AI in Drug Discovery and Development

Prof. Rajnish Kumar
Dept. of Pharmaceutical Engineering and Technology
IIT-(BHU), Varanasi

Week-03
Lecture-12

Welcome to the course "AI in Drug Discovery and Development." Earlier, we saw machine learning models. And in today's session, we will have an overview of neural networks. So, by the end of this lecture, you will be able to understand the basic structure and functioning of neural networks, including neurons, layers, weights, and activation functions. Differentiate between key types of neural networks, such as feedforward, CNN, or RNN, and their relevance to drug discovery tasks; recognize the training process of neural networks, including backpropagation and optimization techniques. As well as identifying appropriate neural network architecture for tasks such as QSAR modeling, molecule generation, and protein structure prediction.

Appreciate the advantages of neural networks in handling biological and chemical data. So, let us see what those are, you know, neural networks, and why they are becoming famous and important for accelerating drug discovery and development. So, humans have always been trying to, you know, replicate the functions of the brain, and you know that the brain is made up of these neurons, actually. So, basically, a neural network is a computational model inspired by the way biological neural networks in the human brain process information.

So, it consists of interconnected layers of nodes or neurons that transform input data into output through weighted connections. So we look at the components of a neuron. So, a neuron contains the cell body, which is also known as the soma, that processes the signals from the dendrites, and then it has the, you know, the dendrites, which receive the incoming signals. And then it has the axons which carry the output signal, and then it has the synapse which acts as a junction transmitting signals between one neuron and another. And then the signal transmission occurs where the signal sums up at the soma if the total exceeds a threshold, and an action potential fires, and this binary behavior inspires the perceptron model.

So, humans have always been trying to replicate this, you know, this mechanism, and finally, they came up with this perceptron model where they could do something like how the brain works. If we talk about the biological neural network and its difference from, you know, the artificial neural network. So, in the biological neural network, the neurons are the brain cells, and in the artificial neural network, we have mathematical functions, which we call nodes or units. And then the synapse is, you know, the connection between the

neurons where the chemical signal is transmitted in the case of the brain, while the synapse in the case of an artificial neural network is the weighted connection between the nodes. Like how much of the signal will be transferred to the next neuron and that weight decides the amount of that signal, actually.

And then the learning process in the case of a biological neural network is synaptic plasticity and experience-driven adaptation. However, we use algorithms like backpropagation to update the weights, which helps the artificial neural network learn the task. And then, signal transmission in this case of a biological neural network happens through electrochemical signals. However, in the ANN, we have those activation signals in the form of numeric values. And for the purpose, biological neural networks are necessary for sensory processing, decision-making in motor control, etc.

However, these artificial neural networks are helping us; nowadays, you can see a lot of applications where they can help us recognize patterns. They can help us with the classification of, you know, things, and then they can help us with the prediction and all sorts of, you know, fancy things as well. So, the perceptron is the simplest type of neural network; actually, it is a simple type of neural network designed for binary classification, mapping inputs to outputs like 0 or 1. So, how it works is that it takes the input features and multiplies them by weights, and then it computes a weighted sum and applies an activation function. So, usually in the earlier perceptron, the activation function used is a step function where the output is either 0 or 1.

So, if the sum exceeds the threshold, it outputs 1; otherwise, the output will be 0. And then it was inspired by McCulloch's neural model, which was given in 1940s. And it demonstrated how machines could learn linear separable patterns. And it also paved the way for multilayer perceptrons and more complex neural networks. So, talking about the key component of an artificial neuron, here you can see that we have, for example, the inputs x1 to xn.

And then you have the weights, and then we calculate the weighted sum, okay. And then we have this activation function. So, in the early perceptron model, this activation function was a step function; there are multiple activation functions that can be used, and then finally, we actually get the output. So, we have the weights which determine the strength and importance of each input; if our weight is higher, then it means that factor has a greater influence on the output. And then we use the bias, which is a constant value that we add to the weighted sum of the inputs.

