Using Neural Networks to Identify Gene Regulatory Relationships

Samiul Haque
Electrical & Computer Engineering
North Carolina State University
Raleigh, USA
shaque2@ncsu.edu

Kavya Sri Kondaveeti
Electrical & Computer Engineering
North Carolina State University
Raleigh, USA
kkondav@ncsu.edu

Selene R. Schmittling
Electrical & Computer Engineering
North Carolina State University
Raleigh, USA
srschmi3@ncsu.edu

***Abstract*—Gene regulatory network (GRN) inference using gene expression data (microarray or RNA-seq) remains a challenging task in computational biology. In this work we propose to use RNNs to identify regulatory relationships between genes using time series gene expression data. We test our model on synthetic data from the DREAM4 challenge. We expect our RNN to perform as well, if not better than existing non-supervised methods.**

***Index Terms*—gene regulatory networks, GRN, neural networks, recurrent neural networks, RNN, CNN, AE, LSTM, Keras**

I. Introduction

Gene regulatory network (GRN) inference involves identifying regulatory relationships between genes of an organism using gene expression data (microarray or RNA-seq). It remains a challenging task in computational biology. The goal of GRN inference is to generate a directed graph (like that shown in Figure 1) that shows regulating genes and their targets. The edges can also indicate whether the regulation results in activation or repression.

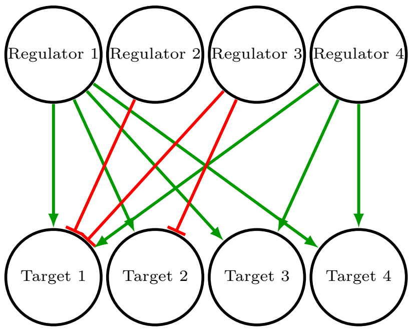


Fig. 1. A sample Gene Regulatory Network (GRN), using a Neural network architecture we want to infer these regulatory relationships from time series gene expression data.

Many algorithms have been used to perform GRN inference [1], [2]. However, semi-supervised and supervised classification using support vector machines (SVMs) shows improved performance compared to unsupervised techniques [1]. While useful, SVMs do not capture time dependencies present in time-series data. Recurrent Neural Networks (RNNs) capture time dependencies and have been used in GRN inference. However, they are generally used in conjunction with Particle Swarm Optimization (PSO) or Genetic Algorithms (GAs) to identify parameters and dynamic relationships [3]–[5].

In this project, we explore the feasibility of using three different architectures that have the abililty to incorporate temporal information in the learning process. We chose a recurrent neural network (RNN), an autoencoder (AE) the output of which is fed into a traditional neural network, and a convolutional neural network (CNN). We discuss our choices in the Architectures section below.

II. Data

Our data was created for the Dialogue for Reverse Engineering Assessments and Methods (DREAM) DREAM4 challenge [2], [6]. The DREAM4 challenge was a competition which tasked the participants with identifying techniques to infer GRNs from *in silico* gene expression data. The results and gold standard networks from this competition are well documented and available to the public through the DREAM consortium website. The time series training data for 10- and 100-node networks is available online. The data represents synthetic data generated for 21 time points and 10 perturbations for 5 different networks. While the data is synthetic it was generated based on existing networks in yeast and bacteria. We generated data in gene pairs such that the regulator's expression was always first and target always second. We generated one record for each positive and each negative connection. This left us with 2500 positive examples and ¿ 9000 negative examples. In order to avoid issues with class imbalance we generated balanced data by randomly sampling from the data with negative labels such that we had the same number of negative as positive.

Due to the data requirements for neural networks, we trained our networks using the 100-node data. For each pair of genes (nodes) we generated a label based on their regulatory connection (i.e +1 for positive causal connection , and 0 for no connection). The time series data was generated in slightly different configurations depending on the architecture. These details are found in the Architecture section below.

Popular techniques for identifying regulatory relationships are based on features like correlation, time-lagged correlation and mutual information. Fig. 2 shows these features for our data. It is easy to see that they are not well suited for classification in their raw form.

