Utilizing the gene expression prediction algorithm Enformer to enhance classification of noncoding variants

Nimrod de Wit, Brent Pedersen, Lucía Barbadilla and Jeroen de Ridder

De Ridder Group, Center for Molecular Medicine, UMC Utrecht, 3584 CG Utrecht, The Netherlands

Studying the relation between genetic variation and phenotype provides insight into molecular mechanisms and possibilities for treatment of disease. Since 98% of the genome is noncoding, the noncoding genome is an important research domain. However, determining the effect of noncoding genetic variation remains a challenge. One solution to this challenge is to use the predictive power of the state-of-the-art Enformer algorithm, which predicts gene expression and other assays from sequence. These predictions can be used as input features to a classification algorithm that predicts the effect of a genetic variant. However, since the Enformer output is too large to directly use as input features, it needs to be compressed. In this study we present new methods to compress the Enformer output, which enhance classification performance compared to currently used methods. We use an eQTL dataset with eQTLs from 49 different tissue types and train multiple random forest classifiers to label noncoding variants as positive or negative eQTLs. Compared to the currently used scoring method Sum, our Sum Abs scoring method increases the average AUROC from 0,748 to 0,777. Additionally, we explore the application of an autoencoder to compress the Enformer predictions and propose multiple ways of continuing this research.

Genetic variation plays a role in almost all human diseases, either by causing the disease, increasing susceptibility to the disease, or enhancing its symptoms.[1] Consequently, studying the relation between genetic variation and clinical disease will provide a powerful tool for identifying the molecular mechanisms underlying disease, opening the way to curing them and enhancing functional interpretation of the genome. More specifically, within the field of genetics, noncoding variants are important. In the first place, because the largest part of the human genome is noncoding (up to 98%). Secondly, research shows that noncoding variants play a role in disease. For example, noncoding driver mutations have become more important in cancer research.[2] In addition, regulatory regions show signs of negative selection, suggesting noncoding variants having deleterious effects.[3] Similarly, studies of inherited common variants show enriched disease association in noncoding regions.[4] Moreover, noncoding variants affecting gene expression have been found to cause Mendelian diseases[5] and to be enriched in cancer[6].

Although noncoding variants are proven to be important for disease, they are challenging to explore. The reasons for this are, firstly, that not much is known about the function of noncoding DNA. Secondly, their effect is not as straightforward as coding mutations, and as a consequence functional interpretation of noncoding variants remains a challenge,[7,8] also because there are many mechanisms of action. And thirdly, the effect of variants located in the noncoding region of the genome can vary from no effect at all to vast changes in gene regulation.

For these reasons there is need for methods that prioritize variants as potentially causal for disease.[9,10] Multiple methods have been proposed and put into use by which variants can be prioritized.[9] Examples of these methods are comparative genomics data[11], high-throughput epigenomics and functional genomics data (e.g., massively parallel reporter assays

(MPRAs))[12], and the development of public databases that integrate various annotation sources for noncoding variants[13].

A promising development within the field of variant prioritization is the occurrence of a new generation of data resources. They distinguish themselves by providing quantitative scores for the deleterious impact of non-coding variants on gene regulation. Most of them deploy supervised machine learning to calculate the deleterious impact. For example, Combined Annotation-Dependent Depletion (CADD) was one of the earliest methods integrating a wide range of annotation data and able to provide a score for every variant of the whole genome (both coding and noncoding).[14] The authors used a linear kernel support vector machine (SVM) for this. With the publishing of a later generation of these quantitative score providing algorithms, called Deleterious Annotation using Neural Networks (DANN)[15], the use of deep learning algorithms was introduced to this problem.

Deep learning in general is a promising tool for biological research. It is a class of machine learning algorithms, able to identify highly complex patterns in large datasets.[16] In general, genomics data are too large and too complex to be studied by eye or using simple correlations. Since machine learning algorithms are designed to automatically detect patterns in data, they are especially well suited for genomics.[17] A model's performance on a classification task depends heavily on the quality and the relevance of the input features. Therefore, deep learning is a great option for genomics, since it addresses the problem of input features by embedding the computation of features into the model itself, creating end-to-end models. This means that raw data – a bare DNA strand – can be used as input features.

One solution for researching noncoding variants is using a new type of deep learning models using a transformer architecture.[17] A particular model of this type, Enformer, can take a bare DNA strand as input (ranging from 1kb[18] to 200kb[19]), and as output predict gene expression levels and chromatin states (e.g., DNase, ChIP-seq and CAGE genomic tracks). Utilizing these models to the noncoding variant prioritization problem is a promising route to take for multiple reasons. Firstly, because – by using raw data as input – these models learn the "language" of the DNA, which we humans not yet understand. Secondly, wet lab experiments are laborious and with these models there is no need for these experiments since the model itself will be able to predict at experiment accuracy the results of thousands of experiments. E.g., performing a MPRA is an option, but this is laborious.[20] In contrast, these models make it possible to easily and within seconds determine the effect of a variant *in silico*.

Enformer is the state of the art for predicting gene expression and other assays from sequence[19] (outperforming Basenji2[21] and other precursors, e.g., ExPecto[22], Basenji[23] and Xpresso[24]). This model is suitable for using it in the process of classifying noncoding variants for several reasons. Firstly, gene regulation by the noncoding DNA includes long ranging interactions (e.g., enhancers). And since the model has a transformer architecture (a type of deep learning architecture where each position attends to the

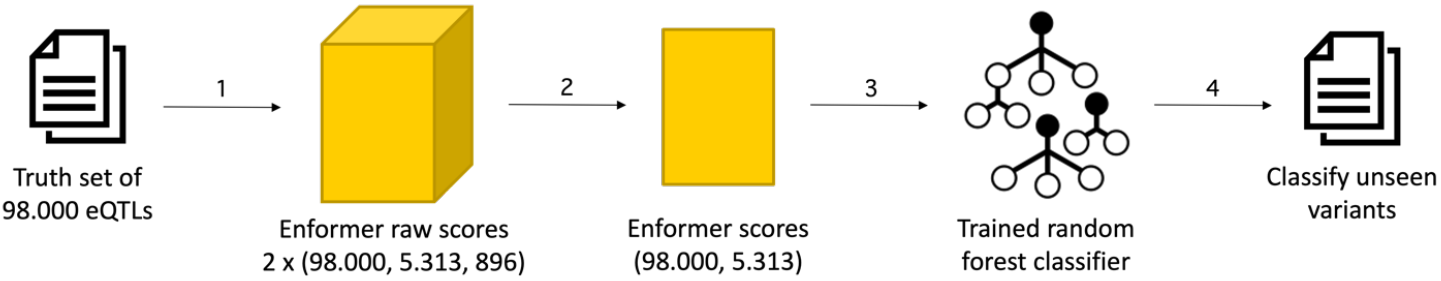


Figure 1. Project overview. This is the overview of the project, with step 1) inserting two DNA strands containing the reference and alternative alleles into the Enformer algorithm to obtain the Enformer raw scores, step 2) compressing the Enformer raw scores (2 x [98.000, 5.313, 896]) to obtain the Enformer scores ([98.000, 5.313]), step 3) using the Enformer scores as input features to train and validate a random forest classifier, and step 4) using the trained classifier to classify unseen variants.

information of all other positions, increasing information flow between distal elements[25]), it can account for long range interactions, taking a 200 kb DNA sequence as input. Secondly, the mechanism of action of noncoding variants is influencing epigenetics, and the epigenetic state is what is predicted by Enformer. It predicts 5.313 genomic tracks: 2.131 transcription factor (TF) chromatin immunoprecipitation and sequencing (ChIP–seq), 1.860 histone modification ChIP–seq, 684 DNase-seq or ATAC-seq, and 638 CAGE tracks, from multiple different cell types (see methods for full description of genomic tracks). Tracks represent a 115 kb DNA strand aggregated into 128bp bins, resulting in 896 values per track.

