Feature Sparsity in Protein Sequence Design using LassoNet

Max Guo
(idea developed in partnership with William Zhang)

1 Introduction

In the modern age, the development of sophisticated computational protein sequence design processes, in concurrence with efficient experimentation techniques, allows for the rapid engineering and testing of protein sequences [1]. The design of novel protein folds, known as *de novo* problems, tends to be of a much grander challenge than that of redesigning existing protein sequences [2]. Within the latter category, one application of significance is that of finding *diverse* mutations of an existing protein sequence that retain the functionality of the original protein. Generating diverse mutations is not automatic; existing fixed backbone computational techniques for redesigning proteins often output sequences that remain similar to the original sequence [3].

The importance of diversity within proteins is exemplified through the example of the adeno-associated virus (AAV) capsid. AAV capsids are used in gene delivery and have been integrated into gene therapies that have been approved for human use [4]. However, it has been known for several decades that humans exposed to AAV will develop antibodies to certain AAV sequences, thus necessitating the design of new AAV sequences [5, 6]. Moreover, designed proteins that share too many similarities with natural AAV sequences are often unable to surmount the immunity of patients with previous AAV exposure [1]. Diversity and viable mutants that deviate significantly from the natural AAV sequences are thus a necessary component for effective treatment of patients with antibodies.

Machine learning is an indispensable tool for protein sequence design. Computationally, the problem can be interpreted as a search process in a high dimensional space, and machine learning models can be used to guide the search process [7]. However, if the machine learning models are utilized as black-box prediction models, these search processes may not necessarily be well-informed. One potential way to inform these search processes may be to apply feature selection, which is a technique that can help with the interpretability and generalizability of a model. The LassoNet is an example of a nonlinear neural network model which achieves feature sparsity [8]. Our proposal seeks to utilize the LassoNet model design in order to make existing AAV sequence design processes that rely on machine learning models more efficient and principled.

2 Related Work

Our project application extends the work of Bryant et al., in which they identify diverse variations of an AAV wildtype protein sequence via directed evolution [9, 10], an iterative mutation process that is guided by artificial selection via trained machine learning models [1]. However, their iterative mutation process may be inefficient, as it fails to consider the possibility that certain mutations may be more important for viability. Our proposal seeks to use feature sparsity to improve this process.

Our proposal utilizes LassoNet, as introduced by Lemhadri et al., and variations of the model [8]. LassoNet combines a fully connected neural network with a skip layer to ensure feature sparsity. A feature only participates in the neural network if it is active in the skip layer, allowing for both nonlinear predictive ability along with regularization. Existing theoretical methods for feature sparsity include methods that select features independently of predictor, methods that evaluate features based on predictive power, and embedded methods like the linear Lasso method [11], which efficiently combine feature selection and learning [8]. LassoNet extends Lasso by incorporating a neural network, thus allowing to learn nonlinear functions.

We provide novelty in the application of LassoNet in two ways. Firstly, the LassoNet as introduced only supports a fully connected neural network with a skip layer. We propose to extend the LassoNet to other architectures, and doing so in a fitting manner is nontrivial[1]. Secondly, the quality of the features selected by LassoNet is usually benchmarked by the performance of classifiers trained on the features. Direct applications of the LassoNet model include predicting disease risk from genome-wide association studies [12] and predicting electricity consumption [13], and applications of feature sparsity in general are even more prevalent and diverse. However, our proposal intends to use feature sparsity as part of the

[1] As noted by Yanke's question, which will be addressed in Section 3.4

process for *generating* variations of protein sequences rather than strictly for prediction, as these related applications of LassoNet and feature sparsity do.

3 Proposal

Our overall methodology is very similar to that of Bryant et al. in [1], though the ideas can be applied more broadly. Please reference Figure 1 for an overview of the paper methodology, taken from [1]. Section 3.1 and Section 3.2 describe the paper's methodology, whereas Section 3.3 and Section 3.4 discuss our proposed extensions[2]. Section 3.5 discusses our preliminary experiments. We assume knowledge of the LassoNet model for brevity ([8]).

