{i-1}, h_j, C_{i-1,j}) = v_a^\top tanh(W_a t_{i-1} + U_a h_j + v_a C_{i-1,j}), \tag{2.30}$$

where the variables W_a, U_a, and V_a are learned parameters. The NMT is then coupled with the coverage and trained end-to-end. They carried on experiments on Chinese-English translation resulting in an outperformance over the Bahdanau's model.

Data-to-text

Besides translation, other NLP problems where attention mechanisms play an important role is in generating text from data. Data-to-text generation tasks suffer the alignment problem, that is, to specify which parts of input data correspond to the generated text. One attempt to solve this problem is with a probabilistic framework treating the alignment as a latent variable.[97,98] Nevertheless, the alignment problem can be overpassed with attention mechanisms achieving vector representations for different parts of the data. Some types of data-to-text problems are where a description must be provided from data. For example, the selection of relevant parts of event records for an appropriate describing sentence.

A work that addresses this particular problem using attention methods is Mei *et al.*[99] In this case, it is important to perform a pre-selection given the number of non-important information within the source. Thus, the main objective is to select a subset of features from encoded data to perform alignment and generate the describing text. In their work they propose a coarse-to-fine aligner as a modification of the conventional attention model. Basically, the coarse-to-fine approach consists of a pre-selector and selection refinement of the hidden encoder states. As a first step, the concatenation of the one-hot encoded input with its own hidden state is assigned to the variable m. Then, a pre-selector assigns to each record r a probability p_j of being selected. This p_j with the aligner weights w are then re-weighted by the refiner resulting in the selection decision z. Figure 2.11 outlines the coarse-to-fine

approach components.

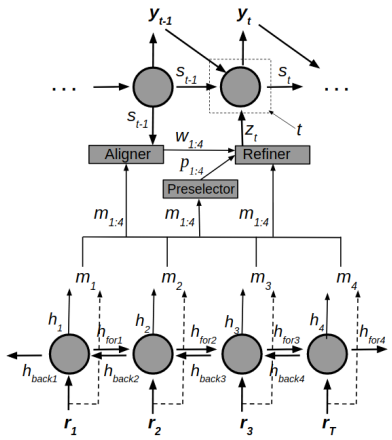


Figure 2.11: Representation of the architecture model of Mei *et al.*[99] with an encoder, decoder, and the coarse-to-fine aligner.

Another well-known problem in the computer vision area is image caption generation where the objective consists of providing a scene description for a particular image. Here, the RGB channels continuous values from the input are commonly subjected to pooling operations after convolution (see Convolutional Neural Networks , DL section) to pass to an initial hidden state of a RNN decoder.[100] Thus, if image caption generation task is conceived as an image 'translation' it can be well suited to the encoder-decoder framework. Following the attempts to better generate sentences attending to elements with different importances, we could mention the work of Xu *et al.* This approach proposes a scheme for obtaining the context vector that conditions the RNN (LSTM) decoding. From the proposal, two ways for the attention model emerges: stochastic 'hard' and deterministic 'soft.' The general framework for both methods employs a location variable, s_t, that represents where the model focuses on. The 'hard' attention mechanism treats the attention locations s_t as latent variables and stochastically estimates the corresponding weights for the annotation vectors in the context vector. A disadvantage is that stochastic attention requires sampling the attention location s_t each time. Thus, one could take the probability-weighted average of the attention locations yielding to annotation vec-

tors with its corresponding weight. This is called 'soft' attention and is equivalent to the Bahdanau's attention mechanism (eq. 2.23). An advantage of the 'soft' model is that it is differentiable, but it is expensive with large input sequences. Results show an improvement of the SOTA performance with the proposed attention model using BLEU[78] and METEOR[101] metrics. In addition, as a result of the attention architectures, the Xu's scheme provides a way to analyze what and where the model is paying attention to. Nearly at the same time of the attention paper of Bahdanau *et al.*, Chorowski *et al.* published[102] an application for another NLP important task, namely, speech recognition. Speech recognition is similar to translation in the sense that an initial sequence of data must be the starting point to an output sequence. However, in speech recognition the source sequence is about a thousand of frames coming from signal processing. Nevertheless, the encoder scheme consisting of a certain number of the processed elements could also be harnessed. It is important to note that traditional attention schemes have the issue of equally scoring identical annotations h (eq. 2.22). So, this method is extended with a location-based framework which attends only to previous attention weights α_{i-1}. This is done by extracting k vectors $f_{i,j} \in \Re^k$ for every position j of the previous weight α_{i-1} by convolving it with a matrix $\mathbf{F}$. The scoring (eq. 2.24 and eq. 2.25) is the extended as:

$$e_{i,j} = v_a^\top tanh(W_a s_{i-1} + U_a h_j + v f_{i,j}). \tag{2.31}$$

Given the fact that attending to all features could introduce noise information unnecessarily, Chorowski *et al.* propose sharpening and smoothing variations to the scoring function for addressing the noise issues. Broadly, the inverse temperature $\beta > 1$ is applied in the softmax function for the sharpening case, while for the smoothing variation, the terms in Eq. 2.23 are substituted by logistic sigmoids applied to the e variable (Eq. 2.24). The resultants equations are then:

$$a_{i,j} = \frac{exp(\beta e_{i,j})}{\Sigma_{j=1}^{L} exp(\beta e_{i,j})} \tag{2.32}$$

where L is the number of sequence representations. And,

$$a_{i,j} = \frac{exp(\sigma e_{i,j})}{\Sigma_{j=1}^{L} exp(\sigma e_{i,j})} \tag{2.33}$$

for the smoothing case. Finally, let's mention another data-to-text problem described as text-to-text generation. Within this category, summarization is one of the most representative objectives. Also, there are two types of summarization: extractive and abstractive. The first one is the production of a subset of the original sentences preserving the most important phrases. Abstractive summarization instead, contains words that are not included in the original source. This problem has a similarity with machine translation where attention mechanisms could have an important presence in an encoder-decoder structure. This is exemplified in the work[103] of Rush *et al.* 2015 called 'Attention-Based Summarization.' In their model, they coupled an attention encoder based on the Bahdanau's attention and a beam-search decoder. The results obtained are comparisons of different methodologies like greedy beam search and BoW (Bag of Words). Other related work on attention for summarization tasks is the publication of See at al.[104] They basically propose two main characteristics that extend attention models. First the inclusion of the coverage term, similar to Tu *et al.*[96] (see above) and the explicit selection of input elements via pointer networks.[105]

Self-attention

It is important to mention that the attention works presented up to here are related with text generation via a decoder part. Nevertheless, for NLP tasks that don't output sequences, but categories, attention could be applied within source elements. The general term for attention functions that are implemented between each of the different considered input elements is self-attention (also called $intra-attention$ or $inner-attention$). We can mention some schemes regarding self-attention applied in NLP tasks like text-entailment, sentiment analysis, and author profiling. Firstly, the Lin *et al.* model[106] that uses embedding, annotation, and hidden 2D matrices. These matrices result after considering n input elements segmented into r parts

where the selected partitions of the source attend each of the input elements. The hidden matrix H with dimensions n by $2u$ is multiplied by the annotation matrix A with dimensions r by n. The product then is the embedding matrix M that has r by $2u$ dimensions. The $2u$ term of the hidden matrix H results after the processing of a BiRNN with u hidden dimensions, similar to Bahnadau's proposal. In this architecture a penalization term is also included to diminish redundancy effects and the model is presumed to better disentangle the latent information from the input sentence. The results in 3 different tasks showed better performance than non-self-attentive methodologies. Second, a scheme similar to Lin's where dependencies between elements are taken into account is the work of Cheng *et al.*[107] They coupled the traditional Bahdanau's attention with an $intra-attention$ mechanism employing a memory network to preserve hidden states. Although the main contribution is to expand a LSTM with a memory network, representing relations between sequence elements via $intra-attention$ resulted in successful performance of language modeling, sentiment analysis and natural language inference.

Finally, we mention two other works that perform self-attention. The one by[108] Liu *et al.*, that extends the encoder with attention between elements for text-entailment and called their method $inner-attention$ and the Kim's 'Structured Attention Networks' model[95] mentioned above, that inherently performs self-attention for obtaining representations between input elements.

Others

A lot of attention and self-attention based algorithms were proposed and examples vary from generative models,[109] meta-learning,[110,111] Graph Attention models,[112] optimization,[105] among others. As a matter of fact, the reader should be aware that the computer vision domain also has its trajectory on attention models (see for example[113–116]) but those approaches for object recognition differ from the alignment motivation treated in most works cited here.

2.2.7 The Transformer

One of the leading proposals in Neural Machine Translation is the pure attention-based mechanisms implemented in the Transformer architecture.[36] This is the first translation method relying only on attention and self-attention getting rid of recurrence and convolutions. As self-attention could be assigned to tasks other than translation, the Transformer architecture was a breakthrough not only in translation tasks, but in different NLP challenges. The Figure 2.12 shows the transformer scheme based on the original reference. Although the model preserves the encoder-decoder framework, notice that there are no RNNs in the model.

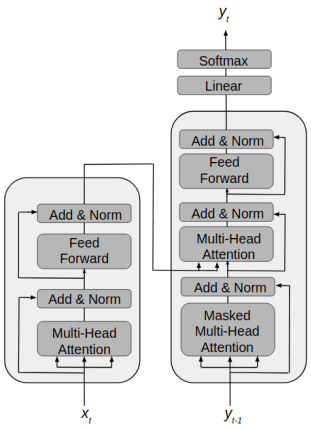


Figure 2.12: Transformer model. The left part is the encoder and the right part the decoder. The predictions are shifted in time steps t.

Attention is applied in different ways all along the architecture. The encoder and decoder are based on self-attention but a modification is required in the decoder to preserve the autoregressiveness. That is, the decoder prediction of symbols/words only from attending previous seen elements and masking future positions (with -inf values). In contrast to Bahdanau's attention, which uses a MLP to obtain the scoring function, this scheme performs a variation of the dot attention version (see above). In order to explain the dot attention in Vaswani's paper, it is convenient to remark some points regarding terminology and differences from the previous described schemes. First, the encoded representations of the input and outputs are

projected into three different vectors with their respective weight matrices. Now, given that attention and self-attention is performed between the source and target elements, a generalization to their representations is achieved by employing the 'query', 'key', and 'value' terms. 'Query' is the expected objective to attend and is compared with the actual objective represented as 'key'. The 'value' is simply added to the key for the numeric coherence, indeed this mixed entity is found sometimes as 'key-value.' The relationships of query and key-value vectors could be illustrated in Figure 2.13 (left).

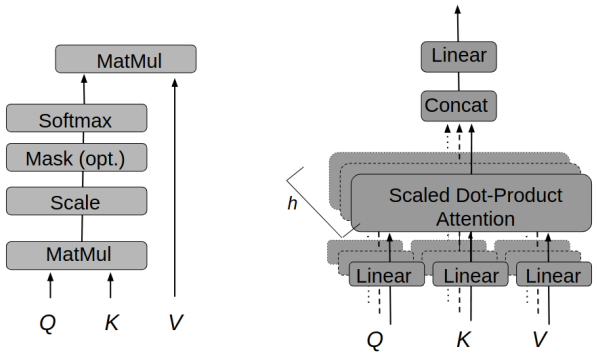


Figure 2.13: Representations of the scaled dot-product operations (left) and the corresponding introduction to the Multi-Head attention (right).

In the Bahdanau's model, the query would be the hidden state of the decoder and the key-values the annotation from the encoder. Now, in the Transformer model the dot attention function is modified with a scaling term $1/\sqrt{d_k}$ it is called scaled dot-product:

$$Attention(Q, K, V) = Softmax(\frac{QK^\top}{\sqrt{d_k}})V. \tag{2.34}$$

And the multihead term:

$$MultiHead(Q, K, V) = Concat(head1, ..., head_h)W^0. \tag{2.35}$$

where

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V), \tag{2.36}$$

and W_i^Q, W_i^K, and W_i^V are projector matrices. All attention functions are performed with several attention layers running in parallel. The outputs of those layers are concatenated in a component called Multi-Head layer and Masked Multi-Head for the decoder. The Multi-Head attention layer is depicted in the scheme of Figure 2.13 (right). It is important to mention that the Multi-Head layer that has entries from the encoder, called 'encoder-decoder layer,' the queries come from the previous decoder layer and the keys and values come from the output of the encoder. This allows a mechanism similar to the Bahdanau's traditional attention where every position in the decoder attends over all positions in the input sequence. The authors propose h heads with the same number of dimensions $d = 64 * h$. This relatively low number of dim allows reducing the computational cost because of parallelization. The authors propose $h = 8$ and 6 stacked identical layers for the encoder and 6 for the decoder. Another important part is the residual subarchitecture. Also, position encoding is added in both encoder and decoder to simply take into account the order of the sequence. The BLEU scores in English-German translation achieved two points over the SOTA at the moment. The Transformer authors also explained motivations for relying only on self-attention. These followed the objectives of increasing parallelization, diminishing the complexity, and treating long-sequences. The first point is achieved mainly by multihead attention that is computed parallely as depicted above. For the complexity case, if $n < d$ self-attention layers are less complex and faster than recurrent layers. And the long-sequences were proposed to be addressed by restricting the input to only a neighborhood of size r. After the Transformer publication, several language models that rely on attention and self-attention have been proposed for translation and other NLP problems. Most of them have been developed taking the Transformer architecture as a base and have been improved and/or adapted for specific tasks and issues. For example, one

architecture that modifies the autoregressiveness (probability of the n word conditioned to the $n-1$ words) scheme of the Transformer decoder in order to consider bidirectional context is BERT (Bidirectional Encoder Representations from Transformers).[38] The main importance to consider bidirectionality is that in tasks other than translation, such as Q & A, the context and information after the current position play an important role. Furthermore, besides proposing a modified Transformer architecture, the publication of BERT considers providing a pre-trained model to perform fine-tuning in posterior specific NLP tasks. Another architecture from 2019 is the XLNet[117] that outperforms BERT in several NLP tasks. Mainly, this model returns to the autoregressive scheme implemented in the original Transformer, but allowing bidirectionality with permutations of the training sentences. This permutation proposal alleviates the fact that BERT has a discrepancy in pre-training and fine-tuning because of the way masking is performed, that is, only the masking is found in the pre-training step and not in the fine-tuning step. Other important models have been published with different modifications such as size of the architecture and number of parameters involved as well as memory resources and parallelization management. In fact, given the similarities with the Transformer, several models have been grouped into the so-called 'transformers family.' Examples are: TransformerXL,[118] GPT-2,[57] T5,[119] ElMo,[39] Universal Transformers,[120] TinyBERT,[121] Reformer,[122] RoBERTa,[123] ALBERT,[124] distilBERT,[125] ELECTRA,[126] etc.

2.2.8 Set-input problems

Lot of prediction tasks like regression or classification are implemented via algorithms that process ordered data in fixed-dimensions. However, there are cases where the input to some model must be independent of the order and size of the inputs. For example, a typical situation is the task of sorting numbers, where the ordering of the input elements is not clear. One of the first works that addressed this and related problems is the one by Vinyals, *et al.* In their study, they proposed a model called Set2Set,[127] to solve the mentioned sorting case as well as combinato-

rial and language modeling examples. The main intention was to experiment with different input/output arrangements to analyze the relevance of different orders. Problems where conditions of permutation invariance and independence of the set size are satisfied are known as $set - input$ problems.[23,25] That is, the output of the model must be equal independently of the order and size of the input. The general awareness of different treatments to these cases is relatively recent[24] and we can name a few examples from distinct applications. Some examples within the area of supervised learning that fail into the $set - input$ category are multiple instance learning where an input set of instances is processed in order to predict a label. Another case is 3D shape recognition, where the model preserves the permutation invariant property.[128] Another example is few shot learning where the task is to diminish the number of examples to classify an image. Two works that address this case are the ones from Snell[129] *et al.* and Finn[130] *et al.* where the main idea is to use a support set of images to perform predictions with query images. In the search of molecular properties, the treatment and representation involving graphs is also permutation invariant.[131] Moreover, $set - input$ problems are not restricted to supervised learning. The problem is to learn a representation for the set in order to select from a large collection of sets, sets similar to query sets, for example, in image tagging and concept retrieval.

DeepSets In 2017, Zaheer *et al.* performed an analysis[24] for providing theoretically well-founded models for $set - input$ problems. In their work, they defined permutation invariant and equivariant functions and their characteristics for designing architectures that operate on sets. First, they established a theorem and its proof that specify the necessary and sufficient conditions for a function to be permutation invariant. This theorem states that a function f acting on sets must be permutation invariant to the order of the set,

$$f(x_1, ..., x_m) = f(x_{\pi(1)}, ..., x_{\pi(m)}), \tag{2.37}$$

for any permutation π, if and only if it can be decomposed in the form $\rho(\Sigma_{x \in X} \phi(x))$,

for suitable transformations ϕ and ρ. The second point in the NN designing goal for sets is the permutation equivariance preservation in the functions. In this case, equivariance is the property of a function that upon permutation of the input instances, permutes the output values:

$$f([x_{\pi(1)}, ..., x_{\pi(m)}]) = [f_{\pi(1)}(x), ..., f_{\pi(m)}(x)]. \tag{2.38}$$

The specification of invariance and equivariance properties allows the implementation of architectures for permutation invariant sets called *DeepSets*. From these architectures two categories emerge: the invariant and equivariant models. The invariant version resultant model is depicted in Figure 2.14.

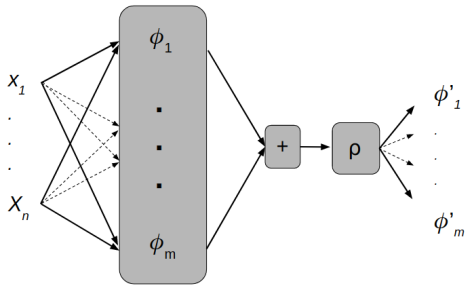


Figure 2.14: Representation of the permutation invariant *DeepSets* model.

Briefly described, for preserving the permutation invariance, each set element is firstly transformed to any continuous representation ϕ. Then, equals ϕs for the different elements are added up and the result is subjected to a nonlinearity ρ such as a sigmoid. This results in principle, in an universal approximator. That is, it is assumed that this network can approximate any continuous mapping. The architecture also considers the possibility of conditioning the mapping to an additional information z. This arrangement can also be considered as an encoder-decoder, where the encoder would be the transformation ϕ acting independently on each set element and the decoder the aggregation and the nonlinearity ρ. The second resultant model, the equivariant version, implies the results of the permutation equivariance necessary and sufficient conditions. The authors states in a lemma that if $f(\Theta)$ is

restricted to a NN layer such as $f_\Theta(x) = \sigma(\Theta(x))$ where $\Theta \in \Re^{MxM}$, it is permutation equivariant if and only if all the off-diagonal elements of Θ are tied together and all the diagonal elements are equal as well, i.e.,

$$\Theta = \lambda\mathbf{I} + \gamma(\mathbf{1}\mathbf{1}^\top) \; for \; \lambda, \; \gamma \in \Re, \; \mathbf{1} = [1, ..., 1]^\top \in \Re^{MxM}, \; and \; \mathbf{I} \in \Re^{MxM}. \quad (2.39)$$

This represents a weighted combination of its input $\mathbf{I}x$ and the sum of input values $(\mathbf{1}\mathbf{1}^\top)x$. For example, the weights that multiply an input x_i are $(\lambda + \gamma)$ for the mapping to the output y_i and λ for the other mappings. This is schematized in Figure 2.15.

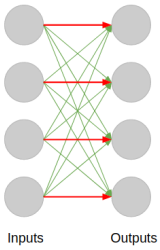


Figure 2.15: Visualization of weights sharing in a permutation equivariant NN layer.

The next step is then a nonlinear activation. The equivariant model allows stacking of nn layers without losing the equivariant and invariant properties. This then enables a general treatment for sets on supervised and unsupervised settings. The results of their experiments on anomaly detection and clustering show that in fact the architecture proposed works better than deep networks which are not backed by theory. Also unsupervised tasks were tested on concept-set retrieval and image tagging.

2.2.9 Set-Transformer

After the Zaheer proposal, J. Lee *et al.* conceived an architecture[23] for $set - input$ problems but extended the model with self-attention mechanisms. The main reason for including self-attention was to take into account interactions between the set

elements that could be important for the task in hand. That information regarding dependence between set elements was not considered in previous works on sets.[24,132] In those models every element is processed independently and there is a lack of information regarding interactions between elements. With this in hand, the J. Lee model called *Set Transformer*, is an architecture for $set - input$ problems extended with self attention mechanisms for problems where interactions between elements could play an important role. One example is the case of amortized clustering, where interactions between the centers of the clusters are relevant. As well as the *DeepSets* and Edwards′ work, the *Set Transformer* model, has also the properties of being permutation invariance and equivariance. In the model, the attention mechanism is employed in several ways, mostly in the Multi-Head adaptation from the Vaswani[36] *et al.* method. The main components consist of Multi-Head Attention Blocks (MAB), and Set Attention Block (SAB). The MAB is an adaptation of the encoder block of the Transformer[36] without positional encoding. It processes each instance with a row-wise feedforward (rFF) layer and employs a layer normalization (LayerNorm):

$$MAB(X, Y) = LayerNorm(H + rFF(H)), \tag{2.40}$$

where X and Y represent two sets of d-dimensional vectors and n points, and ω are learnable parameters, and

$$H = LayerNorm(X + Multi - Head(X, Y, Y; \omega)). \tag{2.41}$$

The SAB then is the application of the MAB between set elements resulting in a set of equal size.

$$SAB(X) := MAB(X, X). \tag{2.42}$$

An important property of the SABs is that they can be stacked to approximate

higher order interactions as a consequence of containing information of higher order interactions. Indeed, another contribution from the *Set Transformer* is the induced SAB (ISAB) that diminishes the computational complexity in the Set Attention Block. The ISAB is basically a MAB applied over inducing points I that is a set with m points as learnable parameters. The overall architecture is constructed then with an encoder and decoder, being the first one the mapping $X \to Z \in \Re^{nxd}$:

$$Encoder(X) = SAB(SAB(X)),\ or \tag{2.43}$$

$$Encoder(X) = ISAB(ISAB(X)). \tag{2.44}$$

That is, the encoder is a stack or SABs or ISABs and an important point is that the complexity for l stacks of SABs and ISABs are $O(l * n2)$ and $O(l * n * m)$, respectively. The decoder is then:

$$Decoder(Z; \lambda) = rFF(SAB(PMA_K(Z))), \tag{2.45}$$

where

$$PMA_K(Z) = MAB(S, rFF(Z)), \tag{2.46}$$

and the set of k vectors are employed instead of the n elements of the representation z. Figure 2.16 shows the general scheme of the *Set Transformer* with the encoder-decoder components and the multihead attention variations.