So, it helps shift the activation function and improve the model flexibility. And then the activation function, which we will talk about later as well, introduces non-linearity into the

model. Because those earlier perceptron models were not good at, you know, separating non-linear data. So, they were good at separating the linear data where the task is actually very simple. And the activation function, it allows the network to learn complex pattern.

The common functions that are being used are sigmoid, which squashes output between 0 and 1, and then you have the ReLU function, rectified linear unit, which outputs 0 or a positive value. So, that means we have the tanh output function, which gives us a value between minus 1 and 1. Therefore, the median is 0. So, we compare the perceptron and the modern neural network neuron. So, as we talk about the inputs.

So, in the perceptron, the inputs are multiple features, such as molecular weight or logP. And then in the modern neural network, they can have multiple inputs because they have multiple layers as well. So they can have inputs from the previous layer's neurons as well. And then, in the case of weights, each input has a weight to control its influence in the case of a basic neuron, along with the weights, bias, and summation. So, these are similar in both cases in basic neuron and advanced or modern neural network neuron.

However, the main difference is the activation function, where in the basic neuron we were using the step function, which will output either 0 or 1. However, in the modern neural network, we can obtain a kind of non-linear decision as well because we are using different kinds of activation functions. The learning capability of the earlier perceptrons is very basic, so they can only solve linearly separable problems. However, the modern neural network neurons can take on a lot of tasks, and they can help us deal with all sorts of problems, actually. Talking about the uses, perceptrons are foundational models; they are the foundation for early single-layer networks.

However, those modern neural networks are used in multilayer architectures like CNNs, GNNs, and Transformers that we will see. Okay, so what is the importance of, you know, the activation function? So this is kind of an example of data where we have soluble compounds represented by the red dots and insoluble compounds represented by the green dots. And then we have the features, feature 1 and feature 2, which are responsible for or related to their, you know, solubility, and the data looks like this. So, in the case of using a perceptron, a basic perceptron, or if you use a step-up function, it will not be able to classify this data into soluble and insoluble because those linear activation functions only produce linear decisions. If we wanted to classify this or solve this problem.

So, then we have to induce nonlinearity and which allows us to approximate the complex function and this is possible with the help of these advanced functions. So, you can see here that these nonlinear functions can segregate the soluble and insoluble data into two classes, actually. So, as I said, there are multiple activation functions. So, let's talk about

their characteristics and uses. So, the ReLU function's output is from 0 to infinity, and the key characteristics of the ReLU function are its simple and efficient computation.

and it mitigates the vanishing gradient problem and it can lead to you know dying ReLU when neurons become inactive for negative inputs. So, these are commonly used as hidden layers in deep neural networks. And then we have the sigmoid function, where the output is 0 or 1. So, the key characteristics of a sigmoid function are that it is smooth, has a smooth gradient, and is useful for probabilistic interpretations. However, it is prone to vanishing gradients, especially for large input magnitudes.

And the outputs are more zero-centered, which can also actually affect the convergence. So, it is largely used as an output layer for binary classification tasks. And then we have the Tanh, where the output is in the range of minus 1 and 1. So, it is having zero centered output which can aid in optimization. However, it is still susceptible to the vanishing gradient for large input values and generally performs better than the sigmoid in the hidden layer due to the center output, actually.

So, it is commonly used as a hidden layer where zero-centered data is beneficial. And then with the Softmax function, it can have an output range from 0 to 1 or sum to 1 as well. So, it converts raw scores into probability distributions and ensures the output probability sums to 1, and it is sensitive to large input values, which can lead to numerical instability often mitigated by the input normalization. It is commonly used as an output layer for multiclass classification problems. Okay, when it comes to the structure of a neural network, basically, as I said, these neural networks are made up of neurons, just like the human brain is made up of neurons, and those are interconnected with each other.

