Efficient inference of nonlinear feature attributions with Scrambling Neural Networks

Johannes Linder
Paul G. Allen School of
Computer Science & Engineering
University of Washington
Seattle, WA 98195
jlinder2@cs.washington.edu

Alyssa La Fleur
Paul G. Allen School of
Computer Science & Engineering
University of Washington
Seattle, WA 98195
lafleur1@cs.washington.edu

Sreeram Kannan
Paul G. Allen School of
Computer Science & Engineering
University of Washington
Seattle, WA 98195

Zibo Chen
Institute for Protein Design
University of Washington
Seattle, WA 98195

Ajasja Ljubetič
Institute for Protein Design
University of Washington
Seattle, WA 98195

David Baker
Institute for Protein Design
University of Washington
Seattle, WA 98195

Georg Seelig*
Paul G. Allen School of
Computer Science & Engineering
University of Washington
Seattle, WA 98195

*Affiliated with Department of Electrical & Computer Engineering, University of Washington

2nd Conference on Machine Learning in Computational Biology (MLCB 2020), Vancouver, Canada.

Abstract

We present a feature attribution method based on generative modeling for interpreting biological sequence predictive neural networks – *Scrambler Neural Networks* – which learns the smallest set of features to either preserve or perturb such that downstream model predictions are maximally reconstructed or distorted.

1 Introduction

Neural networks are increasingly used for biological sequence prediction problems (Alipanahi et al., 2015; Avsec et al., 2019; Movva et al., 2019; Bogard et al., 2019; Sample et al., 2019; Eraslan et al., 2019; Jaganathan et al., 2019; Senior et al., 2020, Yang et al., 2020), and developing accurate interpretation methods for these models is critical. In particular, the task of *attributing* a given prediction to a set of input features can guide scientific discovery. For example, researchers have uncovered rules for DNA and RNA biology (Bogard et al., 2019; Kelley et al., 2019; Zeng et al., 2020), chromatin regulation (Zhou et al., 2015; Kelley et al., 2016; Zeng et al., 2018; Singh et al., 2019), and complex structural features in protein folding (Norn et al., 2020) by interrogating neural networks. Many current neural network attribution methods rely on local approximation to estimate individual input feature contribution (Simonyan et al., 2013; Zeiler et al., 2014; Springenberg et al., 2014; Sundararajan et al., 2017; Shrikumar et al., 2017; Lundberg et al., 2017). However, this approach has two main shortcomings: first, it assumes independence of input features, which is often incorrect. Second, it struggles with saturated activations within the network.

Here we take an attribution method from computer vision which naturally overcomes issues with saturation and feature dependence and adapt it to the biological sequence domain: We train a deep generative network to either preserve or perturb a small set of features to preserve or distort a downstream prediction. Our version of this approach, which we call *Scrambler Neural Networks*, is tailored for categorical input patterns such as sequences. Previous methods from computer vision (Fong et al., 2017; Dabkowski et al., 2017) perturb the input by fading or blurring, which is ill-suited for one-hot encoded patterns. Similarly, related feature selection methods for discrete input variables (Chen et al., 2018, *L2X*; Yoon et al. 2018, *INVASE*) learn to select a small set of dimensions using a 0/1 mask, but zeroing out entire 'one-hot' positions can push sequence-predictive models out of distribution. Instead, Scramblers rely on a categorical feature sampling distribution at each sequence position which smoothly interpolates between the current input and a background distribution based on the generated importance scores. This perturbation operator ensures that Gumbel-sampled sequences stay in distribution. In this paper, we show that Scramblers learn meaningful feature attributions and outperform other methods on several biological sequence-predictive tasks.

