JOTA manuscript No.
(will be inserted by the editor)

Drugs discovery by shape similarity using Deep Learning

Felipe Romero, Luis F. Romero, Juana L. Redondo, Pilar M. Ortigosa

Received: date / Accepted: date

Abstract Searching for one or several molecules in a database using their shapes has great interest from a biochemical point of view, but requires a huge computational effort due to the complexity of the algorithms and the sizes of the databases in the pharmaceutical industry. This work uses Deep Learning by training neural networks with hundreds of images of each molecule, rendered by projections (using GPUs) on planes whose normal vectors are equally distributed in the 3D space (using Fibonacci spirals). The results obtained, both in accuracy and time, exceeded expectations, opening a hopeful path of research.

Keywords Deep Learning, Drugs discovery, Hybrid Computing, 3D Object Recognition, Embarrassingly Parallel

"For I frequently observed that towers, which at a distance seemed round, appeared square, when more closely viewed, and that colossal figures, raised on the summits of these towers, looked like small statues when viewed from the bottom of them."

Descartes, Discourse on Method

1 Introduction

The visual recognition of a 3D object is a process that requires, in most cases, a 2D projection of that object as a "preprocessing stage" before executing the recognition itself. For instance, this happens at the human eye's retina as an

Felipe Romero and Luis F. Romero
Computer Architecture. University of Málaga
Blvd Louis Pasteur, 29071, Spain
fr@uma.es, felipe@uma.es

Juana L. Redondo and Pilar M. Ortigosa
Informatics, University of Almeria, Spain
ortigosa@ual.es, jlredondo@ual.es

almost invisible previous step before the brain processes the received information, with or without the help of other images (a second eye, for example) or from other sensory organs. This also occurs when capturing an image in a photographic film or a CCD sensor before a machine or a person proceeds with the final recognition.

The animal capacity of object recognition, with no dependence on the spatial position of the obtained projection, including its six degrees of freedom, is astonishing. For example, when we see a remote control, we recognize it immediately, no matter its position on the table or our distance from it. Certain places make the recognition process successful, like the canonical perspective [22], although there is also a probability of failure. However, sometimes, we can learn from those failures and harness them to identify, for instance, similar objects.

A novel technique is presented in this work, which uses a high amount of 2D, low–resolution projections of three–dimensional molecules so that a neural network learns to identify them. This is not the first work to use different perspectives of an object to improve recognition via artificial intelligence. There are hundreds of studies, most compiled in [24], where other pictures or renderings were taken from an object. Only some of these studies, like RotationNet or VERAN [5, 15], render projections from different sphere coordinates. However, its scope is quite limited due to very few perspectives, and none exploit the embarrassing parallel of the projections, which is critical in preprocessing for deep learning.

In this work, isometric projections were employed. This decision was driven by the critical need to manage computational costs and dataset size, which are pivotal considerations in deep learning applications. Isometric projections ensure that the essential geometrical properties are preserved in the 2D representation. Alternatively, topological or Riemannian manifolds could be used to represent 3D molecules, as these approaches would provide a richer and more detailed representation of the 3D structures. However, the detailed and high-dimensional nature of these representations would require more sophisticated algorithms and greater computational power, leading to higher costs and smaller datasets.

The projection of molecules into low-resolution images results in a significant reduction in the precision of the information when considering a single image in isolation. However, similar to discrete tomography [7] and the Radon transform [26], hundreds or even thousands of images allow the neural network to recover a substantial amount of information associated with the original structure. This approach is also analogous to geometry reconstruction from remote sensing using machine learning techniques (such as random forest and support vector regression) [19], where the integration of data from multiple projections or images enhances the accuracy of three-dimensional reconstruction and analysis.

As a case study, we utilized a dataset of thousands of molecules from the full `DrugBank` Database [34] to train the neural network, but not to learn to recognize them, instead to search for similarities. As this is a preliminary study

in biochemistry, there is no predefined objective. However, it has shown to be an excellent tool in the search for drugs with similar structures to an existing one and, consequently, with similar pharmacological properties. Eventually, some drug functions with fewer side effects on the body could be achieved. Knowledge of another work that uses this technique or a similar one focused on drugs or molecule recognition has not been found.

2 Omnidirectional projections of a molecule

Each molecule can be represented as a set of atoms and bonds in three-dimensional space. Mathematically, we can denote a molecule M as a graph $G = (V, E)$ where:

- V is the set of vertices (atoms) with their coordinates (x_i, y_i, z_i).
- E is the set of edges (bonds) between the atoms.

Projections of the molecule at different angles can be modeled as linear transformations. Let P_i be the projection of molecule M in the plane from point i. The transformation can be represented using rotation matrices R_i,

$$P_i = R_i \cdot M. \tag{1}$$

As it was mentioned above, generating a low-resolution image dataset (lower than 256×256 pixels) is one of the objectives of this work. The images are obtained from multi-directional orthogonal projections of each molecule onto a plane. In other words, as if they were captured from cameras located onto an infinity radius sphere. And the first step of this proposed mechanism is selecting those points and their corresponding projection axes.

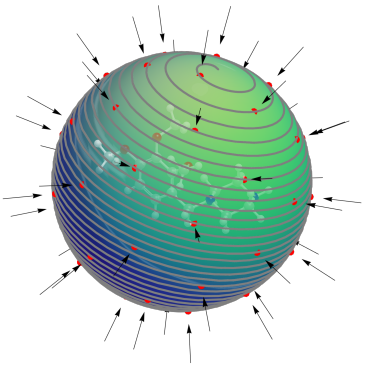

Fig. 1 Projection axes of a molecule using a Fibonacci spiral of 50 points onto a unit sphere.

Searching a set of N points, or tiles, which are equally distributed on a sphere surface is one of the more challenging tasks of geometry, not only for its

relevance in many science areas, such as Astronomy, Meteorology, Farming or Energy, but also for mere curiosity. Already Plato, in ancient Greece, proposed a possible solution for small values of N, such as the tetrahedron (4), the dodecahedron (12), or the icosahedron (20). But not even the platonic solids (or rather, their projection on the sphere in which they are inscribed) can give a correct answer to a question that is already ambiguous in its formulation [28]. What is considered an equitable distribution? The solution is usually sought using Euclidean distances (as in the *Tammes* problem). However, energies or fields can also be used (as in the *Thomson* problem, which distributes the electrons in the corresponding atomic model) [16].

