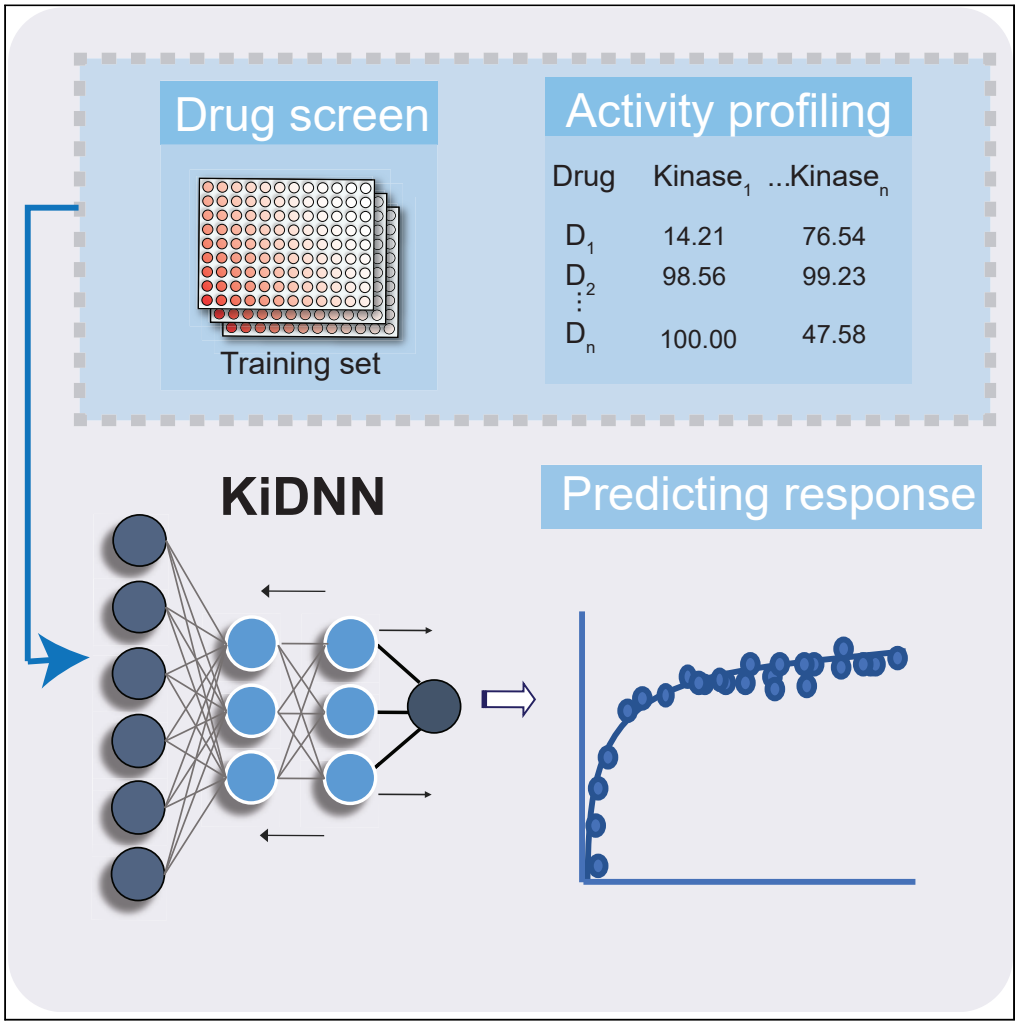
iScience

CellPress
OPEN ACCESS

Article

Non-linear Deep Neural Network for Rapid and Accurate Prediction of Phenotypic Responses to Kinase Inhibitors



Siddharth Vijay,
Taranjit S. Gujral

tgujral@fredhutch.org

HIGHLIGHTS

Deep Neural Networks mimic non-linear, complex intracellular signaling pathways

Multi-phase grid search identified best networks with less computation time

Prediction accuracy of KiDNN outperformed linear models

KiDNN can accelerate drug discovery and development efforts

Vijay & Gujral, iScience *23*, 101129
May 22, 2020
https://doi.org/10.1016/j.isci.2020.101129

iScience

CellPress
OPEN ACCESS

Article

Non-linear Deep Neural Network for Rapid and Accurate Prediction of Phenotypic Responses to Kinase Inhibitors

Siddharth Vijay[1] and Taranjit S. Gujral[1,2,3,*]

SUMMARY

Protein kinase inhibitors are one of the most successful targeted therapies to date. Despite this progress, additional kinase inhibitors are needed to expand the target space as well as overcome drug resistance that has emerged in clinical setting. Here, we developed KiDNN (Kinase inhibitor prediction using Deep Neural Networks). KiDNN utilizes non-linear, multilayer feedforward network that mimics complex and dynamic kinase-driven signaling pathways. We used KiDNN to predict the effect of ~200 kinase inhibitors on migration of breast and liver cancer cells. We show that the prediction accuracy of KiDNN outperformed other prediction tools based on linear models. We validated that an inhibitor of tyrosine kinase receptors, and an inhibitor of Src family kinases, decreased migration of triple-negative breast cancer cells, consistent with the role of these kinases in driving motility. Overall, we show that non-linear, DNN-based models provide a powerful approach to *in silico* screen hundreds of kinase inhibitors.

INTRODUCTION

Kinase inhibitors represent a large class of US Food and Drug Administration-approved drugs and are currently in use for the treatment of various cancers. Advances in "omics" technologies have led to the development of high-throughput methods to efficiently and reliably profile the target selectivity of kinase inhibitors *in vitro* and in a cellular environment (Klaeger et al., 2017; Schmidlin et al., 2019; Zegzouti et al., 2016). Several groups have profiled hundreds of kinase inhibitors against sizable fractions of the ~500 human protein kinases (Anastassiadis et al., 2011; Davis et al., 2011; Duong-Ly et al., 2016; Klaeger et al., 2017; Schmidlin et al., 2019). The resulting "kinase-inhibitor interaction maps" revealed that majority of kinase inhibitors bind multiple targets, despite the efforts to generate selective agents (Duong-Ly et al., 2016; Klaeger et al., 2017). In some cases, these unintended targets result in off-target toxicity, whereas in other instances, off-target binding is of clinical benefit, leading to the effect commonly referred to as polypharmacology. Although broadly acknowledged as a dominant mode of action, polypharmacology is currently poorly understood and difficult to predict.

We and others have developed linear regression-based approaches, such as Kinome Regularization (KiR), which exploit the polypharmacology of kinase inhibitors and rely on linear combinations of the contributions of kinases to cellular behavior to build models for predicting the effect of a particular inhibitor on specific cellular phenotypes (Gujral et al., 2014a, 2014b). However, kinase-driven signaling pathways are highly complex and dynamic, and the outcome of a perturbation is difficult to predict from the linear combination of the individual parts. Here, we hypothesize that there is a complex, non-linear dependence between the activity profiles of inhibitors and their respective contribution to phenotypes, such as cell proliferation or migration. This suggests that a non-linear, multilayer feedforward network would exhibit improved performance over the linear approach. To test this hypothesis, we developed a deep neural network (DNN) based on non-linear principles called KiDNN for Kinase Inhibitor prediction using Deep Neural Networks.

KiDNN takes advantage of the DNN framework that mimics the human brain. DNNs incorporate processing nodes that are analogous to neurons (Zupan, 1994). In KiDNN, these nodes are connected by weighted links into a complex, multi-layered neural network. All nodes, except those comprising the input layer, receive weighted sums of the output from the nodes in the previous layer and transmit their output to nodes in successive layers until the final output layer is reached (Dongare et al., 2012). We applied KiDNN to

[1]Division of Human Biology, Fred Hutchinson Cancer Research Center, Seattle, WA, USA

[2]Department of Pharmacology, University of Washington, Seattle, WA, USA

[3]Lead Contact

*Correspondence: tgujral@fredhutch.org

https://doi.org/10.1016/j.isci.2020.101129

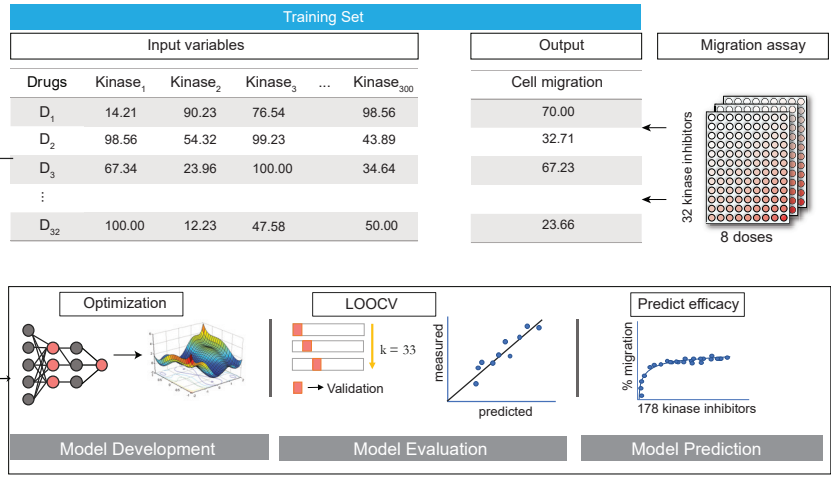


Figure 1. The Development of KiDNN

A schematic illustrating the supervised learning approach to develop, evaluate, and predict kinase inhibitor-mediated changes in migration using KiDNN. The input variable is defined by a set of 32 kinase inhibitors and their target profiles against 300 kinases (32X300 matrix). The numbers in input variable indicate percent residual kinase activity. Output variable is defined by the response of 32 inhibitors on migration of cancer cells measured using wound healing assay in 96-well plates. The numbers in output variable indicate percent wound density. The development of KiDNN consisted of model development phase during which hyperparameters are optimized, model evaluation where LOOCV is employed to generate predictions, and model prediction where response to naive kinase inhibitors are predicted and experimentally validated.

predict the effect of ~200 kinase inhibitors on migration of triple-negative breast cancer (TNBC) cell line (Hs578t) and liver cancer cells (FOCUS). We experimentally tested a subset of the inhibitors and determined that KiDNN predictions outperform predictions from the linear KiR model. As predicted by KiDNN, we showed that inhibition of the tyrosine kinase receptor including PDGFRb and VEGFR and nonreceptor tyrosine kinases Src decreases migration of Hs578t cells, consistent with the role of these kinases in driving motility of TNBC cells (Jechlinger et al., 2006; Simiczyjew et al., 2018; Van Swearingen et al., 2017). Overall, our results indicated that the application of models based on non-linear DNNs is superior to linear-based models for predicting the cellular response to kinase inhibitors with known activity profiles.

RESULTS

Developing an Optimized KiDNN for Predicting Kinase Inhibitor Effect on Hs578t Cell Migration

Kinase inhibitors are widely used to identify cellular signaling pathways underlying complex phenotypes. Here, we developed a predictive DNN model and screened for kinase inhibitors that impaired migration of cancer cells as a disease-relevant phenotype (Figure 1). In our previous study, we demonstrated that a set of 32 kinase inhibitors as well as non-specific compounds provides >85% coverage of the 300 kinase targets (Gujral et al., 2014b). To develop and optimize KiDNN models, we used a previously generated experimental dataset for the quantitative effects of these 32 inhibitors on migration of Hs578t TNBC cells (Gujral et al., 2014b).

To develop DNN models that effectively predict changes in cell migration in response to kinase inhibitors, we devised a five-phase strategy for optimizing the neural network hyperparameters for maximum predictive capability on Hs578t cell migration (Figure 2A). The first phase was deducing the range of overfitting, where we used the training and validation loss to identify the optimal range of epochs. In subsequent phases (phases 2–5), we performed optimization using Grid Search (Bergstra et al., 2011), in which numerous models were built with every combination of hyperparameter values, and the top models (and their respective combination of hyperparameter values) were identified. Various hyperparameters were optimized in each phase. Batch size and epochs were optimized in phase 2. In phase 3, activation function, weight initialization, and optimizers were tuned, whereas hidden layers, nodes per hidden layer, and dropout were tuned in phase 4 (Figure 2A). Table S1 lists the tested hyperparameter values, and the Transparent Methods outlines the definitions of the hyperparameters. When performing Grid Search,

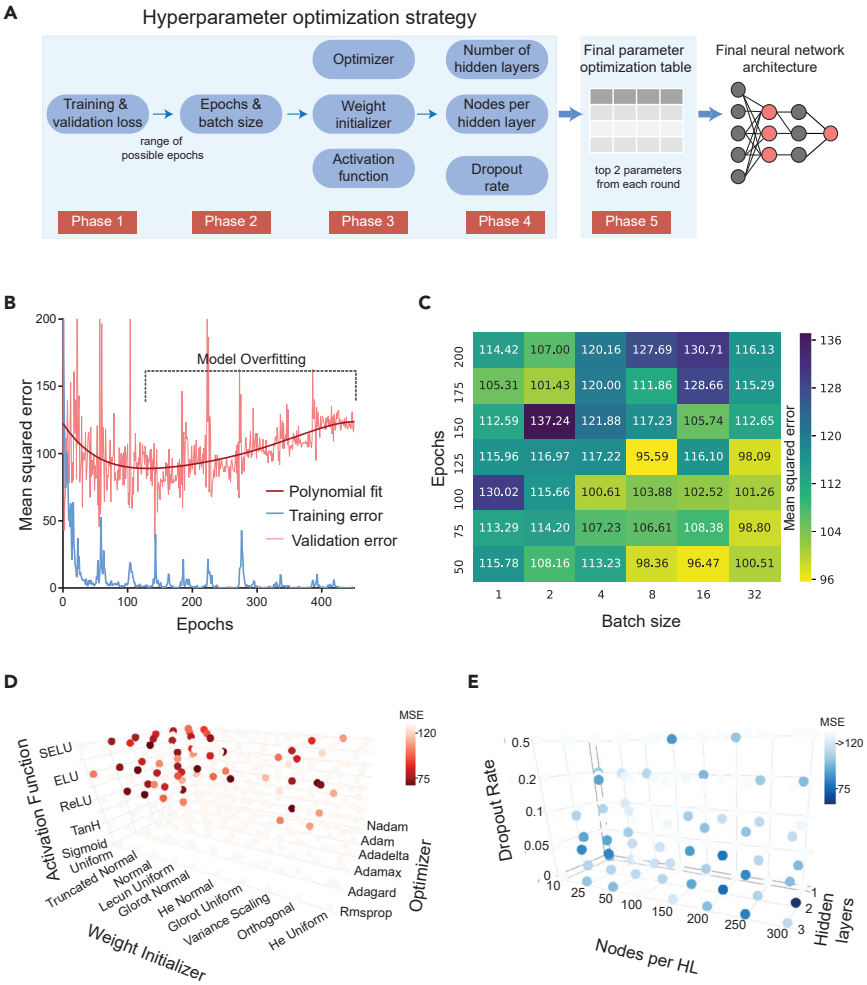


Figure 2. Tuning Network Hyperparameters and Structure for Optimal KiDNN Performance

(A) A schematic showing the progressive, multi-phase approach of optimizing the network hyperparameters/structure for peak network performance.