The PMA block is one of the main contributions of the model. This is motivated by the amortized clustering where the interactions of k vectors for k clusters are represented in a beneficial way. The *Set Transformer* architecture preserves the permutation equivariance and invariance properties as a result of the permutation equivariant of the SAB and the permutation invariance property of the PMA. In addition, the authors formally proved that the *Set Transformer* is a universal approximator of permutation invariant functions. The *Set Transformer* was tested

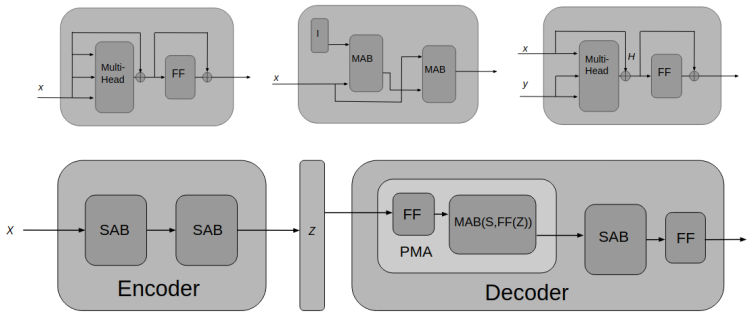


Figure 2.16: Schemes from the *Set Transformer* model. From top left to top right are shown the SAB, the ISAB, and the MAB blocks. The detailed encoder-decoder including the SAB and MAB components is shown at the bottom of the figure.

with the amortized clustering task and others such as character recognition and anomaly detection resulting in an outperformance of other comparative methods.

Chapter 3

Machine Learning and Nuclear Magnetic Resonance for Natural Products

3.1 Introduction

Natural products (NP) are secondary metabolites usually obtained from plants, animals, microorganisms, marine species, or other natural sources.[21,22,133] Partial characterization of these molecules is carried out using a priori knowledge, that is, some biological species contain molecules with structures closely related to those found and characterized in similar organisms.[134] However, molecules widely distributed in specific living organisms could be present in entirely different species. Cholesterol is an example because, in principle, it only exists in animals, but it has also been detected in plants.[134,135] Given the high probability that an uncharacterized NP adopts a structure relating to a particular family, the first step for its structural elucidation could be the prediction of the NP class. There are currently approaches, like metabolomics, which recognize NP in comparative studies and employ statistical methods.[136] Nevertheless, complementary information of the molecular species is required to achieve an appropriate determination. Most of this information is

based on Infrared Spectroscopy (IR), Mass Specrometry (MS), and Nuclear Magnetic Resonance (NMR) spectroscopies, among others.[137–139] Another well-known technique to identify NPs consists of NMR spectra matching of previously known compounds.[140] When a database is not available for the structure-spectra matching or de novo elucidation of structures, another kind of assistance is inevitably required.[1] This assistance could be provided by CASE (Computer-Assisted Structure Elucidation) systems that have been developed in order to help people to confirm or to declinate a proposed elucidated structure. Most of them consist mainly of matching the peaks of the experiments with those of a certain molecule included in a database. In addition, some other algorithms to jointly perform atom pair matching and fine details arrangements are included. Of course, some CASEs and their databases are not freely available and this could restrict the development of related and improved software as well as potential contributions to the CASEs themselves. An example is the ACD/StrucEluc commercial software.[7] This tool is based on a deterministic approach to solve the structure where one of the principal components is the molecular connectivity diagram, obtained from two-dimensional NMR data. Regarding ^{13}C NMR shift values, more than 200,000 structures are contained in its database. More recently, in 2013, the Nuzillard group published two open-source softwares to elucidate organic structures, LSD[141] and CCASA.[5] Both of them rely on two-dimensional NMR data but the first is an attempt to achieve better results when fuzzy two-dimensional data is available and the corresponding interpretation is complicated. These algorithms propose and validate structures employing database substructure matching. Following the open-source philosophy[5,141] and mainly, taking into account the modern machine-learning friendly tools in constant development, we explore the prediction capabilities of different algorithms depending only on the one-dimensional ^{13}C NMR data and without database peak matching. It should be noted that a trade-off between the spectra available and the elucidated structure completeness emerges naturally. Thus, considering that in many cases spectroscopic information is available, herein we propose a tool to predict the pres-

ence or absence of common NP classes starting from the ^{13}C spectroscopic data. It needs to be pointed out that the results of algorithm design and implementations within an open-source community play the most important role in the present study and possible future works. Moreover, this article gets its results from a purely free database and leads to motivation in the community not to be limited by commercial data and packages. In this work, employing Machine Learning (ML) algorithms and a ^{13}C NMR database, we investigate the capability of prediction of eight common natural product classes: Sesquiterpenoids, diterpenoids, triterpenoids, lignans, steroids, chromans, flavonoids, and alkaloids. The prediction of glycosides is also included given that a carbohydrate-type structure is often present in NP.

3.2 Methods

Machine learning has recently experienced a considerable rise due to the advances in algorithmic and computational capabilities. It allows describing data within mathematical representations that consider non-intuitive forms for pattern recognition. i. e., relating multiple dimensions and projections. Thus, problems like regression, classification, clustering, and some others, can be attacked in a variety of ways, from the simplest to the most complex ones. There are several areas where algorithms are currently outperforming human experts.[6,10] Within the arena of properties prediction in molecules, considerable effort has been devoted to physicochemical properties. Regarding the prediction of ^{13}C NMR employing ML, we should mention two significant contributions. The first is that by Meiler and co-workers, who trained a neural network in order to predict the spectra of organic molecules.[142] The other one is by Steinbeck that predicted the NMR spectra of metabolites using decision trees, random forests, and support vector machines.[143] The inverse problem, given specific properties, predicts the most probable structures, is mostly explored in the materials and drug design areas.[16,144] In the structure prediction from NMR data tasks, we should mention several papers. The Genius code by Meiler and Will mixes NMR prediction of a trained network with a genetic algorithm in order to predict

the complete structure.[145] Another one is that of Dzeroski et al., who implemented an algorithm to predict diterpene skeletons from a dataset of 1,500 molecules using ^{13}C NMR info and ML algorithms.[146] The terpene skeletons prediction using ^{13}C NMR is discussed by Ferreira et al,[147] concluding that peak matching to a database played an important role. Pereira[148] reported a method to determine trisaccharides anomeric conformation and linkage information. Another remarkable contribution is SISTEMAT by Emerenciano et al.[149] This tool predicts NP skeletons and has now coupled to the LSD software.[150] Recently, Boiteau et al. published an approach to predict metabolites from MS and 2D NMR correlation spectroscopies.[151] Nevertheless, this method relies on the prediction of 2D spectra from commercial packages to match the possible structures iteratively. Finally, Harn et al. developed an NP structure predictor that uses MS using combinatorial and not ML algorithms.[152] Clearly, there is an under-exploration of the structural elucidation just from ^{13}C NMR, information that is free from databases and commercial software for spectroscopic properties prediction. It is worth to remind that data availability and quality play essential roles in prediction problems. It has been stated that there is a lack of consensus in the way chemical shifts are published limiting their data mining.[153] Herein, we employed the Naproc13 database.[154] This compilation of ^{13}C NMR spectra from natural products is an extraordinary effort made by the University of Salamanca. It has entries for more than 20,000 structures, and besides the NMRShiftDB2 database,[155] so it is the most complete ^{13}C NMR database publicly available. In summary, we are exploring the possibilities of prediction of NP families with the free ^{13}C NMR data availability and ML computational tools paving the way for future related works.

3.3 Dataset

Naproc13 dataset was subjected to a web scraping bot in order to download each spectrum. The dataset consists of one file per structure with the ^{13}C NMR spectra, as well as the class information. We selected eight NP classes simply because these

classes contain the highest numbers of samples in the Naproc13 dataset. After the removal of incomplete and duplicated spectra (stereoisomers were also removed), the number of structures is 18,740 and the total number of shifts is 457,047. Another restriction was that the number of signals (shifts) should be the same as the number of carbon atoms. The complete clean dataset (set **1**) was then split into eight subsets for each of the NP class classifications. Each subset is composed of all the structures from the set **1** that have the same or fewer carbon atoms than the maximum number found in the NP class in question. For example, the maximum number of carbon atoms in a family classified as a steroid is 57. Then, the subset for the steroid classification contains all the molecules having at most 57 carbon atoms, which are 18,529, and so on for each NP class structure. This allows the classification of a molecule even if the expected number of carbon atoms differs drastically from experimental values. For example, a triterpenoid that could have approximately 30 carbon atoms, the maximum number found is 79. The spectra were arranged in an arbitrarily ascending order and padded with out of range values (-999), where the number of carbon atoms is less than the maximum number of carbon atoms. We should remark that this representation is delegated to the classifier as well as the robustness to noisy data. The final *csv* (Comma Separated Value, a commonly used data file) file for each subset contains N rows of spectra and M + 1 columns. N is the number of samples and M the maximum number of carbon atoms for the NP class. The last column indicates a 1 if the spectra correspond to the NP class and 0 otherwise. See the Supporting Information (Tables S1 and S2) for an example of the raw files to *csv* file transition.

Figure 3.1 shows the distribution of frequencies for the ^{13}C NMR chemical shifts in set **1**. Note three remarkable areas in the shift values corresponding to the saturated carbons (15-60 ppm), ethers and alcohols (50-80 ppm), and aromatic carbons (100-150 ppm). The last follows the abundance of the chemical substructures in NPs. For instance, terpenoid families, which are constituted by isoprene units, are known to be the most abundant compounds in nature. Another kind of chemi-

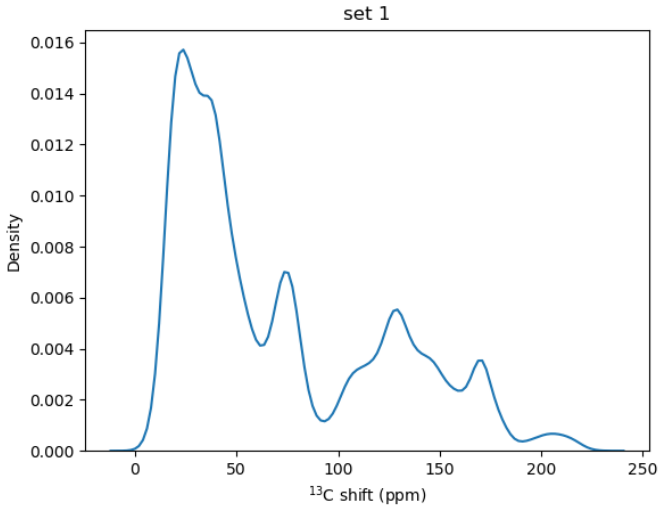


Figure 3.1: Frequency distribution plot for set **1**.

cal moieties frequently found is the aromatic rings involved in families like lignans, chromans, flavonoids, and alkaloids. Finally, the area corresponding to the oxygen-bonded carbons coincides with functional groups like ethers and alcohols detected in glycosides (see Figure 3.2). Moreover, to compare the frequencies with the NMRShiftDB2 dataset, a plot of frequency distributions is depicted in Figure 3.3. So, in contrast with set **1** from the Naproc13 database, the NMRShiftDB2 set contains a higher balance between the saturated and unsaturated hydrocarbons. Furthermore, the peak corresponding to the oxygen-bonded carbons is not presented as expected, given that the NMRShiftDB2 set focuses on all types of organic molecules.

Figure 3.4 illustrates the different kinds of structures for the eight NP classes of interest, as well as a glycoside example. Notice that the selected steroid in this figure is also a glycoside given the carbohydrate-type structure coupled to the four-ring system. The structure marked as triterpenoid (bottom left) is also a glycoside.

3.4 Binary Classification

Our goal is to develop a model able to predict the presence or absence of a particular natural product family from the ^{13}C NMR spectra. In the machine learning

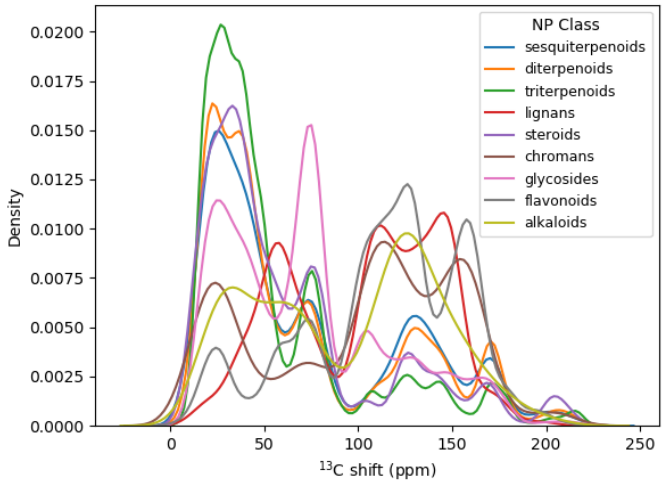


Figure 3.2: Frequency distribution plot for the nine natural product classes.

terminology, we are looking for a discriminant function mapping a set of continuous variables to a discrete variable belonging to the set $\{0, 1\}$. This is known as a binary classification problem. In our case, the continuous variable is the spectra, where 0 and 1 values correspond to the absence and presence of the family of interest, respectively. In order to select the appropriate algorithm for the task in question, let us use a logistic regression (Log Reg), which is a simple binary classifier used in this work as a base model for comparison. Three more complex classifiers were also employed, namely: Multilayer Perceptron (MLP), Support Vector Machine (SVM), and Extreme Gradient Boosting (XGBoost). Detailed description of these four methods can be found in the ML background section of Chapter 2. A grid search over the hyperparameters of the four classifiers was performed. In Table 3.1 the optimal hyperparameters found for each of the classifiers are shown.

Table 3.1: Hyperparameters selection for the grid search.

Classifier	**Log Reg**	**MLP**	**SVM**	**XGB**
Hyperparameters	Penalty terms: q (1 or 2), regularization term: λ	Number of hidden units, number of hidden layers, learning rate.	Gamma, C	Depth, number of estimators, learning rate.

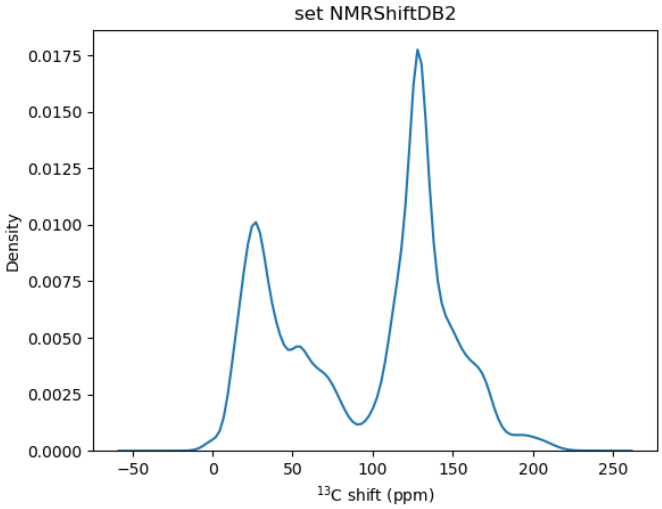


Figure 3.3: Frequency distribution plot for the set NMRShiftDB2.

The *Scikit-learn*[156] library was employed for the Log Reg, SVM, and MLP algorithm implementations. For the XGBoost classifier, the XGBoost *Python* library implementation was used. Cross-Validation 10-fold was performed in all cases. For the MLP classifier, the activation function employed was the sigmoid unit and the optimizer was Adam. The maximum number of iterations was 500. The Gaussian kernel was employed in the SVM classifier and the gamma parameter corresponds to the kernel coefficient. While the grid-search with the MLP took more than 20 hours, the XGBoost took just a couple of hours. Of course, the fastest algorithm is the Log Reg and the SVM lies in the 4-5 hours range. All of them with 16 cores on a Linux server and 16GB RAM. The variations in time are due to the different number of samples in the data sets. For more details on the hyperparameter grid specification, see Tables S3 and S4.

3.5 Metrics

We report the accuracy, precision, recall, f1-score, and AUROC (Area Under Receiver Operating Characteristic) metrics. Accuracy is simply the fraction of correct predictions. It is represented by the ratio (tp + tn)/(tp + fp + tn + fn), being tp,

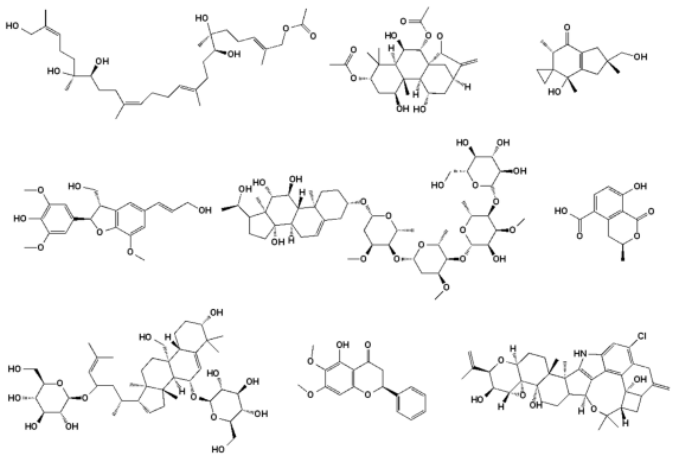


Figure 3.4: Structures of selected examples for the eight different NP classes and one glycoside. Top, from left to right: triterpenoid, diterpenoid, sesquiterpenoid. Middle, left to right: lignan, steroid, chroman. Bottom, left to right: glycoside, flavonoid, and alkaloid.

fp, tn, and fn, the number of true positives, false positives, true negatives, and false negatives, respectively.[157] However, note that the imbalanced sets could yield to a high accuracy score, but related mostly to one label classification. Thus, we must consider metrics other than accuracy. The precision and recall scores are defined as the ratio tp/(tp + fp) and tp/(tp + fn), respectively. Intuitively, while recall is the ability to find all the positive samples, precision is the ability not to classify all samples as positives. The f1-score is defined as the harmonic mean of precision and recall metrics:

$$f1 = 2 * (\frac{precision * recall}{precision + recall}). \tag{3.1}$$

Finally, ROC is the evaluation of tpr (true positive rate = tp/(tp + fn)) and fpr (false positive rate = fp/(fp + tn)) under different thresholds of the binary classifiers. AUROC then is the area under that curve that represents the closest values that maximize tpr and minimize fpr.

3.6 Results and Discussion

Table 3.2 shows the f1-score for the eight different natural product classes and glycosides for the four selected classifiers. Accuracy, precision, recall, AUROC, and specificity were obtained employing the classifiers trained with the hyperparameters that yielded to the best f1-scores (see Tables S5-S9).

Table 3.2: F1-scores for each classifier and family. Besides the 8 NP families, results for the glycoside fragment prediction are also included.

Class	Samples*	% Pos.	Log Reg	MLP	SVM	XGB
Sesquiterpenoid	18366	19.3	0.772	0.785	0.896	0.929
Diterpenoid	18468	34.0	0.748	0.820	0.919	0.948
Triterpenoid	18587	20.1	0.781	0.849	0.930	0.966
Lignan	18365	1.5	0.0436	0.010	0.780	0.895
Steroid	18529	3.8	0.334	0.085	0.830	0.906
Chroman	16228	1.5	0.0340	0.085	0.783	0.793
Glycoside	18584	8.4	0.318	0.434	0.782	0.830
Flavonoid	18098	7.9	0.694	0.740	0.880	0.915
Alkaloid	17376	0.79	0.0133	0.003	0.431	0.345

*Notice that the max number of samples (triterpenoid) differs from the total number of the dataset (18740) because there are structures that exceed the max number of carbon atoms (79 for triterpenoid) and do not belong to any of the considered families.

For almost all families, it is notorious the outperformance of XGBoost over the rest of the classifiers. There is also a consistency between XGBoost and SVM results. For Log Reg and MLPC, the best results are found in the terpenoid families, probably due to the positive samples percentage. Generally, an imbalance in the data is present and this probably yields poor results in some cases. This is illustrated in the alkaloid prediction performance that reaches an f1-score of 0.345 with the XGBoost classifier. So, in order to explore the effect of data imbalance, we performed experiments with the data set subjected to different resampling techniques.

3.7 Resampling the data set

Five methods for resampling were explored. Three methods for undersampling the majority class and two for oversampling the minority class. The employed undersam-

pling techniques were random deletion of the majority class, the Instance Hardness Threshold[158], and the NearMiss[159] removal. The Instance Hardness Threshold is a data analysis that pretends to identify and understand why some samples are difficult to classify. This algorithm assigns samples a measure regarding probabilities of misclassification. After the selection of difficult instances the removal of those examples are performed. In that sense, the analysis of difficult examples depends on the chosen classifier as well as on the cross validation. The Near Miss algorithm detects and removes positive samples near the negative zone. There are three versions of this technique depending on the quantity of minority samples that are near the majority samples. In this work we employed version 1 (NearMiss-1) that selects some minority samples near the majority zone. The oversampling was performed with the borderline-SMOTE[160] (borderline-Synthetic Minority Oversampling Technique) and ADASYN[161] (Adaptive Synthetic Sampling Approach) algorithms. Those methods are based on synthetic generated samples of the minority class with different strategies. The borderline-SMOTE is a variation of the SMOTE algorithm and consists in adding samples interpolated from the k-nearest neighbors but focusing on the border between classes. The ADASYN is a similar technique but it relies on synthetic samples that are hard to classify. Also, this method uses the density distribution to decide the number of synthetic minority samples generated. The imbalanced-learn[162] python package was employed for all algorithm implementations. For the Instance Hardness Threshold, the scores assigned to difficult samples were estimated with cross validation 10-fold and Gradient Boosting classifier with 200 estimators. In order to achieve the maximum f1-scores a grid search over the XGBoost hyperparameters was done in a Cross-Validation (CV) 10-fold fashion. For the undersampling algorithms the maximum f1-scores vs. % of positive samples are shown in Figure 3.5.

Note that for all families and methods, there is an increase in the scores while augmenting the positive percentage. For all cases, the best results are achieved at 40 and 50 %. The alkaloid family gives the lowest values possibly due to the number

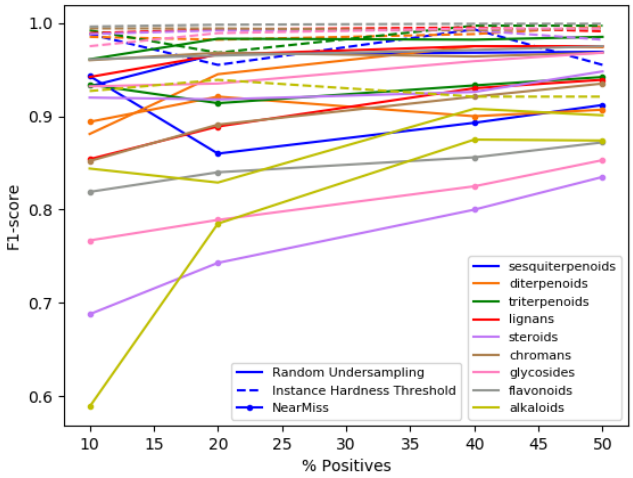


Figure 3.5: XGBoost f1-scores for different percentages of positive samples using different undersampling techniques.

of training samples (274 on the 50 % data set to 1370 in the 10 % data set). The Instance Hardness Threshold outperforms Random Undersampling and NearMiss-1, having this last one the poorest results.

The maximum f1-scores for the borderline-SMOTE and ADASYN resampled data sets are shown in Figure 3.6. Both methods yield practically similar results being the ADASYN method results slightly lower than the borderline-SMOTE results.

