**An open-source machine learning pipeline tuned for predicting peptide therapeutic activities**

**Written by Peter Sarvari, Visiting Undergraduate Researcher at Research Laboratory of Electronics, Massachusetts Institute of Technology, MEng student at Imperial College, London**

**Supervised by Dr Cesar de la Fuente Nunez and Dr Raul Castro Fernandez**

***Abstract****.* *We identified a machine learning procedure, which allows for very accurate, easily-interpretable and realistic prediction of the potential therapeutic effects of small sequence peptides. A Machine Learning pipeline was developed in Python to predict peptide activity consisting of the following 5 steps: (1) down sampling of the negative (no therapeutic activity) dataset to achieve class balance, (2) removing >90% similar sequences from the datasets, (3) extraction of the baseline (amino acid composition, 20 features), 82, 114 or 281 features, (4) training state of the art neural network model in the keras framework and (5) evaluation of model performance using one of the 7 options that can be chosen by the user and 5 metrics to report performance. Using this pipeline, we got above 80% accuracy in the prediction of all of the 7 therapeutic activity we tested, namely (1) antiviral, (2) cell penetrating, (3) antibacterial, (4) anticancer, (5) toxic, (6) tumour homing and (7) antihypertensive. For (1) we found no previous prediction in the literature and we achieved 0.61 Matthew’s Correlation Coefficient (MCC) and 80.5% accuracy. We exceeded the to-date highest known prediction accuracy for cell penetrating, tumour homing, antibacterial, toxic and antihypertensive peptides, obtaining 0.77 MCC with 89.9% accuracy, 0.72 MCC with 85.8% accuracy, 0.76 MCC with 88.0% accuracy, 0.72 MCC with 86.7% accuracy and 0.78 MCC and 89.3% accuracy, respectively using this developed rigorous protocol. We further came up with a method to deduce some of the features whose high values help obtaining the desired protein function. We named this method ADENOLIM. The user manual describing how to use our pipeline on your PC is included in the supplementary information. It details the installation of Python (Conda) on Windows as well as all the relevant libraries and the setting up of the pipeline on a remote computer via Google Cloud. The link to the GitHub page of the project is: “https://github.com/sarvarip/Protein-prediction-pipeline”.*

***Supplementary information***.

1. Setting up the pipeline on Windows (user-friendly)

Follow the steps below to install Python:

* 1. Use this link: <https://conda.io/miniconda.html> to download Miniconda 64-bit version with Python 3.6
  2. After installation open the Anaconda Command Prompt and type the following to install and activate rdkit:
     1. Conda create -c rdkit -n my-rdkit-env rdkit
     2. Activate my-rdkit-env
  3. Install other required libraries by typing the followings into the Anaconda Command Prompt
     1. Conda install keras
     2. Conda install biopython
     3. Pip install pydpi
     4. Conda install sickit-learn
  4. In this case, we will be using the Theano backend with Keras
     1. First, make sure Theano is installed (may have already been installed by Keras) by typing: conda install theano
     2. Second, make sure that numpy is installed by typing: conda install numpy
     3. For Theano to be used, pygpu and libgpuarray need to be downgraded from default to 0.6.2, so type:
        1. conda install pygpu=0.6.2
        2. conda install libgpuarray=0.6.2
     4. Since Keras is based on TensorFlow by default, Theano needs to be set as the backend in the \_\_init\_\_.py file in the keras\backend folder. The exact path to this folder will vary, but as an example, my path is: C:\Users\Peter\AppData\Local\conda\conda\envs\my-rdkit-env\Lib\site-packages\keras\backend. This should be done manually, by editing this \_\_init\_\_.py document, for example, by Notepad, looking for backend and specifying ‘theano’.
  5. Replace all the pydpi .py files with the ones with the corresponding name from GitHub. The pydpi files will be found in the pydpi library in site-packages. The exact path to the folder will vary, but, as an example, my path is: C:\Users\Peter\AppData\Local\conda\conda\envs\my-rdkit-env\Lib\site-packages\pydpi. This ensures that the scripts are compatible with Python 3.6.
  6. Download an editor, for example, Atom (<https://atom.io/>). Download our protein\_prediction\_pipeline.py as well as inputoutput.py, config.py, modelbuilder\_pipeline.py and featurex\_pipeline\_v3.py scripts from GitHub. Save it into a new folder, which contains your positive and negative dataset in the “.fasta” or “.text” format. Then open the python script with your editor and edit the relevant fields (outlined in the code, see point 3).
  7. Copy the path of the folder you saved the script in and put cd (standing for change directory) in front of it in the Anaconda Command Prompt. In my case, this is: cd C:\Users\Peter\Desktop\MIT\code
  8. To start executing the script, type: python protein\_prediction\_pipeline.py
  9. If you close the Anaconda Command Prompt (or restart your computer), you need to do steps b/ii) and g again!

