Deep Synergy Feed Forward model

SujayKumar Reddy M

AstraZeneca R&D IT Intern

May 2024

Abstract

This document presents an overview of the Deep Synergy model, a feed-forward neural network included in the ChemicalX package. This neural network demonstrates exceptional performance, effectively handling most datasets available in the package, achieving rapid convergence, and exceeding 85% accuracy within 20 epochs. Despite these impressive capabilities, this document argues that the Deep Synergy model is not yet suitable for clinical deployments. Originally introduced in 2017 by Kristina et al., the model, while powerful, lacks the necessary intuitive approach required for clinical applications, whether for classification or risk score prediction. The Deep Synergy model is compared and analyzed with all the datasets provided in the chemicalx with all 3 tasks interaction, synergy prediction and Polypharmacy task.

1 Data Problem

The dataset used for training the Deep Synergy model consists of three major feature sets: drug-left features, drug-right features, and context features. These features are concatenated and fed into the feed-forward model (i.e., Deep Synergy). The data is highly sparse, sourced primarily from the DrugComb and DrugBankDDI datasets. Specifically, the DrugComb dataset comprises 527,466 rows and 800 columns, while the DrugBankDDI dataset consists of 306,892 rows and 598 columns. More dataset statistics are available in the Table 1 below.

2 Deep Synergy

While drug combination therapies are a well-established approach in cancer treatment, identifying novel synergistic combinations remains a significant challenge due to the vast combinatorial space. The task of synergy prediction or drug-drug interaction (DDI) assessment is crucial because traditional methods often detect adverse drug effects only after the drugs have been released, posing significant risks—particularly for elderly individuals who frequently take multiple medications. A risk score can be instrumental in identifying critical drug interactions, thereby providing a preemptive alert before the drugs are released.

The left side of the code refactoring below presents the original package code, which includes a Dropout layer, two hidden layers, one input layer, and one output layer. It is important to note that the Vanilla Deep Synergy model on the left uses 32 units for all hidden layers. Conversely, the right side of the refactored code gradually decreases the number of hidden units to transform the sparse data matrix into relevant features for task classification.

Dataset name	**Total features**	**samples**
DrugBankDDI	598	306,892
DrugComb	800	527,466
DrugCombDB	624	153112
TwoSides	522	399665

Table 1: Data Statistics for Deep Synergy model

Original Code (chemicalx)

```

 def forward(
         self,
         context_features: torch.FloatTensor,
         drug_features_left: torch.FloatTensor,
         drug_features_right: torch.FloatTensor,
     ) -> torch.FloatTensor:

         hidden = torch.cat([context_features, drug_features_left, drug_features_right], dim=1)
         self.final = nn.Sequential(
             nn.Linear(drug_channels + drug_channels + context_channels, input_hidden_channels),
             nn.ReLU(),
             nn.Linear(input_hidden_channels, middle_hidden_channels),
             nn.ReLU(),
             nn.Linear(middle_hidden_channels, final_hidden_channels),
             nn.ReLU(),
             nn.Dropout(dropout_rate),
             nn.Linear(final_hidden_channels, out_channels),
             nn.Sigmoid(),
         )
         return self.final(hidden)
```