3.8 Test cases

Besides the predictions made for the validation set in the cross-validation step, we also performed predictions in recently reported natural products. We selected five examples per NP class reported from 2018-2019 that are not included in our data set. With 8 NP classes the total number of test cases is then 40. The references 73-105 report new natural product elucidation of families that we are interested in (see Tables S10 - S17). Six different XGBoost models were tested on these cases. Five models with the hyperparameters that achieved the highest f1-scores while training

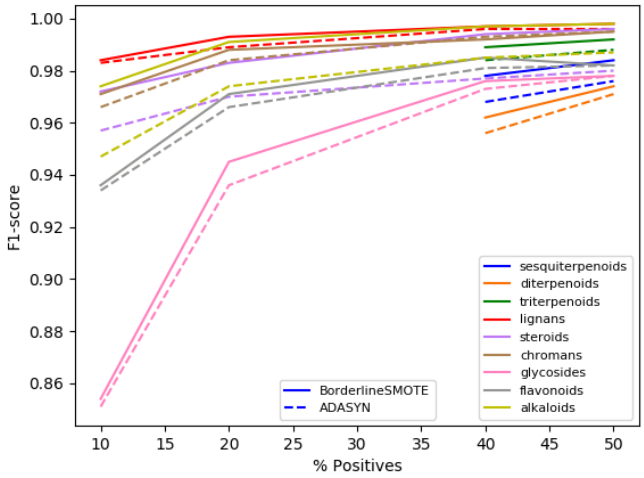


Figure 3.6: XGBoost f1-scores for different percentages of positive samples using different oversampling techniques. Notice that the sesquiterpenoids, diterpenoids, and triterpenoids sets have originally more or equal than 20 % of positive samples.

with the different resampling techniques, and a model that employed a different input representation. This representation for the ^{13}C NMR values is composed of histogram frequencies. That is, we divided the NMR space from 0 to the max ^{13}C NMR shift (284.9) and counted the frequencies by each bin. Each bin has a 1 ppm width. In general, this kind of representation is more expensive than the original described featurization. In order to select the best model, we wish to diminish the number of fps (false positives), while preserving a high accuracy. It has to be noticed that with the current threshold value (0.5) the presence of more than one NP class could be predicted. For example a model trained to predict a certain NP class could predict with a similar score the presence of another NP class than the one in turn. In that case we are looking for a model with high accuracy and low fps. Table 3.3 resumes the obtained accuracy and fps for models corresponding to different resampling techniques, as well as the model created with the histogram representation.

The reason Instance Hardness Threshold undersampling technique gives the highest accuracy and false positives is that this method is based mostly on removing

Table 3.3: Accuracies and the Number of fps for Models Trained with Different Resampling Techniques and a Model Trained with Different Encoding (Frequencies Histogram)

Model	Random	NearMiss-1	Inst. Hard. Thres.	ADASYN	borderline-SMOTE	frequencies histogram*	ADASYN+ random
Overall Accuracy**	0.80	0.83	0.98	0.73	0.63	0.65	0.88
FP	28	111	116	9	9	35	19

*Model with the hyperparameters that achieved the highest f1-scores trained on a balanced (50% of each category) set using Random Undersampling. **The fraction of correct predictions with respect to the total test cases (40).

overlapping points. That is, overlapping regions that are inherent to the ^{13}C NMR data from different substructures are oversimplified and lots of different NP classes are predicted as present. The final model is composed of the weights from the models trained with the Random Undersampling and the ADASYN oversampling methods. The weights of the first method applied to the triterpenoid, sesquiterpenoid, steroid, lignand, and alkaloid, and the weights of the second one applied to the rest of the NP classes. The probabilities that the spectra correspond to a particular family are shown in Table 3.4 for each test example (probabilities for all families are shown in Tables S19-S27, Appendix A.1). From the Tables S19-S27 one could notice that if the number of ^{13}C NMR shifts is greater than the maximum number of shifts for a NP class training data set, the probability for that NP class is 0.

The test cases predicted as negative (prob. < 0.5), as well as the chroman family, are discussed below. We also tested cases containing a glycoside fragment.[196] Those examples are from the steroid, flavonoid, lignan, sesquiterpenoid, and diterpenoid families. Table 3.5 shows the results for the presence of glycosides prediction with a model trained in the resampled set with Random Undersampling method.

3.9 The failure cases

Let us analyze the possible reasons why the XGBoost predictor fails in particular cases (Figure 3.7 and Table 3.4). The structural prediction of the diterpenoid, D-2 (Table S11) fails (Prob = 0.027) probably due to the similarity of the sample with a triterpenoid. This diterpenoid has a long aliphatic chain causing certain similarities

Table 3.4: Final model probabilities prediction for the test cases.

Sesquiterpenoid			Diterpenoid			Triterpenoid			Steroid			Flavonoid			Lignan			Alkaloid	
Comp.	**Ref.**	**Prob.**	**Comp.**	**Ref.**	**Prob.**	**Comp.**	**Ref.**	**Prob.**	**Comp.**	**Ref.**	**Prob.**	**Comp.**	**Ref.**	**Prob.**	**Comp.**	**Ref.**	**Prob.**	**Comp.**	**Ref.**	**Prob.**
S-1	163	0.974	**D-1**	164	0.999	**T-1**	165	0.998	**St-1**	166	0.953	**F-1**	167	0.999	**L-1**	168	0.967	**A-1**	169	0.954
S-2	170	0.999	**D-2**	171	0.027	**T-2**	172	0.999	**St-2**	173	0.998	**F-2**	172	0.999	**L-2**	174	0.978	**A-2**	175	0.965
S-3	176	0.999	**D-3**	171	0.996	**T-3**	177	0.999	**St-3**	178	0.999	**F-3**	179	0.556	**L-3**	180	0.987	**A-3**	181	0.879
S-4	182	0.986	**D-4**	183	0.998	**T-4**	184	0.995	**St-4**	185	0.998	**F-4**	186	0.999	**L-4**	187	0.999	**A-4**	188	0.320
S-5	189	0.999	**D-5**	190	0.997	**T-5**	191	0.999	**St-5**	192	0.057	**F-5**	193	0.668	**L-5**	194	0.002	**A-5**	195	0.992

Table 3.5: Glycoside prediction probablities.

Ref.	Prob.
166	0.989
172	0.977
168	0.995
163	0.994
190	0.983

to a triterpenoid. This is also apparent from the frequency distribution plot (Figure 3.2), where diterpenoids show a lower frequency in the aliphatic NMR area than the triterpenoids.

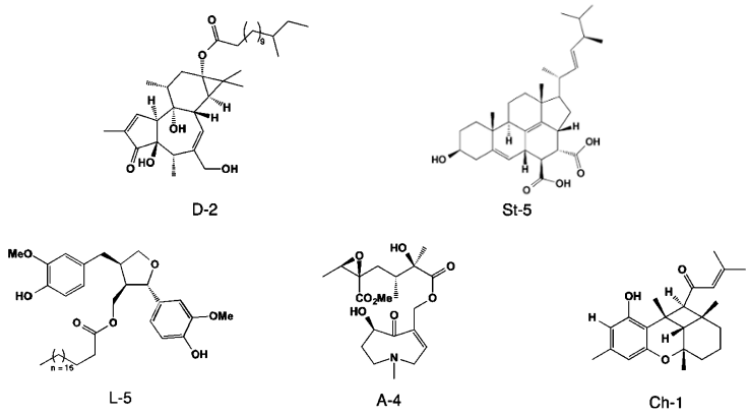


Figure 3.7: Structures of the failure cases.

For steroid St-5 (Prob= 0.057, Table S13), the model predicts a non-steroid structure. Notice that this novel 6/6/6/6/5 skeleton is not contained in our data set. Thus, the model fails in recognizing this structure as a steroid. The lignan failure case (L-5, Table S15) is due to the 16 carbon atoms aliphatic side chain (Prob = 0.002). This kind of fragment is not included in our data set. As with the diterpenoid case, St-5 and L-5 are predicted as triterpenoid with a probability value of 0.992 (Table S22) and 0.655 (Table S24), respectively. Given that the alkaloid classes obtained the lowest probability values, we decided to test more examples to find a failure example (prob. < 0.5). Compound A-4 (Table S16) yields a probability value of 0.320. This system does not contain a heterocycle, that is common in the alkaloid data set samples. Furthermore, for A-1 and A-2,

Table 3.6: Chroman predictions with non-coumarin data (left) and coumarin data (right).

Comp.	Ref.	Prob.	Comp.	Ref.	Prob.
Ch-1	[198]	0.001	**Co-1**	[199]	0.930
Ch-2	[200]	0.951	**Co-2**	[201]	0.986
Ch-3	[200]	0.844	**Co-3**	[202]	0.998
Ch-4	[200]	0.567	**Co-4**	[203]	0.995
Ch-5	[204]	0.000	**Co-5**	[205]	0.858

there are slightly higher probabilities (0.992 and 0.974, respectively) if they are assigned as lignans (see Table S25). This is presumably due to the small number of samples is not enough to cover the alkaloid chemical space. The case of chromans is a different story. Table 3.6 shows the probabilities predicted by the model. The reason for these failures is that the chemical space covered by chromans is also a subset of other NP classes. In other words, many examples that are not classified as chromans also contain benzopyran or benzodihydropyran fragments. For example, Ch-1 is predicted as diterpenoid with a probability value of 0.965 (Table S26). Thus, a proper prediction of chromans is a challenging problem mostly because of the intersection of the chroman space with other non-chroman classes. Now, if we restrict the classification to the keto substituted derivatives of the benzopyran fragments, which form the coumarins subclass,[197] the prediction is improved. Table 3.6 shows probabilities for the chroman class in new coumarins reported. This is in accordance with the chromans data set where more than 75 % of the samples belong to the coumarins subclass.

3.10 Multiclassification

The binary classifiers proposed in this section belong to an scheme called one-vs-rest. It consists of learning a decision boundary separating the positive class from the rest of the classes. However, linear decision boundaries for more than two classes yield to ambiguous regions where it is difficult to predict the correct class. Being aware of the possible ambiguity regions learned by the linear weak classifiers in the

XGBoost model, another classification method is investigated. This scheme is called multiclassification and the main differnce regards the loss function to be minimized. In the one-vs-rest scheme, the probability of the class presence represented with the sigmoid function for each classifier, allows the presence of more than one class (See SI Tables S19-S26). In the multiclassification scheme, instead, the softmax function represents the normalized probabilities of each class presence. This results in not having ambiguous regions in the classes space.

A hyperparameter search was performed in a similar way to the binary classifiers. The resulting best hyperparameters are learning rate = 0.35, max depth = 7, and number of estimators = 240. Mean f1 scores from CV-10 fold are shown in Table 3.7 as well as the results from the Table 3.2 corresponding to the binary classifiers using the one-vs-rest scheme.

Table 3.7: F1-scores for the 8 NP classes predicted in the multiclassification and binary classification models.

Class	**Multiclass**	**Binary**
	XGB	**XGB**
Sesquiterpenoid	0.945	0.929
Diterpenoid	0.959	0.948
Triterpenoid	0.967	0.966
Lignan	0.897	0.895
Steroid	0.915	0.906
Chroman	0.844	0.793
Flavonoid	0.958	0.915
Alkaloid	0.455	0.345

It is notorious the higher f1-score values than the ones obtained with the one-vs-rest scheme, mainly in the chroman, flavonoid, and alkaloid classes.

The trained model was then used to predict the test cases and the probability predictions are shown in Table 3.8. The predicted NP classes are almost similar to the binary classifier selected (Table 3.4). It is mainly for the alkaloid case that the predicted results differ, but also for one steroid case. For the alkaloid case, it is clear that the high imbalance provokes a poor predictive performance. The overall accuracy of 77% (27/35*100) is then lower than the 88% from the selected binary classifier model.

Table 3.8: Multiclass probabilities prediction for the test cases.

Sesquiterpenoid			Diterpenoid			Triterpenoid			Steroid			Flavonoid			Lignan			Alkaloid	
Comp.	**Ref.**	**Prob.**	**Comp.**	**Ref.**	**Prob.**	**Comp.**	**Ref.**	**Prob.**	**Comp.**	**Ref.**	**Prob.**	**Comp.**	**Ref.**	**Prob.**	**Comp.**	**Ref.**	**Prob.**	**Comp.**	**Ref.**	**Prob.**
S-1	163	0.999	**D-1**	164	0.999	**T-1**	165	0.999	**St-1**	166	0.000	**F-1**	167	0.999	**L-1**	168	0.999	**A-1**	169	0.843
S-2	170	0.999	**D-2**	171	0.004	**T-2**	172	0.996	**St-2**	173	0.999	**F-2**	172	0.999	**L-2**	174	0.995	**A-2**	175	0.010
S-3	176	0.999	**D-3**	171	0.994	**T-3**	177	0.999	**St-3**	178	0.999	**F-3**	179	0.989	**L-3**	180	0.667	**A-3**	181	0.002
S-4	182	0.966	**D-4**	183	0.997	**T-4**	184	0.999	**St-4**	185	0.846	**F-4**	186	0.999	**L-4**	187	0.999	**A-4**	188	0.000
S-5	189	0.999	**D-5**	190	0.999	**T-5**	191	0.999	**St-5**	192	0.000	**F-5**	193	0.999	**L-5**	194	0.000	**A-5**	195	0.005

3.11 Conclusions

We developed a predictive model of natural product families and glycosides based on XGBoost. We showed that the XGBoost classifier outperforms the SVM, MLP, and Log Reg classifiers with the selected hyperparameters. As the failed test cases show, every model is not entirely accurate nor reliable and has its limitations. Besides the number of samples that in some cases are not enough to cover a sufficient chemical space, it is the not restriction of some class subfragments that makes a prediction difficult as in the chroman and alkaloid examples. Nevertheless, we are exploring the capability of the ML classifiers subjected to the available spectroscopic ^{13}C NMR data. In this sense, this new tool could assist spectroscopists to identify the possible structure of the sample. It is important to mention that another kind of spectra values representation and algorithms must be explored. For example, a set representation that is permutation invariant and using another neural network architecture for this case.[23] Finally, we encourage the use of modern and free ML tools that could provide an extension of the classification and similar elucidation problems. The complete data sets for each natural product family and the code to perform predictions are available at https://github.com/saulhazelius/np-pred.

Chapter 4

Deep learning and Mass Spectrometry for organic molecules

4.1 Introduction

When a chemical synthesis is performed or when prospective new molecules are obtained from a complex mixtures extraction one must be sure about the molecular structure arrangement. This, in order to validate either the expected product of the synthesis reaction or the probably novel compounds in a natural product extraction. For achieving the goal of molecular structure elucidation, spectroscopy tools and techniques are an indispensable assistance. Spectral data is typically obtained via Nuclear Magnetic Resonance (NMR), Mass Spectrometry (MS), Infrared Spectroscopy (IR), and generally more than one method is used if a complete and reliable elucidation is attempted.[138,139] MS consists in a molecular fragmentation with ionization techniques for an ulterior separation of fragments depending on the ion weights. This yields the acquisition of the relative abundance peaks ordered by the mass-to-charge ratio m/z values. The resultant obtained values constitute the mass spectra and it differs between different ionization schemes, being the Elec-

tron Ionization (EI) and Electrospray Ionization (ESI) two of the most common ionization methods.[206] Practical usage differences between both equipment are related principally to the sample's phase and the expected molecular weight. EI for example, requires a relatively volatile sample for the transition to the gas phase in the ionization chamber. Also, given that the kinetic energy in EI provokes a high molecular fragmentation including the molecular ion, EI is limited to small molecules. Thus, the more 'soft' technique ESI is well-suited for nonvolatile liquid compounds and it is convenient, but not limited, for larger molecules. Indeed, if the molecules analyzed with ESI are separated from liquid samples mixtures using liquid chromatography (LC), the coupling of LC and ESI equipment (LC-ESI) is commonly employed in practice.[207] The range of application in this case usually cover large molecules such as biomolecules and natural products and small organic molecules as well. Now, it is important to notice that, regardless of the type of MS, routinary elucidation tasks include a peak matching of the obtained spectra with the peaks from a library of reported molecules.[208,209] However, in cases where the molecule present is unreported or unknown, the peak search fails and an alternative method for trying to elucidate the structure is needed. Recently, this situation has been addressed for EI spectrometry using ANNs (Artificial Neural Networks, see Deep Learning section) by two reported proposals with different approaches. The first work[18] by Wei *et al.* predicts spectra from a set of candidate small molecules and matches the predicted peaks with the data. The second one[19] by Fine *et al.*, instead, predicts functional groups and it is supported with IR spectroscopy information. For the LC-ESI case, two recent works that predict molecular information using ANNs methods and LC-ESI data, but focusing on peptides determination only, can be mentioned.[210,211] Thus, following the motivation of unknown or unreported molecules scenarios and considering the relevance of LC-ESI spectral data not only in the proteins and biomolecules context but also in a small molecules context, a predictive method for organic molecular structures and functional groups via LC-ESI is proposed. The ANN model is inspired from Deep Learning (DL) encoder-decoder ar-

chitectures related with sequence prediction in Natural Language Processing (NLP) tasks. In order to harness sequence models from DL, the DeepSMILES[35] sequence representation of molecules is employed. The encoder component consists in a *set-input*[23,25,26] representation of the LC-ESI values and the decoder is a Recurrent Neural Network (RNN) for the sequence prediction. The set representations are incorporated in the model via the *Set Transformer* architecture[23] that is harnessed for modeling high order relationships between the input elements using a self-attention mechanism. Self-attention is now an ubiquitous scheme in a lot of AI architectures that have reached state-of-the-art results in different tasks. This has repercuted in different AI problems such as NLP,[38,39,118] computer vision,[212,213] graph neural networks,[214] among others.[109,110] Finally, the main contributions of this work are specified as:

- A novel methodology based on set representations and RNN using DeepSMILES for the prediction of small organic molecules.

- A highly accurate, fast and cheap prediction of functional groups with potential practical use as a complementary assistant tool.

The present proposal could be used as a motivation for future restructurations on elucidation tools augmented with deep neural networks. In addition, it is important to remark that the data-driven methodology proposed here is different from the knowledge-based AI paradigm implemented in most elucidation software packages.

4.2 Methods

4.2.1 Data

The data used in this study is obtained from the MassBank repository[215], a freely available database of mass spectra. It contains several types of MS categorized into different instrument types and resolutions. For the particular case of LC-ESI instruments, the LC-ESI-QTOF (LC-ESI-Quadrupole time of flight) instrument type was the one selected because it contained the highest number of spectra at the date

of the data retrieval. The data provided by the MassBank repository consisted of a SDF file, a typical chemoinformatics file, containing the LC-ESI-QTOF spectra with their respective molecular structures represented via cartesian coordinates. All the subsequent data preprocessing for selecting the type and molecular size was performed with a *Python* script and the RDKit[216] package starting from the SDF file. The number of LC-ESI-QTOF spectra of organic molecules of different sizes was 35988. A distribution of molecular weights (MW) and DeepSMILES lengths of the contained molecules are shown in Figure 4.1.

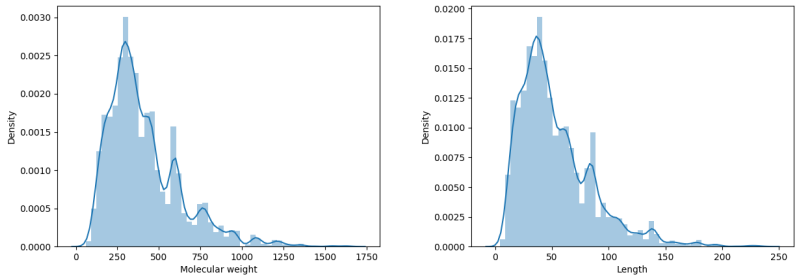


Figure 4.1: Frequency distribution of molecular weights (left) and string lengths (right) from the MassBank structures with LC-ESI-QTOF spectra.

After removing duplicates, the resulting data set was reduced to contain only the molecules of the CHONPS elements set. In addition, taking into account the motivation of small molecules context as well as considering that sequence predictions generally underperform with long sequence sizes, the final selected molecules were the ones with a MW of less than 300 Da. The mentioned preprocessing steps led to the final *csv* file containing the spectra, molecular formula, and MW values, as well as the corresponding spectra type (MS1 or MS2). The resulting distributions of MW and lengths are shown in Figure 4.2.

Considering the task of structure or substructures prediction, a brief data exploration of functional groups is also carried on. So, a histogram of selected functional groups present in the final data set is shown in Figure 4.3. That is, the number of molecules containing at least one of the selected functional groups.

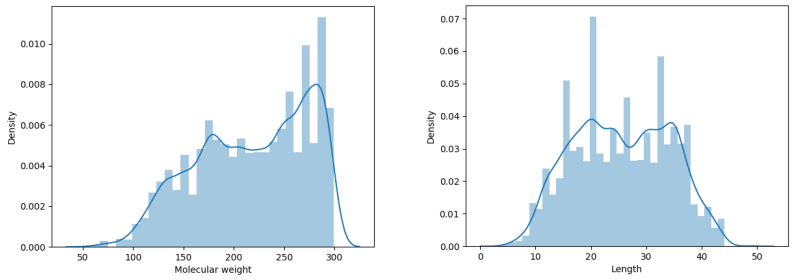


Figure 4.2: Frequency distribution of molecular weights (left) and string lengths of selected data (right).

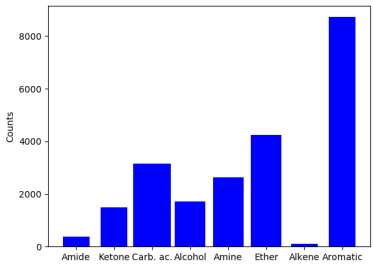


Figure 4.3: Histogram of selected functional groups in the data set.

4.2.2 Molecular representation

The selected molecular sequence representation for structure prediction is a variation of the SMILES (Simplified Molecular Input Line Entry) representation, DeepSMILES. SMILES is a one line representation of molecules designed for efficient computing processing and was proposed[33] as a system and as a language by Weininger in 1987. As a system, the SMILES consists in the codification rules that yields a desired SMILES line which has the proper grammar of the SMILES language. The method of codification treats the molecule as a graph and employs basic rules to generate the final SMILES string. The Figure 4.4 exemplifies the interconversion of a traditional 3-dimensional molecular representation with its corresponding SMILES string. Briefly, the parentheses in the SMILES represent branches of the molecule and the numbers represent the start and ending of a ring. Another common symbols used in the SMILES are =, and #, corresponding to a double and a triple bond,

respectively. The reader is referred to the reference 33 in order to get a complete explanation of the SMILES rules.

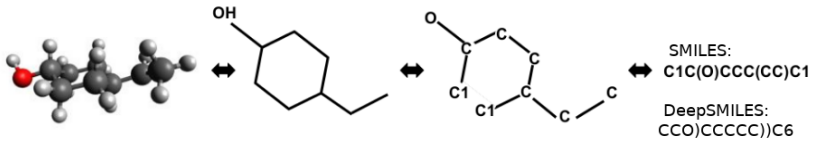


Figure 4.4: Scheme of the interconversion of a 3-dimensional structure of 4-ethylcyclohexanol to the SMILES and DeepSMILES representations.

