scientific reports

OPEN

Scaling up DNA digital data storage by efficiently predicting DNA hybridisation using deep learning

David Buterez

Deoxyribonucleic acid (DNA) has shown great promise in enabling computational applications, most notably in the fields of DNA digital data storage and DNA computing. Information is encoded as DNA strands, which will naturally bind in solution, thus enabling search and pattern-matching capabilities. Being able to control and predict the process of DNA hybridisation is crucial for the ambitious future of *Hybrid Molecular-Electronic Computing*. Current tools are, however, limited in terms of throughput and applicability to large-scale problems. We present the first comprehensive study of machine learning methods applied to the task of predicting DNA hybridisation. For this purpose, we introduce an in silico-generated hybridisation dataset of over 2.5 million data points, enabling the use of deep learning. Depending on hardware, we achieve a reduction in inference time ranging from one to over two orders of magnitude compared to the state-of-the-art, while retaining high fidelity. We then discuss the integration of our methods in modern, scalable workflows.

The use of DNA to facilitate computation is an active area of research, dating back to 1994 when Leonard Adleman solved a seven-node instance of the Hamiltonian path problem, using the toolbox of (DNA) molecular biology[1]. This pioneering work opened the gate to many interesting questions: can molecular machines be used to solve intractable problems? Is DNA suitable for long-term storage of digital information? More recently, are such methods scalable in the era of Big Data?

The focus has gradually changed from solving difficult computational problems to exploiting desirable properties of DNA, leading to the development of *DNA digital data storage*. It is now generally accepted that the amount of digital data is doubling at least every two years. Predictions from Seagate estimate that this quantity, called the *Global Datasphere*, will grow from 33 ZB (zettabytes) in 2018 to 175 ZB by 2025[2]. Furthermore, the dominant storage mediums are traditional, with 59% of the storage capacity expected to come from hard disk drives and 26% from flash technologies. Synthetic DNA has been argued to be an attractive storage medium, for at least three prominent reasons[3]:

1. *Density* - The theoretical maximum information density of DNA is 10^{18} B/mm^3. Comparatively, this is a 7 orders of magnitude increase over tape storage.
2. *Durability* - Depending on storage conditions, DNA can be preserved for at least a few hundred years. Fossil studies reveal that DNA has a half-life of 521 years[4]. In appropriate conditions, researchers estimate that digital information can be recovered from DNA stored at the Global Seed Vault (at − 18 °C) after over 1 million years[5].
3. *Future-proofing* - Next-generation sequencing and technologies such as Oxford Nanopore[6] have made reading and writing DNA more accessible than ever. Furthermore, since DNA is the fundamental building block of life, it will be relevant for as long as humans exist.

In silicon-based computers, information is loaded and stored from unique locations denoted by a numerical address or index (random access). Implementing this property in DNA storage applications requires a short, arbitrary sequence of DNA acting as the index, enabling selection and amplification by polymerase chain reaction (PCR)[7,8]. Using this design, the index DNA sequence acts as both a unique identifier and a PCR primer, thus it must fulfil standard primer length requirements, often between 18 and 26 nucleotides[9], and annealing and melting temperature requirements, often between 30 and 65 °C[10–12]. As the scale of the primer library increases, designing sequences that strongly interact with their desired targets and weakly (preferably never) interact with non-intended sequences becomes progressively more difficult. Such sequence libraries are called **orthogonal**.

[1]Present address: Department of Computer Science and Technology, University of Cambridge, Cambridge, UK. [2]Department of Computing, Imperial College London, London, UK. email: db804@cam.ac.uk

It is also possible to exploit non-precise hybridisation to enable fuzzy or similarity search, by encoding closely-related entities (by some distance metric) in DNA, leading to **similar** datasets.

Successful and precise DNA hybridisation is required to implement computational features directly at the level of DNA. The OligoArchive project[7] introduced a technique capable of performing SQL operations such as selection, projection and join directly on the DNA molecules, by means of hybridisation and finding appropriate encodings for primary keys and attributes. Stewart et al.[13] used an approximation for thermodynamic analysis that is differentiable (necessary for backpropagation) in the form of a modified sigmoid function, with the goal of creating a content-addressable DNA database. Unfortunately, the estimation is far from perfect and the authors remarked that an important future direction is a more accurate approximation for thermodynamic yield.

