fastISM: Performant *in-silico* saturation mutagenesis for convolutional neural networks

Surag Nair[1], Avanti Shrikumar[1], Anshul Kundaje[1,2]

Abstract

Deep learning models such as convolutional neural networks are able to accurately map biological sequences to associated functional readouts and properties by learning predictive *de novo* representations. *In-silico* saturation mutagenesis (ISM) is a popular feature attribution technique for inferring contributions of all characters in an input sequence to the model's predicted output. The main drawback of ISM is its runtime, as it involves multiple forward propagations of all possible mutations of each character in the input sequence through the trained model to predict the effects on the output. We present fastISM, an algorithm that speeds up ISM by a factor of over 10x for commonly used convolutional neural network architectures. fastISM is based on the observations that the majority of computation in ISM is spent in convolutional layers, and a single mutation only disrupts a limited region of intermediate layers, rendering most computation redundant. fastISM reduces the gap between backpropagation-based feature attribution methods and ISM. It far surpasses the runtime of backpropagation-based methods on multi-output architectures, making it feasible to run ISM on a large number of sequences. An easy-to-use Keras/TensorFlow 2 implementation of fastISM is available at https://github.com/kundajelab/fastISM, and a hands-on tutorial at https://tinyurl.com/y26ytojn.

Introduction

Deep learning models of biological sequences such as DNA have been highly effective on several key prediction tasks such as prediction of protein-DNA binding, splicing and molecular effects of genetic variants[1–5]. Convolutional neural networks (CNNs) are widely used architectures for modeling regulatory DNA since they are well suited to capture known properties and invariances encoded in these sequences[2,4,6]. For example, convolutional filters are reminiscent of classical DNA motif representations known as position weight matrices (PWMs)[7]. Also compared to recurrent neural networks (RNNs)[8,9,10], CNNs have the advantage of being more computationally efficient.

Several feature attribution methods have been developed to infer contribution scores (or importance scores) of individual characters in input sequences with respect to output predictions of neural network models such as CNNs. A popular class of feature attribution methods use backpropagation to efficiently decompose the output prediction of a model, given an input sequence, into character-level attribution scores[11–14]. These attribution scores can be used to infer predictive subsequences within individual input sequences which can then be aggregated over multiple sequences to learn recurring predictive features such as DNA motifs[15].

In-silico saturation mutagenesis (ISM) is an alternate feature attribution approach that involves making systematic mutations to each character in an input sequence and computing the change in the model's output due to each mutation. ISM has been used to score the effects of genetic variants in DNA sequences[2,6,16]. However, ISM can be orders-of-magnitude more computationally expensive than backpropagation-based feature attribution methods[5], since it involves a forward propagation pass of the model for every mutation of every position in an input sequence. By contrast, backpropagation-based methods can compute attribution scores of all possible characters at all positions in an input sequence in one or a few backward propagations of the model. The inefficiency of ISM is particularly onerous when it is performed on a large number of sequences or for a large number of models, as is often the case[16].

At the same time, ISM offers salient benefits over backpropagation-based methods. In comparison to other feature attribution methods that are often heuristic in nature, ISM faithfully represents the model's response to mutations at individual positions. This makes it the method of choice when evaluating the

[1]Department of Computer Science, Stanford University [2]Department of Genetics, Stanford University

effect of genetic variants on the output[1,2,16], and it is also used as a benchmark reference when evaluating fidelity of other feature attribution methods[17]. Unlike ISM, backpropagation-based methods like DeepLIFT and Integrated Gradients rely on a predefined set of "neutral" input sequences that are used as a reference to compute attribution scores. The choice of reference sequences can influence the scores and so far the selection of reference sequences has been ad-hoc[5]. The advantages of ISM are more apparent in the case of multi-output models. Each forward propagation performed during ISM reveals the impact of a single mutation on every output of the model. For example, a recent class of models called profile models have been developed to map regulatory DNA sequences to base-resolution binding profiles of proteins[18]. These models output a vector of signal values that is often as long as the input sequences. ISM can reveal how perturbing individual nucleotides in the input alters the signal across all positions in the output profile. By contrast, backpropagation-based importance scoring methods need to perform a separate backpropagation for every position in the output profile in order to provide comparable information, which would linearly increase the computational cost in the number of outputs. For these reasons, a computationally efficient implementation of ISM would be attractive.

We introduce fastISM, an algorithm that speeds up ISM for CNNs. fastISM is based on the observation that CNNs spend the majority of computation at prediction time in convolutional layers and that single point mutations in the input sequence affect a limited range of positions in intermediate convolutional layers. We benchmark the speedup obtained by running fastISM on a variety of architectures and output types and show that fastISM can achieve order-of-magnitude improvements over standard ISM implementations. fastISM reduces the gap between ISM and backpropagation-based methods in terms of runtime on single-output architectures, and far surpasses them on multi-output architectures.