Be that as it may, many algorithms generate approximate solutions for the calculation of points or spherical meshes, mainly due to their relevance in the numerical resolution of differential equations, in parallel computing, computational graphics, and even in space exploration, to cite a few [6], [31], [9]. It should not be forgotten that projects as extraordinarily costly as the GPS, Galileo, or Starlink programs depend mainly on an equitable distribution of satellites [11]. However, it is only sometimes reasonable to incur a high cost for calculating a fine mesh. It is preferable to use simple algorithms, such as Bauer's method [3], or even the elegant and efficient Fibonacci spherical spiral [8]. In the Fibonacci spiral, the sine of the points polar angle θ_i are equally distributed across $-\pi/2, \pi/2$,

$$\Delta \sin\theta = \frac{2}{N-1}, \tag{2}$$

so the polar angle is given by:

$$\theta_i = \arcsin\left(-1 + \frac{2i}{N-1}\right) \tag{3}$$

The *golden ratio*,

$$\phi = \frac{\dot{(}1-\sqrt{5})}{2}, \tag{4}$$

is used to calculate the point azimuthal angle,

$$\varphi_i = 2\pi\phi i. \tag{5}$$

Therefore, the cartesian coordinates of each projection axis are calculated by the transformations:

$$x_i = \cos\theta_i \cos\varphi_i;\ \ y_i = \cos\theta_i \sin\varphi_i;\ \ z_i = \sin\theta_i. \tag{6}$$

The spherical Fibonacci point sets are not the best distribution. For example, they do not match the platonic bodies. But they achieve excellent sampling properties and are extremely simple to construct compared to other more sophisticated spherical sampling schemes. A Fibonacci spiral created from 50 points is shown in Figure 1 [25].

Choosing the number of points on the sphere to project the molecules is complex and requires trial and error. It seems clear that 10 points or less are

insufficient and that making more than 1000 projections does not make much sense, even more so when the projected images will not have the resolution to be worthwhile. There is also the possibility of discarding the axes of a hemisphere, as the projections are similar to one side or the other. This idea has been discarded in this work for three reasons. First, the axes are not symmetric in the Fibonacci spherical spiral sequence. Secondly, and due to the chosen projection mechanism, the axes of the second hemisphere produce inverted images that improve the results of the training of the neural network. Finally, the atoms of molecules can be rendered by culling from the observer's perspective.

Once the set of observation axes has been chosen, rendering the projection is a relatively simple task involving:

- Computing the coordinates of each molecule onto the new frame of reference, where Z' is the projection axis.
- Optionally, placing an order of the atoms regarding the projection axis (from far to near to the observer) to determine the rendering order.
- Choosing a vertical axis for rendering the image within that plane.
- Plotting the segments which represent the molecular bonds (optional).
- Choosing a scale and the image size.
- Plotting each atom, from far to near, onto the image, considering opacity, color, and size optionally.

Observe that there are several options when constructing the projected image and that, a priori, how they could affect the recognition process is yet to be known. However, intuition says the artificial neural network will behave similarly to our brain, so providing more quality information will optimize the recognition process. A clear example can be seen in our brain's ease in identifying the three aromatics rings of the Ademetionine molecule in Figure 2 thanks to the presence of atomic bonds.

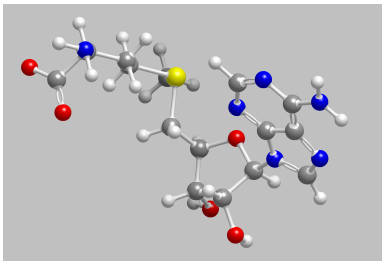

Fig. 2 Ademetionine molecule (DrugBank Accession Number: DB00118), rendered with the VIDA application.

The size and color of the atoms are also optional. However, it adds highly relevant information to differentiate, for example, the different hetero-aromatic rings present in the compounds. In this work, the diameter is calculated from

the atomic mass of the isotopes while using a color palette similar to that of the VIDA application [23], with colors depending on the atomic number, although in the larger atoms, certain external transparency has been included that allows partially visualizing the atoms that are behind (see, for example, Figure 3). A black background has been used per recommendations in the literature [24].

Fig. 3 Ademetionine molecule of the Figure 2, rendered on an image of 256x256 pixels (cropped) by using the 144th axis of a Fibonacci sequence of 512 points.

Regarding the representation of atomic bonds, an initial experiment was conducted without considering them. However, the results (recognition accuracy) were clearly of lower quality. Furthermore, rendering the bonds incurs a little computational cost, as the segments are directly drawn (using the Bresenham algorithm) between the pixels corresponding to the centers of the atoms that have already been projected onto two dimensions.

2.1 Image size and memory hierarchy

Also, through preliminary experiments in CPUs, it was shown that the image generation part was considerably more expensive than the time used in the projection of the molecules on the two-dimensional map and that it would later be converted into images, being something surprising in a first moment, since very computationally expensive trigonometric functions dominated the first operation. In contrast, hardly any read and memory operations are involved in the second case. The main reason lies in the quasi-random access to the pixel map and the size of the map itself.

To improve resolution and network training, a 256×256 pixel map was chosen since it is the largest image in which the position of the atoms can be indexed with a single byte, which is critical for factors such as transfer costs between GPU and CPU. Smaller sizes, such as 128×128, would provide insufficient resolution, while slightly larger sizes will result in a significant increase in transfer, storage, and especially recognition times, probably due to byte indexing being exceeded[1].

[1] The analysis of the source code of `TensorFlow` to find out why the recognition time is triggered (when the images exceed a certain threshold) exceeds the objectives of this work.

Table 1 Used architectures for image generation.

	Computer 1	Computer 2	Computer 3
Name	Desktop–PC	Server	HPC node
CPUs	1x Intel	32x Intel	40x Intel
	i7-10700K	Xeon E5-2698 v3	Xeon E5-2698 v4
GPUs	1x NVidia	4x NVidia	8x NVidia
	Ampere RTX3080	Maxwell GTX980	Volta V100

However, the size 256^2 of the pixel map (65536) was causing an overflow of the L1 data cache, since this, in the vast majority of processors, and of course, in those used in this study (Table 1), is 64kB or less in size. The overflow occurs because within each processor core, and even though the byte vector with the image is reused, the image (plus, at least, the vector with the coordinates of the atoms) has no place in L1. As initially stated, this problem is multiplied when a 4-byte color depth is used.