2 Scrambling Neural Networks

The Scrambler architecture is illustrated in Figure 1A. We let an auto-encoding network $\mathcal{S}$, called the *Scrambler*, reconstruct the input pattern $\hat{\boldsymbol{x}}_{\mathcal{S}} = \mathcal{S}(\boldsymbol{x})$ subject to a conservation penalty $\mathcal{C}_{\text{Cons}}\big(\hat{\boldsymbol{x}}_{\mathcal{S}}\big)$ which enforces high entropy. In practice, $\mathcal{S}$ does not directly reconstruct $\hat{\boldsymbol{x}}_{\mathcal{S}}$, but rather predicts a pattern of real-valued importance scores that are used for reconstruction (see below). Samples $\big\{\hat{\boldsymbol{x}}_{\mathcal{S}}^{(k)}\big\}_{k=1}^{K}$ drawn from this scrambled input distribution $\hat{\boldsymbol{x}}_{\mathcal{S}}$ are passed to the predictor $\mathcal{P}$. By comparing the perturbed predictions $\mathcal{P}(\hat{\boldsymbol{x}}_{\mathcal{S}}^{(k)})$ of the scrambled input samples to the original prediction $\mathcal{P}(\boldsymbol{x})$, we train the scrambler $\mathcal{S}$ to minimize a predictive reconstruction error $\mathcal{C}_{\text{Pred}}\big(\mathcal{P}(\boldsymbol{x}), \mathcal{P}(\hat{\boldsymbol{x}}_{\mathcal{S}}^{(k)})\big)$. We refer to this formulation as the *Inclusion*-Scrambler, as it must learn to include the few important features which allow reconstruction of the original prediction while maintaining high entropy of $\hat{\boldsymbol{x}}_{\mathcal{S}}$ (Figure 1B). We define the prediction error as the KL-divergence between scrambled and original predictions. While we could define the conservation loss as the negative entropy of $\hat{\boldsymbol{x}}_{\mathcal{S}}$, the predictor may not have been trained on uniformly random inputs, meaning a negative entropy penalty might push the samples to pathological regions of input space. Instead, we minimize the KL-divergence between $\hat{\boldsymbol{x}}_{\mathcal{S}}$ and a background distribution $\tilde{\boldsymbol{b}}$, which is here taken as the mean input pattern across the training set. The complete training objective for the Inclusion-Scrambler is given in Equation 1:

$$\min_{\mathcal{S}} \lambda \cdot \text{KL}\big[\mathcal{P}(\hat{\boldsymbol{x}}_{\mathcal{S}}^{(k)}) || \mathcal{P}(\boldsymbol{x})\big] + (1 - \lambda) \cdot \text{KL}\big[\tilde{\boldsymbol{b}} || \hat{\boldsymbol{x}}_{\mathcal{S}}\big] \tag{1}$$

The intuition behind the Inclusion-Scrambler is that if most of $\boldsymbol{x}$ is randomized, the small feature set **not** scrambled must coincide with the most important features to maintain predictive accuracy. Alternatively, we could train $\mathcal{S}$ to find the smallest set of features in $\boldsymbol{x}$ to randomize (i.e., maximize conservation of $\hat{\boldsymbol{x}}_{\mathcal{S}}$) to maximally perturb $\mathcal{P}(\hat{\boldsymbol{x}}_{\mathcal{S}}^{(k)})$ from $\mathcal{P}(\boldsymbol{x})$. These extremal perturbations should also coincide with an important feature set, not necessarily the same as those found by the Inclusion-Scrambler. This inverse formulation, referred to as the *Occlusion*-Scrambler, maximizes Equation 1. In practice, mean squared error is used to reach a target KL-divergence of $\hat{\boldsymbol{x}}_{\mathcal{S}}$ rather than maximizing the KL-divergence unbounded. For both formulations, by training $\mathcal{S}$ on a large representative data set, we learn a parametric model of feature importance whose predictions generalize to new examples, preventing overfitting to spurious predictor signals.

