# CNN In Drug Design – Recent Developments

Joerg D. Wichard
Department of Investigational Toxicology,
Bayer HealthCare, Berlin, Germany.

Maciej J. Ogorzałek
Department of Information Technologies,
Faculty of Physics, Astronomy
and Applied Computer Science
Jagiellonian University, Kraków, Poland
email: maciej.ogorzalek@uj.edu.pl

Christian Merkwirth
Kraków, Poland

***Abstract*—We describe a method for construction of specific types of Neural Networks composed of structures directly linked to the structure of the molecule under consideration. Each molecule can be represented by a unique neural connectivity problem (graph) which can be programmed onto a Cellular Neural Network. The idea was to translate chemical structures like small organic molecules or peptides into a self learning environment which is CNN based. In the case of small molecules, each cell of the CNN stands for one atom of the molecule under consideration. But in contrast to the standard CNN architecture where each cell is connected to the neighboring cells, only those cells of the feature net are connected for which there also exists a chemical bond in the molecule under consideration. This implies that the feature net topology varies from molecule to molecule. In the case of peptides, the amino acids that form the building blocks of the peptide are reflected by the CNN cells wherein the amino acid sequence defines the network topology. Unlike the standard CNN used for image processing, there are no input values like the input image that are fed into the feature net. Instead, all information about the input molecule is supplied to the feature net by means of the topology. The output of several feature nets is fed into a supervisor neural network which computes the final output value. The combination of several feature nets and a supervisor networks constitutes the Molecular Graph Network (MGN). The designed networks are used for selection of molecules representing wanted properties such as activity against specific diseases, interactions with other compounds, toxicity etc. and possibly being candidates to be tested further as new drugs.**

## I. Introduction

Cellular systems are defined by cells that have an internal state and well defined local interactions between neighboring cells that govern the dynamics of the whole system. Several systems of this type have been developed and investigated so far. Wolfram's cellular automata [1] are examples of discrete time cellular systems with discrete internal states. The Cellular Neural Network (CNN) introduced by Chua and Yang [2] could be considered as a large array of simple continuous time processing elements characterized by local connections. A discrete time version of the CNN principle has been introduced by Harrer et al. (DT-CNN) [3]. Furthermore a finite iteration DT-CNN is described, wherein the network operates as a one-step-forward computing engine [4] that can be trained with stochastic gradient descend [5]. The network and the training algorithm were developed and reported in the context of pattern recognition tasks [6], [5], [7]. The idea of translating the topology of a chemical compound into a so called Molecular Graph Network (MGN) was developed in this context [8], [9], [10]. In a MGN a compound can be described as graph where each atom is a node and each chemical bond is an edge. This concept was also developed in the neural network community where the name Graph Neural Network was coined [11].

In this way a 2D chemical structure is translated into cellular network topology: Each atom becomes a cell, each bond a local interaction between the cells and the weights depend on element and bond types. The MGN works in this case as a DT-CNN. We further extend this approach and translate the chemical structure of a peptide chain into the topology of a DT-CNN. Similar to the MGN we could translate the primary structure of a peptide directly into the topology of a learning machine wherein the network cells represent the amino acids of the peptide chain.

## II. Adaptive Cellular Systems

An important task in modern drug discovery is to understand the quantitative structure activity relation (QSAR). QSAR problems can be divided into a coding and learning part. The learning part could be solved with regression techniques that correlate the biological activities with the physiochemical or structural properties of those chemical compounds under investigation. The most important part in QSAR analysis is the identification of molecular descriptors which encode the essential properties of the compounds. Alternative approaches of the classical QSAR circumvent the problem of computing and selecting a representative set of molecular descriptors. Therefore the molecules are considered as structured data - usually represented as graphs - and these graphs are used as topology templates for learning machines (i.e. the Graph Machines [12]).

The key idea of our approach is to translate the molecular topology into a CNN topology. Each cell of the CNN stands for one chemical entity, an atom in the case of small organic molecules or an amino acid in case of peptides. But in contrast to the standard CNN architecture where each cell is connected to the neighboring cells, only those cells of the network are connected for which also exists a chemical bond in the molecule under consideration. This implies that the net topology varies from molecule to molecule. Unlike the standard CNN used for image processing, there are no input values like the input image that are fed into the network. Instead, all information about the input molecule is supplied to the network by means of the topology.

