nature computational science

ARTICLES

https://doi.org/10.1038/s43588-022-00234-z

Improving lab-of-origin prediction of genetically engineered plasmids via deep metric learning

Igor M. Soares[1,3] ✉, Fernando H. F. Camargo[1,3] ✉, Adriano Marques[1] ✉ and Oliver M. Crook[2] ✉

Genome engineering is undergoing unprecedented development and is now becoming widely available. Genetic engineering attribution can make sequence–lab associations and assist forensic experts in ensuring responsible biotechnology innovation and reducing misuse of engineered DNA sequences. Here we propose a method based on metric learning to rank the most likely labs of origin while simultaneously generating embeddings for plasmid sequences and labs. These embeddings can be used to perform various downstream tasks, such as clustering DNA sequences and labs, as well as using them as features in machine learning models. Our approach employs a circular shift augmentation method and can correctly rank the lab of origin 90% of the time within its top-10 predictions. We also demonstrate that we can perform few-shot learning and obtain 76% top-10 accuracy using only 10% of the sequences. Finally, our approach can also extract key signatures in plasmid sequences for particular labs, allowing for an interpretable examination of the model's outputs.

Genetic engineering and synthetic biology are fast growing areas of biotechnology. We are now able to transform organisms in highly efficient and sophisticated manners. As this biotechnology becomes more widespread, it is vital that we attribute genetically engineered organisms to their makers or lab of origin. This will prevent plagiarism, encourage responsible development and allow designers to gain due credit, and could be used as an approach to hold genetic engineers/designers accountable for their work. Tools for attributing this biotechnology to their owners, often referred to genetic engineering attribution (GEA), have only recently become sufficiently well performing[1–3]. When making design choices for nucleic acid sequences, an engineer will impart a design signature which could be detectable by GEA methods. Powerful methods could, in principle, identify the true designer of a biological sequence and hence be excellent tools for accountability.

Several approaches to GEA have now been proposed based on predicting the lab of origin of plasmid sequences from the Addgene[4] data repository. The performance of these approaches has quickly improved, rising from 70% top-10 accuracy[2] to 85% top-10 accuracy[1,3] in recent years. Nielsen and Voigt[2] used convolutional neural networks (CNNs), Alley et al.[1] used recurrent neural networks (RNNs) and Wang et al.[3] used a pan-genome approach, suggesting that a variety of methods could perform well on this task. However, further improvements are still possible since these approaches make other downstream tasks challenging and require many training instances to perform well. In this paper, we first improve upon the use of CNNs by employing circular shift augmentation rather than the typically used reverse-complement augmentation and applying byte pair encoding (BPE) to encode the plasmids. With these changes alone, we created a model that surpasses the current state of the art. However, interpretation is also valuable for the goal of genetic forensics. Hence, we propose, here, a training methodology for such CNNs by using metric learning[5] to teach the model to extract embeddings from DNA sequences while simultaneously learning embeddings of known labs. Our method improves over the state of the art by 5 percentage points and allows clustering of sequence and labs as well as other downstream tasks. Our approach also allows us to perform one-shot learning[6], which provides the ability to predict lab associations with only one training instance. Furthermore, we developed a method to help identify important tokens inside the sequence. We can extract design signatures from our model using integrated gradients, allowing us to interpret the model outputs and highlight possible relevant parts of the sequence for further investigation.

We evaluate our approach on a dataset from the Addgene repository containing 81,834 DNA sequences along with minimal phenotypic information: antibiotic resistance, copy number, growth temperature, growth strain, selectable markers and in which species (Methods). The plasmids are designed by 3,751 different labs and grouped into 1,313 categories, along with a single additional category to represent 'unknown engineered' (Methods). We measured the solutions using accuracy and the top-10 accuracy metric. Top-10 accuracy means that the model must rank the correct lab of origin within the ten most likely labs. Ranking often requires a slightly different approach to classification, and so we developed a metric learning approach[5] (more specifically, triplet networks[7]). We begin by outlining our method, demonstrating that it improves over the state of the art. We then demonstrate that the embedding approach allows us to perform other tasks of interest and finally demonstrate how we can perform one- or few-shot learning (FSL)[8–10] with our approach.

Results

Metric learning model and model evaluation. Our proposed method uses (deep) metric learning[5], where a distance function between objects is learnt. Here, this can be thought of as learning the similarity between plasmid sequences and labs. The result is an embedding where distances between sequences have their similarity preserved. We use deep learning, in particular a CNN-based approach, to extract embeddings of DNA sequences while learning the embeddings of the known labs. To demonstrate that using deep metric learning indeed provides an advantage, we also developed a regular classifier with a similar architecture to compare with our deep metric learning approach.

[1]Amalgam, Chicago, IL, USA. [2]Oxford Protein Informatics Group, University of Oxford, Oxford, UK. [3]These authors contributed equally: Igor M. Soares, Fernando H. F. Camargo. ✉e-mail: igor.muniz.ims@gmail.com; fernando.camargo.ai@gmail.com; adriano@amalgam.ai; oliver.crook@stats.ox.ac.uk

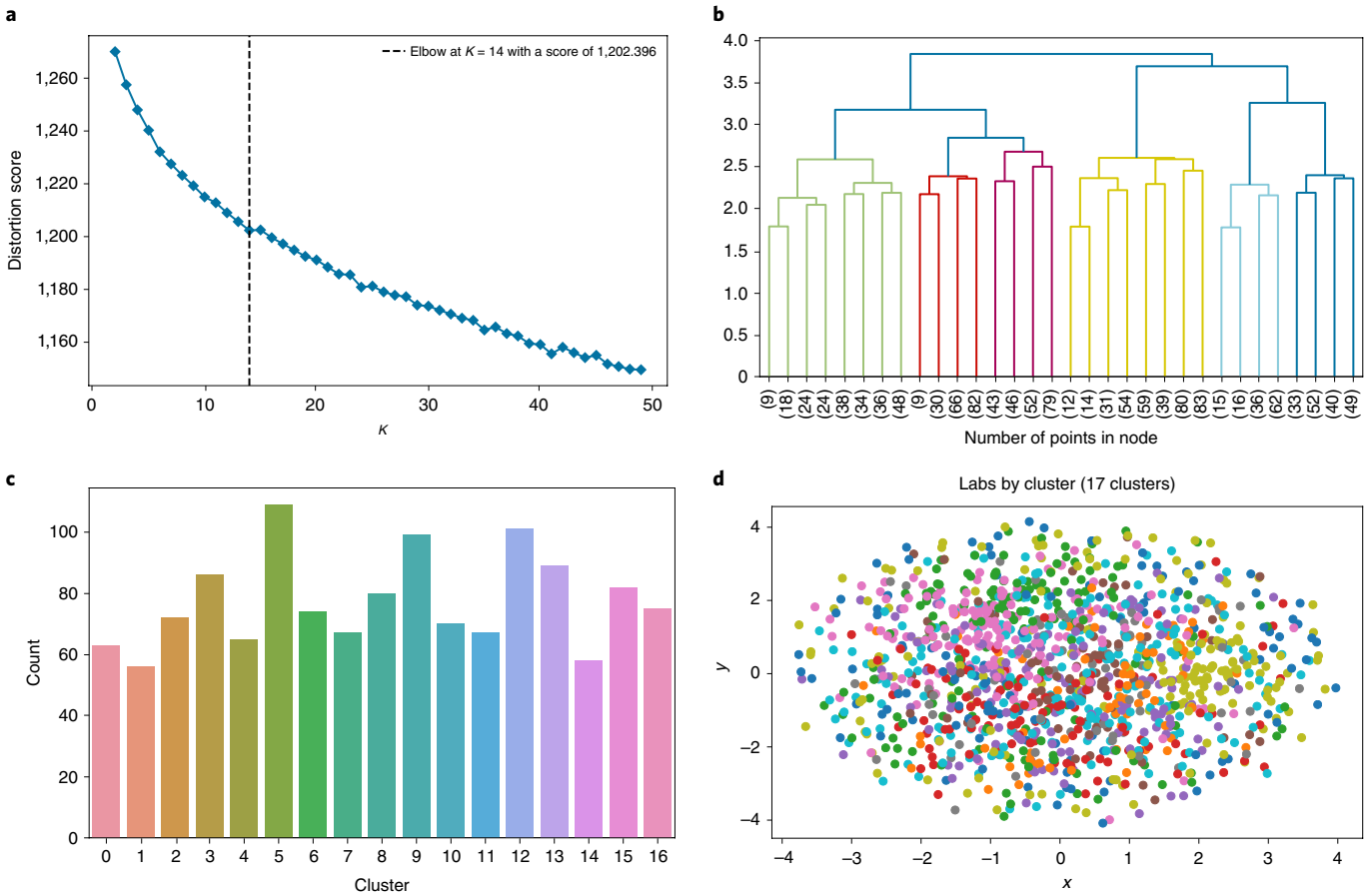


Fig. 1 | Embeddings analysis. a, The application of the elbow method by plotting the explained variation versus the number of clusters to highlight the elbow in the curve (vertical dashed line at $K=17$). **b**, A hierarchical clustering dendrogram, revealing the different points to split clusters further. Note that some labs can be easily differentiated, while some are very similar according to Euclidean distance. **c**, Number of labs per cluster, using the optimal number of 17 clusters given by the elbow method. In general, the clusters are very similar in size, but some of them have a substantial difference from the average. **d**, The labs (using the same number given by the elbow method) in a 2D space (x and y coordinates) after compressing information using t-SNE[65] (colors represent clusters).

Each model is composed of a CNN with multiple kernels of differing sizes[11]. The CNN is used to extract features from the sequence, which are then concatenated with the phenotypic metadata of that plasmid sequence. The key difference between our two proposed models can be understood by considering the final layers. The classification model's final layer has a softmax activation function, resulting in each lab's being given a probability vector associating it with each lab. Instead, our metric learning approach passes these features through a dense layer that generates our sequence embedding. In parallel, we have an embedding layer that learns the lab embeddings. The principal advantage of our metric learning approach is that, once it has been trained, it can extract the embeddings of any DNA sequence, allowing us to group or cluster new sequences to existing labs based on similar characteristics.

Our classification model is trained using regular supervised learning. However, our metric learning approach is trained differently. Specifically, our metric learning method employs the idea of triplet networks[7]. Here, we create a triplet (anchors, positives and negatives) as part of the model training process. Our model is anchored around the DNA sequences, which are thus the anchors in this approach. The positive object in this scenario is the true lab of origin, while the negative object is some other lab. To be concrete, suppose that s_1 is a plasmid sequence made by the Church lab, then a possible triplet would be (s_1, Church lab, Voigt lab). Hence, the goal of our approach is to generate embeddings in which the DNA sequences are near their labs-of-origin but far from the sequences of other labs. Extended Data Fig. 1 better illustrates this process.

Using the Addgene dataset (Methods), we trained each approach to perform DNA sequence prediction to 1 of 1,313 laboratories or 'unknown engineered' (Methods). The dataset has a total of 81,834 DNA sequences, from which 18,817 were separated for testing. During training, we split the data into 85% (53,564) for algorithm training and 15% (9,453) to validate convergence and avoid overfitting[12,13].