The solution consists precisely in creating the image using an indexed palette with a depth of only 16 colors (4 bits), which consumes only one byte for every two pixels, reserving a specific color for the most frequently presented atoms (such as Hydrogen, Carbon, etc.), and reusing some colors for infrequent atomic numbers.

Table 2 Benchmarking between the two used depth of color in the image. (4 byte (RGBA) vs 4 bit).

Stage	RGBA	4-bit palette
Atom projection	0.20 s	
+ coloring	11.8 s	0.28 s
+ png encoding	285 s	54 s
File size	572MB	559MB

To compare both image encodings, 256 images from a selection of 1969 molecules were constructed using a desktop personal computer (Computer 1 in table 1) with eight cores and eight threads. As in the rest of the calculations, the programming language is C++.

A dramatic decrease in execution time when reducing the pixel array size below 32kB is observed (Table 2). The encoding time for images in *png* format is also reduced (by more than 80%). In addition, the time reduction is concentrated in the stage in which the pixel map is reset (from 10.7s to 0.03s), which shows not only that the cause of the poor performance is cache misses but also that it was affecting the L2 level (of 256kB for each core, in the processor used). Otherwise, the size of the compressed data set is reduced by only a 3%, which is positive news. This reduction suggests that despite the substantial decrease in color depth, the information in the data set has been preserved.

2.2 Scaling the images

Another parameter to consider is the image scale. It is clear that the stereoscopic vision of the human eye helps to determine the difference between, for example, the model of a Boeing 747 compared with the original plane, but this is not possible in 2D images. In our problem, we could draw all the molecules at the same scale, but that would turn out to be a severe problem since the scale would have to be adjusted so as not to clip the worst case (the largest molecule, projected on the axis perpendicular to the principal axis [2]). However, this would make it necessary to draw small molecules (hydrofluoric acid, for example) with a tiny size and corresponding loss of visual quality. All molecules below a specific threshold size have been drawn with the same scale to solve this problem, while larger molecules are scaled based on size. During recognition, the neural network will use the scale label (see Section 4.1).

2.3 Number of axes and spin rate of the camera

The last issue to consider, maybe the most critical, is the number of projection axes. To tune this parameter, a reasonable range between 10^2 and 10^3 axes has been considered.

The goal is to maximize the recognition accuracy while minimizing the number of required projections. Let $f(P)$ be the recognition accuracy as a function of the set of projections P. We aim to maximize $f(P)$ subject to minimizing the number of projections $|P|$.

The optimization problem can be formulated as:

$$\max f(P), \tag{7}$$

subject to:

$$|P| \leq K, \tag{8}$$

where K is the maximum number of projections allowed by time or computational constraints.

We can use discrete optimization algorithms such as Hill Climbing, Genetic Algorithms, or Stochastic Optimization methods to find the optimal set of projections P. However, the complexity of these algorithms can make them computationally prohibitive when evaluating many possible projection counts. By selecting powers of two, we leverage their exponential growth to cover a wide range of potential projection counts with a minimal number of tests. This approach significantly reduces the computational burden while still exploring a broad and representative subset of the projection space.

Empirical studies in similar optimization contexts often show that performance improvements tend to plateau beyond certain points. By testing powers of two, we can quickly identify if there is a point where increasing the number

[2] The main axis of a molecule is considered to be the line joining the two atoms furthest from each other

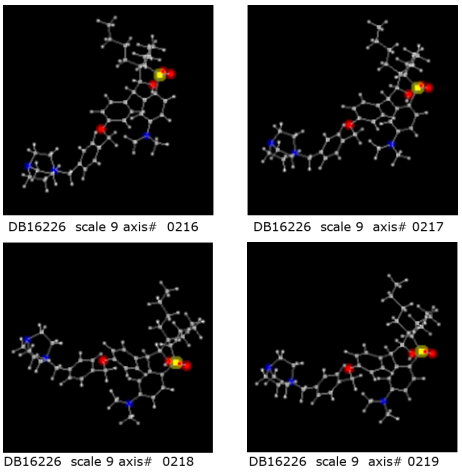

Fig. 4 Ethylhexyl methoxycrylene molecule (DrugBank Accession Number: DB11226), projected on four consecutive axes of the Fibonacci spiral.

of projections yields diminishing returns. If substantial improvements are not observed between these powers, it is likely that intermediate values will also show minimal gains. In fields like machine learning and signal processing, it is common practice to use powers of two for parameters such as batch sizes, number of nodes, or dimensions due to their alignment with underlying hardware optimizations. This precedent supports our approach from a practical standpoint. Therefore, trying with powers of two will reduce the number of attemps to find the optimal solution.

Regarding whether the same image should be rotated on its axis, it seems advisable to do so since multiple recognition experiments with Deep Learning demonstrate it. But on the other hand, more information is provided if instead of rotating, for example, $k = 16$ times the same image from the same axis j, the camera is rotated simultaneously with the movement from an axis j to the next (in the spiral) $j + 1$, thus inserting $k = 15$ intermediate axes. This technique has been named FSST (Fibonacci Spherical Spiral with Twist). In [27] an animation of 4 molecules and 1024 axes is shown.

Nine data sets have been made with 256, 512, and 1024 axes to assess the number of images and camera rotations. Three camera twist speeds have been tested: one turn every 10 and 30 frames, and without spin. A summary of the training results is shown in Table 3, although the description of the chosen parameter and a discussion of the results are detailed in Section 4.1.

These results show that the optimal case is obtained with a *dataset* of 512 images and slow rotation. Also, it uses less than a third of the resources of 1024, so this *dataset* will be used in the following experiments.

Table 3 Recognition accuracy benchmarking between the number of axes and the spin rate of the projections (symmetry index).

No. of images	No spin	1 r./30axes	1 r./10axes
256	68.5%	82.0%	78.7%
512	7.6%	86.4%	75.7%
1024	74.1%	85.8%	82.9%

3 Multi–Image rendering of molecules

The generation of the images that will train the neural network requires an enormous computational effort because the molecule databases have sizes with hundreds of thousands of samples that, in combination with the hundreds of projections of each of them, resulting in hundreds of millions of images that will have to be generated with tens and even hundreds of trigonometric operations each (depending on the number of atoms and bonds in the molecule).