1. Setting up the pipeline on Google Cloud (advanced)
   1. Download the Google Cloud SDK Shell from: <https://cloud.google.com/sdk/downloads>
   2. Sign into your google account, go to: <https://console.cloud.google.com> and create a project and instance as described here: <http://cs231n.github.io/gce-tutorial/>
      1. Stick to the Ubuntu system, but increase the specifications if you are planning to use parallelization: we used 16 vCPUs and 14.4GB memory
   3. Take note of your instance name
   4. Open Google Cloud SDK and type: gcloud init
   5. Follow the instructions on screen and take note of the time zone you choose
   6. Create SSH by typing something like: gcloud compute ssh --zone=us-central1-c instance-2 , where you need to modify the zone (us-central1-c) and the instance name (instance-2) to your own time zone and instance name
   7. You should see the PuTTY window popping up
   8. Install Miniconda 3 and add it to your homedirectory by typing
      1. wget [http://repo.continuum.io/miniconda/Miniconda3-latest-Linux-x86\_64.sh -O ~/miniconda.sh](http://repo.continuum.io/miniconda/Miniconda3-latest-Linux-x86_64.sh%20-O%20~/miniconda.sh)
      2. bash ~/miniconda.sh -b -p $HOME/miniconda
      3. export PATH="$HOME/miniconda/bin:$PATH"
   9. Install required things by typing words in *italics*
      1. Install rdkit (same as under poin 1): *conda create -c rdkit -n my-rdkit-env rdkit*
      2. Activate environment: *source activate my-rdkit-env*
      3. *Conda install keras* (tensorflow should automatically come with it, luckily on linux it is compatible with rdkit!)
      4. *Conda install biopython*
      5. *Pip install pydpi*
      6. *Conda install jupyter notebook*
      7. *Conda install sickit-learn*
      8. Use the cs231n description (<http://cs231n.github.io/gce-tutorial/>) to make static IP and set firewall rule on konzol, then use nano to append to the end of the juypter init file and use paste to create the ‘\*’ (I could not figure out any other way other way to type a \* in nano other than pasting it from clipboard)
      9. *jupyter-notebook --no-browser --port=7000*
      10. Navigate to the required static IP ‘/’ <Port number> (for example, <http://164.178.50.189:7000> ) by typing it into your browser and use token given in the PuTTy command window in the hyperlink that it asks you to paste into browser (DO NOT PASTE THAT THOUGH, just use the token from the link, after ‘token=’) To find IP, see external IP for the VM instance you are using on the google cloud console.
   10. Upload the required files (protein\_prediction\_pipeline.py as well as inputoutput.py, config.py, modelbuilder\_pipeline.py and featurex\_pipeline\_v3.py scripts from GitHub and your positive and negative datasets) using the upload tab on the browser after opening jupyter notebook and edit the relevant fields of the Python script (see point 3)
   11. Since Python version 3 is used, we need to overwrite the print and include commands in the pydpi in the following path (your path will be the same, but the second directory will be your name rather than mine): "/home/Peter/miniconda/lib/python3.6/site-packages/pydpi " in folders drug as well as protein and “pydrug.py”. Since we have done this already, just replace the imported files in the relevant directories with the ones we uploaded to GitHub.
   12. Use CTRL-C in the PuTTY window to terminate Jupyter Notebook and type: python protein\_prediction\_pipeline.py to execute the script
   13. If you get an error (I did) while loading shared libraries, here is the solution: <https://www.cyberciti.biz/faq/debian-ubuntu-linux-wkhtmltopdf-error-while-loading-shared-libraries-libxrender-so-1/>

Basically, search for the missing libs using “apt-cache search libXrender”, which returns the missing libraries and then install them using, for example, “sudo apt-get install libfontconfig1 libxrender1”

* 1. If you close your PuTTY window (the one that appeared after you computed the ssh in step f) you need to do step f), h/iii) and i/ii) again! If you restart your computer you will also need to open Google Cloud SDK first. If you want to use Jupyter notebook you will need to do steps i/ix) and i/x) again.