Likewise, we have, you know, the basic structure of a neural network: an input layer, a hidden layer, and an output layer. So the input layer takes the raw data; in the case of drug discovery, it takes the features of the molecules or the features of the proteins, etc. And then these hidden layers, they can actually be multiple hidden layers; for example, here I have shown two hidden layers: hidden layer one and hidden layer two. These hidden layers process the data, identify patterns, and transform input into meaningful features. And then you have the output layer, which provides the final prediction, such as drug efficacy, interaction score, or the side effect probability.

So, how is information flowing? So, we have these weighted connections which are represented by W. So every connection between nodes has a weight that determines its importance. Like we have this feature, for example, here, and then this is a weighted connection to this hidden layer 1. So this W will decide how much weight, how much importance this feature has actually. And then we have the activation function, which

introduces non-linearity, which we talked about in the earlier slide, helping the network capture complex relationships in the data.

And then usually these are forward propagations where the data flows from input to output through the layers, transforming at each step. So we have the input data that is going through all these network layers, and then finally we are getting the outcome. So now another thing is how the network is actually learning. So that is a very important thing to understand.

So it has to learn from the data. So the network adjusts itself by learning patterns from large data sets, such as drug properties and biological responses. Okay, so you can see here that we have this data set, and we divide this data set into the training data, validation data, and testing data. Once we have this data, what we do is input it into the neural network, and now this neural network is learning the features of the data. And then we have another thing here, which is the loss function. So, which measures the difference between the predicted output and the actual result, and the goal is to minimize this difference.

So, for example, we have a set of molecules where we are trying to, you know, train a model for predicting the solubility. And then we have a LogP, the number of rotatable bonds, and we have, you know, how to say aromaticity index, actually aromaticity ratio. So, these are the features that we are trying to correlate or see whether they are responsible for the solubility. So, what this model will do is. So, we have the training data, and then we have the features associated with it.

So, now each of these features, like this is a LogP, this is the aromaticity index, and this is the number of rotatable bonds. Likewise, we have features, and each of the features, based on their importance, will have some weights, right? So, in this case, we know that LogP is a major determinant of solubility. So, the LogP will have a major, you know, weight. So, after you learn which of the features are important. So, we will get an output from this model that predicts the solubility of those molecules from the training data.

Now, for the training data, we already have the experimental solubility. So, we can compare this experimental solubility with the predicted solubility, and that is our, you know, loss, actually, and that we discussed in the machine learning models as well. So, now we have the loss, and the idea is to reduce that loss, which is where we use backpropagation or optimization, actually. So, the backpropagation is a key process in training where the network calculates the error at the output and propagates the error backward through the network and just waits to improve the predictions. And then we have the optimization algorithms; we use techniques like stochastic gradient descent and the Adam optimizer, which help the network to minimize the cost function efficiently.

So, we have the loss here, determined by the loss function, and then we optimize the network. So, we adjust those weights to reduce the difference between the actual experimental value and the predicted value, or to reduce the loss, actually. So, this is the whole idea of training a neural network. So, then there are multiple kind of loss functions.

So, we will talk about MSE and cross-entropy. So, a loss function measures the difference between the predicted output and the actual target. It guides the model during training by telling it how wrong its predictions are. So, we have this MSE, mean square error. So, which is largely used in regression problems. So, it measures the average squared difference between the predicted values and the actual values.

So, this is how we calculate MSE; we are determining the average squared difference between the predicted and the actual values. However, it is sensitive to outliers, and a smaller error means the model is actually better, so its predictions are better. And then we have another type of loss function, which is cross-entropy loss. So, it is largely used in classification problems. So, either we can use it in binary classification with a binary cross-entropy loss function, or we can use it in multiclass classification with a categorical cross-entropy loss function.

