Appendix

PopPhy-CNN: A Phylogenetic Tree Embedded Architecture for Convolutional Neural Networks to Predict Host Phenotype from Metagenomic Data

Algorithms for Tree Population and Tree-Matrix Representation

In PopPhy-CNN, a phylogenetic tree is used as a template to construct a populated tree for each sample in the dataset. In order to populate a tree, the value of each OTU from a sample is assigned to it's respective node in the tree. The tree is then populated such that an abundance value for each internal node is equal to the sum of its children's abundance values. The method is outlined in Algorithm S1, which was presented in [22]. Once the tree has been annotated with abundance values, it is transformed into a matrix format by placing the root's abundance in the top left corner of a matrix. Then for a given row, the children of the nodes from that row are selected and their abundances are placed in the subsequent row in the order that their parents appear, starting with the left most column. The rest of the matrix is filled with zeros. The method is outlined in Algorithm S2.

Data: A phylogenetic tree $G = \{V,E\}$ and taxa abundance vector $\mathbf{x}$
Result: A populated phylogenetic tree $G = \{V,E\}$

for *l from the maximum tree depth to 0* **do**
  for *each node v in layer l* **do**
    if *the label of v is an OTU in vector* $\mathbf{x}$ **then**
      assign node v the abundance of the OTU from vector $\mathbf{x}$
    end
    if *v has any children* **then**
      add its children's abundances to the abundance of v
    end
  end
end

Algorithm S1: Tree Population

Data: A populated phylogenetic tree $G = \{V,E\}$
Result: A matrix $\mathbf{M}$ containing the tree representation

Construct a zero matrix $\mathbf{M}$ with a row for each tree layer and columns equal to the largest number of nodes in any layer;
$C \leftarrow$ Root Node of G;
for *j from 0 to the number of layers of G* **do**
  $i \leftarrow 0$;
  $Q \leftarrow \{\}$;
  for *each node v in C* **do**
    $M(i,j) \leftarrow$ abundance of node v;
    Push children of node v into queue Q;
    $i \leftarrow i + 1$;
  end
  $C \leftarrow Q$;
end
Return $\mathbf{M}$

Algorithm S2: Tree-Matrix Representation

PopPhy-CNN Architecture and Training

Standard CNNs are composed of multiple convolutional layers, which are usually followed by at least one fully connected layer. Each convolutional layer is composed of multiple kernels, each of which transforms an input matrix M into a set of feature maps of velocities through a convolutional operation. For a given kernel k with weights $W^{(k)}$ of size m x n and input X, the velocity of point (i,j) is calculated as:

$$vel^{(k)}(i,j) = \sum_{r=0}^{m}\sum_{s=0}^{n} M(i+r, j+s) * W^{(k)}(m-r, n-s) \tag{1}$$

The feature maps composed of these velocities are then passed through a non-linear activation function and subsampled through max or mean pooling to give a matrix of activations.

The CNN architectures used in this study consist of one convolutional layer with a rectangular kernel and one convolutional layers with a single 1x1 kernel. This is followed by two fully connected layers and a single output layer. Each layer hidden layer used the exponential linear unit activation function (ELU) and the softmax activation function was applied to the output layer for class prediction. Cross-entropy, defined as $-\sum_c p_c \ln(a_c)$, was used in our cost function to ensure the distribution of the output class is resemble to that of input class. Here p_c is the true probability that the sample belongs to class c and a_c is the predicted probability that the sample belongs to class c. When the output is categorical, the cross-entropy becomes identical to the negative log loss since all but one p_c will be 0, with p_c, where c is the true class, being equal to 1. This makes the loss of a single input as:

$$C = -\ln(a_c) \tag{2}$$

Where a_c is the predicted probability, i.e., softmax activation, of the sample belonging to class c when the true class of the observed sample is c.

To handle class imbalance, we applied a penalty weight that is determined by the total number of samples divided by the number of samples in a given class. This would scale the cost in a way such that samples of less frequent classes were scaled higher. We then regularized our model by adding $l1$ and $l2$ penalization to the weights in order to prevent large weight values. Our final model was trained using the cost function:

$$C = -(\frac{n_{total}}{n_c})\ln(a_c) + \lambda_1 \sum_L ||W_L||_2 + \lambda_2 \sum_L ||W_L||_1 \tag{3}$$

Where given an input whose true label is c, n_{total} is the total number of samples, n_c is the number of samples for class c, and λ are the regularization parameters to penalize the weights W for each layer L. For further regularization, we used dropout in our network over the fully connected layers in order to help prevent overfitting. The dropout method works by randomly selecting nodes within the hidden layers and temporarily removing them from the training, preventing both feed-forward information from that node as well as feedback information from back-propagation. This allows the network to train subnetworks to reach the desired output, creating multiple paths to predict the correct output.

Visualization of Multiclass Datasets

Principal Coordinate Analysis (PCoA) of IBD, Multi-Disease, and Syn9 datasets using the Bray-Curtis dissimilarity as the distance metric are shown in S1.

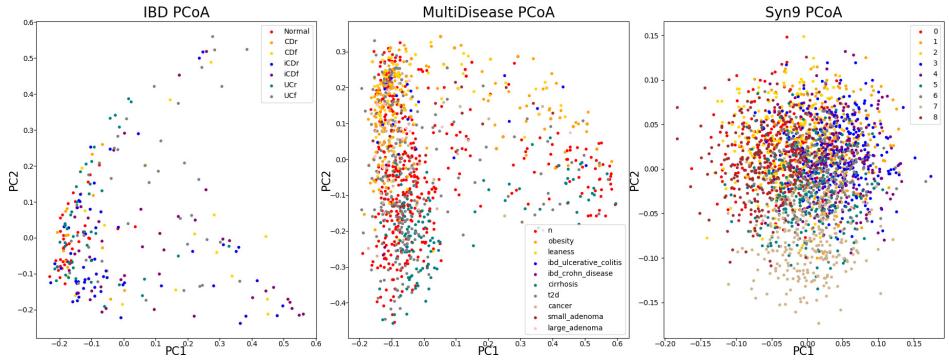

Fig S1. PCoA plots for the IBD, multiple disease, and synthetic 9 class datasets using Bray-Curtis dissimilarity.