Training and inference methodology. Genetic sequences and phenotype information are input into our model. All sequences are processed in order to better demonstrate the characteristics of each plasmid and improve the model's ability to identify patterns (Extended Data Fig. 2a). We then use CNNs as the base model for our two approaches. In this model, convolutional operations extract information present in the sequence based on a fixed kernel size. Our method uses convolutional structures with different kernel sizes in parallel, simulating the observation of sequences by pieces of different sizes. The information extracted by each structure is aggregated and added to the phenotype information, and then assigned to one of the laboratories (Extended Data Fig. 2b).

The difference between the two approaches is the training and output of the model. The standard approach treats the genetic attribution problem as a classification problem where a softmax layer is

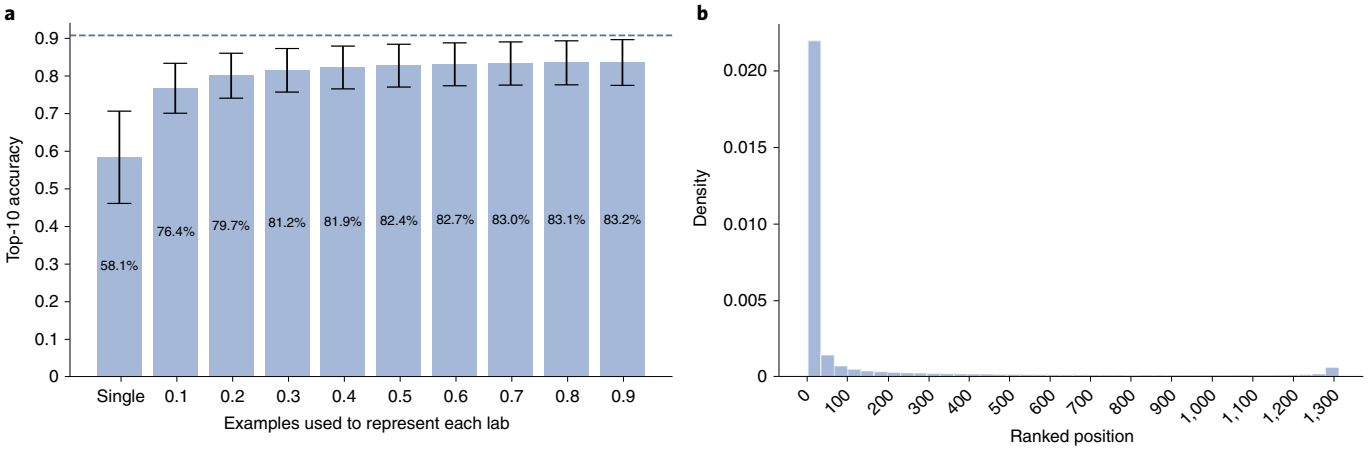


Fig. 2 | FSL results. a, Mean ± s.d. of top-10 accuracy across all experiments (1,000 runs). Note that all the labs in the dataset had a chance of being left out of the training more than once in these experiments. The first bar refers to the extreme case where we pick a single plasmid to represent the lab. The other bars refer to picking a percentage of the plasmids to represent the lab and using the rest to evaluate. It is worth noting that the number of plasmids per lab varies a lot. Some labs have hundreds of examples while other have less than ten. This means that 10% might be as little as a single example in some cases. For reference, the mean ± s.d. of each percentage are as follows: 10%: 5 ± 18, 20%: (9 ± 35), 30%:(13 ± 53), 40%: (17 ± 71), 50%: (22 ± 88), 60%: (26 ± 106), 70%: (31 ± 124), 80%: (35 ± 142), 90%: (39 ± 159). The dashed line shows the top-10 accuracy when retraining the model. **b**, Histogram of the ranked position of the lab of origin when using a single plasmid to represent it. In most cases (79%), a single plasmid is enough to rank the lab of origin at least within the top 100 (given the labs in the dataset), while in half of the cases it was within the top 50. The variance can be high because the chosen plasmid may not be representative of the lab.

applied to determine the probability that the sequence belongs to each of the laboratories seen by the model in the training phase. On the other hand, a metric learning approach (triplet network) determines how far the features representation of a new sequence is from the sequences cluster of a laboratory in the base. Smaller distances indicate greater similarities between the features of a sequence and a lab of origin (Extended Data Fig. 2c).

Before training, all sequences from a given laboratory are grouped according to their Levenshtein distances. We do not use sequences from the same group in training and validation at the same time, ensuring that sequences that are too close do not cause leakage in training and overfitting of the model (Extended Data Fig. 2d). DNA sequences are compressed by using the BPE algorithm[14]. It works by looking for common patterns in the sequence and unifying them into tokens, thus increasing the vocabulary while reducing the sequences' size (Extended Data Fig. 2e). Since plasmids are circular sequences, we randomly shift the starting point of the sequence, increasing our number of training data. This method is performed 'online' during sequence loading and preparing for network entry. After all this processing, we limited the sequence size to 1,000 characters to optimize the training time and convergence capability of the algorithm (Extended Data Fig. 2f).

Improving predictions with the metric learning model. Nielsen and Voigt[2] developed a deep learning model by applying CNNs. The network was trained on the Addgene plasmid dataset and independently verified in ref. [1]. In brief, their approach can be summarized as follows: First, DNA sequences were one-hot encoded, then used as input to a network composed of one convolutional layer of 128 filters, a max-pooling operation and two dense layers. While they demonstrated that machine learning can be used for this task, this seminal approach was hopeful and achieved an accuracy of only 48% with a top-10 accuracy of 70% in predicting the lab of origin.

More recently, Alley et al.[1] proposed deteRNNt, an RNN-based model. The main insight of this approach was to treat the DNA sequence as a text problem, using techniques from the field of natural language processing (NLP) to extract features from the sequence. They tokenized the sequence using BPE, generating larger tokens and decreasing the size of the sequence. These tokens then served as input to a word embedding layer[15] followed by RNNs[16,17]. The authors showed that their approach achieves 84.7% top-10 accuracy.

We also compared with BLAST[18], using our test set of 18,817 samples. Despite being a relatively simple tool that employs no modern machine learning and simply finds similar local regions between sequences, it was able to predict source labs, achieving 76.9% top-10 accuracy in our tests and thus outperforming the approach of Nielsen and Voigt[2].

PlasmidHawk[3], a recently launched tool, uses Plaster[19], a state-of-the-art pan-genome algorithm, to construct a synthetic plasmid resulting in a set of sequence fragments. It then aligns the original plasmid to the synthetic one and makes comparisons in order to match fragments with the plasmid. This method has so far outperformed other machine-learning-based methods by obtaining 75.8% accuracy and 85% top-10 accuracy, while employing no machine learning.

Here, we find that our metric learning model alongside our training methodology improves the current state of the art for attributing the lab of origin to an engineered DNA sequence, achieving 75.8% accuracy and 90.39% top-10 accuracy in a test set of 18,817 entries. For the classic approach of a classification model that predicts an input sequence's probability of coming from any of the possible labs seen during training, our methodology also surpasses all previous methods, reaching 76.33% accuracy and 89.36% top-10 accuracy. These methods represent an absolute improvement of 4–5 percentage points over the current state of the art in terms of top-10 accuracy. Meanwhile, our metric learning approach provides an improvement over a simple softmax-based method using a similar CNN architecture by 1 percentage point. Extended Data Fig. 1 highlights the main idea behind using metric learning to predict the lab of origin of engineered plasmids. For all experiments, the data were partitioned into training, validation and test sets ('Data splitting and results' section), and Supplementary Table 1 summarizes all our results.

Analyzing the effects of different methods. A commonly used method for encoding DNA sequences is the one-hot encoding scheme[20,21]. However, following recent advances in the field of

engineered plasmid attribution, we did not use the one-hot method to encode our sequence as it encodes only at the character level (nucleotides) but does not provide information on combinations. We performed experiments with the different encoding techniques, that is, BPE and one-hot encoding, with the same triplet model, limiting the encoded sequence length. For the one-hot encoding scheme, we used 4,000 characters as a limit, as this method does not tokenize characters and has worse results with lower limits. Supplementary Table 2 presents the difference in results when using BPE versus one-hot encoding. Using BPE is essential for better accuracy in our method and also enables a faster training time.

Further, to decrease the model's complexity, we also limited the sequence to 1,000 tokens. As plasmids are usually extensive sequences, they can cause difficulties in model convergence and considerably increase the training and inference times. To avoid these problems and find a good trade-off between limiting the sequence and not losing too much information, we performed experiments while limiting the sequence to different lengths. We found that limiting the sequence to 1,000 tokens after applying BPE reduces the computational cost and improves the results. The results of these tests are presented in Supplementary Table 3. The disadvantage is evident in both the results and performance of the one-hot encoding scheme. This method does not allow us to limit the sequence, which considerably increases the training time while also causing complexity in pattern recognition, leading to lower accuracy. Although all these experiments were performed with the triplet learning model, we expect similar results for softmax, since the issue lies in the sequence's pre-processing.

We also compared the differences in the training time and inference time between the softmax model and the triplet model (Supplementary Table 4). As one might expect, the softmax model achieved more iterations per second because of the lower complexity of calculating the system loss. The big difference appears in the total training time, showing that this model converges faster and therefore requires fewer training epochs compared with triplet learning. As GEA is generally an investigation process, the difference in inference time of the two models is not substantial, but for near-real-time classification purposes, use of the softmax approach should be considered.

Using triplet networks to embed the DNA sequences. We train triplet networks to learn embeddings (vector representations with preserved distance) for both labs and DNA sequences. These vectors live in the same vector space, which allows us to compare them in a variety of ways. For example, we can compare the distance between two labs, between a lab and a sequence or between a sequence and another sequence. This allows us to perform tasks other than ranking the possible labs-or-origin of a given DNA sequence.

Clustering is one common application that can provide insights[22–24]. Many labs will share information about design techniques, will have been mentored or trained in another lab or will be directly collaborating. However, these relationships and similarities are not always known. Even though these similarities are not directly apparent through labs' embeddings themselves, we can also examine which DNA sequences of a particular lab are more similar to those of other labs, if necessary. Once we get embeddings, we can apply any of the well-known clustering techniques. Figure 1 showcases common clustering techniques that an analyst could apply to the labs, something that is possible due to our triplet approach and was not presented in any previous work. Figure 1a shows the application of the well-known elbow method to find a good number of clusters while avoiding overfitting. It works by plotting the explained variation versus the number of clusters and picking the elbow in the curve. As shown, we could group the labs into various numbers of clusters, but 17 seems to be the optimal number (with a slight margin), which is why we use this number in Fig. 1c,d.