## III. Molecular Graph Networks

A graph is a dimensionless mathematical object representing a set of nodes, called vertices, and connections between pairs of nodes called edges. Two vertices connected by an edge are called adjacent. In cheminformatics graphs are used to represent molecules. In such a molecular graph, the vertices represent atoms and the edges represent bonds Graph vertices and edges are labeled, vertices by their atom type (C,N,O etc.) and edges by the bond type (single, double, aromatic etc.). While hydrogens bound to carbon are usually not taken into account, for this approach the user can either choose to consistently suppress hydrogens or to consistently include them in the representation of the molecular graph. A multiple bond between two adjacent atoms is represented by a single edge with adequate bond type (e.g. double, triple or aromatic). The structure of most organic molecules having covalent bonds can be represented as such a molecular graph. Ionic bonds can be included into the range of bond types if necessary. In the version described in this study, the method is not able to account for stereochemistry. By imposing new bond types and making weights dependent on the sterical conformation it should be possible to account for stereochemistry without a major redesign of the algorithm.

## *A. CNN like Feature Nets*

A molecular graph $m$ can be coded as a vector of element types $\vec{e}$ of the molecule's atoms and an adjacency matrix $B$ storing information on the covalent or ionic bonds between every pair of atoms within the molecule. Whenever atoms $i$ and $j$ are connected by a bond, the corresponding entry $B_{ij}$ will be a nonzero integer coding for the type of the bond (single, double, aromatic etc.). Also element types are coded as positive integers such that they can be used for indexing. Let $N$ denote the number of atoms in the molecule under consideration.

To compute this output $z$ for a given molecule $m$, the molecular graph is translated into a CNN of similar topology. No additional input values are fed into the feature net, instead the molecular structure $m$, given by $(\vec{e}, B)$, can be considered as the *input* given to the feature net.

A feature net is implemented as a discrete-time CNN like system where cell/node states evolve over $T$ iterations. Each node $i$ of the dynamical system starts with an initial state $y_i^0$ depending on its element type $e_i$ (e.g 6 for C, 92 for U), taken from the vector of initial states $h$ :

$$y_i^0 = h_{e_i} \qquad i = 1, \ldots, N \tag{1}$$

To calculate the output for one compound we compute the dynamic evolution of the nodes states $y_i^t$ for iterations $t = 0, \ldots, T-1$ according to following equations :

$$\begin{array}{rcl} x_i^{t+1} & = & \sum_{\text{atom } j \text{ adjacent to } i} \quad A^t_{e_i, B_{ij}} y_j^t + c^t_{e_i} \\ y_i^{t+1} & = & \sigma(x_i^{t+1}) \end{array} \tag{2}$$

$A^t$ is the weight table, $c^t$ the offset vector used for iteration $t$. The molecule's topology, stored in $e_i$ and $B_{ij}$, is used in Eq. 2 to generate the indices into matrices $A^t$ and vector $c$. E.g. when $e_i = 8$ and $B_{ij} = 2$, $A^t_{e_i, B_{ij}}$ refers to element $A^t_{8,2}$. Hence the terms involved in the sum in Eq. 2 will be different for each unique molecule processed.

Since a feature net usually contains more than one atom, we have to convert the output states $y_i^T$ into a single, scalar output value $z$ suitable as decision variable. This is accomplished by computing the weighted average over the state values of all nodes. The scalar output $z$ of a feature net is thus independent of the ordering of the atoms in the element table:

$$z = \frac{1}{N} \sum_{i=1}^{N} b_{e_i} y_i^T$$

Individual weight tables $A^t$ and offsets $c^t$ are used for each iteration $t$. In combination with the vector of initial states $h$ and the vector of output weights $b$ they constitute the adjustable parameters $\vec{p}$ or so called weight tables of a single feature net :

$$\vec{p} = \left\{ h, b, A^0, c^0, A^1, c^1, \ldots, A^{T-1}, c^{T-1} \right\}$$

During the dynamic evolution, the state information of each node spreads through the network along the bonds. After $T$ iterations the information on a node is propagated over a maximal distance of $T$ edges. This interaction allows for detection of functional groups without explicitly defining these groups. It also allows for detection of interactions between functional groups when the number of iterations $T$ is sufficiently large.