3.1 Optimizations of image representation

To optimize the image representation, we define the resolution r and color depth c of the projection images in a way that minimizes the computational cost $C(r, c)$ while maximizing the recognition quality $f(r, c)$. For our study, we use images with a resolution of 256x256 pixels and 4-bit indexed color depth. This resolution is sufficient to capture the necessary details of the molecules, allowing the coordinates to be indexed with a single byte, and the 4-bit color depth balances the need for detailed representation with computational efficiency.

4 The Neural Network

In this work, a Convolutional Neural Network (CNN) classifier is proposed to capture shape similarity between molecules. CNNs are a type of deep learning model that has been highly successful in image classification tasks, as demonstrated in the Imagenet competition [18].

This study aims to closely emulate the human visual recognition process, where the brain analyzes an image and then considers its scale. To achieve this, the information about scale is entered into the CNN after the convolutional layer and before the fully connected layers.

The labeled data used to train the CNN is obtained using the preprocessing methodology described in Section 2. Each 2D image of a molecule is labeled with the corresponding identifier of the molecule in the `DrugBank` Database. The CNN is trained using supervised learning, using 2D projections of random molecules as input. The output is a probability array indicating the likelihood of each molecule in training set being the correct match.

The similarity between molecules is evaluated using a confusion matrix [21,33], a widely used machine learning technique for visualizing related classes and comparing probabilities to accuracy scores.

4.1 Preprocessing of the generated dataset

The generated dataset must include both images and information on scale and label. As such, performing preprocessing operations to extract this information before feeding it into the CNN is crucial. Additionally, shuffling the dataset is recommended to enhance the training process, as the data is organized into folders.

As the preprocessing steps are executed on CPUs, the CPUs and GPUs alternate between working and idling during the training steps. This can often be the bottleneck in the training process, as input/output operations from/to internal storage are involved.

To optimize performance, the preprocessing tasks do not have to be performed sequentially (except the first training step). The CPU can perform preprocessing on the input pipeline while the GPU works. To achieve this optimization, the dataset can be configured to use the `prefetch` function in TensorFlow. Parallelizing tasks such as loading images and extracting label and scale information from file paths is also advisable.

Batching the dataset is essential to take advantage of the GPU's capability to perform training on multiple images simultaneously. The batch size must be chosen considering the cost of computing the gradient and its ground truth. A small batch size makes the gradient computation easier, but the parallelization capacity of the GPU will not be exploited, and the gradient will be stochastic (noisy). On the other hand, a large batch size will result in a computationally intensive gradient computation, which may exceed the GPU's memory limits. However, the gradient will be closer to the ground truth.

Table 4 presents various configurations of the input pipeline to determine which design optimizes the training process. The `map` method, which can be executed either sequentially or in parallel, is called to preprocess the images, scales, and labels. Only 4096 images were used for training during three epochs for improved efficiency in the study. Results showing significant improvement (green) or detriment (red) compared to the previous configuration are colored.

The results indicate that prefetching the data significantly reduces the time required for input preprocessing. However, it may impact other resources, as the global elapsed time is not necessarily improved.

The elapsed time is sensitive to the batch size, as more images are processed simultaneously on the GPU. However, a larger batch size also implies more calls to the preprocess functions on the CPU and a higher volume of data transfer between the GPU and CPU. This explains why the input pipeline has a more significant impact if the data is batched in larger groups (Table 4).

Given these results, the batch size was further studied by training the entire dataset during five epochs (Figure 5). A batch size of 128 was determined to

Table 4 Input pipeline, measured with TensorBoard in global elapsed time, average train step time, and weight of the input preprocessing.

Configuration	ETA	Av. Time	Weight
`ds.map(seq).suffle`	0:02:52.2	11.3 ms	6,2%
`ds.map(parallel).suffle`	0:02:39.2	10,1 ms	5,0%
`| .prefetch`	0:02:38.8	10,1 ms	1,0%
`.batch(32)`	0:01:09.9	35,6 ms	53,1%
`.batch(64)`	0:01:08.0	72,5 ms	52,1%
`.batch(128)`	0:01:12.5	136,3 ms	61,8%
`.batch(32)`	0:01:09.9	35,6 ms	53,1%
`| .prefetch`	0:01:10.0	26,4 ms	1,1%
`.batch(64)`	0:01:08.0	72,5 ms	52,1%
`| .prefetch`	0:01:02.9	46,3 ms	28,3%
`.batch(128)`	0:01:12.5	136,3 ms	61,8%
`| .prefetch`	0:01:08.3	84,2 ms	35,5%

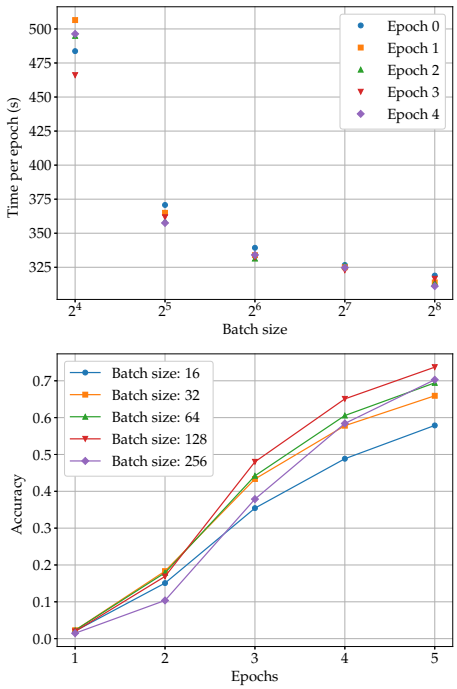

Fig. 5 Performance measured in time per epoch (upper) and accuracy (bottom) as a function of batch size.

be the optimal choice, despite the higher weight of the input pipeline. Training with a batch size of 256 may be more appropriate if more epochs are required, as accuracy increases with each trained epoch.

4.2 Model structure and relevant parameters

The structure of the image recognition model typically uses convolutional operations by applying a set of filters to the input image. Thus, the model structure

includes convolutional layers that extract meaningful features from the image, which are then classified into each class (molecule) by three fully connected layers. Before the fully connected layers, the scale information of the image is input into the model.

The above study of the input pipeline and batch size was initially performed with a simple and shallow model structure without strictly following the recommendations of other works, as a faster computation was required to make initial conclusions about the training configurations.