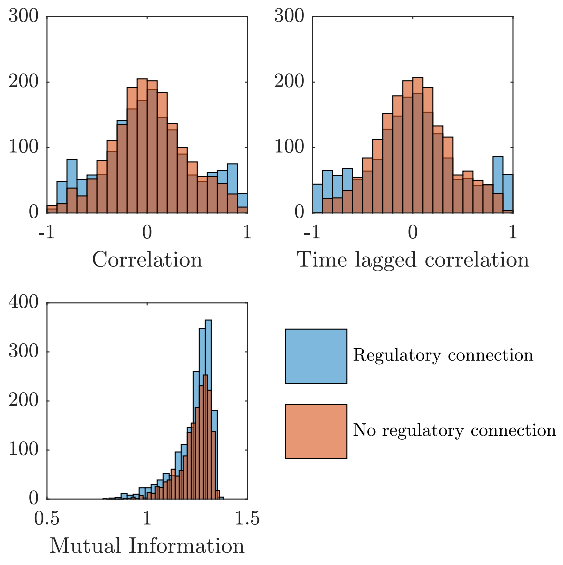


Fig. 2. Distribution of the correlation, time-lagged correlation and mutual information metrics for regulatory and non-regulatory pairs of genes.

III. Architectures

All architectures were developed using keras with a tensorflow back-end.

A. Recurrent Neural Network

We modeled our recurrent neural network to try to extract temporal information to better classify the gene regulatory relationships. We fed the time course data as a 2-dimensional vector of 21x2 into LSTM units layer. We found a dropout of 0.2 resulted in better classification during hyperparamter optimization of layers. We also found that a ”sigmoid” activation function had better accuracy and did not converge when using ”softmax” for the final layer. The learning rate of 0.01 during the Adam optimizer resulted in better accuracy and loss than other learning rate. We did not have the opportunity to further tune the hyperparameters for this network except in the adhoc fashion, as described above. The final architecture for this RNN is shown in Fig. 3

B. Convolutional Neural Network

In order to see if it could extract time features from our data, we implemented a convolutional neural network. The architecture for this model is shown in Fig. 4. We fed the time course data as a 1-channel 2x21 ”image” with labels as indicated above. Fig. 5 shows an example of input data to the

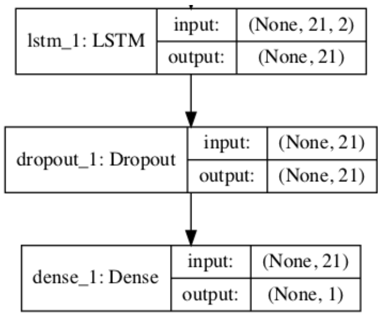


Fig. 3. Architecture for RNN model

CNN. The convolutional neural network was adapted from the LeNet-5 architecture. [7]

We did not have the opportunity to tune hyperparameters for this network except in an adhoc fashion.

C. Autoencoder & Neural Network

We implemented an AE to see if temporal information could be extracted as features in a lower dimensional representation of the data. We used the architecture shown in Fig. 10. We extracted 3-dimensional vectors and fed them into a traditional NN, the architecture of which is shown in Fig. 6.

For the AE, we loaded the expression patterns for each gene into the network separately. We then fed these lower dimensional representations into the NN. An example of the lower dimensional data is shown for Network 1 in Fig. 7. High dimensional data is shown in Fig. 8.

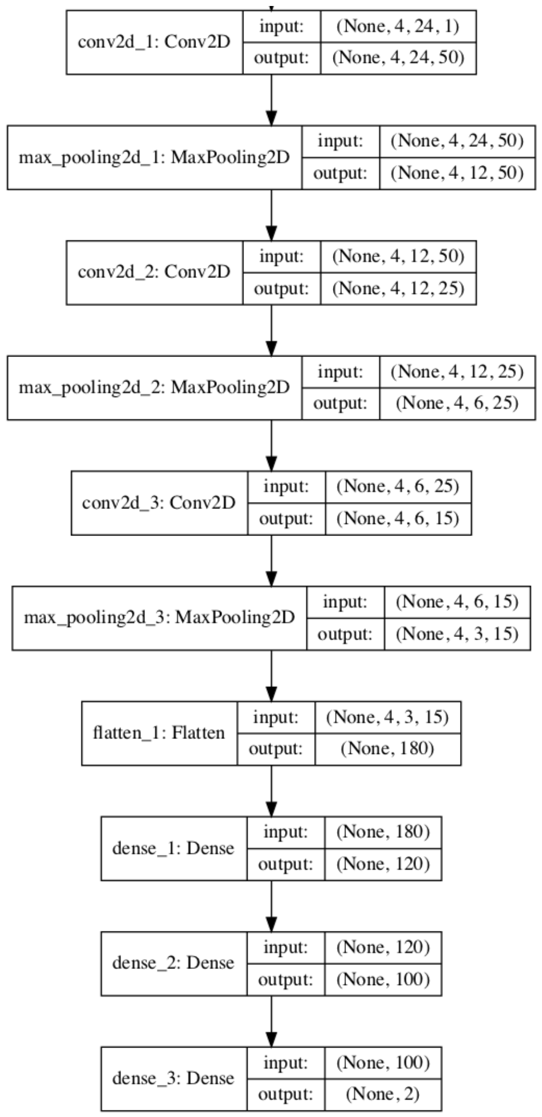
For the NN, we generated gene expression pairs. We fed the expression patterns for the regulators into one multi-layer sub-network and the patterns for their targets into a matching sub-network. These two networks were concatenated together and fed into three final layers including the output layer.