(B) A plot showing the network validation MSE (k = 26) and training MSE as a function of the number of epochs. A polynomial fit (n = 5) of the validation error is also shown and the range of overfitting is indicated.

(C) A heatmap showing the respective LOOCV MSE of 42 networks built with selected combinations of batch sizes and epochs. Yellow regions indicate low relative errors.

(D) A 3D scatterplot illustrating respective LOOCV MSE of 300 networks built with selected combinations of activation functions, weight initializations, and optimizers. Darker spheres indicate low relative errors corresponding to specific combinations of hyperparameters.

(E) A 3D scatterplot illustrating respective LOOCV MSE of 120 networks built with selected combinations of hidden layers, nodes per hidden layer, and dropout rate. Darker blue spheres indicate low relative errors corresponding to specific combinations of hyperparameters. Complete list of hyperparameters tuned are listed in Table S1 and additional trials of the top five hyperparameters are evaluated in Tables S2–S4.

hyperparameters that were not undergoing optimization were kept constant according to a set of baseline hyperparameter values (see the Transparent Methods for details on how the baseline hyperparameter values are updated). In the last phase (phase 5), the top two hyperparameter values identified in phases 2, 3, and 4 were used to perform a final Grid Search that selects the single top combination of all eight hyperparameter values. This set of eight hyperparameter values was then used to build the final KiDNN model for predicting Hs578t cell migration (Figure 2A).

To evaluate and select the top hyperparameters in each phase, we applied leave-one-out-cross-validation (LOOCV) (Zhang, 1993). In LOOCV, each time the KiDNN model was trained on the activity profiles of 31 of

the 32 inhibitors and their effects on cell migration (Gujral et al., 2014b) to predict the effect of the excluded inhibitor on cell migration. The process was repeated 33 times (including control), leaving out and predicting the effect on migration for each of the 32 inhibitors. The average mean squared error (MSE) of all 33 predictions was used to evaluate predictive accuracy of various networks and select the hyperparameters that produced the lowest LOOCV MSE.

In phase 1, we determined the range of model overfitting starting with an unoptimized baseline neural network. We trained the baseline neural network on the Hs578t migration responses to 26 randomly selected inhibitors (representing ~80% of the data) to predict the effect on cell migration of the six excluded inhibitors (the remaining ~20% of the data). The model was trained for 400 epochs, which is the total number of times KiDNN learned from the entire training dataset to optimize its weights. We plotted the training MSE for the 26 kinase inhibitors and the cross-validation MSE of predicted and observed migration for the six excluded kinase inhibitors as a function of the number of epochs (Figure 2B). A polynomial fit (n = 5) for the cross-validation error indicated that the network's predictive performance improved until 125 epochs, after which the network gradually started to overfit the data. Although too few epochs can cause underfitting of the data, suggesting the model stops learning too early, too many epochs can cause the model to overfit the training data such that the model cannot generalize to effectively predict the response to six left out kinase inhibitors on cell migration. Consequently, an optimal range of epochs where the MSE reached a global minimum was selected using the validation error: 50, 75, 100, 125, 150, 175, 200 (Figure 2B). Since batch size (optimized in phase 2) can affect the optimal number of epochs, a buffer of 75 epochs above and below the actual global minimum of 125 epochs ensured that the top combination of epochs and batch size was selected with variations in batch size in phase 2.

In phase 2, we evaluated the optimal combination of batch size and epochs. The batch size is the number of samples (individual observations in the dataset) input into KiDNN before updating the weights (Chollet, 2018). Using Grid Search, we constructed 42 networks representing the combinations of the seven epoch quantities and six batch sizes and evaluated the accuracy of the network by LOOCV MSE (Figure 2C). Our data showed that network error was minimized with larger batch sizes and lower quantities of epochs (Figure 2C, lower right quadrant). We re-evaluated the top five combinations of epochs and batch sizes an additional five times, which revealed that the top two combinations of batch size and epochs were 32-sample batch size and 75 epochs (average MSE 100.48) and 16-sample batch size and 50 epochs (average MSE 101.47) (Table S2).

In phase 3, we tuned the weight initialization, optimizer, and activation function hyperparameters. We built and evaluated 300 networks (5 activation functions x 10 weight initializations x 6 optimizers combinations) using Grid Search (Figure 2D, Table S1). Our data indicated that networks built with the TanH (hyperbolic tangent) and sigmoid activation functions were completely ineffective at predicting changes in cell migration (Figure 2D); thus, we excluded them from subsequent optimizations for activation function. We re-evaluated the top five combinations of weight initialization, optimizer, and activation function (Table S3). The top two combinations of hyperparameters were truncated normal initialization, Adagrad optimizer, and ELU activation function (average MSE 92.90) and truncated normal initialization, Adagrad optimizer, and ReLU activation function (average MSE 94.62).

In phase 4, we optimized the dropout rate, the number of hidden layers, and the number of nodes per hidden layer. With Grid Search, we built and evaluated 120 networks (5 dropout rates x 3 hidden layer quantities x 8 nodes per hidden layer). Networks built with lower dropout rates and more than one hidden layer had greater predictive power as indicated by their lower average MSE values (Figure 2E). These results confirm our initial hypothesis that the activity profiles of kinase inhibitors and their respective effect on Hs578t cell migration have a complex, non-linear dependence that Deep Neural Networks can better predict rather than shallow neural networks with one hidden layer or linear models. We re-evaluated the top five combinations of the hidden layer quantity, nodes per hidden layer, and dropout rate (Table S4). The top two combinations of the hyperparameters were two hidden layers, 300 nodes per hidden layer, and a dropout rate of 0 (average MSE 85.74) and two hidden layers, 200 nodes per hidden layer, and a dropout rate of 0 (average MSE 89.86).

In phase 5, we tested the top two sets of hyperparameters (shaded rows in Tables S2–S4), which yielded 8 (2^3) networks that combined the two sets of hyperparameters from each of the three phases. To identify the top performing model, we calculated the average LOOCV MSE and MAE (Mean Absolute Error) of five runs

Epochs	Batch Size	Weight Initializer	Optimizer	Activation	Hidden Layers	Nodes per HL	Drop out Rate	Mean MSE	Mean MAE
50	16	Truncated Normal	Adagrad	ReLU	2	300	0	108.742	6.6
50	16	Truncated Normal	Adagrad	ReLU	2	200	0	101.4	6.6
75	32	Truncated Normal	Adagrad	ReLU	2	300	0	116.341	6.2
75	32	Truncated Normal	Adagrad	ReLU	2	200	0	99.2	5.9
50	16	Truncated Normal	Adagrad	ELU	2	300	0	90.3	5.9
50	16	Truncated Normal	Adagrad	ELU	2	200	0	95.7	6.1
75	32	Truncated Normal	Adagrad	ELU	2	300	0	87.4	5.6
75	32	Truncated Normal	Adagrad	ELU	2	200	0	87.2	5.68

Table 1. Evaluation of Networks from Combination of Top Hyperparameters Selected in Each Optimization Phase
The network with the lowest average MSE used to build KiDNN is highlighted. ReLU, rectified linear unit; ELU, exponential linear unit; Adagrad, adaptive gradient.

of each of the eight models (Table 1). The top two performing models both had similar MSE and MAE values (Table 1). Given the similarity in performance between the two top-performing models, we selected the model with the lowest average MSE (87.17) and simpler structure to build the final KiDNN model (KiDNN-Hs578t).

The network architecture consisted of 300 input nodes (representing activity of each of the kinases from the *in vitro* kinase profiling work that tested 178 inhibitors against 300 kinases, Dongare et al., 2012), two hidden layers with 200 nodes per layer, and a single output node for predicted migration (Figure 3). The batch size of 32 equaled input sample size, commonly referred to as batch gradient descent, and the entire training dataset was input through KiDNN-Hs578t 75 times (epochs) in total. Because the final model included only 200 nodes in the hidden layer rather than the top-performing hyperparameter for the number of nodes in the hidden layer (300 nodes, Table S4), this last phase captured a further optimized combination of hyperparameters. Thus, our results support performing this last optimization phase rather than just building the final model from the top-performing hyperparameters (lowest MSE) identified in each phase.

Evaluating the Predictive Capability of KiDNN Models for New Drugs

Having optimized the hyperparameters of KiDNN-Hs578t to yield the lowest validation MSE in predicting changes in migration of Hs578t cells in response to kinase inhibitors, we evaluated the predictive capability of KiDNN-Hs578t. First, we used LOOCV to predict the response to each of the 32 kinase inhibitors (entire training dataset) and compared those with experimentally observed migration (Figure 4A). The MSE of the predictions was 78.15, whereas the Pearson correlation was 0.88. On average, the model differed from experimentally observed changes in cell migration by 5.80% for any given prediction.

Although the accuracy of predictions is informative in evaluating a model, a low variance of predictions can ensure a high-precision network. A potential limitation of DNN models is that these models can produce extremely high variance among predictions, resulting in drastically varying output from one run to another. To test for variance in KiDNN-Hs578t, we determined LOOCV predictions of changes in cell migration in response to each of the 32 inhibitors in our training set during 10 different evaluations by KiDNN-Hs578t (Figure 4B). The mean standard deviation of all 33 (including control) predictions was 1.44, indicating an

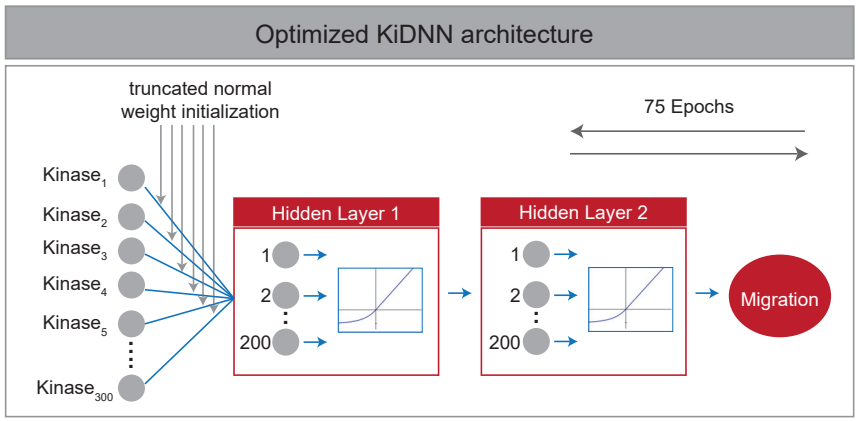


Figure 3. Optimized KiDNN Architecture

A schematic illustrating the final architecture of KiDNN after hyperparameter optimization using training dataset from Hs578t cell migration.

