# PREDICTING MOLECULAR BIOACTIVITY

**Fisher Moritzburke**
University of California, Santa Cruz
fmoritzb@ucsc.edu

**Martin Hoffman**
University of California, Santa Cruz
maedhoff@ucsc.edu

## ABSTRACT

Virtual screening of biological molecules reduces the costs of drug discovery. We show that new techniques of implementing virtual screening models provide large improvements over past results. We focus on testing recent advances in neural networks and confirm that new optimizers such as Adam and RMSprop result in significantly better model performance. We also contribute research to residual network models, showing that residual networks train faster than their standard network counterparts while obtaining comparable results.

## 1 Introduction

Drugs are typically small organic molecules that achieve their medicinal effects by binding to a target site on a receptor. Drug discovery is the process by which new candidate medications are discovered. Traditionally, drug discovery researchers will identify and isolate the receptor to which they want a drug to bind and use in vitro or in vivo assays to test many candidate drugs on the receptor in a process called high throughput screening [1]. Each of the drugs tested can then be classified as either active (binds to the target receptor), inactive (does not bind), or somewhere in between (some binding affinity), but the process is costly and time consuming.

Virtual screening aims to reduce the costs of drug discovery by narrowing down the number of drugs that will undergo high throughput screening to only those which show the most promise of success [2]. This is done by encoding a candidate drug's structural characteristics as numerical data, which allows a machine learning algorithm to predict the activity of the drug. This way, only the most promising drugs continue on to the standard high throughput screening process.

### 1.1 Objective

We evaluate machine learning architectures and techniques for their effectiveness in virtual screening, focusing on new developments for this task since the 2012 Merck Molecular Activity Challenge. We strive to find a new standard machine learning model for virtual screening.

### 1.2 Clinical Relevance

Virtual screening can significantly speed up the resource intensive high throughput screening process by predicting the most promising compounds for further assays. We take a quantitative structure-activity relationship (QSAR) [2] approach to virtual screening, which offers fast virtual throughput and a good hit-rate.

### 1.3 Technical Significance

We built three distinct models: (1) A standard multilayer perceptron (MLP), (2) an MLP with encoder, and (3) a residual MLP. While the first two are common, we did not find any literature on a residual MLP model similar to ours. The MLP with encoder performed the best of the three models and outperformed the benchmark set by [3] on the Merck challenge datasets, and is simple to build using TensorFlow [4]. These results confirm the power of modern optimizers such as RMSprop [5] and Adam [6]. The residual MLP model shows promise for significantly faster training while obtaining similar performance, and we hope our research can serve as a foundation for more investigation into reducing compute with residual models.

## 2 Merck Molecular Activity Challenge

The Merck Molecular Activity Challenge1 Kaggle competition concluded in 2012. We chose to test our models on the Merck Challenge for two reasons: first, it provides real data from a research laboratory, which ensures that our models are tested on relevant data; second, other researchers have used this dataset since the challenge closed [3], [7], [8], so comparisons abound.

### 2.1 Data

The challenge dataset consists of 15 separate biological activity training and test sets, with molecules in some data sets having upwards of 5,000 features. The data is sparse, with more than 80% of values being zero. Our models were trained and tested separately on the HIVINT, HIVPROT, OX1, and THROMBIN data sets (table 1). This way, we had four networks for each model, one for each dataset.

Table 1: Datasets provided in the Merck Molecular Activity Challenge. Each model was trained and tested separately on each dataset.

| **Dataset Name** | **Description** | **Molecules** | **Features** |
|---|---|---|---|
| HIVINT | Inhibition of HIV integrase | 2421 | 4306 |
| HIVPROT | Inhibition of HIV protease | 4311 | 6274 |
| OX1 | Inhibition of orexin 1 receptor | 7135 | 4730 |
| THROMBIN | Human Thrombin inhibition | 6924 | 5552 |

In all four datasets, each row represents a molecule (table 2). The first two columns contain the molecule's ID and the activity score for that molecule, and the remaining columns describe the physical features of the molecule (descriptors). The activity score and descriptors are continuous variables, and most descriptors have a zero value.