Zhang et al.[14] predicted hybridisation kinetics from 36-nucleotide sequence pairs derived from human genes (non-synthetic), on a much more limited scale (100 pairs) using non-neural methods. We first introduced the concept of using CNNs to predict DNA hybridisation as a poster in June 2019, at the *25th International Conference on DNA Computing and Molecular Programming* (DNA25), reporting significant gains in inference time. Close to our research direction, in a recent preprint Bee et al.[15] present a similarity search DNA database that employs a hybridisation predictor, a CNN trained on one-hot encodings of DNA. The authors state that hybridisation yields are either close to 0 or close to 1, suggesting that they do not consider intermediate values during training.

Unfortunately, NUPACK, the state-of-the-art in the analysis and design of nucleic acid systems[16], cannot be used for large-scale applications. Firstly, while NUPACK employs efficient dynamic programming algorithms, these still require $O(N^4)$ time and $O(N^2)$ memory, rendering multi-million applications intractable. Secondly, NUPACK 3 is only available as a collection of standalone executables, massively slowing down the performance due to repeated I/O operations. Thirdly, and perhaps most importantly, the computation of NUPACK is not differentiable, thus it cannot be used in neural networks. We acknowledge that the very recently introduced NUPACK 4[17] has completely removed the second drawback. However, while NUPACK 4 provides faster, vectorised operations, the first point is only partially mitigated, as the massive speed improvements only apply to large complexes of long sequences. The third point still applies in full.

We recognise the need for scalable and accurate prediction of DNA hybridisation, necessarily as a differentiable operation that can be seamlessly plugged into neural network models. Hybridisation of two single-stranded sequences is quantified by the equilibrium concentration of the DNA duplex, here referred to as *yield*. We propose a modern, scalable deep learning approach for predicting DNA hybridisation (schematically illustrated in Fig. 1), designed and evaluated on an in silico-generated hybridisation dataset of over 2.5 million single-stranded DNA sequence pairs with ground truth yields computed at 37.0 °C, 42.0 °C, 47.0 °C, 52.0 °C, 57.0 °C and 62.0 °C (Celsius), capturing a wide range and largely mirroring the temperatures used in[14,18], making the following contributions:

1. *Integration with neural networks* - We investigate Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNN) and Transformers. Naturally, we formulate the problem such that backpropagation can be applied.
2. *Speed and accuracy* - A decrease in inference time of one to over two orders of magnitude compared to non-neural methods, while we empirically show that our best models can reach over 0.965 in F_1 score per class after binarising the predicted yields.

Results

Designing a diverse hybridisation dataset.

To achieve a high-quality, comprehensive hybridisation dataset, the following two criteria must be satisfied: **size**—large enough to allow a broad range of machine learning methods (including deep learning) and **variety**—capture as many DNA interactions as possible, not only those corresponding to maximum or minimum yield. Unfortunately, we are not aware of any hybridisation dataset of biological origin that satisfies these conditions. Therefore, we introduce a synthetic dataset, where the DNA sequence pairs are generated from scratch and the yield computations are carried out by NUPACK.

Following the procedures presented in the "Methods" section, a final dataset of 2,556,976 sequence pairs or just over 2.5 million data points is assembled, with sequence lengths varying in the range 18–26. By examining the properties of extreme and intermediary samples, a machine learning model should ideally be able to deduce the Watson-Crick base-pairing rules and how these affect the yield. The yield distribution is summarised visually in Fig. 2. For evaluation, we used stratified splits of 2,045,579 for training, 255,696 for validation and 255,701 for testing.

To establish a baseline and study the difficulty of the task, we extend the dataset with extracted features for each sequence, enabling classical machine learning techniques to be applied. Alignment scores reveal the relatedness of two biological sequences and are hypothesised to be good predictors of DNA annealing even for arbitrary, artificially-generated sequences that are much shorter than typical naturally-occurring DNA fragments. As such, the first extracted feature is given by alignment scores, computed according to the "Methods" section. Furthermore, calculations such as the minimum free energy (MFE) of secondary structures indicate potentially undesirable behaviour. Motivating examples are provided in Fig. 3.

Overall, we extend the dataset with the alignment score, secondary structure MFE, concentration of the single-stranded and double-stranded forms, and the GC content percentage of each sequence (ablation study in Supplementary Information 12).