And then what it does is measure the distance between the true labels and the predicted probability distributions. And it heavily penalizes confident but wrong predictions. Okay, then once we get the loss function, we calculate the loss; the next step is to, you know, do the backpropagation, which means going back to the model and adjusting the weights so that we can reduce that loss. So, the goal of backpropagation is to efficiently update weights by propagating the error backward. And then we have the steps: the forward pass, which calculates the output; loss calculation; and comparison of the prediction with the true output.

Backward pass, which uses the chain rule to compute gradients layer by layer, and weight update, where we apply gradient descent to adjust the weights. And why it works is that it distributes the error to each weight, guiding the update toward reducing the overall loss. However, a challenge is that vanishing gradients, especially with deep neural networks, are an issue. However, the ReLU and batch normalization help mitigate this. And these are some of the optimizers that are used for adjusting the weights.

So we have this stochastic gradient descent, which updates weights using the gradient of the loss with respect to each parameter. So the advantage is that it is simple and memory-efficient. However, it has slow convergence and is sensitive to the learning rate. And then we have the RMS probe, which is root mean square propagation, that maintains a moving

average of squared gradients to normalize updates. The advantage is that it works well for non-stationary objectives.

And then the disadvantage is that hyperparameter tuning is needed to make it efficient, actually. And then we have the Adam Adaptive Movement Estimation. So, it combines momentum and RMS probe; it tracks both the first and the second movement of the gradients. So, the advantages are that it has fast convergence, is widely used, and needs less tuning.

However, it may generalize poorly in some cases. Okay, and then we talk about some training hyperparameters, like epochs, batch size, and learning rate. And all of this is done during the optimization, you know, actually. So an epoch is one complete pass through the entire training data set. So the whole cycle is called one epoch. And then more epochs mean better learning up to a point because if we are running so many epochs, we can also risk overfitting because that model can get overfit, and then it will, you know, look very good.

However, in reality, it is not. So, it tunes based on training and validation performance. And then, the batch size is the number of samples processed before model weights are updated. So, if you are using a small batch, we will get the kind of noisy updates. However, better generalization occurs when you are using large batches, making it faster and having smooth convergence.

However, it needs more memory. And then the learning rate, which is represented by eta. So, it controls how much weight is updated during the training. So if we are using too high of a learning rate, it may overshoot or diverge. If we are using too low a value, then there will be slow convergence or it can get stuck in the local minima. Okay, after this kind of introduction to what neural nets are and how they work.

So let us talk about some types of neural network architectures. So there are kind of multiple types, and we cannot cover all of them because this is just some kind of introductory lecture where I'm trying to give you an idea of what neural networks are and how they work. So some of the common neural networks that are being extensively used in drug discovery and development include feedforward neural networks, where the data flows in one direction from input to output, and they are great for basic classification and regression tasks. Then we have the convolutional neural networks, CNNs. These are specialized for image data. They use these convolutions to detect spatial patterns, such as edges or textures.

And then we have recurrent neural networks. These are designed for sequential data. They can remember the previous inputs like time series data and text data. And then we have

transformers. These are powerful architectures for sequence data. They have no recurrence, but they rely on the self-attention mechanism.

An example is all those, you know, kind of chat GPT and all. And then you have the graph neural networks, which work on graph-structured data. These are ideal for molecular graphs, social networks, or relational data. And especially, you know, finding the drug targets where we can make a graph of the interactions between the proteins, drugs, and nucleic acids, and all. Talking about the feedforward neural network, this is the simplest kind of neural network where the data will have the input layer, hidden layer, and output layer. And where the data is flowing in one direction only, like from the input layer, it is passed on to the hidden layer, and from the hidden layer, it is coming to the output layer, and we actually get the outcome.