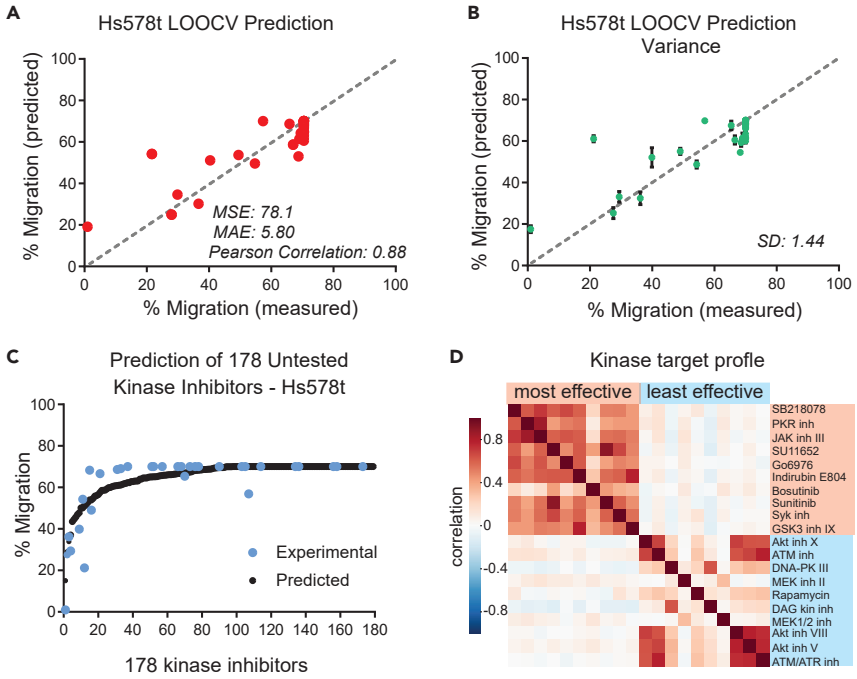
overall high prediction precision. With the low MSE and MAE (Figure 4A), this model has both high prediction precision and accuracy for the effect of kinase inhibitors on Hs578t cell migration.

The previous analysis determined the accuracy and precision using the observed Hs578t migration in response to inhibitors used in our training set, whereas we next used KiDNN-Hs578t to predict the effect of 178 compounds (for which *in vitro* activity profiles against 300 kinases have previously been determined [Anastassiadis et al., 2011]) on Hs578t cell migration. The experimentally measured effects of the 32 compounds in our training set closely aligned with the curve formed by the predicted migration effects of all 178 kinase inhibitors, indicating that the predicted effects are likely close to what one might expect to observe experimentally (Figure 4C, Table S5). Of the 178 kinase inhibitors predicted by the model, 175 had differences between the predicted and interpolated migrations (based on polynomial interpolation (n = 6) of the 32 kinase inhibitors' experimental migration) <10% and 137 had differences <5%.

To gain insight into the key signaling nodes targeted by the top KiDNN predicted inhibitors, we determined the correlation among the top 10 highly ranked and the least 10 effective inhibitors predicted by KiDNN to effect migration of Hs578t cells. Staurosporine, K252a, and CDK1/2 inhibitor II, the three most promiscuous inhibitors were omitted from this analysis. We found a strong positive correlation ($r > 0.5$ Pearson) among the top 10 highly ranked inhibitors and negative correlation ($r < -0.5$) in terms of their activity profiles across all 300 kinases (Figure 4C). Furthermore, unsupervised clustering of the kinases clearly showed that the kinases that are inhibited (<40% residual activity) by the top 10 most effective inhibitors are not inhibited (>90% residual activity) by any of the least 10 effective predicted inhibitors (Figure S1). These kinases include Src family kinases and several RTKs, including PDGFR, VEGFR, and FGFR1. Together, these data provide further support and biological rationale to KiDNN predicted most effective drugs that inhibit migration of Hs578t cells.

Experimental Validation of KiDNN-Hs578t Predictions in Hs578t Cells

A stringent test of any predictive model is its ability to forecast responses to a completely new dataset that was not used for training the model. To evaluate predictive accuracy of KiDNN, we experimentally tested the migration of Hs578t cells in response to eight kinase inhibitors that were not part of the 32 previously evaluated. The overall small differences between the observed and predicted migration of the Hs578t cells in response to the eight kinase inhibitors confirmed the high predictive performance of KiDNN (Figure 4D) with the predictions differing on average from the observed responses by 4.99% (Figure 4E). Response to five kinase inhibitors (Purvalanol A, Staurosporine N-benzoyl-, SU11652, PD98059, and Dovitinib) were predicted extremely well with a <5% difference (Figure 4D). The strong effect of Dovitinib, which inhibits Src family kinases (Bello and Gujral, 2018), and SU11652, which inhibits receptor tyrosine kinases, such as PDGFRb and VEGFR (Bello and Gujral, 2018), was evident in the wound migration assay (Figures 5A and 5B). With a 4.2% and 2.5% difference between the predicted effect and observed effect, respectively, these results indicate that KiDNN could accurately identify and predict migration in response to drugs that had a profound effect on the measured phenotype.



E

Evaluating KiDNN predictions with experimentally observed migration

Kinase inhibitor	Measured (%)	KiDNN Prediction (%)
Purvalanol A	70	68.6
Staurosporine, N-benzoyl-	56.3	51.4
AMPK Inhibitor Compound C	65.8	57.8
PDK1/Akt/Flt Dual Pathway Inhibitor	70	63.6
SU11652	44.4	48.6
JAK Inhibitor I	70	57.5
PD98059	70	70
Dovitinib	55.7	58.2
Mean Squared Error		38.8
Mean Absolute Error		4.99

Figure 4. Predicting Response to Naive Kinase Inhibitors

(A) A plot showing comparison of KiDNN-predicted and measured percent migration in response to 32 inhibitors in Hs578t cells using LOOCV. Migration in response to untreated/DMSO control was 70%. The MSE, MAE, and Pearson correlations are also indicated.

(B) KiDNN model was run for 10 iterations to generate predictions for all 32 inhibitors. Green circles denote the mean predicted migration, and the error bars show the standard deviation of predictions for each kinase inhibitor. The standard deviation is also listed.

(C) A plot showing KiDNN-predicted response to 178 small molecule kinase inhibitors (146 untested). The black circles denote the predicted percent migration, whereas blue circles denote the 32 experimentally validated kinase inhibitors. The kinase inhibitors are ranked by predicted migration.

(D) A heatmap showing correlation of kinase target profiles of the 10 most and 10 least effective inhibitors predicted by KiDNN.

(E) A table showing comparison of KiDNN-prediction and experimental changes in migration of Hs578t cells in response to eight kinase inhibitors. Each inhibitor was tested at multiple doses (10 nM–10 μM), and the effect of kinase inhibitor at 500 nM dose was calculated using the dose-response curves. Hs578t cells treated with DMSO control showed migration of 70%.

In addition to identifying drugs that are effective, it is also desirable to identify kinase inhibitors with minimal effects or opposite effects from those sought for therapeutic intervention. Of the eight kinase inhibitors evaluated, four had a limited effect (70%) on Hs578t cell migration relative to control (Figure 4D). All four kinase inhibitors were predicted to have minimal effect by the model (>50% migration). Purvanol A, PD98059, and PDK1/Akt/Flt

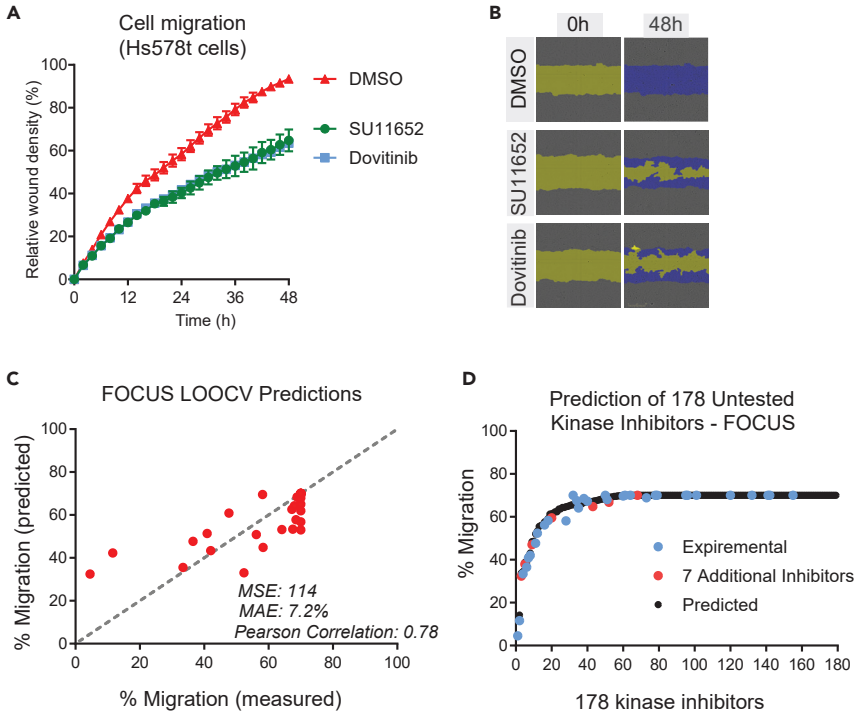


Figure 5. Evaluating Predictive Capability of KiDNN

(A) A plot showing relative migration of Hs578t cells in response to DMSO control, SU11652 (1 μM), or Dovitinib (1 μM). Each point is a mean of three replicates, and the error bars denote SEM.

(B) Representative images of Hs578t cells at time 0 h (wounding) and 48 h post wounding. Wound area is highlighted in yellow, and the migrating cells in the wound area are shown in blue mask.

(C) A plot showing comparison of KiDNN-predicted and measured percent migration in response to 32 inhibitors in FOCUS cells using LOOCV. Migration in response to untreated/DMSO control was 70%.

(D) A plot showing KiDNN-predicted response to 178 small molecule kinase inhibitors (146 untested) in FOCUS cells. The black circles denote the predicted percent migration, whereas the blue circles and red circles denote the percent migration for 32 experimentally validated kinase inhibitors and 7 unseen inhibitors respectively. The kinase inhibitors are ranked by predicted migration.

Dual Pathway Inhibitor were all predicted accurately (<10% error), whereas JAK1 inhibitor I was not predicted as accurately (>10% error). Together, these results validate that the KiDNN model can accurately predict changes in cell migration in response to both highly effective and ineffective kinase inhibitors, even with a minimal training dataset (~18% of entire set of all 178 kinase inhibitors).

Comparing Predictions from KiDNN with Those from KiR

The initial application of KiDNN was to predict the effects of kinase inhibitors on TNBC Hs578t cell migration, whereas here we generated a KiDNN model for the effects of kinase inhibitors on liver cancer cells. Each KiDNN must be optimized and trained for the specific cell and phenotype under investigation. Previously, we applied the KiR approach to build a model based on a response to training set (32 inhibitors) and determined its ability to predict the effect of 178 kinase inhibitors on migration of liver cancer cells (FOCUS) (Gujral et al., 2014b). Here, we used the same dataset to build an optimized KiDNN model and compared the predictions of the KiDNN model with those of the KiR model. Using the same five-phase hyperparameter optimization strategy that we used to build the KiDNN model for predicting migration of Hs578t cells (Figure 2), we determined an optimal KiDNN architecture for the dataset collected on FOCUS cells.

The KiDNN-FOCUS architecture consisted of two hidden layers with 100 nodes per layer. Weights were updated in batch sizes of two (mini-batch gradient descent) with the entire dataset input through KiDNN-FOCUS a total of 120 times (epochs). The other network hyperparameters included the uniform weight initialization, SELU activation function, and the Adagrad optimizer. The only hyperparameters in common with KiDNN-Hs578t were the optimizer, the number of hidden layers, and the Dropout rate. The differences

Kinase Inhibitor	Measured	Linear Regression Predictions	KiDNN Predictions	KiDNN LOOCV Predictions
Aminopurvalanol A	66.8	60.3	64.2	66.3
AMPK compound C	64.8	54.5	57.1	56.6
Cdk2 inhibitor IV, NU6140	70	52.7	66.3	63.8
Dovitinib	32.3	33.6	51.3	50.1
GSK-3 inhibitor XIII	59.5	53.7	59.9	62.0
Staurosporine, N-benzoyl-	37.9	30.9	55.3	48.0
SU11652	47.0	20.7	49.4	50.9
Mean squared error		**175.1**	**106.5**	**78.2**

Table 2. Comparison of Predicted Migrations of the Seven Additional Inhibitors for Both KiDNN and KiR on FOCUS
In the KiDNN Predictions column, KiDNN was trained with the 32 kinase inhibitors and applied to the 7 additional inhibitors, whereas in the KiDNN LOOCV Predictions column, each time 6 of the 7 additional inhibitors were added to the training set and the remaining inhibitor's migration was predicted for each of the 7 inhibitors. The MSE of the models' predictions are indicated.

in the two KiDNN models are consistent with the cancer cells coming from different tissues and having different characteristics. As we did for KiDNN-Hs578t, we used LOOCV to evaluate the accuracy of the KiDNN-FOCUS predictions. Using the 32 inhibitors from the training set plus the control, the MSE of the LOOCV predictions was 114 and on average the model's predictions differed from experimentally observed migrations by 7.2% (Figure 5C).

We also used KiDNN-FOCUS to predict the effects of the seven kinase inhibitors that were not part of the original dataset. Using the analyses of these seven inhibitors, we compared the predictive capability of KiR (Gujral et al., 2014b) and KiDNN using the FOCUS models (Table 2). Our data showed that KiDNN predictions outperformed KiR predictions with KiDNN reducing the prediction error (MSE) by ~40% compared with that of KiR. The average differences between KiR predictions and experimental observations (10.65%) was also greater than those between KiDNN predictions and experimental observations (7.58%).