Supervised learning using extracted features.

The choice of machine learning algorithms is motivated by previous work (LDA in[18]), scalability to large problems (millions of samples) and widespread adoption and recognition. The four algorithms we evaluate are Linear Discriminant Analysis (LDA), Quadratic Discri-

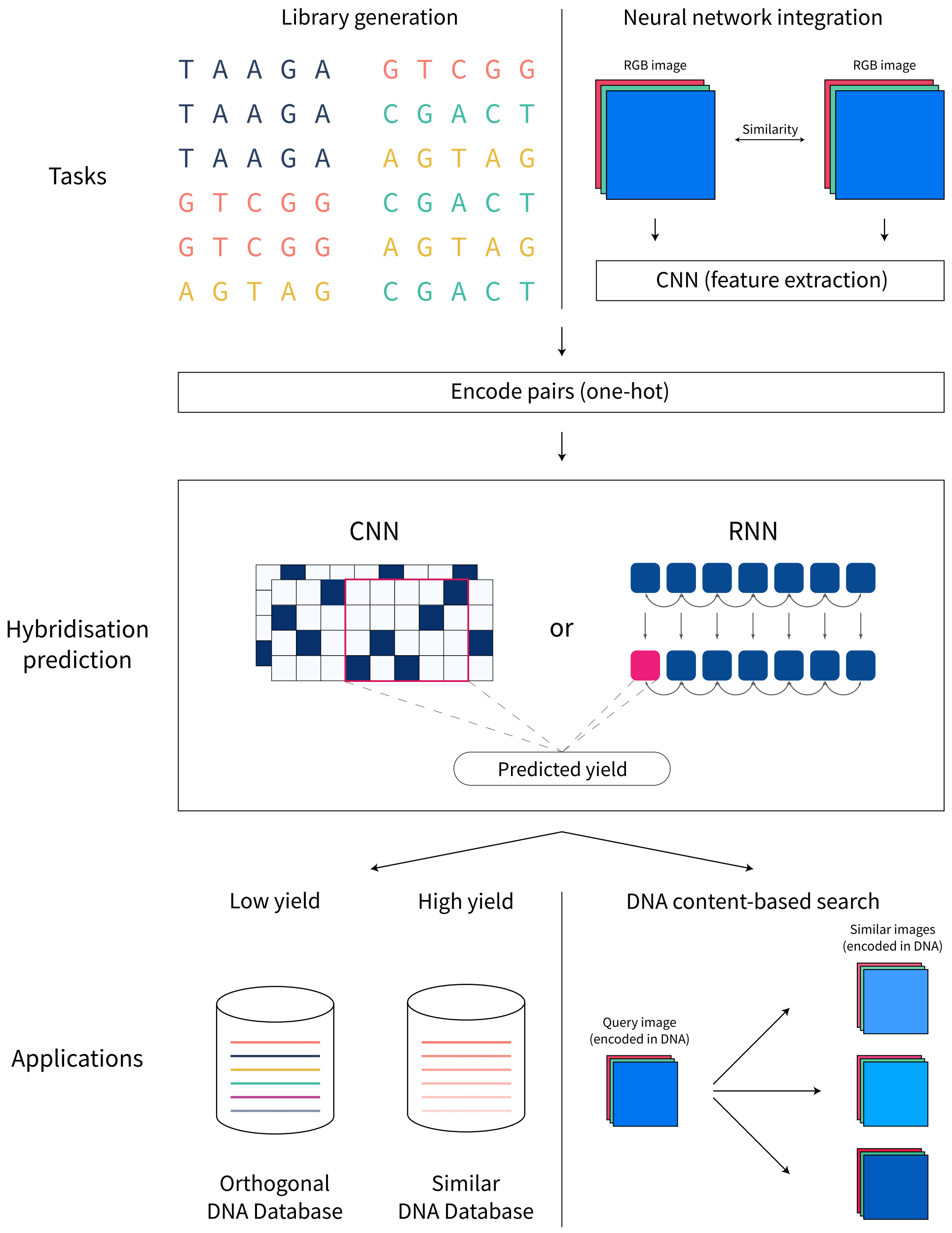
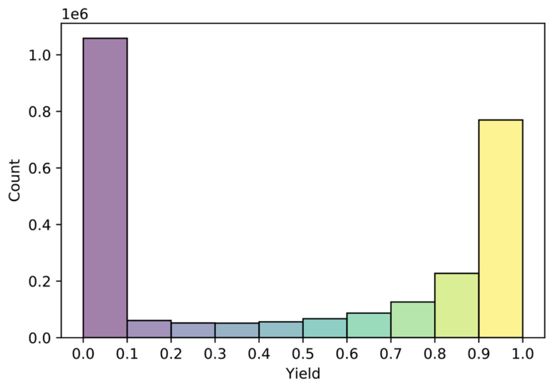