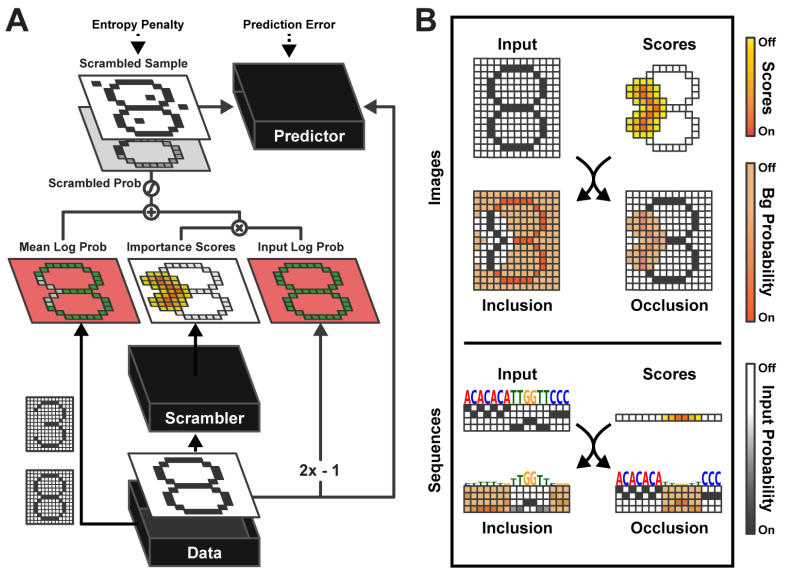

Figure 1: (A) The Scrambling network architecture. Only 'Scrambler' is trainable ('Predictor' is pre-trained). (B) Illustration of the Inclusion and Occlusion scrambling operations.

Part of the novelty of Scramblers is their efficient, probabilistic formulation for discrete input patterns such as biological sequences. We define $\boldsymbol{x} \in \{0,1\}^{N \times M}$ as a one-hot-coded sequence pattern. Let $\mathcal{S}(\boldsymbol{x}) \in (0,\infty]^N$ be the real-valued importance scores predicted by the scrambler for the N-length input sequence $\boldsymbol{x}$. We first broadcast the importance score $\mathcal{S}(\boldsymbol{x})_i$ at position i to all channels j, obtaining $\dot{\mathcal{S}}(\boldsymbol{x}) \in (0,\infty]^{N \times M}$, and recast $\boldsymbol{x}$ from values $\{0,1\}$ to $\{-1,1\}$. We then use $\dot{\mathcal{S}}(\boldsymbol{x})$ as interpolation coefficients in log-space to get the scrambled input probability distribution $\hat{\boldsymbol{x}}_{\mathcal{S}}$:

$$\hat{\boldsymbol{x}}_{\mathcal{S}} = \sigma\big(\log \tilde{\boldsymbol{b}} + \boldsymbol{x} \times \dot{\mathcal{S}}(\boldsymbol{x})\big) \tag{2}$$

In equation 2, σ denotes the softmax $\sigma(\boldsymbol{l})_{ij} = \frac{e^{l_{ij}}}{\sum_{k=1}^{4} e^{l_{ik}}}$. Samples $\{\hat{\boldsymbol{x}}_{\mathcal{S}}^{(k)}\}_{k=1}^{K}$ drawn from $\hat{\boldsymbol{x}}_{\mathcal{S}}$ are passed to $\mathcal{P}$ and gradients are backpropagated to $\mathcal{S}$ using either Softmax Straight-Through estimation (Chung et al., 2016) or the Gumbel distribution (Jang et al., 2016). The importance scores $\mathcal{S}(\boldsymbol{x})$ take on a meaningful interpretation: when $\mathcal{S}(\boldsymbol{x})_i$ is close to 0, $\hat{\boldsymbol{x}}_{\mathcal{S},i}$ becomes $\tilde{\boldsymbol{b}}_i$ (the background distribution) and when $\mathcal{S}(\boldsymbol{x})_i$ is close to ∞, $\hat{\boldsymbol{x}}_{\mathcal{S},i}$ becomes $\boldsymbol{x}_i$ (the original input). $\mathcal{S}(\boldsymbol{x})_i$ thus defines the log-probability of keeping input feature i. Inversely, for the Occlusion-Scrambler we replace $\boldsymbol{x} \times \dot{\mathcal{S}}(\boldsymbol{x})$ in Equation 2 with $\boldsymbol{x}/\dot{\mathcal{S}}(\boldsymbol{x})$. Now, $\mathcal{S}(\boldsymbol{x})$ corresponds to log-probabilities of replacing input features with samples from the background distribution.