3.1 Data

In [1], we have a wildtype AAV protein sequence of length 28, which we denote as WT. Each mutation is represented as an array of size 58×20 with at most one 1 in each row. There are 20 possible amino acids, so each row is either a one-hot encoded vector or a vector of zeros. A single mutation is either an insertion between WT amino acids or a substitution of a WT amino acid. There are 29 possible positions for insertion and 28 possible substitutions, giving us 58 possible positions (the first row is always all zeros), alternating substitution, insertion, substitution, ..., insertion.

There are 3 primary datasets that are used in training: $C_1 + R_2$, $C_1 + R_{10}$, $R_{10} + A_{39}$. C_1 denotes the complete set of all single mutations. There are 19 possible substitutions at each WT position, contributing $28 \cdot 19$ mutations, and 20 possible substitutions at each insertion, contributing $29 \cdot 20$ mutations, for a total of 1112 sequences in C_1. R_2 denotes a random subset of sequences of size 1756, each with 2 mutations. R_{10} denotes a random subset of sequences of size 7908, each with between 2 and 10 mutations. Finally, A_{39} has 56372 sequences, designed by an additive model.

3.2 Selecting and Designing Sequences

Bryant et al. apply 3 machine learning models: Logistic Regression, CNN, and RNN to select and design new sequences. For each dataset, they train each machine learning model 11 times with independently initialized weights to form an ensemble. Now for each mutation distance 5 to 25 from WT, they sample 100 million sequences at those mutation distances. For each of the 9 dataset-architecture combinations, they rank the 1000 highest-scoring sequences. These are the *model-selected sequences*. Due to our computational limitations and other factors[3], our implementation started with 1000 sequences and ranked the top 50 sequences as our model-selected sequences.

To obtain the *model-designed sequences*, we iteratively apply random single mutations to the currently designed sequences (initializing with the model-selected sequences), then use our ensembles to choose the sequences with the highest viability probability for the next iteration. Finally, Bryant et al. experimentally tested the viabilities of the top scoring sequences. The results in the paper are visualized in Figure 2.

3.3 Informed Directed Evolution

Our first extension is to incorporate feature sparsity to inform our directed evolution process in designing sequences. We followed the paper methodology and applied a LassoNet model with one hidden layer of size 1 to make it as similar to a logistic regression model as possible, and we compare it with an `scikit-learn` implementation of logistic regression. After training an ensemble of LassoNet models, we visualized the feature sparsity and determined a set of features (we denote I) that the LassoNet model considered important for its predictions. Let A be the set of all features, where $|A| = 58 \times 20 = 1160$. Then $A \setminus I$ is the set of features whose weights in the LassoNet model have shrunk to 0 from the L1 regularization. Then our proposed directed evolution process would be to iteratively find the most viable sequences among the starting sequences according to our model, then randomly change positions in I or $A \setminus I$ depending on the viability of our sequence. In our proposal, the viability of our sequence could refer to the *true* viability (if we had resources to experimentally verify our viabilities in the directed evolution process), or the viability according to our model (which is our implementation). This decision tree is illustrated in Figure 3.

Our motivation for this decision tree is the following. If a sequence is currently viable,

[2]We unfortunately did not have enough time to extend LassoNet to an RNN architecture.

[3]The code provided by Bryant et al. could not be run as it was implemented in a version of TensorFlow that is no longer supported, so we had to rewrite almost everything ourselves.

then changing positions in $A \setminus I$ would likely maintain the sequence's viability, as positions in $A \setminus I$ are unimportant for viability prediction according to the LassoNet model. If a sequence is currently not viable, then mutating the positions in I are more likely to change the viability prediction than positions in $A \setminus I$.

3.4 Extending LassoNet to CNN

The LassoNet architecture for fully-connected feedforward neural networks is not immediately applicable to all architectures. For extending the LassoNet to CNN, Lemhadri et al. mention in their original paper that they could apply the convolutional layers first, then take the resulting learned features and apply LassoNet to them. However, this process would not allow us to isolate individual mutations that the LassoNet uses, which is required for our procedure in Section 3.3. Thus, we applied a variant of the LassoNet that still enforces feature sparsity at the mutation level.

