Perelman School of Medicine
UNIVERSITY of PENNSYLVANIA

An Automated Deep Learning Analysis Pipeline for Classification in Tabular Biomedical Data

Keshav Ramji, Ryan J. Urbanowicz

Background

- Machine Learning strategies have become an integral part of the biomedical informatics domain to better model, forecast, and classify complex interactions in large datasets.
- For classification tasks on such datasets, researchers in the biomedical field have often opted to use more traditional machine learning techniques (Random Forest, Support Vector Machines, Logistic Regression, etc.), as they are inherently more explainable in terms of the methods through which the classification is generated.
- More recently, however, the fast-growing field of deep learning (DL) has gained traction, particularly in the ability to accommodate a variety of tasks (binary classification, multiclass classification, multilabel classification, regression, etc.) and data modalities (image, text, tabular, speech, etc.).
  - Furthermore, these models train very well with large volumes of data and can often result in higher accuracies than traditional methods.
- While deep learning models are notably less interpretable (often considered "black box" methods), the higher accuracies presented by these models merits further exploration, particularly in the biomedical domain where interpretability has long been prized.

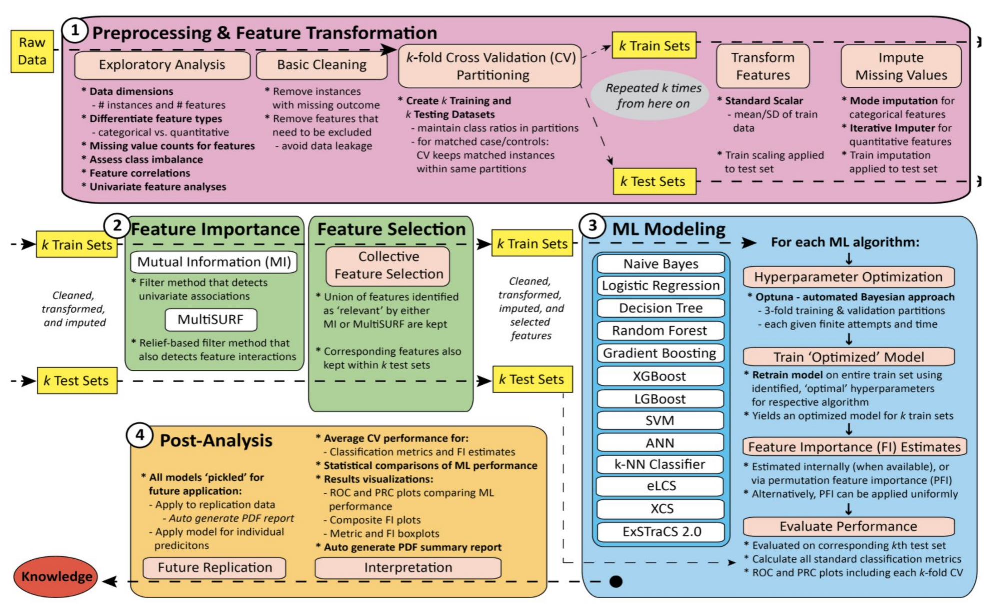


Fig 1. This schematic[1] from the AutoMLPipe-BC paper demonstrates the key stages in the pipeline, including data pre-processing and cleaning, feature importance and selection, predictive modeling, and statistical analysis.

GOAL: Expand upon the artificial neural network (ANN) approach in AutoMLPipe-BC by studying recent deep learning literature to identify the most viable and efficient approaches for tabular data. Implement the most promising DL models and compare the results to the AutoMLPipe-BC models. While we will develop the pipeline with generic classification tasks in mind, the results will be evaluated on a binary classification task.

- We will effectively develop a DL modeling pipeline to include the neural networks of interest, similar to ML Modeling component of the schematic, while keeping the remainder of the pipeline stages intact.