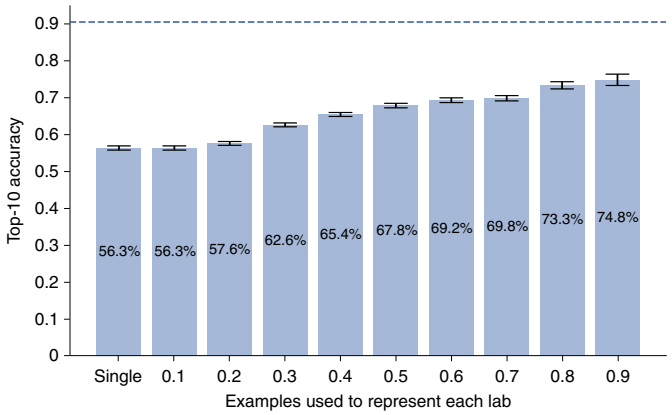


Fig. 3 | FSL top-10 accuracy for unknown labs (two to ten plasmids). Mean ± s.d. of top-10 accuracy across all experiments (1,000 runs), run only on labs originally grouped together for having too few plasmids (two to ten).

Figure 1d demonstrates a vector relationship between the laboratories, making it possible to group them to create a hierarchy (Fig. 1b) and reduce the search space. It is worth noting that an analyst would apply these techniques with a specific goal in mind, for example, to study the relationship of a target lab with others. For example, these clusters may link researchers geographically or collaboratively, which may help genetic forensics to identify groups of labs even if a single lab is not identifiable.

FSL. Machine learning algorithms typically require large quantities of training data, and in many applications, this makes it challenging when new classes are added. Sparse training data for these classes can result in poor-quality predictions. In the case of GEA, some new labs may only have a single or few appropriate training instances. Training machine learning algorithms to perform well in this task is known as FSL[8–10]. Based on knowledge already acquired by a model trained in a similar task, the FSL method can generalize to a new task using just a few samples, without the need to retrain the model. Our proposed method can be straightforwardly adapted to the FSL situation. Since embeddings are the representation of features, it is possible to use them as the input for other machine learning algorithms or to calculate approximations between the embedding of a new sequence and all previously observed sequences. Thus, in a scenario of new laboratories with few sequences, one can store these few samples and perform the FSL process by calculating the distance between an unknown sequence and the embeddings of these laboratories. It is sufficient that only one of the stored samples has characteristics similar to the unknown sequence for the prediction to be carried out.

To test the ability of our approach to perform FSL, we undertook the following experiment: We trained our method 100 times, each time removing all the plasmids from 50 different labs from the training set. For each lab that was left out, we picked a random sample of plasmids to generate the embeddings that represent that lab. All remaining plasmids were then used to evaluate our model. In the extreme case (also known as one-shot learning), we used a single plasmid to represent each lab while testing with all the others. Figure 2a shows the mean and s.d. of the top-10 accuracy of our model when using different sample sizes to represent each lab, revealing that, the larger the sample, the higher the top-10 accuracy. However, there are diminishing returns as the sample size is increased. The results show that, in general, only a few representative examples are needed for high-accuracy predictions. We see that our approach obtains better top-10 accuracy than the previously published CNN approach of Nielsen and Voigt[2] while only

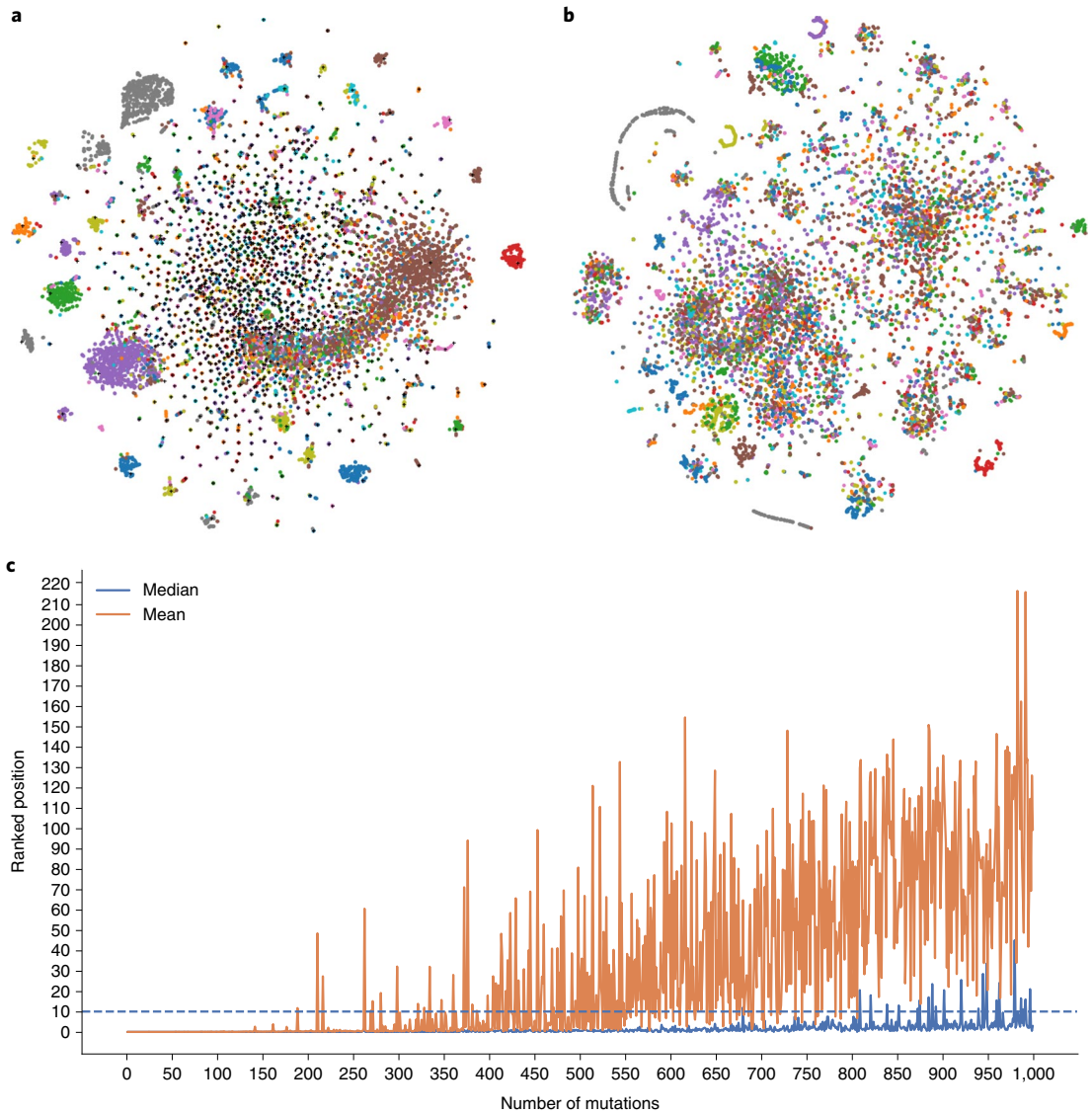


Fig. 4 | Two-dimensional visualization of models and effect of point mutations. a, Each circle represents a DNA sequence, with its color highlighting its lab of origin. Each plus sign represents a lab. We project all of them from 200 dimensions to 2D using t-SNE for presentation purposes. The DNA sequences group together very well with their lab of origin. Some labs display very similar DNA sequences, while others are slightly dispersed, highlighting the differences between large and small labs. **b**, t-SNE visualization of the 3,072-dimensional last hidden layer of the softmax model. Although clusters can be seen in this feature map, they are not as well defined as in the triplet model. **c**, Effect of point mutations on the triplet model. The mean and median in ten runs of the position of the correct lab in the model prediction ranking is shown, with the number of mutations ranging from 1 to 1,000. The horizontal dashed line highlights the 10th position, as top-10 accuracy is our main metric.

using 10% of the training data and without requiring retraining of our model. Figure 2b and Supplementary Table 5 show the rank of the lab of origin when taking a single plasmid to represent each lab. We observe that, if we want to ensure that the true lab is within our selected sample with probability 0.9, we need to include around 685 labs. In other words, we can rule out around 50% of the labs as the origin with a confidence of 90% when there is only a single plasmid available for that lab. Furthermore, these results demonstrate that, if an analyst can pick a single representative plasmid of a lab by using domain knowledge, our approach has a good chance of attributing an unknown plasmid to that lab without any need to retrain the model.

So far, all work on GEA has ignored labs with few samples, grouping them into a single class called 'unknown engineered' because such small amounts of data are often harmful to machine learning models and also provide weak information for any pattern-recognition technique. We thus decided to examine this meta-class using our FSL approach. Figure 3 shows the extreme case of applying FSL to those rarest labs. We removed those labs (with two to ten plasmids) from the training data and then evaluated the model on them with FSL. We observed a marked drop in performance when examining these classes. However, this was expected given the small amount of data used for prediction in FSL (one to nine plasmids) and the vast number of classes available for the prediction.

Model interpretability and robustness. Interpreting deep learning models provides valuable information, such as understanding how the model works and the relative importance of features within the data. It can also reveal why some approaches work better than others, which can be used to further improve the model. However, interpretation techniques for deep learning are still naive and are

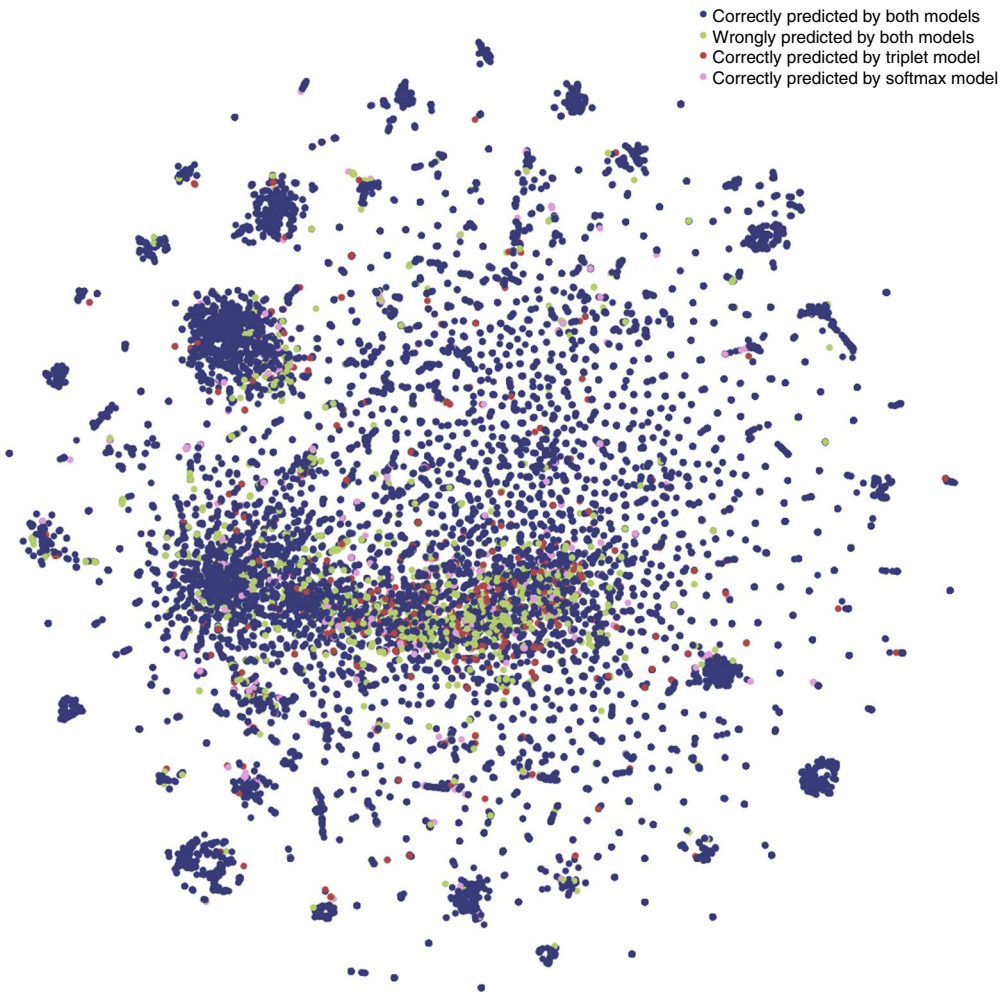


Fig. 5 | Two-dimensional visualization of t-SNE triplet network predictions. Each point (plasmid) is labelled as correctly predicted by both models, wrongly predicted by both models, or correctly predicted by either the triplet model or softmax model. Triplet embeddings are used for visualization purposes as they create better-defined clusters. The most difficult plasmids to predict (green points) are mainly centred in the chaotic region, where there are also many plasmids that are correctly predicted only by the triplet model. Future work on this task might focus on improvements for such plasmids.

an area of active research[25,26]. In this work, we focus on understanding the differences between a triplet network and a conventional classification model, how robust our model is when performing point mutation and, most importantly, which tokens (parts of the sequence) are critical for identifying a lab.