DeepSMILES is a variant of the SMILES proposed in 2018.[35] The motivation of this work is to create a simplification of the SMILES systems in order to avoid grammar problems that emerge in deep learning generative models. The typical issues in the randomly generated SMILES is the presence of inverse parentheses and incomplete pairs of numbers. With this in hand, the DeepSMILES consist simply in the removal of one of the 2 parentheses symbols and one of the 2 numbers that represent rings. The equivalence of the representation involved, i.e., branches for parentheses and rings for numbers, is obtained with repetitions and position changes of the remaining symbols. For example, consider the SMILES string of the 4-ethylcyclohexanol C1C(O)CCC(CC)C1. The DeepSMILES is then, CCO)CCCCC))C6. As noticed, in the DeepSMILES string there is only one parentheses symbol per branch and one number per ring.

SELFIES is another molecular string representation that has recently received attention in the inverse design tasks related with virtual screening and candidate drug search. This proposal emerged as the necessity of a robust and consistent generation that overpass the randomness difficulty in other methods. That is, any generated molecule has a chemical valid syntax and meaning. This is the consequence of designing a context free grammar specifically for the chemical rules. Briefly, the basic idea is to include atoms and derivation rules in the possible strings. In addition, information of branch length and ring size is also stored yielding to a complete chemical valid sequence string.

4.2.3 Proposed architecture

The model architecture can be categorized within the encoder-decoder framework. The encoder is based on the *Set Transformer* permutation invariant framework for mapping the input values to a latent state. The reason for using the *Set Transformer* is two-fold. Firstly, given that the input must be independent of the order and the size of the input elements our problem is a $set - input$ problem. Secondly, for modeling high order interdependencies between the fragments in the ESI ionization technique leveraging the self-attention mechanism implemented in the *Set Transformer* framework. With respect to the decoder, two LSTMs units in a stacked fashion are proposed given the sequence representation of the molecules. The information of the encoder mapping is used in the LSTMs as the initial hidden state. The encoder-decoder constituted in this way is similar to Cho's encoder-decoder model[59] being the main difference that the mass spectra input is not treated as a sequence employing another RNN in the encoder but treated as a $set - input$ problem. A general representation of the proposed model is schematized in Figure 4.5.

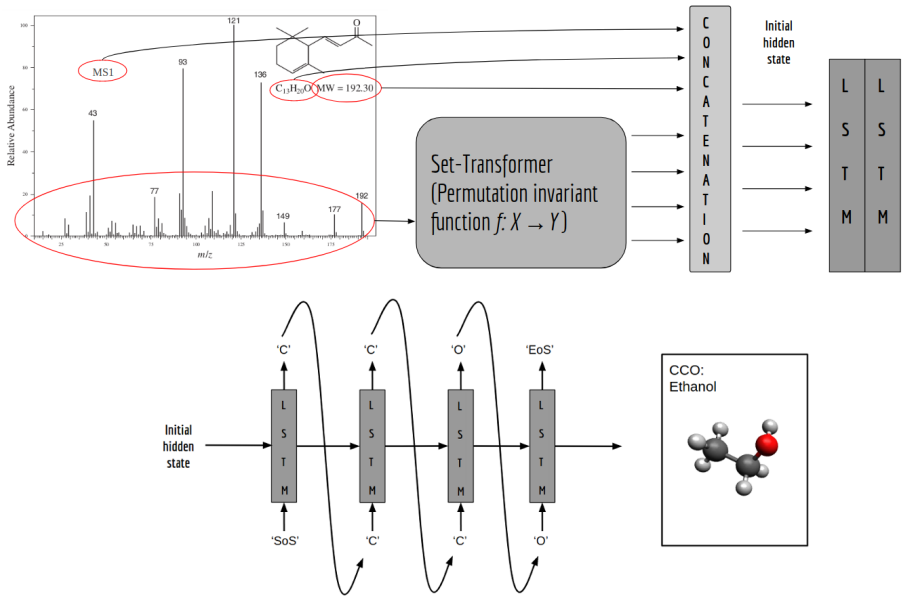


Figure 4.5: Architecture representation (top) and LSTM unfolded for DeepSMILES prediction (bottom).

The mass spectra is a two dimensional vector of variable size where the dimensions are the m/z and the relative abundance values. The molecular formula is codified as a vector representing the number of atoms in a certain set of elements. For example the formula C_3H_5ON in the CHONPS set is [3, 5, 1, 1, 0, 0]. The spectra fragmentation stage is represented as an integer of value 1 if it is MS1 and of value 2 if it is MS2. The resultant vectors of the molecular formula and spectra fragmentation are then concatenated with the mass spectra and the MW value. This vector is the initial hidden state of the first LSTM layer in the stack.

4.2.4 Hyperparameters Optimization

Performance of deep learning models relies importantly on the hyperparameter values. In the proposed model, these correspond to architectural entities from the *Set Transformer* and from the RNN. Remembering the *Set Transformer* scheme, roughly, it comprises mostly the attention blocks and these include variable dimensional values for the vector projections. In the RNN case, the hyperparameters consist of the embedding dimensions for the string characters representation and hidden state. Given the complexity of hyperparameters values combination, only few were selected for a performance analysis. The chosen *Set Transformer* hyperparameters for optimization are: number of heads, number of induced points units, and number of hidden dimensions in the projections. Regarding the LSTM decoder, the set of hyperparameters are: embedding dimensions, hidden dimensions, and dropout. The considered values are shown in the Table 4.1

Table 4.1: Selected hyperparameters for optimization.

Hyperparameter	**Domain**
Embedding Dims	{128,256}
Hidden Dims	{128,256}
FF units	{32,64,128,256}
Induced Points	{4,8,16,32}
Heads	{4,8}
Dropout	{0.1,0.2}

The combination of all values gives a total number of 256 points. Considering the

computational cost for performing a grid search over the hyperparameters values, the approach for optimizing the set of best hyperparameters was Bayesian Optimization with Gaussian Processes. This scheme is an iterative sampling procedure within the *a priori*/*a posteriori* bayesian framework relying on the Gaussian distribution as a surrogate model. The optimization was performed with the Expected Improvement acquisition function 10 evaluation points and 8 iterations, where the test error values are the evaluation points.

4.2.5 Implementation

The *Python* code for implementing the neural network includes the PyTorch DL framework[217] for the automatic differentiation and related computations. Moreover, the *Set Transformer* implementation as well as the LSTM architecture are also incorporated as PyTorch modules. For the hyperparameter optimization, the employed package is GPyOpt.[218] Regarding the pre and post processing tasks, the DeepSMILES and the RDKit[216] *Python* packages are employed for data conversion and chemoinformatic analysis (see Metrics).

4.2.6 Training

The model is trained in an end-to-end fashion. This implies a complete learning of the parameters all along the model without relying on handcrafting features or data transformation. That is, the learning range covers from the parameters of the raw data spectra representations in the *Set Transformer* mapping function to the parameters in the RNN for the string prediction. The number of epochs was 200 with the Adam[219] optimizer. The learning rate was 1e-5 and the teacher forcing ratio was set to 0.75.

4.2.7 Metrics

For evaluating the predictive model, three aspects are considered. Percentage of complete molecular prediction, f1-score of functional groups, and similarity mea-

sures. The DeepSMILES of the complete predicted molecule is canonized and compared with the canonical real DeepSMILES. The f1-score is chosen given the imbalance of functional groups presence in the data set. It is the harmonic mean of precision and recall values (see Chapter 2). In this particular case, the prediction is for the total number of a certain functional group in a molecule. For example, in the precision and recall computation, a true positive is reached if the predicted number of a specific functional group presence matches the real quantity of the functional group. The selected common functional groups are shown in the Table 4.2. In addition, similarity metrics are performed between the predicted structures and the correct test structures. These metrics include the Tanimoto similarity index[220] with different fingerprints and the edit distance. Roughly, the Tanimoto similarity consists in a comparison of sets elements, where the elements are the fingerprints of the molecular structure. To establish a fingerprint, different approaches exist with varying complexity. In this case the topological and the ECFP6[221] fingerprints are chosen. The topological fingerprint simply considers the kind of electrons, bonds, atoms in a certain substructure. The ECFP6 is an algorithm that retrieves information of surrounding atoms within a certain point lying in the center of an imaginary sphere. The ratio of this sphere is set to 6 entities conformed by bonds and atoms. The information includes stereochemistry and this fingerprint is indeed an extension of a topological fingerprint, reaching good performances in drug design and biological activities studies. The edit distance is a measure of string sequences similarities that is widely used in string matching tasks. It quantifies the number of operations for passing from one string sequence to another, for example, substitution, deletion, or insertion.

Table 4.2: Selected functional groups for model validation.

Functional group
Amide
Ketone
Carboxilic Acid
Alcohol
Amine
Ether
Alkene
Aromatic

4.3 Results

4.3.1 Hyperparameters Optimization

The data set containing the CHONPS molecules with a MW $<$ 300 Da. was randomly shuffled and splitted into a train and validation sets with a 10:1 ratio, giving (10683 and 1188 samples). The validation set was established for measuring the final model performance and the train set was used for the hyperparameter search. The hyperparameter optimization was then performed via cross-validation 3-fold where the mean error values on the test set were the evaluation points in the optimization algorithm. The best values obtained for the hyperparameters are resumed in the Table 4.3:

Table 4.3: Hyperparameters best values.

Hyperparameter	Best value
Embedding Dims	256
Hidden Dims	256
FF units	32
Induced Points	32
Heads	4
Dropout	0.1

4.3.2 Training

After selecting the optimized hyperparameters, the model training was carried on the whole original training data (10683 samples). The error in the validation set was

measured for preventing overfitting. The obtained error curves for the training and validation set are shown in Figure 4.6. It can be observed the convergence of the optimization at the epoch 200 approximately. The time required for 200 training epochs was 36 hours with a Tesla T4 GPU.

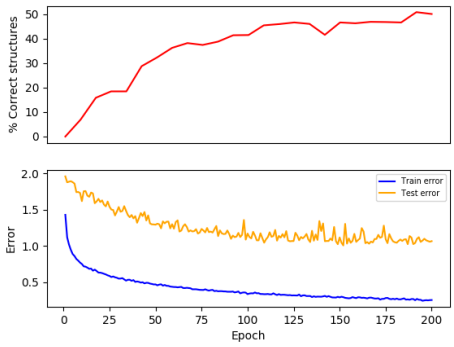


Figure 4.6: Train and test error and % of correct predicted samples.

From the Figure 4.6, it can also be seen one of the main results of the neural architecture training. Approximately, 50% of the structures are correctly predicted after training for 200 epochs. Althought 50% of correctly predicted samples can not be considered as a practical application in comparison with the CASEs employed in structure elucidation, it gives us a first insight of a neural machine capability for learning appropriate representations for mapping a mass spectra to its corresponding molecular structure and substructures without an expert intervention. In addition, one important factor that could be responsible for the achieved performance is the collision energy. This entity is implictly treated as a type of spectra (MS1 or MS2) but it was not considered explicitly. That is, the exact collision energy was not considered as a feature in the neural model.

4.3.3 Metrics

Below are shown the f1-scores of selected important functional groups vs the epoch number. The high values of the alkene and aromatics groups are notorious. This is probably due to the number of molecules containing at least one of these functional

groups (see Fig. 4.3). For example, the alkene is almost predicted as non present and indeed there are almost no alkenes in the data set, and similarly for the aromatics group. The rest of the functional groups have a similar behavior and the presence of them in the data set are not so imbalanced.

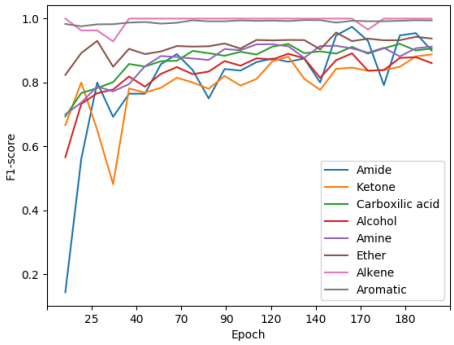


Figure 4.7: F1-scores for selected functional groups.

Regarding the similarity metrics, the mean Tanimonto similarities between the predicted and real structures for the ECFP6 and the topological fingerprints are shown in Fig 4.8 (left). This result only illustrates a general comparison of structural fragments where an increase in the similarity is observed as the number of epochs increases. The reason why similarity using topological fingerprints is higher can be adjudicated to the pattern matching methods. The topological fingerprint has a broader extension for considering similar structures than the EFCP6. For example, ECFP6 takes into account stereochemistry, yielding an extra factor for considering a substructure as similar or not. Finally, the mean edit distance between the predicted sequence and the real sequence diminishes as the number of epochs increases. Fig 4.8 (right) shows this comportment. This is what was expected given that the lower the edit distance, the higher similar the sequence. Thus, this is in accordance with the Tanimoto similarity results.

Trying to improve the predictive model from ESI spectra could be performed with an heuristic method relying on the ESI spectra prediction. For example, the predicted spectra of a candidate(s) structure(s) is compared with the input spectra.

Then, the evaluation of this spectra comparison modifies the proposed candidate structure and its spectra is predicted and compared again with the input spectra. This is done iteratively until convergence. However, prediction of the ESI spectra is not straightforward and most practical methods for mass spectra prediction are designed for EI spectra. An attempt for predicting structures from EI data was conceived and the corresponding description is found in the next section.

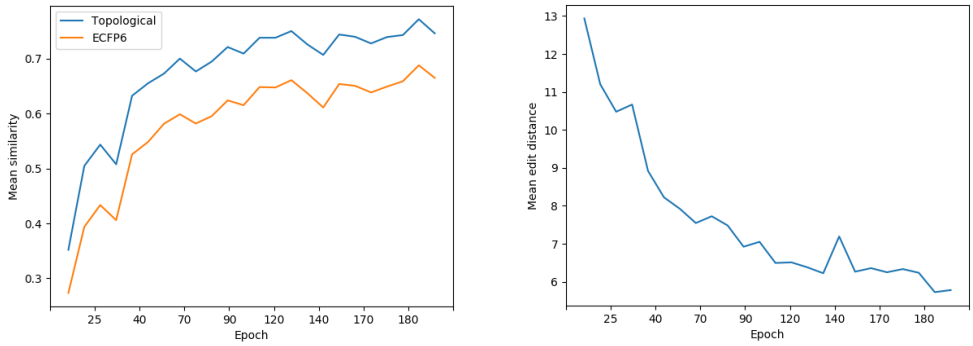


Figure 4.8: Mean Tanimoto similarities (left) and mean edit distances (right).

4.4 Prediction from EI-MS

4.4.1 Data

The data of EI spectra was also obtained from the MassBank repository. Similar to the ESI data, the EI spectra data set contains organic molecules from the CHONPS set with a MW $<$ 300 Da. After removal of duplicates, the total number of spectra is 10152. A distribution of MW and SELFIES length of the final data set is shown in Figure 4.9:

4.4.2 Results

The data set was splitted into the train and validation sets with a 10:1 ratio. The train set was subjected to hyperparameter search using CV 5-fold. The hyperparameter search was performed with Bayesian Optimization using Gaussian Processes.

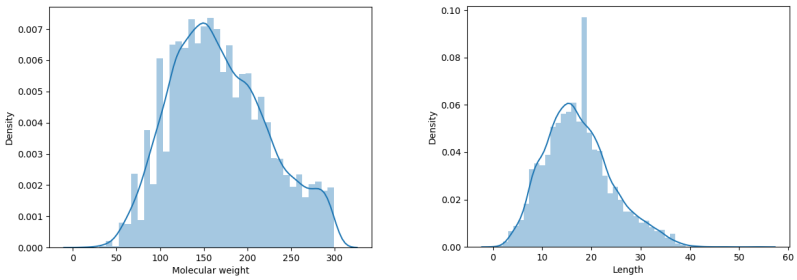


Figure 4.9

The resulting optimal values are shown in Table 4.4 The model with the optimal hyperparameters was trained with the whole training set for 120 epochs. The optimizer was Adam and the teacher forcing ratio was set to 1.0. The learning rate was scheduled from an initial value of 1e-4 and decreasing by a factor of 0.5 every 25 epochs. Error curves are shown in Figure 4.10.

Table 4.4: Hyperparameters best values.

Hyperparameter	Best value
Embedding Dims	512
Hidden Dims	512
FF units	32
Induced Points	4
Heads	4
Dropout	0.1

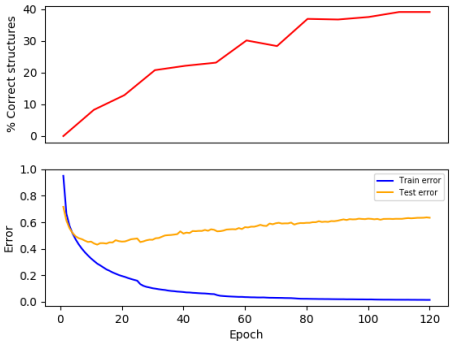


Figure 4.10

The F1-scores of the same selected functional groups as the ESI scheme are also

reported. The obtained values are shown in the Figure 4.11.

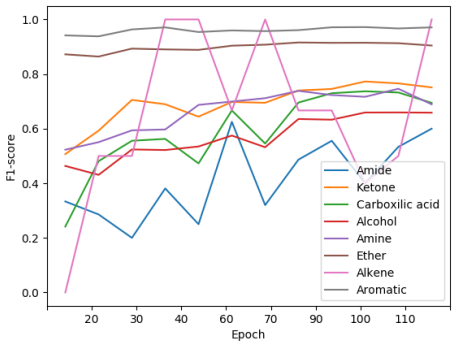


Figure 4.11

In general, lower f1-score values than the ESI performance were obtained. The respective explanation of this comportment could be adjudicated to the different teacher forcing ratio, that in this case was set to 1.0 and in the ESI model to 0.75. Moreover, as the teacher forcing ratio, there are more hyperparameters such as the learning rate, that were not subjected to optimization. This could be included in the reasons for low predictive performance.

4.4.3 RAML

As stated in the final lines from the Chapter 3 regarding the prediction of structures from ESI spectra, one possible way to improve the model is via heuristics that depends on the spectra prediction. For the case of predicting structures from EI spectra, the first attempt for using predicted spectra in an heuristic methodology considered the NN predictor of Wei *et al.* and the Wasserstein distance between the predicted and real structures as the quantity to minimize. A similar approach for elucidating a molecular structure using a Monte Carlo tree search algorithm and Wasserstein distances between real and predicted NMR is found in[222]. Nevertheless, in the MS case studied here, there was a huge overlapping region between the wrong and correct predicted structures Wasserstein distance provoking a difficult heuristics method. This overlapping region can be illustrated in the Wasserstein distance

frequency distributions from Figure 4.12.

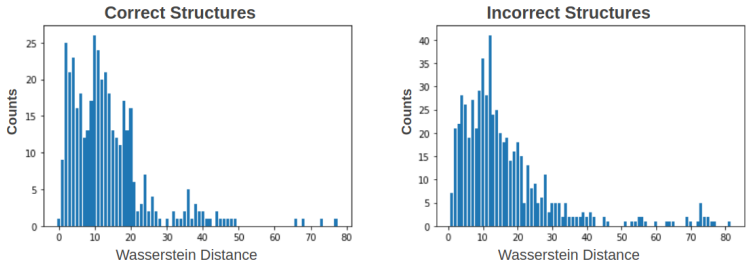


Figure 4.12

As a consequence, using the predicted EI spectra for adjusting the sequence prediction was discarded and a different improvement method was analyzed. The selected approach is based on the work of Norouzi *et al.*[223] Briefly, their method consists of an improvement of translation and speech recognition models proposing a unified framework, coined RAML (Reward Augmented Maximum Likelihood), for the maximum likelihood and reward tasks schemes. This linking is based on the fact that the optimal reward is obtained when the conditional distributions of the output sequences given the input is proportional to the scaled rewards. Especifically, when an *exponential payoff distribution* is equal to the model distribution, the optimal reward is obtained. The *exponential payoff distribution* is:

$$q(y|y*;\tau) = \frac{1}{Z(y*,\tau)} exp^{\frac{r(y,y*)}{\tau}}, \tag{4.1}$$

where r is the employed reward and it is the edit distance between the output sequence and the true sequence. The term τ is an smoothness distribution hyper-parameter and Z is computed considering the set of possible sequence outputs $\mathcal{Y}$:

$$Z(y*,\tau) = \Sigma_{y\in\mathcal{Y}} exp^{\frac{r(y,y*)}{\tau}}. \tag{4.2}$$

Then the RAML objective optimization, $\mathcal{L}_{RAML}(\theta;\tau)$, depends on sampling from q

and the gradient is expressed in terms of the expectation of q samples $E_{q(y|y*;\tau)}$:

$$\nabla_\theta \mathcal{L}_{RAML}(\theta;\tau) = \mathbb{E}_{q(y|y*;\tau)}[-\nabla_\theta log\, p_\theta(y|x)]. \tag{4.3}$$

The implementation is described as follows. First, the number of sequences with a certain edit distance from a sequence of a specific length, e, is computed for each possible sequence length. The combinatory in this estimation depends on the dictionary size, the max length of possible outputs (max length in the test set samples) and the insertion, deletion, and substitution operations for obtaining the edit distance. The counts are then reweighted by $exp^{(-e/\tau)}$ and normalized. The τ selected is 0.9, the value that yielded best results in the authors experiments. In the optimization procedure for finding the model parameters, the first term in equation 4.3 refers to sampling n examples given a batch of n true outputs and the second term to the negative log likelihood of the previously trained model. From the RL point of view, this procedure can be depicted as finding new parameters given the change in the agent policy, where the agent is the previously trained model.

Optimization was carried on with 90 steps using the Adam optimizer. The batch size was set to 4 and the initial learning rate was 1e-4 and scheduled with a decayment of 0.5 every 10 steps after the step 30. The train and test sets splitting was equal to the previous model. Results for the reparameterized model via RAML optimization scheme are shown in Fig. 4.13:

The curves show a low correct prediction rate at first, and then a convergence in the % of correctly predicted structures at the final steps. However, the values at the final step does not reach the original percentages of correct predictions from the model before the RAML reparameterization. Indeed, the percentage of correct predicted structures lowered from 40% to 30% in the test set and from 82% to 70% in training set.

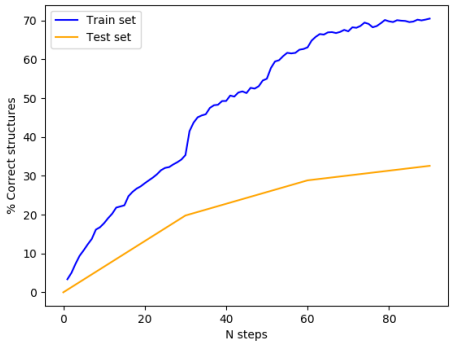


Figure 4.13

4.5 Conclusion

An encoder-decoder architecture consisting of a set representation of LC-ESI-QTOF spectra and a RNN for the prediction of molecular structures and functional groups is proposed. The model design was conceived given the employed molecular sequence representation that leverages the LSTM predictive capabilities. Another relevant characteristic of the model regarding representations is the way the input spectral data is interpreted. The fact that the input representation belongs to the $set - input$ category provides the architecure a unique encoder-decoder design. Moreover, interdependence between input elements is modeled within a self-attention scheme. Main results show that the prediction of a complete structure from ESI spectra and the proposed architecture reach a performance not suited for practical usage. However, this DL approach for elucidating organic small molecules could be the starting point for ulterior improvements, probably with an increase in the data set. The prediction of functional groups yields instead a predictive performance that can be employed as a complement in the elucidation tasks for *de novo* molecules. Finally, future work could be performed employing an architecture entirely based on the Transformer. Although the Transformer is not strictly designed for $set - input$ problems, it could be suited for the sequence prediction problem in question given the complete recurrence elimination in both input and output representations. The neural architecture proposal and the dataset for the experiments are available at

https://github.com/saulhazelius/neural-mass2mol.