Model Selection for Non-Tabular Methods

- Aside from tabular model architectures, we also consider models designed for other modalities - specifically, we explore Convolutional Neural Network[5] (CNN) architectures designed for image classification tasks.
- In order to use CNNs in AutoMLPipe-DL, we would need prepare the input in the form of images for the described model architectures.
  - CNNs have traditionally not been used on tabular data due to simple Python Image Library (PIL) image transformations not considering relationships between features, which is an important component of training CNNs
  - We use DeepInsight[6], which applies the t-distributed stochastic neighbor embedding (t-SNE) method for feature extraction for dimensionality reduction and performs an image transformation on the scaled data
  - We then use the transformed image data, which considers the feature locations and values in the process of mapping cartesian coordinates to pixels, as input for the CNNs.
- In order to construct a CNN as part of AutoMLPipe-DL, we will use SKORCH (as was done with the MLP) to implement the model in PyTorch
- The convolutional layer involves sliding a filter matrix over the image (which itself is a matrix of scaled values) and iteratively computing a Frobenius inner product to generate the feature map of the image.
- The max pooling layer is a method of spatial dimensionality reduction wherein clusters of the feature map are condensed by using the maximum value within the cluster (size of cluster determined by stride)
- For this implementation, we consider a "unit" to consist of a convolutional layer followed by a max pooling layer, and conducted the hyperparameter search to determine the number of these "units" to be included in the architecture

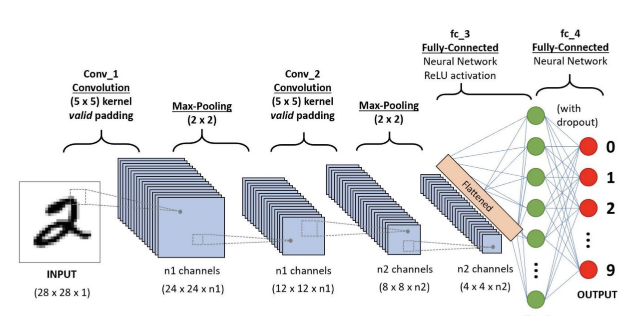


Fig 4. Convolutional Neural Network architecture[7] (from Toward Data Science).

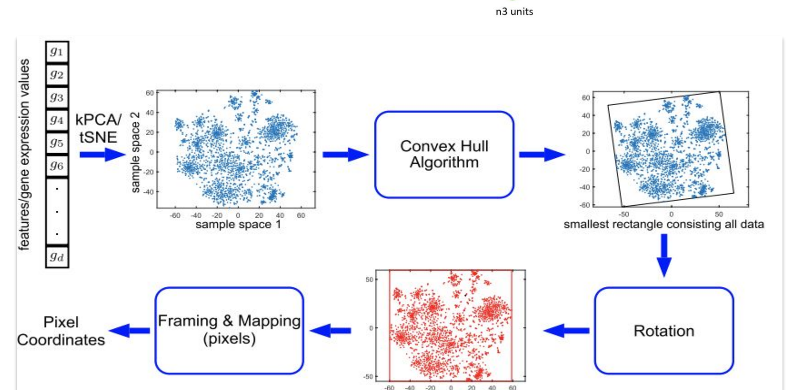


Fig 5. Schematic[6] of DeepInsight image transformation procedure (from DeepInsight paper).

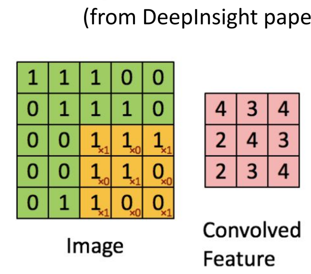


Fig 6. Visual depiction[8] of the convolutional layer in CNN.

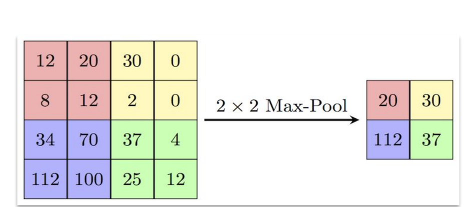


Fig 7. Visual depiction[9] of the max pooling layer in CNN.

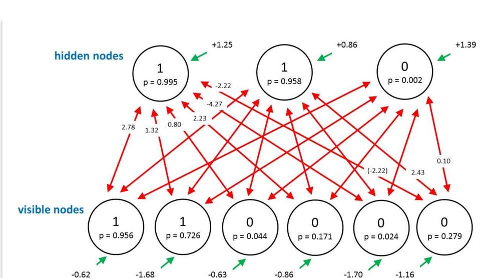


Fig 8. Restricted boltzmann machines as a binary valued undirected bipartite graph[12] with a visible and a hidden layer.

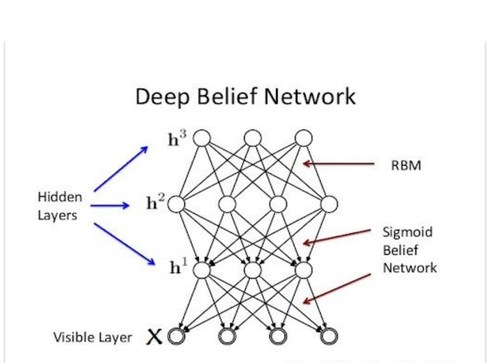


Fig 9. Illustration[14] of deep belief network with a directed sigmoid belief network and undirected RBM.

