MACHINE LEARNING FOR REGULATORY GENOMICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Promoter Expression Prediction Dream Challenge

Dmytro Vynnyk, Kaan Guney Keklikci, Yevhenii Sharapov, Cristian Sandu

Contents

1 Introduction

Machine Learning in Regulatory genomics plays an important role in contemporary studies, Previously, scientists had to overcome the hardships of the limited inference toolkit that was constrained to hypothesis-driven tests and relied drastically on high dimensional statistics and alignment approaches, which alas didn't provide researchers with hardly any generalization. They also didn't enjoy much data at their disposal as well as effective ways to process them and retrieve results. With the advent of the convolution neural networks albeit and their rapid development not only these problems were overcome, but also we had learned to interpret their parameters, which in turn could save a lot of either effort or investments.

One of the particularly interesting areas is deemed to be cis-regulatory genomics, which studies the way genes are expressed using proteins. Certain Transcription Factors (TFs) play a vital role in fostering or repressing the expression of genes. To unravel the *apriori* unknown knowledge of these motifs, a need for successive experiments arises. On contrary using big genome banks, containing numerous sequences has shown to be sufficient to restore a lot of information about the TFs when dealing with yeast genetic material, specifically 92 percent appear in each 10^5 bps [1]. Further experiments also showed that random sampling of DNA sequences enables to restore most of the expression levels. The motivation behind using random sequences is the abundance of DNA TF binding sites included in random DNA by chance whose expression distribution is similar to the actual DNA [2].

Researchers in many branches of the genetic machine learning are in the active search of the most generalization capable algorithms and models, to be able to seize the common traits present in versatile cell-lines. Another pivot of curiosity is as well an opportunity to train the model, already performing decently on a specified genome to be able to produce correct results for the different one.

The aforementioned approaches had enabled the development of numerous models to be trained and fitted on the genome banks. Since the size of the human genome surpasses the one of yeast dramatically, it contains a lot of unknown promoters, that govern expression and are yet to be unraveled. We might hope for the transfer learning models, trained on yeast genome, that possesses already known expression favoring motifs, to be able correctly to capture those of human. Taking into account the size of the randomly sampled promoter sequences, a vast space for model training is created.

2 Data Acquisition

The data, that we've been working on was gathered from Saccharomyces cerevisiae (Brewery yeast), by applying RNA-seq. To produce the dataset, the cells were uniformly sorted into 3 batches, each containing 6 bins. An exception was the scaffold library, situated into a non-uniform bin to account for the variance. The cell library is then processed by Flow cytometric screening (FACS) to sort the tags into the bins, corresponding to their level of expression.

Eventually, multiplexing indices had been added, which in turn resulted into 2x76 bp length of the library. The resulting number of sequences was equal to 31 million, which after discarding the repetitions produced about 8.5 million sequences in the library. As a result, ending up with sequences flanked by constant motifs on the ends with a length of sequences, that doesn't exceed 142 bp.