Results

In-silico mutagenesis for CNNs is bottlenecked by redundant computations in convolution layers

We motivate fastISM by using an example based on the multi-task Basset CNN model that maps 1000 bp DNA sequences to binary labels of chromatin accessibility across 100s of cell types and tissues (tasks)[6]. We consider a slight modification of the original architecture. We map 1000bp input sequences to 10 binary scalar outputs. In addition, all convolution layers are assumed to be padded such that the length of the output sequence is the same as the input to the layer (**Fig 1**). For a given convolution layer, the number of computations is proportional to kernel size, input filters, output filters and output length. The numbers in black in **Fig 1** show the approximate computations required at each convolution and fully connected layer for a single forward propagation. The computations required in the convolutional layers combined exceed that of the fully connected layers by a factor of 50x.

For a given reference input sequence, a standard implementation of ISM involves highly redundant computations. Typically, ISM is implemented by inserting mutations at each position in the input one at a time and making a forward pass through the entire model using the perturbed sequences as inputs. However, local perturbations in the input only affect local regions in intermediate convolutional layers, while regions farther away remain identical to their values for the reference (unperturbed) input sequence. For each layer, the regions that are affected by the single base pair mutation in the input, and the minimal regions required to compute the output of the next layer are shown in **Fig 1**. By design, a single mutation in the input sequence has the potential to affect every position in the fully-connected layer; thus, the activations of all subsequent layers in the network must be recomputed for the mutated input, as is the case with standard ISM. By restricting ISM computations to only positions affected by the input mutation at each layer, the number of computations can theoretically be reduced from approximately 271M to 15.5M computations (~17x). This suggests that it should be possible to define a custom model that performs only the required computations for a mutation at each input position. However, it would be cumbersome to write an architecture specifically for the purpose of ISM for each model. Hence, we developed fastISM, a method to speed up ISM by leveraging the above mentioned redundancies without requiring any explicit re-specification of the model architecture by the user (Supplementary Methods).