3 Results

In all experiments, the Scrambler consisted of a residual network with 20 blocks of dilated convolutions and filter size 3 (He et al., 2016), and scramblers were trained on input examples separate from those used in the benchmarks. The baseline method used for comparison, *Perturb*, exchanges the categorical value of one letter or pixel at a time and estimates the absolute value in predicted change as the importance score. Comparisons are made against Perturb (baseline), Gradient Saliency (Simonyan et al., 2013), Guided Backprop (Springenberg et al., 2014), Integrated Gradients (Sundararajan et al., 2017), DeepLIFT (Shrikumar et al., 2017; using RevealCancel for MNIST and Rescale from DeepExplain for the remaining tasks; Ancona et al., 2017), SHAP DeepExplainer

(Lundberg et al., 2017) and the extremal preservation/perturbation methods of Fong et al. (2019; 'TorchRay') and Dabkowski et al. (2017; 'Saliency Model'). For comparison, we also use a version of the Scrambler with a perturbation identical to L2X and INVASE (referred to as 'Zero Scrambler'): $\min_{\mathcal{M}} \mathrm{KL}\big[\mathcal{P}(\boldsymbol{x} \times \mathcal{M}(\boldsymbol{x}))||\mathcal{P}(\boldsymbol{x})\big] + \lambda\|\mathcal{M}(\boldsymbol{x})\|$, where $\mathcal{M}(\boldsymbol{x})$ is a binary $0/1$ mask generator.

3.1 Image feature attribution

We first benchmarked the Scrambler on visualizing important regions in binarized MNIST '3' and '5' digits given a CNN trained to discriminate between all ten digits (Figure 2). Importance scores were generated for all test images and comparison methods. All but the top $X\%$ most important pixels were replaced with random values, and the KL-divergence between original and scrambled predictions was measured to ascertain how well the kept features maintained the prediction. We also tested how these top features perturbed the prediction when replaced with random values. The Scramblers proved superior for each benchmark. The Inclusion-Scrambler identified the upper open regions of '3' and '5' as important, while the Occlusion-Scrambler learned to 'flip' '3' into '5' and vice versa.

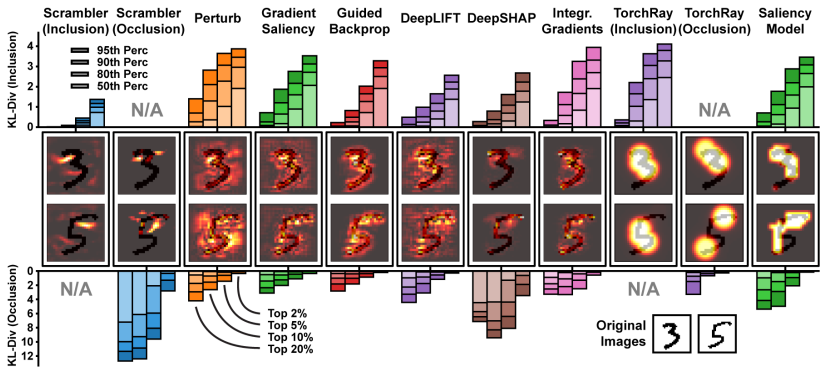

Figure 2: Attributing feature importance for binarized MNIST digits '3' and '5'. (Upper bar chart) Keeping the top $X\%$ pixels according to importance scores and replacing all other pixels with random values. (Lower bar chart) Replacing the top $X\%$ pixels with random values.

3.2 DNA feature attribution

To assess Scrambler detection of non-linear feature sets, an Inclusion-Scrambler was trained for the Optimus 5-Prime model (Sample et al., 2019). This model predicts mean ribosomal load (MRL) from 5' UTR sequences and provides an ideal non-linear evaluation task as complex regulatory logic can occur in 5' UTRs. For example, in-frame (IF) start and stop codons exhibit NAND-type logic by creating an IF upstream open reading frame (uORF), which represses MRL. That is, an IF start followed by an IF stop is necessary to repress MRL, and an IF start or stop in isolation does not result in repression. By including multiple IF stops, we can create challenging NAND-OR-hybrid 5' UTRs for feature attribution tests. The Inclusion-Scrambler exhibited an improved ability to identify non-linear input patterns in such sequences, as exemplified in Figure 3 (two target Scrambler KL-values, 0.125 and 0.25, were tested).