Parameter Complexity of Network Models

We observed that PopPhy-CNN scaled 5.08 times faster than other the two other models. This is due to the fact that the input matrices of PopPhy-CNN were on average the size of 5 times the number of nodes in the tree.

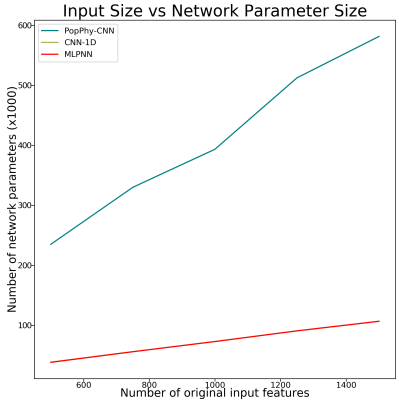

Fig S2. Number of parameters in PopPhy-CNN, CNN-1D, and MLPNN models based on original input size.

Construction of Ranked Lists for Feature Evaluation and Visualization

The features for the T2D dataset were extracted using $\theta_1 = 0.01$ and $\theta_2 = 0$ from models that performed better than the mean AUC value of the 10-fold cross-validation. From each evaluated model, we obtained a score for every feature from the

perspective of each class (disease status). Then, in every model, the scores in each class were ranked, where the lower rank indicates a more important feature. We then took the 25th percentile of the ranks for each class across the evaluated models as the overall ranking of that feature for that class. Because each CNN model can often train differently, emphasizing different areas that may be sufficient (such as using internal nodes rather than their children), we used the 25th percentile to better capture a general ranking landscape.

Next we constructed a joint ranked list in order to combine features across all classes into a single ranked list. To do this, we calculated the difference of every feature's score from the healthy class and from the disease class. This gave us a single score for every feature and every model. Then, to reduce the number of scores per feature to a single score, we take the 75th percentile of the absolute value of the calculated differences as the overall score for a feature. These values were then ranked, giving a single rank per feature.

In order to calculate the scores for visualizing the annotated tree, we utilized both the feature rankings within each class and the absolute values of the score differences used in constructing the joint ranked list. Using the score differences, if a feature was better ranked in the healthy class, the score was multiplied by -1. If the feature was better ranked in the disease class, it was kept positive. Lastly, if the rankings in each class for a single feature were within 20 ranks of each other, the score was dropped due to being ranked similarly.

Further Evaluation of Extracted Features

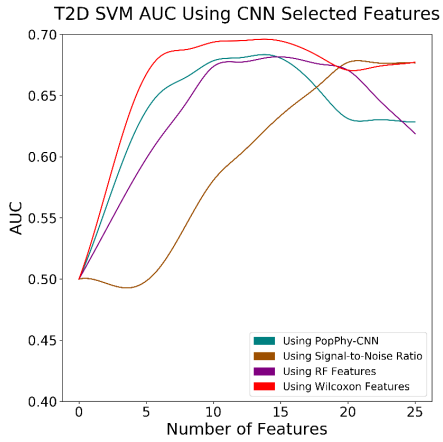

Fig S3. Benchmarking of top 25 features extracted from PopPhy-CNN for T2D. Features extracted from PopPhy-CNN (teal), RF (purple), signal-to-noise ratio (brown), and a Wilcoxon rank-sum test (red) are shown.

Benchmarking of features for the T2D dataset showed little difference between the methods as shown in Fig. S3. In addition, we analyzed the overlap between the features extracted by PopPhy-CNN, RF, and a Wilcoxon rank-sum test. We observed that there was the most overlap in the cirrhosis dataset, which was the easiest dataset. Likewise, there was the least overlap in the obesity dataset, which was the hardest dataset of the three. Venn Diagrams of the overlap are shown in Fig. S4

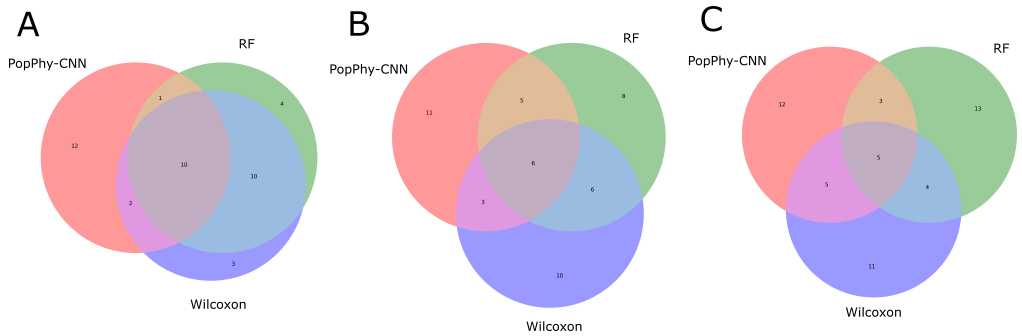

Fig S4. Venn Diagrams showing the amount of overlap in the top 25 selected features from PopPhy-CNN, RF, and a Wilcoxon rank-sum test. Images are shown for overlaps in the (A) cirrhosis dataset, (B) type 2 diabetes dataset, and (C) obesity dataset.