- Another approach we explore is probabilistic models - particularly, Restricted Boltzmann Machines (RBM).
- RBM are a bipartite, undirected probabilistic graphical model, which learns a joint probability distribution over the feature set and have been found to be effective as pre-trainer models, particularly in feature extraction as representation learners
  - This is a form of recurrent neural networks (RNN) that is stochastic in nature.
  - RBMs are trained using the contrastive divergence[10] algorithm, which uses gradient descent and applies Gibbs sampling (a Markov Chain Monte-Carlo randomized algorithm for Bayesian inference) to approximate the distribution over the features.
- We fit the training data using the scikit-learn BernoulliRBM method in conjunction with a number of state-of-the-art (SOTA) ML models as downstream classifiers.

- Deep Belief Networks[11] are a directed graphical model that can be constructed by stacking multiple RBMs such that the hidden layer of the previous RBM is the visible layer of the next
- Each RBM is trained sequentially using contrastive divergence and the resulting output is passed into the next RBM - it is trained greedily, and is quite effective
- We use an open-source implementation[12] which pairs the DBN with the softmax classifier

Deep Learning Introduction and Tabular Methods

Deep Neural Network Background:

- The foundational model for deep learning is the perceptron neural network.
- The multilayer perceptron (MLP) is a more complex form of the perceptron with hidden layers between the input and output layers (Fig. 1).
- In a feed-forward neural network such as the MLP, inputs are propagated through the network from neuron to neuron, with learned weights and biases used to select the output in each layer.
- Each neuron consists of a linear function followed by a non-linear activation function such as Rectified Linear Unit (ReLU), Sigmoid, or Hyperbolic Tangent (tanh), to generate the output of the neuron.
- The eventual task in this supervised learning approach is to minimize loss, which is calculated by a cost function comparing the predicted output to the actual output as identified in the data.
- Loss minimization is typically done with gradient descent or stochastic gradient descent, in which the gradient of the loss function is iteratively computed to update the model.
- The network learns in gradient descent through backpropagation, in which a chain rule of partial derivatives is used to calculate the gradient of the total loss with respect to the weights, which is then propagated backwards through the network for parameter update (Fig. 2)

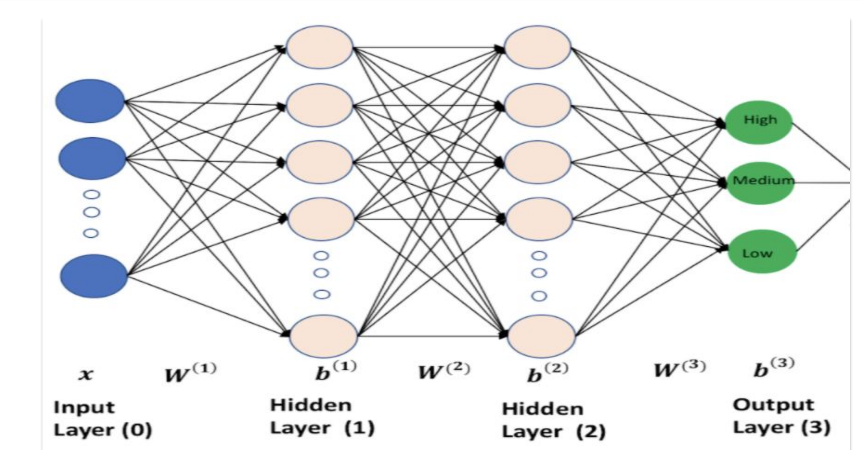


Fig 1. Illustration of multilayer perceptron model with two hidden layers and a multiclass (3 class) classification task.