Predictions of KiDNN and KiR models did not agree (>10% difference between the models) for four of the seven kinase inhibitors: Cdk2 Inhibitor IV (NU6140), Dovitinib, Staurosporine (N-benzoyl-), and SU11652. Of these, SU11652 and Cdk Inhibitor IV (NU6140) were better predicted by KiDNN; Dovitinib and Staurosporine (N-benzoyl-) were better predicted by KiR. In total, five inhibitors (Aminopurvanol A, AMPK Compound C, Cdk Inhibitor IV [NU6140], GSK-3 Inhibitor XIII, and SU11652) were better predicted by KiDNN with an average MSE of 17.09 and an average difference from observed migration of 3.36%. Dovitinib and Staurosporine (N-benzoyl-) were better predicted by KiR with an average MSE of 25.79 and an average difference from observed migration of 4.17%.

To further improve the KiDNN-FOCUS predictions, each time we included six of the seven inhibitors as part of the training set and predicted the excluded inhibitor's effect on migration of FOCUS cells (LOOCV). The MSE of KiDNN-FOCUS LOOCV prediction for the seven kinase inhibitors was 78.19, a ~26% decrease in prediction error from KiDNN-FOCUS original predictions and a more than a 2-fold decrease in MSE compared with KiR. This decrease in prediction error demonstrates that addition of experimental data can drastically improve KiDNN performance enabling an iterative approach to build accurate KiDNN models. We used this improved KiDNN-FOCUS model to predict the effect on FOCUS cell migration of 178 inhibitors (139 untested) with previously known activity profiles (Anastassiadis et al., 2011) and reported effects on migration (Gujral et al., 2014b) (Figure 5D, see Table S6 for predicted and observed effects of each kinase inhibitor). Of the 178 kinase inhibitors' effect on FOCUS cell migration predicted by the model, 177 had differences between the predicted and interpolated migrations (based on polynomial interpolation (n = 7) of the 39 kinase inhibitors' experimental migration) <10% and 166 had differences <5%.

DISCUSSION

Technological advancement in "omics"-based approaches for collecting large-scale datasets, particularly those involving drug-target profiling, gene expression, and protein abundance and modification, have unlocked the door to data-driven systems biology-based studies. Although studying a large number of interactions even in a network of closely related proteins is powerful in principle, it is often hard to find actionable or informative patterns in the data. Thus, system biologists rely on computational models to analyze and make scientific breakthroughs from large-scale datasets. In particular, machine learning and deep learning approaches are used to aid in model design and selection of compounds in pre-clinical drug discovery (Zhavoronkov et al., 2019). These computational approaches offer an effective strategy for prioritizing compounds that can lower the cost of pre-clinical trials by reducing the experimental search to smaller, model-predicted subsets (Zhang et al., 2019).

Here, we developed KiDNN, a multilayer DNN approach (Figure 1) and applied it to predict changes in migration of metastatic cancer cells in response to hundreds of kinase inhibitors. DNNs can capture non-linear relationships among features, such as the effects of multispecific inhibitors on kinase activities and the effects of kinase activities on complex cellular phenotypes like migration, and are better suited than linear regression-based approaches for investigating questions with a large number of samples and complex features. We applied KiDNN to predict kinase inhibitor effects on the migration of two cancer cell lines, one a TNBC cell line and the other a hepatocellular carcinoma cell line. For each cell line KiDNN is optimized and trained using existing data for the polypharmacology of a subset of kinase inhibitors and data on the effects of those inhibitors on migration of each cell line. We used LOOCV MSE to assess the predictive capability for each cell line-specific KiDNN and determined that KiDNN-Hs587t had an MSE of <80 with predictions that differed from observed effects by <6%, while KiDNN-FOCUS had an MSE of 114 with predictions that differed by 7%. These small differences indicated that the two KiDNN models had biologically acceptable performance. Furthermore, experimental validation showed that KiDNN predictions (MSE 78) were more accurate than KiR-based predictions (175) (Table 2), suggesting that using a multilayered DNN substantially improves overall accuracy and performance over linear models.

The molecular heterogeneity of cancer underscores a need for screening drug candidates in multiple patient-derived cell lines or animal models (Barretina et al., 2012). In oncology drug discovery and development, this is often a cost-prohibitive and laborious undertaking. To address this, we showed that DNN hyperparameters optimized on inhibitor responses in breast mesenchymal cells (Hs578t) coupled with a new training dataset can be used as a starting point to build a KiDNN model that accurately predicted inhibitor responses in liver mesenchymal (FOCUS) cancer cells. Such promising results suggested that the hyperparameters learned from one set of data can be used to generate KiDNN that predicts responses in multiple relevant cancer cell lines with minimal training data from each cell line.

Finally, we applied the KiDNN approach to predict the effect of ~200 kinase inhibitors on migration of Hs578t and FOCUS mesenchymal cancer cells. The models predicted drugs that were effective at impairing migration and those that were ineffective. Such information is valuable in drug discovery. Noteworthy for the Hs578t cells, KiDNN-Hs578t accurately predicted response to Dovitinib (58.2% predicted versus 55.7% observed) and SU11652 (48.6% predicted versus 44.4%, observed) (Figures 5A and 5B). Dovitinib is a Src family kinase inhibitor (Anastassiadis et al., 2011), and SU11652 is an inhibitor of multiple receptor tyrosine kinases, including PDGFRb and VEGFR (Bello and Gujral, 2018). Both Src and PDGFRb are important for migration of TNBC cell lines (Ho-Yen et al., 2015; Jechlinger et al., 2006; Sausgruber et al., 2015; Simiczyjew et al., 2018; Van Swearingen et al., 2017), highlighting validating the relevance of the KiDNN predictions.

The high predictive accuracy for individual kinase inhibitors demonstrated that KiDNN is a powerful deep learning approach that overall performed significantly better than linear models for predicting the effects of a large panel of kinase inhibitors on a complex cellular phenotype. Furthermore, the models can be retrained on limited data from different cells. We predict that application of KiDNN modeling coupled with polypharmacology profiling will enable cheaper and more effective screening than exhaustive, unbiased testing of compound libraries. We anticipate future development of KiDNN models could integrate additional parameters, such as non-kinase targets of kinase inhibitors, chemical moieties, pharmacokinetic properties, and other medicinal chemistry properties, to enable *de novo* compound design, prioritization, and drug discovery. Furthermore, combining KiDNN predictions for different cellular outcomes, such as proliferation, apoptosis, and migration, could identify lead candidate drugs that produce desired outcomes without stimulating undesired ones.

Limitations of the Study

A particular limitation of KiDNN and Deep Neural Network models broadly is their lack of interpretability in predictions. Deep Neural Networks have traditionally been labeled black-box models as it is increasingly difficult to inspect how the neural network model reaches its predictions as its complexity increases. Although KiDNN prediction of cell migration of Hs578t and FOCUS cells closely followed experimental values, the precise biological basis behind the predictions in terms of specific kinases or their combination are difficult to extract. To address this limitation, future development of KiDNN could involve a feature importance algorithm to deduce the primary kinases the KiDNN model relies on to make predictions.

Resource Availability

Lead Contact

Further information and requests for resources and reagents should be directed to and will be fulfilled by the Lead Contact, Taranjit S Gujral (tgujral@fredhutch.org).

Materials Availability

This study did not generate new unique reagents.

Data and Code Availability

Source code for KiDNN is available in Data S1.

METHODS

All methods can be found in the accompanying Transparent Methods supplemental file.

SUPPLEMENTAL INFORMATION

Supplemental Information can be found online at https://doi.org/10.1016/j.isci.2020.101129.

ACKNOWLEDGMENTS

This work was supported by the NIH/NCI (K22CA201229, P30CA015704), Fred Hutch Evergreen Fund, and the Sidney Kimmel Foundation (Kimmel Scholar Award). We thank Drs. Nancy Gough and Milka Kostic for helpful comments on the manuscript.

AUTHOR CONTRIBUTIONS

T.S.G. and S.V conceived the project. S.V. developed the KiDNN algorithm. T.S.G performed experiments. S.V. and T.S.G. performed analysis. S.V. and T.S.G. wrote the manuscript.

DECLARATION OF INTERESTS

The authors declare no competing interests.

Received: February 13, 2020
Revised: April 4, 2020
Accepted: April 29, 2020
Published: May 22, 2020

REFERENCES

Anastassiadis, T., Deacon, S.W., Devarajan, K., Ma, H., and Peterson, J.R. (2011). Comprehensive assay of kinase catalytic activity reveals features of kinase inhibitor selectivity. Nat. Biotechnol. *29*, 1039.

Barretina, J., Caponigro, G., Stransky, N., Venkatesan, K., Margolin, A.A., Kim, S., Wilson, C.J., Lehár, J., Kryukov, G.V., and Sonkin, D. (2012). The Cancer Cell Line Encyclopedia enables predictive modelling of anticancer drug sensitivity. Nature *483*, 603.

Bello, T., and Gujral, T.S. (2018). KInhibition: a kinase inhibitor selection portal. iScience *8*, 49–53.

Bergstra, J.S., Bardenet, R., Bengio, Y., and Kégl, B. (2011). Algorithms for hyper-parameter optimization. Paper presented at: Advances in neural information processing systems.

Chollet, F. (2018). Keras: the python Deep Learning Library (Astrophysics Source Code Library).

Davis, M.I., Hunt, J.P., Herrgard, S., Ciceri, P., Wodicka, L.M., Pallares, G., Hocker, M., Treiber, D.K., and Zarrinkar, P.P. (2011). Comprehensive analysis of kinase inhibitor selectivity. Nat. Biotechnol. *29*, 1046.

Dongare, A., Kharde, R., and Kachare, A.D. (2012). Introduction to artificial neural network. Int. J. Eng. Innov. Technol. *2*, 189–194.

Duong-Ly, K.C., Devarajan, K., Liang, S., Horiuchi, K.Y., Wang, Y., Ma, H., and Peterson, J.R. (2016). Kinase inhibitor profiling reveals unexpected

opportunities to inhibit disease-associated mutant kinases. Cell Rep. *14*, 772–781.

Gujral, T.S., Chan, M., Peshkin, L., Sorger, P.K., Kirschner, M.W., and MacBeath, G. (2014a). A noncanonical Frizzled2 pathway regulates epithelial-mesenchymal transition and metastasis. Cell *159*, 844–856.

Gujral, T.S., Peshkin, L., and Kirschner, M.W. (2014b). Exploiting polypharmacology for drug target deconvolution. Proc. Natl. Acad. Sci. U S A *111*, 5048–5053.

Ho-Yen, C.M., Jones, J.L., and Kermorgant, S. (2015). The clinical and functional significance of c-Met in breast cancer: a review. Breast Cancer Res. *17*, 52.

Jechlinger, M., Sommer, A., Moriggl, R., Seither, P., Kraut, N., Capodiecci, P., Donovan, M., Cordon-Cardo, C., Beug, H., and Grünert, S. (2006). Autocrine PDGFR signaling promotes mammary cancer metastasis. J. Clin. Invest. *116*, 1561–1570.

Klaeger, S., Heinzlmeir, S., Wilhelm, M., Polzer, H., Vick, B., Koenig, P.-A., Reinecke, M., Ruprecht, B., Petzoldt, S., and Meng, C. (2017). The target landscape of clinical kinase drugs. Science *358*, eaan4368.

Sausgruber, N., Coissieux, M., Britschgi, A., Wyckoff, J., Aceto, N., Leroy, C., Stadler, M., Voshol, H., Bonenfant, D., and Bentires-Alj, M. (2015). Tyrosine phosphatase SHP2 increases cell motility in triple-negative breast cancer through the activation of SRC-family kinases. Oncogene *34*, 2272.

Schmidlin, T., Debets, D.O., van Gelder, C.A., Stecker, K.E., Rontogianni, S., van den Eshof, B.L., Kemper, K., Lips, E.H., van den Biggelaar, M., and Peeper, D.S. (2019). High-throughput assessment of kinome-wide activation states. Cell Syst. *9*, 366–374.e5.

Simiczyjew, A., Dratkiewicz, E., Van Troys, M., Ampe, C., Styczeń, I., and Nowak, D. (2018). Combination of EGFR inhibitor lapatinib and MET inhibitor foretinib inhibits migration of triple negative breast cancer cell lines. Cancers (Basel) *10*, 335.

Van Swearingen, A.E., Sambade, M.J., Siegel, M.B., Sud, S., McNeill, R.S., Bevill, S.M., Chen, X., Bash, R.E., Mounsey, L., and Golitz, B.T. (2017). Combined kinase inhibitors of MEK1/2 and either PI3K or PDGFR are efficacious in intracranial triple-negative breast cancer. Neuro Oncol. *19*, 1481–1493.

Zegzouti, H., Hennek, J., and Goueli, S.A. (2016). Using bioluminescent kinase profiling strips to identify kinase inhibitor selectivity and promiscuity. In Kinase Screening and Profiling, H. Zegzouti and S.A. Goueli, eds. (Springer), pp. 59–73.

Zhang, P. (1993). Model selection via multifold cross validation. Ann. Stat. *21*, 299–313.

Zhang, H., Ericksen, S.S., Ching-Pei, L., Ananiev, G.E., Wlodarchak, N., Mitchell, J.C., Gitter, A., Wright, S.J., Hoffmann, F.M., and Wildman, S.A. (2019). Predicting kinase inhibitors using bioactivity matrix derived informer sets. PLoS Comput. Biol. *15*, e1006813.