To test if the neural network (NN) could recognize the molecules, some structures used for multi-view 3D object recognition [24] were tried as a starting point. These structures were built from backbone networks for the convolutional layers, some of which were winners of the Imagenet competition, such as the VGG networks [29], ResNets [4,12,13], and EfficientNets [32]. Although promising results were obtained with all these backbone networks in terms of reciprocal similarity between molecules, loss, and overfitting, the results were not significantly different from much simpler architectures, and the computational cost in memory (a reduction of the batch size was required) and time (training was around five times slower) was high. Therefore, a custom neural network called MolNet2D was chosen, whose configuration was tuned through several experiments. The design of MolNet2D is shown in Figure 6.

The following conclusions were made based on experiments with MolNet2D:

- A rescaling of the input images is suggested to stabilize the gradient during computation.
- A batch size of 9x9 is used initially to capture local atom groups. Reduced filters are applied as the network becomes deeper, and zero-padding is added to maintain the image size. Atom groups are preferred over atom edges to find shape similarities.
- Dropout is only applied in fully connected layers, with a rate of 50%.
- Batch normalization coordinates the update of multiple layers in the model, reducing the "internal covariate shift" [10] and overfitting risk. It also accelerates training by allowing for larger learning rates. The order of the layers and dropout would follow other authors' recommendations [14].
- The number of filters is increased to make the classifier stricter when comparing molecules. Sixteen filters are applied at the first layer, increasing as the network becomes deeper.
- Increasing the stride of filters is more efficient than pooling operations, except for the last convolutional layer where an average pool is used.

4.3 Training of the neural network

A reduction of the objective function (9):

$$J(\mathbf{W}) = \frac{1}{B}\sum_{i=1}^{B}\mathcal{L}\left(f\left(x^{(i)};\mathbf{W}\right),y^{(i)}\right), \tag{9}$$

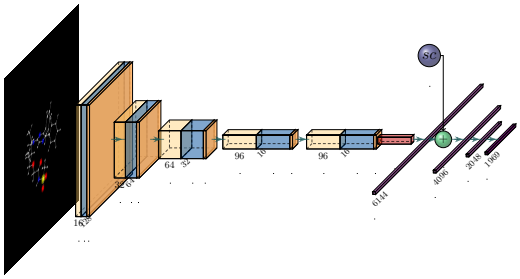

Fig. 6 MolNet2D structure. Yellow: convolutional layers; blue: batch normalization; orange: ReLU activation function; red: `Average Pooling`; purple: fully connected layers.

is the target of a NN, where $\mathbf{W}$ are the weights of all the layers, B is the batch size, $f(x^{(i)}; \mathbf{W})$ is the prediction for a specific input $x^{(i)}$, $y^{(i)}$ its ground truth, and $\mathcal{L}$ is the loss function. For a classification application, the Cross-Entropy Loss has opted,

$$\mathcal{L} = -\sum_{i=1}^{C} t_i \log(s_i), \tag{10}$$

by considering that the label is coded as one-hot,

$$y^{(i)} = [t_1, t_2, \ldots, t_C], \tag{11}$$

and the fact that the prediction is a probability array to belong to each class,

$$f(x^{(i)}; \mathbf{W}) = [s_1, s_2, \ldots, s_C], \tag{12}$$

since the output activation function `softmax` is applied,

$$s_i \leftarrow f(s_i) = \frac{\exp(s_i)}{\sum_{j=1}^{C} \exp(s_j)}. \tag{13}$$

The loss is obtained after entering a batch of images to the NN, the forward propagation, and the computing of predictions. Then, the gradient of the loss regarding weights, $\frac{\partial J(\mathbf{W})}{\partial \mathbf{W}}$ is computed to be backpropagated later by the optimizer to update the weights,

$$\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{\partial J(\mathbf{W})}{\partial \mathbf{W}}, \tag{14}$$

where η is the learning rate.

`Adam` [17] optimizer is applied since it stabilizes the training process. Several experiments with different learning rates were done, and it has been concluded that $\eta \sim 10^{-4}$ makes the training faster, ensuring stability. Accuracy is registered as a relevant metric of the training and evaluation.

4.4 Evaluation of the classifier

The dataset is divided into two parts, with 80% of the data used as the training set and 20% used as the validation set. During each epoch of the training process, the Cross-Entropy Loss (as defined in equation (10)) and accuracy (if the highest output probability s_i is equal to the label $y^{(i)}$) are measured for both sub-datasets. A measure of the model's accuracy on these two sub-datasets lets us know if the model is overfitted due to memorization of the training sub-dataset. In addition, it is a proper way to know how long the training must be. An overly short training will end in a classifier that cannot distinguish the molecules from each other. In contrast, an excessively long one will increase the over-fit risk.

After training the nine datasets with the same configuration of the NN as described earlier and training for six epochs, the Cross-Entropy Loss, as a function of various projections per molecule and twist rate applied, is calculated and depicted in Figure 7.

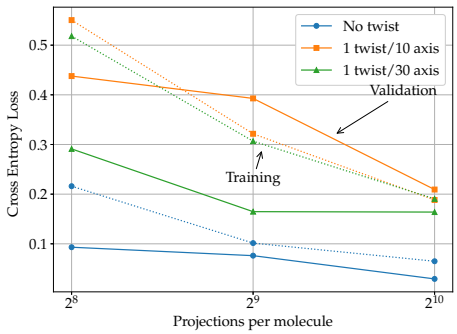

Fig. 7 Cross Entropy Loss as a function of several projections per molecule and twist rate.

A first analysis shows that a faster twist rate in projections leads to a worse classifier performance in both the training and validation sub-datasets. This difficulty in learning can be attributed to the increased differences between projections. It is worth mentioning that when each projection is more similar to its previous and its following ones, the training and validation sub-dataset will also be more similar. Therefore, this widely acknowledged technique to measure the over-fitting is not representative enough. The results also indicate that larger datasets perform better, as expected in Deep Learning.