We did not have time to tune the Autoencoder. However, we were able to tune the following hyperparameters for the NN: batch size, number of epochs and optimizer. We did a grid search for batch size = [5, 10, 50, 100, 250], epochs = [5, 10, 50, 75] and optimizer = [’rmsprop’, ’adam’, ’sgd’]. Fig. 9. The best performing hyperparameters were batch size = 100, epochs = 75 and the Adam optimizer.

IV. Results

In order to quantify the performance of our networks, we calculated the area under the receiver operating characteristic (AUROC) and the area under the precistion-recal curve (AUPR). We were able to compare these values to the top 3 methods identified in the DREAM4 competition. However, the following caveats should be kept in mind: (i) we are comparing supervised methods to unsupervised methods; (ii) our test dataset was a subset of the sets used in the DREAM4 competition.

The AUPR indicates that our networks outperformed the DREAM4 competition networks. However, the AUROC indicates generally the reverse.



conv2d_1: Conv2D | input: (None, 4, 24, 1) | output: (None, 4, 24, 50)
max_pooling2d_1: MaxPooling2D | input: (None, 4, 24, 50) | output: (None, 4, 12, 50)
conv2d_2: Conv2D | input: (None, 4, 12, 50) | output: (None, 4, 12, 25)
max_pooling2d_2: MaxPooling2D | input: (None, 4, 12, 25) | output: (None, 4, 6, 25)
conv2d_3: Conv2D | input: (None, 4, 6, 25) | output: (None, 4, 6, 15)
max_pooling2d_3: MaxPooling2D | input: (None, 4, 6, 15) | output: (None, 4, 3, 15)
flatten_1: Flatten | input: (None, 4, 3, 15) | output: (None, 180)
dense_1: Dense | input: (None, 180) | output: (None, 120)
dense_2: Dense | input: (None, 120) | output: (None, 100)
dense_3: Dense | input: (None, 100) | output: (None, 2)

Fig. 4. Architecture for CNN model

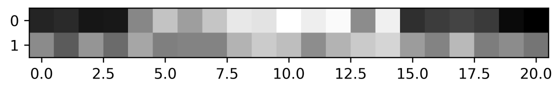
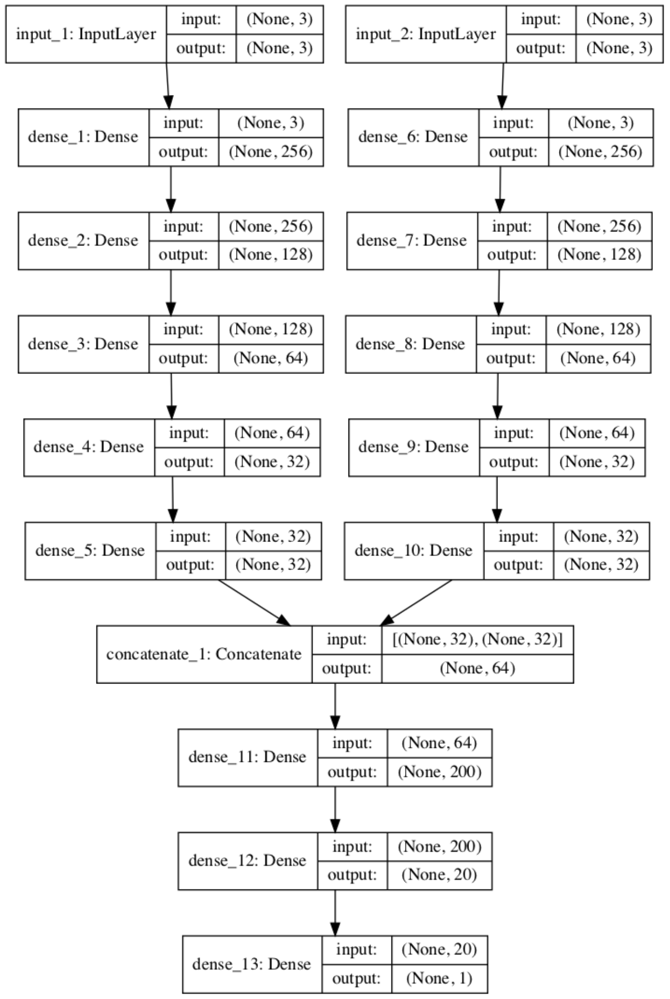