Zhavoronkov, A., Ivanenkov, Y.A., Aliper, A., Veselov, M.S., Aladinskiy, V.A., Aladinskaya, A.V., Terentiev, V.A., Polykovskiy, D.A., Kuznetsov, M.D., and Asadulaev, A. (2019). Deep learning enables rapid identification of potent DDR1 kinase inhibitors. Nat. Biotechnol. *37*, 1038–1040.

Zupan, J. (1994). Introduction to artificial neural network (ANN) methods: what they are and how to use them. Acta Chim. Slov. *41*, 327.

iScience, Volume *23*

Supplemental Information

Non-linear Deep Neural Network for Rapid and Accurate Prediction of Phenotypic Responses to Kinase Inhibitors

Siddharth Vijay and Taranjit S. Gujral

Non-linear Deep Neural Network for Rapid and Accurate Prediction of Phenotypic Responses to Kinase Inhibitors

Siddharth Vijay[1], and Taranjit S. Gujral[1,2*]

1. Division of Human Biology, Fred Hutchinson Cancer Research Center, Seattle, WA. U.S.A.

2. Department of Pharmacology, University of Washington, Seattle, WA. U.S.A.

*Lead contact. Correspondence: tgujral@fredhutch.org

Running title: Artificial neural networks predict response to kinase inhibitors in cancer cells

Keywords: kinase inhibitor/ deep learning/ neural networks/cell migration/

SUPPLEMENTAL DATA

TRANSPARENT METHODS

Model development & optimization of network hyperparameters

The development and implementation of KiDNN was accomplished using the python Keras framework with a TensorFlow backend (Chollet, 2018). Leave-One-Out-Cross-Validation (LOOCV) was the primary network evaluation method used in this investigation. In LOOCV, each time [n-1] drugs are used to train the network to predict the excluded inhibitor's effect on cell migration. The process is repeated a total of [n] times leaving out and predicting the effect on migration for every inhibitor. The average mean squared error (MSE) of all [n] inhibitors was used to cross-evaluate various networks. Since large errors in predicted and observed migration are undesirable as they can cause false positives and negatives, MSE was the primary error function used to optimize the network.

For KiDNN to reach peak predictive performance for a specific cell-line, the optimal network architecture and hyperparameters need to be chosen. The complete set of hyperparameters and their definitions are shown below. Epochs; the number of iterations KiDNN is supplied the entire training dataset. Typically, the network needs to be input the entire dataset multiple times to effectively fit the network to the data(Jayachandiran). Batch Size; the number of samples (rows/individual observations of the dataset) input through KiDNN before updating weights(Chollet, 2018). Initializer; the distribution in the start of the training process from which starting weights are assigned. Activation function; function applied to the weighted sum of inputs and biases to produce output (Dongare et al., 2012). Optimizer; the algorithm that helps the network converge to the optimal set of weights. Hidden layers; the quantity of layers (consisting of several nodes) in-between the output and input layer. Multiple hidden layers can ensure that the network can capture complex, non-linear dependence between input and output (Zupan, 1994). Nodes per hidden layer; the number of individual units (nodes) per every hidden layer. Dropout rate; the percentage of individual nodes in a layer that is temporarily removed from the network along with its connections. Dropout is a common method of preventing model overfitting (increased performance in training set, but poor performance in test sets) and nodes from co-adapting too much (Srivastava et al., 2014). A complete

list of all the specific values/names of the 8 hyperparameters are shown in **Table S1**. After optimization, the single top combination of the 8 hyperparameters is selected to build KiDNN. Besides these hyperparameters, the input and output layer remain set as there are 300 nodes in the input layer corresponding to the 300 kinases' inhibition measured in the activity profile and 1 output node for the predicted cell migration.

A common method of optimization used in numerous studies is Grid Search(Pontes et al., 2016), where all the various combinations of hyperparameter values are individually used to build several networks and the top combination is selected based on lowest MSE. Here, an exhaustive Grid Search optimizing all 8 hyperparameters at once wasn't a viable method because of the pure volume of required computations as more than 1,728,000 various networks would need to be evaluated. Consequently, a progressive, phase-by-phase Grid Search approach (evaluated with LOOCV MSE) was used where Grid Search was performed on 2 to 3 hyperparameters at once, rather than all 8. This significantly reduces the computation time to optimize KiDNN as only ~470 networks were evaluated. Since this method would prevent combinations of the top hyperparameter values across phases from being evaluated, a final phase 5 was performed, where the top 2 combinations of hyperparameter values were combined across multiple phases to select one final combination used to build the final KiDNN model. This multi-phase Grid Search is particularly effective because values of hyperparameters that perform poorly in terms of predictive performance are disregarded in future phases of the Grid Search, while also significantly reducing computational time to optimize KiDNN. Additionally, between phases, the top selected hyperparameters optimized in the previous phase were used in the successive phases by updating a baseline network. The initiation of the optimization process began with a completely unoptimized baseline network with default hyperparameters. The baseline neural network consisted of 300 input nodes for each kinase's activity, 2 hidden layers with 100 nodes per layer, and one output layer. The Adam optimizer, Rectified Linear Unit (ReLU) activation and Normal weight initialization distribution were used. Using the default baseline network, the network was optimized in 5 different phases to develop the final, fully-optimized KiDNN.

In phase 1, the range of overfitting was identified, where the model starts to fit the training data too much to the point where its ability to predict response of the test set was compromised. The baseline model was trained on response to 26 (~ 80%) randomly selected inhibitors in the training set and tested on the remaining 6 inhibitors (~20%) for 400 epochs. The fluctuation of MSE between predicted and observed migration of the 6 excluded inhibitors and the 26 inhibitors of the training set was measured as a function of the number of epochs. Using this data, a range of epochs is selected where MSE reaches a global minimum, before overfitting of the data starts to occur.

Using the range of epochs and batches sizes (listed in **Table S1**), Grid Search was performed to deduce the optimal combinations of epochs and batch size in phase 2. The top 5 combinations based on lowest LOOCV MSE are then re-run 5 additional times to ensure that the low MSE isn't due to chance by using average LOOCV MSE rather than single-iteration MSE. From the 5 additional runs, the 2 top combinations are identified based on average LOOCV MSE and are later used in Phase 5. In phase 3, the activation function, optimizers and weight initializers are optimized, while in phase 4, the number of hidden layers, nodes per hidden layer and Dropout rate were optimized. In both phase 3 and phase 4, the same process as executed in phase 2 was repeated.

As stated previously, the optimization process is progressive, where the top hyperparameters in one phase are used for the next. For example, the top combination of epochs and batch size selected in phase 2 were used to update the baseline network in phase 3, and the top combination of activation functions, optimizers and weight initializations in phase 3 were used to update the baseline network in phase 4. In the final phase 5, the top 2 sets of hyperparameter values across phases 2 through 4 are combined to create 8 (2^3) separate networks with various combinations of each set of hyperparameters from each phase using Grid Search. The 8 networks are run a total of 5 times and the network with the lowest average LOOCV MSE is selected as the final network architecture used to build KiDNN. The efficacy of the final KiDNN model was further evaluated using the LOOCV MSE of the predicted and observed values and the mean

standard deviation between 10 iterations of predictions to ensure a low-bias and high-precision neural network.

Supervised Deep Learning Approach

KiDNN was developed with Deep Neural Networks (DNNs), a non-linear, multi-layer feed-forward network. DNNs mimic the human brain, with processing nodes analogous to neurons in our brain and collections of neurons representing complete, multi-layered neural networks (Zupan, 1994). In KiDNN, these nodes are connected by weighted links, with all nodes, except those composing the input layer, receiving weighted sums of the output from the nodes in the previous layer and transmitting their output to nodes in successive layers until the final output layer (e.g. measured phenotypes such as cell migration) is reached (Dongare et al., 2012). The output transmitted to the successive nodes from a prior node is computed using the following three steps. First, the weighted sum of the output of nodes in the previous layer is computed, then biases are added, and lastly, an activation function is applied to the output limiting the output between a finite range (-1 to 1 or 0 to 1 in this study) (Dongare et al., 2012). This computation is repeated until the final layer is reached with the final predicted migration outputted. The computation performed in an individual node is shown below:

$$output = f\left(\sum_{i=1}^{n} x_i\, w_i + b\right)$$

input: $(x_1\ to\ x_n)$ *weights:* $(w_1\ to\ w_n)$ *bias:* (b) *activation function* (f)

Ultimately, KiDNN is trained on activity profiles by repeatedly computing errors of millions of combinations of weights, feeding it back to the network, and adjusting its weights accordingly until the optimal set of weights and biases between individual nodes are found where the error function between predicted and observed migration is minimal. Keeping these optimal weights and biases constant, remaining

untested inhibitors' activity profiles can be inputted through KiDNN and the predicted migration can be computed.

Applying KiDNN to naïve datasets (Hs578t & FOCUS)

After selecting the optimal architecture and hyperparameters to develop KiDNN for Hs578t, the network was applied to completely unseen inhibitors to predict migration. The network was trained on the 32 inhibitors' activity profile and their resulting migration to predict migration in the 178 other inhibitors. The same method of optimization used to optimize KiDNN for Hs578t was used to optimize KiDNN for a new cell line (FOCUS). After the optimal hyperparameters were found, KiDNN was trained on the 32 inhibitors' activity profile and their resulting migration to predict migration in the 178 other inhibitors for FOCUS.

Cell lines

Hs578t was obtained from American Type Culture Collection. Hepatocellular carcinoma cell line (FOCUS) was obtained from J. Wands (Brown University). Both cell lines were grown at 37°C under 5% CO2, 95% ambient atmosphere and maintained in Dulbecco's minimum essential medium (DMEM) supplemented with 10% FBS (Sigma).

Kinetic cell migration assay

To study the effect of kinase inhibitors on migration of Hs578t or FOCUS cells a wound-healing assay was employed as described previously (Gujral et al., 2014b). The details on the structure and preparation of this dataset has been disclosed previously (Gujral et al., 2014b). Briefly, cells were plated on 96-well plates (Essen Image Lock, Essen Instruments), and a wound was scratched with wound scratcher (Essen Instruments). Inhibitors at different doses were added immediately after wound scratching, wound confluence was monitored with Incucyte Live-Cell Imaging System and software (Essen Instruments).

Wound closure was observed every 2 hours for 24-72 hours by comparing the mean relative wound density of at least three biological replicates in each experiment.

Fig. S1

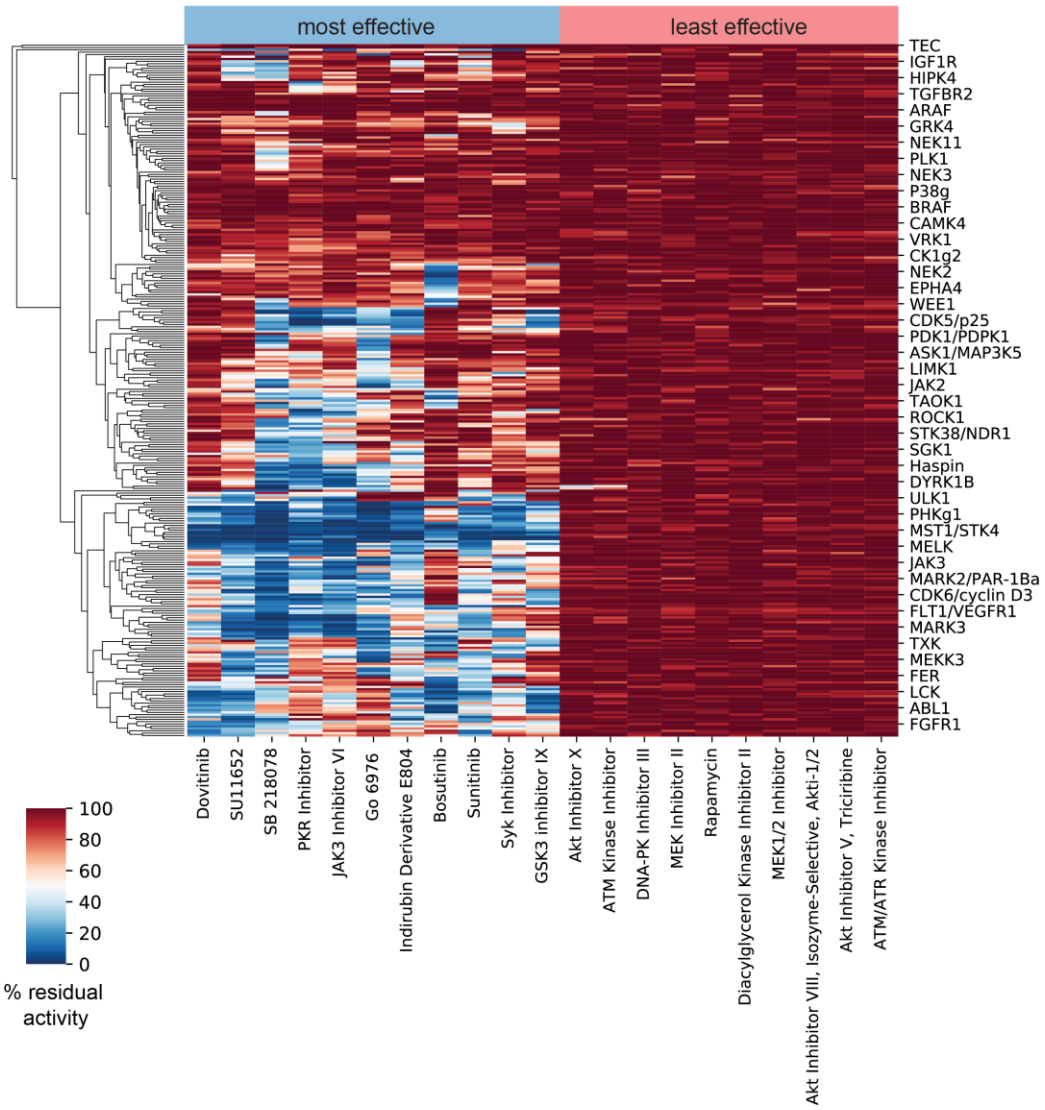


Figure S1. Kinase target profiles of the most effective KiDNN-predicted inhibitors, Related to Figure 4. A heatmap of showing kinase targets profiles of the most effective and the least effective KiDNN predicted inhibitors.