## *B. Supervisor Network*

A single feature network usually does not offer enough capacity to approximate a difficult regression or classification setting. We therefore feed the outputs $z_q, q \in 1, \ldots, Q$ of $Q$ individual feature nets into a conventional fully connected supervisor neural network. The output $f$ of this supervisor network for a molecule $m_n$ constitutes the final output of the MGN :

$$\begin{array}{rcl} f(m_n, \vec{P}) & = & f(z_1(m_n, \vec{p}_1), \ldots, z_q(m_n, \vec{p}_Q), \vec{p}_0) \\ \vec{P} & = & \{\vec{p}_1, \ldots \vec{p}_Q, \vec{p}_0\} \end{array} \tag{3}$$

We chose a simple feedforward neural network with a single hidden layer as topology of the supervisor

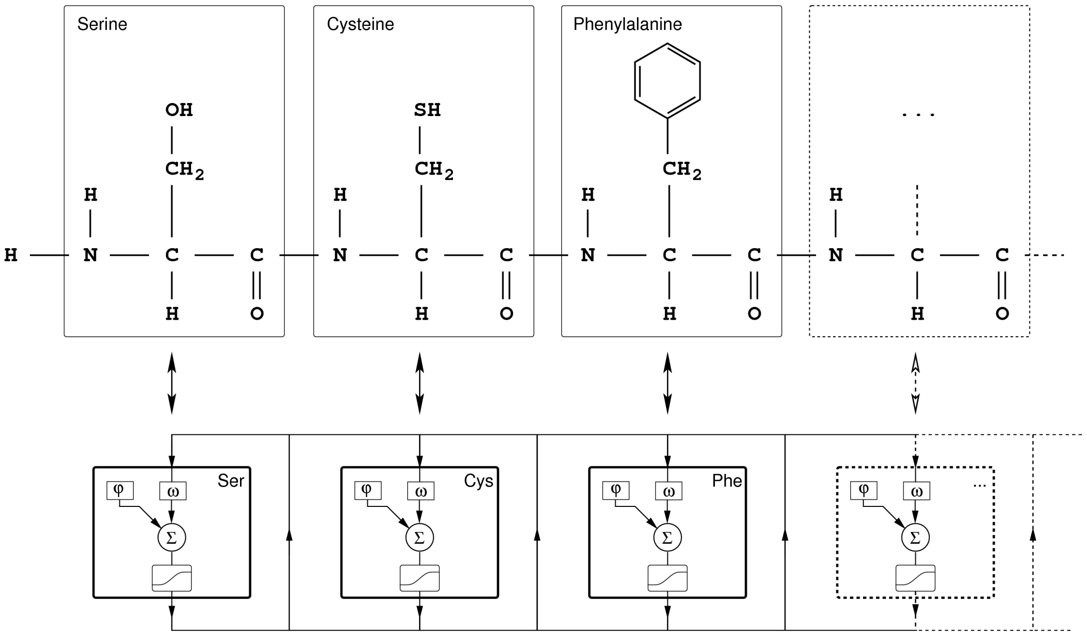

Fig. 1. This is an example of the translation process from a peptide starting with the amino acids Ser-Cys-Phe into a DT-CNN. Note the one-to-one correspondence between the amino acids of the peptide and the elementary cells of the DT-CNN. The internal weight of the cell is given by $\varphi$ and the inputs from the neighboring cells are connected with the weight vector $\omega$.

network. The number of neurons in this hidden layer can be adjusted to the complexity of the task, common choices range from 4 to 32 neurons. The concept of topology as is used for feedforward neural networks cannot be applied to the feature nets since the number of nodes and their connectivity varies from sample to sample of the data set. The only remaining parameter controlling the complexity of a feature net is thus the number of iterations $T_q$. All adjustable parameters of the feature nets $\vec{p}_1, \ldots, \vec{p}_Q$ and of the supervisor network $\vec{p}_0$ are bundled according to Eq. 3 into the total parameter vector $\vec{P}$ that exhaustively describes the MGN.

## IV. Network Representation of Peptides

Peptides are short linear polymers built from amino acids that are linked with an amide bond. About 20 proteinogenic amino acids are the building blocks of almost all proteins in nature. The string representation of the peptide $\vec{S}$ is called the *peptide sequence* and it is given by the order in which the amino acid are connected in the peptide chain. In the DT-CNN each amino acid from the alphabet is represented as a particular cell with individual weights that are adjusted during the network training. An example of the translation process from a peptide starting with the amino acids Ser-Cys-Phe into a fully connected DT-CNN is shown in Fig. 1. Hence each cell in the network corresponds one-to-one to an amino acid in the peptide and the amino acid sequence of a peptide determines the topology of the network.