For each of the 58 mutation positions i (here we applied feature sparsity at the position level, and not at the individual feature level), our CNN portion of the architecture multiplies the row vector at that position by a trainable weight W_i and then applies a normal CNN, and our skip layer portion multiplies θ_i by another weight U_{a_i}, where a_i is the one-hot amino acid at row i, and U_j for $1 \leq j \leq 20$ are additional trainable weights of the model. Then we enforce the constraint that $|W_i| < M|\theta_i|$ for each $1 \leq i \leq 58$, specifying M beforehand.

These specifications allow the CNN LassoNet to ensure feature sparsity among the positions of the sequence by reducing the weight W_i of a position if it is not used within the skip layer ($|\theta_i|$ is small).

3.5 Preliminary Implementation and Results

We implemented our informed directed evolution method on our three datasets for both logistic regression and the simple LassoNet, and the results we obtained show some promise. We trained a synthetic model (a baseline CNN which had high accuracy) on data not seen by the model as our ground truth, and we evaluated model proposed sequences using the synthetic model with both the paper's iterative mutation method and our informed search. In a majority of the dataset and model combinations, the informed search generated larger proportions of viable sequences. This is shown in Figure 4.

We had some optimization issues with the CNN LassoNet and could not resolve the issues in time, so the poor performance shown in Figure 4 should not be representative of its actual performance. We also visualized the sparsity conditions from our simple LassoNet and our CNN LassoNet, and they both show that the latter half of the sequence seems less important to viability, for both models (Figure 5). It is promising that these two models generally agree on the importance of various features. It also appears to be the case that the simple LassoNet requires many features in order to do accurate prediction (Figure 6).

4 Future Directions

The most direct future direction we would like to pursue is that of applying LassoNet to different neural network architectures other than feedforward networks and CNNs, such as RNNs. Even our CNN implementation is a bit contrived for this particular application, and may not be the best way of obtaining original feature sparsity. This is especially important for our application area of protein sequences, in which specialized neural network architectures are often required [14].

Another future direction would be to extend the decision tree made in Figure 3 to not only consider the viabilities but also the mutations that are already within the protein sequences. For example, hypothetically, if a feature which LassoNet considers important is such that the presence of the feature positively contributes to the viability of the model, then our decision tree may do better to mutate this position only if it's not already present. Moreover, LassoNet uses different weights on each feature, and we do not use this information to inform our mutation strategy. It seems reasonable that features with different weights may have different importances in determining viability. The best mutation strategy given sparse features and how LassoNet uses them is an interesting research question worth exploring further.

Finally, we ask more generally: given any approximation of a fitness landscape for some biological sequence, how can we effectively guide directed evolution given feature sparsity or other information from machine learning models? Answering this question can potentially make any search for new biological sequences that satisfy desirable properties more efficient.

References

[1] D. H. Bryant, A. Bashir, S. Sinai, N. K. Jain, P. J. Ogden, P. F. Riley, G. M. Church, L. J. Colwell, and E. D. Kelsic, "Deep diversification of an aav capsid protein by machine learning," *Nature Biotechnology*, vol. 39, no. 6, pp. 691–696, 2021.

[2] S. M. Lippow and B. Tidor, "Progress in computational protein design," *Current Opinion in Biotechnology*, vol. 18, no. 4, pp. 305–311, 2007. Protein technologies / Systems biology.

[3] G. Murphy, J. Mills, M. Miley, M. Machius, T. Szyperski, and B. Kuhlman, "Increasing sequence diversity with flexible backbone protein design: The complete redesign of a protein hydrophobic core," *Structure*, vol. 20, no. 6, pp. 1086–1096, 2012.

[4] C. E. Dunbar, K. A. High, J. K. Joung, D. B. Kohn, K. Ozawa, and M. Sadelain, "Gene therapy comes of age," *Science*, vol. 359, no. 6372, p. eaan4672, 2018.

[5] K. Erles, P. Sebökovà, and J. R. Schlehofer, "Update on the prevalence of serum antibodies (igg and igm) to adeno-associated virus (aav)," *Journal of medical virology*, vol. 59, no. 3, pp. 406–411, 1999.

[6] R. Calcedo, L. H. Vandenberghe, G. Gao, J. Lin, and J. M. Wilson, "Worldwide epidemiology of neutralizing antibodies to adeno-associated viruses," *The Journal of infectious diseases*, vol. 199, no. 3, pp. 381–390, 2009.