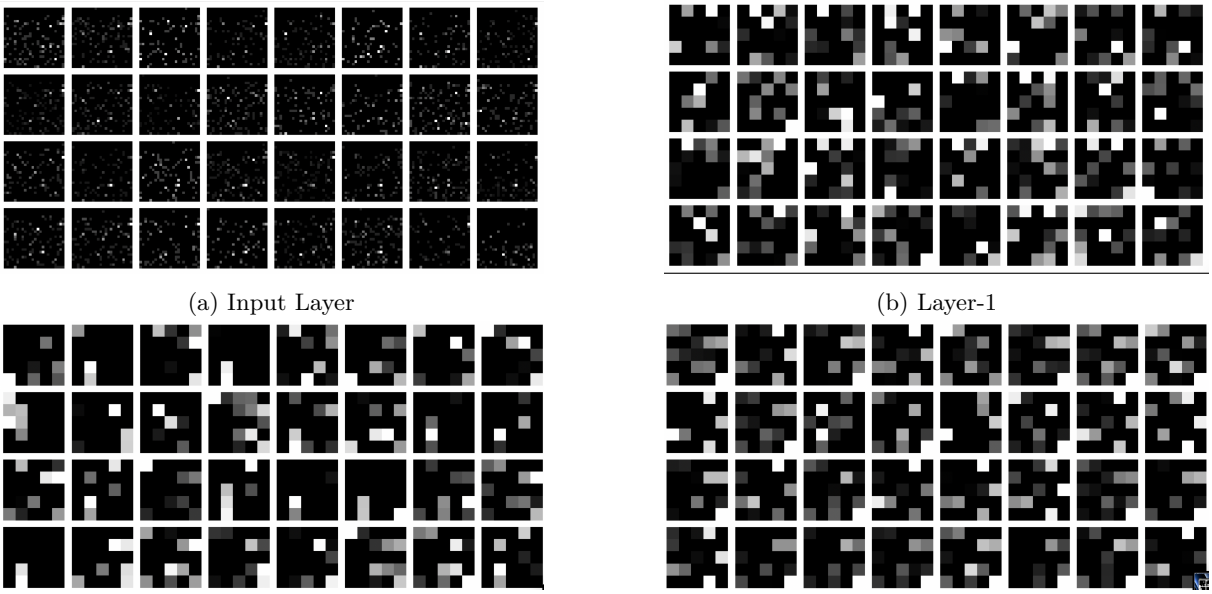
After Code Refractoring

```


 def forward(
     self,
     labels,
     context_features: torch.FloatTensor,
     drug_features_left: torch.FloatTensor,
     drug_features_right: torch.FloatTensor,
     plot = False
 ) -> torch.FloatTensor:

     hidden = torch.cat([context_features, drug_features_left, drug_features_right], dim=1)

     if plot == True:
         plot_cat(hidden,labels=labels)


     x = self.lin1(hidden)
     x = self.relu(x)
     # print(x.shape)

     if plot == True:
         plot_layers(x,labels)

     x = self.lin2(x)
     x = self.relu(x)

     if plot == True:
         plot_layers(x,labels)


     x = self.lin3(x)
     x = self.relu(x)
     x = self.dropout(x)
     if plot == True:
         plot_layers(x,labels)

     # Need to add Dropout Layer for equating the original model

     x = self.lin4(x)
     x = self.sigmoid(x)

     return x
```

3 Model Perspective

In this section, we examine the model's interaction with the data. Initially, we train the model for 50 epochs until it achieves an accuracy exceeding 95%. The model's weights are then saved, and a batch of data is passed through the model to visualize the data after each layer. The visualization algorithm plots the data grids, allowing us to compare the vanilla model with the updated hyperparameter model.In the vanilla Deep Synergy model, each layer consists of 32 units, resulting in unchanged plot shapes across layers. In contrast, the optimized Deep Synergy model transforms the sparse dataset into a dense format by gradually reducing the dimensionality by a factor of two, whereas the vanilla model maintains 32 units across all hidden layers.

```
import numpy as np
import matplotlib
matplotlib.use("TkAgg")