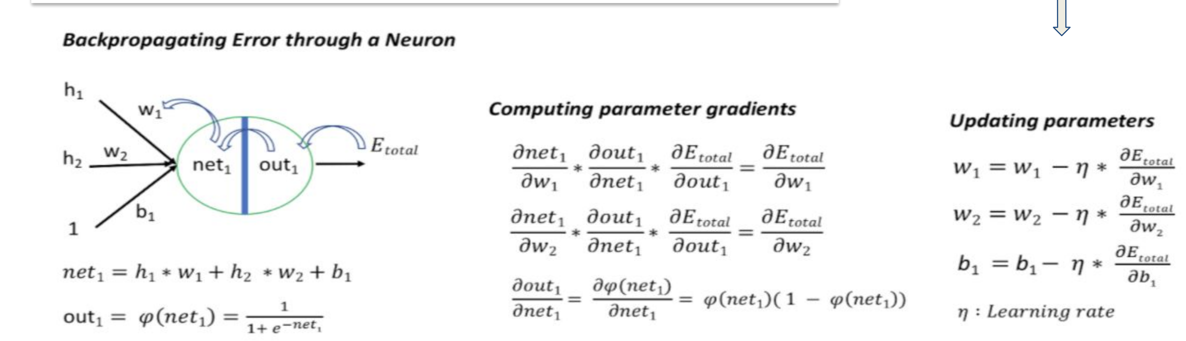


Fig 2. Illustration of backpropagation in a neural network, gradient calculation, and parameter update through chain rule of partial derivatives.

Implementation Plan:

- The ANN in AutoMLPipe-BC was developed using the MLPClassifier method in scikit-learn. However, as PyTorch and Tensorflow are the libraries of choice for deep learning research, it would be beneficial to introduce PyTorch in the pipeline for deeper neural networks (ANN with many hidden layers).
- We plan to develop the deep learning methods for AutoMLPipe-DL using SKORCH, a scikit-learn wrapper for PyTorch models[2], or where available, a scikit-learn compatible implementation for the models.
  - AutoMLPipe-BC utilizes many scikit-learn evaluation metrics in the ML modeling and the statistical analysis sections, so by ensuring that the model is a scikit-learn object, we can maintain the usage of these metrics.

Deep Learning for Tabular Data:

- While gradient boosting decision trees are widely popular for tabular data, recent deep learning models such as TabNet[3] have shown promising results
  - The TabNet model architecture takes inspiration from decision trees, using sequential attention to perform a feature selection procedure at each decision step through a learnable mask for the prediction task.
  - The TabNet implementation[4] could be trained using either a supervised or semi-supervised approach, where the unsupervised pre-trainer for representation learning is followed by a supervised fine-tuner through a masked self-supervision procedure (Fig. 3)

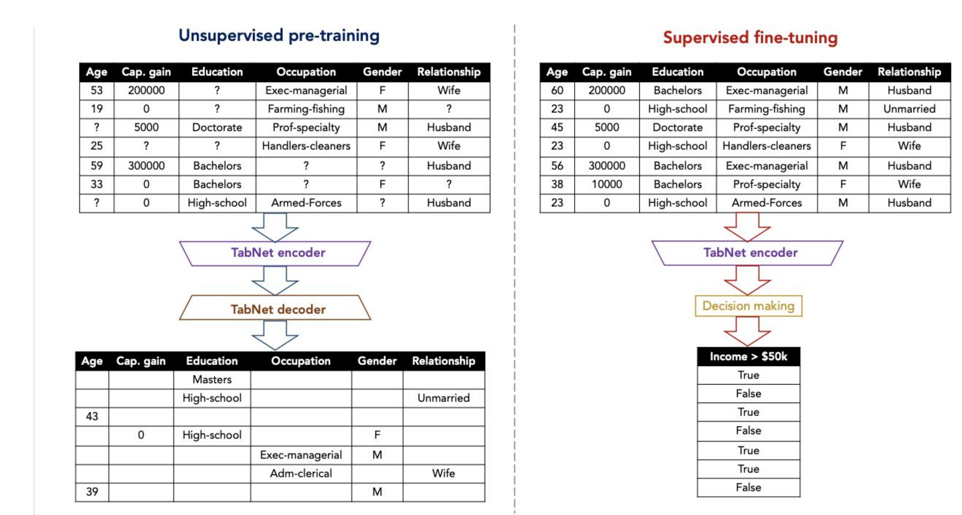


Fig 3. Figure demonstrating the masked self-supervised approach proposed in the TabNet[3] (image from paper), which improves the encoder for the supervised fine-tuning, particularly for smaller datasets.