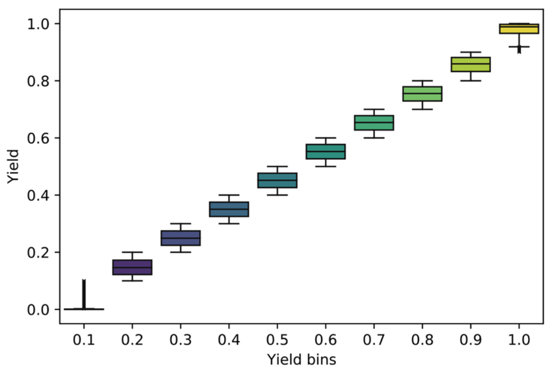
Figure 1. A high-level overview on how to integrate hybridisation prediction into DNA storage workflows. A trained machine learning model can be used as a standalone tool to assemble orthogonal or similar libraries of DNA sequences (left half of the figure). Alternatively, the neural network can be seamlessly integrated into a larger machine learning model as a subcomponent. The presented example is content-based search in a DNA database, where document features are extracted by a neural network (CNN for images, but text, video or audio inputs are conceivable) in a pairwise manner, another neural component generates appropriate encodings (usually one-hot) and the hybridisation predictor outputs the expected yield of the pair. Such a model is trained to associate similar documents to similar single stranded DNA sequences that form stable duplexes with the query sequence (right half of the figure).

minant Analysis (QDA), Random Forests (RF) and Neural Networks (NN). Our criteria prohibit the use of some well-known algorithms such as support-vector machines (SVM) due to limited scalability. For all chosen methods we perform hyperparameter optimisation, with the exception of QDA (very small number of tunable hyperparameters). For additional details see Supplementary Information 2.

The results of Fig. 4c,e confirm that random forests and neural networks are the top performers on this classification task. Furthermore, the more granular performance report from Fig. 4c indicates that all chosen methods perform well on the classification task. A general trend, consistent across all algorithms, can be identified. Precision for the **Low** class is high, indicating that when a low prediction is made, it is very likely correct. However, the recall for this class is relatively low, signalling that **Low** samples are often classified incorrectly as



a Histogram of the hybridisation dataset.



b Box plot of the 10 yield bins (x axis labelled by the right endpoint; means not shown).

Figure 2. Visual summary of the yield distribution at 57 °C (the temperature used throughout the paper). For a discussion on the behaviour of the yield at different temperatures, see Supplementary Information 14. The yield is binned in 10 groups, each spanning a 0.1 interval. Brighter colours correspond to higher yield and are shared for the two subfigures. (**a**) Low and high values are the most numerous, which is expected considering our generative procedure. (**b**) The highest density is achieved at the extremes of 0, respectively 1. Given how sensitive the molecules are even to a 1-base change, we count the entire intermediate range of yields [0.1, 0.9) as one entity when considering how balanced the dataset is. In this regard, 1,058,364 pairs achieve low yields (< 0.1), 769,750 achieve very high yields (≥ 0.9) and 728,862 are in-between. It is important that samples with extremely low or high yields are well represented. In particular, there are virtually endless combinations of base pairs resulting in minimum yield.

High. Almost all members of **High** are identified correctly (high recall). This trend is at its extreme for LDA, with the others (QDA, RF, NN) becoming more and more balanced across the two classes.

We acknowledge two important limitations of this approach. Firstly, **generality**—ideally, yields should be approximated directly, as a regression task, since different applications might require different cut-offs for the classification labels. Unfortunately, the methods presented so far are unable to approximate yields directly on our challenging dataset. Secondly, **performance and scalability**—while the actual inference times of the algorithms in Fig. 4c are low, predictions require 9 pre-computations. Preferably, ML methods should be applied directly to sequences.