```



(a) Input Layer (b) Layer-1

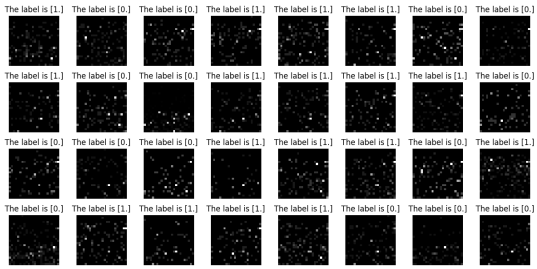
(c) Layer-2 (d) Layer-3

Figure 1: Vannila Deep Synergy model on DrugBank DDI dataset

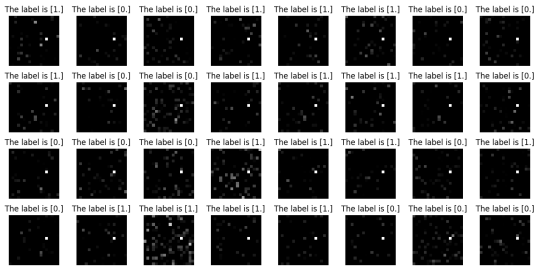
```
import matplotlib.pyplot as plt



def reshape(x):
    images = x.shape[0]
    numq = np.sqrt(x.shape[1])
    image_dim = (int(numq),int(numq))

    new_images = []
    for i,image in enumerate(x):
        image = image[:int(numq)*int(numq),]
        new_images.append(image.reshape(image_dim[0],image_dim[1]))
    new_images = np.array(new_images)
    return new_images

def plot_cat(x,labels):
    new_images = reshape(x)
    fig, axes = plt.subplots(4, 8, figsize=(12, 6))
    for i in range(32):
        ax = axes[i // 8, i % 8]
        ax.imshow(new_images[i], cmap='gray')
        ax.set_title(f"The label is {labels[i].numpy()}")
        ax.axis('off')

    plt.tight_layout()
    plt.show()



def plot_layers(x,labels):
    plot_cat(x.detach().numpy(),labels.detach())
```

4 Explainable AI

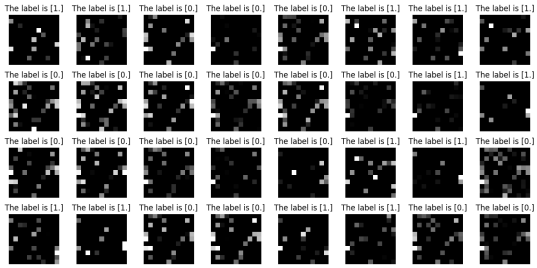
Using black box models like neural networks often yields superior results compared to traditional machine learning techniques due to their capabilities in dynamic feature extraction and improved performance. However, these models are frequently criticized for their lack of interpretability, as it is challenging to understand the rationale behind individual predictions.In this section, we will explore local explanations using the LIME (Local Interpretable Model-agnostic Explanations) package and global explanations using



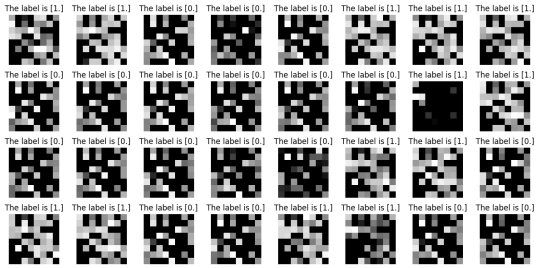
(a) Input Layer



(b) Layer-1



(c) Layer-2



(d) Layer-3

Figure 2: Optimized Deep Synergy model on DrugBank DDI dataset

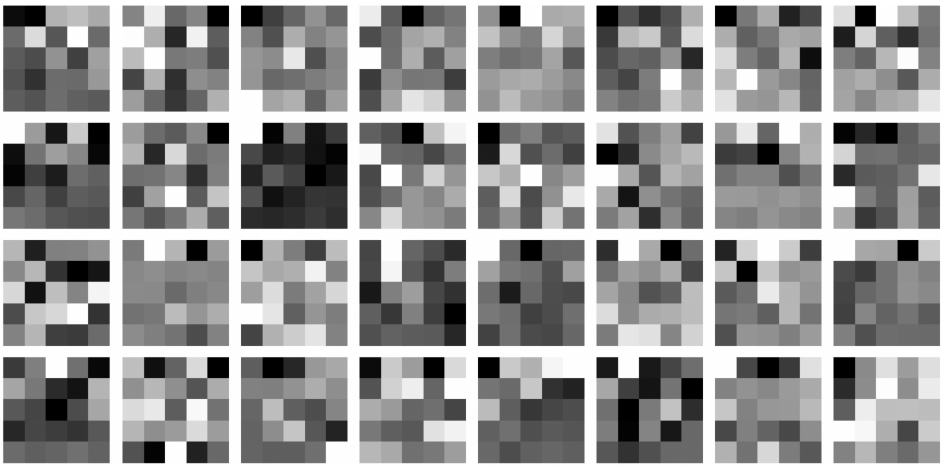
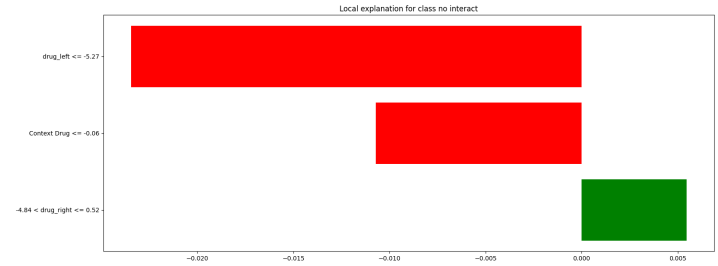


