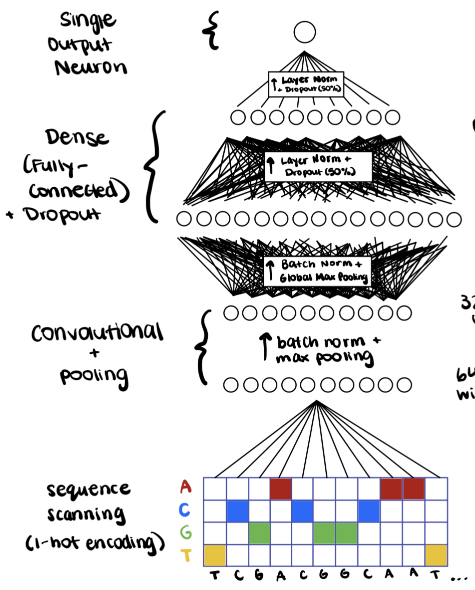
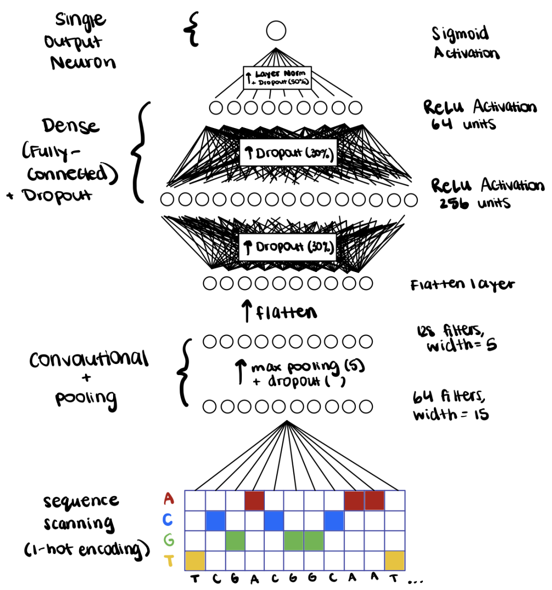
Anna Corcoran and William Skelly- CS260 Project 3 Report

Developed Models:

Model 1 (sim1 and sim2):



Model 2 (sim6 and sim7):



Model 1:

Each model processes DNA sequences of length 249, one-hot encoded at each position into 4 dimensions (one per DNA base). Model 1 starts with a convolutional layer detecting 5-bp patterns with 64 filters, followed by batch normalization and max pooling over pairs of positions. A second convolutional layer, also with a 5-bp kernel but 32 filters, extracts higher-level features, followed by another batch normalization. Global max pooling then reduces each of the 32 feature maps to its strongest activation, producing a fixed-length vector. This is passed to a dense layer with 64 units and ReLU activation, followed by layer normalization and dropout (rate 0.5). This sequence is repeated with 32 units in the dense layer to further compress the representation. The final output layer uses a single neuron with sigmoid activation for binary classification. Training was performed with the ADAM optimizer and binary cross entropy loss.

Model 2:

For Model 2, the network begins with a convolutional layer using 64 filters and a 15-bp kernel to capture longer-range patterns, followed by max pooling over windows of 5 positions and a dropout layer (rate 0.3). A second convolutional layer with 128 filters and a 5-bp kernel extracts higher-level features, again followed by max pooling (size 5) and dropout (rate 0.3). The output is flattened and passed through a dense layer with 256 units and ReLU activation, followed by another dropout layer (rate 0.3). A second dense layer with 64 units and ReLU further compresses the representation. The final output layer uses a single neuron with sigmoid activation for binary classification. Training was performed with the ADAM optimizer and binary cross entropy loss.

Methods:

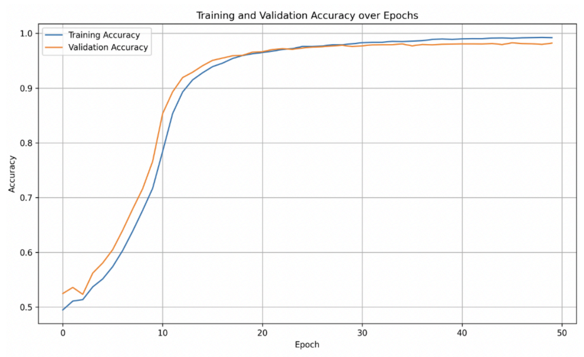
Our general model design process was to create a complex model, compare training and validation accuracy on each set, and then modify the model for each set according to those results.

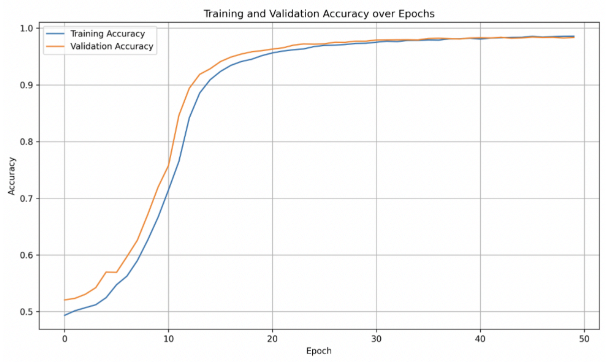
As a baseline, a random prediction model was designed. 0 or 1 was randomly generated and assigned to each sequence, then compared to the true class.