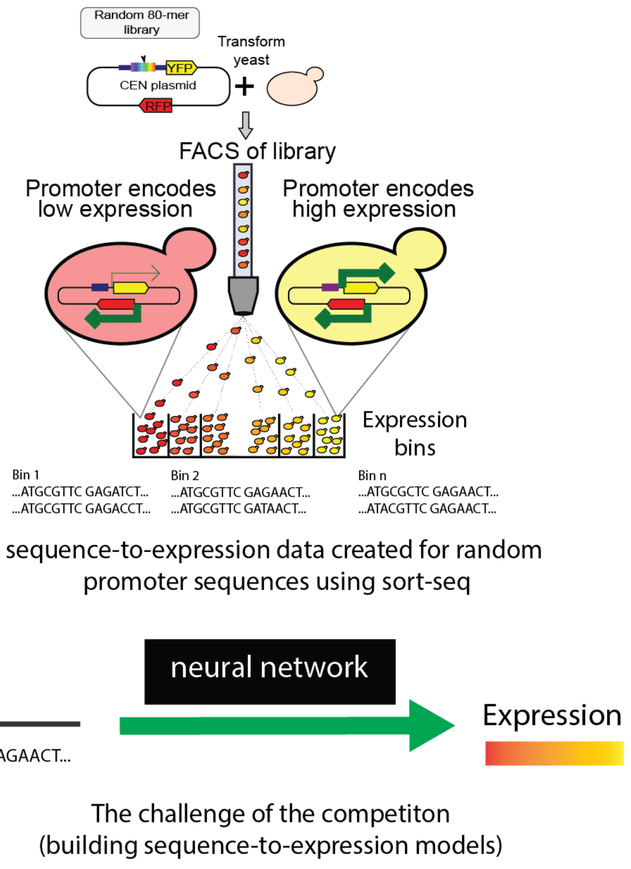


Figure 2.1: A concise depiction of the competition pipeline

3 Data Engineering and Preprocessing

We started preprocessing the data by cutting off the constant flanks, that had been attached to each of the primers. We also transformed the sequences into the form of one-hot encoded vectors. Following the intuition, that if the specified sequence is expressed at a certain level, then its reverse complimented counterpart is expressed at the very same level, which allows an efficient and natural way of data augmentation.

A natural way of expanding the features was to introduce certain biological features, that might convey information about the level of expression of a sequence. 'GC' content pertains to a kind of those. It is well known, that the ratio of these nucleotides present in the sequence plays important role in defining the way it is expressed. We also attempted to add melting temperature, which is also an important biological feature for the model.

The targets value, that is to be learned to regress against is defined as the log ratio of yellow fluorescent protein (YFP) expression values to Red Fluorescent Protein (RFP) expression values [1]. This value is highly concentrated around integer values, ranging between 0 and 18 (boundaries rounded). The skew of the curve is positive and the distribution demonstrates tails, that are heavier than those of the shifted normal 6.

To address an issue of the class-integer imbalance we applied a technique, called label smoothing, that aims to facilitate the training of the regression models by imposing additional noise on the training labels. Moreover, we were making use of weighted sampling to emphasize the importance of the tails.

Another family of models, that we applied during the competition included k-merization [3], which is in essence count vectorization over the subsequences, i.e. k-mers, that constitute a bigger sequence. To accomplish such a feature extraction process, we were sliding over the initial parent sequence with a window with a specified width and stride. Afterwards, we used the obtained features in the shallow models.

Finally, we also reformulated the problem into a classification, which specifically considered discretizing the output data over discrete bins, which makes sense taking into account the concentration of values around integer points. We used this approach to construct several experimental models, which will be discussed in more detail 4 and 5.

4 Methods

Deep Learning Approach

To get a deeper understanding of how the obtained sequences are organized, we tried a variety of deep learning architectures as well as classical ensemble methods. In our experiments, we found out deep learning architectures are more successful in addressing the changes on expression levels across the sequences. Although the models are validated on heavy tailed distribution with possible outlier expression levels, the deep learning architectures in our analysis demonstrate superior performance in identifying the extremes expression level values as well as their corresponding motifs. Moreover, convolutional layers are less prone to overfitting unlike ensemble models with weak learners. This manifests that complex models and architectures are indeed capable of carrying out a robust analysis with sequential data composed of genes.

We utilized several architectures in our methodology but we structured the deep learning part of our analysis in a way that allowed us to focus on three architectures at a time. Primarily, those architectures were residual neural networks (ResNet) [4], convolutional neural networks (CNN) [5] and convolutional neural networks with residual connections. Since we had a lot of architectural changes in the model development process, it is intuitive to do a deep dive analysis on the architecture of the best model while fostering a discussion on possible issues with the rest of the models.

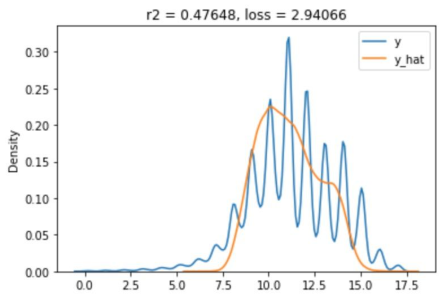
Considering all of our experiments, the best model that we acquired was a convolutional neural network with residual connections. Starting with a shallow architecture with 2 convolutional layers and kernel sizes of 21 and 5, the network didn't manage to overcome $0.5R^2$. The shallow network was unable to capture the extremes of the distribution but rather demonstrated a regression to the mean effect, with most of the density centered at around 10.5-13.5 expression level-wise (see 6.1 for further discussion).