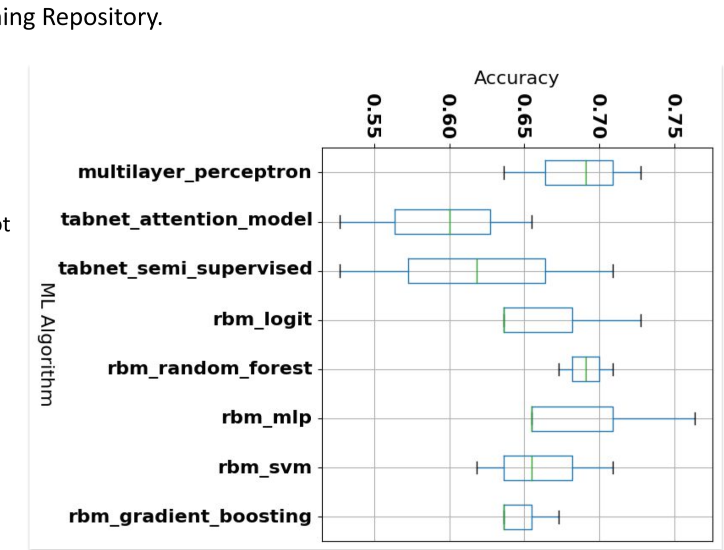
Analysis & Results on Benchmark Data

- We test the functionality of the implemented deep learning algorithms on the UC Irvine Hepatocellular Carcinoma (HCC) dataset (Table 1)
  - We report the area under the receiver operating characteristic curve (AUC-ROC) as the most relevant accuracy metric.
- We run both the AutoMLPipe-DL and AutoMLPipe-BC pipelines to compare the efficacy of our newly implemented models
- Given deep learning models often perform better with larger data, running the pipeline on a larger dataset could potentially yield better results.

Analysis of AutoMLPipe-DL and Deep Learning Methods on HCC Data	AUC-ROC for HCC Data	AUC-ROC for HCC Data with No Covariates
Multilayer Perceptron	**0.768**	**0.746**
Supervised TabNet	0.668	0.583
Semi-Supervised TabNet	0.589	0.557
Restricted Boltzmann Machines + Logistic Regression	**0.693**	0.693
Restricted Boltzmann Machines + Random Forest	0.668	**0.735**
Restricted Boltzmann Machines + Multilayer Perceptron	0.676	0.711
Restricted Boltzmann Machines + Support Vector Machines	0.577	0.711
Restricted Boltzmann Machines + Gradient Boosting Classifier	**0.689**	**0.725**

Table 1: Results of AutoMLPipe-DL on Hepatocellular Carcinoma (HCC) datasets from UC Irvine Machine Learning Repository.

Fig 10. Accuracy boxplot for AutoMLPipe-DL models for the HCC dataset with no covariates included.



- It is worth noting that the models run in AutoMLPipe-DL do not utilize the Optuna hyperparameter optimization due to limitations in the current version while those in AutoMLPipe-BC (Table 2) were tuned with Optuna.
- However, the existing DL models already present comparable performance and we have reason to believe that we can further improve these models using Optuna, especially the complex TabNet models.

Analysis of AutoMLPipe-BC and ML Methods on HCC Data	AUC-ROC for HCC Data	AUC-ROC for HCC Data with No Covariates
Naive Bayes	0.672	0.708
Logistic Regression	0.720	0.728
Decision Tree Classifier	0.617	0.686
Random Forest Classifier	**0.761**	**0.750**
Gradient Boosting Classifier	0.706	0.712
XGBoost Classifier	**0.725**	0.692
LightGBM Classifier	0.696	**0.736**
Support Vector Machines	**0.722**	**0.758**
Multilayer Perceptron	0.702	0.729
k-Nearest Neighbors	0.644	0.691
Educational Learning Classifier System (eLCS)	0.664	0.684
Accuracy-based Learning Classifier System (XCS)	0.667	0.686
Extended Supervised Tracking and Classifying System (ExSTraCS)	0.638	0.670

Table 2: Results of AutoMLPipe-BC on the HCC datasets.

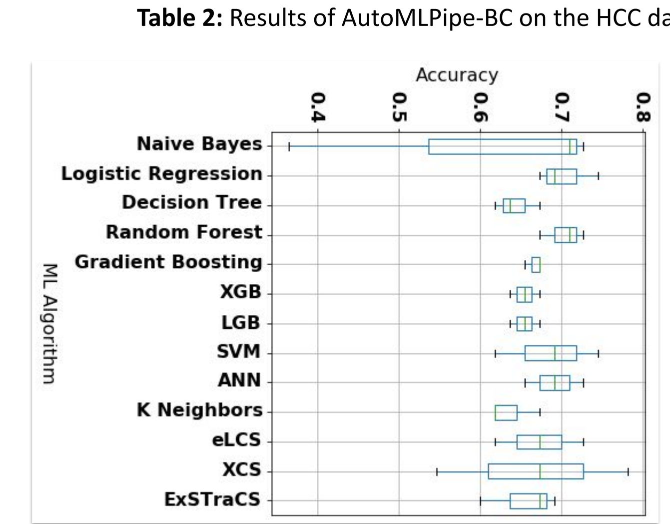


Fig 11. Accuracy boxplot for AutoMLPipe-BC's ML models for the HCC dataset with no covariates included.