[7] S. Sinai and E. D. Kelsic, "A primer on model-guided exploration of fitness landscapes for biological sequence design," *arXiv preprint arXiv:2010.10614*, 2020.

[8] I. Lemhadri, F. Ruan, and R. Tibshirani, "Lassonet: Neural networks with feature sparsity," in *International Conference on Artificial Intelligence and Statistics*, pp. 10–18, PMLR, 2021.

[9] M. S. Packer and D. R. Liu, "Methods for the directed evolution of proteins," *Nature Reviews Genetics*, vol. 16, no. 7, pp. 379–394, 2015.

[10] R. J. Fox, S. C. Davis, E. C. Mundorff, L. M. Newman, V. Gavrilovic, S. K. Ma, L. M. Chung, C. Ching, S. Tam, S. Muley, *et al.*, "Improving catalytic function by prosar-driven enzyme evolution," *Nature biotechnology*, vol. 25, no. 3, pp. 338–344, 2007.

[11] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[12] H. M. Sajwani and S. F. Feng, "Identifying snp associations and predicting disease risk from genome-wide association studies using lassonet," *bioRxiv*, 2021.

[13] X. Zhou, J. Wang, H. Wang, and J. Lin, "Panel semiparametric quantile regression neural network for electricity consumption forecasting," *Ecological Informatics*, vol. 67, p. 101489, 2022.

[14] J. Graves, J. Byerly, E. Priego, N. Makkapati, S. V. Parish, B. Medellin, and M. Berrondo, "A review of deep learning methods for antibodies," *Antibodies*, vol. 9, no. 2, 2020.