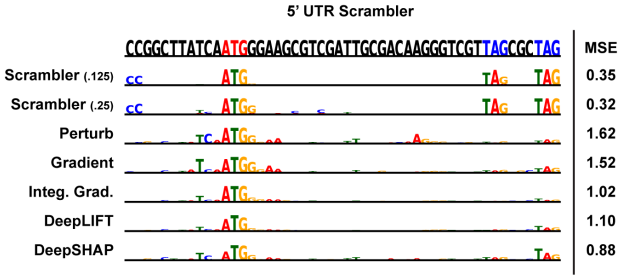

Figure 3: Attributing importance in a 5' UTR. MSE is measured between scrambled and original predictions, when randomizing all but the top 6 nucleotides. (Top sequence) Red letters = IF start codon, Blue letters = IF stop codon.

3.3 Protein feature attribution

In recent work by Chen et al. (2019) a set of computationally generated coiled-coil dimers were designed to have high binding specificity to their cognate binding partner.

Networks of hydrogen bonds were built at the dimer interface to induce binder specificity as part of the design process using HBNet (Boyken et al., 2016; Maguire et al., 2018). We created a negative training set from the original 170,000 designed interacting pairs by randomly pairing binders not designed to interact. We used these data to train an RNN model that could predict dimer binding. Feature attribution for this model is challenging; the naive solution would be to mark hydrophobic residues as important to knock out binding ability to any binding partner. However, a more interesting question is if we can recover HBNets, i.e., the rules used to make binding specific for a designed pair.

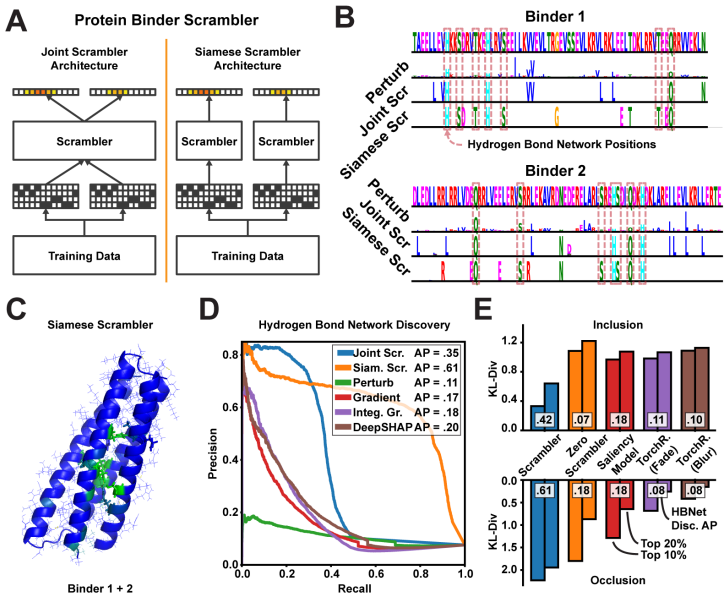

Figure 4: (A) The *joint* and *siamese* Scrambler architectures for protein binder attribution. (B) Example attribution. Hydrogen bonds at the designed binding interface are marked with dashed red boxes. (C) 3D-visualization of binder pair structure. Discovered HBNet residues are colored green. (D) Precision-recall curves for discovering the designed HBNet positions, based on importance scores. (E) Keeping the top $X\%$ residues according to importance scores and replacing the rest with random amino acids (top), or replacing the top $X\%$ with random amino acids (bottom), measuring prediction KL-divergence. Average Precision for discovering HBNet positions annotated on each bar.

We hypothesized that a *joint* Occlusion-Scrambler (Figure 4A), which sees both input binders and can learn to discriminate between binding and non-binding pairs, would exploit the hydrophobic residues to knock out binding. In contrast, a *siamese* architecture that only sees one binder would need to learn features that are good determinants of both binding and non-binding pairs, such as HBNets.