Another baseline model to be taken into consideration was the classification based model, which aimed to correctly classify expression rate into integer bin, using an architecture of several convolutions with the number of channels equal to 40 in each step (note that reference motifs were provided in a number of 40) and followed by a fully connected layer mapping to 19 dimensional space. Unfortunately, this approach failed to beat the regression model, at least in such a minimalistic representation.

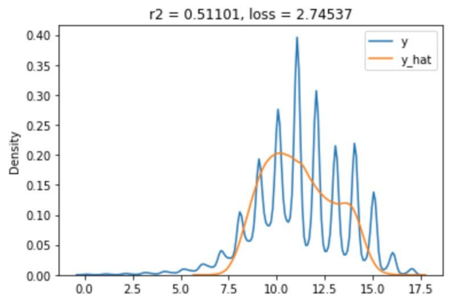
To overcome this issue, the number of layers was increased to 7 with channel size 64 and the previous kernel size at the last layer, 5, was applied 4 more times on each of the layers to capture as much regulatory information as possible. This specific number of kernels can be made to be more lucid if we refer to the primers already known for yeast, their lengths swing up to the length of about 25.

Following the gradual increase in the complexity of the model as well as the execution time, we downsampled the feature maps using max pooling to reduce size 2 times after each residual connection, followed by global max pooling and the regression head of the network to predict the expression levels. Moreover, batch normalization was used in each convolutional layer to ensure stable performance and faster training while potentially preventing poor parameter initialization. Finally, parametric relu (PRELU) [6] was chosen as the activation function in convolutional layers whereas softplus [7] is used in the final layer to predict the corresponding reporter gene expression level. During training, we were also utilizing stochastic weight averaging combined with cosine annealing technique [8] in order to prevent models from overfitting which occurs quite often for local optimas of neural networks. Therefore, after some training time, we decided to increase the number of channels to capture more primers, present in sequential data, this albeit didn't seem to contribute too much to the ultimate performance.

We used MSE loss with weight decay of 10^{-3} to train the model with several types of optimizers. First, we applied a quick adaptive learning rate optimizer, weighted Adam, initilized with 10^{-3} learning rate equipped with ReduceOnPlateau learning rate scheduler, tracking the validation metric. After 5 epochs, when the optima of about 51.2 was localized, we used SGD to help model generalize better while reducing learning rate by a factor of 0.1. This approach helped us to elevate the ultimate score.

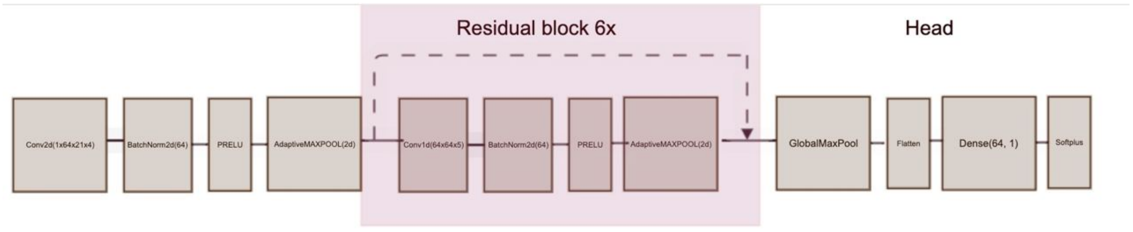


(a) Baseline CNN



(b) ResNet before post-training

Figure 4.1: Expression level density comparison



(a) Baseline CNN

Figure 4.2: ResNet Architecture

Classical Approach

The number of methods applicable to the analysis of sequential data is abundant in machine learning literature. This broad spectrum of algorithms traditionally encapsulates methods from two different, yet interactive disciplines; deep learning and classical approaches. In this section of our analysis, we are going to focus on the latter to provide an overview for our classical analysis, more specifically Light Gradient Boosting Machine, i.e. LightGBM [9]. This is further elaborated in 6 to facilitate a discussion on the performance and robustness of the convolutional neural networks and gradient boosting regressors.