Chapter 5

Conclusion

5.1 Conclusions and Future Work

Knowledge-based methodologies employed in most software tools for spectroscopic assistance, date back to the 60s. Since then, AI paradigms have been evolved and updated. Today's ML and DL belonging to the learning from experience paradigm have shown amazing results in different areas. In this study, modern ML algorithms and representational power of DL architectures are assessed to learn patterns from structural data, in contrast with logic rules and knowledge-based frameworks in spectroscopic assistant tools. Two predictive models are proposed differing in the type of molecules, spectral data, and ML methodology. An important feature of both ML and DL approaches is the database independence, common in commercial software assistant tools, that has repercussions in de novo elucidation. Part 1 is a comparison of ML classifiers employing ^{13}C NMR for the prediction of NP classes presence. The classifier that reached the best performance is XGBoost. From this analysis, the imbalanced data dependence as well as the failure to generalize given the absence of substructures in the data set is also described. A future extension of this analysis should include a set representation of the input. That is, the input ^{13}C NMR data must be independent of the vector size and order. Part 2 covers a more sophisticated DL model based on an encoder-decoder framework and a sequence representation of molecules for a complete structure elucidation. In contrast with the methodology

of chapter 1, in this architecture the input is regarded as a set-input problem. This is implemented via the set transformer model that models high-order interactions between input elements employing self-attention mechanisms. Main results show a prediction rate of 50 % of correctly predicted structures. The obtained value could be adjudicated to the fact that the hyperparameter search is not enough and fine tuning of different points or hyperparameters not considered in the search must be adjusted. Moreover, as popular knowledge points out, several DL models are 'data hungry,' in the sense that a lot of data is required for both avoiding overfitting and augmenting prediction capabilities. Although the predictive performance reached in this model could hardly yield a practical usage, it is the first method for predicting small molecules from LC-ESI via an encoder-decoder scheme and future work could be inspired by the proposed architecture. As a general result, this study shows that pattern recognition, i.e., the prediction of a NP class or functional group presence, from ML and DL algorithms in order to complement elucidation tasks is possible. As individual cases, only one of the two objectives of the present work, the prediction of NP classes, could be considered as complished. The second objective, the prediction of organic molecules, was addressed without a clear completion but allowing to elucidate possible future methods to achieve a better predictive performance.

Appendix A

Appendix

A.1 Supporting Information for Chapter 1

Examples of raw files from the Naproc13 database and the corresponding lines in a csv file.

Table S1. Example of a file for a Sesquiterpenoid class.

(2S)-Hydroxy-6-deoxypseudoanisatin
Terpenoids
Sesquiterpenoids
Prezizaanes
CD3OD
C15H22O6
298.1569
Yokoyama, R., Huang, J.M., Hosoda, A., Kino, K., Yang, C.S., Fukuyama, Y.
J Nat Prod
(2003)
66,
799-803

Num	Tipo	δ(ppm)
1	CH	46.1
2	CH	72.6
3	CH	78.5
4	C	82.4
5	C	48.6
6	CH	49.7
7	C	213.3
8	CH2	48.0
9	C	48.3
10	CH2	36.7
11	C	177.0
12	CH3	8.1
13	CH3	18.0
14	CH2	70.6
15	CH3	8.3

Table S2. Example of a file for a Steroid class.

3β,11R-Dihydroxy-5β,6β-epoxy-9,11-secogorgostan-9-one
Steroids
Gorgostanes
CDCl3
C30H50O4
474.35037
Duh, C.Y., Li, C.H., Wang, S.K., Dai, C.F.
J Nat Prod
(2006)
69,

1188-92
1

Num	**Tipo**	**δ(ppm)**
1	**CH2**	**28.6**
2	**CH2**	**30.5**
3	**CH**	**68.0**
4	**CH2**	**38.6**
5	**C**	**65.4**
6	**CH**	**58.2**
7	**CH2**	**26.2**
8	**CH**	**38.8**
9	**C**	**214.8**
10	**C**	**46.6**
11	**CH2**	**58.8**
12	**CH2**	**41.3**
13	**C**	**45.9**
14	**CH**	**45.3**
15	**CH2**	**22.6**
16	**CH2**	**28.2**
17	**CH**	**50.7**
18	**CH3**	**18.1**
19	**CH3**	**19.7**
20	**CH**	**34.7**
21	**CH3**	**21.0**
22	**CH**	**32.1**
23	**C**	**25.9**
24	**CH**	**50.8**
25	**CH**	**32.0**
26	**CH3**	**22.3**
27	**CH3**	**21.4**
22a	**CH2**	**21.4**
23a	**CH3**	**14.4**
24a	**CH3**	**15.2**

The final *csv* file for the steroid class (maximum number of carbon atoms 57) contains the next two lines from these raw files as well as the remaining lines from all the raw files with 57 or less carbon atoms.

Sorted values from Table S1:
8.1, 8.3, 18.0, 36.7, 46.1, 48.0, 48.3, 48.6, 49.7, 70.6, 72.6, 78.5, 82.4, 177.0, 213.3, -999, 0.

Sorted values from Table S2:

14.4, 15.2, 18.1, 19.7, 21.0, 21.4, 21.4, 22.3, 22.6, 25.9, 26.2, 28.2, 28.6, 30.5, 32.0, 32.1, 34.7, 38.6, 38.8, 41.3, 45.3, 45.9, 46.6, 50.7, 50.8, 58.2, 58.8, 65.4, 68.0, 214.8, -999, 1.

Classifiers grid search

This section describes the classifiers training with different hyperparameters. Table S3 shows the values selected to perform a grid search. The grid search consists of keeping fixed one value while varying the others. Cross-validation 10-fold was conducted to select the hyperparameters that yielded the highest f1-score.

Table S3. Hyperparameters values.

Classifier	Log Reg	MLP	SVM	XGBoost
Values	q: {1, 2} λ: {0.01, 0.1, 1.0, 10}	Hidden units: 10 to 710 with increments of 20. Hidden layers: {1,2} Learning rate: {0.001, 0.0001, 0.00001}	Gamma: {0.0001, 0.001, 0.01, 0.1, 1.0, 10.0} C: {0.1, 1.0, 10.0, 100.0}	Depth (trees): {2,3,4,5,6,7,8,9,10 } num. of estimators: 60 to 200 with increments of 10 learning rate: 0.05 to 0.4 with increments of 0.05

Table S4. Hyperparameters selected.

	Classifier			
Class	**Log Reg**	**MLP**	**SVM**	**XGBoost**
Sesquiterpenoid	λ = 10, q =2	Hidden units = 510, hidden layers = 2, l. r. = 1e-4	Gamma = 0.001, C = 10	Depth = 9 Num. est. = 200 l. r. = 0.3
Diterpenoid	λ = 0.01, q =2	Hidden units = 630, hidden layers = 2, l. r. = 1e-4	Gamma = 0.0001, C = 100	Depth = 8 Num. est. = 200 l. r. = 0.3
Triterpenoid	λ = 0.01, q =1	Hidden units = 270, hidden layers = 2, l. r. = 1e-4	Gamma = 0.0001, C = 100	Depth = 7 Num. est. = 200 l. r. = 0.35
Lignan	λ = 0.1, q =1	Hidden units = 350, hidden layers = 2, l. r. = 1e-5	Gamma = 0.0001, C = 10	Depth = 9 Num. est. = 160 l. r. = 0.15
Steroid	λ = 10, q =1	Hidden units = 630, hidden layers = 2, l. r. = 1e-4	Gamma = 0.001, C = 10	Depth = 6 Num. est. = 140 l. r. = 0.3
Chroman	λ = 0.1, q =1	Hidden units = 690, hidden layers = 2, l. r. = 1e-5	Gamma = 0.001, C = 10	Depth = 9 Num. est. = 200 l. r. = 0.35
Glycoside	λ = 1, q =1	Hidden units = 670, hidden layers = 2, l. r. = 1e-4	Gamma = 0.0001, C = 10	Depth = 8 Num. est. = 180 l. r. = 0.3
Flavonoid	λ = 1, q =1	Hidden units = 630, hidden layers = 2, l. r. = 1e-4	Gamma = 0.0001, C = 10	Depth = 5 Num. est. = 200 l. r. = 0.35
Alkaloid	λ = 0.01, q =1	Hidden units = 430, hidden layers = 2, l. r. = 1e-5	Gamma = 0.001, C = 10	Depth = 7 Num. est. = 180 l. r. = 0.35

Table S5. Accuracy values.

	Accuracy			
Class	**Log Reg**	**MLP**	**SVM**	**XGBoost**
Sesquiterpenoid	0.923	0.926	0.962	0.973
Diterpenoid	0.838	0.872	0.945	0.965
Triterpenoid	0.908	0.913	0.972	0.987
Lignan	0.984	0.888	0.994	0.997
Steroid	0.965	0.962	0.989	0.993
Chroman	0.984	0.888	0.995	0.995
Glycoside	0.922	0.923	0.964	0.972
Flavonoid	0.954	0.955	0.981	0.987
Alkaloid	0.984	0.992	0.994	0.993

Table S6. Precision values.

	Precision			
Class	**Log Reg**	**MLP**	**SVM**	**XGBoost**
Sesquiterpenoid	0.898	0.905	0.958	0.958
Diterpenoid	0.791	0.818	0.915	0.946
Triterpenoid	0.751	0.716	0.932	0.971
Lignan	0.174	0.00147	0.823	0.960

Steroid	0.578	0.00	0.954	0.966
Chroman	0.202	0.00154	0.959	0.943
Glycoside	0.604	0.390	0.792	0.842
Flavonoid	0.719	0.707	0.881	0.914
Alkaloid	0.0125	0.00	0.783	0.825

Table S7. Recall values.

	Recall			
Class	**Log Reg**	**MLP**	**SVM**	**XGBoost**
Sesquiterpenoid	0.678	0.691	0.841	0.901
Diterpenoid	0.710	0.803	0.923	0.950
Triterpenoid	0.814	0.641	0.929	0.962
Lignan	0.0254	0.100	0.747	0.841
Steroid	0.238	0.00	0.737	0.853
Chroman	0.0203	0.100	0.668	0.693
Glycoside	0.217	0.191	0.773	0.819
Flavonoid	0.671	0.736	0.880	0.916
Alkaloid	0.0143	0.00	0.305	0.226

Table S8. AUROC values.

	AUROC			
Class	**Log Reg**	**MLP**	**SVM**	**XGBoost**
Sesquiterpenoid	0.917	0.944	0.979	0.991
Diterpenoid	0.906	0.946	0.973	0.992
Triterpenoid	0.957	0.960	0.985	0.997

Lignan	0.939	0.362	0.955	0.997
Steroid	0.934	0.935	0.973	0.995
Chroman	0.880	0.597	0.958	0.982
Glycoside	0.869	0.902	0.962	0.986
Flavonoid	0.976	0.978	0.969	0.997
Alkaloid	0.694	0.488	0.868	0.916

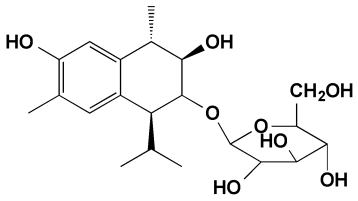
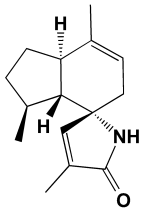
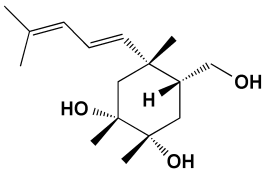
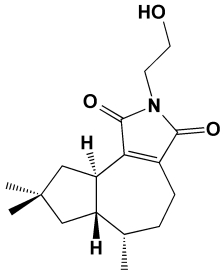
Table S9. Specificity values.

	Specificity			
Class	**Log Reg**	**MLP**	**SVM**	**XGBoost**
Sesquiterpenoid	0.983	0.985	0.990	0.986
Diterpenoid	0.904	0.943	0.955	0.972
Triterpenoid	0.931	0.992	0.983	0.991
Lignan	1.00	1.00	1.00	1.00
Steroid	0.992	0.999	1.00	1.00
Chroman	1.00	1.00	1.00	1.00
Glycoside	0.987	0.987	0.981	0.987
Flavonoid	0.980	0.970	0.990	0.991
Alkaloid	0.990	0.70	1.00	1.00

Test cases details

Tables S10-S18 show the ^{13}C NMR values for the test samples. If more than one compound is reported, the number of the compound tested is indicated.

Table S10. ^{13}C NMR values of sesquiterpenoid test cases.

Compound	Reference	Structure	^{13}C NMR (ppm)
Sq-1	163		38.4, 73.7, 81.6, 52.0, 133.7, 123.2, 155.1, 114.0, 139.8, 128.2, 34.3, 21.2, 22.4, 19.0, 16.1, 104.0, 74.7, 78.0, 71.7, 78.2, 62.8
Sq-2	170		27.1, 31.8, 31.7, 55.3, 62.4, 38.3, 118.6, 137.0, 45.9, 21.8, 148.3, 132.3, 174.2, 10.6, 20.5
Sq-3	176		71.7, 74.2, 30.4, 48.1, 37.8, 52.3, 137.8, 122.0, 126.2, 130.7, 25.5, 18.0, 26.5, 62.3, 27.9
Sq-4	182		37.1, 144.2, 142.9, 19.9, 33.5, 31.8, 48.7, 44.4, 35.9, 45.6, 172.3, 172.6, 12.6, 31.1, 31.4, 40.7, 61.5

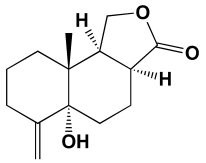
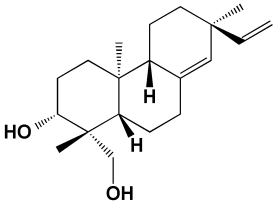
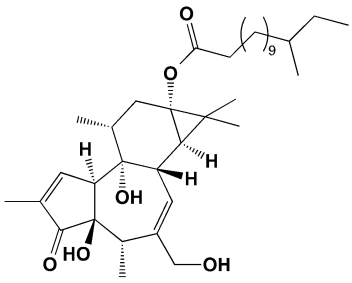
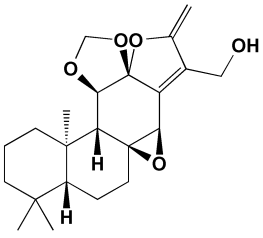
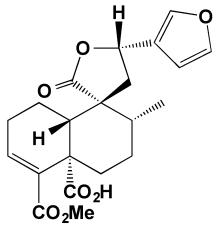
Sq-5	189		32.3, 22.3, 31.8, 150.6, 75.4, 31.2, 20.8, 39.3, 46.0, 40.8, 68.9, 178.3, 109.4, 15.2

Table S11. ^{13}C NMR values of diterpenoid test cases.

Compound	Reference	Structure	^{13}C NMR (ppm)
D-1	164		37.1, 28.1, 81.0, 42.9, 50.3, 19.0, 36.0, 136.1, 55.2, 37.8, 22.2, 34.4, 37.4, 129.2, 148.8, 110.2, 26.0, 22.7, 64.3, 16.0
D-2	171		161.0, 134.8, 210.6, 74.8, 38.5, 142.0, 130.4, 40.0, 77.6, 57.2, 37.5, 33.1, 65.0, 33.3, 24.2, 23.5, 15.9, 19.1, 10.2, 68.3, 177.6, 35.3, 26.0, 30.5, 30.5, 30.5, 30.5, 30.5, 30.5, 28.2, 37.8, 35.7, 31.1, 11.8, 19.7
D-3	171		39.7, 39.7, 19.5, 19.5, 42.7, 42.7, 34.4, 55.0, 22.0, 22.0, 36.6, 36.6, 66.7, 55.1, 40.5, 80.2, 108.9, 155.3, 57.4, 133.5, 170.3, 56.5, 34.0, 22.1, 16.1, 96.7, 96.7
D-4	183		178.7, 176.0, 168.4, 145.5, 142.0, 141.4, 137.6, 127.0, 109.2, 73.8, 52.9, 52.3, 51.9, 47.4, 43.1, 41.0, 33.5, 29.1, 27.3, 20.1, 17.6

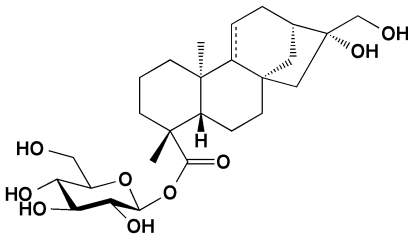
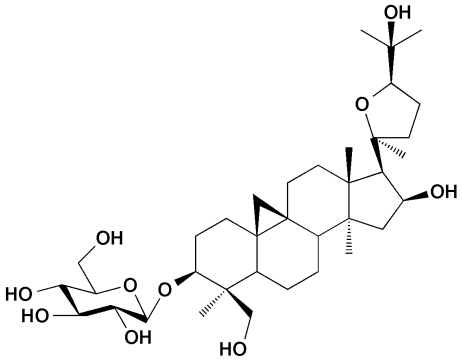
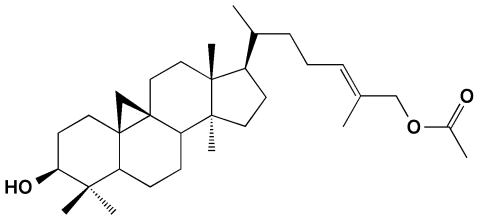
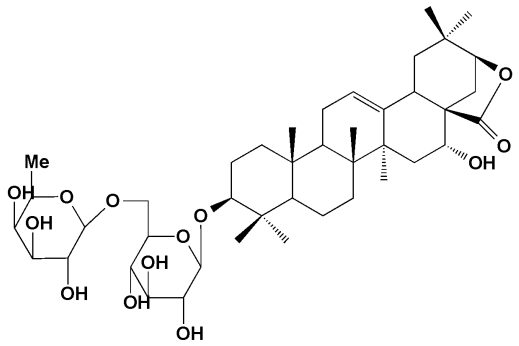
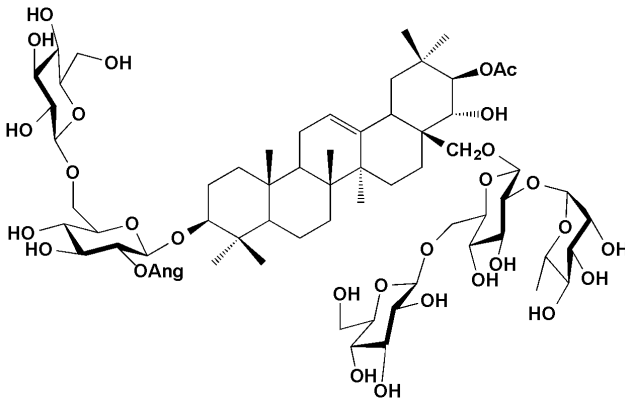
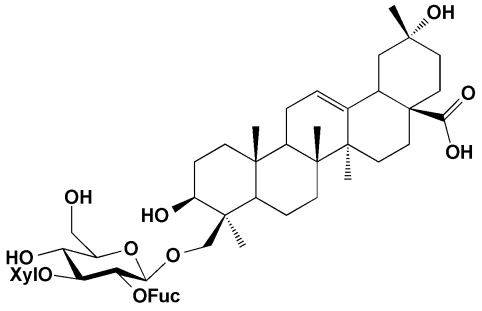
D-5	190		42.2, 21.2, 39.4, 46.2, 48.1, 19.6, 31.3, 43.9, 159.0, 39.9, 114.9, 31.2, 45.0, 44.2, 55.8, 85.6, 68.9, 28.4, 177.7, 24.5, 95.4, 74.0, 78.5, 71.1, 78.7, 62.4

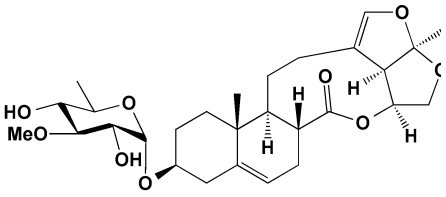
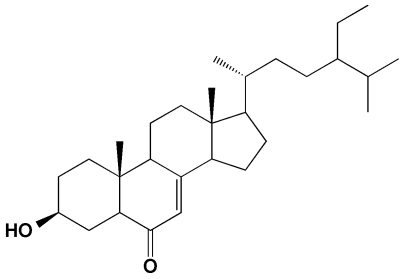
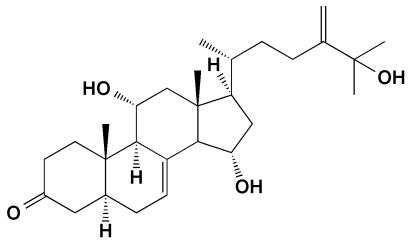
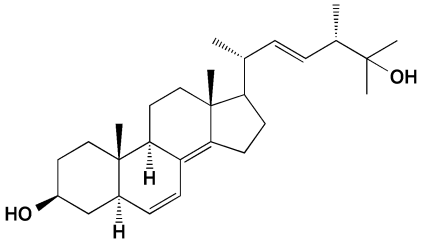
Table S12. ^{13}C NMR values of triterpenoid test cases.