HBNet residues were extracted from a dimer test set to validate Scrambler importance scores and benchmark performance. In Figure 4B-C we visualize an example dimer pair, where the residues in the dimer structure are colored according to importance. Figure 4B shows that the joint Scrambler marks fewer HBNet positions and more leucines as important than the siamese model. Precision-recall curves were generated for each feature attribution method using HBNet residue positions as ground truths. We first compared the Scrambler to methods based on local approximation (Figure 4D). While the joint Scrambler discovered significantly more HBNet positions than other methods – likely because a generative model trained on many examples learns more generalizable attributions – the siamese Scrambler was superior with an average precision of 0.61. Finally, in Figure 4E we compared the Scrambler to similar methods based on either real-valued input perturbations (Fong et al., 2019, 'TorchRay'; Dabkowski et al., 2017, 'Saliency Model') or discrete feature selection (Chen et al., 2018; Yoon et al., 2018; using the 'Zero Scrambler' to approximate both methods). The Inclusion and Occlusion-Scramblers had higher precision in recovering HBNet positions, and their top scored positions had a greater ability to either preserve or distort the RNN network predictions.

4 Conclusion

We presented the Scrambler Neural Network, a feature attribution method based on generative modeling tailored for biological sequence prediction. The Scrambler outperformed other methods on a number of visual- and sequence-predictive tasks. We anticipate that the Scrambler will be a useful tool to discover non-linear feature attributions for an array of biological models.

References

Alipanahi, B., Delong, A., Weirauch, M.T. and Frey, B.J., 2015. Predicting the sequence specificities of DNA-and RNA-binding proteins by deep learning. Nature biotechnology, 33(8), pp.831-838.

Ancona, M., Ceolini, E., Öztireli, C. and Gross, M., 2017. Towards better understanding of gradient-based attribution methods for deep neural networks. arXiv preprint arXiv:1711.06104.

Avsec, Ž., Weilert, M., Shrikumar, A., Alexandari, A., Krueger, S., Dalal, K., ... and Zeitlinger, J., 2019. Deep learning at base-resolution reveals motif syntax of the cis-regulatory code (bioRxiv).

Bogard, N., Linder, J., Rosenberg, A.B. and Seelig, G., 2019. A deep neural network for predicting and engineering alternative polyadenylation. Cell, 178(1), pp.91-106.

Boyken, S.E., Chen, Z., Groves, B., Langan, R.A., Oberdorfer, G., Ford, A., Gilmore, J.M., Xu, C., DiMaio, F., Pereira, J.H. and Sankaran, B., 2016. De novo design of protein homo-oligomers with modular hydrogen-bond network–mediated specificity. Science, 352(6286), pp.680-687.

Chen, Z., Boyken, S.E., Jia, M., Busch, F., Flores-Solis, D., Bick, M.J., Lu, P., VanAernum, Z.L., Sahasrabuddhe, A., Langan, R.A. and Bermeo, S., 2019. Programmable design of orthogonal protein heterodimers. Nature, 565(7737), pp.106-111.

Chen, J., Song, L., Wainwright, M.J. and Jordan, M.I., 2018. Learning to explain: An information-theoretic perspective on model interpretation. arXiv preprint arXiv:1802.07814.

Chung, J., Ahn, S. and Bengio, Y., 2016. Hierarchical multiscale recurrent neural networks (arXiv).

Dabkowski, P. and Gal, Y., 2017. Real time image saliency for black box classifiers. In Advances in Neural Information Processing Systems (pp. 6967-6976).

Eraslan, G., Avsec, Ž., Gagneur, J. and Theis, F.J. 2019. Deep learning: new computational modelling techniques for genomics. Nature Reviews Genetics 20, 389-403.

Fong, R.C. and Vedaldi, A., 2017. Interpretable explanations of black boxes by meaningful perturbation. In Proceedings of the IEEE International Conference on Computer Vision (pp. 3429-3437).

Fong, R., Patrick, M. and Vedaldi, A., 2019. Understanding deep networks via extremal perturbations and smooth masks. In Proceedings of the IEEE International Conference on Computer Vision (pp. 2950-2958).