The internal weight of the cell is $\varphi$, the inputs from the neighboring cells are connected with the weights $\omega_{-N,\ldots,N}$ and the feedback is controlled by $\omega_0$. The weights are combined in the weight vector $\vec{\omega}$. The cells are connected to form a chain as shown in figure 1 with an one-to-one correspondence between the amino acids in the alphabet that build the peptides and the cells in the network. The DT-CNN is iterated several times which governs the system dynamics. The internal states of the network are denoted in the state vector $\vec{y}$ with respect to their order. The state of the $i$-th DT-CNN cell $y_i^t$ evolves for iterations $t = 0, \ldots, T-1$ according to

$$\begin{aligned} x_i^{t+1} &= \varphi_i + \vec{y}^t \vec{\omega}_i \\ y_i^{t+1} &= \sigma(x_i^{t+1}), \end{aligned} \tag{4}$$

wherein the activation function $\sigma(x)$ is a hyperbolic tangent with an additional linear term that leads to non-vanishing derivatives of the training error

$$\sigma(x) = \tanh(x) + \lambda x \quad \text{with } \lambda \ll 1. \tag{5}$$

The number of iterations $T$ is set to be the average length of the sequences under investigation. The final output of the network is simply the sum over all internal states $y_i^T$ after the final iteration.

## V. DT-CNN training

The training procedure of a DT-CNN is a combination of stochastic gradient descend and back propagation with several improvements that make the training of the shared weights feasible. The true gradient is approximated by the gradient of the loss function which is evaluated on single training samples. The network weights are adjusted by an amount proportional to this approximate gradient. While the variable topology of the dynamical system constructed by a feature net appears unsuitable for a training by gradient based optimization schemes, it turns out to be actually possible to efficiently compute the gradient for the feature nets much in the same way as it is done for neural networks. Thus we can revert to the whole range of gradient based optimization techniques developed for neural networks, even with the more complicated structure of a whole MGN making a careful choice necessary. The large number of adjustable weights additionally raises the problem of choosing an efficient training algorithm. From the numerous methods existing for neural network training we found two algorithms to perform well in our case, *stochastic gradient descent* and *resilient propagation* (RPROP). Both training procedures start from randomly initialized weight tables and advance iteratively to a local minimum of the training loss [8].

## VI. Computer Assisted Peptide Design with DT-CNN Models

The goal is to recover the fundamental characteristics of the structure activity relation in an biological assay with a DT-CNN model in an iterative process. We explore the highdimensional space of all possible peptides with a genetic algorithm (GA) wherein the output of the CT-CNN model defines the fitness function of the GA. Only the top ranking virtual peptides are selected for synthesis and in vitro testing in order to produce new measurements for the next optimization cycle. Therefore a start population of peptides is selected. The peptide sequence together with the measurements from the biological assay deliver the data for DT-CNN training as described in the previous section. Training the network means adapting the weights of the cells that correspond to the amino acids of the peptides in the training set. The adapted cells work as building blocks of new *virtual* peptides which are generated by rearranging the order of the cells and calculating the output of the network as depicted in Fig. 2. The resulting DT-CNN-model defines the fitness function in a GA that is generating new suggestions for peptide synthesis based on the learned structure activity relation. The reason for the GA approach is the huge dimension of the ”virtual peptide space” that has to be explored. In the training set, the model is taught the properties of the current peptides (biological activity, stability) and the adopted cells build virtual peptides that are evaluated in the genetic algorithm-based optimization. The peptide optimization process is organized in multiple consecutive cycles. The start population of peptides is based on experts’ knowledge concerning the target, e.g. natural ligands, known analogs or compounds that bind

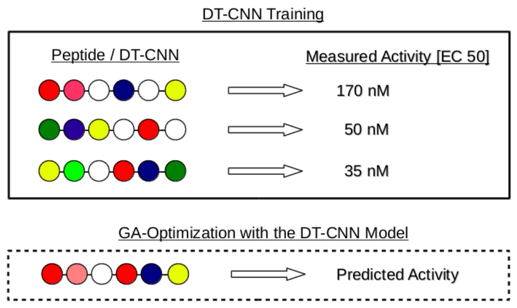

Fig. 2. The picture shows how the DT-CNN based peptide design works: The model learns the properties (in this case the activity in a biological assay) of the peptides in the training set and the adopted cells (different colors symbolize the different amino acids) build a *virtual* peptide that is evaluated in the GA-optimization.

to similar targets. The trained DT-CNN is used as a fitness function in a genetic algorithm. Newly generated sequences then have to be synthesized and analyzed in biological assays, before the next DT-CNN training is initiated (see Fig. 3).