Compound	Reference	Structure	^{13}C NMR (ppm)
T-1	165		33.0, 29.6, 83.4, 46.0, 41.7, 21.6, 27.2, 49.5, 20.8, 27.1, 27.5, 34.6, 47.9, 47.7, 49.4, 74.3, 57.2, 21.8, 31.6, 26.2, 88.0, 38.3, 24.8, 85.5, 71.5, 26.6, 27.5, 64.4, 11.9, 20.8, 106.3, 73.3, 75.3, 70.3, 76.5, 62.4
T-2	172		32.1, 30.4, 78.9, 40.6, 47.2, 21.2, 26.2, 48.1, 20.1, 26.1, 26.5, 33.0, 45.4, 48.9, 35.8, 28.2, 52.3, 18.1, 30.0, 36.0, 18.1, 35.6, 24.8, 130.7, 129.6, 75.9, 14.0, 25.5, 14.1, 18.3, 171.1, 21.1

T-3	177		39.0, 27.0, 89.2, 39.8, 56.2, 18.7, 32.8, 40.6, 47.5, 37.2, 24.0, 124.9, 140.4, 43.6, 38.5, 67.0, 50.3, 42.0, 43.3, 34.5, 83.8, 27.5, 28.4, 17.3, 16.0, 16.5, 29.1, 181.6, 28.9, 24.6
T-4	184		38.9, 26.7, 89.4, 39.4, 55.8, 18.7, 33.0, 40.5, 48.1, 37.1, 24.2, 124.7, 142.3, 42.0, 25.9, 18.9, 43.7, 41.0, 46.1, 36.3, 79.7, 69.5, 28.3, 17.1, 16.2, 16.9, 26.5, 74.2, 29.9, 20.5, 104.2, 75.0, 76.3, 72.2, 77.3, 70.3, 105.6, 75.3, 78.6, 71.9, 78.5, 62.9, 103.7, 80.3, 74.4, 71.9, 76.7, 70.5, 100.6, 72.7, 72.7, 74.5, 69.0, 18.9, 105.9, 75.3, 78.5, 71.9, 78.7, 62.9, 167.4, 128.8, 138.2, 16.3, 21.1, 171.5, 21.4
T-5	191		39.2, 28.6, 79.6, 43.4, 56.9, 19.6, 33.8, 40.0, 48.3, 37.5, 24.3, 122.5, 144.1, 42.4, 28.4, 24.2, 47.1, 44.7, 48.4, 70.2, 36.5, 35.4, 23.8, 72.9, 16.0, 17.6, 26.3, 180.6, 26.0, 103.9, 79.1, 87.8, 69.8, 78.5, 62.5, 103.9,

			73.5, 77.1, 76.5, 70.0, 18.9, 105.7, 75.2, 78.9, 73.2, 68.2

Table S13. ^{13}C NMR values of steroid test cases.

Compound	Reference	Structure	^{13}C NMR (ppm)
St-1	166		36.2, 27.5, 75.8, 39.7, 140.3, 120.2, 28.0, 40.4, 52.8, 38.5, 23.6, 29.6, 118.1, 175.6, 67.6, 75.1, 55.7, 143.3, 18.0, 114.0, 24.6, 95.2, 34.5, 78.2, 76.8, 67.5, 17.8, 56.5
St-2	173		36.5, 31.2, 70.5, 41.8, 49.9, 202.3, 126, 165, 45.4, 38.3, 21.2, 38.7, 43.1, 49.9, 26.3, 28.5, 54.7, 12, 17.3, 36.1, 18.9, 33.9, 26.1, 45.8, 29.1, 19.8, 19.0, 23.0, 12.0
St-3	178		37.4, 38.6, 211.2, 43.6, 52.6, 39.5, 119.2, 133.0, 61.6, 34.5, 68.9, 50.6, 43.0, 61.6, 69.1, 34.4, 42.5, 11.4, 13.3, 35.1, 17.8, 29.0, 26.7, 155.6, 72.8, 28.5, 28.6, 106.2
St-4	185		35.1, 31.5, 71.4, 36.57, 44.8, 129.6, 125.7, 125.5, 48.1, 35.7, 19.7, 36.6, 43.5, 147.0, 25.0, 28.3, 55.7, 19.4, 11.3, 39.7, 21.0, 139.0, 129.5, 48.3, 72.5, 27.0, 26.3, 15.9

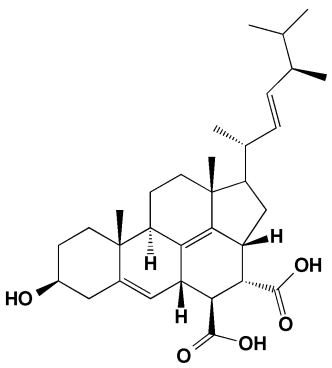
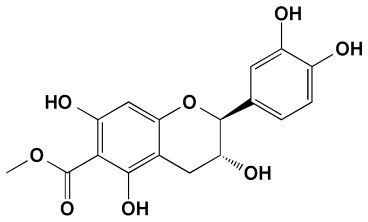
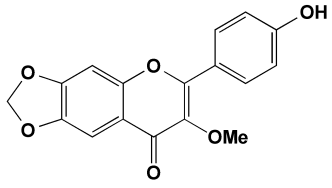
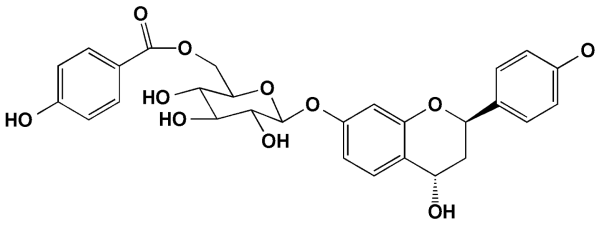
St-5	192		37.2, 32.5, 71.0, 42.6, 145.6, 122.0, 37.9, 130.0, 48.4, 37.2, 21.0, 39.9, 43.3, 146.2, 35.9, 34.8, 56.4, 20.8, 18.3, 41.2, 20.7, 136.8, 133.6, 44.3, 34.3, 20.4, 20.2, 18.2, 178.9, 50.1, 50.1, 177.9

Table S14. ^{13}C NMR values of flavonoid test cases.

Compound	Reference	Structure	^{13}C NMR (ppm)
F-1	167		81.4, 65.4, 26.3, 160.0, 94.5, 158.9, 95.0, 160.0, 100.0, 170.6, 52.3, 129.7, 114.1, 145.0, 145.0, 115.3, 118.0
F-2	172		155.9, 138.2, 178.1, 160.9, 99.3, 163.0, 95.1, 156.0, 105.9, 119.3, 106.3, 147.9, 139.4, 147.9, 106.3, 100.1, 69.7, 76.5, 73.2, 77.4, 60.7, 59.8, 56.2, 56.2
F-3	179		61.3, 94.4, 103.0, 106.0, 113.0, 116.3, 116.3, 121.8, 128.3, 128.3, 136.2, 140.6, 153.0, 154.3, 161.0, 161.1, 176.3
F-4	186		74.3, 39.8, 64.2, 132.4, 110.3, 157.1, 105.4, 159.8, 119.1, 133.4, 128.8, 116.2, 158.3, 116.2, 128.8, 101.8, 74.8, 77.8,

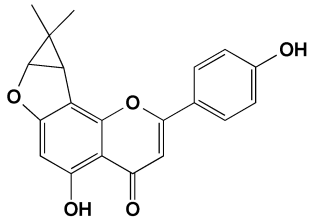
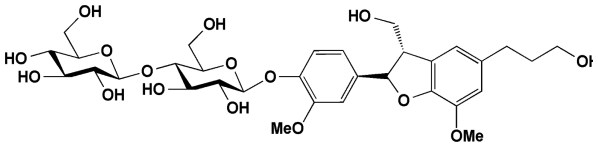
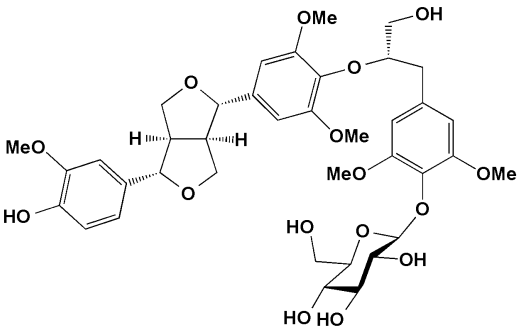
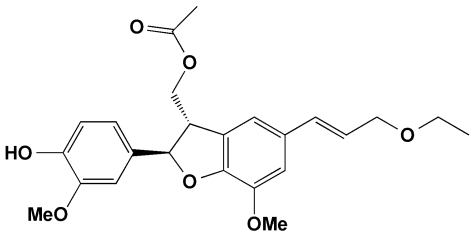
			72.0, 75.4, 65.0, 121.8, 132.9, 116.3, 163.9, 116.3, 132.9, 168.1
F-5	193		164.3, 103.2, 182.6, 161.2, 93.9, 168.0, 105.9, 153.0, 104.8, 122.0, 128.2, 128.2, 116.0, 116.0, 160.9, 28.7, 73.1, 15.0, 22.6, 12.6

Table S15. ^{13}C NMR values of lignan test cases.

Compound	Reference	Structure	^{13}C NMR (ppm)
L-1	168		136.1, 110.7, 149.4, 146.3, 115.7, 118.3, 87.0, 54.0, 63.5, 135.7, 116.9, 129.2, 145.9, 143.8, 112.9, 32.0, 35.1, 60.7, 56.1, 56.1, 100.0, 73.4, 75.4, 80.2, 75.6, 60.4, 103.6, 73.7, 76.9, 70.5, 77.2, 61.5
L-2	174		132.7, 108.9, 146.8, 145.3, 114.4, 118.9, 85.7, 54.0, 72.1, 137.5, 102.9, 153.5, 134.7, 153.5, 102.9, 86.0, 54.5, 71.6, 136.0, 106.8, 152.4, 133.8, 152.4, 106.8,38.1, 83.7, 62.3, 106.3, 74.2, 76.0, 70.1, 76.5, 62.5, 56.0, 56.2, 56.3
L-3	180		88.9, 50.5, 115.2, 131.2, 110.4, 144.5, 132.4, 124.5, 71.4, 65.5, 171.0, 21.0, 132.4, 108.7, 146.8, 146.0, 114.4, 119.7, 127.7, 148.0, 56.1, 56.1, 65.8, 15.4

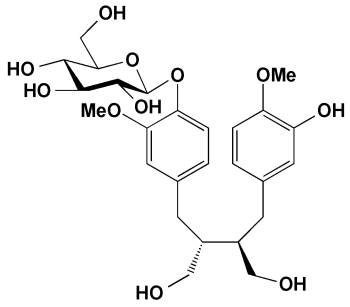
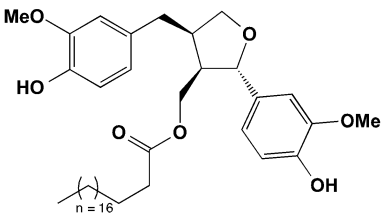
L-4	187		36.4, 36.4, 44.9, 45.1, 56.6, 56.6, 61.7, 61.9, 63.0, 71.9, 75.5, 79.0, 79.3, 103.4, 114.2, 114.9, 116.9, 117.2, 122.7, 123.1, 133.7, 137.0, 146.9, 149.2, 150.2, 150.7
L-5	194	 **63/3**	132.8, 113.1, 148.3, 145.8, 115.8, 121.8, 33.8, 43.5, 73.1, 135.6, 110.4, 148.3, 146.8, 115.5, 119.5, 83.8, 50.3, 63.1, 56.3, 56.3, 173.7, 34.7, 25.7, 28.6, 28.6, 28.6, 28.6, 28.6, 28.6, 28.6, 28.6, 28.6, 28.6, 28.6, 28.6, 28.6, 28.6, 28.6, 28.6, 28.6, 28.6, 14.4

Table S16. ^{13}C NMR values of alkaloid test cases.

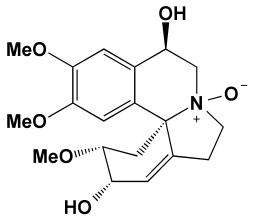
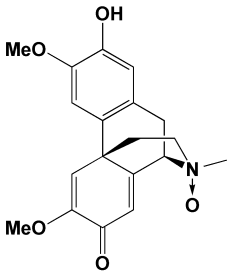
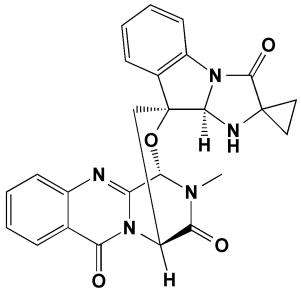
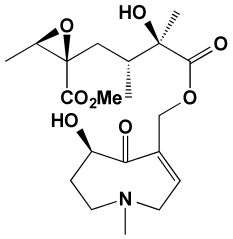
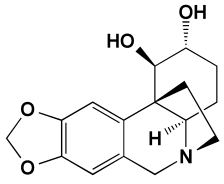
Compound	Reference	Structure	^{13}C NMR (ppm)
A-1	169		127.4, 62.6, 76.3, 29.8, 81.6, 140.3, 26.4, 65.5, 66.3, 64.8, 127.2, 127.5, 110.4, 149.5, 150.7, 114.2, 55.9, 56.1, 56.2
A-2	175		115.8, 148.3, 153.1, 111.1, 121.6, 158.4, 182.4, 127.9, 77.7, 36.2, 130.5, 126.3, 42.8, 149.7, 38.2, 61.3, 57.3, 56.4, 58.3
A-3	181		167.8, 84.6, 148.6, 147.0, 128.1, 135.2, 127.7, 126.6, 120.7, 158.7, 51.4, 33.5, 80.9, 85.1, 139.4, 116.2, 130.1, 123.6, 124.6, 137.1, 46.2, 176.3, 14.7, 12.9, 32.8
A-4	188		186.9, 176.5, 170.4, 134.9, 132.2, 77, 72.0, 64.9, 61.9, 60.5, 58.2, 52.9, 52.3, 44.1, 37.5, 35.0, 33.3, 23.1, 13.8, 13.6
A-5	195		147.3, 147.7, 142.7, 125.7, 106.9, 106.7, 101.9, 74.1, 70.9, 69.4, 62.5, 52.4, 50.5, 37.6, 29.5, 21.4

Table S17. ^{13}C NMR values of chroman (non coumarin) test cases.

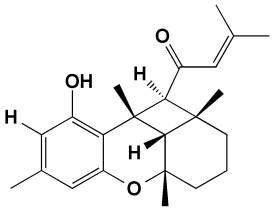
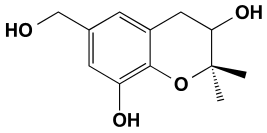
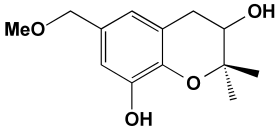
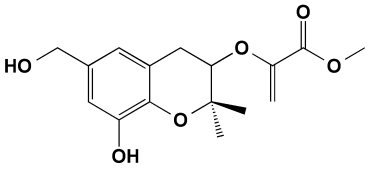
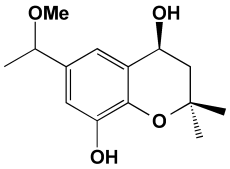
Compound	Reference	Structure	^{13}C NMR (ppm)
Ch-1	198		75.3, 45.1, 24.0, 112.0, 154.4, 111.5, 137.9, 110.6, 153.4, 36.4, 17.4, 34.4, 39.4, 58.0, 203.2, 122.4, 158.0, 21.5, 28.2, 29.6, 25.8, 21.5
Ch-2	200		76.7, 68.4, 31.2, 120.4, 117.8, 133.5, 111.9, 145.4, 139.7, 20.0, 25.7, 62.9
Ch-3	200		76.8, 68.2, 31.1, 120.6, 119.2, 129.2, 112.8, 145.5, 140.4, 20.1, 25.7, 73.7, 57.1
Ch-4	200		77.4, 67.8, 31.0, 122.3, 124.2, 134.2, 117.0, 142.0, 142.7, 20.3, 25.4, 62.2, 150.0, 100.5, 162.4, 52.3
Ch-5	204		75.4, 42.9, 63.9, 125.7, 135.3, 127.4, 117.4, 124.2, 26.2, 29.1, 79.3, 23.6, 56.4

Table S18. ^{13}C NMR values of chroman (coumarin) test cases.

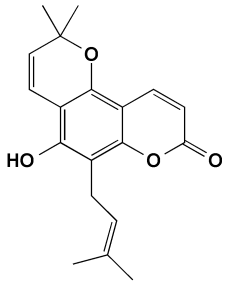
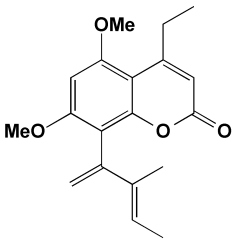
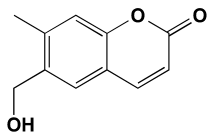
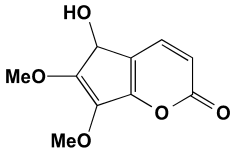
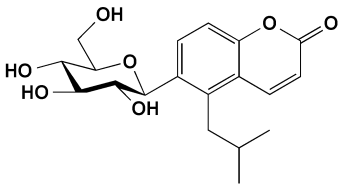
Compound	Reference	Structure	^{13}C NMR (ppm)
Co-1	199		162.0, 110.5, 139.2, 148.9, 106.2, 154.5, 106.1, 152.6, 103.6, 116.2, 128.0, 77.5, 28.0, 28.0, 21.8, 121.0, 136.5, 18.0,25.9
Co-2	201		91.1, 110.2, 143.0, 29.5, 10.6, 13.5, 15.1, 55.9, 56.1, 160.4, 159.2, 159.1, 153.1, 194.3, 103.9, 111.1, 139.7, 159.2
Co-3	202		160.6, 110.8, 144.8, 111.2, 126.9, 126.6, 158.1, 101.3, 154.1, 57.5
Co-4	203		161.5, 109.1, 141.1, 104.8, 152.3, 133.5, 157.0, 89.7, 60.6, 56.5
Co-5	205		163.04, 111.26, 144.71, 97.23, 125.63, 160.05, 112.18, 154.99, 125.63, 28.81, 21.8, 23.1, 23.1, 73.5, 90.06, 70.2, 71.0,76.6, 60.8

Table S19. Probabilities from the sesquiterpenoid cases.

Class	S-1	S-2	S-3	S-4	S-5
Sesquiterpenoid	0.974	0.999	0.999	0.986	0.999
Diterpenoid	0.007	0.000	0.000	0.082	0.000
Triterpenoid	0.015	0.000	0.000	0.000	0.000
Lignan	0.042	0.001	0.002	0.002	0.001
Steroid	0.000	0.000	0.000	0.024	0.000
Chroman	0.058	0.002	0.000	0.002	0.002
Glycoside	0.994	0.000	0.000	0.000	0.004
Flavonoid	0.000	0.00	0.000	0.000	0.000
Alkaloid	0.003	0.543	0.071	0.103	0.166

Table S20. Probabilities from the diterpenoid cases.

Class	D-1	D-2	D-3	D-4	D-5
Sesquiterpenoid	0.002	0.027	0.002	0.029	0.006
Diterpenoid	0.999	0.041	0.997	0.998	0.997
Triterpenoid	0.000	0.103	0.001	0.000	0.001
Lignan	0.000	0.000	0.000	0.000	0.000
Steroid	0.000	0.077	0.035	0.000	0.000
Chroman	0.000	0.000	0.001	0.000	0.000
Glycoside	0.000	0.286	0.969	0.005	0.983
Flavonoid	0.000	0.000	0.000	0.000	0.000
Alkaloid	0.002	0.000	0.000	0.000	0.000

Table S21. Probabilities from the triterpenoid cases.

Class	T-1	T-2	T-3	T-4	T-5
Sesquiterpenoid	0.000	0.002	0.000	0.000	0.000
Diterpenoid	0.000	0.005	0.000	0.000	0.001
Triterpenoid	0.998	0.999	0.999	0.995	0.999
Lignan	0.001	0.001	0.002	0.000	0.001
Steroid	0.319	0.174	0.000	0.000	0.004
Chroman	0.000	0.002	0.000	0.000	0.000
Glycoside	0.961	0.000	0.000	0.000	0.997
Flavonoid	0.000	0.000	0.000	0.000	0.000
Alkaloid	0.003	0.019	0.010	0.000	0.000

Table S22. Probabilities from the steroid cases.

Class	St-1	St-2	St-3	St-4	St-5
Sesquiterpenoid	0.000	0.000	0.000	0.006	0.001
Diterpenoid	0.896	0.000	0.005	0.041	0.004
Triterpenoid	0.026	0.281	0.004	0.184	0.992
Lignan	0.005	0.003	0.004	0.003	0.004
Steroid	0.953	0.998	0.999	0.998	0.057
Chroman	0.000	0.002	0.003	0.002	0.000
Glycoside	0.989	0.013	0.004	0.038	0.001
Flavonoid	0.001	0.000	0.000	0.000	0.000

Alkaloid	0.0112	0.002	0.009	0.002	0.001

Table S23. Probabilities from the flavonoids cases.

Class	**F-1**	**F-2**	**F-3**	**F-4**	**F-5**
Sesquiterpenoid	0.000	0.000	0.000	0.000	0.001
Diterpenoid	0.000	0.000	0.000	0.000	0.000
Triterpenoid	0.000	0.000	0.000	0.000	0.000
Lignan	0.021	0.027	0.001	0.003	0.000
Steroid	0.000	0.000	0.000	0.000	0.000
Chroman	0.886	0.001	0.002	0.328	0.00
Glycoside	0.093	0.003	0.977	0.999	0.039
Flavonoid	0.999	0.999	0.556	0.999	0.669
Alkaloid	0.037	0.000	0.000	0.000	0.000

Table S24. Probabilities from the lignan cases.

			Reference		
Class	**L-1**	**L-2**	**L-3**	**L-4**	**L-5**
Sesquiterpenoid	0.013	0.001	0.001	0.004	0.009
Diterpenoid	0.000	0.000	0.000	0.000	0.007
Triterpenoid	0.000	0.000	0.000	0.000	0.655
Lignan	0.967	0.978	0.987	0.999	0.002

Steroid	0.000	0.000	0.000	0.002	0.008
Chroman	0.001	0.000	0.399	0.028	0.000
Glycoside	0.995	0.964	0.024	0.997	0.014
Flavonoid	0.586	0.965	0.895	0.336	0.000
Alkaloid	0.080	0.034	0.887	0.739	0.000

Table S25. Probabilities from the alkaloid cases.

Class	**A-1**	**A-2**	**A-3**	**A-4**	**A-5**
Sesquiterpenoid	0.030	0.002	0.0122	0.954	0.983
Diterpenoid	0.000	0.000	0.000	0.628	0.000
Triterpenoid	0.000	0.000	0.000	0.005	0.000
Lignan	0.992	0.974	0.456	0.002	0.012
Steroid	0.000	0.000	0.007	0.002	0.000
Chroman	0.002	0.056	0.019	0.004	0.004
Glycoside	0.071	0.000	0.034	0.145	0.034
Flavonoid	0.307	0.933	0.997	0.000	0.013
Alkaloid	0.954	0.965	0.879	0.320	0.992

Table S26. Probabilities from the chroman (non-coumarin) cases.

Class	**Ch-1**	**Ch-2**	**Ch-3**	**Ch-4**	**Ch-5**
Sesquiterpenoid	0.001	0.010	0.003	0.032	0.149

Diterpenoid	0.965	0.000	0.000	0.002	0.000
Triterpenoid	0.003	0.000	0.001	0.000	0.001
Lignan	0.000	0.000	0.003	0.000	0.000
Steroid	0.000	0.000	0.000	0.000	0.000
Chroman	0.001	0.951	0.844	0.567	0.000
Glycoside	0.000	0.002	0.004	0.013	0.000
Flavonoid	0.000	0.000	0.000	0.018	0.000
Alkaloid	0.004	0.211	0.021	0.011	0.005

Table S27. Probabilities from the chroman (coumarin) cases.