Fig. 5. Example of a sample input (regulatory gene pairs) to CNN



input_1: InputLayer | input: (None, 3) | output: (None, 3)
input_2: InputLayer | input: (None, 3) | output: (None, 3)
dense_1: Dense | input: (None, 3) | output: (None, 256)
dense_6: Dense | input: (None, 3) | output: (None, 256)
dense_2: Dense | input: (None, 256) | output: (None, 128)
dense_7: Dense | input: (None, 256) | output: (None, 128)
dense_3: Dense | input: (None, 128) | output: (None, 64)
dense_8: Dense | input: (None, 128) | output: (None, 64)
dense_4: Dense | input: (None, 64) | output: (None, 32)
dense_9: Dense | input: (None, 64) | output: (None, 32)
dense_5: Dense | input: (None, 32) | output: (None, 32)
dense_10: Dense | input: (None, 32) | output: (None, 32)
concatenate_1: Concatenate | input: [(None, 32), (None, 32)] | output: (None, 64)
dense_11: Dense | input: (None, 64) | output: (None, 200)
dense_12: Dense | input: (None, 200) | output: (None, 20)
dense_13: Dense | input: (None, 20) | output: (None, 1)

Fig. 6. Architecture of Neural Network Model

A. *Area under precision recall curve (AUPR)*

The AUPR is shown in Fig. 11.

B. *Area under ROC curve (AUROC)*

The AUROC is shown in Fig. 12.

V. Discussion and Conclusions

In general the RNN and AE outperformed the CNN with the exception of Network 3. Overall, the AE performed the strongest. Even when the RNN performed better than the AE, it only performed slightly better. The performance for our networks was consistent for both AUPR and AUROC.

When we compare our networks to the DREAM4 competition networks, however, we get mixed results. Our networks performed better according to the AUPR, but worse according to the AUROC. In general, the ROC is a better measure when the priors on your classes are relatively equal. However, AUPR is a better measure for imbalanced data [8].

Unsupervised methods are good at capturing regulatory connections where expression patterns are tightly correlated. However, they also identify many indirect connections (i.e