Table S1. List of Parameters Optimized, Related to Figure 2. Epochs were selected from the training and validation loss plot (Fig. 2B) by choosing 3 values above and 3 below the 125 epochs at which overfitting was observed. Batch size is an integer. Various types of kernel initializers, optimizer and activation functions were evaluated.

Epochs	*Batch Size*	*Weight Initializer*	*Optimizer*	*Activation*	*Hidden Layers (HL)*	*Nodes per HL*	*Dropout Rate*
50	1	Uniform	RMSprop	Sigmoid	1	10	0
75	2	Truncated Normal	Adagrad	TanH	2	25	0.05
100	4	Normal	Adamax	ReLU	3	50	0.1
125	8	Lecun Uniform	Adadelta	ELU		100	0.2
150	16	Glorot Normal	Adam	SELU		150	0.5
175	32	He Normal	Nadam			200	
200		Glorot Uniform				250	
		Variance Scaling				300	
		He Uniform					
		Orthogonal					

ReLU; rectified linear unit, ELU; exponential linear unit, SELU; scaled exponential linear unit, TanH; hyperbolic tangent
RMSprop; root mean square propagation, Adagrad; adaptive gradient, Adam; Adaptive moment estimation, Nadam; Nesterov-accelerated adaptive moment estimation

Table S2. Network Evaluation of Top Batch Size and Epoch Combinations, Related to Figure 2. The top 2 combinations are shaded. The one that was used for subsequent optimization is indicated in bold.

Batch Size	*Epoch*	*Average MSE*
8	50	106.41
8	125	111.92
16	50	101.47
32	**75**	**100.48**
32	125	103.00

Table S3. Network Evaluation of Top Weight Initializers, Optimizers and Activation Function Combinations, Related to Figure 2. The top 2 combinations are shaded. The one that was used for subsequent optimization is indicated in bold.

Weight Initializer	*Optimizer*	*Activation*	*Mean MSE*
Uniform	Adagrad	ReLU	99.4
Truncated Normal	Adagrad	ReLU	94.6
Truncated Normal	**Adagrad**	**ELU**	**92.9**
Lecun Uniform	Adamax	ELU	107.3
Variance Scaling	Nadam	ReLU	101.7

ReLU; rectified linear unit, ELU; exponential linear unit, Adagrad; adaptive gradient, Adam; Adaptive moment estimation, Nadam; Nesterov-accelerated adaptive moment estimation

Table S4. Network Evaluation of Top Hidden Layer Quantity, Nodes and Dropout Rate Combinations, Related to Figure 2. The top 2 combinations are shaded. The one that was used for subsequent optimization is indicated in bold.

Hidden Layers (HL)	*Nodes per HL*	*Dropout Rate*	*Average MSE*
2	200	0	89.9
2	**300**	**0**	**85.7**
2	200	0.05	91.4
3	250	0	94.0
3	50	0.05	92.8

Table S5. Predicted migration of Hs578t cells in response to top 20 most effective kinase inhibitors selected by KiDNN, Related to Figure 4

Rank	*Kinase Inhibitor*	*Predicted Migration*
1	Staurosporine	12.3
2	K-252a	28.1
3	SB 218078	35.9
4	Cdk1/2 Inhibitor III	37.5
5	PKR Inhibitor	46.8
6	JAK3 Inhibitor VI	48.1
7	SU11652	48.6
8	Go 6976	49.9
9	Indirubin Derivative E804	50.2
10	Staurosporine, N-benzoyl-	51.4
11	Bosutinib	52.7
12	Sunitinib	54.2
13	Syk Inhibitor	56.3
14	GSK3 inhibitor IX	57.0
15	JAK Inhibitor I	57.5
16	Dasatinib	57.5
17	AMPK Inhibitor, Compound C	57.8
18	Dovitinib	58.2
19	Aurora Kinase/Cdk Inhibitor	61.8
20	Indirubin-3’-monoxime	61.8

Table S6. Predicted migration of FOCUS cells in response to top 20 most effective kinase inhibitors selected by KiDNN-FOCUS, Related to Figure 5

Rank	*Kinase Inhibitor*	*Predicted Migration*
1	Staurosporine	5.0
2	Dasatinib	14.2
3	Dovitinib	33.8
4	Bosutinib	34.5
5	Staurosporine, N-benzoyl-	38.8
6	LCK inhibitor	38.8
7	GSK3 inhibitor IX	41.6
8	SB 218078	43.1
9	SU11652	48.5
10	Indirubin Derivative E804	48.6
11	PDGFR RTK inhibitor	52.0
12	K-252a	53.4
13	PDK1/Akt/Flt Dual Pathway Inhibitor	55.5
14	GSK-3 Inhibitor X	55.9
15	Syk Inhibitor	56.4
16	Sunitinib	57.6
17	Flt-3 Inhibitor II	58.0
18	TWS119	59.2
19	Tozasertib	61.3
20	GSK-3 Inhibitor XIII	61.4

Table S7. Predicted and measured migration of Hs578t cells in response to all 40 kinase inhibitors, Related to Figure 4.

Kinase Inhibitor	*Measured Migration*	*KiDNN Predictions*	*KiR Predictions*
Bosutinib	39.9	52.7	42.5
Casein Kinase I Inhibitor D44	70	70.0	70.0
Cdk1/2 Inhibitor III	36.2	37.5	35.5
Dasatinib	21.2	57.5	28.7
EGFR ErbB-2 Erbb-4	70	70.0	68.7
Erlotinib	70	70.0	70.0
Go 6983	70	67.8	67.2
Gefitinib	70	70.0	66.5
GSK3 inhibitor IX	49	57.0	51.7
GSK-3b Inhibitor I	70	70.0	70.0
H-89	70	70.0	68.3
Imatinib	70	70.0	70.0
JNK Inhibitor II	70	68.1	66.9
K-252a	27.5	28.1	27.6
Lapatinib	70	70.0	70.0
LCK inhibitor	70	68.5	68.3
LY294002	70	70.0	70.0
Masitinib	70	70.0	68.4
Met Kinase Inhibitor	68.6	65.3	67.5
Nilotinib	70	68.2	67.8
PDGFR RTK inhibitor	69	67.7	67.6
Rapamycin	70	70.0	68.5

ROCK Inhibitor	70	70.0	68.9
SB 218078	29.4	35.9	30.0
SB220025	56.9	70.0	57.6
Sorafenib	65.4	70.0	68.4
Src Kinase Inhibitor I	70	70.0	69.0
Staurosporine	0.98	12.3	6.0
Sunitinib	54.3	54.2	52.9
Tofacitinib	70	70.0	70.0
TWS119	68.3	62.2	61.2
Vandetanib	66.6	65.3	62.4
Aminopurvalanol A	70	68.6	61.8
Staurosporine, N-benzoyl	56.3	51.4	45.5
AMPK Inhibitor Compound C	65.8	57.8	56.6
PDK1/Akt/Flt Dual Pathway Inhibitor	70	63.6	65.0
SU11652	44.4	48.6	50.8
JAK Inhibitor I	70	57.5	46.8
PD 98059	70	70.0	70.0
Dovitinib	55.7	58.2	54.9
Mean Squared Error		***38.91***	***109.39***
Mean Absolute Error		***4.99***	***7.95***

Table S8. Predicted and measured migration of FOCUS cells in response to all 39 kinase inhibitors, Related to Figure 5.

Kinase Inhibitor	*Measured Migration*	*KiDNN Prediction*	*KiR Prediction*
Staurosporine	4.5	5.0	20.1
Dasatinib	11.6	14.2	25.2
Bosutinib	33.4	34.5	39.7
LCK inhibitor	36.5	38.8	42.0
GSK3 inhibitor IX	40.9	41.6	49.5
SB 218078	42.0	43.1	45.0
PDGFR RTK inhibitor	47.7	52.0	49.7
K-252a	52.4	53.4	43.2
Sunitinib	56.2	57.6	57.0
Sorafenib	58.1	64.7	65.6
TWS119	58.3	59.2	47.3
Vandetanib	64.2	65.9	58.6
EGFR ErbB-2 Erbb-4	67.3	67.1	70.0
Nilotinib	67.5	65.7	60.5
Go 6983	68.0	68.9	68.5
Src Kinase Inhibitor I	68.5	66.7	60.8
Gefitinib	68.9	70.0	67.2
SB220025	69.7	69.9	69.6
Casein Kinase I Inhibitor D44	70.0	70.0	70.0
Cdk1/2 Inhibitor III	70.0	70.0	61.9
Erlotinib	70.0	70.0	70.0
GSK-3b Inhibitor I	70.0	70.0	70.0

H-89	70.0	68.8	70.0
Imatinib	70.0	70.0	68.9
JNK Inhibitor II	70.0	70.0	69.3
Lapatinib	70.0	70.0	70.0
LY294002	70.0	70.0	70.0
Masitinib	70.0	65.6	62.9
Met Kinase Inhibitor	70.0	70.0	65.4
Rapamycin	70.0	70.0	68.5
ROCK Inhibitor	70.0	70.0	70.0
Tofacitinib	70.0	70.0	70.0
Aminopurvalanol A	66.78	64.18	60.34
AMPK Compound C	64.82	57.12	54.46
Cdk2 Inhibitor IV, NU6140	70	66.31	52.71
Dovitinib	32.34	51.26	33.6
GSK-3 Inhibitor XIII	59.5	59.93	53.69
Staurosporine, N-benzoyl-	37.94	55.32	30.87
SU11652	47.04	49.41	20.72
Mean Squared Error		**106.48**	**175.12**

Data S1. KiDNN code, Related to Figures 1, 2 and 3.

Predicting Effect of Untested Kinase Inhibitors on Hs578t Cell Migration

```
# Importing the libraries
import numpy as np
import pandas as pd

#Importing migration and activity profile data
response_data = pd.read_csv('hs578t_migration.csv')
drug_list = response_data.iloc[:, 0].values
alldrugs = pd.read_csv('allDrugs_migration.csv', encoding='latin1')
alldrugs = alldrugs.set_index('compound')
dataset = alldrugs.loc[drug_list]
response = response_data['Hs578t'].values
dataset["migration"] = response

# Slicing the dataset
X = dataset.iloc[:, 0:300].values
y = dataset.iloc[:, 300].values


# Importing the Keras libraries and packages
import keras
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout

# Initializing the ANN
classifier = Sequential()

# Adding the input layer and the first hidden layer
classifier.add(Dense(units = 200, kernel_initializer = 'TruncatedNormal', activation = 'elu',
input_dim = 300))

# Adding the second hidden layer
classifier.add(Dense(units = 200, kernel_initializer = 'TruncatedNormal', activation = 'elu'))

# Adding the output layer
classifier.add(Dense(units = 1, kernel_initializer = 'TruncatedNormal' ))

# Compiling the ANN
classifier.compile(loss = 'mean_squared_error', optimizer='adagrad')

# Fitting the ANN to the Training set
classifier.fit(X, y, batch_size = 32, epochs = 75)
```

```
# Predicting effect on cell migration for all 178 kinase inhibitors
X_predict = alldrugs
prediction_index = X_predict.index.tolist()
X_predict = X_predict.iloc[:, 0:300].values
y_pred = classifier.predict(X_predict)
```