Simple Model: We first built a very simple model, with only a flatten layer then three dense layers. This had around 50% accuracy for both the training and validation sets, supporting our theory that we should add convolutional filters to recognize patterns.
First Complex Model: We then built a more complex model and trained and validated it on each set. We had two convolutional filters, two dense layers, and normalization and pooling throughout. The convolutional layers were designed to identify conserved sequence motifs, as enhancers often share evolutionarily preserved patterns, and we hoped this would improve accuracy. We applied layer normalization in the dense layers instead of batch normalization to stabilize training across individual feature vectors. Since the convolutional layers are followed by global max pooling, the fully connected layers operate on single feature vectors, making layer normalization more effective here. This model had high (>99%) accuracy on the training and validation sets for sim1 and sim2, but low (48-50%) accuracy on the training and validation sets for sim6 and sim7. We decided to keep this model for sim1 and sim2, but modify it for sim6 and sim7.
Second Complex Model: For sim6 and sim7, we tested several models to improve accuracy. We first modified our original architecture by adding additional and more varied convolutional layers, and also explored a sequential dilated stack to capture motifs of different lengths. We chose not to implement GRU or LSTM architectures due to their long runtimes and high memory demands. Next, we developed a model based on the DeepSTARR framework, which had better accuracy. We used this framework moving forward. To balance unequal training and validation performance, we experimented with different dropout rates—lowering dropout to improve training accuracy, and increasing it when validation accuracy outperformed training. We found that a 30% dropout rate provided the best balance, achieving high accuracy without overfitting (shown below).

Sim6 Dropout Optimization			Sim7 Dropout Optimization		
Dropout Rate	Training Accuracy	Validation Accuracy	Dropout Rate	Training Accuracy	Validation Accuracy
50%	96.84%	97.11%	50%	96.77%	97.15%
30%	**98.46%**	**98.18%**	**30%**	**98.78%**	**98.37%**
20%	99.05%	98.04%	20%	99.34%	98.22%

Example: Sim7 Dropout Rate 20% (Left) and Dropout Rate 30% (Right)



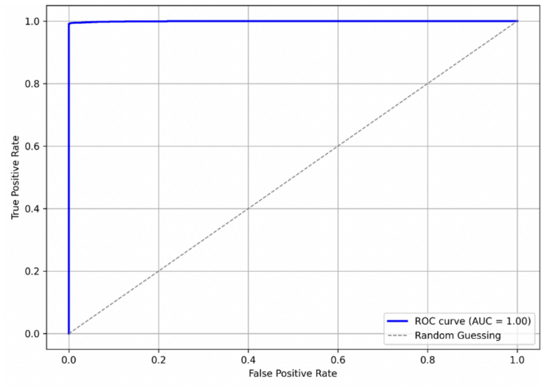


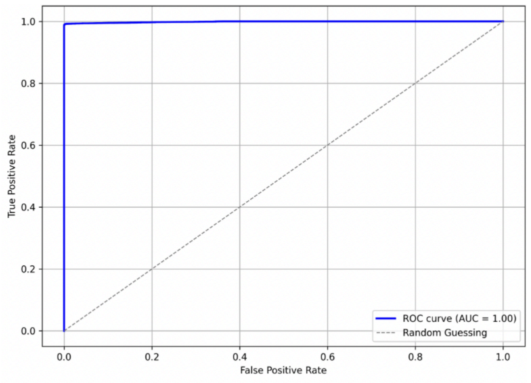
The first graph shows validation accuracy lower than training, demonstrating that the model overfits the data. The second graph shows no overfitting, demonstrating that the optimal dropout rate was reached.

Results:
Our baseline model (random classification) had a 49.99% accuracy, as expected for binary classification. Final results for each model are below:

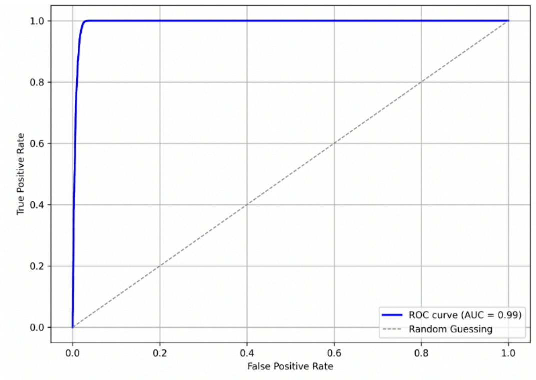
Accuracy	**Sim1 (Model 1)**	**Sim2 (Model 1)**	**Sim6 (Model 2)**	**Sim7 (Model 2)**
Training	99.75%	99.42%	98.65%	98.28%
Validation	99.55%	99.39%	98.45%	98.27%
Testing	99.50%	99.26%	98.38%	98.43%
Loss	**Sim1 (Model 1)**	**Sim2 (Model 1)**	**Sim6 (Model 2)**	**Sim7 (Model 2)**
Training	0.0126	0.0279	0.0365	0.0547
Validation	0.0275	0.0313	0.0432	0.0649
Testing	0.0279	0.0367	0.0585	0.0459

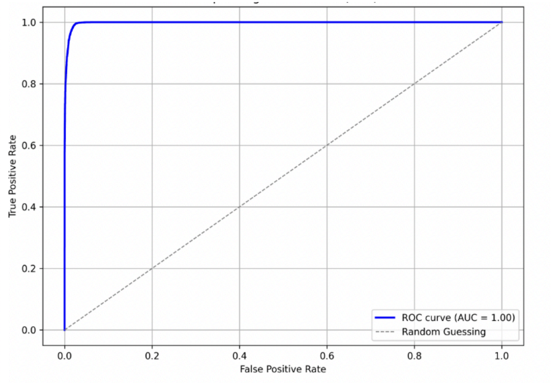
ROC Curves:
Model 1 (Sim1 AUC: 1.00, Sim2 AUC: 1.00)





Model 2 (Sim6 AUC: 0.99, Sim7 AUC: 1.00):





All of our AUCs were 0.99 or 1.00, demonstrating high accuracy for our binary classification models.