Table 2: Several example rows from the HIVINT dataset.

| **Molecule ID** | **Activity Score** | **Feature 1** | **...** | **Feature n** |
|---|---|---|---|---|
| M_5605 | 5.3001 | 0 | ... | 0 |
| M_7241 | 4.6012 | 0 | ... | 0 |
| M_11511 | 5.3001 | 0 | ... | 0 |
| ... | ... | ... | ... | ... |

### 2.2 Preprocessing

Molecular features that had non-zero values in fewer than 30 molecules were dropped to reduce overfitting in the model. In the case of the HIVINT dataset, 1993 feature columns out of 4188 remained after pruning. A logarithmic transformation was also applied to the remaining non-zero feature values in the dataset, such that for every feature value $A_i$: $A'_i = log(A_i + 1)$.

## 3 Models and Algorithms

In this section, we describe our methods for building and training the models. The following three subsections correspond to the three models we tested: a standard multilayer perceptron, a multilayer perceptron with encoder, and a residual multilayer perceptron. All of these models have two common characteristics: they are all variations of the multilayer perceptron, and they all have output layers with one neuron to predict a continuous activity score between 0 and 10.

### 3.1 Standard Multilayer Perceptron

To get a baseline accuracy on the datasets, we implemented a multilayer perceptron (MLP) (fig. 1). Our model was based on the model described in [3], published by the winners of the Merck Molecular Activity Challenge.

The model contains four hidden layers. The first hidden layer consists of 2000 neurons, followed by 3 hidden layers with 1000 neurons each, and an output layer with a single neuron. Most hyperparameters are identical to those of [3].

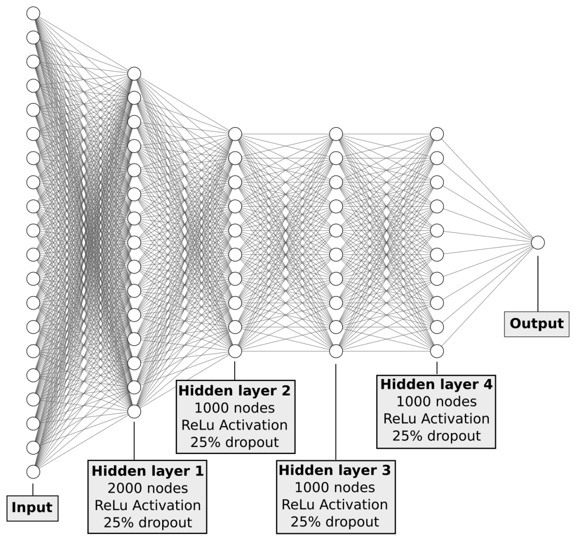

Figure 1: The multilayer perceptron model. The number of nodes, activation function, and dropout rate is shown for each layer. This model was trained for 150 epochs with a batch size of 32. The loss function was mean squared error, and the optimizer used was RMSprop.

### 3.2 Multilayer Perceptron with Encoder

The next model we tested was a multilayer perceptron with the encoder half of a pre-trained deep autoencoder attached to the front (fig. 2). The encoder acts as a feature selector so that the network is provided with a smaller subset of the data that has the highest predictive power [9].

The deep autoencoder was built with five hidden layers between an input layer and output layer, and trained for 40 epochs to reproduce each molecular sample. The encoder consists of the encoding layers, the first three layers of the autoencoder, which shrink the data down to a latent space of 50 nodes. The regression model was built by connecting the encoder to an additional 2 hidden layers of size 50 and 20, followed by the output layer of 1 node. Dropout was implemented at 25% in the first hidden layer, and 10% in the second. The 2 hidden layers map the encoded input to an activity score.

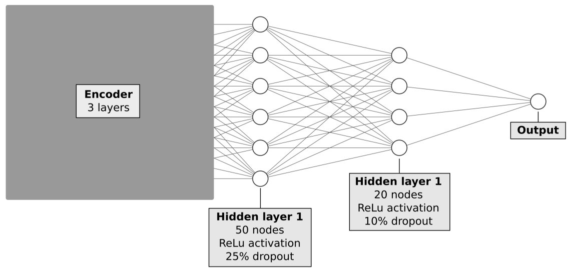

Figure 2: The multilayer perceptron with encoder model. The number of nodes, activation function, and dropout rate is shown for each layer. This model was trained for 150 epochs with a batch size of 32. The loss function was mean squared error, and the optimizer used was RMSprop.