We start by visualizing the differences between the space of features mapped between the two models. For a triplet network, this space is the sequence embeddings. Meanwhile, for the softmax model, we take the output of the 3,072-dimensional last hidden layer. This layer is the concatenation between all the convolutional layers and contains all the features used by the model. The two multi-dimensional vectors are reduced to two-dimensional (2D) space using *t*-distributed stochastic neighbour embedding (t-SNE). As mentioned by Alley et al.[1], the model is more accurate when the plasmid features are more separable in the latent (unobserved) space. We observe in Fig. 4 that the triplet network model has better-defined clusters.

We then performed a deeper analysis on both model's predictions, where it was possible to see similarities and differences between the results presented. In general, the triplet learning model and the softmax model present an intersection of 94% in their results, with 16,481 of 18,816 plasmids having the lab of origin correctly positioned in the top 10. On the other hand, 1,222 plasmids were wrongly predicted by the two models. The triplet model correctly ranks 637 plasmids that the softmax model could not, while the opposite happens for 476 samples. We present the t-SNE of the test-set plasmids labelled by when they were correctly or wrongly predicted by both models in Fig. 5. It is easy to see that most of the wrongly predicted plasmids are located at the centre of the figure, which lacks well-defined clusters. It is also in this region where more samples are correctly predicted only by the metric learning, indicating that this method works better for plasmids with no prominent features.

It is also important to understand the robustness of the model, as small changes in plasmids can occur frequently. We perform random point mutations in the sequence from a lab and report the ranking of the correct lab generated by the model. Figure 4c shows the mean and median of the correct positions after ten runs for each number of point mutations (from 0 to 1,000). We found that nearly all the runs with up to 1,000 mutations predicted the correct lab within the top 10 guesses. Increasing the number of mutations makes the average of the runs become unstable, with the average of the positions being higher for all cases above 400 mutations. However, if we examine the median of the predicted positions, the median rank for the correct lab remains within the top 10, even with 800 mutations. This indicates that our model is robust to most sequence perturbations, excluding cases where these mutations affect features that are essential for the model's prediction.

We proceed to analyze all sequences to discover the importance of each plasmid feature to the model output. Unlike perturbation analysis in each sequence, here we use more recent methods that generate better insights into the interpretation of the model. Our method is based on integrated gradients[27]. The idea is to compute the gradient of the model's output relative to the embeddings token layer. This makes it possible to visualize the importance of each token for the model's prediction. After calculating the integrated gradient for all sequences, we obtain the token importance of each lab by averaging all sequences in that lab. The same process can be performed with all sequences to determine the most seen tokens in the dataset.

As seen from Fig. 6, there are some tokens that appear to be shared by all labs. When generating the token importance of a lab, we can subtract it from the most seen tokens in the dataset to obtain a relative importance. This allows us to examine those tokens and those not to be expected for a particular lab. Furthermore, we can compare the token importance of one lab versus the most distant lab from it in the embedded space. The graphs of token importance from the two labs are essentially mirrored, indicating that tokens are quite different in each case. Figure 6 show these analyses, where in the left-hand column, we plot the normalized token importance (NTI) as a function of the token, while the right-hand column highlights the sequences with the largest token importance.

We further explored this model by looking at the token importance for David Root's lab (Fig. 6b). This analysis reveals a cluster of sequences that are typical of this lab, allowing us to identify the potential design signatures. Furthermore, this lab is the furthest lab from the 'unknown engineered' category, which is a mixture of possible labs and so has poorly defined features. The fact that David Root's lab is the most different from this class suggests that it has well-defined and perhaps highly unique design choices. We note that, for the 'unknown engineered' class, the scale for the NTI is shallow and the color gradient is mostly red (Fig. 6d). This demonstrates that there is not a clear design choice or discriminating feature of this category, which is expected as it is a mixture of many possible labs (Methods). This analysis could be repeated for any of the labs in the dataset to identify key signatures or potential collaborations based on token proximity.

We next examined the use of integrated gradients for a single sequence. One of the major goals of GEA approaches is to examine plasmids of unknown origin and be able to extract valuable sequence information, leading to a correct assignment or further importation avenues to explore. Extended Data Fig. 3 shows that, using our approach, we can obtain the importance of each token within an unknown sequence. When comparing the sequence token importance with the lab predicted by the model, we see a concordant behaviour in the plots, demonstrating similarity in the highlighted features. This allows us to carefully examine the sequence features that the model is using for prediction, hence enabling secondary expert evaluation on the veracity of the prediction.

Finally, we generated the NTI for the custom sequence designed by Alley et al.[1]. This sequence combines a plasmid designed in the Omar Akbari lab (AAEL010097-Cas9, Addgene #100707, https://www.addgene.org/100707/) and another plasmid from Edward Boyden's lab (pAAV-Syn-SomArchon, Addgene #126941, https://www.addgene.org/126941/). In the analysis by Alley et al.[1], their model was uncertain how to classify this plasmid and assigned it to the category 'unknown engineered'. They then applied scanning *K*-mer analysis to find part of the sequences that increased the probability of predicting that sequence as designed by one of the laboratories, obtaining probability peaks in parts associated with the plasmids used from each laboratory. However, their method showed a low probability of prediction for the Omar Akbari lab (3%) and Boyden lab (0.3%), with superior analyses only possible when the ground-truth label was available. Here, we predict this same sequence using our metric learning model, which returned the Akbari lab (top 1) as the closest to the sequence, followed by the Boyden lab (top 2), thus demonstrating that our model successfully recognized the characteristics of both labs. Our NTI analysis (Extended Data Fig. 4) shows that this sequence has similar token importance to the two analyzed labs. We found that sequences with the highest NTI are derived from the end of the plasmid sequence located in the Ori region of the plasmid, a portion from the Akbari lab. However, the approach of Alley et al.[1] gives greater predicted probability for this region to correspond to the Boyden lab, suggesting that this helped our metric learning approach to better discriminate this plasmid's lab of origin. It is important to note that the use of the NTI interpretation method does not depend on having the ground-truth label of the sequence, because the importance is defined through the weights of the already trained neural network. However, our method is directly linked to the tokens generated by the BPE since it gives importance to each grouping of characters, a grouping that is optimized for NLP tasks but that does not necessarily have a biological meaning.

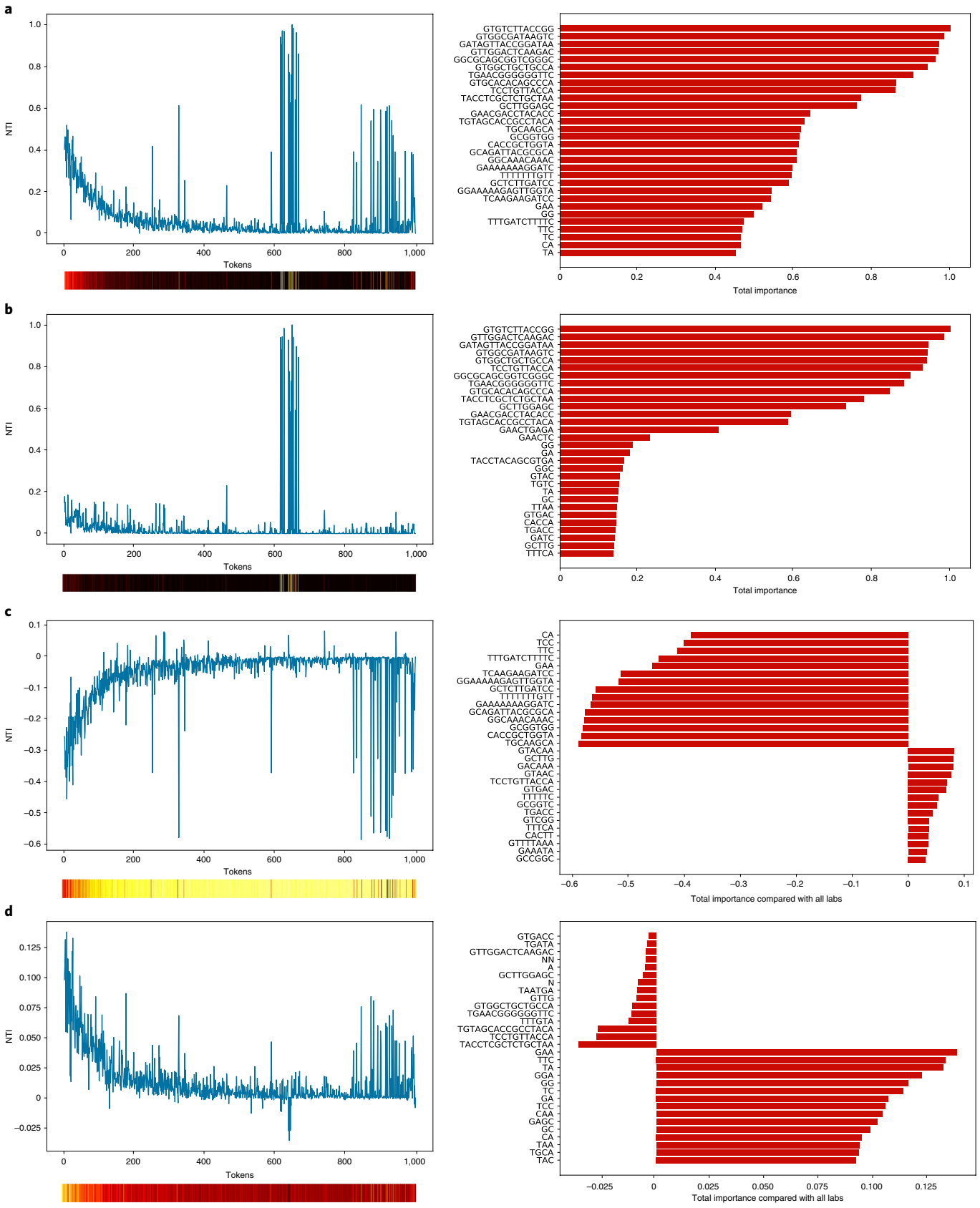
Discussion

Metric learning is quite common in other areas such as recommendation systems[28,29] but has not been applied to GEA so far. This methodology, alongside parallel convolution with different kernel sizes, improves the accuracy of our model, reaching 90.4% top-10 accuracy, a 5.4% improvement compared with the former state of the art, while offering several advantages such as creating a vector representation of labs, enabling the comparison and clustering of DNA sequences and labs-of-origin and the ability to examine design styles and the robustness to unseen labs. For example, a plasmid sequence might be too distant from known labs, resulting in low similarity values. Furthermore, we also have a particular embedding for unknown labs. If a new plasmid sequence is nearer to this embedding than to the known labs, it is possible that this sequence is from a currently unobserved lab. In contrast, a classifier model does not usually know how to handle such uncertainty. Typically, it spreads probabilities for each lab, summing to 1.0. Hence, any plasmid sequence is assigned to known labs, even if it is from a completely unknown lab.