To address these limitations we introduce the use of deep learning methods to predict DNA hybridisation directly from pairs of sequences.

Deep learning strategies for DNA hybridisation. We employ two different deep learning paradigms: image-based, or more generally, models that operate on 2-dimensional grids with explicitly defined coordinates, and sequence-based. For the first category, we choose Convolutional Neural Networks (CNNs) as a representative, while for sequence models we investigate both Recurrent Neural Networks (RNNs) and Transformers.

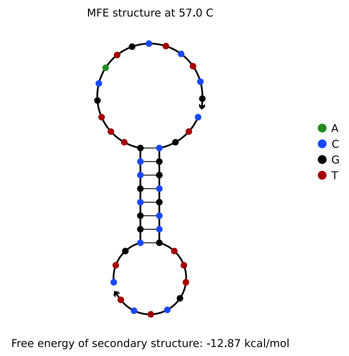
One of the first challenges in designing CNN models is to find a 2-dimensional representation of DNA sequences that enables convolutions. Our inputs are pairs of single-stranded DNA represented using the alphabet $\{A, C, G, T\}$. The proposed 2D representation is as $4 \times N \times 2$ images (*height* $\times$ *width* $\times$ *channels*), where N is the maximum sequence length.

A simplified, schematic overview of the CNN architecture is provided in Fig. 4a. Our CNN implementation is described in the "Methods" section and the precise architectural details and hyperparameters are discussed in Supplementary Table 3 and Supplementary Information 5.

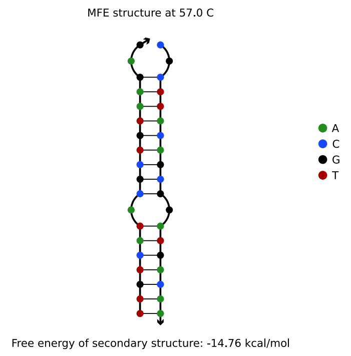
By studying sequence-based algorithms, the existing knowledge from language processing can be transferred to DNA storage and computing. The string representation of DNA sequences is naturally fit for RNNs. Here, we propose a character-level RNN based on bi-directional Long Short-Term Memory (LSTM)[19] layers.

Compared to CNNs, which capture the spatial information within a receptive field and learn pixel-level filters, RNNs process the input data (characters) sequentially and maintain an internal memory state. Here, we hypothesise that the LSTM can effectively model the interaction between hybridising DNA strands, which requires capturing long-distance nucleobase relationships. Our choice of encoding DNA pairs also naturally suggests using a bi-directional implementation (see Fig. 4b). A simplified overview of the RNN architecture is provided in Fig. 4b, and the architectural details and hyperparameters are provided in Supplementary Information 5.

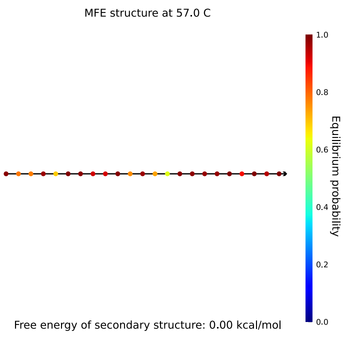
Finally, we turn our attention to the Transformer[20], an architecture now prevalent in deep learning, with its origins in Natural Language Processing. We refer the readers to[21] for an authoritative review, and to Supplementary Information 6 for a short summary. For this study, we adapt an established Transformer implementation, RoBERTa[22]. Our use falls within the *encoder-only* category: in language processing jargon, we formulate our task as a *sentence-pair* regression problem, where the inputs are two "sentences" (DNA sequences) with the goal of outputting a numerical value (hybridisation yield). At a high level, the workflow is very similar to Fig. 4b, in the sense that the inputs are tokenised and propagated through multiple Transformer blocks. Distinctly from other



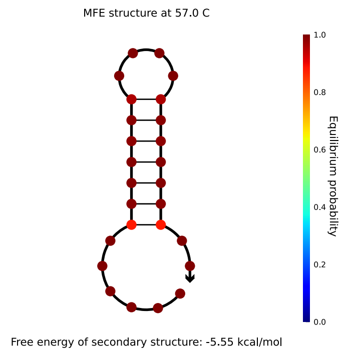
a Pair with alignment score 31.