### 3.3 Residual MLP

The residual MLP is based on the multilayer perceptron introduced first, but has added shortcut connections similar to those described in [10]. Our residual model has 13 hidden layers, with shortcut connections skipping 2 hidden layers each (fig. 3, dashed line shortcut connections), the same number of layers skipped as in [10]. Each shortcut connection terminates on an ‘add’ layer, which adds the inputs from the shortcut connection to the inputs from the final fully connected layer from the residual block. An additional 'shrink' layer was implemented between each residual block that reduces the size of the input to the next block. Dropout was used in the hidden layers within a residual block at 20%, in the shrink layer at 15%, and not at all in the add layer.

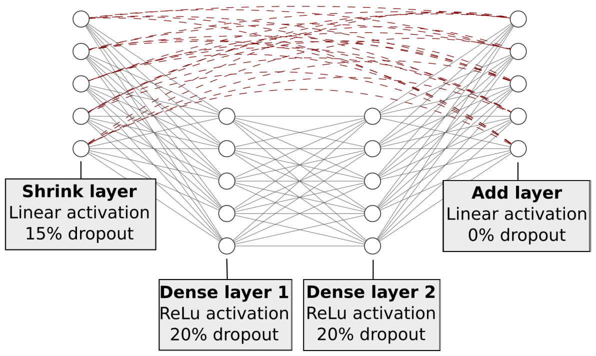

Figure 3: A single residual block from the residual MLP model. 6 of these blocks are stacked in our resMLP. The activation for the dense layers was ReLu, while the shrink and add layers had a linear activation. The amount of dropout is specified for each layer. This model was trained for 200 epochs with a batch size of 32. The loss function was mean squared error, and the optimizer was RMSprop.

## 4 Results and Analysis

We trained and tested our models on four datasets from the Merck Molecular Activity Challenge. Figure 4 below shows the R2 accuracy (coefficient of determination) of our models next to the R2 accuracy of the winning model from [3] for each dataset. Our accuracy values come from one trial run for each data set, repeated for each of our three different model architectures. In the following subsections we analyze each of the models implemented, comparing frequently to [3], which provides a baseline performance.

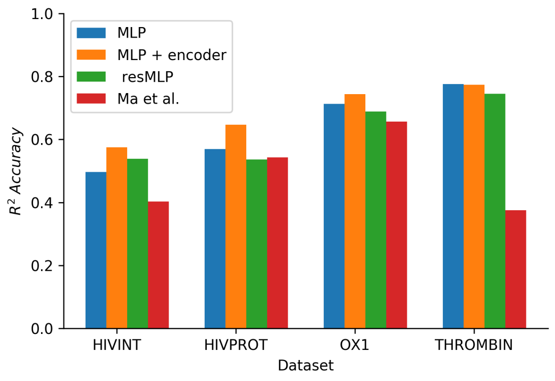

Figure 4: The R2 accuracy of the MLP, MLP with encoder, and resMLP (blue, orange, and green bars, respectively) compared to the accuracy of [3] (red bar). The MLP with encoder performs the best on average, with the standard MLP close behind, and the resMLP performing third best out of our three models.

### 4.1 Standard MLP

The standard multilayer perceptron model (Figure 4, blue bar) performed second best or better on all but one of the datasets. It consistently obtained better accuracy than [3] (Figure 4, red bar). Below, we investigate why our model performed better in order to extract useful techniques.

There are a couple key differences between our model and that of [3]. First, our model was trained on one dataset at a time and tested only on that dataset it was trained for, while [3]'s 'joint DNN' was trained on multiple datasets and was used for evaluating all of the datasets. The researchers note that they experimented with single dataset models but obtained better results with the joint approach. We did not attempt the joint training method, but can conclude that our improved accuracies did not come from training on single datasets alone if we accept [3]'s finding that joint methods perform better.

Another difference between our MLP and [3]'s is the optimization algorithm. Our model uses RMSprop [5], an unpublished optimizer proposed by Geoffrey Hinton in the 2012 online course "Neural Networks for Machine Learning."[1] It utilizes an adaptive learning rate to make better weight updates. On the other hand, [3] use Stochastic Gradient Descent (SGD) with momentum [11]. We conclude that the optimizer contributed the most to our model's improved performance (supporting experiments in section 5.1).