Class	**Co-1**	**Co-2**	**Co-3**	**Co-4**	**Co-5**
Sesquiterpenoid	0.001	0.309	0.000	0.000	0.002
Diterpenoid	0.043	0.000	0.000	0.000	0.005
Triterpenoid	0.006	0.001	0.000	0.000	0.000
Lignan	0.005	0.012	0.006	0.005	0.004
Steroid	0.000	0.026	0.000	0.000	0.001
Chroman	0.930	0.986	0.998	0.995	0.858
Glycoside	0.003	0.004	0.002	0.001	0.181
Flavonoid	0.206	0.621	0.009	0.001	0.618
Alkaloid	0.0796	0.147	0.0662	0.124	0.037

Bibliography

[1] Steinbeck, C. Recent developments in automated structure elucidation of natural products. *Nat. Prod. Rep.* **2004**, *21*, 512–518.

[2] Russell, S.; Norvig, P. Artificial intelligence: a modern approach. **2002**,

[3] Buchanan, B. G.; Feigenbaum, E. A. *Readings in artificial intelligence*; Elsevier, 1981; pp 313–322.

[4] Lindsay, R. K.; Buchanan, B. G.; Feigenbaum, E. A.; Lederberg, J. DENDRAL: a case study of the first expert system for scientific hypothesis formation. *Artificial intelligence* **1993**, *61*, 209–261.

[5] Plainchont, B.; Nuzillard, J.-M. Structure verification through computer-assisted spectral assignment of NMR spectra. *Magnetic Resonance in Chemistry* **2013**, *51*, 54–59.

[6] LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *nature* **2015**, *521*, 436–444.

[7] ACD/Structure Elucidator, Version 14.0; Advanced Chemistry Development, Inc.: Ontario, Canada. **2014**,

[8] Elyashberg, M.; Argyropoulos, D. Computer Assisted Structure Elucidation (CASE): Current and future perspectives. *Magnetic Resonance in Chemistry*

[9] Goodfellow, I.; Bengio, Y.; Courville, A.; Bengio, Y. *Deep learning*; MIT press Cambridge, 2016; Vol. 1.

[10] Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A., et al. Mastering the game of go without human knowledge. *nature* **2017**, *550*, 354–359.

[11] Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A., et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165* **2020**,

[12] Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial nets. *Advances in neural information processing systems* **2014**, *27*, 2672–2680.

[13] Sutton, R. S.; Barto, A. G. *Reinforcement learning: An introduction*; MIT press, 2018.

[14] Zhou, Z.; Kearnes, S.; Li, L.; Zare, R. N.; Riley, P. Optimization of molecules via deep reinforcement learning. *Scientific reports* **2019**, *9*, 1–10.

[15] Popova, M.; Isayev, O.; Tropsha, A. Deep reinforcement learning for de novo drug design. *Science advances* **2018**, *4*, eaap7885.

[16] Sanchez-Lengeling, B.; Aspuru-Guzik, A. Inverse molecular design using machine learning: Generative models for matter engineering. *Science* **2018**, *361*, 360–365.

[17] Kwon, Y.; Lee, D.; Choi, Y.-S.; Kang, M.; Kang, S. Neural Message Passing for NMR Chemical Shift Prediction. *Journal of Chemical Information and Modeling* **2020**, *60*, 2024–2030.

[18] Wei, J. N.; Belanger, D.; Adams, R. P.; Sculley, D. Rapid prediction of electron–ionization mass spectrometry using neural networks. *ACS central science* **2019**, *5*, 700–708.

[19] Fine, J. A.; Rajasekar, A. A.; Jethava, K. P.; Chopra, G. Spectral deep learning for prediction and prospective validation of functional groups. *Chem. Sci.* **2020**, *11*, 4618–4630.

[20] McCarthy, M.; Lee, K. L. K. Molecule Identification with Rotational Spectroscopy and Probabilistic Deep Learning. *The Journal of Physical Chemistry A* **2020**, *124*, 3002–3017.

[21] Faulkner, D. J. Marine natural products. *Nat. Prod. Rep.* **2002**, *19*, 1–49.

[22] Carroll, A. R.; Copp, B. R.; Davis, R. A.; Keyzers, R. A.; Prinsep, M. R. Marine natural products. *Nat. Prod. Rep.* **2019**, *36*, 122–173.

[23] Lee, J.; Lee, Y.; Kim, J.; Kosiorek, A.; Choi, S.; Teh, Y. W. Set transformer: A framework for attention-based permutation-invariant neural networks. International Conference on Machine Learning. 2019; pp 3744–3753.

[24] Zaheer, M.; Kottur, S.; Ravanbakhsh, S.; Poczos, B.; Salakhutdinov, R.; Smola, A. Deep Sets. *arXiv* **2017**, *1703.06114*.

[25] Wagstaff, E.; Fuchs, F. B.; Engelcke, M.; Posner, I.; Osborne, M. On the Limitations of Representing Functions on Sets. 2019.

[26] Skianis, K.; Nikolentzos, G.; Limnios, S.; Vazirgiannis, M. Rep the Set: Neural Networks for Learning Set Representations. 2020.

[27] Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; Philip, S. Y. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems* **2020**,

[28] Xiong, Z.; Wang, D.; Liu, X.; Zhong, F.; Wan, X.; Li, X.; Li, Z.; Luo, X.; Chen, K.; Jiang, H.; Zheng, M. Pushing the Boundaries of Molecular Representation for Drug Discovery with the Graph Attention Mechanism. *Journal of Medicinal Chemistry* **2020**, *63*, 8749–8760.

[29] Coley, C. W.; Barzilay, R.; Green, W. H.; Jaakkola, T. S.; Jensen, K. F. Convolutional embedding of attributed molecular graphs for physical property prediction. *Journal of chemical information and modeling* **2017**, *57*, 1757–1772.

[30] Ertl, P.; Lewis, R.; Martin, E.; Polyakov, V. In silico generation of novel, drug-like chemical matter using the LSTM neural network. *arXiv preprint arXiv:1712.07449* **2017**,

[31] Prykhodko, O.; Johansson, S. V.; Kotsias, P.-C.; Arús-Pous, J.; Bjerrum, E. J.; Engkvist, O.; Chen, H. A de novo molecular generation method using latent vector based generative adversarial network. *Journal of Cheminformatics* **2019**, *11*, 74.

[32] Gómez-Bombarelli, R.; Wei, J. N.; Duvenaud, D.; Hernández-Lobato, J. M.; Sánchez-Lengeling, B.; Sheberla, D.; Aguilera-Iparraguirre, J.; Hirzel, T. D.; Adams, R. P.; Aspuru-Guzik, A. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science* **2018**, *4*, 268–276.

[33] Weininger, D. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *Journal of chemical information and computer sciences* **1988**, *28*, 31–36.

[34] Krenn, M.; Häse, F.; Nigam, A.; Friederich, P.; Aspuru-Guzik, A. Self-Referencing Embedded Strings (SELFIES): A 100% robust molecular string representation. *Machine Learning: Science and Technology* **2020**, *1*, 045024.

[35] Dalke, A. DeepSMILES: An Adaptation of SMILES for Use in Machine-Learning of Chemical Structures. *ChemArxiv preprint ChemArxiv* **2018**,

[36] Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; Polosukhin, I. Attention Is All You Need. *arXiv* **2017**, *1706.03762*.

[37] Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural computation* **1997**, *9*, 1735–1780.

[38] Devlin, J.; Chang, M.-W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv* **2018**, *1810.04805*.

[39] Peters, M. E.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; Zettlemoyer, L. Deep contextualized word representations. *arXiv* **2018**, *1802.05365*.

[40] Chen, T.; Guestrin, C. Xgboost: A scalable tree boosting system. Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. 2016; pp 785–794.

[41] Hastie, T.; Tibshirani, R.; Friedman, J. *The elements of statistical learning: data mining, inference, and prediction*; Springer Science & Business Media, 2009.

[42] Krizhevsky, A.; Sutskever, I.; Hinton, G. E. Imagenet classification with deep convolutional neural networks. *Communications of the ACM* **2017**, *60*, 84–90.

[43] Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; Fei-Fei, L. Imagenet: A large-scale hierarchical image database. 2009 IEEE conference on computer vision and pattern recognition. 2009; pp 248–255.

[44] Bengio, Y.; Courville, A.; Vincent, P. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence* **2013**, *35*, 1798–1828.

[45] Hinton, G. E.; Zemel, R. S. Autoencoders, minimum description length and Helmholtz free energy. Advances in neural information processing systems. 1994; pp 3–10.

[46] Ranzato, M.; Poultney, C.; Chopra, S.; Cun, Y. L. Efficient learning of sparse representations with an energy-based model. Advances in neural information processing systems. 2007; pp 1137–1144.

[47] Hinton, G. E.; Salakhutdinov, R. R. Reducing the Dimensionality of Data with Neural Networks. *Science* **2006**, *313*, 504–507.

[48] Lawrence, S.; Giles, C. L.; Tsoi, A. C.; Back, A. D. Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks* **1997**, *8*, 98–113.

[49] Badrinarayanan, V.; Kendall, A.; Cipolla, R. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence* **2017**, *39*, 2481–2495.

[50] Nam, H.; Han, B. Learning multi-domain convolutional neural networks for visual tracking. Proceedings of the IEEE conference on computer vision and pattern recognition. 2016; pp 4293–4302.

[51] Zhu, Z.; Liang, D.; Zhang, S.; Huang, X.; Li, B.; Hu, S. Traffic-Sign Detection and Classification in the Wild. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016; pp 2110–2118.

[52] Badue, C.; Guidolini, R.; Carneiro, R. V.; Azevedo, P.; Cardoso, V. B.; Forechi, A.; Jesus, L.; Berriel, R.; Paixão, T. M.; Mutz, F.; de Paula Veronese, L.; Oliveira-Santos, T.; De Souza, A. F. Self-driving cars: A survey. *Expert Systems with Applications* **2021**, *165*, 113816.

[53] Agostinelli, F.; McAleer, S.; Shmakov, A.; Baldi, P. Solving the Rubik's cube with deep reinforcement learning and search. *Nature Machine Intelligence* **2019**, *1*, 356–363.

[54] Vinyals, O.; Babuschkin, I.; Czarnecki, W. M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D. H.; Powell, R.; Ewalds, T.; Georgiev, P., et al. Grand-

master level in StarCraft II using multi-agent reinforcement learning. *Nature* **2019**, *575*, 350–354.

[55] Schrittwieser, J.; Antonoglou, I.; Hubert, T.; Simonyan, K.; Sifre, L.; Schmitt, S.; Guez, A.; Lockhart, E.; Hassabis, D.; Graepel, T., et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature* **2020**, *588*, 604–609.

[56] Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G., et al. Human-level control through deep reinforcement learning. *nature* **2015**, *518*, 529–533.

[57] Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I. Language models are unsupervised multitask learners. *OpenAI Blog* **2019**, *1*, 9.

[58] Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient Estimation of Word Representations in Vector Space. *arXiv* **2013**, *1301.3781*.

[59] Cho, K.; van Merrienboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv* **2014**, *1406.1078*.

[60] Bryson, A. E. *Applied optimal control: optimization, estimation and control*; CRC Press, 1975.

[61] Francois, C. Deep learning with Python. 2017.

[62] Venkatesan, R.; Li, B. *Convolutional neural networks in visual computing: a concise guide*; CRC Press, 2017.

[63] Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* **2014**, *1409.1556*.

[64] Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. Proceedings

of the IEEE conference on computer vision and pattern recognition. 2015; pp 1–9.

[65] Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the inception architecture for computer vision. Proceedings of the IEEE conference on computer vision and pattern recognition. 2016; pp 2818–2826.

[66] Sun, Q.; Liu, Y.; Chua, T.-S.; Schiele, B. Meta-Transfer Learning for Few-Shot Learning. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2019.

[67] Lafferty, J.; McCallum, A.; Pereira, F. C. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. **2001**, 282–289.

[68] Elman, J. L. Finding Structure in Time. *Cognitive Science* **1990**, *14*, 179–211.

[69] Graves, A.; Mohamed, A.-R.; Hinton, G. Speech recognition with deep recurrent neural networks. 2013 IEEE international conference on acoustics, speech and signal processing. 2013; pp 6645–6649.

[70] Eck, D.; Schmidhuber, J. Finding temporal structure in music: Blues improvisation with LSTM recurrent networks. Proceedings of the 12th IEEE workshop on neural networks for signal processing. 2002; pp 747–756.

[71] Sutskever, I.; Vinyals, O.; Le, Q. V. Sequence to Sequence Learning with Neural Networks. *arXiv* **2014**, *1409.3215*.

[72] Graves, A.; Schmidhuber, J. Offline handwriting recognition with multidimensional recurrent neural networks. *Advances in neural information processing systems* **2008**, *21*, 545–552.

[73] Cho, K.; van Merrienboer, B.; Bahdanau, D.; Bengio, Y. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. *arXiv* **2014**, *1409.1259*.

[74] Graves, A.; Schmidhuber, J. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural networks* **2005**, *18*, 602–610.

[75] Bisk, Y.; Holtzman, A.; Thomason, J.; Andreas, J.; Bengio, Y.; Chai, J.; Lapata, M.; Lazaridou, A.; May, J.; Nisnevich, A.; Pinto, N.; Turian, J. Experience Grounds Language. *arXiv* **2020**, *2004.10151*.

[76] Zhou, Z.; Guan, H.; Bhat, M.; Hsu, J. Fake News Detection via NLP is Vulnerable to Adversarial Attacks. *Proceedings of the 11th International Conference on Agents and Artificial Intelligence* **2019**, *40800*.

[77] Mehta, Y.; Majumder, N.; Gelbukh, A.; Cambria, E. Recent Trends in Deep Learning Based Personality Detection. *arXiv* **2019**, *1908.03628*.

[78] Papineni, K.; Roukos, S.; Ward, T.; Zhu, W.-J. Bleu: a Method for Automatic Evaluation of Machine Translation. Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics. Philadelphia, Pennsylvania, USA, 2002; pp 311–318.

[79] Adiwardana, D.; Luong, M.-T.; So, D. R.; Hall, J.; Fiedel, N.; Thoppilan, R.; Yang, Z.; Kulshreshtha, A.; Nemade, G.; Lu, Y.; Le, Q. V. Towards a Human-like Open-Domain Chatbot. *arXiv* **2020**, *2001.09977*.

[80] Rajpurkar, P.; Zhang, J.; Lopyrev, K.; Liang, P. SQuAD: 100,000+ Questions for Machine Comprehension of Text. *arXiv* **2016**, *1606.05250*.

[81] Zellers, R.; Bisk, Y.; Schwartz, R.; Choi, Y. SWAG: A Large-Scale Adversarial Dataset for Grounded Commonsense Inference. *arXiv* **2018**, *1808.05326*.

[82] Paperno, D.; Kruszewski, G.; Lazaridou, A.; Pham, Q.; Bernardi, R.; Pezzelle, S.; Baroni, M.; Boleda, G.; Fernández, R. The LAMBADA dataset: Word prediction requiring a broad discourse context. 2016; pp 1525–1534.

[83] Hu, J.; Ruder, S.; Siddhant, A.; Neubig, G.; Firat, O.; Johnson, M. XTREME: A Massively Multilingual Multi-task Benchmark for Evaluating Cross-lingual Generalization. *arXiv* **2020**, *2003.11080*.

[84] Reiter, E. A Structured Review of the Validity of BLEU. *Computational Linguistics* **2018**, *44*, 393–401.

[85] Sulem, E.; Abend, O.; Rappoport, A. BLEU is Not Suitable for the Evaluation of Text Simplification. Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. Brussels, Belgium, 2018; pp 738–744.

[86] Wang, A.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; Bowman, S. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP. Brussels, Belgium, 2018; pp 353–355.

[87] Wang, A.; Pruksachatkun, Y.; Nangia, N.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; Bowman, S. R. SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems. *arXiv* **2019**, *1905.00537*.

[88] Searle, J. The Rediscovery of the Mind. **1992**,

[89] Kalchbrenner, N.; Blunsom, P. Recurrent Continuous Translation Models. Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. Seattle, Washington, USA, 2013; pp 1700–1709.

[90] Pennington, J.; Socher, R.; Manning, C. Glove: Global Vectors for Word Representation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Doha, Qatar, 2014; pp 1532–1543.

[91] Bahdanau, D.; Cho, K.; Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv* **2014**, *1409.0473*.

[92] Luong, M.-T.; Pham, H.; Manning, C. D. Effective Approaches to Attention-based Neural Machine Translation. *arXiv* **2015**, *1508.04025*.

[93] Britz, D.; Goldie, A.; Luong, M.-T.; Le, Q. Massive Exploration of Neural Machine Translation Architectures. *arXiv* **2017**, *1703.03906*.

[94] Gregor, K.; Danihelka, I.; Graves, A.; Rezende, D. J.; Wierstra, D. DRAW: A Recurrent Neural Network For Image Generation. *arXiv* **2015**, *1502.04623*.

[95] Kim, Y.; Denton, C.; Hoang, L.; Rush, A. M. Structured Attention Networks. *arXiv* **2017**, *1702.00887*.

[96] Tu, Z.; Lu, Z.; Liu, Y.; Liu, X.; Li, H. Modeling Coverage for Neural Machine Translation. *arXiv* **2016**, *1601.04811*.

[97] Liang, P.; Jordan, M.; Klein, D. Learning Semantic Correspondences with Less Supervision. Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP. Suntec, Singapore, 2009; pp 91–99.

[98] Konstas, I.; Lapata, M. A Global Model for Concept-to-Text Generation. *J. Artif. Int. Res.* **2013**, *48*, 305–346.

[99] Mei, H.; Bansal, M.; Walter, M. R. What to talk about and how? Selective Generation using LSTMs with Coarse-to-Fine Alignment. *arXiv* **2015**, *1509.00838*.

[100] Vinyals, O.; Toshev, A.; Bengio, S.; Erhan, D. Show and Tell: A Neural Image Caption Generator. *arXiv* **2014**, *1411.4555*.

[101] Denkowski, M.; Lavie, A. Meteor 1.3: Automatic Metric for Reliable Optimization and Evaluation of Machine Translation Systems. Proceedings of the Sixth Workshop on Statistical Machine Translation. Edinburgh, Scotland, 2011; pp 85–91.

[102] Chorowski, J.; Bahdanau, D.; Serdyuk, D.; Cho, K.; Bengio, Y. Attention-Based Models for Speech Recognition. *arXiv* **2015**, *1506.07503*.

[103] Rush, A. M.; Chopra, S.; Weston, J. A Neural Attention Model for Abstractive Sentence Summarization. Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. Lisbon, Portugal, 2015; pp 379–389.

[104] See, A.; Liu, P. J.; Manning, C. D. Get To The Point: Summarization with Pointer-Generator Networks. *arXiv* **2017**, *1704.04368*.

[105] Vinyals, O.; Fortunato, M.; Jaitly, N. Pointer Networks. *arXiv* **2015**, *1506.03134*.

[106] Lin, Z.; Feng, M.; dos Santos, C. N.; Yu, M.; Xiang, B.; Zhou, B.; Bengio, Y. A Structured Self-attentive Sentence Embedding. *arXiv* **2017**, *1703.03130*.

[107] Cheng, J.; Dong, L.; Lapata, M. Long Short-Term Memory-Networks for Machine Reading. *arXiv* **2016**, *1601.06733*.

[108] Liu, Y.; Sun, C.; Lin, L.; Wang, X. Learning Natural Language Inference using Bidirectional LSTM model and Inner-Attention. *arXiv* **2016**, *1605.09090*.

[109] Zhang, H.; Goodfellow, I.; Metaxas, D.; Odena, A. Self-Attention Generative Adversarial Networks. *arXiv* **2018**, *1805.08318*.

[110] Mishra, N.; Rohaninejad, M.; Chen, X.; Abbeel, P. A Simple Neural Attentive Meta-Learner. *arXiv* **2017**, *1707.03141*.

[111] Bengio, Y. The Consciousness Prior. *arXiv* **2017**, *1709.08568*.

[112] Lee, J. B.; Rossi, R. A.; Kim, S.; Ahmed, N. K.; Koh, E. Attention Models in Graphs: A Survey. *arXiv* **2018**, *1807.07984*.

[113] Ba, J.; Mnih, V.; Kavukcuoglu, K. Multiple Object Recognition with Visual Attention. *arXiv* **2014**, *1412.7755*.

[114] Mnih, V.; Heess, N.; Graves, A.; kavukcuoglu, k. In *Advances in Neural Information Processing Systems 27*; Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., Weinberger, K. Q., Eds.; Curran Associates, Inc., 2014; pp 2204–2212.

[115] Oktay, O.; Schlemper, J.; Folgoc, L. L.; Lee, M.; Heinrich, M.; Misawa, K.; Mori, K.; McDonagh, S.; Hammerla, N. Y.; Kainz, B.; Glocker, B.; Rueckert, D. Attention U-Net: Learning Where to Look for the Pancreas. *arXiv* **2018**, *1804.03999*.

[116] Anderson, P.; He, X.; Buehler, C.; Teney, D.; Johnson, M.; Gould, S.; Zhang, L. Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering. *arXiv* **2017**, *1707.07998*.

[117] Yang, Z.; Dai, Z.; Yang, Y.; Carbonell, J.; Salakhutdinov, R.; Le, Q. V. XLNet: Generalized Autoregressive Pretraining for Language Understanding. *arXiv* **2019**, *1906.08237*.

[118] Dai, Z.; Yang, Z.; Yang, Y.; Carbonell, J.; Le, Q. V.; Salakhutdinov, R. Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context. *arXiv* **2019**, *1901.02860*.

[119] Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; Liu, P. J. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *arXiv* **2019**, *1910.10683*.

[120] Dehghani, M.; Gouws, S.; Vinyals, O.; Uszkoreit, J.; Łukasz Kaiser, Universal Transformers. *arXiv* **2018**, *1807.03819*.

[121] Jiao, X.; Yin, Y.; Shang, L.; Jiang, X.; Chen, X.; Li, L.; Wang, F.; Liu, Q. TinyBERT: Distilling BERT for Natural Language Understanding. *arXiv* **2019**, *1909.10351*.

[122] Kitaev, N.; Łukasz Kaiser,; Levskaya, A. Reformer: The Efficient Transformer. *arXiv* **2020**, *2001.04451*.

[123] Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; Stoyanov, V. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv* **2019**, *1907.11692*.

[124] Lan, Z.; Chen, M.; Goodman, S.; Gimpel, K.; Sharma, P.; Soricut, R. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. *arXiv* **2019**, *1909.11942*.

[125] Sanh, V.; Debut, L.; Chaumond, J.; Wolf, T. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv* **2019**, *1910.01108*.

[126] Clark, K.; Luong, M.-T.; Le, Q. V.; Manning, C. D. ELECTRA: Pretraining Text Encoders as Discriminators Rather Than Generators. *arXiv* **2020**, *2003.10555*.

[127] Vinyals, O.; Bengio, S.; Kudlur, M. Order Matters: Sequence to sequence for sets. *arXiv* **2015**, *1511.06391*.

[128] Qi, C. R.; Su, H.; Mo, K.; Guibas, L. J. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. *arXiv* **2016**, *1612.00593*.

[129] Snell, J.; Swersky, K.; Zemel, R. S. Prototypical Networks for Few-shot Learning. *arXiv* **2017**, *1703.05175*.

[130] Finn, C.; Abbeel, P.; Levine, S. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. *arXiv* **2017**, *1703.03400*.

[131] Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; Dahl, G. E. Neural Message Passing for Quantum Chemistry. *arXiv* **2017**, *1704.01212*.