He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

Jaganathan, K., Kyriazopoulou Panagiotopoulou, S., McRae, J. F., Darbandi, S. F., Knowles, D., Li, Y. I., ... and Farh, K. K.-H., 2019. Predicting Splicing from Primary Sequence with Deep Learning. Cell 176, 535-548.

Jang, E., Gu, S. and Poole, B., 2016. Categorical reparameterization with gumbel-softmax (arXiv).

Kelley, D.R., Snoek, J. and Rinn, J.L., 2016. Basset: learning the regulatory code of the accessible genome with deep convolutional neural networks. Genome research, 26(7), pp.990-999.

Kelley, D.R., Reshef, Y.A., Bileschi, M., Belanger, D., McLean, C.Y. and Snoek, J., 2018. Sequential regulatory activity prediction across chromosomes with convolutional neural networks. Genome research, 28(5), pp.739-750.

Lundberg, S.M. and Lee, S.I., 2017. A unified approach to interpreting model predictions. In Advances in neural information processing systems (pp. 4765-4774).

Maguire, J., Boyken, S., Baker, D. and Kuhlman, B., 2018. Rapid Sampling of Hydrogen Bond Networks for Computational Protein Design. J Chem Theory Comput., (14(5), pp.2571-2760.

Movva, R., Greenside, P., Marinov, G. K., Nair, S., Shrikumar, A. and Kundaje, A., 2019. Deciphering regulatory DNA sequences and noncoding genetic variants using neural network models of massively parallel reporter assays. PloS One, 14(6).

Norn, C., Wicky, B.I., Juergens, D., Liu, S., Kim, D., Koepnick, B., Anishchenko, I., Baker, D. and Ovchinnikov, S., 2020. Protein sequence design by explicit energy landscape optimization. bioRxiv.

Sample, P. J., Wang, B., Reid, D. W., Presnyak, V., McFadyen, I. J., Morris, D. R. and Seelig, G., 2019. Human 5' UTR design and variant effect prediction from a massively parallel translation assay. Nature Biotechnology 37, 803-809.

Senior, A. W., Evans, R., Jumper, J., Kirkpatrick, J., Sifre, L., Green, T., ... and Penedones, H., 2020. Improved protein structure prediction using potentials from deep learning. Nature, 1-5.

Shrikumar, A., Greenside, P. and Kundaje, A., 2017. Learning important features through propagating activation differences. arXiv preprint arXiv:1704.02685.

Simonyan, K., Vedaldi, A. and Zisserman, A., 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034.

Singh, S., Yang, Y., Poczos, B. and Ma, J., 2019. Predicting enhancer-promoter interaction from genomic sequence with deep neural networks. Quantitative Biology, 7(2), pp.122-137.

Springenberg, J.T., Dosovitskiy, A., Brox, T. and Riedmiller, M., 2014. Striving for simplicity: The all convolutional net. arXiv preprint arXiv:1412.6806.

Sundararajan, M., Taly, A. and Yan, Q., 2017. Axiomatic attribution for deep networks. arXiv preprint arXiv:1703.01365.

Yang, J., Anishchenko, I., Park, H., Peng, Z., Ovchinnikov, S. and Baker, D., 2020. Improved protein structure prediction using predicted interresidue orientations. Proceedings of the National Academy of Sciences.

Yoon, J., Jordon, J. and van der Schaar, M., 2018, September. INVASE: Instance-wise variable selection using neural networks. In International Conference on Learning Representations.

Zeiler, M.D. and Fergus, R., 2014, September. Visualizing and understanding convolutional networks. In European conference on computer vision (pp. 818-833). Springer, Cham.

Zeng, W., Wu, M. and Jiang, R., 2018. Prediction of enhancer-promoter interactions via natural language processing. BMC genomics, 19(2), pp.13-22.

Zeng, W., Wang, Y. and Jiang, R., 2020. Integrating distal and proximal information to predict gene expression via a densely connected convolutional neural network. Bioinformatics, 36(2), pp.496-503.

Zhou, J. and Troyanskaya, O.G., 2015. Predicting effects of noncoding variants with deep learning–based sequence model. Nature methods, 12(10), pp.931-934.