The successful use of the metric learning approach in this work supports future applications in other DNA data-based computational problems where more recent deep learning models can also be used, such as bidirectional encoder representations from transformers[30], representing the state of the art in natural-language embedding. It is also possible to create a vector representation of DNA, not based on laboratories as described here but only on its subsequence contextual information, similar to word embeddings[31–33] in

Fig. 6 | Interpreting plasmid feature importance using integrated gradients. a, Left: the NTI for all labs in the dataset obtained by averaging the token importance of all sequences and normalizing between 0 and 1. Right: the top 30 tokens for all data. **b**, Left: the normalized token importance for David Root's lab, showing that it has similar tokens to all labs in the ID range between 600 and 700, but in some other regions it differs a lot. Right: the top 30 tokens for David Root's lab. **c**, Left: difference between the token importance of David Root's lab and all labs, highlighting which tokens make a difference for this particular lab, considering the presence or absence of a token when compared with other labs. Right: the tokens that should be observed when analyzing this lab. **d**, The token importance (left) and top 30 tokens (right) of the furthest lab from David Root's lab in the embedding space. The NTI values are practically mirrored with respect to **c**, indicating their opposite characteristics. In the color bar below the left panels, the lighter the bar, the higher the NTI value at that point.

the NLP field. Furthermore, our FSL results illustrate the possibility of identifying new laboratories with few samples or even just a single genetic sequence. Clearly, there is a trade-off between sample quantity and model accuracy, but we believe that such a methodology could be useful in extreme cases. Finally, these embeddings are also feature rich, which means we can use them as input for other



machine learning models to tackle other problems. For example, we are able to extract the defining signatures for labs and compare them with others using our approach.

Although we present a state-of-the-art classifier and a training methodology with interpretability, our approach has shortcomings. The accuracy has room for improvement, and we believe that future work should delve into more modern NLP approaches. This could include using more advanced machine learning architectures, other pre-processing methods and data augmentation techniques that would lead to better convergence and algorithm training. One future direction of work would be to adapt the model for diploid features[34]. We also believe that techniques such as transformers[35] and graph convolutional networks[36] would be good candidates for this task, since patterns inside the sequence can be considered to be contextualized, for which transformers generally show good performance[35].

Deep learning approaches rely on large amounts of high-quality data. Here, our FSL approach is designed to handle laboratories with few samples, generally classified as 'unknown engineered'. However, neither the data nor the evaluation methods for GEA are concise between different works. To better track advances in the area, future work from the biotechnology community should focus on creating more robust datasets with a specific test set for evaluation, similar to what we see in computer vision[37–40] or NLP[41–44] tasks. This will increase the reliability of the results and prevent accidental misreporting.

Finally, our interpretation method using integrated gradients highlights the most important tokens within the sequence in the model view. We show that our method does not depend on the ground-truth label, needs only one execution for the entire sequence and can help specialists in further investigations. However, the tokens generated by BPE are purely analyzed from the perspective of pattern recognition in NLP, and they may not have a biological representation. We believe that a major improvement would be to map subsequences with biological characteristics into tokens, similar to the process performed by BPE but done by a genetic engineering specialist. Future work that focuses on this will help improve interpretation techniques and the understanding of essential patterns by computer scientists who are not experts in genetic engineering. We hope that our methodology and results will stimulate new directions for forthcoming research.

Methods

Addgene dataset description and data splitting. The Addgene data[4] comprised all plasmids deposited in the Addgene repository up to 27 July 2018, a total of 81,834 entries. For each plasmid, the dataset included a DNA sequence along with metadata on growth strain, growth temperature, copy number, host species, bacterial resistance markers and other selectable markers. Each of these categorical metadata fields was re-encoded as a series of one-hot feature groups:

- Growth strain: growth_strain_ccdb_survival, growth_strain_dh10b, growth_strain_dh5alpha, growth_strain_neb_stable, growth_strain_other, growth_strain_stbl3, growth_strain_top10 and growth_strain_xl1_blue
- Growth temperature: growth_temp_30, growth_temp_37 and growth_temp_other
- Copy number: copy_number_high_copy, copy_number_low_copy and copy_number_unknown
- Host species: species_budding_yeast, species_fly species_human, species_mouse, species_mustard_weed, species_nematode, species_other, species_rat, species_synthetic and species_zebrafish
- Bacterial resistance: bacterial_resistance_ampicillin, bacterial_resistance_chloramphenicol, bacterial_resistance_kanamycin, bacterial_resistance_other and bacterial_resistance_spectinomycin
- Other selectable markers: selectable_markers_blasticidin, selectable_markers_his3, selectable_markers_hygromycin, selectable_markers_leu2, selectable_markers_neomycin, selectable_markers_other, selectable_markers_puromycin, selectable_markers_trp1, selectable_markers_ura3 and selectable_markers_zeocin

In addition to the sequence and the above metadata fields, the raw dataset also contained unique sequence IDs, as well as separate IDs designating the origin lab. Both sequence and lab IDs were obfuscated through 1:1 replacement with random alphanumeric strings. The number of plasmids deposited in the dataset by each lab was unbalanced, with many labs depositing one or a few sequences. To deal with this problem, Alley et al.[1] grouped labs with fewer than ten data points into a single auxiliary category labelled 'unknown engineered'. This reduced the number of categories from 3,751 (the number of labs) to 1,314 (1,313 unique labs + unknown engineered). In addition to issues with small labs, the dataset also contains 'lineages' of plasmids, that is, sequences that were derived by modifying other sequences in the dataset. If left unmitigated, this introduces unintended correlations between the test and validation set. To overcome this, Alley et al.[1] inferred lineage networks among plasmids in the dataset, based on information in the complete Addgene database acknowledging sequence contributions from other entries. Lineages were identified by searching for connected components within the network of entry-to-entry acknowledgements in the Addgene database. Refer to Alley et al.[1] for more details.

Data splitting and results. The data were partitioned into training, validation and test sets, with the constraints that (1) every category has at least three data points in the test set and (2) all plasmids in a given lineage are assigned to a single dataset. Following the split, the training set contained 63,017 entries (77.0%), the validation set contained 7,466 entries (9.1%) and the test set contained 11,351 entries (13.9%). To evaluate the model's performance in this dataset, we established the test set as a hold-out set, and we computed its results only once. This methodology is standard in other machine learning tasks to avoid biasing researchers during the experiment. Therefore, the reported results (Extended Data Fig. 2g) refer to this hold-out set. The other tests performed during the construction of this work were evaluated using the validation set (Supplementary Table 1). It is important to note that all our experiments were conducted on a machine with four TITAN V 12 GB graphics processing units, 128 GB of random-access memory and an AMD EPYC 7401P 24-core processor, with one graphics processing unit being used per experiment. Different computational devices may result in slightly different results from the reported ones.

While training all models, we performed a fourfold cross-validation strategy[45] for each experiment within the training set (63,017 entries). To be precise, for each hyperparameter setting, we split the data into k parts (we used $k=4$), using one of them to validate and the remaining to train, repeating this process k times. After that, we evaluated each model by taking the mean of the metrics for that experiment. This approach helps to avoid overfitting and improve generalization. It also enables us to ensemble the k models, to further improve generalization.

Grouping sequences by their Levenshtein distance. Genetic sequences from the same lab display high degrees of similarity. These sequences can make it easier to identify similar sequences in the training and validation sets. However, when training a machine learning algorithm, this may be perceived as data leakage between these sets[46], as the model does not need to learn to extract different features to identify such sequences. To ameliorate this issue, we developed a more robust model by group sequence from each lab based on their Levenshtein distance[47]. The Levenshtein formula used is shown in equation (1)

$$\mathrm{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} \mathrm{lev}_{a,b}(i-1,j)+1 \\ \mathrm{lev}_{a,b}(i,j-1)+1 \\ \mathrm{lev}_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases} \quad (1)$$

After grouping, each laboratory has N groups of sequences. We then split the dataset, ensuring that there will be sequences from the same group only in training or validation, which means they will never be present in both sets at the same time. This entire process was performed using Python and the python-Levenshtein library (https://github.com/ztane/python-Levenshtein). As this is a costly algorithm and there are thousands of sequences to be grouped together, the entire process was performed on the same machine used for training and evaluating the models ('Data splitting and results' section). This approach complements the lineage-based strategy, which also avoids data leakage.

BPE. We use the BPE algorithm[14] to process all sequences in the dataset, grouping subsequences into new tokens, increasing the vocabulary while reducing the sequences' size. The BPE algorithm first examines all the sequences in the dataset to learn how to perform the grouping. The trained algorithm is saved and used to transform the sequences during the CNN training. This last process is performed 'online'. This means that, while loading our batch of samples, we transform the batch into new, small sequences. We converted our vocabulary size from four DNA bases into 1,001 different tokens: 1,000 tokens from the new vocabulary generated by the BPE algorithm plus 1 for an unknown token. The training and inference of the BPE algorithm were performed using the sentencepiece package (https://github.com/google/sentencepiece), and, as in the 'Grouping sequences by their Levenshtein distance' section, we use Python to perform this operation.

Circular data augmentation. Machine learning models and especially deep learning models are highly dependent on large amounts of data. One of the fundamental methods for adding variance to those models, increasing generalizability and reducing overfitting[48] is data augmentation[49]. Generally, data augmentation performs transformations on the sample, considerably changing some characteristics. In this work, performing such transformations can be dangerous as it may end up modifying some parts that are essential to assign the sequence to a lab of origin. However, it is possible to take advantage of the fact that plasmids are circular and create a circular shift data augmentation process. This contrasts with a reverse complement augmentation one might use. During training, we show different versions of the same DNA sequence by shifting it circularly as shown in Extended Data Fig. 2. This approach helps the model understand that the same pattern can happen at different positions within the sequence, increasing the generalizability of the training.

We also perform test-time augmentation[50], which helps to improve the model's prediction capability. To perform this analysis, during inference we run the model multiple times. Each time, the model sees a shifted version of the sequence and makes a prediction of the same sample seen from different angles. We then take the average of the outputs (class probabilities for the classifier and embeddings for our proposed method).