### 4.2 MLP with Encoder

The multilayer perceptron with encoder consistently outperformed or obtained comparable results to our standard MLP across all datasets(figure 4). Autoencoders have been shown to effectively reduce the dimensionality of data without losing key features [9], so the resulting increase in our model's performance with the encoder was expected. Training a deep autoencoder and adding the encoder portion to an MLP model is a simple process, yet we show it provides a valuable increase in performance. We recommend experimenting with this architecture.

### 4.3 Residual MLP

Inspired by resNet's state-of-the-art results on several datasets [10], we assumed that a deeper network would make better predictions on our non-image data. However, our residual MLP on average performed third best (Figure 4, green bar).

The resMLP model, with 13 hidden layers, was significantly deeper than the standard MLP. The shortcut connections in the resMLP allow increased depth by nullifying output degradation [10]. There are several explanations for their effectiveness. One is that shortcuts allow nonlinear layers to be skipped, if the solver deems this beneficial, by driving the weights of the nonlinear layers to zero [10]. Another explanation is that the shortcut connections precondition the model mapping to be closer to an identity mapping, but why a similarity to identity mapping would be beneficial is unclear [12].

We theorize that our residual model did not perform as well as the standard MLP because residual architectures are not as applicable to tabular style data as they are to image data. Each pixel of an image is part of a larger feature of the image, so a network with more layers theoretically builds an increasingly complex representation with each extra layer, using previous layer representations as building blocks [13]. Representation building at each layer is not necessary for interpreting our tabular data.

This led us to investigate whether it was the residual architecture (shortcut connections) or the extra layers that caused our residual model to perform worse than the standard one (section 5.2).

## 5 Supporting Experiments

In this section we describe experiments that were performed to test the conclusions drawn in the analysis section. Section 5.1 compares RMSprop to SGD, and section 5.2 investigates the residual architecture.

### 5.1 Optimizer Comparison

We theorize that RMSprop is the main reason our MLP performs better than that of [3]. In order to directly compare RMSprop to SGD, we took our MLP model and changed the optimizer to SGD. The SGD learning rate had to be changed from 0.05, the value recommended in [3], to 0.01 to allow training. In three trials on the OX1 dataset, the model with SGD consistently stuck at what could be a local minimum, while the model employing RMSprop did not (fig. 5). While Ma's model undoubtedly did not stick at this specific local minimum, this experiment shows the superiority of RMSprop. We conclude that the use of a better optimization algorithm contributed the most to our model's improved performance.

[1]Course video where RMSprop is proposed: `https://www.cs.toronto.edu/~hinton/coursera/lecture6/lec6e.mp4`

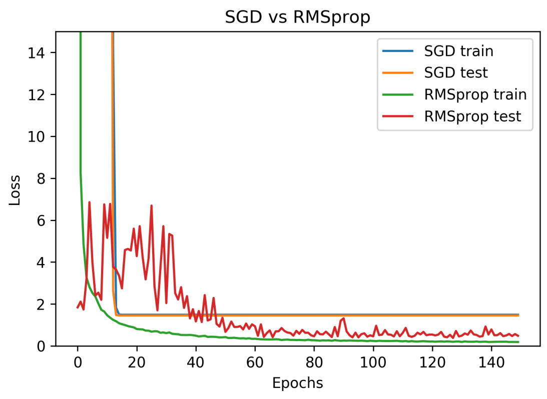

Figure 5: Loss values for the models trained using stochastic gradient descent (SGD) vs RMSprop on the OX1 dataset. The SGD model consistently becomes stuck at a local minimum during training, as seen by the flat line for both the training loss and test loss (blue and orange lines).

## 5.2 Residual Architecture

The residual model was expected to perform better than the other models tested because of its increased depth. It did not (section 4.3). In this subsection we experiment with the residual architecture to determine whether it was the residual architecture (shortcut connections) or increased depth that caused a decrease in performance compared to our standard MLP model.

To compare the residual architecture to the standard one, we built two identical shallow MLP models with the exception of two shortcut connections in the residual version (fig. 6). Each model had 3 hidden layers with 1000 nodes each. In 5 training trials, the residual model outperformed the standard one 3 times. The shallow MLP obtained an average R2 accuracy of 0.538, while the shallow residual MLP obtained an average of 0.525. The average accuracies of the two models do not distinguish them, indicating that the residual architecture by itself has little effect on the model performance. This experiment supports the hypothesis that the additional layers in the full size residual MLP caused it's decrease in accuracy, rather than the residual architecture. We propose that the increased number of layers introduces unneeded complexity, making the model difficult to train.