In the study of Avsec *et al.* the Enformer predictions are used for prioritization and classification of variants. We sought to optimize the training and perf ormance of this final variant classification step.

In this study, we introduce improved methods on how to use the output of the Enformer model to classify noncoding variants. More specifically, we designed new scoring methods to compress the Enformer output to utilize it for classification of genomic variants, resulting in better classifier performance. Additionally, we propose promising ways forward in the application of the Enformer algorithm to the study of noncoding variants.

Results

General workflow. The general workflow of this study is that we, firstly, take a certain truth set of (noncoding) variants. This truth set is a set of variants of which the labels are known, which therefore can be used for training a model. Then, for each of the variants, we take the surrounding 196.608 bp of the hg38 reference genome, insert the reference or alternative allele into the sequence and use this sequence as an input to the Enformer model, generating two arrays of dimensions 5.313 x 896 per variant (figure 1, step 1). After this we compress the Enformer raw scores to be able to use it as input to the classifier (figure 1, step 2). These compressed scores will then be used as input features to train, validate and test a classifier (figure 1, step 3). Finally, this trained classifier can be used to classify unseen variants, possibly providing new insights (figure 1, step 4) and helping researchers prioritize non-coding variants.

Use eQTLs as truth set. The method we propose in this study is applicable to a range of different truth sets. For this study we specifically used eQTLs from the GTEx project.[26] We focused on eQTLs because these are variants that change gene expression and thus likely also influence the phenotype or cause disease. One of the problems with eQTLs is that common ways of identifying eQTLs cannot distinguish individual variants, due to linkage disequilibrium (LD). To overcome this, we used a dataset in which the eQTLs are separated into positive and negative eQTLs. This was done by studying the statistical fine mapping of GTEx v8 eQTLs using the Sum of Single Effects (SuSiE) method.[27] This method provides a posterior inclusion probability (PIP) score for each eQTL by which they can be divided into positive (PIP score > 0.9) and negative (PIP score < 0.01) variants. The PIP score is thus a way to distinguish between likely causal and likely non-causal eQTLs, that are grouped due to LD. One advantage of using non-causal eQTLs as the negative set is that this results in a stringent truth set. All variants are eQTLs, but not all are causal. These variants are more challenging to distinguish than, e.g., eQTLs and common variants. So, the classifier must learn the features of causal eQTLs. Moreover, this prevents the classifier just using the genomic location (or promoter sequences in our case) as a feature to distinguish the variants. This prevents confounding, which is a common pitfall in applying ML to genomics.[28] However, there is a downside to using eQTLs. eQTL studies are systematically biased.[29] Despite of this, the study of eQTLs is still worthwhile.

We used the same eQTL dataset as Avsec *et al.* used, which ranged from 54 variants for kidney cortex tissue to 2.740 variants for tibial nerve tissue, with a total of 49 different tissue types (figure 2, see methods).

Improved scoring method of Avsec *et al.* Once we obtain the Enformer predictions for each eQTL, this data needs to be compressed to be able to use it as input for a classifier, because classification using raw data (only Alt – Ref, figure 3a) performs poorly (see Delta in figure 3b) and is compute-intensive. Therefore, we searched for the best scoring method to compress the genomic tracks into a single value per track. The performance of the scoring method was then assessed by training and validating a random forest classifier per

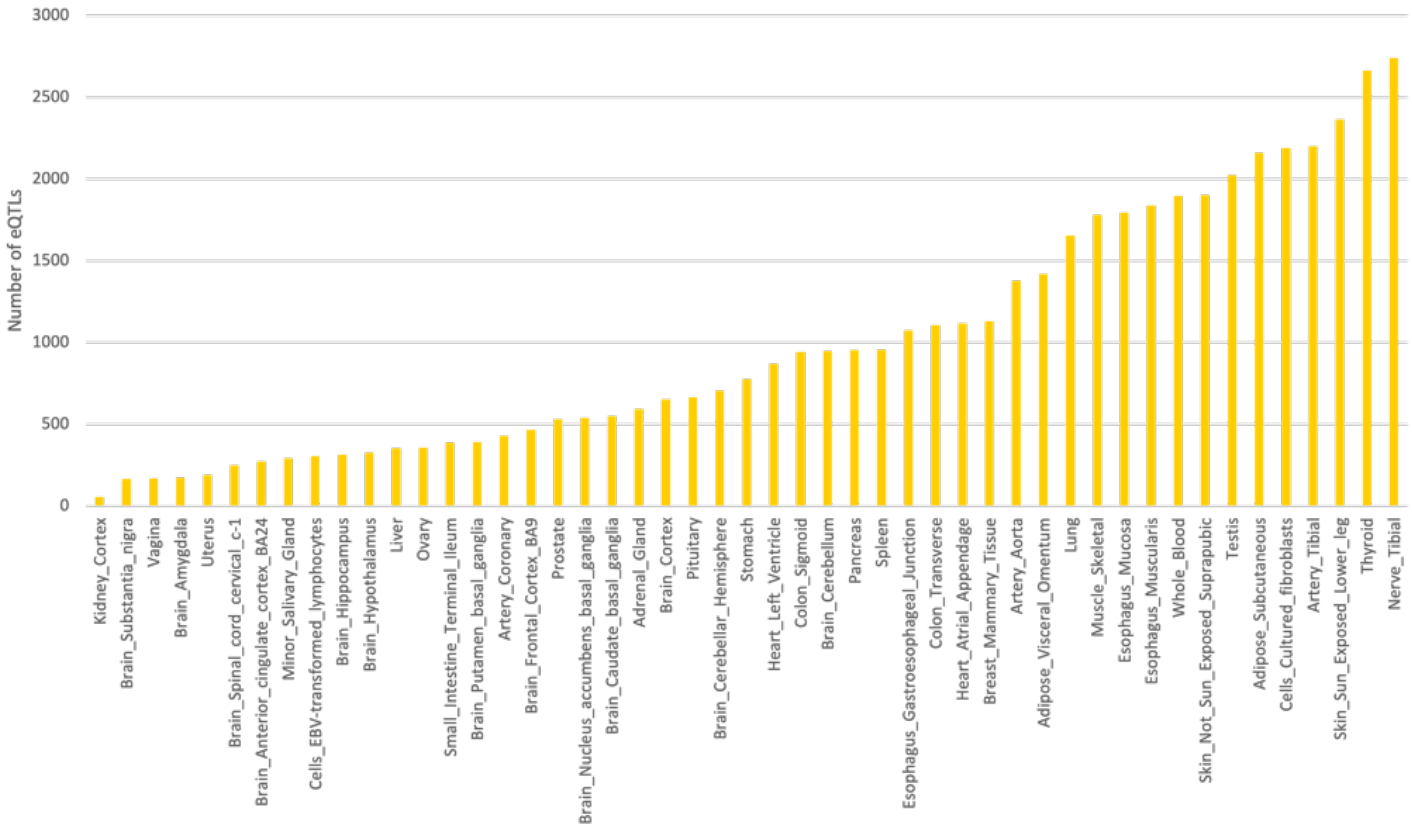


Figure 2. Overview of the eQTL distribution for the 49 different tissue types. An overview of the eQTL dataset used in this study, with the number of positive eQTLs per tissue type, for each of the 49 tissue types. The number of eQTLs per tissue type ranges from 54 positive eQTLs for the kidney cortex to 2741 positive eQTLs for tibial nerve tissue, with an average of 1000 positive eQTLs per tissue type. The dataset is balanced, meaning that the number of negative eQTLs is equal to the number of positive eQTLs. The full dataset contains 98.000 eQTLs.

tissue type, taking the average AUROC of a 10-fold cross validation as a measure of classifier performance (see methods).