CNN base architecture and training details. Both types of models are composed of a shallow CNN with multiple kernels of different sizes, as proposed by Kim[11]. The idea is that the model will learn multiple filters to extract features in an *n*-gram fashion. We used kernels with size from 2 to 12 to cover multiple lengths of subsequences that can be recognized by the model. It is worth noting that it is applied to the tokenized sequences, which already have multiple subsequences grouped into tokens. Compared with a deep CNN, this architecture cannot recombine the features in further layers but has fewer parameters to learn and a lower complexity, which helps to avoid overfitting. When compared with RNNs, this architecture can more easily deal with large sequences without worrying about vanishing gradients while also having much better computational performance. Transformers would also be an option, but they perform the best when pre-trained with a huge amount of data in a given domain.

We use this CNN to extract features from the sequence and concatenate them with the binary features provided in the dataset. The difference between the classification and triplet network models lies in the final layers. The final base structure is composed of an embedding layer, several convolutional layers in parallel with different kernel sizes and a custom dropout layer for regularization. The embedding layer has a shape of $1{,}001 \times 200$, where 1,001 is our vocabulary size and 200 is the vector embedding dimension found empirically. Its purpose is to map each token into a 200-dimensional vector containing the features representation of that token[51]. For the convolutional layers, we have a total of 12 layers in parallel, where the first layer has kernel size 1, the second has kernel size 2 and so on, until the last layer has kernel size 12. All convolutional layers are followed by a scaled exponential linear unit activation function[52] and a max-pooling operation. We concatenate the features extracted by each of them, obtaining the final representation with different windowings of the sequence. We also implemented a custom dropout layer. A standard dropout layer[53] randomly masks out parts of a tensor to regularize the neural network. However, if we did that on the embeddings before applying a similarity function, the output would be too unstable. So, we created a layer that randomly masks out the same parts of all the embeddings involved before applying the similarity function. We found this approach instrumental in regularizing our model.

The entire architecture was developed and trained using Python and PyTorch[54]. Although the training methodology is different between the two approaches, all the training details, such as the optimizer, learning rate scheduler and regularization techniques, remain the same. We use the Adam[55] optimizer together with the One Cycle learning rate scheduler[56]. This scheduler was essential to achieve a better convergence in training. Its settings were a maximum learning rate of $1e^{-3}$ and cycle execution in 200 epochs. To regularize our model and prevent overfitting, we used a weight decay of $1e^{-5}$ during training and dropped 20% of the embedded sequences using our custom dropout layer.

Triplet network learning. To generate the triplets, we use the labelled dataset to provide us with the anchor and positive. We then use a technique known as hard negative mining[57] to select the negative (an incorrect lab). This means that, rather than choosing a random lab as a negative example, we choose the most challenging one given the current state of the embeddings. Thus, in our case, it would be the nearest lab to our sequence in the latent space.

One of the most challenging parts of this work was the implementation of the algorithm to mine the negative examples during training efficiently. We could have used a library called PyTorch Metric Learning[58], which includes hard negative mining implemented per batch (it does not take the whole dataset into account while finding the negative) as well as cross-batch memory for embedding learning[59]. However, this library only supports a single entity type. Furthermore, we also have easy access to the whole lab embeddings, since we use an embedding layer. So, we re-implemented the approach for our specific needs (Supplementary Algorithm 1). It is worth noting that we implemented it using tensors to make it as efficient as possible. Our source code provides a PyTorch implementation, and it should be straightforward to implement it in TensorFlow or other frameworks.

Cosine similarity. We used cosine similarity as the metric to measure how similar the vectors were in the embeddings. Mathematically, this measures the cosine of the angle between two vectors projected in a multi-dimensional space, resulting in a value in the range from −1 to 1, where −1 indicates opposite vectors and 1 indicates equal vectors. Given two non-zero vectors of embeddings, A and B, the cosine similarity is

$$\text{Similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\| \mathbf{A} \| \| \mathbf{B} \|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}. \quad (2)$$

t-SNE and *K*-means. Throughout this work, we used two techniques to analyze and visualize the results: the t-SNE algorithm and the *K*-means cluster algorithm. The t-SNE algorithm[60] is a dimensionality-reduction technique that can reduce dimensions with non-linear relationships. It is particularly well suited to the visualization of high-dimensional complex real-world datasets. Making use of this, we are able to reduce the embeddings generated by the triplet network and visualize them in 2D space. *K*-means[61] is a clustering algorithm that attempts to organize the data into *K* clusters. The objective of the algorithm is to group similar data together by their Euclidean distance from *K* centroids, where *K* is a value chosen by the user, while keeping all centroids distant from each other. Each sample will be linked (or allocated/assigned) to the cluster with the nearest centroid.

Interpreting the model. To visualize the mapped features of both models, we took different approaches since they are different models. For the triplet network model, we first inferred all sequences from the validation set and obtained their embeddings. These embeddings were 200-dimensional and were reduced to 2 dimensions using t-SNE in scikit-learn project[62] with default parameters. For the classification model, we extracted the activations of the last hidden layer that maps features from all convolutional layers, before concatenating those features with the extra inputs (sequence metadata) and passing through the last layer, which outputs logits. These hidden features were 3,072-dimensional, which was reduced to 2 dimensions in the same way as for the embeddings. The visualization was done using matplotlib[63], and we colored each point by the corresponding lab.

To analyze the influence of perturbations on the model prediction, we took a specific plasmid from the validation set and randomly generated perturbation in its sequence. As the process is random, we performed ten experiments for each number of mutations, ranging from 0 to 1,000. To make these mutations, we use a random integer function to select the specific position to be changed and a random choice operation to choose one of the five possible bases in the sequence (N, G, C, T, A). For each of the 10,100 runs, we took the position of the correct laboratory in the model's prediction ranking. Figure 4c presents the mean and median of those positions.

To identify which tokens are the most important within a sequence, we decided to use a similar methodology to integrated gradients[27]. Integrated gradient is an interpretability technique for deep neural networks which finds the input features that contribute the most to the model prediction. We started by computing gradients between model predictions with respect to the sequence embedding layer, getting a matrix of gradients in the shape of $1{,}001 \times 200$ (number of tokens × embedding dimension). Each gradient measures the relationship between the embedding weight and the output. We then calculate the absolute value element-wise and sum them up in the second axis, generating a final vector of 1,001 positions containing the summed importance of each token.

$$\text{TokenImportance}_i(x) = \sum_{k=1}^{200} \| \frac{\partial F(x)}{\partial x_{ik}} \| \times \frac{1}{200}, \quad (3)$$

where $F(\cdot)$ is the model's prediction function, $\frac{\partial F}{\partial x_{ik}}$ is the gradient of model F's prediction function relative to each embedding feature x_{ik}, i is the number of tokens and k is the embedding dimension position.

After generating the token importance of each sequence in the validation set, we took the token importance of each laboratory by averaging the token importance of all sequences in that laboratory. The visualization was done using matplotlib, and to better present the figure, the token importance values were normalized between 0 and 1 (NTI). To compare the NTI of a specific lab with the furthest lab from it, we compute the cosine similarity between the analyzed lab embedding and all other lab embeddings by performing a dot product. The lowest value indicates the least similar laboratory.

Data availability

The pAAV-Syn-SomArchon sequence (126941, https://www.addgene.org/126941/) and AAEL010097-Cas9 sequence (100707, https://www.addgene.org/100707/) are available on Addgene. The custom sequence developed by Alley et al. is available in

their GitHub repository (https://github.com/altLabs/attrib/blob/master/sequences/custom_drive.fasta). All other data to reproduce experiments are available in our Code Ocean capsule[64]. Source data for Figs. 1, 2, 4c and 6 and Extended Data Figs. 3 and 4 are provided with this paper. The other figures need large raw data, so their source data are available in the capsule[64].

Code availability

All code necessary to reproduce experiments and generate figures is available in our Code Ocean capsule[64].

Received: 11 November 2021; Accepted: 22 March 2022;
Published online: 28 April 2022