Training & Validation Loss – Hs578t

```
# Importing the libraries
import numpy as np
import pandas as pd
import keras
from keras.models import Sequential
from keras.layers import Dense
import matplotlib.pyplot as plt

# Importing migration and activity profile data
response_data2 = pd.read_csv('hs578t_migration.csv')
drug_list2 = response_data2.iloc[:, 0].values
alldrugs2 = pd.read_csv('allDrugs_migration.csv', encoding='latin1')
alldrugs2 = alldrugs2.set_index('compound')
dataset2 = alldrugs2.loc[drug_list2]
response2 = response_data2['Hs578t'].values
dataset2["migration"] = response2

# Importing the dataset
X2 = dataset2.iloc[:, 0:300].values
y2 = dataset2.iloc[:, 300].values

#building model function
def build_classifier():
    classifier = Sequential()
    classifier.add(Dense(units = 100, kernel_initializer = 'normal', activation = 'relu', input_dim =
300))
    classifier.add(Dense(units = 100, kernel_initializer = 'normal', activation = 'relu'))
    classifier.add(Dense(units = 1, kernel_initializer = 'normal'))
    classifier.compile(loss = 'mean_squared_error', optimizer= 'adam', metrics=[ 'mse'])
    return classifier
classifier = build_classifier()
history = classifier.fit(X2, y2, validation_split= .2, epochs=500, batch_size=1, verbose=0)
hist = pd.DataFrame(history.history)
print(history.history.keys())

#plotting training and validation MSE as a function of epochs
```

```
def plot_history(history):
  hist = pd.DataFrame(history.history)
  hist['epoch'] = history.epoch

  plt.figure()
  plt.xlabel('Epoch')
  plt.ylabel('Mean Squared Error')
  plt.plot(hist['epoch'], hist['mean_squared_error'],
           label='Train Error')
  plt.plot(hist['epoch'], hist['val_mean_squared_error'],
           label = 'Validation Error')
  z = np.polyfit(hist['epoch'].values, hist['val_mean_squared_error'].values, 5)
  f = np.poly1d(z)
  x_new = np.linspace(hist['epoch'].values[0], hist['epoch'].values[-1], 50)
  y_new = f(x_new)
  plt.plot(x_new, y_new, label = 'Polynomial Fit')
  plt.title("Training & Validation MSE")
  plt.ylim([0,200])
  plt.xlim([0,450])
  plt.legend()

  plt.savefig("Training & Validation MSE.svg")
  plt.show()


plot_history(history)

# exporting data
best_fit_df = pd.DataFrame({"xnew": x_new, "ynew": y_new})
best_fit_df.to_excel("best_fit_line.xlsx", sheet_name='1')
hist.to_excel("Train & Val Loss.xlsx", sheet_name='1')
```

Epoch & Batch Size Optimization – Hs578t

```
# Importing the libraries
import numpy as np
import pandas as pd
import keras
from keras.models import Sequential
from keras.layers import Dense
from keras import initializers
from keras.layers import Dropout
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from scipy.stats.stats import pearsonr
```

```
#Importing migration and activity profile data
response_data2 = pd.read_csv('hs578t_migration.csv')
drug_list2 = response_data2.iloc[:, 0].values
alldrugs2 = pd.read_csv('alldrugs_migration.csv', encoding='latin1')
alldrugs2 = alldrugs2.set_index('compound')
dataset2 = alldrugs2.loc[drug_list2]
response2 = response_data2['Hs578t'].values
dataset2["migration"] = response2

# Slicing the dataset
X2 = dataset2.iloc[:, 0:300].values
y2 = dataset2.iloc[:, 300].values

def cross_val( hl_units, init, opt, batch_size, epochs, act, dropout):
    y_pred_all = []
    y_test_all = []
    for num in range(len(X2)):
        y_test = y2[num]
        X_test = X2[num, :]
        X_test = np.array([X_test])
        X_test.T
        X_train = np.delete(X2, (num), axis=0)
        y_train = np.delete(y2, (num), axis=0)
        classifier = Sequential()
        classifier.add(Dense(units = hl_units, kernel_initializer = init, input_dim = 300, activation =
act))
        classifier.add(Dense(units = hl_units, kernel_initializer = init, activation = act))
        classifier.add(Dense(units = 1, kernel_initializer = init))
        classifier.compile(loss = 'mean_squared_error', optimizer = opt)
        classifier.fit(X_train, y_train, batch_size = batch_size, epochs = epochs)
        y_pred = classifier.predict(X_test)
        y_pred_all.append(y_pred[0][0])
        y_test_all.append(y_test)

    return (mean_squared_error(y_pred_all, y_test_all), mean_absolute_error(y_pred_all,
y_test_all), pearsonr(y_pred_all, y_test_all)[0])


ep = [50,75,100,125,150,175,200]
bs = [1,2,4,8,16,32]

scores = []

#Performing Grid Search
for epoch in ep:
```

```
    for batch_size in bs:
        scores.append(cross_val(100, 'normal', 'adam', batch_size, epoch, 'relu', 0.0))
```

Weight Initialization, Optimizer, Activation Function Optimization – Hs578t

```
# Importing the libraries
import numpy as np
import pandas as pd
import keras
from keras.models import Sequential
from keras.layers import Dense
from keras import initializers
from keras.layers import Dropout
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from scipy.stats.stats import pearsonr

#Importing migration and activity profile data
response_data2 = pd.read_csv('hs578t_migration.csv')
drug_list2 = response_data2.iloc[:, 0].values
alldrugs2 = pd.read_csv('alldrugs_migration.csv', encoding='latin1')
alldrugs2 = alldrugs2.set_index('compound')
dataset2 = alldrugs2.loc[drug_list2]
response2 = response_data2['Hs578t'].values
dataset2["migration"] = response2

# Slicing the dataset
X2 = dataset2.iloc[:, 0:300].values
y2 = dataset2.iloc[:, 300].values

def cross_val( hl_units, init, opt, batch_size, epochs, act, dropout):
    y_pred_all = []
    y_test_all = []
    for num in range(len(X2)):
        y_test = y2[num]
        X_test = X2[num, :]
        X_test = np.array([X_test])
        X_test.T
        X_train = np.delete(X2, (num), axis=0)
        y_train = np.delete(y2, (num), axis=0)
        classifier = Sequential()
        classifier.add(Dense(units = hl_units, kernel_initializer = init, input_dim = 300, activation =
act))
        classifier.add(Dense(units = hl_units, kernel_initializer = init, activation = act))
```

```
        classifier.add(Dense(units = 1, kernel_initializer = init))
        classifier.compile(loss = 'mean_squared_error', optimizer = opt)
        classifier.fit(X_train, y_train, batch_size = batch_size, epochs = epochs)
        y_pred = classifier.predict(X_test)
        y_pred_all.append(y_pred[0][0])
        y_test_all.append(y_test)

    return (mean_squared_error(y_pred_all, y_test_all), mean_absolute_error(y_pred_all,
y_test_all), pearsonr(y_pred_all, y_test_all)[0])


initializer = ['uniform', 'TruncatedNormal', 'normal', 'lecun_uniform', 'glorot_normal',
'he_normal', 'glorot_uniform', 'VarianceScaling', 'orthogonal', 'he_uniform']
optimizer = ['rmsprop','adagrad', 'adamax', 'adadelta', 'adam', 'nadam']
activation = ['sigmoid', 'tanh', 'relu', 'elu', 'selu']

scores = []

#Performing Grid Search
for init in initializer:
    for opt in optimizer:
            for act in activation:
                    scores.append(cross_val(100, init, opt, 32, 75, act, 0.0))
```

Hidden layer, Nodes per Hidden Layer and Dropout Rate Optimization – Hs578t

```
# Importing the libraries
import numpy as np
import pandas as pd
import keras
from keras.models import Sequential
from keras.layers import Dense
from keras import initializers
from keras.layers import Dropout
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from scipy.stats.stats import pearsonr

#Importing migration and activity profile data
response_data2 = pd.read_csv('hs578t_migration.csv')
drug_list2 = response_data2.iloc[:, 0].values
alldrugs2 = pd.read_csv('alldrugs_migration.csv', encoding='latin1')
alldrugs2 = alldrugs2.set_index('compound')
dataset2 = alldrugs2.loc[drug_list2]
response2 = response_data2['Hs578t'].values
```

```
dataset2["migration"] = response2

# Slicing the dataset
X2 = dataset2.iloc[:, 0:300].values
y2 = dataset2.iloc[:, 300].values

#LOOCV function for 1 hidden layer
def cross_val1( hl_units, init, opt, batch_size, epochs, act, dropout):
    y_pred_all = []
    y_test_all = []
    for num in range(len(X2)):
        y_test = y2[num]
        X_test = X2[num, :]
        X_test = np.array([X_test])
        X_test.T
        X_train = np.delete(X2, (num), axis=0)
        y_train = np.delete(y2, (num), axis=0)
        classifier = Sequential()
        classifier.add(Dense(units = hl_units, kernel_initializer = init, input_dim = 300, activation =
act))
        classifier.add(Dropout(dropout))
        classifier.add(Dense(units = 1, kernel_initializer = init))
        classifier.compile(loss = 'mean_squared_error', optimizer = opt)
        classifier.fit(X_train, y_train, batch_size = batch_size, epochs = epochs)
        y_pred = classifier.predict(X_test)
        y_pred_all.append(y_pred[0][0])
        y_test_all.append(y_test)

    return (mean_squared_error(y_pred_all, y_test_all), mean_absolute_error(y_pred_all,
y_test_all), pearsonr(y_pred_all, y_test_all)[0])

#LOOCV function for 2 hidden layers
def cross_val2( hl_units, init, opt, batch_size, epochs, act, dropout):
    y_pred_all = []
    y_test_all = []
    for num in range(len(X2)):
        y_test = y2[num]
        X_test = X2[num, :]
        X_test = np.array([X_test])
        X_test.T
        X_train = np.delete(X2, (num), axis=0)
        y_train = np.delete(y2, (num), axis=0)
        classifier = Sequential()
        classifier.add(Dense(units = hl_units, kernel_initializer = init, input_dim = 300, activation =
act))
        classifier.add(Dropout(dropout))
```

```
        classifier.add(Dense(units = hl_units, kernel_initializer = init, input_dim = 300, activation =
act))
        classifier.add(Dropout(dropout))
        classifier.add(Dense(units = 1, kernel_initializer = init))
        classifier.compile(loss = 'mean_squared_error', optimizer = opt)
        classifier.fit(X_train, y_train, batch_size = batch_size, epochs = epochs)
        y_pred = classifier.predict(X_test)
        y_pred_all.append(y_pred[0][0])
        y_test_all.append(y_test)

    return (mean_squared_error(y_pred_all, y_test_all), mean_absolute_error(y_pred_all,
y_test_all), pearsonr(y_pred_all, y_test_all)[0])

#LOOCV function for 3 hidden layers
def cross_val3( hl_units, init, opt, batch_size, epochs, act, dropout):
    y_pred_all = []
    y_test_all = []
    for num in range(len(X2)):
        y_test = y2[num]
        X_test = X2[num, :]
        X_test = np.array([X_test])
        X_test.T
        X_train = np.delete(X2, (num), axis=0)
        y_train = np.delete(y2, (num), axis=0)
        classifier = Sequential()
        classifier.add(Dense(units = hl_units, kernel_initializer = init, input_dim = 300, activation =
act))
        classifier.add(Dropout(dropout))
        classifier.add(Dense(units = hl_units, kernel_initializer = init, input_dim = 300, activation =
act))
        classifier.add(Dropout(dropout))
        classifier.add(Dense(units = hl_units, kernel_initializer = init, input_dim = 300, activation =
act))
        classifier.add(Dropout(dropout))
        classifier.add(Dense(units = 1, kernel_initializer = init))
        classifier.compile(loss = 'mean_squared_error', optimizer = opt)
        classifier.fit(X_train, y_train, batch_size = batch_size, epochs = epochs)
        y_pred = classifier.predict(X_test)
        y_pred_all.append(y_pred[0][0])
        y_test_all.append(y_test)

    return (mean_squared_error(y_pred_all, y_test_all), mean_absolute_error(y_pred_all,
y_test_all), pearsonr(y_pred_all, y_test_all)[0])


#List of possible values
```

```
nodes_per_hidden_layer = [10,25,50,100,150,200,250,300]
dropout_rate = [0,0.05,0.1,0.2,0.5]

scores = []

#Performing Grid Search
for nodes in nodes_per_hidden_layer:
   for dr in dropout_rate:
         scores.append(cross_val1(nodes, ‘TruncatedNormal’, ‘adagrad’, 32, 75, ‘elu’, dr))

for nodes in nodes_per_hidden_layer:
   for dr in dropout_rate:
         scores.append(cross_val2(nodes, ‘TruncatedNormal’, ‘adagrad’, 32, 75, ‘elu’, dr))

for nodes in nodes_per_hidden_layer:
   for dr in dropout_rate:
         scores.append(cross_val3(nodes, ‘TruncatedNormal’, ‘adagrad’, 32, 75, ‘elu’, dr))
```

Predicting Effect of Untested Kinase Inhibitors on FOCUS Cell Migration

```
# Importing the libraries
import numpy as np
import pandas as pd

#Importing migration and activity profile data
response_data = pd.read_csv('focus_migration.csv')
drug_list = response_data.iloc[:, 0].values
alldrugs = pd.read_csv('allDrugs_migration.csv', encoding='latin1')
alldrugs = alldrugs.set_index('compound')
dataset = alldrugs.loc[drug_list]
response = response_data['FOCUS'].values
dataset["migration"] = response

# Slicing the dataset
X = dataset.iloc[:, 0:300].values
y = dataset.iloc[:, 300].values


# Importing the Keras libraries and packages
import keras
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
```

```
# Initializing the ANN
classifier = Sequential()

# Adding the input layer and the first hidden layer
classifier.add(Dense(units = 200, kernel_initializer = 'TruncatedNormal', activation = 'elu',
input_dim = 300))

# Adding the second hidden layer
classifier.add(Dense(units = 200, kernel_initializer = 'TruncatedNormal', activation = 'elu'))

# Adding the output layer
classifier.add(Dense(units = 1, kernel_initializer = 'TruncatedNormal' ))

# Compiling the ANN
classifier.compile(loss = 'mean_squared_error', optimizer='adagrad')

# Fitting the ANN to the Training set
classifier.fit(X, y, batch_size = 32, epochs = 75)

# Predicting effect on cell migration for all 178 kinase inhibitors
X_predict = alldrugs
prediction_index = X_predict.index.tolist()
X_predict = X_predict.iloc[:, 0:300].values
y_pred = classifier.predict(X_predict)
```