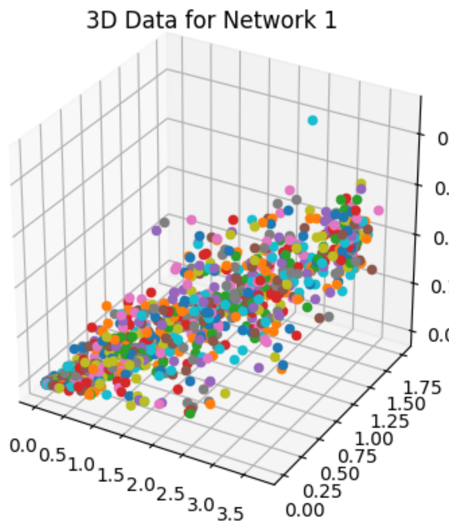


Fig. 7. Expression data reduced to 3 Dimensional latent space using Autoencoder

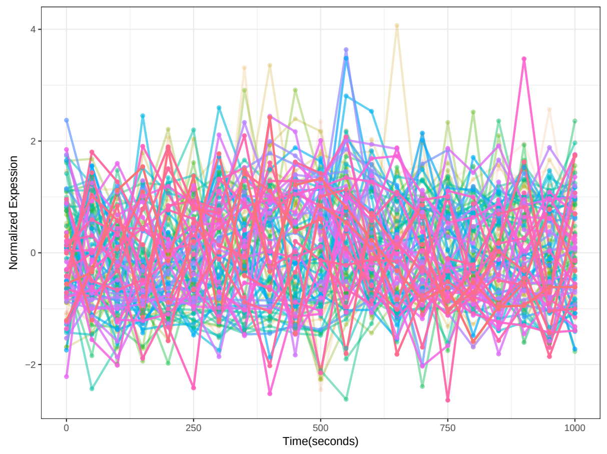


Fig. 8. High dimensional normalized gene expression patterns for DREAM4 network 1

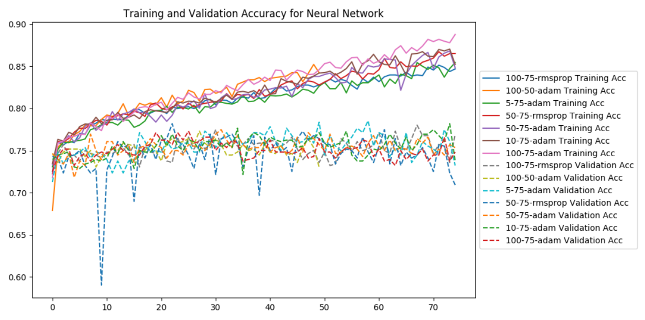


Fig. 9. Training and Validation Accuracy for Neural Network during Hyperparameter Tuning. Hyperparameter settings are indicated as Batch-Epoch-Optimizer.

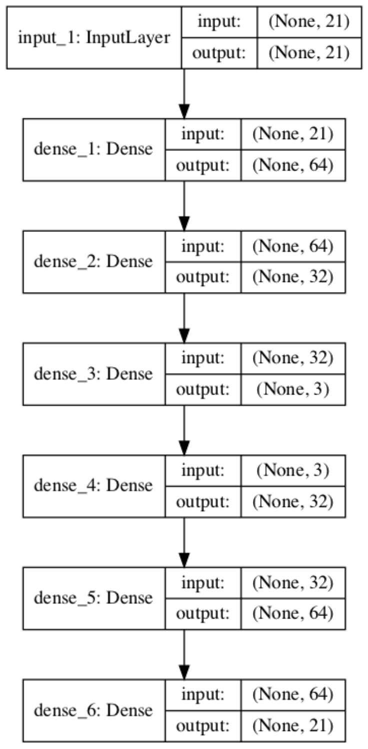


Fig. 10. Architecture of Autoencoder Model

feed forward loops) as direct connections. In addition, many of these methods do not distinguish between direction of the regulation (i.e. causal relationship). In our supervised approach we have discriminated these cases for example we do not label genes that have indirect regulatory relationship as positive class. Also, if a pair of genes (G1,G2) has regulatory relationship than we assign (G1,G2) pair as class 1 and (G2,G1) as class 0. In this way we learn the causal direction between these genes. This approach resulted in less false positive than unsupervised methods, however in this approach we failed to learn some of the true regulatory connections.

This project was a proof-of-concept project. Since we were able to achieve modestly good results, we believe that a next step would be to perform more intensive hyperparameter tuning. It is intriguing and unexpected that the autoencoder performed competitively with the other two architectures. More aggressive tuning might reveal better performance by the RNN and CNN. It would also be interesting to see if a hybrid RNN/CNN architecture performed well. We would also like to try generating higher than 3-dimensional data with the autoencoder would be of interest. We also considered feeding data in different configurations, for instance bi-directionally. In this case two instances would be created: Regulator-Target (class 1 if connected and 0 if not), and Target-Regulator (class 2 if connected and 0 if not).