Given the above results, it may be tempting to conclude that the dataset with 1024 projections and no rotation would be the most suitable for the proposed tool. However, it should be noted that using such a large dataset would entail a higher computational cost, as illustrated in Fig. 8. Training with a dataset of 512 projections per molecule may be more practical to balance computational cost and performance. Additionally, as previously discussed, if a twist rate is not applied, there may not be a sufficient difference between

the validation and training sub-datasets, rendering it an unreliable metric for determining the optimal dataset. As a result, an alternative approach is proposed:

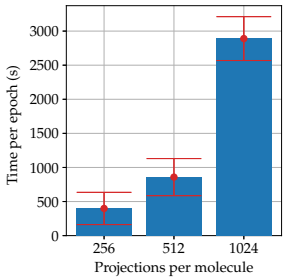

Fig. 8 Time per epoch for different datasets, measured in projections per molecule.

The case study of this work is to find the shape similarity between molecules. To this end, a confusion matrix [33] Ψ is proposed to compare the molecules. Each element $\psi_{i,j}$ in the matrix represents the mean of the probabilities $s_{i,j}$ of an image created from molecule i being a projection of molecule j. Nine matrices $\Psi_k \ \forall \ k \in 1, 2, \ldots, 9$ will be constructed using the nine trained classifiers and the nine datasets. However, the probabilities $s_{i,j}$ will be evaluated using the dataset with 1024 projections per molecule and a different twist rate from the classifier k, providing a proper method to measure over-fitting. The classifier will be considered to produce reliable results when $\psi_{i,j} \simeq \psi_{j,i}$, i.e. when the matrix is symmetric. The symmetry of Ψ can be measured using the expression [20]

$$S_\Psi = \frac{\|\Psi_{sym}\| - \|\Psi_{anti}\|}{\|\Psi_{sym}\| + \|\Psi_{anti}\|}, \tag{15}$$

with

$$\Psi_{sym} = \frac{1}{2}\left(\Psi + \Psi^T\right), \tag{16}$$

and

$$\Psi_{anti} = \frac{1}{2}\left(\Psi - \Psi^T\right). \tag{17}$$

where $-1 \leq S_\Psi \leq 1$, with the upper limit reached when the matrix is fully symmetric.

An additional metric, the recognition value, is proposed as a result of a higher value of S_Ψ due to the location of $s_{i,j}$ along the diagonal of the matrix,

$$R_\Psi = \frac{1}{C}\sum_{i=1}^{C} \psi_{i,i}. \tag{18}$$

This occurs when the model better classifies a molecule as correct, as shown in Figure 9. Over-fitting is particularly pronounced in the non-twist datasets,

especially in the one with 512 projections per molecule. As a result, the dataset with 512 projections per molecule and a smooth twist rate of 1 twist per 30 axes is selected as the optimum dataset for further studies.

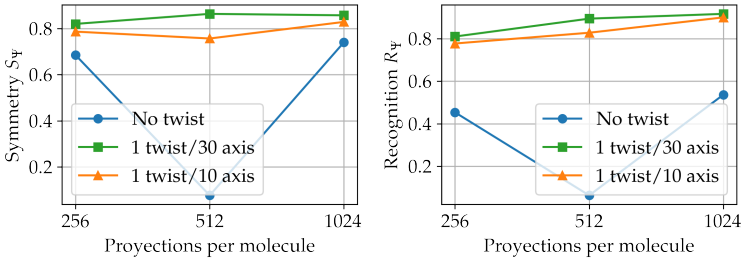

Fig. 9 Symmetry S_Ψ (upper) and recognition value R_Ψ (bottom) for 9 datasets.

4.5 Similarity analysis

Having selected the optimal dataset, various backbone networks were trained as an extension of MolNet2D to assess the reliability of the results. Since it is not possible to obtain ground truth, although shape similarity between molecules can be estimated using optimization algorithms such as Optipharm [23], a Mean Confusion Matrix ($\bar{\Psi}$) was obtained from the results of all the backbone network experiments. To determine if any significant differences existed between the various neural networks, a relative distance ϵ was calculated as the difference between each Confusion Matrix (Ψ) and the Mean Confusion Matrix ($\bar{\Psi}$) as follows:

$$\epsilon = \frac{\|\Psi - \bar{\Psi}\|}{\|\bar{\Psi}\|} \tag{19}$$

As shown in Table 5, the results are relatively close, with a lower value of ϵ than the identity matrix, which would correspond to a perfect classifier. Although the MolNet2D performed acceptably well in short CPU time, the results appear to be more representative of the ground truth when using the efficientNetV2B0 without significantly increasing computational cost.

Table 5 A comparison between backbone neural networks in terms of elapsed time, symmetry, recognition value, and relative distance to the Mean Confusion Matrix.

Network	E. time (s)	S_Ψ	R_Ψ	ϵ
MolNet2D	4026	0.866	0.894	0.120
VGG19 [29]	12858	0.861	0.875	0.125
efficientNetV2B0 [32]	6798	0.907	0.958	0.083
ResNet50 [12]	14787	0.871	0.927	0.094
ResNet50V2 [13]	9840	0.881	0.931	0.082
ResNetRS50 [4]	12192	0.889	0.954	0.107
Identity matrix	-	1	1	0.176

In practice, the same output molecules were observed regardless of the chosen backbone network or dataset, but with varying probabilities. For example, similarities were studied and compared to Optipharm [23] using the dataset with 512 projections per molecule, smooth twist rate, and an EfficientNetV2B0 as the backbone network (Table 6). Three molecules were selected to demonstrate three distinct behaviors:

- For molecule DB01365, the molecule with the highest probability (excluding itself) is the same as the output generated by Optipharm (DB00191).
- For molecule DB00728, the neural network's most similar molecule (DB04834) is not the same as the one generated by Optipharm (DB01339), although the neural network considers it to be quite similar.
- For molecule DB00320, several molecules have a higher probability than the one produced by Optipharm (DB01413).

Since expert biologists' opinions are required to determine which algorithm performs better, a graphical comparison is presented in Figure 10 to allow the human eye to judge the actual similarity. The similarity values shown in the figure have been re-scaled to reflect the correlation with the original molecule.

Table 6 Higher predictions output by the neural network compared to Optipharm (Oph) results.