5 Appendix

5.1 Figures from Bryant et al.

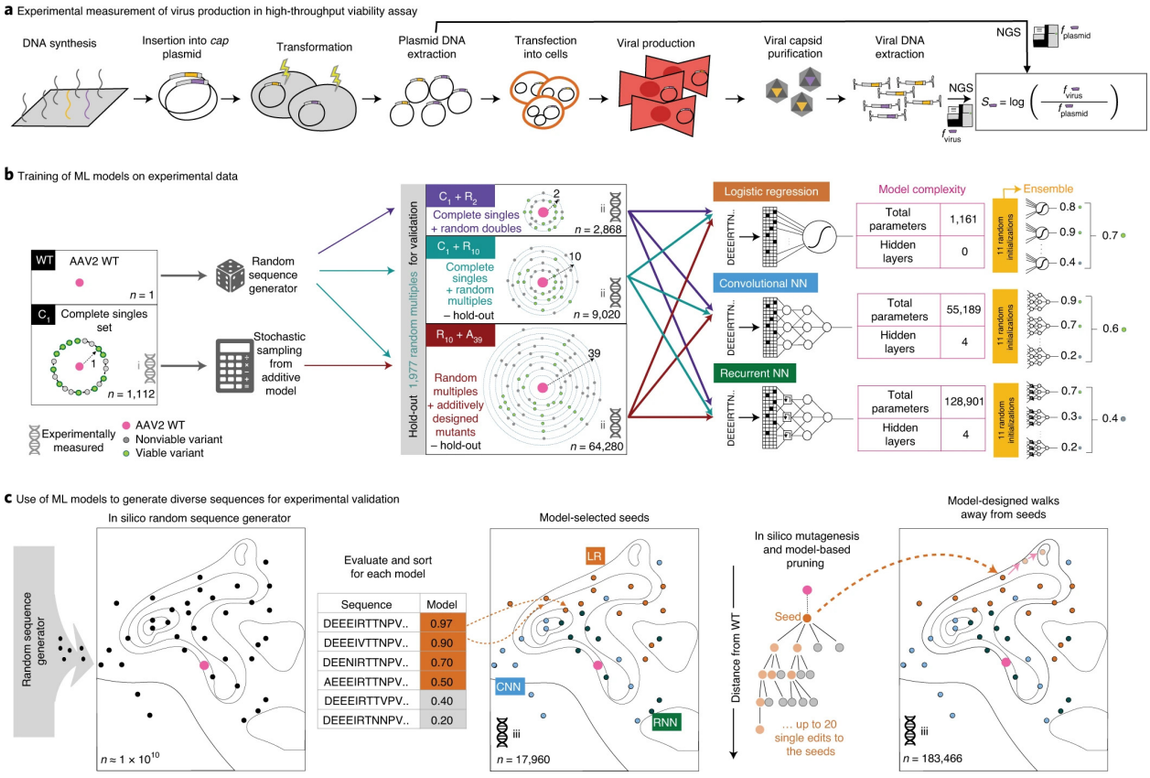


Figure 1: Methodology

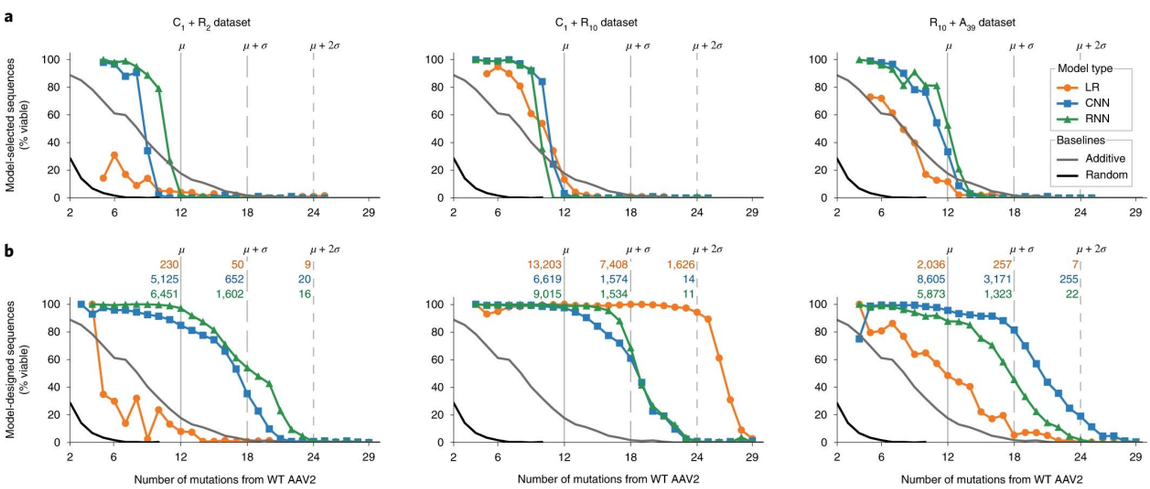


Figure 2: Results from Paper

5.2 Figures from our Proposed Method

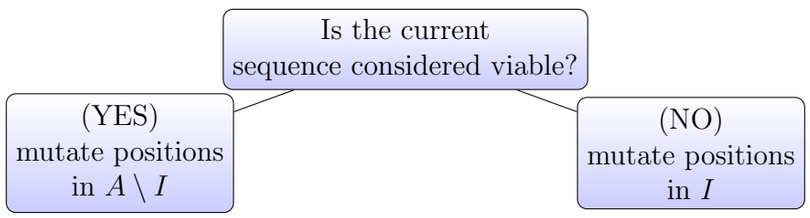
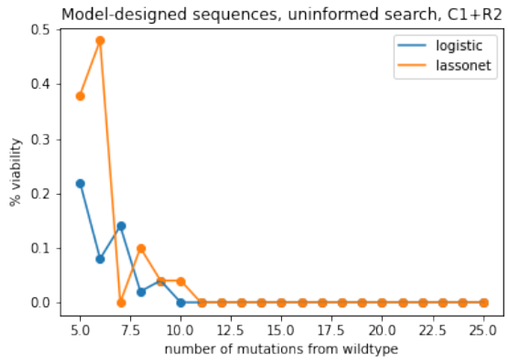
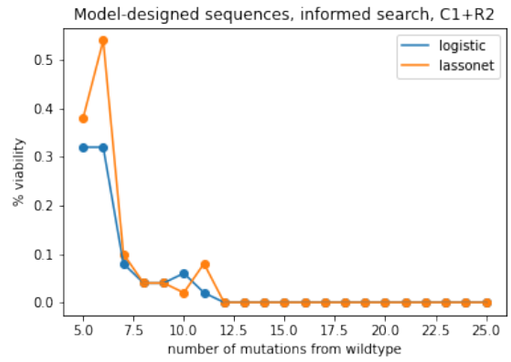