Figure 3: Data After PCA on whole combined feature data

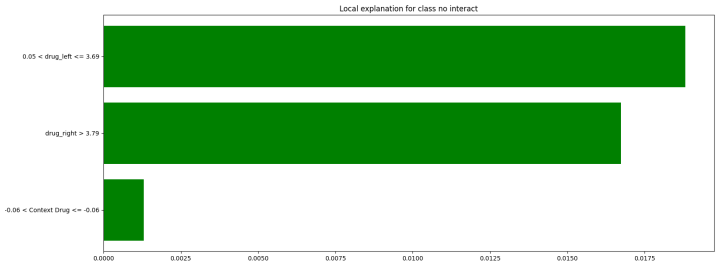
Shapley values. Given the high dimensionality and sparsity of the data, the recommended approach for handling sparse matrices is to apply matrix factorization techniques such as Principal Component Analysis (PCA) or Singular Value Decomposition (SVD). These methods transform sparse matrices into dense matrices with relevant features.The image below represents the data after applying PCA to the entire dataset. However, our objective is to identify the features that contribute most significantly to local explanations. Applying matrix factorization methods to the entire dataset makes it difficult to qualitatively interpret the features. To address this, we separately apply PCA to context features, drug-left features, and drug-right features, resulting in three highly dense feature sets. We then use a statistical machine learning algorithm, such as Random Forest, to derive local explanations.

The Explanations provided in the Figure 4 are local explanations for a specific data sample, the High Interaction example provides that drug right features provides higher weight for a particular prediction which has higher interaction, In Figure 4.b, in the no interaction sample. This Drug left features provides higher weight for that particular sample. This type of local explanations are valid only for a particular sample which changes from sample to sample. Shapely values averages all out to provide global explanations over the test data. We can see in the Figure 5 that the context drug features provides less weightage for a model to predict the drug interaction happens or not. We tried to take the context features and then train the same deep Synergy model, which also acheives 98% with a fair convergence rate. This tells

us we don't need context features for Drug Interaction or Synergy Prediction.



(a) High Interaction



(b) No Interaction

Figure 4: High Interaction and No Interaction of drugs (local Explanations)

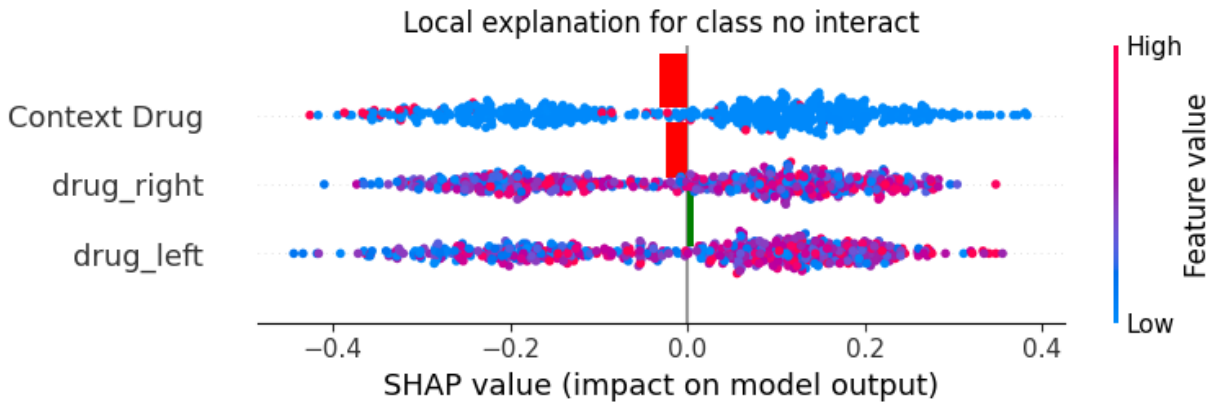


Figure 5: Shapely Values for Global Explanations