References

1. Alley, E. C. et al. A machine learning toolkit for genetic engineering attribution to facilitate biosecurity. *Nat. Commun.* **11**, 6293 (2020).
2. Nielsen, A. A. K. & Voigt, C. A. Deep learning to predict the lab-of-origin of engineered DNA. *Nat. Commun.* **9**, 3135 (2018).
3. Wang, Q., Kille, B., Liu, T. R., Elworth, R. A. L. & Treangen, T. J. PlasmidHawk improves lab of origin prediction of engineered plasmids using sequence alignment. *Nat. Commun.* **12**, 1167 (2021).
4. Kamens, J. The Addgene repository: an international nonprofit plasmid and data resource. *Nucleic Acids Res.* https://doi.org/10.1093/nar/gku893 (2014).
5. Kulis, B. Metric learning: a survey. *Found. Trends Mach. Learn.* **5**, 287–364 (2013).
6. Koch, G., Zemel, R. & Salakhutdinov, R. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop* Vol. 2 (2015).
7. Hoffer, E. & Ailon, N. Deep metric learning using triplet network. In *Similarity-Based Pattern Recognition. SIMBAD 2015. Lecture Notes in Computer Science* Vol. 9370 (eds Feragen, A. et al.) 84–92 (Springer, 2015).
8. Fink, M. Object classification from a single example utilizing class relevance metrics. In *Proc. 17th International Conference on Neural Information Processing Systems* (eds Saul, L. et al.) 449–456 (MIT Press, 2005).
9. Fei-Fei, L., Fergus, R. & Perona, P. One-shot learning of object categories. *IEEE Trans. Pattern Anal. Mach. Intell.* **28**, 594–611 (2006).
10. Wang, Y., Yao, Q., Kwok, J. T. & Ni, L. M. Generalizing from a few examples. *ACM Comput. Surveys* **53**, 1–34 (2020).
11. Kim, Y. Convolutional neural networks for sentence classification. In *Proc. 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* 1746–1751 (Association for Computational Linguistics, 2014).
12. Caruana, R., Lawrence, S. & Giles, C. L. Overfitting in neural nets: backpropagation, conjugate gradient, and early stopping. In *Proc. 13th International Conference on Neural Information Processing Systems* (eds Leen, T. K. et al.) 381–387 (MIT Press, 2000).
13. Ying, X. An overview of overfitting and its solutions. *J. Phys. Conf. Ser.* **1168**, 022022 (2019).
14. Gage, P. A new algorithm for data compression. *C Users J.* **12**, 23–38 (1994).
15. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S. & Dean, J. Distributed representations of words and phrases and their compositionality. In *Proc. 26th International Conference on Neural Information Processing Systems* (eds Burges, C. J. C. et al.) 3111–3119 (Curran Associates, 2013).
16. Lipton, Z. C. A critical review of recurrent neural networks for sequence learning. Preprint at https://arxiv.org/abs/1506.00019 (2015).
17. Sherstinsky, A. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. Preprint at https://arxiv.org/abs/1808.03314 (2018).
18. Altschul, S. F., Gish, W., Miller, W., Myers, E. W. & Lipman, D. J. Basic local alignment search tool. *J. Mol. Biol.* **215**, 403–410 (1990).
19. Wang, Q., Elworth, R., Liu, T. R. & Treangen, T. J. Faster pan-genome construction for efficient differentiation of naturally occurring and engineered plasmids with plaster. In *19th International Workshop on Algorithms in Bioinformatics* (WABI 2019) 19:1–19:12 (Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019).
20. Lv, Z., Ding, H., Wang, L. & Zou, Q. A convolutional neural network using dinucleotide one-hot encoder for identifying DNA n6-methyladenine sites in the rice genome. *Neurocomputing* **422**, 214–221 (2021).
21. Choong, A. C. H. & Lee, N. K. Evaluation of convolutionary neural networks modeling of {DNA} sequences using ordinal versus one-hot encoding method.In *2017 International Conference on Computer and Drone Applications* (IConDA) 60–65 (IEEE, 2017).
22. Karim, M. R. et al. Deep learning-based clustering approaches for bioinformatics. *Brief Bioinformatics* https://doi.org/10.1093/bib/bbz170 (2020).
23. Xu, D. & Tian, Y. A comprehensive survey of clustering algorithms. *Ann. Data Sci.* https://doi.org/10.1007/s40745-015-0040-1 (2015).
24. Omran, M., Engelbrecht, A. & Salman, A. An overview of clustering methods. *Intell. Data Anal.* **11**, 583–605 (2007).
25. Chakraborty, S. et al. Interpretability of deep learning models: a survey of results. In *2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)* 1–6 (IEEE, 2017).
26. Doshi-Velez, F. & Kim, B. Towards a rigorous science of interpretable machine learning. Preprint at https://arxiv.org/abs/1702.08608 (2017).
27. Sundararajan, M., Taly, A. & Yan, Q. Axiomatic attribution for deep networks. Preprint at https://arxiv.org/abs/1703.01365 (2017).
28. Hsieh, C.-K. et al. Collaborative metric learning. *ACM Digital Library* https://doi.org/10.1145/3038912.3052639 (2017).
29. Yu, J., Gao, M., Rong, W., Song, Y., Xiong, Q. A social recommender based on factorization and distance metric learning. *IEEE Access* https://doi.org/10.1109/access.2017.2762459 (2017).
30. Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. {BERT}: pre-training of deep bidirectional transformers for language understanding. In *Proc. 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* Vol. 1 4171–4186 (Association for Computational Linguistics, 2019).
31. Mikolov, T., Sutskever, I., Chen, K., Corrado, G. & Dean, J. Distributed representations of words and phrases and their compositionality. Preprint at https://arxiv.org/abs/1310.4546 (2013).
32. Pennington, J., Socher, R. & Manning, C. {G}lo{V}e: Global vectors for word representation. In *Proc. 2014 Conference on Empirical Methods in Natural Language Processing* 1532–1543 (Association for Computational Linguistics, 2014).
33. Peters, M. E. et al. Deep contextualized word representations. In *Proc. 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* Vol. 1, 2227–2237 (Association for Computational Linguistics, 2018).
34. Garg, S. Computational methods for chromosome-scale haplotype reconstruction. *Genome Biol.* https://doi.org/10.1186/s13059-021-02328-9 (2021).
35. Vaswani, A. et al. Attention is all you need. In *Proc. 31st International Conference on Neural Information Processing Systems* 6000–6010 (Curran Associates, 2017).
36. Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M. & Monfardini, G. The graph neural network model. *IEEE Trans Neur. Netw. Learn. Syst.* **20**, 61–80 (2008).
37. Lin, T.-Y. et al. Microsoft COCO: common objects in context. Preprint at http://arxiv.org/abs/1405.0312 (2014).
38. Deng, J. et al. Imagenet: a large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition* 248–255 (IEEE, 2009).
39. Andriluka, M., Pishchulin, L., Gehler, P. & Schiele, B. 2D human pose estimation: new benchmark and state of the art analysis. In *IEEE Conference on Computer Vision and Pattern Recognition* 3686–3693 (2014).
40. Krizhevsky, A., Nair, V. & Hinton, G. *The CIFAR-10 Dataset* (Canadian Institute for Advanced Research, 2010); http://www.cs.toronto.edu/~kriz/cifar.html
41. Rajpurkar, P., Zhang, J., Lopyrev, K. & Liang, P. SQuAD: 100,000+ questions for machine comprehension of text. In *Proc. 2016 Conference on Empirical Methods in Natural Language Processing* 2383–2392 (Association for Computational Linguistics, 2016).
42. Wang, A. et al. GLUE: a multi-task benchmark and analysis platform for natural language understanding. Preprint at http://arxiv.org/abs/1804.07461 (2018).
43. Williams, A., Nangia, N. & Bowman, S. A broad-coverage challenge corpus for sentence understanding through inference. In *Proc. 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* Vol. 1, 1112–1122 (Association for Computational Linguistics, 2018).
44. Zellers, R., Bisk, Y., Schwartz, R. & Choi, Y. SWAG: A large-scale adversarial dataset for grounded commonsense inference. In *Proc. 2018 Conference on Empirical Methods in Natural Language Processing* 93–104 (Association for Computational Linguistics, 2018).
45. Hastie, T., Tibshirani, R. & Friedman, J. *The Elements of Statistical Learning: Data Mining, Inference and Prediction* 2nd edn (Springer, 2009).
46. Kaufman, S., Rosset, S. & Perlich, C. Leakage in data mining: formulation, detection, and avoidance. In *Proc. 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* 556–563 (Association for Computing Machinery, 2011).
47. Berger, B., Waterman, M. & Yu, Y. Levenshtein distance, sequence comparison and biological database search. *IEEE Trans. Inform. Theory* **67**, 3287–3294 (2021).
48. Chicco, D. Ten quick tips for machine learning in computational biology. *BioData Mining* https://doi.org/10.1186/s13040-017-0155-3 (2017).
49. Mikolajczyk, A. & Grochowski, M. Data augmentation for improving deep learning in image classification problem. In *2018 International Interdisciplinary PhD Workshop* (IEEE, 2018).

50. Moshkov, N., Mathe, B., Kertesz-Farkas, A., Hollandi, R. & Horvath, P. Test-time augmentation for deep learning-based cell segmentation on microscopy images. *Sci. Rep.* https://doi.org/10.1038/s41598-020-61808-3 (2020).
51. Zou, Q., Xing, P., Wei, L. & Liu, B. Gene2vec: gene subsequence embedding for prediction of mammalian n6-methyladenosine sites from mRNA. *RNA* **25**, 205–218 (2018).
52. Klambauer, G., Unterthiner, T., Mayr, A., & Hochreiter, S. Self-normalizing neural networks. Preprint at https://arxiv.org/abs/1706.02515 (2017).
53. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **15**, 1929–1958 (2014).
54. Paszke, A. et al. PyTorch: an imperative style, high-performance deep learning library. In *Proc. 33rd International Conference on Neural Information Processing Systems* (eds Wallach, H. et al.) 8024–8035 (Curran Associates, 2019).
55. Kingma, D. P. & Ba, J. Adam: a method for stochastic optimization. Preprint at https://arxiv.org/abs/1412.6980 (2014).
56. Smith, L. N. & Topin, N. Super-convergence: very fast training of residual networks using large learning rates. Preprint at https://arxiv.org/abs/1708.07120 (2017).
57. Hermans, A., Beyer, L. & Leibe, B. Defense of the triplet loss for Person re-identification. Preprint at https://arxiv.org/abs/1703.07737 (2017).
58. Musgrave, K. et al. PyTorch metric learning. Preprint at https://arxiv.org/abs/2008.09164 (2020).
59. Wang, X., Zhang, H., Huang, W. & Scott, M. R. Cross-batch memory for embedding learning. Preprint at https://arxiv.org/abs/1912.06798 (2019).
60. van der Maaten, L. & Hinton, G. Visualizing data using t-SNE. *J. Mach. Learn. Res.* **9**, 2579–2605 (2008).
61. Arthur, D. & Vassilvitskii, S. *K*-means++: the advantages of careful seeding. In *Proc. Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms* 1027–1035 (SIAM, 2007).
62. Pedregosa, F. et al. Scikit-learn: machine learning in Python. *J. Mach. Learn: Res.* **12**, 2825–2830 (2011).
63. Hunter, J. D. Matplotlib: a 2D graphics environment. *Comput. Sci. Eng.* **9**, 90–95 (2007).
64. Soares, I., Camargo, F., Marques, A. & Crook, O. Improving lab-of-origin prediction of genetically engineered plasmids via deep metric learning. *Code Ocean* https://doi.org/10.24433/CO.9853572.V1 (2022).
65. Hinton, G. & Roweis, S. Stochastic neighbor embedding. In *Proc. 15th International Conference on Neural Information Processing Systems* 857–864 (MIT Press, 2002).

Acknowledgements

I.M.S., F.H.F.C. and A.M. acknowledge Amalgam and XNV for providing the necessary infrastructure and financial support to develop the study and design the experiments. O.M.C. acknowledges funding from a Todd-Bird Junior Research Fellowship from New College, Oxford, as well as Open Philanthropy. I.M.S. acknowledges K. A. Assis for helping with graphical design.

Author contributions

I.M.S. and F.H.F.C. conceived the study and designed the experiments. F.H.F.C. developed the triplet model training algorithm. A.M. developed the circular shift augmentation. O.M.C. split the data. I.M.S. developed the interpretation method. F.H.F.C. did the few-shot experiments. All authors wrote the manuscript, and reviewed and approved the final manuscript. I.M.S. managed the project.

Competing interests

The authors declare no competing interests.

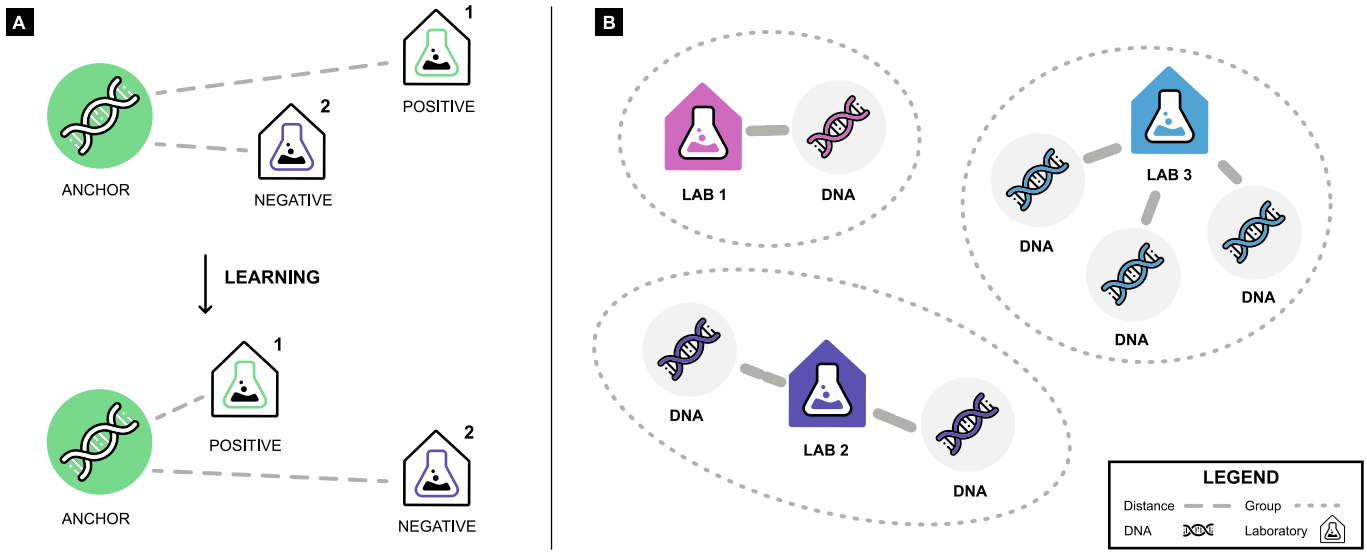
Additional information

Extended data is available for this paper at https://doi.org/10.1038/s43588-022-00234-z.