Boosting is an ensemble method that is enforced on each individual model of the ensemble to gradually increase performance. This broad, vague definition can be broken down to the concept of repetitive calculation of a gradient from each individual member, i.e. weak learner, of the ensemble to decrease residual error from the preceding models [10].

Considering our goal was to predict expression values of the DNA sequences of the reporter gene, boosting is a very valuable tool at hand analyzing the downstream or upstream bps, considering the sequences are not randomly shuffled and we have positional information about the nucleotide. Positional information matters because the iterative calculation of the gradient is susceptible to the change of order in sequence fragments. Unfortunately, there is no trivial solution to reveal positional information and discover the motifs of the sequence. Instead, a scientific workaround to alleviate this problem would be to grant the algorithm a sliding window indirectly by processing the sub-sequences of the sequence, i.e. k-merization 3, to reveal motifs to the collection of trees.

Although prepossessing techniques that we discuss on 3 are useful in addressing positional information, it is common knowledge that machine learning algorithms, whether novel or classical, tend to capture noise when large window sizes are enforced on model iterations which will lead to an increase of bias for individual member of the ensemble [11]. Considering the testing data is purely random which was constructed by random placement of a DNA base across an 80 bp long sequence, the amount of uncertainty introduced to the model is proportional to length of test sequences. Therefore, despite experimenting with both raw and label-encoded sequences, it is intuitive to make raw sequence inputs the center of attention to provide an overall better understanding of the scientific approach to the correlation between weak learner residuals and accumulated error rate.

Having provided a background for our classical approach to scope of the problem, it is vital to discuss techniques that are employed on hyperparameter tuning to combat overfitting and robustness of the ensemble to ensure stable performance. Even though boosting allows the model to regress for low-bias ensemble outputs, it can be greatly influenced by a different set of parameters, for instance, maximum number of leaves on a tree. We discuss the influence of such hyperparameters in 6.

Other factors that make it necessary to tune hyperparameters are learning rate employed in computation of the gradient along with bagging [12] and feature fractions in the ensemble. Primarily, bagging serves for insurance of lower residual error by sampling with replacement to establish a smoother, more uniform distribution of the nucleotides. Feature fraction on the other hand, is for reducing the tendency of the weak learner to capture relative information of possible motifs very fast and prevent overfitting to training data quickly before hitting maximum number of boosting rounds during execution. Finally, an exponentially decaying learning rate [13] has been proven to be beneficial in combatting overfitting problems in gradient boosted trees, in contribution to a correctly set feature fraction and bagging fraction in the ensemble. In 5, we outline the specifics of our configuration further and discuss the results of the classical approach in 6.

5 Experiments

In order to increase the overall score of the model, we were applying numerous techniques, many of which didn't actually contribute much to the final performance of the model. In this section we will list some of them.

One of the core ideas, that we had been pursuing refers to transformer [14] approaches, that could have produced decent embeddings, incorporating valuable information, that would drive model to higher results. Among the embeddings, that we have been considering one might outline Enformer [15] model, which however required much longer input sequences to perform encoding and was therefore discarded. Another embedding model was min gpt (a simple version of GPT-2 [16]), which application was limited to classification setting, described 3. The advantages of this setting was immediate training and extraction of embeddings, which nevertheless was outperformed by the final model.

We also tried to introduce models, distributed under Kipoi project, AttentiveChrome and Xpresso [17] to name a few. The problem that we ran into while trying to utilize these models was that most of these models required sufficiently longer inputs. To omit the possible noise introduced by dummy padding of sequences to the required size, we have decided to refrain from using such approaches.

Other approaches that we were trying to integrate could be in general referred to as dimensionality reduction, where the Autoencoder model was the center of our attention. The downside with this setting was that the dataset was rather artificial and reconstructing a random distribution was not a sensible approach. Therefore, we didn't try to insert the module into the final pipeline. Among Autoencoders [18] that were explored, we tried to incorporate a Sparse-AutoEncoder [19] to our analysis, trained with L2 loss. The model could not outperform the best implementation.