Training & Validation Loss – FOCUS

```
# Importing the libraries
import numpy as np
import pandas as pd
import keras
from keras.models import Sequential
from keras.layers import Dense
import matplotlib.pyplot as plt

# Importing migration and activity profile data
response_data2 = pd.read_csv('focus_migration.csv')
drug_list2 = response_data2.iloc[:, 0].values
alldrugs2 = pd.read_csv('allDrugs_migration.csv', encoding='latin1')
alldrugs2 = alldrugs2.set_index('compound')
dataset2 = alldrugs2.loc[drug_list2]
response2 = response_data2['FOCUS'].values
dataset2["migration"] = response2
```

```
# Importing the dataset
X2 = dataset2.iloc[:, 0:300].values
y2 = dataset2.iloc[:, 300].values

#building model function
def build_classifier():
    classifier = Sequential()
    classifier.add(Dense(units = 100, kernel_initializer = 'normal', activation = 'relu', input_dim =
300))
    classifier.add(Dense(units = 100, kernel_initializer = 'normal', activation = 'relu'))
    classifier.add(Dense(units = 1, kernel_initializer = 'normal'))
    classifier.compile(loss = 'mean_squared_error', optimizer= 'adam', metrics=[ 'mse'])
    return classifier
classifier = build_classifier()
history = classifier.fit(X2, y2, validation_split= .2, epochs=500, batch_size=1, verbose=0)
hist = pd.DataFrame(history.history)
print(history.history.keys())

#plotting training and validation MSE as a function of epochs
def plot_history(history):
  hist = pd.DataFrame(history.history)
  hist['epoch'] = history.epoch

  plt.figure()
  plt.xlabel('Epoch')
  plt.ylabel('Mean Squared Error')
  plt.plot(hist['epoch'], hist['mean_squared_error'],
           label='Train Error')
  plt.plot(hist['epoch'], hist['val_mean_squared_error'],
           label = 'Validation Error')
  z = np.polyfit(hist['epoch'].values, hist['val_mean_squared_error'].values, 5)
  f = np.poly1d(z)
  x_new = np.linspace(hist['epoch'].values[0], hist['epoch'].values[-1], 50)
  y_new = f(x_new)
  plt.plot(x_new, y_new, label = 'Polynomial Fit')
  plt.title("Training & Validation MSE")
  plt.ylim([0,200])
  plt.xlim([0,450])
  plt.legend()

  plt.savefig("Training & Validation MSE.svg")
  plt.show()


plot_history(history)
```

```
# exporting data
best_fit_df = pd.DataFrame({"xnew": x_new, "ynew": y_new})
best_fit_df.to_excel("best_fit_line.xlsx", sheet_name='1')
hist.to_excel("Train & Val Loss.xlsx", sheet_name='1')
```

Epoch & Batch Size Optimization – FOCUS

```
# Importing the libraries
import numpy as np
import pandas as pd
import keras
from keras.models import Sequential
from keras.layers import Dense
from keras import initializers
from keras.layers import Dropout
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from scipy.stats.stats import pearsonr

#Importing migration and activity profile data
response_data2 = pd.read_csv('focus_migration.csv')
drug_list2 = response_data2.iloc[:, 0].values
alldrugs2 = pd.read_csv('alldrugs_migration.csv', encoding='latin1')
alldrugs2 = alldrugs2.set_index('compound')
dataset2 = alldrugs2.loc[drug_list2]
response2 = response_data2['FOCUS'].values
dataset2["migration"] = response2

# Slicing the dataset
X2 = dataset2.iloc[:, 0:300].values
y2 = dataset2.iloc[:, 300].values

def cross_val( hl_units, init, opt, batch_size, epochs, act, dropout):
    y_pred_all = []
    y_test_all = []
    for num in range(len(X2)):
        y_test = y2[num]
        X_test = X2[num, :]
        X_test = np.array([X_test])
        X_test.T
        X_train = np.delete(X2, (num), axis=0)
        y_train = np.delete(y2, (num), axis=0)
        classifier = Sequential()
        classifier.add(Dense(units = hl_units, kernel_initializer = init, input_dim = 300, activation =
act))
```

```
        classifier.add(Dense(units = hl_units, kernel_initializer = init, activation = act))
        classifier.add(Dense(units = 1, kernel_initializer = init))
        classifier.compile(loss = 'mean_squared_error', optimizer = opt)
        classifier.fit(X_train, y_train, batch_size = batch_size, epochs = epochs)
        y_pred = classifier.predict(X_test)
        y_pred_all.append(y_pred[0][0])
        y_test_all.append(y_test)

    return (mean_squared_error(y_pred_all, y_test_all), mean_absolute_error(y_pred_all,
y_test_all), pearsonr(y_pred_all, y_test_all)[0])


ep = [50,75,100,125,150,175,200]
bs = [1,2,4,8,16,32]

scores = []

#Performing Grid Search
for epoch in ep:
    for batch_size in bs:
        scores.append(cross_val(100, 'normal', 'adam', batch_size, epoch, 'relu', 0.0))
```

Weight Initialization, Optimizer, Activation Function Optimization – FOCUS

```
# Importing the libraries
import numpy as np
import pandas as pd
import keras
from keras.models import Sequential
from keras.layers import Dense
from keras import initializers
from keras.layers import Dropout
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from scipy.stats.stats import pearsonr

#Importing migration and activity profile data
response_data2 = pd.read_csv('focus_migration.csv')
drug_list2 = response_data2.iloc[:, 0].values
alldrugs2 = pd.read_csv('alldrugs_migration.csv', encoding='latin1')
alldrugs2 = alldrugs2.set_index('compound')
dataset2 = alldrugs2.loc[drug_list2]
response2 = response_data2['FOCUS'].values
dataset2["migration"] = response2
```

```
# Slicing the dataset
X2 = dataset2.iloc[:, 0:300].values
y2 = dataset2.iloc[:, 300].values

def cross_val( hl_units, init, opt, batch_size, epochs, act, dropout):
    y_pred_all = []
    y_test_all = []
    for num in range(len(X2)):
        y_test = y2[num]
        X_test = X2[num, :]
        X_test = np.array([X_test])
        X_test.T
        X_train = np.delete(X2, (num), axis=0)
        y_train = np.delete(y2, (num), axis=0)
        classifier = Sequential()
        classifier.add(Dense(units = hl_units, kernel_initializer = init, input_dim = 300, activation =
act))
        classifier.add(Dense(units = hl_units, kernel_initializer = init, activation = act))
        classifier.add(Dense(units = 1, kernel_initializer = init))
        classifier.compile(loss = 'mean_squared_error', optimizer = opt)
        classifier.fit(X_train, y_train, batch_size = batch_size, epochs = epochs)
        y_pred = classifier.predict(X_test)
        y_pred_all.append(y_pred[0][0])
        y_test_all.append(y_test)

    return (mean_squared_error(y_pred_all, y_test_all), mean_absolute_error(y_pred_all,
y_test_all), pearsonr(y_pred_all, y_test_all)[0])


initializer = ['uniform', 'TruncatedNormal', 'normal', 'lecun_uniform', 'glorot_normal',
'he_normal', 'glorot_uniform', 'VarianceScaling', 'orthogonal', 'he_uniform']
optimizer = ['rmsprop','adagrad', 'adamax', 'adadelta', 'adam', 'nadam']
activation = ['sigmoid', 'tanh', 'relu', 'elu', 'selu']

scores = []

#Performing Grid Search
for init in initializer:
    for opt in optimizer:
        for act in activation:
            scores.append(cross_val(100, init, opt, 2, 120, act, 0.0))
```

Hidden layer, Nodes per Hidden Layer and Dropout Rate Optimization – FOCUS

```
# Importing the libraries
import numpy as np
import pandas as pd
import keras
from keras.models import Sequential
from keras.layers import Dense
from keras import initializers
from keras.layers import Dropout
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from scipy.stats.stats import pearsonr

#Importing migration and activity profile data
response_data2 = pd.read_csv('focus_migration.csv')
drug_list2 = response_data2.iloc[:, 0].values
alldrugs2 = pd.read_csv('alldrugs_migration.csv', encoding='latin1')
alldrugs2 = alldrugs2.set_index('compound')
dataset2 = alldrugs2.loc[drug_list2]
response2 = response_data2['FOCUS'].values
dataset2["migration"] = response2

# Slicing the dataset
X2 = dataset2.iloc[:, 0:300].values
y2 = dataset2.iloc[:, 300].values

#LOOCV function for 1 hidden layer
def cross_val1( hl_units, init, opt, batch_size, epochs, act, dropout):
    y_pred_all = []
    y_test_all = []
    for num in range(len(X2)):
        y_test = y2[num]
        X_test = X2[num, :]
        X_test = np.array([X_test])
        X_test.T
        X_train = np.delete(X2, (num), axis=0)
        y_train = np.delete(y2, (num), axis=0)
        classifier = Sequential()
        classifier.add(Dense(units = hl_units, kernel_initializer = init, input_dim = 300, activation =
act))
        classifier.add(Dropout(dropout))
        classifier.add(Dense(units = 1, kernel_initializer = init))
        classifier.compile(loss = 'mean_squared_error', optimizer = opt)
        classifier.fit(X_train, y_train, batch_size = batch_size, epochs = epochs)
        y_pred = classifier.predict(X_test)
        y_pred_all.append(y_pred[0][0])
        y_test_all.append(y_test)
```

```
    return (mean_squared_error(y_pred_all, y_test_all), mean_absolute_error(y_pred_all,
y_test_all), pearsonr(y_pred_all, y_test_all)[0])

#LOOCV function for 2 hidden layers
def cross_val2( hl_units, init, opt, batch_size, epochs, act, dropout):
    y_pred_all = []
    y_test_all = []
    for num in range(len(X2)):
        y_test = y2[num]
        X_test = X2[num, :]
        X_test = np.array([X_test])
        X_test.T
        X_train = np.delete(X2, (num), axis=0)
        y_train = np.delete(y2, (num), axis=0)
        classifier = Sequential()
        classifier.add(Dense(units = hl_units, kernel_initializer = init, input_dim = 300, activation =
act))
        classifier.add(Dropout(dropout))
        classifier.add(Dense(units = hl_units, kernel_initializer = init, input_dim = 300, activation =
act))
        classifier.add(Dropout(dropout))
        classifier.add(Dense(units = 1, kernel_initializer = init))
        classifier.compile(loss = 'mean_squared_error', optimizer = opt)
        classifier.fit(X_train, y_train, batch_size = batch_size, epochs = epochs)
        y_pred = classifier.predict(X_test)
        y_pred_all.append(y_pred[0][0])
        y_test_all.append(y_test)

    return (mean_squared_error(y_pred_all, y_test_all), mean_absolute_error(y_pred_all,
y_test_all), pearsonr(y_pred_all, y_test_all)[0])

#LOOCV function for 3 hidden layers
def cross_val3( hl_units, init, opt, batch_size, epochs, act, dropout):
    y_pred_all = []
    y_test_all = []
    for num in range(len(X2)):
        y_test = y2[num]
        X_test = X2[num, :]
        X_test = np.array([X_test])
        X_test.T
        X_train = np.delete(X2, (num), axis=0)
        y_train = np.delete(y2, (num), axis=0)
        classifier = Sequential()
        classifier.add(Dense(units = hl_units, kernel_initializer = init, input_dim = 300, activation =
act))
```

```
        classifier.add(Dropout(dropout))
        classifier.add(Dense(units = hl_units, kernel_initializer = init, input_dim = 300, activation =
act))
        classifier.add(Dropout(dropout))
        classifier.add(Dense(units = hl_units, kernel_initializer = init, input_dim = 300, activation =
act))
        classifier.add(Dropout(dropout))
        classifier.add(Dense(units = 1, kernel_initializer = init))
        classifier.compile(loss = 'mean_squared_error', optimizer = opt)
        classifier.fit(X_train, y_train, batch_size = batch_size, epochs = epochs)
        y_pred = classifier.predict(X_test)
        y_pred_all.append(y_pred[0][0])
        y_test_all.append(y_test)

    return (mean_squared_error(y_pred_all, y_test_all), mean_absolute_error(y_pred_all,
y_test_all), pearsonr(y_pred_all, y_test_all)[0])


#List of possible values
nodes_per_hidden_layer = [10,25,50,100,150,200,250,300]
dropout_rate = [0,0.05,0.1,0.2,0.5]

scores = []

#Performing Grid Search
for nodes in nodes_per_hidden_layer:
    for dr in dropout_rate:
            scores.append(cross_val1(nodes, ‘uniform’, ‘adagrad’, 2, 120, ‘selu’, dr))

for nodes in nodes_per_hidden_layer:
    for dr in dropout_rate:
            scores.append(cross_val2(nodes, ‘uniform’, ‘adagrad’, 2, 120, ‘selu’, dr))

for nodes in nodes_per_hidden_layer:
    for dr in dropout_rate:
            scores.append(cross_val3(nodes, ‘uniform’, ‘adagrad’, 2, 120, ‘selu’, dr))
```