The method by which Avsec *et al.* compress the Enformer output is not optimal. The predictions made by the Enformer model consist of 5.313 x 896 = 4.760.448 values per DNA sequence. A variant thus gives rise to 2 x 4.760.448 = 9.520.896 values (two DNA sequences: the reference and alternative allele). Avsec *et al.* compressed the Enformer output by taking the sum across each of the reference and alternative tracks and subtracting them. So, the sum of scores for the output of Enformer given the reference sequence was subtracted from the sum of the output of the DNA sequence containing the variant (see Sum in figure 3a). However, with this way of handling the Enformer output a lot of information is lost and the full potential of the Enformer scores is not utilized. We therefore designed multiple other scoring methods, all of which take the reference and alternative genomic track as input and output a score (figure 3a, supplemental figure 1). The different scoring methods that we tried can be grouped as follows: the methods from Avsec *et al.*, types of correlation between alternative and reference tracks, quantiles, combinations, and max.

Some of our scoring methods show a sizable increase in classifier performance compared to the scoring method employed by Avsec *et al.* (figure 3b, Sum is the scoring method of Avsec *et al.*). The biggest increase in performance is found with the scoring methods that take the biggest difference between Ref and Alt predictions into account (Max Abs, Q99 Abs and Sum Abs, all significant increases, as determined by a paired t-test). The best performing scoring method is Sum Abs, which is a modified version of the method of Avsec *et al.*, namely taking the absolute of the delta before addition. The reason for this is that by taking the absolute the final score represents the total difference in epigenetic regulation between the reference and alternative allele, instead of considering only the total increase (or decrease in case of a negative value), preventing leveling out the signal when in different regions of the genomic track the behavior of

the signal is different (see the example shown in figure 3a). The difference in performance between taking the maximum difference (Max Abs, average AUROC = 0,762) and the 99th quantile of the differences (Q99 Abs, average AUROC = 0,776) can be explained by the fact that with Q99 Abs outliers are not included, which decreases the noise and enhances the predictive power of the features. In conclusion, we observed that the scoring methods that either consider the biggest difference (Max Abs and Q99 Abs) or the total difference (Sum Abs, average AUROC = 0,777) perform the best.

To check the robustness of the random forest classifier to changing hyperparameters, we trained the classifier with different hyperparameter s ettings. The hyperparameters are changed in such a way that the bias of the model increases. Three sets of hyperparameters were applied, with a maximal tree depth of 64, 8 and 4, a minimal number of samples per leaf of 1, 2 and 3, and a minimal number of samples per split of 2, 4 and 6 respectively (supplemental figure 2). Changing hyperparameters to make the models more biased does not dramatically change classifier performances. Therefore, we can conclude that the random forest classifier is at least partly robust to changing hyperparameters.

Next, we checked whether the increase in classification performance due to the new scoring methods is robust to changing the type of classifier. To this end we swapped the random forest classifier with a k-nearest neighbor classifier, support vector classifier, and a multi-layer perceptron classifier (MLPC). The new scoring methods (Max and Q99 + Pearson) were still increasing classifier performance compared to the default scoring method (Sum, figure 4). Importantly, the classifier hyperparameters are not fine-tuned, leaving much room for improvement. E.g., the reason that the Q99 + Pearson scoring method performs worse than the Max scoring method for the MLPC is possibly because the former scoring method produces

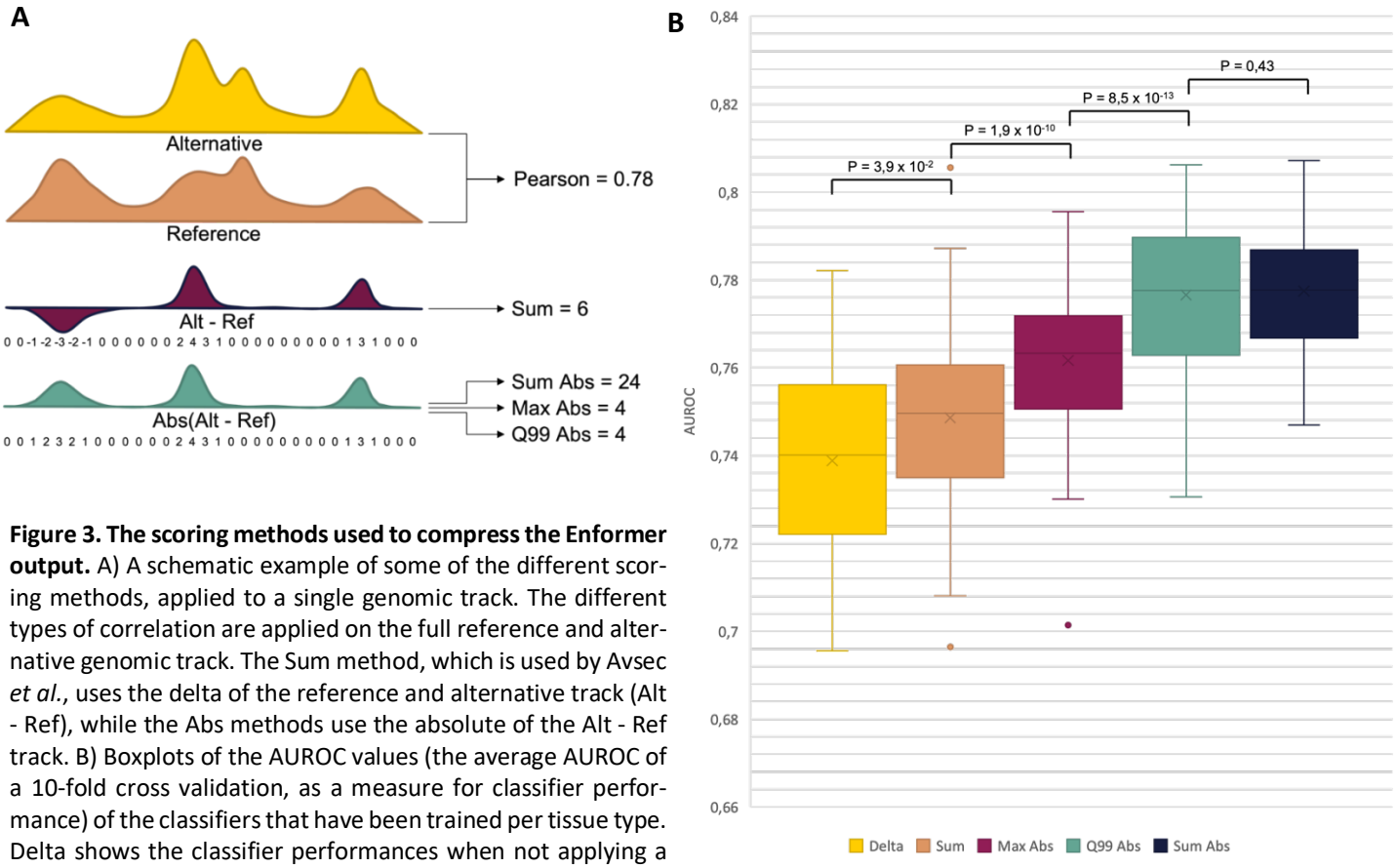


Figure 3. The scoring methods used to compress the Enformer output. A) A schematic example of some of the different scoring methods, applied to a single genomic track. The different types of correlation are applied on the full reference and alternative genomic track. The Sum method, which is used by Avsec *et al.*, uses the delta of the reference and alternative track (Alt - Ref), while the Abs methods use the absolute of the Alt - Ref track. B) Boxplots of the AUROC values (the average AUROC of a 10-fold cross validation, as a measure for classifier performance) of the classifiers that have been trained per tissue type. Delta shows the classifier performances when not applying a scoring method (only subtracting the reference from the alternative tracks). Classifier performances increase from Delta to Sum, from Sum to Max Abs, and from Max Abs to Q99 Abs and Sum Abs. Q99 Abs and Sum Abs have a similar performance. n = 33 for Delta and n = 49 for all other scoring methods. All P values are obtained by a paired two sample t-test (except for the Delta-Sum comparison, of which the t-test was not paired).