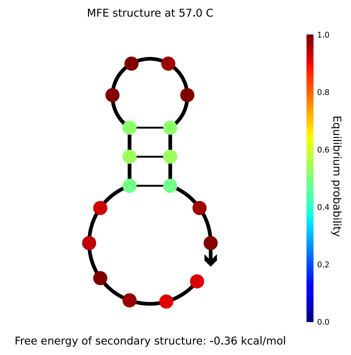
b Pair with alignment score 77.



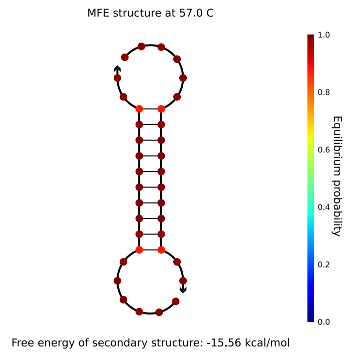
c A sequence with no remarkable secondary structure.



d A sequence that exhibits self-complementarity.



e Equilibrium concentration of the ssDNA is only 0.17 µmol.

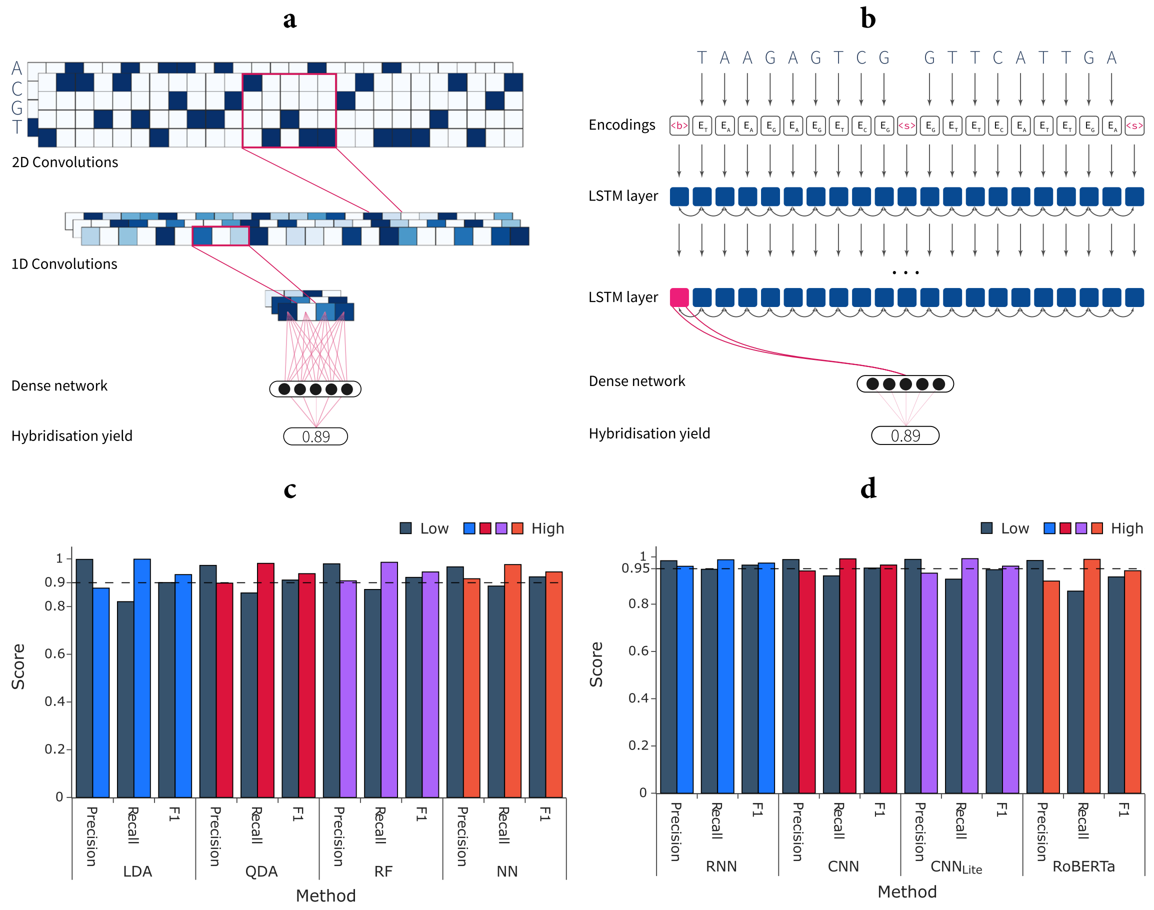


f Equilibrium concentration of the dsDNA is 0.92 µmol.