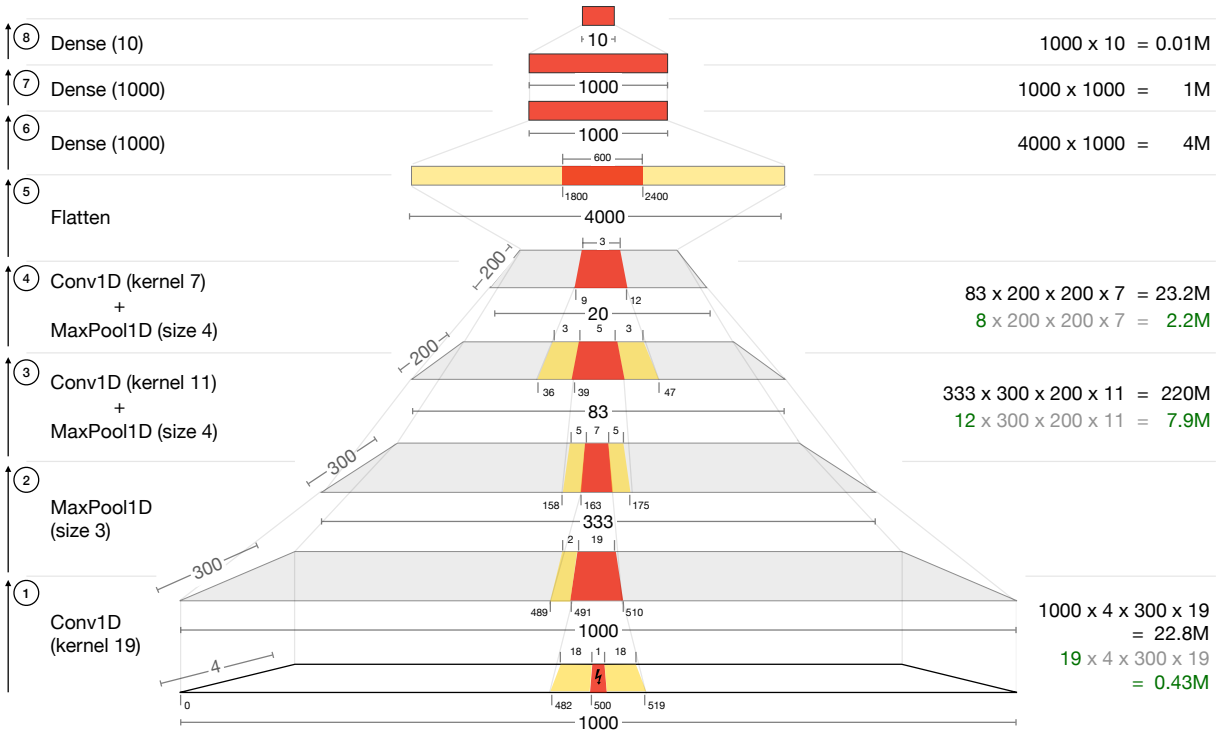

Fig 1. Annotated diagram of a Basset-like architecture[6] on an input DNA sequence of length 1000, with a 1 base-pair mutation at position 500. Positions marked in red indicate regions that are affected by the point mutation in the input. Positions marked in yellow, flanking the positions in red, indicate unaffected regions that contribute to the output of the next layer. Ticks at the bottom of each layer correspond to position indices. Numbers on the right in black show the approximate number of computations required at that layer for a standard implementation of ISM. For convolution layers, the numbers in gray and green indicate the minimal computations required. We omit layers such as activations and batch normalization for simplicity, as they do not change the affected regions.

fastISM yields order-of-magnitude improvements in speed for different architectures

fastISM takes as input any sequence model and first reduces it to a computational graph representation. It chunks the graph into appropriate segments that can each be run as a unit. The model is augmented to return intermediate outputs at the end of each segment for unperturbed inputs. For a given set of input sequences, these intermediate outputs are cached. A second model is then initialized, that largely resembles the original model, but incorporates mechanisms to concatenate slices of the cached unperturbed intermediate outputs with the affected intermediate outputs, and compute each layer's output on the least required input. For each group of sequences, fastISM is run on multiple batches such that sequences in a given batch are all perturbed at the same position.

We benchmarked fastISM against a standard implementation of ISM. We choose 3 types of models that take DNA sequence as input— Basset[6], Factorized Basset[19] and BPNet[18]. The first two output scalar values for each output task, whereas BPNet outputs a profile vector of length equal to the input sequence length, and a scalar count. ISM is performed by recording the outputs for all 3 alternate mutations at each position. We benchmark on 1000bp and 2000bp length inputs. We also compare fastISM to three backpropagation-based feature attribution methods — Gradient x Input (input masked gradient), Integrated Gradients[12], and DeepSHAP[13]. We used DeepSHAP, an extension of DeepLIFT, as it has a more flexible implementation of the DeepLIFT approach. 50 steps are used for Integrated Gradients with a single default reference of all zeros, and 10 dinucleotide reference sequences are used for DeepSHAP. For single scalar output models, the backpropagation-based methods are run with respect to the scalar output. For BPNet, the methods are run with respect to each output position in the profile vector as well as the scalar output, which multiplies their runtime by the number of output tasks.

While ISM returns one value (change in output score) for each of the 3 alternative nucleotides (with respect to the observed nucleotide) at each position; Integrated Gradients, DeepSHAP and Gradients return one value for each of the 4 possible nucleotides at each position.

The results are summarised in **Table 1**. For the Basset and Factorized Basset architectures, fastISM speeds up ISM by more than 10x when computing importance scores for a single output task, and the speedup increases with increasing input sequence length. This is expected since, for a fixed architecture, the length of the affected regions in the convolutional layers are independent of input sequence length. fastISM runtimes, though slower than Gradient x Input, are competitive with runtimes of Integrated Gradients (within 2x) and DeepSHAP (within 4x) for single scalar output models. The speedup of fastISM for the BPNet architecture relative to standard ISM is more modest— 1.6x for 1000bp input and 2.1x for 2000bp input. This can be attributed to the large receptive fields for the later dilated convolutions. The regions affected by a mutation in the input sequence span a sizeable fraction of intermediate layers, and the computations involved beyond those layers approach those of a standard implementation. For the BPNet architecture which outputs a profile vector, DeepSHAP and Integrated Gradients take over 50x time of the fastISM implementation and are not viable for interrogating output position specific importance scores. Thus, fastISM speeds up ISM by an order-of-magnitude and narrows the gap in compute time between backpropagation-based methods and ISM.

Architect-ure	Layers	Input Size	No. of Outputs	fast ISM	Standard ISM	Gradient x Input	Integrated Gradients	Deep SHAP
Basset	3 Conv + 3 max pool, 3 fully connected	1000	1	2.70	27.36 (10.1)	0.04 (<<1)	2.34 (0.8)	1.75 (0.7)
		2000		6.49	100.44 (15.4)	0.08 (<<1)	4.61 (0.7)	3.03 (0.4)
Factorized Basset	9 Conv + 3 max pool, 3 fully connected	1000	1	5.47	68.97 (12.6)	0.09 (<<1)	4.82 (0.9)	2.64 (0.5)
		2000		18.04	262.24 (14.5)	0.17 (<<1)	9.47 (0.5)	4.63 (0.25)
BPNet	2 Conv, 9 Dilated Conv, skip connections	1000	1000 + 1	28.97	46.09 (1.6)	41.49* (1.4)	4399* (151)	1743* (60)
		2000	2000 + 1	81.52	173.96 (2.1)	126.41* (1.5)	12440* (152)	6427* (78)

Table 1: Comparison of fastISM with standard ISM, Gradient x Input, Integrated Gradients with 50 steps and a single all-zeros reference, and DeepSHAP with 10 references for 3 different models with 1000bp and 2000bp length inputs. All times in seconds per 100 input sequences. Time relative to fastISM in parentheses. For BPNet models which output a profile vector as well as a count scalar, Gradient x Input, Integrated Gradients and DeepSHAP were computed in a loop with respect to each output of the profile and the count scalar (*).