twice as many features (10626, instead of 5313), while the MLPC with its default hyperparameter setting has only one hidden layer. Therefore, performance could be enhanced by increasing the model depth as well as by increasing the number of iterations when training.

eQTL classifier for whole blood samples. To work more toward a real-life application of the classifier we created a single classifier to apply on whole blood samples. Here, instead of training a single classifier per tissue type, we combined the eQTLs of alle tissues into one big truth set and trained a single classifier on this dataset. At first sight the classifier performance seemed very high and the new scoring methods don't increase classifier performance (figure 5a). However, we note that this is due to the problem of leakage. This means that the training and validation set are not totally different but partly share the same samples. This in turn is caused by combining the eQTLs of different tissue types, since many eQTLs are shared between tissue types. For example, out of the 2741 positive eQTLs from the tibial nerve tissue, 932 eQTLs are shared with the 2662 positive eQTLs of the thyroid tissue. As a solution to this leakage problem, we removed all eQTLs that were located on chromosome 1 from the dataset that is used for training. (Chromosome 1 was chosen because of its size and averageness. Ideally, one would leave out all chromosomes one by one because they vary in size and importance.) Then, after training the classifier, we tested the classifier on the eQTLs located on chromosome 1 (both the eQTLs from all tissues located on chromosome 1 and only the eQTLs of the whole blood sample that were located on chromosome 1, see figure 5b), which are eQTLs that the classifier has never seen before. Due to this leave-one-chromosome-out strategy, classifier performance decreased significantly because leakage was prevented (AUROC decreasing from 0,9174 to 0,6637 for the Sum scoring method). However, model performance of this single classifier for all tissues was also lower than the average performance o f the tissue-specific classifiers (AUROC 0,6637 for the single classifier compared to 0,7486 for the average of all tissue-specific classifiers). This can be explained by the fact that some eQTLs will have both a positive and a

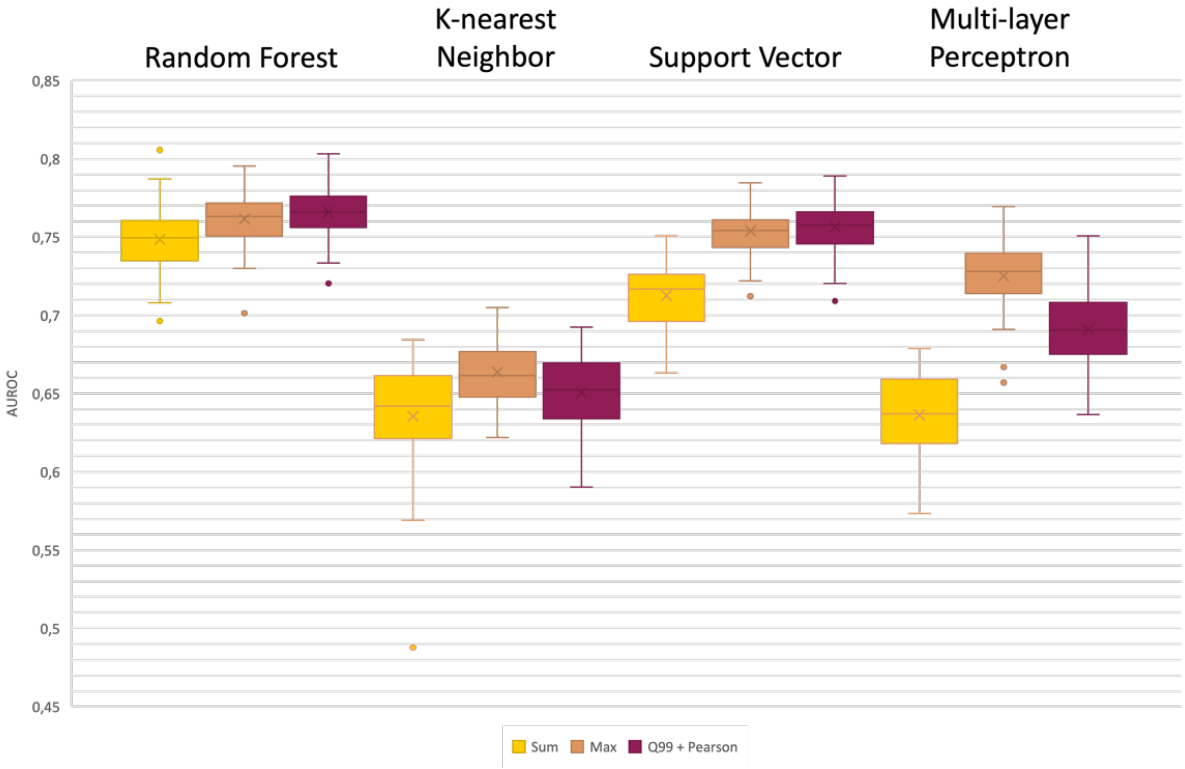


Figure 4. The new scoring methods are also increasing performances of other types of classifiers. When the random forest classifier is replaced by other types of classifiers, the new scoring methods still increase classifier performance compared to the original scoring method. Tested classifiers are a k-nearest neighbor classifier, a support vector classifier and a multilayer perceptron classifier.

negative label, depending on the tissue type they originated from. For example, the variant at position 40008013 on chromosome 15 is labeled positive in the thyroid tissue dataset, while being labeled negative in the tibial artery dataset. But when these datasets are combined into a single dataset, this eQTL is present two times, with contradictory labels. Therefore, the classifier cannot learn what the class of this eQTL is, resulting in a decrease in classifier performance. Also, many eQTLs are tissue specific, making it hard to learn eQTLs expressed in blood from all possible tissues.

Developing an autoencoder that compresses the Enformer output. As described above, we studied the different ways the genomic tracks can be compressed into a single score, by applying different scoring methods. However, there are more possibilities of compressing the Enformer output. For example, the authors of the Enformer paper applied a principal component analysis (PCA) to distill the 5.313 features (resulting from applying the Sum scoring method) into 20 highly informative variant scores.[19] When using these scores as features to train an eQTL classifier (in the same way as is done in this study), the AUROC dropped only from 0,747 to 0,743.

Another way of compressing the Enformer predictions would be to not use the scoring methods described above, but to apply an autoencoder. Autoencoders are a specific type of neural network which are trained in an unsupervised or semi-supervised manner to reconstruct the input.[30] Due to the presence of bottleneck layers (the latent space), the data is compressed and the autoencoder is forced to keep the most informative features of the data.

We have attempted to develop an autoencoder to compress the Enformer output. We trained the autoencoder on the delta values (see figure 3a, Alt-Ref), thus taking all 5.313 genomic tracks in their totality as input. In this way the initial input of 5.313 x 896 = 4.760.448 values will be compressed into a certain number of encoded values, depending on the autoencoder architecture.

To apply an autoencoder model to this problem we build different autoencoder architectures, varying in number of layers, types of layers, and sizes of the latent space. Three architectures that we tried are discussed here (AE1 to AE3. The architecture of AE1 is shown in figure 6. See methods for the other architectures). We applied a 1D convolution to the genomic tracks, with the different genomic tracks as channels.