1. User manual for editing the protein\_prediction\_pipeline.py script

After opening the script, look for this well outlined portion (here I have deleted the comments to make this more readable):

'''User interface'''

'''Options start'''

vector\_name = "pipeline\_default"

model\_name = "pipeline\_default"

numcores = 4

choice = 0

reduce\_by\_similarity = 1

use\_random\_small\_sequence\_negative = 0

already\_extracted = 0

pos\_input\_name = "insert\_name"

neg\_input\_name = "insert\_name"

'''Only If prediction (choice 6) is chosen'''

predict\_pos\_input\_name = "insert\_name"

predict\_neg\_input\_name = "insert\_name"

known\_classes = 1

'''Only change if you get error regarding the sampling or if you are notified that your dataset s not balanced and asked to increase negative dataset sampling'''

sc\_1 = 2

sc\_2 = 1.25

'''Options end'''

'''User interface'''

* 1. Vector\_name refers to the name of the pickle file in which you store the extracted features of your sequences. For the positive dataset the name of the pickle file will be the vector name preceded by “pos\_” and for the negative dataset it will be the vector name preceded by “neg\_”.
  2. Model\_name refers to the names of the files that save the data associated with the trained model. This data is the model structure (json file), the weights data (h5 file), the mean and the standard deviation used for scaling the input data (preceded by “mean\_” and “std\_” respectively, pickle files). I recommend to make the vector name and model name the same, for example, if you are building a classifier with sequences having anti-parasitic effect, you could name it “pipeline\_antiparasitic”.
  3. The numcores variable refers to the number of cores that your machine has. It is set to 4 by default, change it according to the number of processors you have. Only available in protein\_prediction\_pipeline\_parallel.py.
  4. For choice, make it equal to the number that describes the option you would like to use (see Methods section). Default is zero meaning 10-folds cross-validation.
  5. Reduce\_by\_similarity should be one, unless you have a specific negative dataset (see Methods section).
  6. Use random\_small\_sequence\_negative should be zero, unless, you would like to use the sample negative dataset provided by us. If so, you will need to download this from GitHub. This sequence is the negative dataset we used for the anti-cancer classification (see description in Data Mining section).
  7. Already\_extracted should be set to one, if you have already run the script with these inputs (datasets) before, but now you would like to change the model (evaluation: set choice to a different number or -advanced- use a custom-written modelbuilder.py). This is to save time, since the features have already been extracted, so there is no need to do that again.
  8. Pos\_input\_name and neg\_input\_name is the name of the positive and negative fasta or txt files that contain the positive sequences (with a specific activity) and the negative sequences, respectively. For example, in the antibacterial case my neg\_input\_name is "uniprot\_3\_100\_noannotationantibacterial.fasta".
  9. Predict\_pos\_input\_name and predict\_neg\_input\_name should not be edited unless you use choice = 6 (prediction), in which case you will use one of your existing models to predict the classes of a new set of data. If you know which class the sequences in this new data belong to, put the positive sequences in a text/fasta file and make the name of the file equal to predict\_pos\_input\_name. Then put the negative sequences in another file and assign it to predict\_neg\_input\_name. If you have no such information, simply partition the sequences into two files (does not matter how you do it and they do not need to contain the same amount of sequences) and set known\_classes to zero. The probabilities you will see correspond the order of sequences in the negative dataset and then that of the positive dataset.
  10. Sc\_1 and sc\_2 and the sampling coefficients. You should only change them if you get an error message regarding the sampling or if you are asked to do so, because the classes are imbalanced. In the first case (sampling error), you should increase sc\_1 slightly (e.g. to 3 – take care that in this case the negative dataset size should be at least about 3 times the size of the positive dataset!) In the class imbalance case, you should increase sc\_2 (e.g. to 2). Sc\_2 should always remain smaller than sc\_1.

Examples about the usage of the options can be seen on the Github: the parallel versions of the scripts used to predict the 7 therapeutic activities can be accessed via <https://github.com/sarvarip/Protein-prediction-pipeline/tree/master/examples>.

1. FAQ:
   1. Problem: I got out-of-bound error when the script was trying the access the elements of an array.

Solution: Make sure that your negative dataset contains at least two times as many samples as the positive one. If not, reduce sc\_1, but make sure it is bigger than sc\_2.

* 1. Question: Which featurex online code shall I use?

Answer: It depends.