Finally, diverging from deep learning, as outlined in 4.2, we explored whether classical machine learning approaches such as Random Forest Regressors [20] and Light Gradient Boosting Machine could be successful in identifying patterns. While these are completely different approaches (see discussion on bagging vs boosting in 4), we wanted to see whether it is more crucial to investigate classical models with less bias or less variance in order to predict promoter expressions. While constructing the basis of the models, we extended upon k-merization approach mentioned in 3 with frequency-based tokenization which is essentially storing the count of each k-mer but we could not benefit from this. Furthermore, we tried padding to different label-encoded values to see whether the density of expression levels would be disrupted by the value selection since this effects the split of each individual tree. Splits that consistently extend along a single branch of a tree highly inaccurate and counter-intuitive due to the same selection and split at every decision node. Lastly, we tried Huber loss function instead of L2 loss regarding the classical models. However, this widely degraded the performance of the model (see discussion on Huber loss in 6).

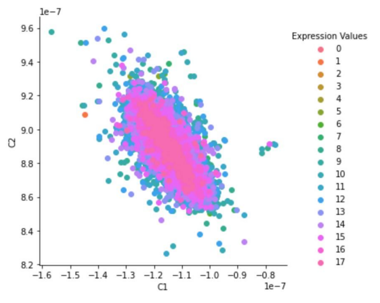
6 Results

To compare the models we are using r2 metric, the following table show Training and validation results as follows:

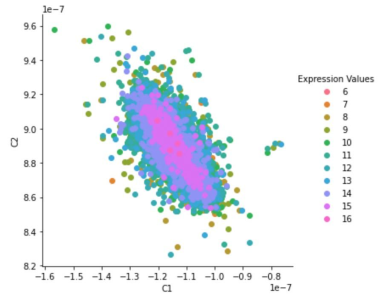
BaselineClassificationCNN	BaselineCNN	LGBM	ResNet
44	47.5	40	55.0
43.2	47.3	37	53.7

In our experiments, ResNet eventually outperformed other models. A potential explanation to this is the following. In case the experiment is subject to shallow networks and the receptive field of the final layer is not high enough to capture all the information, a natural but inaccurate solution is to increase the field. This makes the training process more complicated as larger kernels are more prone to capturing side noise in data. ResNets on the contrary are more stable in learning and allow increasing channels without hindering the overall gradient backward propagation as well as allowing to undo undesired effect of the convolution in case it detracts from the result.

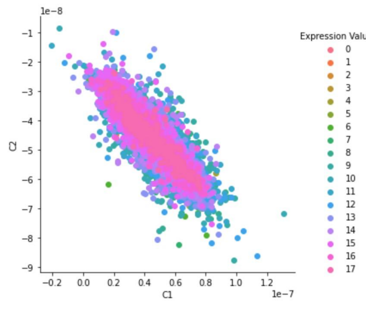
To get a deeper intuition of what is happening behind the stage of the model, we accomplished analysis of the outputs of the model by applying TSNE to train and validation data and comparing the labelling in the projection domain (see 6.1).



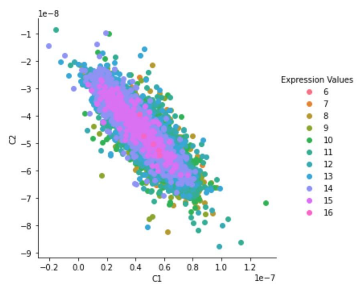
(a) train actual



(b) train fitted



(c) validation actual



(d) validation fitted

Figure 6.1: TSNE plots of train/validation splits for ResNet

It is visible, that the model is not capturing more abundant values, which, as has been stated before signifies the heavy tails of the distribution and inability of the model to capture them accurately. To address the distributions in a more vivid manner, let us glance at the kernel density plots of the distributions.