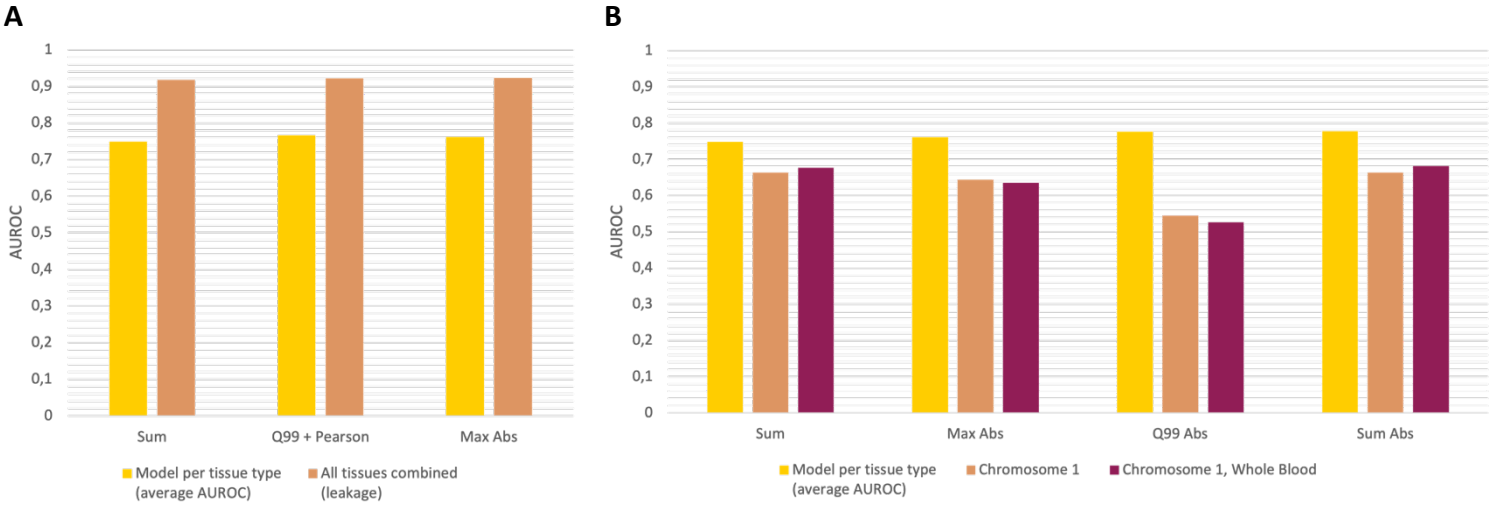


Figure 5. Classifier performances when training a single classifier on eQTLs of all tissue types. A) The average AUROC of the 49 classifiers is compared to the AUROC value of the single classifier trained on all eQTLs. The problem of leakage causes an increase in classifier performance, with the AUROC of the classifier trained on the combined tissues being 0,92. Also, there is no difference in classifier performance between the different scoring methods. Q99 + Pearson is the scoring method where the 99th quantile and the Pearson correlation values are combined, resulting in 2 x 5.313 features per variant. B) The average AUROC of the 49 classifiers is compared to the AUROC value of the single classifier trained on the eQTLs of all chromosomes except for those on chromosome 1 and tested either on the eQTLs on chromosome 1 (orange) or only the whole blood eQTLs on chromosome 1 (dark red). Leaving out chromosome 1 solves the leakage problem, but model performance of the combined classifier is lower than the average of the classifier per tissue.

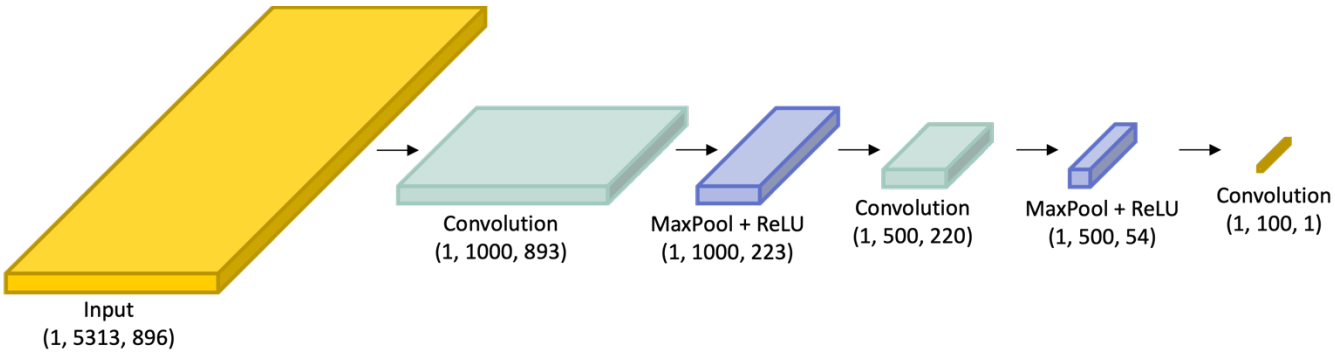


Figure 6. The architecture of the encoder part of AE1. The first half of the architecture (encoder) of the first autoencoder (AE1). The decoder part is the exact mirror image of the encoder part. The details of the autoencoder architecture can be found in the GitHub repository (see methods).

Training the autoencoders resulted in an initial decrease in loss, but training stagnated after only a few batches (supplemental figure 3), even though the model was far from optimized (the loss of the recreated samples was higher than the loss of an array of zeros). We hypothesized that the initial model (AE1) was too small to be able recreate the data. Therefore, we increased the model size (both in number of layers and in the number of values in the latent space, see AE2 in methods), but this did not increase model performance. Even the largest model would not o verfit when trained on a single sample for 2000 epochs, meaning that this autoencoder architecture was not sufficient to encode the Enformer output. One possible reason for this problem is that the autoencoder was reducing the dimension of the genomic track from 896 to 1, while the 2000 values of the latent space where in the channels dimension (see AE2 architecture). Therefore, the next architecture that we tried to train had as its latent space all the values in the genomic track dimension (by applying padding at the convolution layers) and 5 channels. However, even this did not result in the autoencoder being able to reproduce a single sample after training on it for 2000 epochs (see supplemental figure 4). Due to limited time, we were not able to develop a working autoencoder for this problem, but we do think this is a promising route to take (see discussion).

Discussion

We developed new scoring methods to compress the output of Enformer to use it for classification of noncoding variants. To see if the scoring methods are also increasing classifier performance when used for other classification tasks, our method proposed in this paper should also be tested with other truth sets. For example, as truth set the noncoding variants from the ClinVar database could be considered. However, at this point the number of noncoding variants in ClinVar is on the lower side. Kircher *et al.* identify this sparsity of pathogenic noncoding variants as a major bottleneck in the development of interpretative methods for noncoding variants.[20] Other reasons for applying our method to a different truth set are the limitations that are inherent to eQTL experiments, which are shown by the study of Mostafavi *et al.*.[29]

When training a single classifier on the eQTLs of different tissue types one should consider the tissue types the eQTLs originated from. The tissue type label should be implemented in the training. This can, for example, be done by building the classifier in such a way that it predicts for each tissue type whether an eQTL is positive of negative, thus predicting 49 labels per eQTL. It remains to be studied whether combining the different classifiers for each tissue into a single classifier for all tissues increases classification performance. As French and Edwards observe, a major challenge in predicting the effect of noncoding variants is that this effect can very much depend on the tissue type the variant is located in[7], which pleads for a tissue-specific approach and discourages training on eQTLs of multiple different tissue types simultaneously. However, since the Enformer output contains genomic tracks that are specific to certain cell types (over 600 different cell types, see methods), the classifier that uses these as input features will be able to predict tissue type-specific effects.

Concerning the training of an autoencoder, because truth sets of interest in general will not have enough samples to train a neural network, we propose to first train the autoencoder in an unsupervised

manner on the Enformer output of random variants (e.g., from the 1000 Genomes Project[31] or gnomAD[32]). Then, subsequentially, the model can be finetuned by training on the samples from the truth set, possibly in a (semi-)supervised manner.

There are numerous possibilities of moving this data compression problem forward. One example is the application of contrastive learning, in which the model is trained in such a way that samples of the same class lie close to each other in the embedding space, while the distance within the embedding space between samples of different classes is being maximized (e.g., by applying triplet loss[33]). Another way of training the compression model in a supervised manner is considering the classification loss when training the autoencoder. The classification loss can be used as direct feedback to the training of the autoencoder.

When classifying variants in a real-life scenario, the unseen set of variants that you want to classify will highly likely be unbalanced. For example, when you sequence the genome of a patient, only a small fraction of the variants encountered will influence gene expression. It is important to account for this unbalanced data in the development of a variant classifier, by making the test set and optionally also the training set unbalanced (see common ML pitfall 5 in Whalen *et al.*[28]).