* + 1. If you would like to use the 82 features, use featurex\_pipeline\_v3.py.
    2. If you would also like to use the Moreau-Broto autocorrelation, use featurex\_pipeline.py. In this case you need to download this file from GitHub too, put it in the same folder as protein\_prediction\_pipeline.py is and rewrite the import in line 13 to “import featurex\_pipeline as FX”.
    3. If you would like to use the 281 features, use featurex\_pipeline\_v3\_pseaac\_aac.py. In this case you need to download this file from GitHub too, put it in the same folder as protein\_prediction\_pipeline.py is and rewrite the import in line 13 to “import featurex\_pipeline\_v3\_pseaac\_aac as FX”.
    4. If you would like to get a quick baseline prediction with the amino acid composition features (20), use featurex\_pipeline\_aac.py. In this case you need to download this file from GitHub too, put it in the same folder as protein\_prediction\_pipeline.py is and rewrite the import in line 13 to “import featurex\_pipeline\_aac as FX”.
    5. Advanced: Using the featurex scripts as templates, and reading the pydpi paper, you can easily customize feature extraction.
  1. Problem: Inputoutput.py gives an error and it cannot create the pickle file.

Solution: Create subdirectories called “test” and “models” within the directory that you run the scripts from.

1. Tumour homing and antiviral prediction results using the default 82 features

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Prediction according to ML protocol | Matthew’s Correlation Coefficient (MCC) | Accuracy (%) | F1 score | Precision | Recall |
| Antiviral | 0.58(+/-0.06) | 79.49(+/-2.25) | 0.78(+/-0.05) | 0.79(+/-0.04) | 0.79(+/-0.03) |
| Cell penetrating | 0.77(+/-0.04) | 89.89(+/-1.48) | 0.87(+/-0.04) | 0.88(+/-0.02) | 0.89(+/-0.02) |
| Antibacterial | 0.73(+/-0.03) | 86.23(+/-1.82) | 0.87(+/-0.02) | 0.87(+/-0.02) | 0.87(+/-0.02) |
| Anticancer | 0.60(+/-0.04) | 80.88(+/-3.84) | 0.78(+/-0.04) | 0.80(+/-0.02) | 0.80(+/-0.02) |
| Toxic | 0.72(+/-0.02) | 86.72(+/-1.55) | 0.85(+/-0.02) | 0.86(+/-0.01) | 0.86(+/-0.01) |
| Tumour homing | 0.69(+/-0.05) | 83.79(+/-3.97) | 0.85(+/-0.04) | 0.84(+/-0.02) | 0.84(+/-0.02) |
| Antihypertensive | 0.75(+/-0.05) | 87.99(+/-3.29) | 0.87(+/-0.05) | 0.87(+/-0.03) | 0.87(+/-0.02) |

1. Tumour homing and antiviral prediction results using the 281 features

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Prediction according to ML protocol | Matthew’s Correlation Coefficient (MCC) | Accuracy (%) | F1 score | Precision | Recall |
| Antiviral | 0.61(+/-0.04) | 80.45(+/-2.13) | 0.81(+/-0.04) | 0.81(+/-0.03) | 0.81(+/-0.02) |
| Cell penetrating | 0.77(+/-0.04) | 87.96(+/-4.08) | 0.89(+/-0.05) | 0.88(+/-0.02) | 0.88(+/-0.02) |
| Antibacterial | 0.76(+/-0.02) | 88.00(+/-1.37) | 0.87(+/-0.01) | 0.88(+/-0.01) | 0.88(+/-0.01) |
| Anticancer | 0.63(+/-0.05) | 81.56(+/-3.89) | 0.81(+/-0.05) | 0.81(+/-0.03) | 0.81(+/-0.03) |
| Toxic | 0.72(+/-0.04) | 85.64(+/-1.72) | 0.87(+/-0.03) | 0.86(+/-0.02) | 0.86(+/-0.02) |
| Tumour homing | 0.70(+/-0.07) | 85.58(+/-3.76) | 0.84(+/-0.06) | 0.85(+/-0.04) | 0.85(+/-0.04) |
| Antihypertensive | 0.78(+/-0.05) | 89.27(+/-2.70) | 0.89(+/-0.04) | 0.89(+/-0.03) | 0.89(+/-0.03) |