Figure 3. Alignment and thermodynamic properties for various DNA sequences as reported by NUPACK. (**a**,**b**) Duplex structure predicted by NUPACK (see parasail alignments in Supplementary Information 1). The parasail alignment is not in full agreement with the predicted binding. (**c**,**d**) For single-stranded DNA, a low MFE indicates a high probability that the molecule develops self-complementarity or knots. (**c**) A sequence that is expected to be stable (`AGTACAAGTAGGACAGGAAGATA`). (**d**) A sequence that is expected to be more problematic in hybridisation reactions (`TTTCGCACGGACGAGGACGTCCGTTA`). (**e**,**f**) A sequence can be similar enough to its reverse complement that it is more probable to find it in duplex formations with different instances of itself rather than in the normal single-stranded state. Illustrated is the sequence `CCATGGAGGCGCGCCTTT` in a complex of size 2, each strand initially present in solution at concentration 1 µmol. The duplex formation of this sequence (**f**) is more than 5 times as abundant as the single-stranded conformation (**e**).

models, RoBERTa requires two separate training phases: pre-training and fine-tuning. Our implementation is described in the "Methods" section.

Deep learning accurately predicts hybridisation. We evaluate three models termed simply CNN, RNN and RoBERTa and additionally a fourth convolutional model with a reduced number of convolutions and filter sizes: CNN_{Lite}. Performance metrics are summarised in Fig. 4d,e. With the exception of RoBERTa, we



e

Metric	LDA	QDA	RF	NN	RNN	CNN	CNN_{Lite}	RoBERTa
MCC	0.851	0.859	0.884	0.873	**0.940**	0.920	0.909	0.863
AUROC	0.912	0.922	0.938	0.930	**0.968**	0.956	0.949	0.922
MSE	N/A	N/A	N/A	N/A	**76.660**	109.550	133.628	305.050

Figure 4. (**a**,**b**) Simplified representations of the deep learning architectures. (**a**) Convolutional Neural Network overview. Each of the columns in the 2-channel grid (image) corresponds to a one-hot encoding of the four nucleobases for that particular strand position, while each channel represents an entire strand. 2D convolutions on the 2-channel one-hot encoded DNA strands are followed by 1D convolutions (only 3 channels shown) and fully-connected layers. (**b**) Recurrent Neural Network overview. LSTM layers are widely used and recognised for their performance in language modelling tasks[23], as well as other sequence-based tasks[24]. For readability, we represent bi-directional interactions with two-headed arrows between the sequence elements. (**c**–**e**) Classification and regression results for the evaluated machine learning models. The choice of metrics is explained in the "Methods" section. (**c**,**d**) Graphical summary of precision, recall and F_1 score for the two classes of machine learning models. The **Low** class is represented by dark grey and the **High** class corresponds to the four bright colours (one colour for each method eases readability). (**c**) The four baseline ML algorithms. Exact numerical values are provided in Supplementary Table 1. (**d**) Classification metrics for the four deep learning models, after yield binarisation (with a threshold of 0.2). The numerical values are provided in Supplementary Table 2. (**e**) The AUROC and MCC summarise the classification performance of all evaluated machine learning techniques; additionally, the MSE (Mean Squared Error) is reported for the four deep learning models.

Model	37.0 °C	42.0 °C	47.0 °C	52.0 °C	62.0 °C
RNN	0.862	0.872	0.888	0.916	0.833
CNN	0.884	0.893	0.908	0.929	0.811
$\mathrm{CNN}_{\mathrm{Lite}}$	0.893	0.901	0.914	0.929	0.801
RoBERTa	0.925	0.930	0.930	0.914	0.762

Table 1. MCC of the four deep learning models on the selected temperatures. The results suggest that the models perform reasonably well even on temperatures they were not trained on.

report a marked increase in classification performance over the baseline methods, which completely failed in approximating yields.