In conclusion, applying the Enformer algorithm to the problem of determining the effect of noncoding variants is a promising way forward. We show that the best way of handling the Enformer output for classification is applying the Q99 Abs or the Sum Abs scoring method. Additionally, we propose ways of continuing this research to gain more insight into the role of noncoding variants in human disease and to get one step closer to treatment.

Methods

To work with the Enformer algorithm we used the scripts developed by the authors and additional scripts that we wrote ourselves. For an explanation on how the Enformer scores and subsequent compressions were obtained and all necessary scripts, we refer to the GitHub repository: https://github.com/NimrodeWit/UMCInternshipEnformer. A description of all the genomic tracks can be found on https://raw.githubusercontent.com/calico/basenji/master/manuscripts/cross2020/targets_human.txt. The eQTL dataset was downloaded from https://console.cloud.google.com/storage/browser/dm-enformer/data/gtex_fine.

The scoring methods that were tested are the following: Delta (subtract the reference from the alternative tracks), Sum (of the Delta values, per track take the sum of the values), Max Abs (take the absolute of the Delta values and select the maximum value), Q99 Abs (take the absolute of the Delta values and per track take the 0.99 quantile of the values), Sum Abs (take the absolute of the Delta values and per track take the sum of the values), SADR, SAX, SAXR, SAR (see scripts in the Basenji repo), Pearson/Kendall/Spearman correlation (for each genomic track calculate the correlation between the reference and alternative track), Q*n* (take the *n* quantile value of Delta). In case of combinations the values of the two scoring methods are combined, resulting in 10626 values per variant.

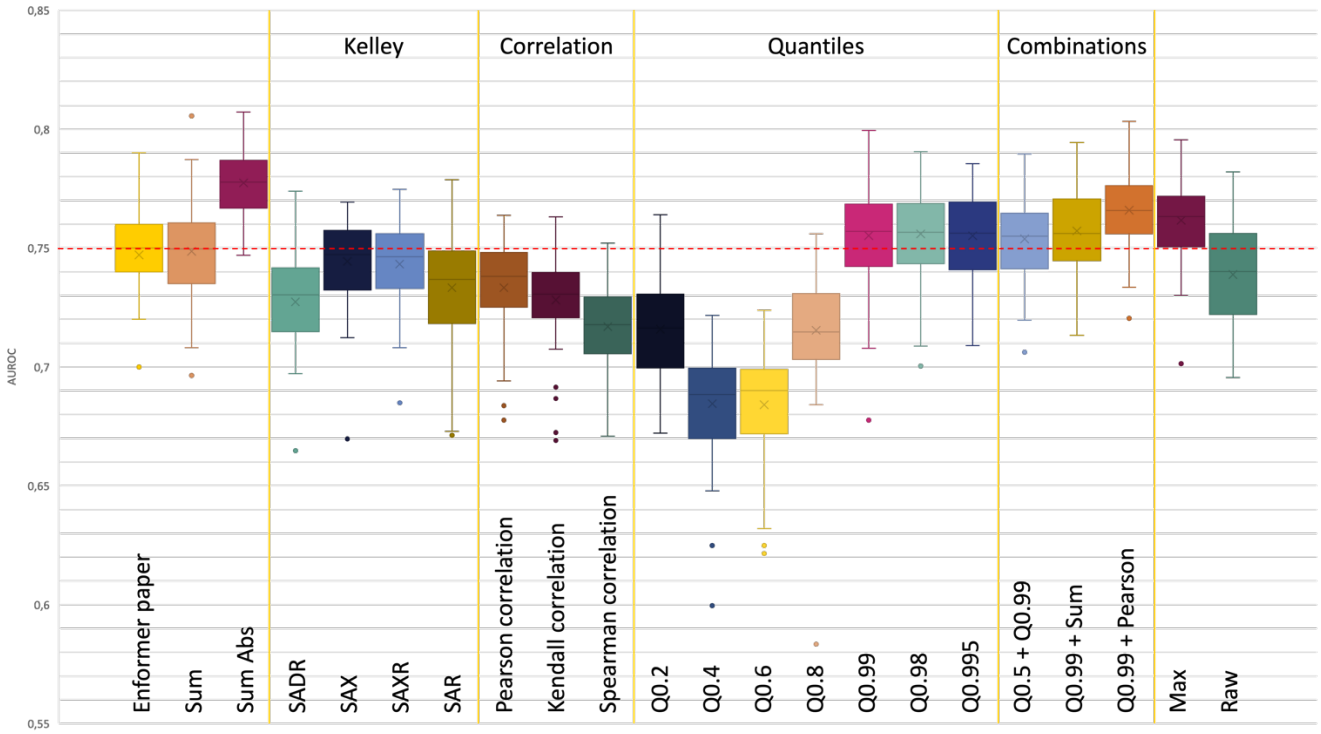
The random forest, k-nearest neighbor, support vector, and multi-layer perceptron classifiers were trained with scikit-learn's default hyperparameters, unless stated otherwise. The AUROCs that were used as a measure of classifier performance are the average AUROCs of a 10-fold cross validation. Except for the classifiers that were tested separately on eQTLs from chromosome 1, where the AUROC value is just the AUROC value that results from predicting the samples of the test set.

For the architectures of AE1, AE2 and AE3 see the AE*_architecture.py scripts in the GitHub repository.

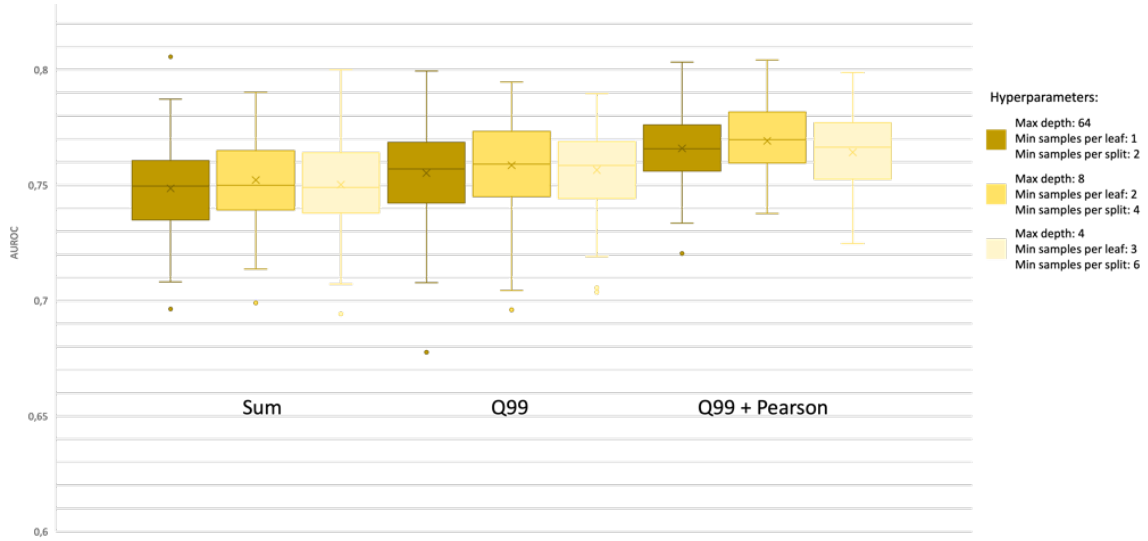
Acknowledgements

We thank dr. ir. Jeroen de Ridder and UMC Utrecht for facilitating the research and providing the resources; Brent Pedersen and Lucía Barbadilla for supervising; and all other members of the De Ridder group (UMC Utrecht, Center for Molecular Medicine) for giving feedback and advice.