1. ADENOLIM Method
   1. Classification of the samples

|  |  |  |
| --- | --- | --- |
| Sequence | Expected | Software prediction |
| GFCK | + | + |
| CCGKKK | + | + |
| CGGCF | + | + |
| FFKKKF | + | - |
| KKCGCCGG | + | + |
| GGGG | + | - |
| GFKCK | + | - |
| CCCFFC | + | + |
| FKCKKK | + | + |
| GFGFF | + | + |
| GFGKCGCK | + | + |
| FFCKGG | + | + |
| CCKGG | + | + |
| GCKKFFCCFKGCF | + | + |
| KKGFFKG | + | + |
| MDETPVQ | - | - |
| DQMEDTT | - | - |
| EEQMMVV | - | - |
| MMMMEM | - | - |
| MMDPT | - | - |
| VPPVTQE | - | - |
| DVDVDEQ | - | - |
| EMMEMM | - | + |
| QTVTQD | - | - |
| EEETTEE | - | - |
| EQDPPPE | - | - |
| DPDEVMQPDMTME | - | - |
| EQTPDM | - | - |
| DQMMTTT | - | - |
| VEDPVVTE | - | - |

MCC = 0.74, F1-score = 0.87

To replicate the results go to: <https://github.com/sarvarip/Protein-prediction-pipeline/tree/master/ADENOLIM> download all the files (except for the .csv files, which reflect the results), run ADENOLIM.py to have a look at the ADENOLIM values and run pipeline\_complete\_antibac\_choice6.py to see the classification results of the sample sequences (see above) designed using the ADENOLIM method.

* 1. Investigation whether the slopes of a feature have the same sign in all cases or not:

The easiest way to deduce this is compare two results: one is the absolute value of the sum of the slopes and the other is the sum of the absolute values of the slopes. If the two are equal for a feature, then it must mean that the derivative of the output with respect to that feature has the same sign in all cases (through all samples).

* 1. Forward – and backpropagation [33] and also see: http://cs231n.github.io/optimization-2/

To make the backpropagation easier, we removed batch normalization from the model when using the ADENOLIM method. We use the ReLU nonlinearity between the layers. We have 4 weights matrices:

* + 1. W1, whose dimension is input features \* number of neurons in the first hidden layer = 20\*64
    2. W2, whose dimension is 64\*32
    3. W3, whose dimension is 32\*16
    4. W4, whose dimension is 16\*1

The input vector has dimension 1\*20 and we also have 4 biases, b1, b2, b3 and b4.

Hence the equations for the forward propagation are

Equations for the backpropagation:

The square brackets in the equations mean that wherever the elements in the matrix in the square bracket fulfil the inequality, the assignment occurs for the element occupying the same places in the leftmost matrix.

***References***.

[1] P. Vlieghe, V. Lisowski, J. Martinez, and M. Khrestchatisky, “Synthetic therapeutic peptides: science and market,” *Drug Discov. Today*, vol. 15, no. 1–2, pp. 40–56, 2010.

[2] G. Wang, “Human antimicrobial peptides and proteins,” *Pharmaceuticals*, vol. 7, no. 5, pp. 545–594, 2014.

[3] C. de la Fuente-Nunez, M. D. Torres, F. J. Mojica, and T. K. Lu, “Next-generation precision antimicrobials: towards personalized treatment of infectious diseases,” *Curr. Opin. Microbiol.*, vol. 37, pp. 95–102, 2017.

[4] S. Lata, N. K. Mishra, and G. P. S. Raghava, “AntiBP2: improved version of antibacterial peptide prediction.,” *BMC Bioinformatics*, vol. 11 Suppl 1, p. S19, 2010.

[5] A. Gautam *et al.*, “In silico approaches for designing highly effective cell penetrating peptides,” *J. Transl. Med.*, vol. 11, no. 1, p. 1, 2013.

[6] A. Sharma *et al.*, “Computational approach for designing tumor homing peptides.,” *Sci. Rep.*, vol. 3, no. Figure 1, p. 1607, 2013.

[7] W. Chen, H. Ding, P. Feng, H. Lin, and K. Chou, “iACP: a sequence-based tool for identifying anticancer peptides,” *Oncotarget*, vol. 7, no. 13, pp. 16895–16909, 2016.

[8] A. A. Bahar and D. Ren, “Antimicrobial peptides,” *Pharmaceuticals*, vol. 6, no. 12, pp. 1543–1575, 2013.

[9] S. Gupta, P. Kapoor, K. Chaudhary, A. Gautam, R. Kumar, and G. P. S. Raghava, “In Silico Approach for Predicting Toxicity of Peptides and Proteins,” *PLoS One*, vol. 8, no. 9, 2013.