## VII. Peptide design results

The DT-CNN approach for peptide optimization was successfully applied for the development of a 9-mer peptide with high activity and metabolic stability to bind to the Chemerin-9-receptor as reported in Bandholtz et al. [13]. The technical details of this approach were presented in an earlier publication [14]. In an iterative process of 3 design cycles including the synthesis and experimental fitness determination of less than 160 different compounds we achieved a 70fold improvement of the metabolic half life (t = 1693 min) in a ligand with subnanomolar activity in the biological assay (EC50 = 0.49 nM).

## VIII. Conclusion

Properly designed DT-CNN corresponding to Molecular Graph Network structures prove to be very useful in cheminformatics and in particular in molecular drug design having pre-described properties. In this paper we show its usefulness in peptide design. Further examples of such applications in molecular drug design are under way.

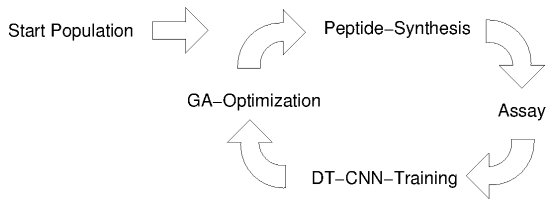

Fig. 3. A start population of peptides is selected, based on prior knowledge of the target under investigation. The sequence strings of the peptides together with the measurements from the biological assay deliver the training data. The resulting DT-CNN-model defines the fitness function in a genetic algorithm that is generating new suggestions for peptide synthesis.

## References

[1] S. Wolfram, Cellular Automata and Complexity, Addison-Wesley, Massachusetts, 1994.

[2] L. Chua, L. Yang, Cellular neural networks: Theory, IEEE Trans. on Circuits and Systems 35, 1988, 1257–1272.

[3] H. Harrer, J. Nossek, Discrete-time cellular neural networks, Int. J. Circuit Theory and Applications 20, 1992, 453–467.

[4] C. Merkwirth, J. Bröcker, M. Ogorzałek, J. Wichard, Finite Iteration DT-CNN - New Design and Operation Principles, in: Proceedings of the ISCAS, Vol. 5, 2004, pp. 504–507.

[5] J. Wichard, M. Ogorzałek, C. Merkwirth, J. Bröcker, Finite iteration DT-CNN with stationary templates, in: Proceedings of the 8th IEEE International Workshop on Cellular Neural Networks and their Applications, Budapest, Hungary, 2004, pp. 459–464.

[6] C. Merkwirth, M. Ogorzałek, J. Wichard, Stochastic gradient descent training of ensembles of DT-CNN classifiers for digit recognition, in: Proceedings of the ECCTD, Vol. 2, Kraków, Poland, 2003, pp. 337–341.

[7] J. Wichard, M. Ogorzałek, C. Merkwirth, Performance of finite iteration DT-CNN with truncated stationary templates, in: Proceedings of the IEEE International Symposium on Circuits and Systems, Vol. 5, Kobe, Japan, 2005, pp. 4657–4660.

[8] C. Merkwirth, T. Lengauer, Automatic generation of complementary descriptors with molecular graph networks, Journal of Chemical Information and Modeling 45 (5), 2005, 1159–1168.

[9] M. Ogorzałek, C. Merkwirth, J. Wichard, Pattern recognition using finite-iteration cellular systems, in: Proceedings of the 9th International Workshop on Cellular Neural Networks and Their Applications, 2005, pp. 57–60.

[10] C. Merkwirth, M. Ogorzałek, Applying CNN to cheminformatics, in: Proceedings of the ISCAS, 2007, pp. 2918–2921.

[11] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, G. Monfardini, The graph neural network model, IEEE Transactions on Neural Networks, Vol. 20, 2009, pp 61–80.

[12] A. Goulon, T. Picot, A. Duprat, G. Dreyfus, Predicting activities without computing descriptors: Graph machines for QSAR, SAR and QSAR in Environmental Research, Vol. 18, 2007, pp. 141–153.

[13] S. Bandholtz, J. Wichard, R. Kühne, C. Grötzinger, Molecular evolution of a peptide gpcr ligand driven by artificial neural networks, PLOS One, 2011, DOI: 10.1371/journal.pone.0036948.

[14] J. Wichard, S. Bandholtz, R. Kühne, C. Grötzinger, Computer Assisted Peptide Design and Optimization with Topology Preserving Neural Networks, Lecture Notes in Computer Science Vol. 6114, 2010, pp. 132–139.