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s43588-022-00234-z.

Correspondence and requests for materials should be addressed to Igor M. Soares, Fernando H. F. Camargo, Adriano Marques or Oliver M. Crook.

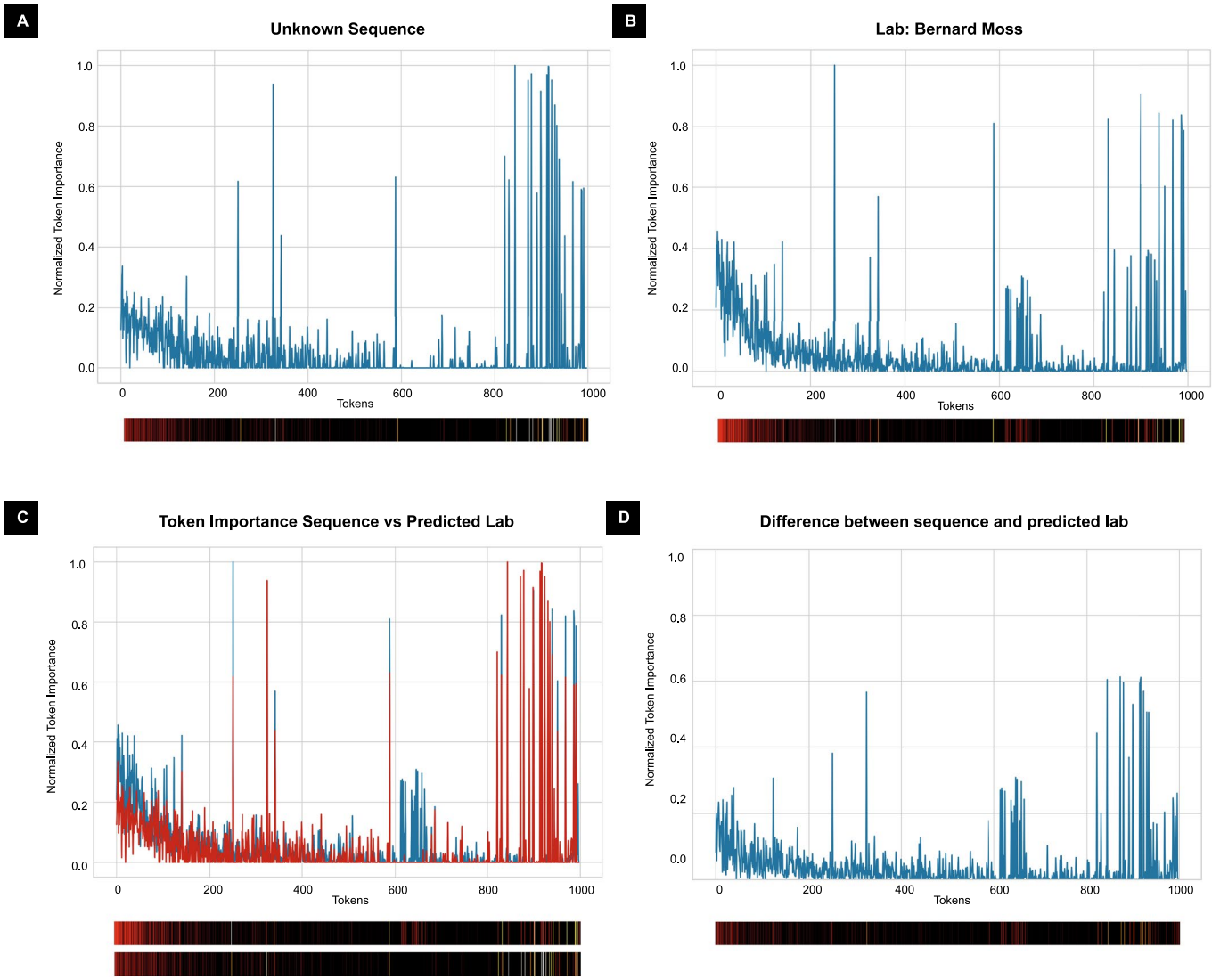
Peer review information *Nature Computational Science* thanks Shilpa Garg and the other, anonymous, reviewer(s) for their contribution to the peer review of this work. Peer reviewer reports are available. Primary Handling Editor: Kaitlin McCardle, in collaboration with the *Nature Computational Science* team.



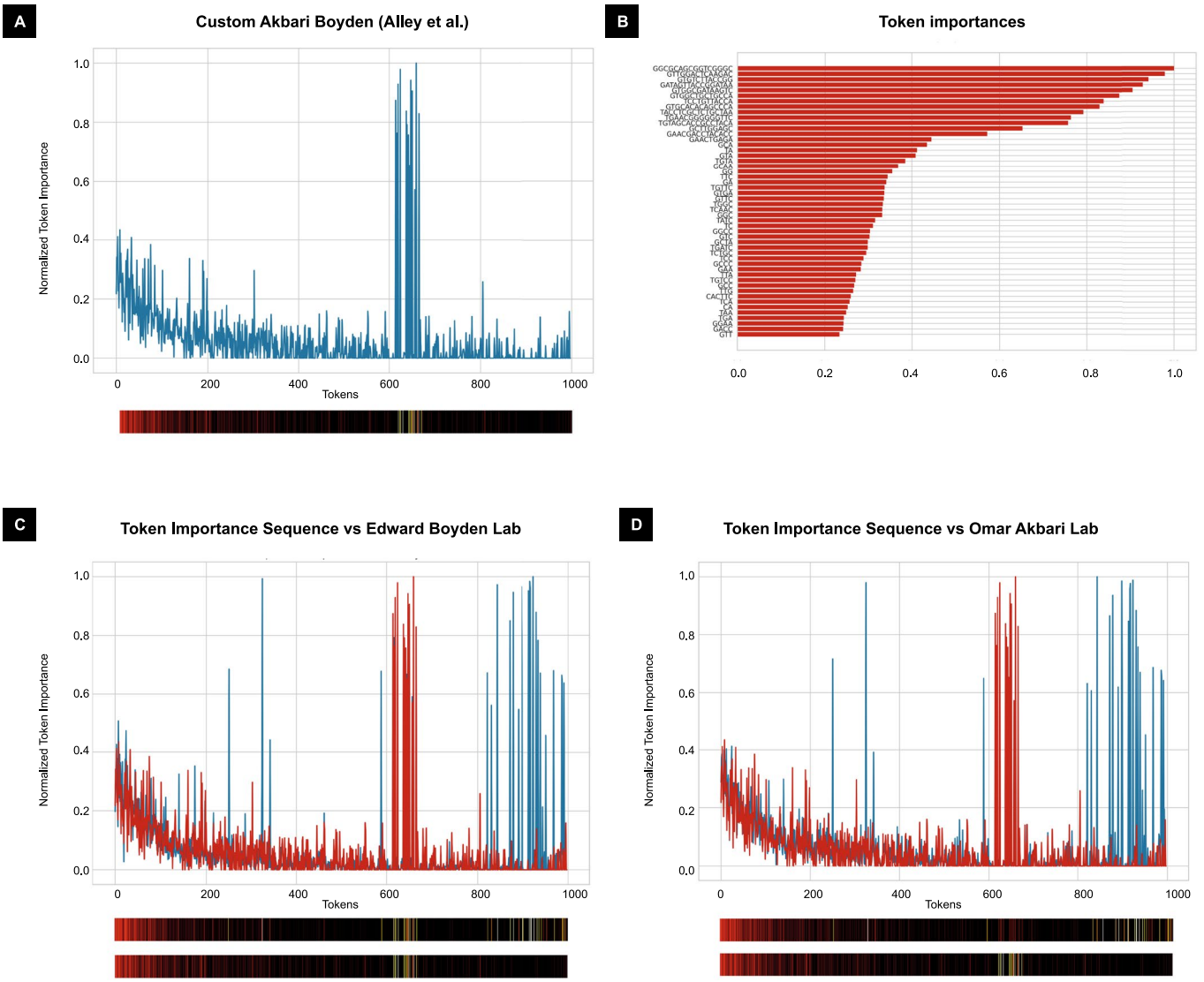
Extended Data Fig. 1 | Triplet method illustration. The triplet is composed of an anchor (DNA), a positive (the lab-of-origin), and a negative example (another lab). a In the beginning, the anchor might be closer to the negative than it is to the positive. During training, we pull the anchor and positive towards each other while pushing the negative away. b In the end, labs and their DNA sequences will be nearer to each other, forming groups. Each group is represented by a different color indicating that they are in separate spaces. We can also expect both labs and DNA sequences to be closer to other similar ones.

Extended Data Fig. 2 | See next page for caption.

Extended Data Fig. 2 | Applying Convolutional Neural Networks with Metric Learning on plasmids leads to state-of-the-art. Images a to f illustrate our method for both models, where they differ in training output and methodology (c). In g, we compare top-10 prediction accuracy on the test set of both approaches to Nielsen and Voigt[2], Alley et al.[1], BLAST baseline[18] and PlasmidHawk[3]



Extended Data Fig. 3 | Plasmid features importance of unknown sequences. a NTI for an unknown sequence. This may help to investigate specific patterns. b The NTI for Bernard Moss's lab which was assigned the sequence author by our model. c Comparison between tokens highlighted for the sequence and important tokens from the predicted lab. The red line represents the sequence. The first bar under the graph represents the laboratory, while the second represents the sequence. d Plotting the difference between the NTI of the sequence and the NTI of the predicted laboratory, we can see that few tokens stand out in this sequence beyond the usual presented by the laboratory, and the bar is almost all black.



Extended Data Fig. 4 | NTI for custom sequence designed by Alley et al.[1] and its closest two labs. NTI for the custom sequence. Tokens in the 600-700 range seem to be the most substantial for this streak. b Top 30 most important tokens. The BPE generates the tokens by grouping characters, and those tokens require further investigation to identify biological features within them. c Comparison between tokens highlighted for the sequence and important tokens from the predicted Omar Akbari lab. They share a similar level of importance for the first 700 tokens. The first bar under the graph represents the Akbari laboratory, while the second represents the custom sequence. d Comparison between tokens highlighted for the sequence and important tokens from the top2 predicted Edward Boyden lab. They are similar only in the first tokens in the BPE dictionary and neither Boyden Lab nor Akbari Lab have the same importance as the last tokens as shown in the NTI of the sequence. The first bar under the graph represents the Boiden laboratory, while the second represents the custom sequence.