[10] R. Kumar *et al.*, “An in silico platform for predicting, screening and designing of antihypertensive peptides,” *Sci. Rep.*, vol. 5, no. 1, p. 12512, 2015.

[11] A. Tyagi *et al.*, “CancerPPD: A database of anticancer peptides and proteins,” *Nucleic Acids Res.*, vol. 43, no. D1, pp. D837–D843, 2015.

[12] W. S. Sanders, C. I. Johnston, S. M. Bridges, S. C. Burgess, and K. O. Willeford, “Prediction of Cell Penetrating Peptides by Support Vector Machines,” *PLoS Comput. Biol.*, vol. 7, no. 7, 2011.

[13] M. Hansen, K. Kilk, and Ü. Langel, “Predicting cell-penetrating peptides,” *Adv. Drug Deliv. Rev.*, vol. 60, no. 4–5, pp. 572–579, 2008.

[14] Z. Hajisharifi, M. Piryaiee, M. Mohammad Beigi, M. Behbahani, and H. Mohabatkar, “Predicting anticancer peptides with Chou’s pseudo amino acid composition and investigating their mutagenicity via Ames test,” *J. Theor. Biol.*, vol. 341, pp. 34–40, 2014.

[15] K.-C. Chou, “Prediction of Protein Cellular Attributes Using Pseudo- Amino Acid Composition,” *Proteins Struct., Funct., Genet.*, vol. 255, no. September 2000, pp. 246–255, 2001.

[16] G. E. Hinton, S. Osindero, and Y. W. Teh, “A fast learning algorithm for deep belief nets.,” *Neural Comput.*, vol. 18, no. 7, pp. 1527–54, 2006.

[17] M. Hällbrink *et al.*, “Prediction of Cell-Penetrating Peptides,” *Int. J. Pept. Res. Ther.*, vol. 11, no. 4, pp. 249–259, 2005.

[18] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. 2016.

[19] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting,” *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, 2014.

[20] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” *Arxiv*, pp. 1–11, 2015.

[21] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” *Conf. Pap. ICLR 2015*, pp. 1–15, 2015.

[22] A. Karpathy, “Convolutional Neural Networks for Visual Recognition,” *Stanford CS class CS231n notes*, 2017. [Online]. Available: http://cs231n.github.io/.

[23] S. Fritsch and F. Guenther, “neuralnet: Training of neural networks,” *R J.*, 2010.

[24] A. Ng, “Evaluating a Learning Algorithm,” *Coursera*, 2011. .

[25] J. Hastie, Trevor, Tibshirani, Robert, Friedman, *The Elements of Statistical Learning The Elements of Statistical Learning Data Mining, Inference, and Prediction, Second Edition*. 2009.

[26] F. Provost, “Machine learning from imbalanced data sets 101,” in *Proceedings of the AAAI’2000 workshop on imbalanced data sets*, 2000.

[27] D. S. Cao, Y. Z. Liang, J. Yan, G. S. Tan, Q. S. Xu, and S. Liu, “PyDPI: Freely available python package for chemoinformatics, bioinformatics, and chemogenomics studies,” *J. Chem. Inf. Model.*, vol. 53, no. 11, pp. 3086–3096, 2013.

[28] S. Singh *et al.*, “SATPdb: A database of structurally annotated therapeutic peptides,” *Nucleic Acids Res.*, vol. 44, no. D1, pp. D1119–D1126, 2015.

[29] P. Agrawal *et al.*, “CPPsite 2.0: A repository of experimentally validated cell-penetrating peptides,” *Nucleic Acids Res.*, vol. 44, no. D1, pp. D1098–D1103, 2016.

[30] D. A. Dobchev *et al.*, “Prediction of Cell-Penetrating Peptides Using Artificial Neural Networks.,” *Curr. Comput. Aided. Drug Des.*, 2010.

[31] A. Tyagi, P. Kapoor, R. Kumar, K. Chaudhary, A. Gautam, and G. P. S. Raghava, “In silico models for designing and discovering novel anticancer peptides.,” *Sci. Rep.*, vol. 3, p. 2984, 2013.

[32] I. Dbchak, I. Munchnik, S. R. Holbrook, and S.-H. Kim, “Prediction of protein folding class using global description of amino acid sequence,” *Proc. Natl. Acad. Sci. USA*, vol. 92, pp. 8700–8704, 1995.

[33] M. A. Nielsen, *Neural networks and deep learning*. Determination Press, 2015.

[34] R. Todeschini and V. Consonni, *Handbook of Molecular Descriptors*. Wiley, 2000.