Conclusions

In-silico saturation mutagenesis (ISM) is an important post-hoc technique that has gained applicability as a tool to interpret deep learning models for genomics and to interrogate the effect of variants. ISM has largely been treated as a static method with unfavourable time complexity compared to more recent backpropagation-based model interpretability methods. We challenge this notion by introducing fastISM, a performant implementation of ISM for convolutional neural networks. fastISM leverages the simple observation that the majority of computations performed in a traditional implementation of ISM are redundant. fastISM improves runtime of ISM by over 10x for commonly used convolutional neural networks, and a factor of 2x for profile networks with exponentially wide dilated convolutions. This brings down ISM's runtime in the ballpark of backpropagation-based methods such as Integrated Gradients and DeepSHAP for single-output models, and dramatically surpasses the runtime of backpropagation-based methods for multi-output methods, making it more feasible to run ISM genome-wide and on a large number of models.

References

1. Zhou, J. *et al.* Deep learning sequence-based ab initio prediction of variant effects on expression and disease risk. *Nat. Genet.* **50,** 1171–1179 (2018).
2. Zhou, J. & Troyanskaya, O. G. Predicting effects of noncoding variants with deep learning-based sequence model. *Nat. Methods* **12,** 931–934 (2015).
3. Jaganathan, K. *et al.* Predicting Splicing from Primary Sequence with Deep Learning. *Cell* **176,** 535-548.e24 (2019).
4. Alipanahi, B., Delong, A., Weirauch, M. T. & Frey, B. J. Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning. *Nat. Biotechnol.* **33,** 831–838 (2015).
5. Eraslan, G., Avsec, Ž., Gagneur, J. & Theis, F. J. Deep learning: new computational modelling techniques for genomics. *Nat. Rev. Genet.* **20,** 389–403 (2019).
6. Kelley, D. R., Snoek, J. & Rinn, J. L. Basset: learning the regulatory code of the accessible genome with deep convolutional neural networks. *Genome Res.* **26,** 990–999 (2016).
7. Trabelsi, A., Chaabane, M. & Ben-Hur, A. Comprehensive evaluation of deep learning architectures for prediction of DNA/RNA sequence binding specificities. *Bioinformatics* **35,** i269–i277 (2019).
8. Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural Comput.* **9,** 1735–1780 (1997).
9. Quang, D. & Xie, X. DanQ: a hybrid convolutional and recurrent deep neural network for quantifying the function of DNA sequences. *Nucleic Acids Res.* **44,** e107 (2016).
10. Hassanzadeh, H. R. & Wang, M. D. Deeperbind: enhancing prediction of sequence specificities of DNA binding proteins. *Proceedings (IEEE Int Conf Bioinformatics Biomed)* **2016,** 178–183 (2016).
11. Shrikumar, A., Greenside, P. & Kundaje, A. Learning Important Features Through Propagating Activation Differences. in **70,** 3145–3153 (Proceedings of Machine Learning Research, 2017).
12. Sundararajan, M., Taly, A. & Yan, Q. Axiomatic Attribution for Deep Networks. *arXiv* (2017).
13. Lundberg, S. M. & Lee, S.-I. A Unified Approach to Interpreting Model Predictions. (2017).
14. Simonyan, K., Vedaldi, A. & Zisserman, A. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. *arXiv* (2013).
15. Shrikumar, A. *et al.* Technical Note on Transcription Factor Motif Discovery from Importance Scores (TF-MoDISco) version 0.5.6.5. *arXiv* (2018).
16. Wesolowska-Andersen, A. *et al.* Deep learning models predict regulatory variants in pancreatic islets and refine type 2 diabetes association signals. *elife* **9,** (2020).
17. Koo, P. K. & Ploenzke, M. Improving representations of genomic sequence motifs in convolutional networks with exponential activations. *BioRxiv* (2020). doi:10.1101/2020.06.14.150706
18. Avsec, Z. *et al.* Deep learning at base-resolution reveals motif syntax of the cis-regulatory code. *BioRxiv* (2019). doi:10.1101/737981
19. Wnuk, K. *et al.* Deep learning implicitly handles tissue specific phenomena to predict tumor DNA accessibility and immune activity. *iScience* **20,** 119–136 (2019).