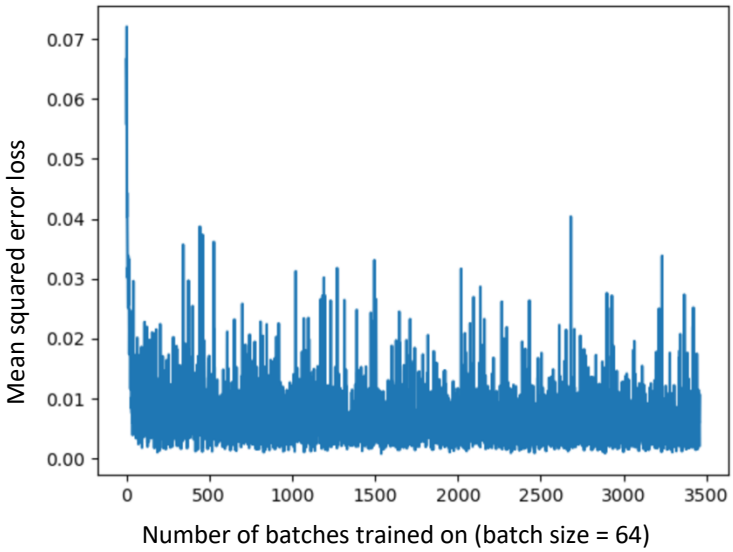
Supplemental figures



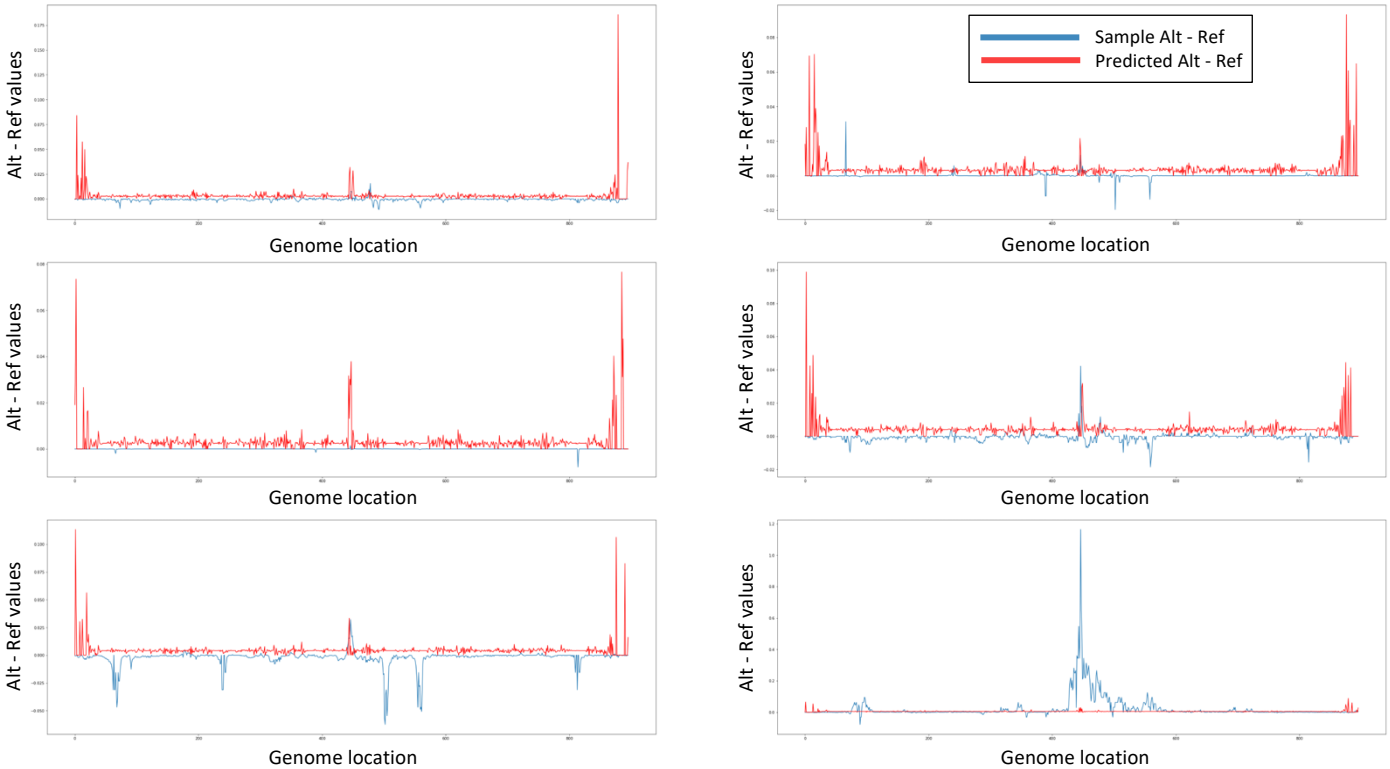
Supplemental figure 1. Classifier performances for all scoring methods. The performances of the classifiers when applying different scoring methods. 'Enformer paper' are the values obtained from figure 3d of ref. [16], Sum is our attempt to recreate these values by applying their Sum method on the same data. The Kelley methods (SADR, SAX, SAXR, SAR) are scoring methods that were present as optional methods in their scripts (see sonnet_sad.py in https://github.com/NimroddeWit/UMCInternshipEnformer). The Q scoring methods are different quantiles, and for the combinations the values of two scoring methods were combined, resulting in 2 x 5.313 values per variant. Max is the same as Max Abs and Raw is the same as Delta. See methods for a more extensive description.



Supplemental figure 2. Random forest classifiers are robust to hyperparameter change. Different hyperparameter settings were tried for the random forest classifiers. Changing the hyperparameters max depth, min samples per leaf and min samples per split did not severely affect classifier performances.



Supplemental figure 3. A typical loss curve for training the autoencoders. The loss typically decreased during training on the first few samples but remained at the same level for the rest of the training. The spikes in the loss curve are caused by the fact that training is done in batches,



Supplemental figure 4. Random examples of original genomic tracks compared to recreated genomic tracks. A representative set of genomic tracks, where the blue lines represent the Alt - Ref genomic tracks from the original sample, and the red lines the version recreated by AE3. The recreated values do not match the original values, also showing spikes at the outer parts of the track.

Layman's summary (Dutch)

In de meeste ziektes speelt genetica een rol. Daarom is onderzoek naar het effect van genetische variatie belangrijk. Ons genoom bestaat uit een coderend en een niet-coderend deel. In deze studie passen we zelflerende algoritmes toe om het effect van genetische variatie in het niet-coderende deel van het genoom te bepalen. Het Enformer algoritme is een state-of-the-art algoritme wat in staat is voor elke willekeurige streng DNA te voorspellen hoe de genexpressie zal zijn. Deze voorspellingen gebruiken we om een ander zelflerend algoritme te trainen zodat het in staat is onderscheid te maken tussen genetische variatie die wel en die geen invloed heeft op genexpressie (een algoritme wat onderscheid probeert te maken tussen twee of meerdere groepen noemen we een classificeerder). Hiervoor gebruiken we een dataset met varianten waarvan het bekend is dat deze een effect hebben op genexpressie (positieve eQTLs) en varianten waarvan bekend is dat deze geen effect hebben op genexpressie (negatieve eQTLs). Voor elke variant in deze dataset verkrijgen we de voorspellingen van het Enfomer algoritme.

Aangezien deze voorspellingen een te grote omvang hebben om direct gebruikt te worden voor het trainen van de classificeerder, moeten de voorspellingen gecomprimeerd worden. Dit is eerder gedaan door Avsec *et al.* (ref.[19]), maar wij hebben deze methode verbeterd. Wanneer de classificeerder getraind wordt op onze scores presteert het beter dan wanneer het getraind wordt op de scores van Avsec *et al.*. Hun methode van scoren, Sum, zorgt voor een gemiddelde classificeerder prestatie (AUROC) van 0,748, terwijl onze methodes Max Abs, Q99 Abs en Sum Abs respectievelijk zorgen voor een gemiddelde prestatie van 0,762, 0,776 en 0,777.