Finally, we would like to investigate what the convnet filters

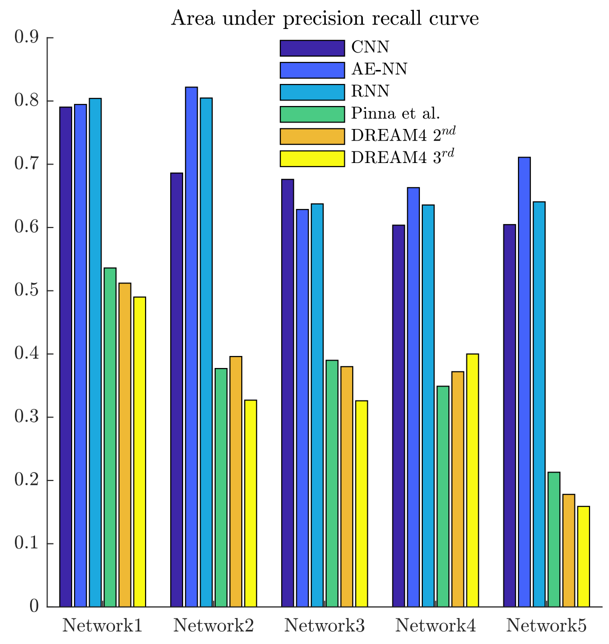


Fig. 11. Comparison of the AUPR for different methods on five DREAM4 networks

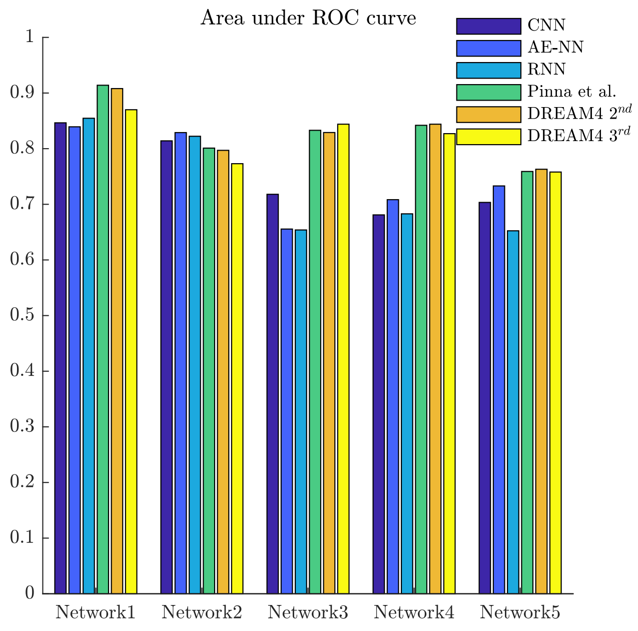


Fig. 12. Comparison of AUROC for different methods on five DREAM4 networks

are learning. It would be interesting to see if any temporal features could be identified from the filters. We would also like to compare the types of network structures that our networks were able to identify.

References

[1] S. R. Maetschke, P. B. Madhamshettiwar, M. J. Davis, and M. A. Ragan, "Supervised, semi-supervised and unsupervised inference of gene regulatory networks," *Briefings in bioinformatics*, vol. 15, no. 2, pp. 195–211, 2013.

[2] D. Marbach, J. C. Costello, R. Küffner, N. M. Vega, R. J. Prill, D. M. Camacho, K. R. Allison, A. Aderhold, R. Bonneau, Y. Chen *et al.*, "Wisdom of crowds for robust gene network inference," *Nature methods*, vol. 9, no. 8, p. 796, 2012.

[3] R. Xu, D. Wunsch II, and R. Frank, "Inference of genetic regulatory networks with recurrent neural network models using particle swarm optimization," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 4, no. 4, pp. 681–692, 2007.

[4] N. Noman, L. Palafox, and H. Iba, "Reconstruction of gene regulatory networks from gene expression data using decoupled recurrent neural network model," in *Natural computing and beyond*. Springer, 2013, pp. 93–103.

[5] M. M. Kordmahalleh, M. G. Sefidmazgi, S. H. Harrison, and A. Homaifar, "Identifying time-delayed gene regulatory networks via an evolvable hierarchical recurrent neural network," *BioData mining*, vol. 10, no. 1, p. 29, 2017.

[6] A. Greenfield, A. Madar, H. Ostrer, and R. Bonneau, "Dream4: Combining genetic and dynamic information to identify biological networks and dynamical models," *PloS one*, vol. 5, no. 10, p. e13397, 2010.

[7] Y. LeCun *et al.*, "Lenet-5, convolutional neural networks," *URL: http://yann. lecun. com/exdb/lenet*, p. 20, 2015.

[8] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 233–240.