[132] Edwards, H.; Storkey, A. Towards a Neural Statistician. *arXiv* **2016**, *1606.02185*.

[133] Newman, D. J.; Cragg, G. M. Natural Products as Sources of New Drugs from 1981 to 2014. *Journal of Natural Products* **2016**, *79*, 629–661.

[134] Jiménez, C.; Riguera, R. Phenylethanoid glycosides in plants: structure and biological activity. *Nat. Prod. Rep.* **1994**, *11*, 591–606.

[135] Behrman, E. J.; Gopalan, V. Cholesterol and Plants. *Journal of Chemical Education* **2005**, *82*, 1791.

[136] Jaroszewski, J. W. Hyphenated NMR techniques and NMR-based metabolomics in studies of medicinal plants. *Planta Med* **2007**, *73*, 986714.

[137] Jaroszewski, J. W. Hyphenated NMR Methods in Natural Products Research, Part 2: HPLC-SPE-NMR and Other New Trends in NMR Hyphenation. *Planta Med* **2005**, *71*, 795–802.

[138] Breton, R. C.; Reynolds, W. F. Using NMR to identify and characterize natural products. *Nat. Prod. Rep.* **2013**, *30*, 501–524.

[139] Exarchou, V.; Krucker, M.; van Beek, T.; Vervoort, J.; Gerothanassis, I.; Albert, K. LC-NMR coupling technology: Recent advancements and applications in natural products analysis. *Magnetic resonance in chemistry : MRC* **2005**, *43*, 681–7.

[140] Nielsen, K. F.; Månsson, M.; Rank, C.; Frisvad, J. C.; Larsen, T. O. Dereplication of Microbial Natural Products by LC-DAD-TOFMS. *Journal of Natural Products* **2011**, *74*, 2338–2348.

[141] Plainchont, B.; de Paulo Emerenciano, V.; Nuzillard, J.-M. Recent advances in the structure elucidation of small organic molecules by the LSD software. *Magnetic Resonance in Chemistry* **2013**, *51*, 447–453.

[142] Meiler, J.; Meusinger, R.; Will, M. Fast determination of 13C NMR chemical shifts using artificial neural networks. *Journal of chemical information and computer sciences* **2000**, *40*, 1169–1176.

[143] Kuhn, S.; Egert, B.; Neumann, S.; Steinbeck, C. Building blocks for automated

elucidation of metabolites: Machine learning methods for NMR prediction. *BMC bioinformatics* **2008**, *9*, 400.

[144] Brown, N.; Fiscato, M.; Segler, M. H.; Vaucher, A. C. GuacaMol: benchmarking models for de novo molecular design. *Journal of chemical information and modeling* **2019**, *59*, 1096–1108.

[145] Meiler, J.; Will, M. Genius: a genetic algorithm for automated structure elucidation from 13C NMR spectra. *Journal of the American Chemical Society* **2002**, *124*, 1868–1870.

[146] Dzeroski, S.; Schulze-Kremer, S.; Heidtke, K. R.; Siems, K.; Wettschereck, D.; Blockeel, H. Diterpene structure elucidation from 13cnmr spectra with inductive logic programming. *Applied Artificial Intelligence* **1998**, *12*, 363–383.

[147] Ferreira, M. J.; Brant, A. J.; Rodrigues, G. V.; Emerenciano, V. P. Automatic identification of terpenoid skeletons through 13C nuclear magnetic resonance data disfunctionalization. *Analytica chimica acta* **2001**, *429*, 151–170.

[148] Pereira, F. 1D 13C-NMR Data as Molecular Descriptors in Spectra—Structure Relationship Analysis of Oligosaccharides. *Molecules* **2012**, *17*, 3818–3833.

[149] Ferreira, M. J.; Costantin, M. B.; Rodrigues, G. V.; Emerenciano, V. P. Application of a new program, H1MACH, for prediction of iridoid skeletons. *Spectroscopy letters* **2004**, *37*, 587–605.

[150] Plainchont, B.; Nuzillard, J.-M.; Rodrigues, G. V.; Ferreira, M. J.; Scotti, M. T.; Emerenciano, V. P. New improvements in automatic structure elucidation using the LSD (logic for structure determination) and the sistemat expert systems. *Natural Product Communications* **2010**, *5*, 1934578X1000500517.

[151] Boiteau, R. M.; Hoyt, D. W.; Nicora, C. D.; Kinmonth-Schultz, H. A.; Ward, J. K.; Bingol, K. Structure elucidation of unknown metabolites in

metabolomics by combined NMR and MS/MS prediction. *Metabolites* **2018**, *8*, 8.

[152] Harn, Y.-C.; Su, B.-H.; Ku, Y.-L.; Lin, O. A.; Chou, C.-F.; Tseng, Y. J. NP-StructurePredictor: Prediction of Unknown Natural Products in Plant Mixtures. *Journal of Chemical Information and Modeling* **2017**, *57*, 3138–3148.

[153] Jeannerat, D. Human-and computer-accessible 2D correlation data for a more reliable structure determination of organic compounds. Future roles of researchers, software developers, spectrometer managers, journal editors, reviewers, publisher and database managers toward artificial-intelligence analysis of NMR spectra. *Magnetic Resonance in Chemistry* **2017**, *55*, 7–14.

[154] López-Pérez, J. L.; Therón, R.; del Olmo, E.; Díaz, D. NAPROC-13: a database for the dereplication of natural product mixtures in bioassay-guided protocols. *Bioinformatics* **2007**, *23*, 3256–3257.

[155] Steinbeck, C.; Krause, S.; Kuhn, S. NMRShiftDB constructing a free chemical information system with open-source components. *Journal of chemical information and computer sciences* **2003**, *43*, 1733–1739.

[156] Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V., et al. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research* **2011**, *12*, 2825–2830.

[157] Sokolova, M.; Lapalme, G. A systematic analysis of performance measures for classification tasks. *Information processing & management* **2009**, *45*, 427–437.

[158] Smith, M. R.; Martinez, T.; Giraud-Carrier, C. An instance level analysis of data complexity. *Machine learning* **2014**, *95*, 225–256.

[159] Mani, I.; Zhang, I. kNN approach to unbalanced data distributions: a case

study involving information extraction. Proceedings of workshop on learning from imbalanced datasets. 2003.

[160] Han, H.; Wang, W.-Y.; Mao, B.-H. Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning. International conference on intelligent computing. 2005; pp 878–887.

[161] He, H.; Bai, Y.; Garcia, E. A.; Li, S. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. 2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence). 2008; pp 1322–1328.

[162] Lemaître, G.; Nogueira, F.; Aridas, C. K. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *The Journal of Machine Learning Research* **2017**, *18*, 559–563.

[163] Chen, D.-L.; Li, G.; Liu, Y.-Y.; Ma, G.-X.; Zheng, W.; Sun, X.-B.; Xu, X.-D. A new cadinane sesquiterpenoid glucoside with cytotoxicity from Abelmoschus sagittifolius. *Natural product research* **2019**, *33*, 1699–1704.

[164] Zhang, S.-N.; Zeng, J.; Tan, Y.-N.; Ma, R.-J.; Zhang, G.-J.; Wang, H.-S.; Tan, Q.-G. 3 α, 19-Dihydroxyl-ent-pimara-8 (14), 15-diene, a new diterpenoid from the rhizomes of Ricinus communi s. *Journal of Asian natural products research* **2019**, *21*, 522–527.

[165] Shi, Z.-N.; Wang, Y.-D.; Gong, Y.; Li, H.-F.; Zhu, Y. New triterpenoid saponins with cytotoxic activities from Ligularia przewalskii. *Phytochemistry Letters* **2019**, *30*, 215–219.

[166] Yu, H.-L.; Long, Q.; Yi, W.-F.; Yang, B.-J.; Song, Y.; Ding, X.; Li, S.-L.; Hao, X.-J. Two New C21 Steroidal Glycosides from the Roots of Cynanchum paniculatum. *Natural Products and Bioprospecting* **2019**, *9*, 209–214.

[167] Hou, Y.; Jin, C.; An, R.; Yin, X.; Piao, Y.; Yin, X.; Jin, L.; Zhang, C. A new

flavonoid from the stem bark of Acer tegmentosum. *Biochemical Systematics and Ecology* **2019**, *83*, 1–3.

[168] Ji, J.; Wang, Q.; Wang, M.; Chen, J.; Li, X. New lignan glycosides from the stems of Securidaca inappendiculata Hassk. *Phytochemistry Letters* **2019**, *31*, 58–62.

[169] Wu, J.; Zhang, B.-J.; Bao, M.-F.; Cai, X.-H. A new erythrinan N-oxide alkaloid from Erythrina stricta. *Natural product research* **2019**, *33*, 2004–2010.

[170] Lai, W.; Qin, S.-Y.; Zou, G.; Liao, X.-J.; Chen, G.-D.; Zhang, H.; Zhao, B.-X.; Xu, S.-H. Sinulaspirolactam A, a novel aza-spirocyclic valerenane sesquiterpenoid from soft coral Sinularia sp. *Journal of Asian natural products research* **2019**, *21*, 494–500.

[171] Adelakun, T. A.; Ding, X.; Ombati, R. M.; Zhao, N.-D.; Obodozie-Ofoegbu, O. O.; Di, Y.-T.; Zhang, Y.; Hao, X.-J. A new highly oxygenated abietane diterpenoid and a new lysosome generating phorbol ester from the roots of Euphorbia fischeriana Steud. *Natural product research* **2020**, *34*, 3027–3035.

[172] Lei, Z.; Zou, G.; Gao, Y.; Yao, Y.; Peng, C.; Shu, J.; Yang, M. A new triterpenoid and a new flavonoid glycoside isolated from Bupleurum marginatum and their anti-inflammatory activity. *Natural product research* **2020**, *34*, 3492–3498.

[173] Ribeiro, P. R.; Ferraz, C. G.; Cruz, F. G. New steroid and other compounds from non-polar extracts of Clusia burle-marxii and their chemotaxonomic significance. *Biochemical Systematics and Ecology* **2019**, *82*, 31–34.

[174] Zhao, M.-Z.; Shen, Y.; Xu, W.; Chen, Y.-Z.; Xiao, C.-J.; Jiang, B. A new lignan glycoside from Astragalus yunnanensis. *Journal of Asian natural products research* **2019**,

[175] Ji, Y.; Wang, C.; Zhang, Y.; Zhang, C.; Cui, D.; Zhang, X. Glutinosine A: A New Morphinandienone Alkaloid from Litsea glutinosa. *Records of Natural Products* **2019**, *13*, 366.

[176] Deng, L.; Wang, R.; Wang, G.; Liu, M.; Liao, Z.; Liao, G.; Chen, M. Roseosporol A, the first isolation of a novel sesquiterpenoid from Streptomyces roseosporus. *Natural product research* **2019**, *33*, 2038–2043.

[177] Wang, X.-D.; Han, Q.-H.; Zhang, J.; Zhang, Q.-Y.; Tu, P.-F.; Liang, H. Three new triterpenoid saponins from Albizia julibrissin. *Journal of Asian natural products research* **2019**, *21*, 535–541.

[178] Guo, W.; Liu, W.; Xiao, F.; Zhang, M.; Li, L.; Yang, P.; Chen, C. New Cytotoxic Steroid Produced by the Soil-Derived Fungus Aspergillus flavus JDW-1. *Natural Product Communications* **2019**, *14*, 1934578X19850376.

[179] de Novais, L. M.; de Arueira, C. C.; Ferreira, L. F.; Ribeiro, T. A.; Sousa Jr, P. T.; Jacinto, M. J.; de Carvalho, M. G.; Judice, W. A.; Jesus, L. O.; de Souza, A. A., et al. 4-Hydroxy-6, 7-methylenedioxy-3-methoxyflavone: A novel flavonoid from Dulacia egleri with potential inhibitory activity against cathepsins B and L. *Fitoterapia* **2019**, *132*, 26–29.

[180] Wang, G.-K.; Jin, W.-F.; Zhang, N.; Wang, G.; Cheng, Y.-Y.; Morris-Natschke, S. L.; Goto, M.; Zhou, Z.-Y.; Liu, J.-S.; Lee, K.-H. Kalshiolin A, new lignan from Kalimeris shimadai. *Journal of Asian Natural Products Research* **2020**, *22*, 489–495.

[181] Si, Y.-Y.; Tang, M.-X.; Lin, S.; Chen, G.; Hua, H.-M.; Bai, J.; Wang, Y.-B.; Wang, H.-F.; Pei, Y.-H. 2-Methyl-versiquinazoline C, a new fumiquinazoline-type alkaloid from the fungus Aspergillus flavipes PJ03-11. *Journal of Asian natural products research* **2019**, *21*, 528–534.

[182] Ding, J.-H.; Li, Z.-H.; Feng, T.; Liu, J.-K. A new tremulane sesquiterpenoid from the fungus Irpex lacteus. *Natural product research* **2019**, *33*, 316–320.

[183] Yannick Stephane, F. F.; Dawe, A.; Angelbert Fusi, A.; Jean Jules, B. K.; Ulrich, K. K. D.; Lateef, M.; Bruno, L. N.; Ali, M. S.; Ngouela, S. A. Crotoliganfuran, a new clerodane-type furano-diterpenoid from Croton oligandrus Pierre ex Hutch. *Natural product research* **2019**, 1–9.

[184] Ding, K.; Guo, S.; Rong, W.; Li, Q.; Liu, R.; Xu, H.; Yin, Y.; Bi, K. A new oleanane type pentacyclic triterpenoid saponin from the husks of xanthoceras sorbifolium bunge and its neuroprotection on PC12 cells injury induced by Aβ25-35. *Natural Product Research* **2019**, 1–7.

[185] Li, J.; Wang, Z.; Yang, F.; Jiao, W.-H.; Lin, H.-W.; Xu, S.-H. Two new steroids with cytotoxicity from the marine sponge Dactylospongia elegans collected from the South China Sea. *Natural product research* **2019**, *33*, 1340–1344.

[186] Liu, Y.; Wang, W.-q.; Chen, T.; Xuan, L.-j. New flavonoid glycosides from seeds of Baccharoides anthelmintica. *Natural product research* **2020**, *34*, 284–289.

[187] Lee, Y.-G.; Seo, K.-H.; Gwag, J. E.; Kim, H.-G.; Ko, J.-H.; Lee, D. Y.; Baek, N.-I. New Lignan from the Flowers of Forsythia koreana. *Chemistry of Natural Compounds* **2019**, *55*, 432–434.

[188] Kitajima, M.; Okabe, K.; Yoshida, M.; Nakabayashi, R.; Saito, K.; Kogure, N.; Takayama, H. New otonecine-type pyrrolizidine alkaloid from Petasites japonicus. *Journal of natural medicines* **2019**, *73*, 602–607.

[189] Liu, S.-S.; Jiang, J.-X.; Huang, R.; Wang, Y.-T.; Jiang, B.-G.; Zheng, K.-X.; Wu, S.-H. A new antiviral 14-nordrimane sesquiterpenoid from an endophytic fungus Phoma sp. *Phytochemistry Letters* **2019**, *29*, 75–78.

[190] Luyen, N. T.; Binh, P. T.; Tham, P. T.; Hung, T. M.; Dang, N. H.; Dat, N. T.; Thao, N. P. Wedtrilosides A and B, two new diterpenoid glycosides from the leaves of Wedelia trilobata (L.) Hitchc. with α-amylase and α-glucosidase inhibitory activities. *Bioorganic Chemistry* **2019**, *85*, 319–324.

[191] Feng, T.-T.; Fu, H.-Z.; Yang, Y.-S.; Zhou, Z.-Q.; Dai, M.; Bi, H.-Y.; Wang, D. Two new noroleanane-type triterpenoid saponins from the stems of Stauntonia chinensis. *Natural product research* **2019**, *33*, 1269–1276.

[192] Liu, Z.; Dong, Z.; Qiu, P.; Wang, Q.; Yan, J.; Lu, Y.; Wasu, P.-a.; Hong, K.; She, Z. Two new bioactive steroids from a mangrove-derived fungus Aspergillus sp. *Steroids* **2018**, *140*, 32–38.

[193] Posri, P.; Suthiwong, J.; Takomthong, P.; Wongsa, C.; Chuenban, C.; Boonyarat, C.; Yenjai, C. A new flavonoid from the leaves of Atalantia monophylla (L.) DC. *Natural product research* **2019**, *33*, 1115–1121.

[194] Li, Y.; Qin, X.-B.; Liu, H.-X.; Xu, Z.-F.; Tan, H.-B.; Qiu, S.-X. Two pairs of enantiomeric propylated flavonoids and a new lignan from the aerial parts of Abrus precatorius. *Fitoterapia* **2019**, *133*, 125–129.

[195] Endo, Y.; Sugiura, Y.; Funasaki, M.; Kagechika, H.; Ishibashi, M.; Ohsaki, A. Two new alkaloids from Crinum asiaticum var. japonicum. *Journal of natural medicines* **2019**, *73*, 648–652.

[196] Choi, Y.-H.; Seo, C.; Jeong, W.; Lee, J. E.; Lee, J. Y.; Ahn, E.-K.; Kang, J.-S.; Lee, J.-H.; Choi, C. W.; Oh, J. S., et al. Glycopentanolones AD, four new geranylated quinolone alkaloids from Glycosmis pentaphylla. *Bioorganic Chemistry* **2019**, *87*, 714–719.

[197] Poumale, H. M. P.; Hamm, R.; Zang, Y.; Shiono, Y.; Kuete, V. *Medicinal plant research in Africa*; Elsevier, 2013; pp 261–300.

[198] Yang, Y.-x.; Wang, J.-x.; Wang, Q.; Li, H.-l.; Tao, M.; Luo, Q.; Liu, H. New chromane and chromene meroterpenoids from flowers of Rhododendron rubiginosum Franch. var. rubiginosum. *Fitoterapia* **2018**, *127*, 396–401.

[199] Lou, L.-L.; Zhao, P.; Cheng, Z.-Y.; Guo, R.; Yao, G.-D.; Wang, X.-B.; Huang, X.-X.; Song, S.-J. A new coumarin from Juglans mandshurica Maxim

induce apoptosis in hepatocarcinoma cells. *Natural product research* **2019**, *33*, 1791–1793.

[200] Sadorn, K.; Saepua, S.; Boonyuen, N.; Komwijit, S.; Rachtawee, P.; Pittayakhajonwut, P. Phenolic glucosides and chromane analogs from the insect fungus Conoideocrella krungchingensis BCC53666. *Tetrahedron* **2019**, *75*, 3463–3471.

[201] Tee, K. H.; Ee, G. C. L.; Ismail, I. S.; Karunakaran, T.; Teh, S. S.; Jong, V. Y. M.; Mohd Nor, S. M. A new coumarin from stem bark of Calophyllum wallichianum. *Natural product research* **2018**, *32*, 2565–2570.

[202] Duan, S.-L.; Zhang, J.; Han, Q.-H.; Zhang, Q.-Y.; Liang, H. A new coumarin and a new norlignan from Ficus tsiangii. *Journal of Asian natural products research* **2019**, *21*, 337–342.

[203] Bao, W.; Wang, Q.; Hao, J. Structural Elucidation of a Coumarin with New Skeleton from Artemisia ordosica. *Records of Natural Products* **2019**, *13*, 413–417.

[204] Sun, J.; Yu, J.-H.; Zhang, J.-S.; Song, X.-Q.; Bao, J.; Zhang, H. Chromane enantiomers from the flower buds of Tussilago farfara L. and assignments of their absolute configurations. *Chemistry & biodiversity* **2019**, *16*, e1800581.

[205] El-Sharkawy, E.; Selim, Y. Three new coumarin types from aerial parts of Ammi majus L. and their cytotoxic activity. *Zeitschrift für Naturforschung C* **2018**, *73*, 1–7.

[206] De Hoffmann, E. Mass spectrometry. *Kirk-Othmer Encyclopedia of Chemical Technology* **2000**,

[207] De Vijlder, T.; Valkenborg, D.; Lemière, F.; Romijn, E. P.; Laukens, K.; Cuyckens, F. A tutorial in small molecule identification via electrospray ionization-mass spectrometry: The practical art of structural elucidation. *Mass spectrometry reviews* **2018**, *37*, 607–629.

[208] Stein, S. E. Chemical substructure identification by mass spectral library searching. *Journal of the American Society for Mass Spectrometry* **1995**, *6*, 644–655.

[209] Stein, S. E.; Scott, D. R. Optimization and testing of mass spectral library search algorithms for compound identification. *Journal of the American Society for Mass Spectrometry* **1994**, *5*, 859–866.

[210] Tran, N. H.; Zhang, X.; Xin, L.; Shan, B.; Li, M. De novo peptide sequencing by deep learning. *Proceedings of the National Academy of Sciences* **2017**, *114*, 8247–8252.

[211] Schoenholz, S. S.; Hackett, S.; Deming, L.; Melamud, E.; Jaitly, N.; McAllister, F.; O'Brien, J.; Dahl, G.; Bennett, B.; Dai, A. M., et al. Peptide-spectra matching from weak supervision. *arXiv preprint arXiv:1808.06576* **2018**, *1808.06576*.

[212] Xu, K.; Ba, J.; Kiros, R.; Cho, K.; Courville, A.; Salakhutdinov, R.; Zemel, R.; Bengio, Y. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. *arXiv* **2015**, *1502.03044*.

[213] Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* **2020**, *2010.11929*.

[214] Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; Bengio, Y. Graph attention networks. *arXiv preprint arXiv:1710.10903* **2017**, *1710.10903*.

[215] Horai, H.; Arita, M.; Kanaya, S.; Nihei, Y.; Ikeda, T.; Suwa, K.; Ojima, Y.; Tanaka, K.; Tanaka, S.; Aoshima, K., et al. MassBank: a public repository for sharing mass spectral data for life sciences. *Journal of mass spectrometry* **2010**, *45*, 703–714.

[218] RDKit: Open-source cheminformatics; http://www.rdkit.org. **2020**,

[216] Paszke, A. et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems 32* **2019**, 8024–8035.

[217] Paleyes, A.; Gonzalez, J. GPyOpt: A Bayesian Optimization framework in python; http://github.com/SheffieldML/GPyOpt. 2016.

[219] Kingma, D. P.; Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* **2014**, *1412.6980*.

[220] Butina, D. Unsupervised data base clustering based on daylight's fingerprint and Tanimoto similarity: A fast and automated way to cluster small and large data sets. *Journal of Chemical Information and Computer Sciences* **1999**, *39*, 747–750.

[221] Rogers, D.; Hahn, M. Extended-connectivity fingerprints. *Journal of chemical information and modeling* **2010**, *50*, 742–754.

[222] Zhang, J.; Terayama, K.; Sumita, M.; Yoshizoe, K.; Ito, K.; Kikuchi, J.; Tsuda, K. NMR-TS: de novo molecule identification from NMR spectra. *Science and Technology of Advanced Materials* **2020**, *21*, 552–561.

[223] Norouzi, M.; Bengio, S.; Jaitly, N.; Schuster, M.; Wu, Y.; Schuurmans, D., et al. Reward augmented maximum likelihood for neural structured prediction. Advances In Neural Information Processing Systems. 2016; pp 1723–1731.