Figure 3: Proposed Directed Evolution Decision Tree

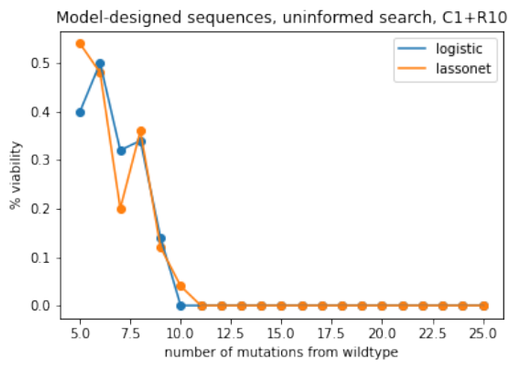
5.3 Figures from our Results



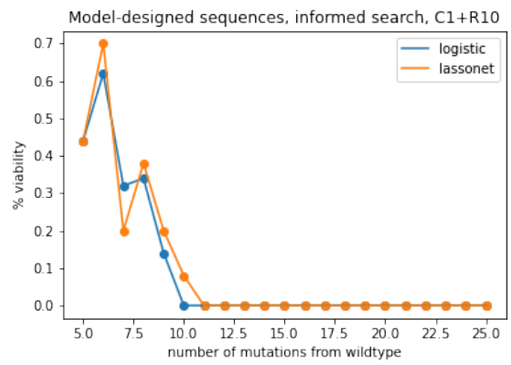
(a) $C_1 + R_2$, uninformed



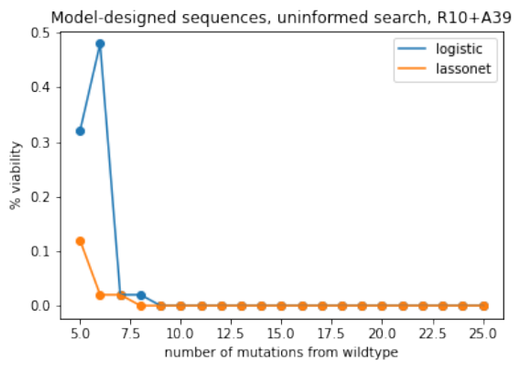
(b) $C_1 + R_2$, informed



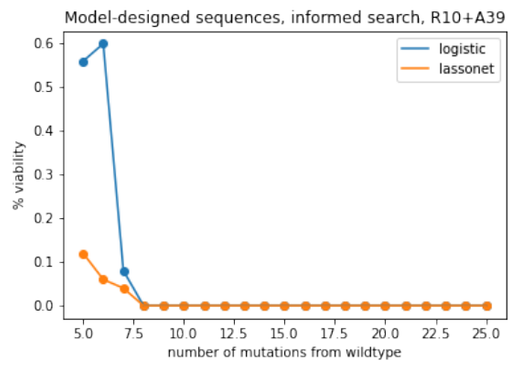
(c) $C_1 + R_{10}$, uninformed



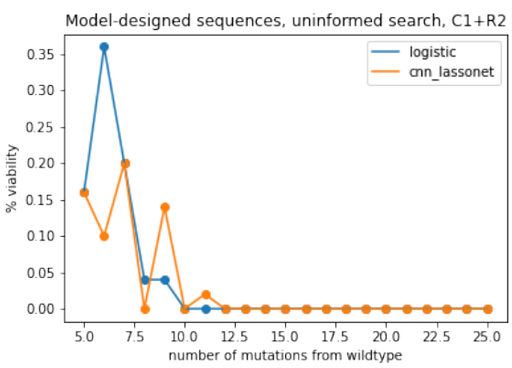
(d) $C_1 + R_{10}$, informed



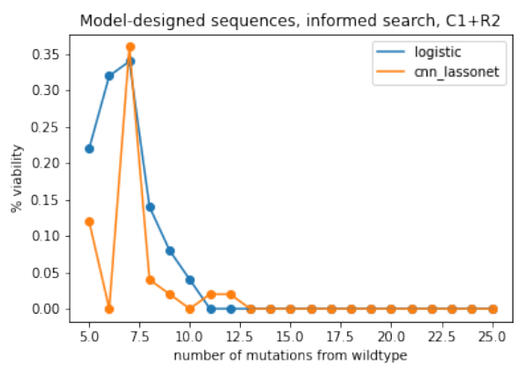
(c) $R_{10} + A_{39}$, uninformed



(d) $R_{10} + A_{39}$, informed



(a) $C_1 + R_2$, uninformed CNN LassoNet



(b) $C_1 + R_2$, informed CNN LassoNet

Figure 4: Comparing our performance between uninformed search to our informed search