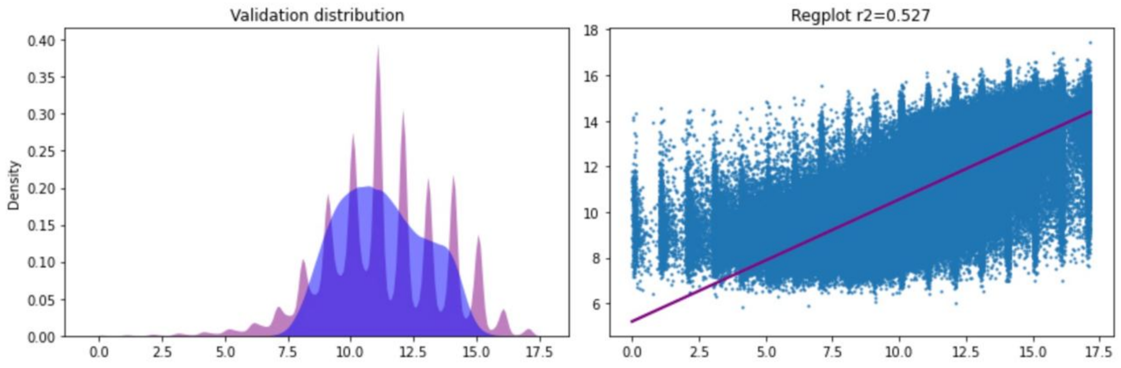


Figure 6.2: Kernel density plots of validation split for best ResNet

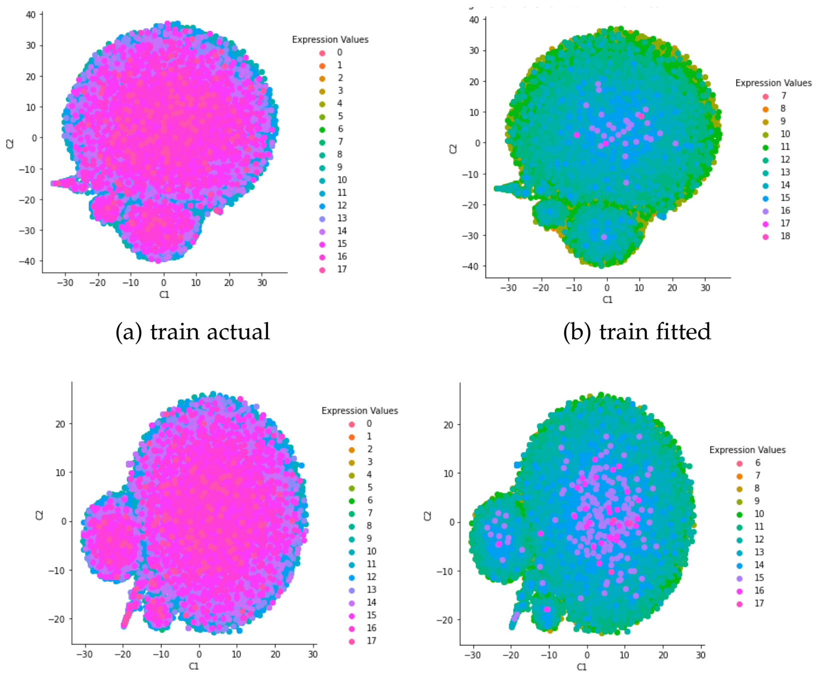
Looking back at the density plot, ResNet definitely accounts for a subsample set that lie close to the tails of the distribution, however, in general, the network still fails to identify the expression spikes concentrated across all parts of the distribution.

On the other hand, considering the classical approaches we included in our work, there are several findings that we would like to discuss. Since there is a significant prediction performance difference between best ResNet and LightGBM, a need to discuss potential issues of the classical methods arises. First of all, we found out the number of leaves on a tree is by far the most important parameter to tune. Since LightGBM uses a leaf-wise algorithm [21], the parameter heavily influences the complexity of the ensemble. Provided that it is insufficiently tuned, the variance in the model quickly becomes explosive, caused by the sudden increase in the norm of the gradient at each iteration. This is especially valid in sequential data that is specific to biological domains since sequence lengths can be arbitrarily long.