So, the input layer encodes molecular descriptors, or it can be molecular features, fingerprints, and all those sorts of data like molecular weight and logP. And then the hidden layer processes the data through weights and activations. And the output layers, as I said, produce the predictions, such as whether the compound is soluble or insoluble, or toxic or non-toxic. The working principle, as I said, is that data is a one-directional input; data comes from the input layer, is passed on to the hidden layer, processed, and then we get the output. So, where they can be used is that they are being extensively used in ADMET prediction, such as the prediction of absorption, distribution, metabolism, excretion, and toxicity profiling.
as well as the hit identification, where they can predict the biological activity of new chemical entities.

So, they can basically be used for predictive modeling where we can predict the binding affinity, predict the solubility, predict the metabolic stability, or ADMET properties, all those sorts of things. And then we have convolutional neural networks, so what happens is, for example, here we have, in the case of a feedforward network, we have, like, for example, 10 features, or in this case, three features, and those three features can be easily processed. However, if we need to work with an image, actually, like we have an image here, then this image is made up of pixels. Actually, if this image is 224 by 224 pixels, then there will be 50,176 pixels, and that is only for a unicolor, like just a black and white picture; if it is in color, then we have to multiply it by three.

So, we will actually have that many inputs. So, handling so many input neurons will be really, you know, computationally cumbersome, especially for dealing with images. So, that is why we developed this convolutional neural network technique where instead of using the individual features, we are just taking those spatial features. Okay, and then we are trying to identify the main features, and then we are working with them. So let us see what it is. So it has an input layer that encodes spatial or structural data, like molecular

graphs, grids, 2D molecular images, or 3D voxelized structures, and then we have the convolution layer.

So we have this convolution layer, and what it is doing is applying filters to extract features like edges, shapes, or substructures in the molecules. And then we have the pooling layer, which reduces dimensionality while preserving the key features. And then we have the fully connected layer, which integrates extracted features for final decision-making. And then we have the output layer, which generates predictions.

For example, it can be binding affinity, bioactivity, or solubility. And that is how they are, you know, very good at processing the images and all those applications which you see, where they can be used for diagnosing diseases based on the mammogram pictures, MRI scans, or PET scans; these CNNs are being used here. So the working principle is that the input data is processed through convolutions and pooling, followed by the dense layer, and each convolutional neural network computes. And then talking about the application in drug discovery, we have the ADMET, which we can use for ADMET prediction, where it can predict pharmacokinetic and toxicity profiles from molecular structures. And then we can use it for hit identification as well, where it can classify its compounds based on their predicted biological activity. And we can use it for image-based screening, where it can analyze microscopy images in phenotypic screening campaigns as well.

Okay, then another type of network is, you know, recurrent neural networks, and these are a class of neural networks designed for sequential data where the order of information matters. Unlike the traditional feedforward network, RNNs maintain a memory of previous inputs, making them ideal for time series and sequence-based tasks. So, they have the you know the hidden state which maintains memory of the previous input. So, you can see here that the output from this neuron can be used as an input as well. So, it is keeping a memory, and then that is, you know, it can remember the sequence as well.

So we have the neurons with a loop where the feed output from one time step goes into the next, and then we also use the technique called backpropagation through time. It is specialized for the backpropagation used to train the RNN on the sequential data. Okay, so we process the sequence like either protein chains or SMILES strings one step at a time, learn the features from that, and make decisions on the basis of those. So how can they be used in drug discovery? So they can be used for SMILE string generation, where they can generate new molecules from chemical sequence data. Time-dependent drug response modeling can capture how cells respond to drugs over time.

And then they can be used for, you know, peptide drug design as well, where they can model the amino acid sequence for the peptide-based drugs. And they are very popular, or

they are very frequently used in generative modeling, actually de novo drug design. Okay, and then coming to the transformers. So we talked about this paper, "Attention Is What All You Need," from Google, actually. So, the transformers are a class of neural networks designed to handle sequential data without relying on recurrence.

