Supplementary Materials:

Details of compared deep learning architecture:

The embedding deep learning architecture is compared with following models. The same training and testing dataset was used for comparison. One hot encoding features were generated for training and test data and fed to the input layer. Thus following are the one hot encoding architecture we compared with. All models are implemented by Keras library with the Tensorflow backend.

Model 1 : Long Short Term Memory(LSTM) architecture: The model contain four layers, input layer, LSTM layer, fully connected later and a output layer. The loss function used is binary cross entropy and optimizer is adam.

Input Layer: The protein sequences of window size 33 were one hot encoded with the feature space of 21(added '-' for no sequence) providing a shape of (33,21).

LSTM layer: It uses 32 neurons, and the input shape of (33,21) as (timesteps, features).

Fully connected layer: A fully connected layer acts as a hidden layer with 16 parameters.

Output layer: It uses softmax activation an the output as the number of labels. In our case, 2 either succinylated or not.

Model 2: RNN Architecture: The model contain of three layers with one hidden layer. The loss function used is mean squared error and optimizer as adam.

Input layer: The protein sequences of window size 33 were one hot encoded with the feature space of 21(added '-' for no sequence) providing a shape of (33,21).

RNN Layer: Simple RNN layer was used with 32 neurons, input shape as (timesteps, features) of (33,21) similar to the LSTM architecture. Activation function of "Relu" was used.

Fully connected layer: A hidden layer with 21 parameters was used.

Output layer: It uses two neurons as output to our model as succinylated or not, activation used is softmax.

Model 3: RNN-LSTM Architecture: The model contain of five layers. The loss function used is mean squared error and optimizer as adam.

Input layer: The protein sequences of window size 33 were one hot encoded with the feature space of 21(added '-' for no sequence) providing a shape of (33,21).

LSTM layer: It has 4 neurons, and the input shape of (33,21) as (timesteps, features) with stateful kept as False.

Fully connected layer: A hidden layer with 21 parameters was used.

RNN layer: A Simple RNN layer with 32 neurons and activation function is ReLU.

Output layer: It uses two neurons as output to our model as succinylated or not, activation used is softmax.

Model 4: Random Forest

To compare deep learning architecture with traditional machine learning algorithm, we compared the results with a Random Forest. Scikit-learn was used to implement random forest, feature reduction was done using the best features from the random forest classifier.

Results:

Table: Independent Test Results

Model	MCC/25	Sensitivity	Specificity	AUC
LSTM	0.36/0.34	0.74/0.69	0.66/0.64	0.68/0.67
LSTM-RNN	0.24/0.23(37)	0.63/0.71(37)	0.61/0.51(37)	0.66/0.61(37)
RNN	0.20	0.70	0.49	0.59
RF	0.29	0.74	0.62	0.72

DeepSuccinylSite architecture using features:

We implemented additional 'Deep Learning architecture where the input is other 'physico-chemical' based features. Essentially, this implementation takes 'PAAC': Pseudo Amino acid Composition, AAP 'k-Spaced Amino Acid Pairs', Autocorrelation features like Moreau-Broto autocorrelation, Composition, Transition and distribution features, and Entropy Features: Shannon entropy, Relative entropy, and Information gain.

We excluded any sequences with '-', while calculating the features. We then used XGBoost to extract prominent features which provided better accuracy and obtained a total of 160 features at threshold 0.00145.

Following are the results:

MCC = 0.27

AUC = 0.69

Specificity = 0.44

Sensitivity = 0.8

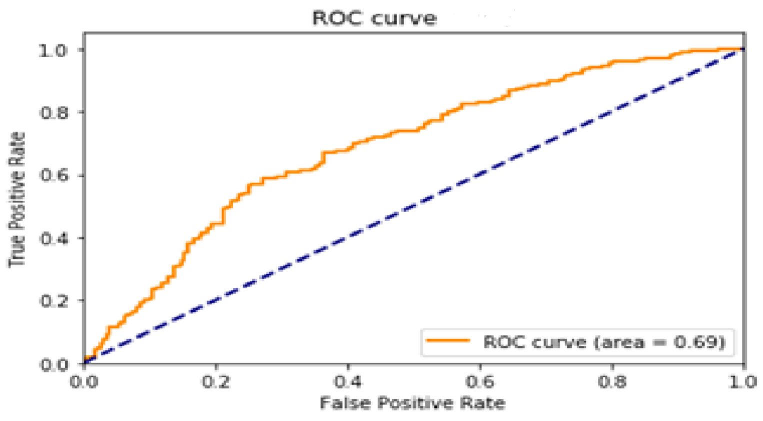

Figure 1: ROC curve for feature based DL-model