Moreover, as briefly mentioned in 5, despite k-merization inputs to the ensemble to capture positional information, Random Forest regressor demonstrated a tendency to predict corresponding expressions into almost randomized output values. One potential reason behind this is that k-merization and frequncy-based tokenization don't take into account the relative position of nucleotides and the resulting counts are simply interpreted as arbitrary counts if the dataset is indeed random enough. However, provided non-random sequences, they definitely convey information about possible motifs as our analysis was mostly disrupted by the sparsity of the sequence count matrix, rather than the randomness itself. We can make this claim confidently because we assessed the performance of another ensemble, LightGBM with the exact padding method that we applied to the sparse count matrix.

With LightGBM, since we found out count vectorization was not as successful as we'd hoped, we focused on tuning number of maximum leaves and number of boosting iterations the dataset was exposed to. Maximum number of leaves when increased to 2047 (excluding the root, $2^{depth} - 1$) greatly increased the model success on training to 40 percent but the discrepancy with validation grew fairly larger in comparison, with a validation increase of 0.021 after 600 boosting rounds.

Finally, we experimented with Huber loss and L2 loss and found out the analysis is indeed disrupted by the padding that we had to do for consistency of the split criterion for the tree at each node split. This is especially clear in TSNE plots for the most successful LightGBM regressor that we could build (see 6.3). However, since the amount of padding was drastic, this contributed to the amount of noise in the dataset to a greater extent. Since Huber loss accounts for a mixture of absolute error and mean squared error, in normal circumstances, this would not be the case and there would be room for improvement. Since the distribution alone was disrupted, Huber loss failed to contribute to our analysis.



(a) train actual

(b) train fitted

(c) validation actual

(d) validation fitted

Figure 6.3: TSNE plots of train/validation splits for LightGBM

7 Conclusion

In conclusion, we tried to predict promoter expressions by acquiring techniques from two diverse but interacting research disciplines; deep learning and traditional ensemble learning methods. Both of these methods had their advantages and disadvantages, with deep learning surpassing traditional ensemble methods in various performance metrics.

The deep learning methods primarily demonstrated the fine tuning of learning rate with weighted optimizers along with cross switching to different optimizers under a reliable scheduler helps with robustness as well as decreasing overfitting proneness. Furthermore, the advantage to seek out residual connections with batch normalization to impose regularization on the network enables stable performance and helps with robustness as well as enabling faster computation. The key to the ResNet's success can be attributed to these factors as compared to classical approaches.

Classical approaches on the other hand have been extensively researched, specifically two ensemble methods random forest regressor and LightGBM to foster a discussion on effects of boosting and bagging. The most significant finding of this section was the fact that regression to promoter expression values valued individual bias decrease in weak learners as in LightGBM as opposed to average reduction in variance in Random Forest regressors. This is intuitive because individual bias reduction is more tunable across boosting rounds with several parameters as number of leaves and maximum depth of the tree (which are helpful in accounting for the gradient) whereas the overall variance of the forest ensemble relies on the number of votes of each individual tree. Hence, we prove that this consensus mechanism does not work and reducing the bias to produce the output prediction matters much more for predicting promoter expressions.

Bibliography

[1] C. de Boer, E. Vaishnav, R. Sadeh, E. Abeyta, N. Friedman, and A. Regev. "Deciphering eukaryotic gene-regulatory logic with 100 million random promoters". In: *Nature Biotechnology* 38 (Jan. 2020). DOI: 10.1038/s41587-019-0315-8.

[2] E. Vaishnav, C. de Boer, J. Molinet, M. Yassour, L. Fan, X. Adiconis, D. Thompson, J. Levin, F. Cubillos, and A. Regev. "The evolution, evolvability and engineering of gene regulatory DNA". In: *Nature* 603 (Mar. 2022). DOI: 10.1038/s41586-022-04506-6.

[3] N. Bhandari, S. Khare, R. Walambe, and K. Kotecha. "Comparison of machine learning and deep learning techniques in promoter prediction across diverse species". In: *PeerJ Computer Science* 7 (Feb. 2021), e365. DOI: 10.7717/peerj-cs.365.