So, instead of processing data step by step like RNN, transformers use a mechanism called attention to capture relationship between all parts of the sequence simultaneously. And they are especially powerful for modeling long-range dependencies and parallelizing computation. So, what they have is that the key components are a self-attention mechanism, which allows the model to focus on relevant parts of the input sequence regardless of the distance. And then we have the positional encoding, which injects information about the position of tokens since transformers lack inherent sequence order. And then it has multi-head attention as well, which improves the model's ability to capture different types of relationships in parallel.

So, looking at the working principles, it processes the entire sequence at once, attending to different parts as needed, and it learns the relationship between all elements in the sequence through the attention scores. Some of the applications of these transformers in drug discovery are that we can use them for molecular property prediction, where models represent molecules as graphs or SMILES strings to predict properties like solubility or activity. We can use it for de novo molecule generation, which can design novel molecules by learning chemical rules from large data sets. And we can do protein structure prediction, which can predict 3D structures from amino acid sequences, like you know, AlphaFold. And then we can do the drug-target interaction modeling where it can learn complex interactions between the drugs and biological targets.

And then another important kind of neural network is the graph neural network. And that is also quite popular, especially in the de novo drug design and generative modeling. So, where the molecules are represented, as you know, especially in chemistry and drug discovery, the molecules, or even the protein molecules or small molecules. Those are actually represented as graphs, where the nodes represent atoms or proteins and the edges represent bonds or interactions between the nodes. And then it uses a message passing layer which propagates information between nodes, updating their embeddings. So the genome treats molecules as graphs, and then each node updates its representation by aggregating information from the neighboring nodes.

And then, final node embeddings are pooled for molecular-level prediction, and we can use them again for molecular property prediction, such as predicting solubility, permeability, and toxicity directly from the molecular graphs. We can use it for drug target interaction prediction, where we can model the protein-ligand interaction as graphs. And

we can use it for protein-protein interaction networks as well, where it can help us identify neutral targets by modeling the biological networks. One of the examples is antimicrobial discovery, specifically the discovery of halicin, where we could predict the antibacterial properties from chemical structure graphs, which, as I said, led to the discovery of halicin. Okay, so now we have seen what those popular neural networks are, at least the ones that are being used in drug discovery and development.

So, why are neural networks the right choice? So, one of the important things they can do is pattern recognition. They excel at identifying hidden patterns in high-dimensional data crucial for understanding drug-target interactions, side effects, and biological pathways. Like in the case of target discovery, we have data coming from multiple sources, including multi-omics, proteomics, metabolomics, epigenomics, and transcriptomics. So getting a relationship or identifying a pattern among that huge amount of data, like diverse data, is a big task, and these neural networks can actually help us there. So, they can handle complex and diverse data because, as I said, drug discovery involves multiple data types: chemical structures, protein sequences, biological responses, and clinical data.

So, they can efficiently integrate and process multimodal data as well. Scalability is also good; neural networks scale with large data sets, which are common in pharmaceutical research, and more data improve model performance. And talking about the predictive power, by learning intricate relationships, these can predict the efficacy, toxicity, interactions, and dosing recommendations faster and more accurately than traditional methods. And then automation and efficiency are another important things, as it reduces manual efforts by automating feature extraction and the decision-making process and speeding up candidate identification and testing. Okay, coming to the summary, the neural networks are the foundation of modern deep learning and are inspired by the structure of the human brain. So they consist of interconnected neurons organized into layers that learn patterns from data through iterative optimization, and covalent architecture includes feedforward neural networks, convolutional neural networks, and recurrent neural networks, each suited to different types of data and tasks.

Activation function, loss function, and optimizers play a key role in training neural networks efficiently. And neural networks have shown remarkable success in drug discovery, powering applications such as virtual screening, molecular generation, drug-target interaction prediction, and image analysis. While powerful, they require a large, well-curated data set and often act as black boxes, highlighting the need for interpretability and proper validation. And I have some suggestions for you.

If you want to learn more about this, you can go through these references. And then I have

an open question for you. Just think about why CNNs work better for images and GNNs for molecules. And with that, thank you.