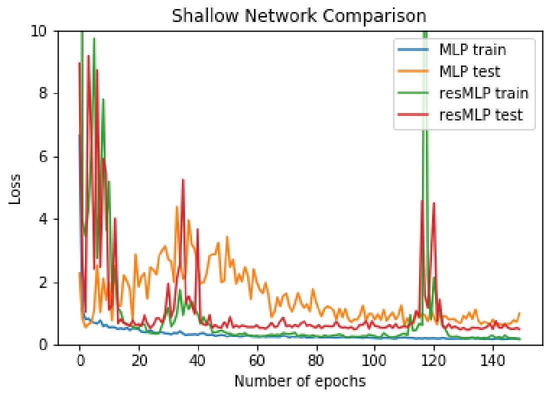

Figure 6: Comparing the training of a shallow standard MLP to that of a shallow residual MLP. The standard MLP shows much longer training times, but more stability.

Several interesting points come from comparing the shallow MLP to the shallow residual MLP. The shallow residual MLP can be trained to the same test accuracy as its shallow standard MLP counterpart in many fewer epochs. Figure 5 shows that the shallow residual MLP's test accuracies reach their minimum after about 20 epochs, while the shallow standard MLP's test accuracies converge only after 150 epochs. An experiment, repeated 10 times, recorded the shallow residual MLP's test set accuracy after training for 20 epochs. The mean R2 accuracy of the 10 trials was 0.489. The highest accuracy was 0.639, and the lowest was 0.229. The average accuracy of 0.489 is marginally less than the shallow standard MLP's 0.538, which was obtained after training for over 7 times the number of epochs. However, the residual MLP's training proved much more unstable. Notably, after being stable for about 60 epochs, both the training and testing loss values for the shallow residual MLP shot upwards and became erratic for about 30 epochs before settling back into stability. The cause of the sudden instability is unclear. Sadly, we did not have time to study the stability or shorter training of the residual MLP.

We hope this experiment contributes to the understanding of residual architectures and provides a starting point for further research into their faster training times.

## 6 Conclusion

Three models were tested on a QSAR analysis task and compared to a baseline [3]. We show that the best methods for this task have matured since 2015. We confirm RMSprop's superiority over SGD. RMSprop or a similar advanced optimization such as Adam should be used in all models. We also found that residual MLPs train faster than their standard counterparts of equal depth. The faster training of residual MLPs requires further research.

## 7 Acknowledgements

The work detailed in this paper was carried out as part of the Applied Machine Learning Course at UC Santa Cruz. We would like to thank Professor Narges Norouzi, Rafael Espericueta and Marcelo Siero for their help and support.

## References

1. Bajorath, J. Integration of virtual and high-throughput screening. *Nature Reviews Drug Discovery* **1,** 882 (2002).
2. Neves, B. J. *et al.* QSAR-based virtual screening: advances and applications in drug discovery. *Frontiers in pharmacology* **9,** 1275 (2018).
3. Ma, J., Sheridan, R. P., Liaw, A., Dahl, G. E. & Svetnik, V. Deep neural nets as a method for quantitative structure–activity relationships. *Journal of chemical information and modeling* **55,** 263–274 (2015).
4. Abadi, M. *et al. Tensorflow: A system for large-scale machine learning* in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)* (2016), 265–283.
5. Tieleman, T. & Hinton, G. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning* **4,** 26–31 (2012).
6. Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
7. Henaff, M., Bruna, J. & LeCun, Y. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163* (2015).
8. Kato, Y., Hamada, S. & Goto, H. *Molecular activity prediction using deep learning software library* in *2016 International Conference On Advanced Informatics: Concepts, Theory And Application (ICAICTA)* (2016), 1–6.
9. Hinton, G. E. & Salakhutdinov, R. R. Reducing the dimensionality of data with neural networks. *science* **313,** 504–507 (2006).
10. He, K., Zhang, X., Ren, S. & Sun, J. *Deep residual learning for image recognition* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), 770–778.
11. Bottou, L. in *Proceedings of COMPSTAT'2010* 177–186 (Springer, 2010).
12. Simonyan, K. & Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).
13. Sharif Razavian, A., Azizpour, H., Sullivan, J. & Carlsson, S. *CNN features off-the-shelf: an astounding baseline for recognition* in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops* (2014), 806–813.