[4] K. He, X. Zhang, S. Ren, and J. Sun. *Deep Residual Learning for Image Recognition*. 2015. DOI: 10.48550/ARXIV.1512.03385. URL: https://arxiv.org/abs/1512.03385.

[5] K. O'Shea and R. Nash. *An Introduction to Convolutional Neural Networks*. 2015. DOI: 10.48550/ARXIV.1511.08458. URL: https://arxiv.org/abs/1511.08458.

[6] A. F. Agarap. *Deep Learning using Rectified Linear Units (ReLU)*. 2018. DOI: 10.48550/ARXIV.1803.08375. URL: https://arxiv.org/abs/1803.08375.

[7] H. Zheng, Z. Yang, W.-J. Liu, J. Liang, and Y. Li. "Improving deep neural networks using softplus units". In: July 2015, pp. 1–4. DOI: 10.1109/IJCNN.2015.7280459.

[8] I. Loshchilov and F. Hutter. *SGDR: Stochastic Gradient Descent with Warm Restarts*. 2016. DOI: 10.48550/ARXIV.1608.03983. URL: https://arxiv.org/abs/1608.03983.

[9] R. P. Sheridan, A. Liaw, and M. Tudor. *Light Gradient Boosting Machine as a Regression Method for Quantitative Structure-Activity Relationships*. 2021. DOI: 10.48550/ARXIV.2105.08626. URL: https://arxiv.org/abs/2105.08626.

[10] R. Sheridan, A. Liaw, and M. Tudor. "Light Gradient Boosting Machine as a Regression Method for Quantitative Structure-Activity Relationships". In: (Apr. 2021).

[11] J. a. G, F. Sue, E. Johnson, M. Niranjan, and A. Gee. "Global Optimisation Of Neural Network Models Via Sequential Sampling-Importance Resampling". In: (Mar. 1999).

[12] P. Bühlmann and B. Yu. "Analyzing Bagging". In: *Annals of Statistics* 30 (Aug. 2002). DOI: 10.1214/aos/1031689014.

[13] S. Park, S.-g. Lee, H. Nam, and S. Yoon. "An Efficient Approach to Boosting Performance of Deep Spiking Network Training". In: Dec. 2016.

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. *Attention Is All You Need*. 2017. DOI: 10.48550/ARXIV.1706.03762. URL: https://arxiv.org/abs/1706.03762.

[15] Ž. Avsec, V. Agarwal, D. Visentin, J. R. Ledsam, A. Grabska-Barwinska, K. R. Taylor, Y. Assael, J. Jumper, P. Kohli, and D. R. Kelley. "Effective gene expression prediction from sequence by integrating long-range interactions". In: *Nature Methods* 18.10 (Oct. 2021), pp. 1196–1203. ISSN: 1548-7105. DOI: 10.1038/s41592-021-01252-x. URL: https://doi.org/10.1038/s41592-021-01252-x.

[16] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. “Language Models are Unsupervised Multitask Learners”. In: (2019).

[17] V. Agarwal and J. Shendure. “Predicting mRNA Abundance Directly from Genomic Sequence Using Deep Convolutional Neural Networks”. In: *Cell Reports* 31 (May 2020), p. 107663. DOI: 10.1016/j.celrep.2020.107663.

[18] R. Xie, J. Wen, A. Quitadamo, J. Cheng, and X. Shi. “A deep auto-encoder model for gene expression prediction”. In: *BMC Genomics* 18.9 (Nov. 2017), p. 845. ISSN: 1471-2164. DOI: 10.1186/s12864-017-4226-0. URL: https://doi.org/10.1186/s12864-017-4226-0.

[19] A. Ng. “Sparse autoencoder”. In: (). URL: https://web.stanford.edu/class/cs294a/sparseAutoencoder.pdf.

[20] A. Liaw and M. Wiener. “Classification and Regression by RandomForest”. In: *Forest* 23 (Nov. 2001).

[21] H. Alshari, Y. Saleh, and A. Odabas. “Comparison of Gradient Boosting Decision Tree Algorithms for CPU Performance”. In: *Erciyes Tip Dergisi* (Apr. 2021), pp. 157–168.