Molecule DB01365		Molecule DB00728		Molecule DB00320	
Oph	DB00191	**Oph**	DB01339	**Oph**	DB01413
Neural Network		**Neural Network**		**Neural Network**	
DB00191	9.25%	DB00728	96.35%	DB00320	57.49%
DB01037	0,23%	DB01336	0.13%	DB00696	27.89%
DB01365	87.28%	DB01337	0.42%	DB09238	0.24%
DB01577	1.76%	DB01339	1.14%	DB11274	8.79%
DB09571	1,06%	DB04834	1.56%	DB13345	4.56%

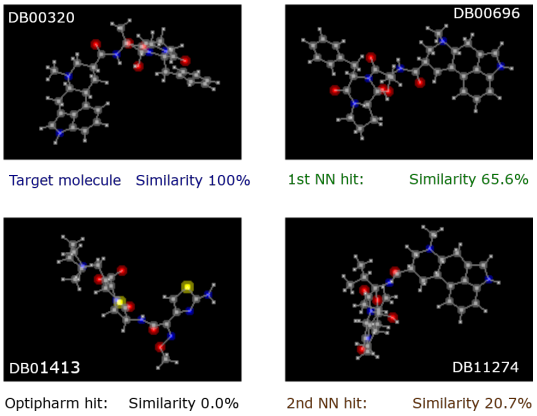

Fig. 10 Similarity of the molecule DB00320 according to the neural network.

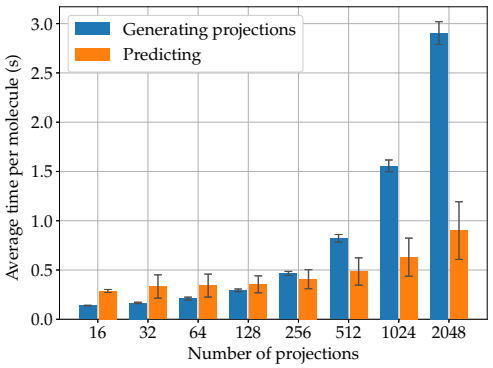

Fig. 11 Computing times averaged from 10 random molecules for each number of projections.

4.6 NN performance with the entire database

To demonstrate the relevance of this innovative approach for visual recognition of 3D objects, particularly in drug discovery, a performance timing study was conducted using the entire DrugBank dataset [34]. 9213 molecules were included, meaning the NN's output layer must have this size. This necessarily increases the number of training parameters and results in memory problems on the GPU (NVIDIA GeForce RTX 3080). To address this issue, the resolution of the projection images was reduced to 128×128px, which surprisingly resulted in almost the same recognition response. Based on the study's results, a new dataset was created from the 9213 molecules, containing 512 projection images of 128×128px, each with a smooth twist rate of 1 twist per 30 axes. Using the EfficientNetV2B0 backbone network, each epoch took approximately 21,000 seconds to complete. However, this investment of time is worth it since it only needs to be done once, after which the NN and weights can be imported into an executable script for prediction and similarity discovery between molecules. Table 7 shows a sample of the timings obtained from the program, called "Molecule Finder," when run once. Ten random samples were measured and averaged to account for the timing variability between molecules. This was repeated for different numbers of projections per molecule, and the results are shown in Figure 11. Generating the projections were deemed relevant for two reasons: it saves the user from downloading over 4 million images in the dataset and prevents using the same images for training, despite the increased computing time from I/O hard disk operations. Based on the error bars in Figure 11, it can be concluded that the classifier's performance does not depend on the molecule, unlike other optimization algorithms, and can achieve times up to 1000 times faster [23]. Moreover, it has been checked that its output is worth the results.

Table 7 Sample of timings obtained in the Molecule Finder program.

Task	E. Time (s)
Importing Tensorflow	10.4
Building NN	2.40
Loading weights	4.96
Generating 32 random projections (first time)	0.83
Predicting 32 random projections (first time)	7.38
Generating 32 random projections (following)	0.20
Predicting 32 random projections (following)	0.34

5 Conclusions and further studies

This work reports on training a neural network with a database of thousands of three-dimensional objects. Each object was transformed into hundreds of low-resolution images, leveraging available computing resources via heterogeneous computation. The images of each object were captured by projecting from a set of equidistantly spaced points on a hypothetical sphere with an infinite focal length, including camera rotation about its axis, using a technique called "FSST" (Fibonacci Spherical Spiral with Twist). The training results using a dataset of molecules from a pharmaceutical database were entirely satisfactory, demonstrating that the proposed technique represents a significant advance in 3D object recognition. It was also shown that low-resolution projections barely affect recognition capability while drastically reducing model productivity. The application, available in a public repository, behaves as a pre-processing layer that enhances the versatility of traditional neural networks for two-dimensional image recognition.

This work opens a hopeful path for research. Therefore, some branches for further research are proposed:

- Clustering of similar molecules using Autoencoders Neural Networks (ANN) [2]. These networks would let us understand which variables the NN follows to classify, besides providing an alternative way to find shape similarities.
- Adding Evidential Deep Learning at the end of the NN [1]. In this case, the NN will also compute the confidence level in the output probability array. For example, if a new molecule, which has not been used for training, is presented to the NN, a high probability does not have to mean that the molecules are similar. These techniques have already been applied to Deep Learning of molecules [30].
- Developing a web application where a molecule of the `DrugBank` is input and molecules with a high similarity percentage is output. This should include Evidential Deep Learning.
- Exploring potential synergies between 3D object recognition techniques using AI and optimization and optimal control, using these theories to reformulate the error function to make it more accurate.

Acknowledgements This work has been financed by the National R+D+i Plan Projects PID2019-105396RB-I0, PID2021-123278OB-I00 and PDC2022-133370-I00 of the Spanish Ministry of Science and Innovation and EIE funds, and by the Junta de Andalucía under project UMA20-FEDERJA-127. The authors thank the Supercomputing and Bioinnovation Center (SCBI) of the University of Malaga for their provision of computational resources and technical support.