Naast deze vernieuwde methodes om de voorspellingen te comprimeren, hebben we ook geprobeerd de voorspellingen op een geheel andere wijze te comprimeren, namelijk met behulp van een zogenaamde automatische encoder. Dit is een algoritme dat leert om een grote hoeveelheid waardes samen te vatten. Het is ons niet gelukt om dit algoritme werkend te krijgen en we doen voorstellen hoe dit onderzoek vervolgd kan worden.

Concluderend kan worden gesteld dat het gebruiken van het Enformer algoritme om meer inzicht te krijgen in niet coderende varianten een veel belovende weg is en in dit onderzoek presenteren we nieuwe methodes om dit te doen.

References

1. Claussnitzer M, Cho JH, Collins R, et al. A brief history of human disease genetics. *Nature*. 2020;577(7789):179-189. doi:10.1038/s41586-019-1879-7

2. Elliott K, Larsson E. Non-coding driver mutations in human cancer. *Nat Rev Cancer*. 2021;21(8):500-509. doi:10.1038/s41568-021-00371-z

3. Dunham I, Kundaje A, Aldred SF, et al. An integrated encyclopedia of DNA elements in the human genome. *Nature*. 2012;489(7414):57-74. doi:10.1038/nature11247

4. Finucane HK, Bulik-Sullivan B, Gusev A, et al. Partitioning heritability by functional annotation using genome-wide association summary statistics. *Nat Genet*. 2015;47(11):1228-1235. doi:10.1038/ng.3404

5. Stenson PD, Mort M, Ball EV, et al. The Human Gene Mutation Database: 2008 update. *Genome Med*. 2009;1(1):13. doi:10.1186/gm13

6. Feigin ME, Garvin T, Bailey P, et al. Recurrent noncoding regulatory mutations in pancreatic ductal adenocarcinoma. *Nat Genet*. 2017;49(6):825-833. doi:10.1038/ng.3861

7. French JD, Edwards SL. The Role of Noncoding Variants in Heritable Disease. *Trends in Genetics*. 2020;36(11):880-891. doi:10.1016/j.tig.2020.07.004

8. Zhang F, Lupski JR. Non-coding genetic variants in human disease. *Human Molecular Genetics*. 2015;24(R1):R102-R110. doi:10.1093/hmg/ddv259

9. Lee PH, Lee C, Li X, Wee B, Dwivedi T, Daly M. Principles and methods of in-silico prioritization of non-coding regulatory variants. *Hum Genet*. 2018;137(1):15-30. doi:10.1007/s00439-017-1861-0

10. Vitsios D, Dhindsa RS, Middleton L, Gussow AB, Petrovski S. Prioritizing non-coding regions based on human genomic constraint and sequence context with deep learning. *Nat Commun*. 2021;12(1):1504. doi:10.1038/s41467-021-21790-4

11. Lindblad-Toh K, Garber M, Zuk O, et al. A high-resolution map of human evolutionary constraint using 29 mammals. *Nature*. 2011;478(7370):476-482. doi:10.1038/nature10530

12. Harrow J, Frankish A, Gonzalez JM, et al. GENCODE: the reference human genome annotation for The ENCODE Project. *Genome Res*. 2012;22(9):1760-1774. doi:10.1101/gr.135350.111

13. Boyle AP, Hong EL, Hariharan M, et al. Annotation of functional variation in personal genomes using RegulomeDB. *Genome Res*. 2012;22(9):1790-1797. doi:10.1101/gr.137323.112

14. Kircher M, Witten DM, Jain P, O'Roak BJ, Cooper GM, Shendure J. A general framework for estimating the relative pathogenicity of human genetic variants. *Nat Genet*. 2014;46(3):310-315. doi:10.1038/ng.2892

15. Quang D, Chen Y, Xie X. DANN: a deep learning approach for annotating the pathogenicity of genetic variants. *Bioinformatics*. 2015;31(5):761-763. doi:10.1093/bioinformatics/btu703

16. LeCun Y, Bengio Y, Hinton G. Deep learning. *Nature*. 2015;521(7553):436-444. doi:10.1038/nature14539

17. Eraslan G, Avsec Ž, Gagneur J, Theis FJ. Deep learning: new computational modelling techniques for genomics. *Nat Rev Genet*. 2019;20(7):389-403. doi:10.1038/s41576-019-0122-6

18. Zhou J, Troyanskaya OG. Predicting effects of noncoding variants with deep learning–based sequence model. *Nat Methods*. 2015;12(10):931-934. doi:10.1038/nmeth.3547

19. Avsec Ž, Agarwal V, Visentin D, et al. Effective gene expression prediction from sequence by integrating long-range interactions. *Nat Methods*. 2021;18(10):1196-1203. doi:10.1038/s41592-021-01252-x

20. Kircher M, Xiong C, Martin B, et al. Saturation mutagenesis of twenty disease-associated regulatory elements at single base-pair resolution. *Nat Commun*. 2019;10(1):3583. doi:10.1038/s41467-019-11526-w

21. Kelley DR. Cross-species regulatory sequence activity prediction. *PLOS Computational Biology*. 2020;16(7):e1008050. doi:10.1371/journal.pcbi.1008050

22. Zhou J, Theesfeld CL, Yao K, Chen KM, Wong AK, Troyanskaya OG. Deep learning sequence-based ab initio prediction of variant effects on expression and disease risk. *Nat Genet*. 2018;50(8):1171-1179. doi:10.1038/s41588-018-0160-6

23. Kelley DR, Reshef YA, Bileschi M, Belanger D, McLean CY, Snoek J. Sequential regulatory activity prediction across chromosomes with convolutional neural networks. *Genome Res*. 2018;28(5):739-750. doi:10.1101/gr.227819.117

24. Agarwal V, Shendure J. Predicting mRNA Abundance Directly from Genomic Sequence Using Deep Convolutional Neural Networks. *Cell Reports*. 2020;31(7):107663. doi:10.1016/j.celrep.2020.107663

25. Vaswani A, Shazeer N, Parmar N, et al. Attention is All you Need. In: *Advances in Neural Information Processing Systems*. Vol 30. Curran Associates, Inc.; 2017. Accessed May 30, 2022. https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

26. GTEx Consortium. The GTEx Consortium atlas of genetic regulatory effects across human tissues. *Science*. 2020;369(6509):1318-1330. doi:10.1126/science.aaz1776

27. Wang G, Sarkar A, Carbonetto P, Stephens M. A simple new approach to variable selection in regression, with application to genetic fine mapping. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*. 2020;82. doi:10.1111/rssb.12388

28. Whalen S, Schreiber J, Noble WS, Pollard KS. Navigating the pitfalls of applying machine learning in genomics. *Nat Rev Genet*. 2022;23(3):169-181. doi:10.1038/s41576-021-00434-9

29. Mostafavi H, Spence JP, Naqvi S, Pritchard JK. Limited overlap of eQTLs and GWAS hits due to systematic differences in discovery. Published online May 8, 2022:2022.05.07.491045. doi:10.1101/2022.05.07.491045

30. Hinton GE, Salakhutdinov RR. Reducing the Dimensionality of Data with Neural Networks. *Science*. 2006;313(5786):504-507. doi:10.1126/science.1127647

31. Auton A, Abecasis GR, Altshuler DM, et al. A global reference for human genetic variation. *Nature*. 2015;526(7571):68-74. doi:10.1038/nature15393

32. Karczewski KJ, Francioli LC, Tiao G, et al. The mutational constraint spectrum quantified from variation in 141,456 humans. *Nature*. 2020;581(7809):434-443. doi:10.1038/s41586-020-2308-7

33. Schroff F, Kalenichenko D, Philbin J. FaceNet: A Unified Embedding for Face Recognition and Clustering. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. ; 2015:815-823. doi:10.1109/CVPR.2015.7298682