Figure 5: Comparing Sparsities between the CNN LassoNet model (left) and the one hidden layer LassoNet model (right). The black pixels are not used for prediction, whereas the white pixels are. Notice how both models consider positions on the left to be more important in general

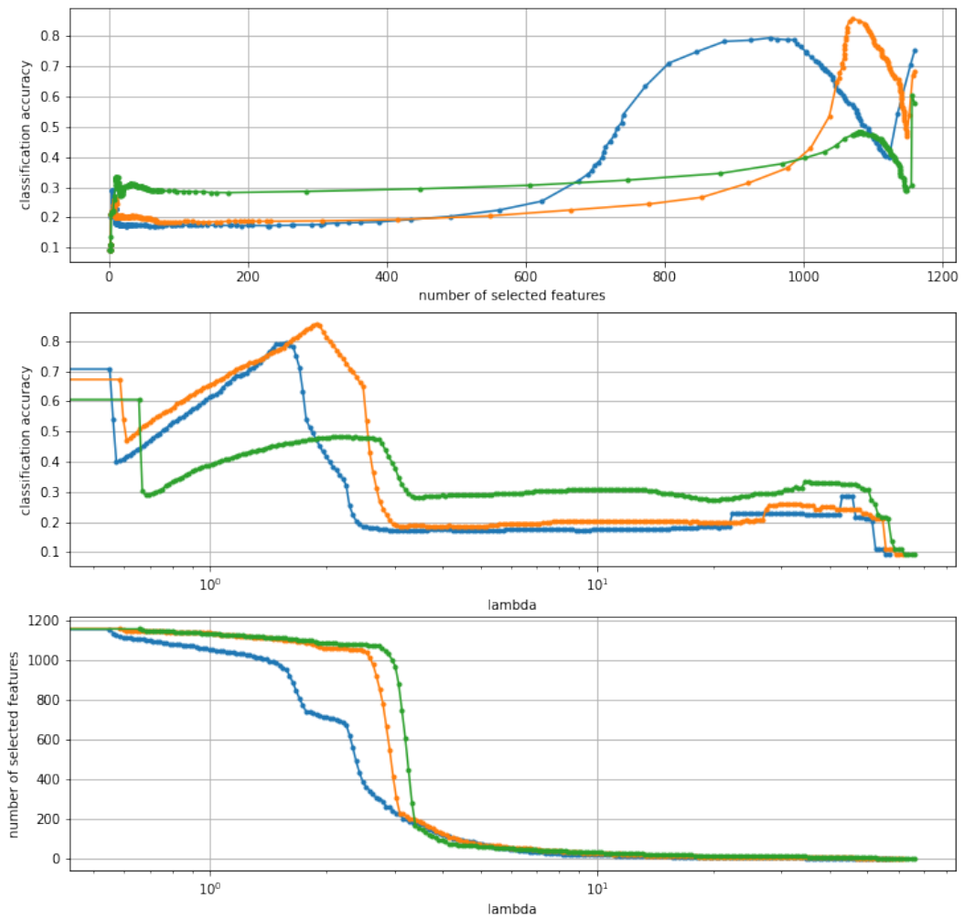


Figure 6: LassoNet's performance with varying number of selected features and λ values.

5.4 Small note on the bonus portion

I would like to express that I worked with William extensively in order to implement our new model and the procedure described in [1]. Our code is located at this link. I believe we both spent a lot of time trying to verify the legitimacy of our proposal (equivalent or more so than if we had just discussed the proposal and then worked individually toward the bonus). I personally wrote most of the code within the various `experimentation.ipynb` notebooks, while William helped write the LassoNet CNN architecture code. I hope this can be taken into consideration should our grades be borderline (and up to an additive factor that is less than the bonus quantity).