References

1. Amini, A., Schwarting, W., Soleimany, A., Rus, D.: Deep evidential regression. Advances in Neural Inform. Process. Sys. **33** (2020)
2. Baldi, P.: Autoencoders, unsupervised learning, and deep architectures. In: Proceedings of ICML Workshop on Unsupervised and Transfer Learning, *Proceedings of Machine Learning Research*, vol. 27, pp. 37–49. PMLR, Bellevue, Washington, USA (2012)
3. Bauer, R.: Distribution of points on a sphere with application to star catalogs. Journal of Guidance Control and Dynamics **23**, 130–137 (2000). DOI 10.2514/2.4497
4. Bello, I., Fedus, W., Du, X., Cubuk, E.D., Srinivas, A., Lin, T., Shlens, J., Zoph, B.: Revisiting resnets: Improved training and scaling strategies. CoRR **abs/2103.07579** (2021). URL https://arxiv.org/abs/2103.07579
5. Chen, S., Zheng, L., Zhang, Y., Sun, Z., Xu, K.: Veram: View-enhanced recurrent attention model for 3d shape classification. IEEE transactions on visualization and computer graphics **25**(12), 3244–3257 (2018)
6. Chukkapalli, G., Karpik, S., Ethier, C.: A scheme for generating unstructured grids on spheres with application to parallel computation. Journal of Computational Physics **149**, 114–127 (1999). DOI 10.1006/jcph.1998.6146
7. Diner, O., Diner, C., Doğan, A., Weber, G.W., Özbudak, F., Tiefenbach, A.: On the applied mathematics of discrete tomography. Vychislitel'nye Tekhnologii **9** (2004)
8. Dixon, R.F.: The mathematics and computer graphics of spirals in plants. Leonardo **16** (1983)
9. Enomoto, T.: Quasi-uniform grids using a spherical helix. In: Workshop on the Partial Differential Equations on the Sphere, p. 1 (2014)
10. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning, pp. 317–319. The MIT Press (2016)
11. Gorski, K.M., Hivon, E., Banday, A.J., Wandelt, B.D., Hansen, F.K., Reinecke, M., Bartelmann, M.: HEALPix: A framework for high-resolution discretization and fast analysis of data distributed on the sphere. The Astrophysical Journal **622**(2) (2005). DOI 10.1086/427976
12. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. CoRR **abs/1512.03385** (2015). URL http://arxiv.org/abs/1512.03385
13. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. CoRR **abs/1603.05027** (2016). URL http://arxiv.org/abs/1603.05027
14. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. CoRR **abs/1502.03167** (2015). URL http://arxiv.org/abs/1502.03167
15. Kanezaki, A., Matsushita, Y., Nishida, Y.: Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (2018). DOI 10.1109/CVPR.2018.00526
16. Katanforoush, A., Shahshahani, M.: Distributing points on the sphere, i. Experimental Mathematics **12**(2), 199–209 (2003)
17. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization (2014). DOI 10.48550/ARXIV.1412.6980. URL https://arxiv.org/abs/1412.6980
18. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Proc. of the 25th International Conference on Neural Inform. Process. Sys. - Volume 1 (2012)

19. Kuter, S.: Completing the machine learning saga in fractional snow cover estimation from modis terra reflectance data: Random forests versus support vector regression. Remote Sensing of Environment **255**, 112,294 (2021). DOI 10.1016/j.rse.2021.112294
20. Lanzi, O.: How symmetric a matrix is (2016). https://math.stackexchange.com/q/2048830 [22-05-30]
21. Liu, G., Zeng, H., Gifford, D.K.: Visualizing complex feature interactions and feature sharing in genomic deep neural networks. BMC Bioinformatics **20**(1), 401 (2019). DOI 10.1186/s12859-019-2957-4. URL https://doi.org/10.1186/s12859-019-2957-4
22. Palmer, S.: Canonical perspective and the perception of objects. Attention and performance pp. 135–151 (1981)
23. Puertas-Martín, S., Redondo, J.L., Pérez-Sánchez, H., Ortigosa, P.M.: Optipharm: An evolutionary algorithm to compare shape similarity. Scientific Reports **9**(1), 1398 (2019). DOI 10.1038/s41598-018-37908-6. URL https://doi.org/10.1038/s41598-018-37908-6
24. Qi, S., Ning, X., Yang, G., Zhang, L., Long, P., Cai, W., Li, W.: Review of multi-view 3d object recognition methods based on deep learning. Displays **69**, 102,053 (2021). DOI https://doi.org/10.1016/j.displa.2021.102053. URL https://www.sciencedirect.com/science/article/pii/S0141938221000639
25. Roberts, M.: How to evenly distribute points on a sphere more effectively than the canonical fibonacci lattice. https://kdd.es/fibonacci, Accessed: 23-01-04 (2021)
26. Romero, F., Ortigosa, P., Bandera, G., Romero, L.: Skewengine: enhancing performance of intensive calculations on regular meshes. The Journal of Supercomputing **80**, 1–19 (2024). DOI 10.1007/s11227-024-05923-2
27. Romero, F., Romero, L.F.: 1024 proyecciones de 4 moléculas. https://youtu.be/MtwLkJn71AU, Accessed: 2023-01-04 (2022)
28. Saff, E.B., Kuijlaars, A.B.: Distributing many points on a sphere. The mathematical intelligencer **19**(1), 5–11 (1997)
29. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition (2014). DOI 10.48550/ARXIV.1409.1556. URL https://arxiv.org/abs/1409.1556
30. Soleimany, A.P., Amini, A., Goldman, S., Rus, D., Bhatia, S.N., Coley, C.W.: Evidential deep learning for guided molecular property prediction and discovery. ACS Central Science **7**(8), 1356–1367 (2021). DOI 10.1021/acscentsci.1c00546. URL https://doi.org/10.1021/acscentsci.1c00546
31. Swinbank, R., Purser, R.: Fibonacci grids. In: AMS 13th Conference on Numerical Weather Prediction (1999)
32. Tan, M., Le, Q.V.: Efficientnet: Rethinking model scaling for convolutional neural networks. CoRR **abs/1905.11946** (2019). URL http://arxiv.org/abs/1905.11946
33. Townsend, J.T.: Theoretical analysis of an alphabetic confusion matrix. Perception & Psychophysics **9**(1), 40–50 (1971). DOI 10.3758/BF03213026. URL https://doi.org/10.3758/BF03213026
34. Wishart, D.S., Feunang, Y.D., Guo, A.C., Lo, E.J., Marcu, A., Grant, J.R., Sajed, T., Johnson, D., Li, C., Sayeeda, Z., Assempour, N., Iynkkaran, I., Liu, Y., Maciejewski, A., Gale, N., Wilson, A., Chin, L., Cummings, R., Le, D., Pon, A., Knox, C., Wilson, M.: DrugBank 5.0: a major update to the DrugBank database for 2018. Nucleic Acids Research **46**(D1), D1074–D1082 (2017). DOI 10.1093/nar/gkx1037