Conclusions

- Successfully implemented the AutoMLPipe-DL pipeline with 12 deep learning models (MLPClassifier, SKORCH MLP, CNN, Supervised TabNet, Semi-Supervised TabNet, RBM with 6 different downstream classifiers, and DBN).
- Achieved promising results on 8 DL models as compared to SOTA techniques in AutoMLPipe-BC using UC Irvine Hepatocellular Carcinoma data.
- This DL pipeline provides an interesting research opportunity to explore deep learning models on larger and more complex biomedical datasets.

Future Work:

- Continue testing and debugging of SKORCH models to integrate them in the pipeline and apply Optuna hyperparameter optimization for all the models.
- Expand upon the CNN model implementation to incorporate other notable CNN architectures, including VGGNet-19, AlexNet, ResNet, Inception-v4, and DenseNet.
- Implement the TabTransformer, an effective self-attention model which requires creating an implementation using SKORCH.
- Complete the minimal changes necessary to adapt AutoMLPipe-DL to multiclass classification tasks, and examine its performance on other biomedical datasets.

References

1. Urbanowicz, Ryan J., et. al. "A Rigorous Machine Learning Analysis Pipeline for Biomedical Binary Classification: Application in Pancreatic Cancer Nested Case-control Studies with Implications for Bias Assessments", arXiv: 2008.12829 [cs.LG]
2. Skorch-Dev. (n.d.). *Skorch-Dev/Skorch: A scikit-learn compatible neural network library that WRAPS PYTORCH*. GitHub. https://github.com/skorch-dev/skorch.
3. Arik, Sercan O. and Pfister, Tomas. "TabNet: Attentive Interpretable Tabular Learning", arXiv: 1908.07442 [cs.LG]
4. Dreamquark-Ai. (n.d.). *Dreamquark-Ai/Tabnet: PyTorch implementation Of tabnet paper : Https://arxiv.org/pdf/1908.07442.pdf*. GitHub. https://github.com/dreamquark-ai/tabnet.
5. Y. LeCun et al., "Backpropagation Applied to Handwritten Zip Code Recognition,", *Neural Computation*, vol. 1, no. 4, pp. 541-551, Dec. 1989, doi: 10.1162/neco.1989.1.4.541.
6. Sharma, A., *et al.* "DeepInsight: A methodology to transform a non-image data to an image for convolution neural network architecture" *Sci Rep* 9, 11399 (2019). https://doi.org/10.1038/s41598-019-47765-6
7. Saha, S. (2018, December 17). *A comprehensive guide to convolutional neural networks - the eli5 way*. Medium. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.
8. Ujjwalkarn. (2017, May 29). *An intuitive explanation of convolutional neural networks*. the data science blog. https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/.
9. *Max-pooling / pooling*. Max-pooling / Pooling - Computer Science Wiki. (n.d.). https://computersciencewiki.org/index.php/Max-pooling_/_Pooling.
10. Hinton, G. E. (2002). "Training products of experts by minimizing contrastive divergence". *Neural computation*, *14*(8), 1771-1800.
11. Hinton, G. E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, *18*(7), 1527-1554.
12. Albertbup. (n.d.). *Albertbup/Deep-Belief-Network: A Python implementation of deep belief networks built upon numpy and TensorFlow with scikit-learn compatibility*. GitHub. https://github.com/albertbup/deep-belief-network.
13. *Restricted boltzmann machines using c#*. James D. McCaffrey. (2017, June 2). https://jamesmccaffrey.wordpress.com/2017/06/02/restricted-boltzmann-machines-using-c/.
14. *Deep belief network*. Deep Belief Network Introduction. (n.d.). http://www.ashokrahulgade.com/skills/DL/DBN/Introduction.html.

DEPARTMENT of BIOSTATISTICS EPIDEMIOLOGY & INFORMATICS

Institute for Biomedical Informatics

Acknowledgements

This work was supported by the University of Pennsylvania Center for Undergraduate Research and Fellowships (CURF) as part of the PURM program.

Lab/Research Links

www.med.upenn.edu/urbslab/
www.ryanurbanowicz.com

Code Availability

github.com/UrbsLab
github.com/UrbsLab/AutoMLPIpe-DL