We interpret the results at the level of false positives (FP) and false negatives (FN) as they enable a more fine-grained discussion. Examining the performance of the two best models (CNN and RNN), we find that the CNN model is slightly better at avoiding false negatives, at 1,174 FNs compared to 1,174 for the RNN. However, only 413 FNs are shared, meaning that a large portion of the FNs which are wrongly labelled by one model would be correctly classified by the other. It is important to note that the accurate labelling of the positives (**High** class) is an easier task than for negatives on our dataset, as indicated by the baseline methods.

More interestingly, the RNN offers a pronounced decrease in false positives, at 5,871 compared to 8,981 for the CNN. In this case, 4,732 FPs are shared, suggesting that, in general, the RNN makes the same mistakes as the CNN, but less often. False positives are the area where both the baseline and deep learning models struggled the most. FNs and FPs for the RNN indicate a trade-off, where the model loses some power in predicting true positives but significantly reduces false positives.

The $\mathrm{CNN}_{\mathrm{Lite}}$ performs worse, on the whole, compared to the deeper CNN and the RNN. Generally, the erroneously classified samples are close to being subsets of the corresponding FNs and FPs of the CNN. While $\mathrm{CNN}_{\mathrm{Lite}}$ is not as strong a performer as the first two models, its benefits are in inference times, which will be presented shortly.

The Transformer model, RoBERTa, has significant difficulties in discerning positives and exhibits the highest number of false positives, at 16,212. About half of these are overlapping with those reported by the CNN model, and about a third are shared with the RNN. Thus, the dataset contains difficult examples that are not correctly classified by any model and this is exacerbated for RoBERTa. The number of false negatives is low, at 1,174, a trend observed by the other models as well. These mediocre results could be partially explained by Transformers being notoriously data-hungry. It is possible that not even our training dataset of over 2.5 million sequence pairs is enough to fully exploit the architecture.

Pre-trained models allow estimation on different temperatures. Studying the performance of the deep learning approach on different temperatures can prove or refute the generalisability of the models. Furthermore, for time and cost-saving purposes, it is essential to investigate if existing trained models can be repurposed for different temperature ranges that were not seen during training.

For brevity, we report only the Matthews Correlation Coefficient in Table 1. Additional metrics are provided in Supplementary Table 6 and Supplementary Table 7.

We evaluate the models of Fig. 4d on the five additional temperatures. Table 1 and Supplementary Table 6 suggest that the models perform reasonably well even on temperatures they were not trained on. The MSE continues to increase as the temperature is lowered and the distance from the training setting increases. For RNN, CNN and $\mathrm{CNN}_{\mathrm{Lite}}$, this also translates to a progressive alteration of the classification metrics. Overall, the RNN model is less generalisable, i.e., performs worse on different temperatures compared to the CNNs. The most dramatic drop in performance is for 62.0°, as high-scoring pairs tend to achieve much lower yields at this temperature.

Perhaps surprisingly, RoBERTa performs better in terms of classification metrics on the lower temperatures. In fact, this agrees with our previous observation that RoBERTa exhibits many false positives: as the temperature is lowered, low-scoring pairs are more likely to truly have high yields.

Deep learning reduces prediction time by orders of magnitude. Short inference times are crucial in scaling DNA hybridisation computations for the zettabyte future. To study this aspect, we perform a comprehensive empirical evaluation of the time required for the forward pass, i.e., prediction time. The choice of experimental platforms is described in detail in Supplementary Information 8, Supplementary Information 9 and the approach taken to measure time in the "Methods" section and Supplementary Information 10. Results are reported in Table 2.

Depending on the hardware used, we observe a speedup of one to over two orders of magnitude over multi-threaded NUPACK. When trained on a GPU, the previously identified best models (RNN and CNN) are around $\times 23$ and respectively $\times 16$ faster in making predictions, while TPUs almost trivialise inference times. The RNN's speed is comparable to the CNN's despite having less than one tenth of the number of trainable parameters. This is expected, as the sequential nature of RNNs prohibits the massively parallel optimisations possible for convolutions. For this reason, we introduced $\mathrm{CNN}_{\mathrm{Lite}}$, whose architecture is described in Supplementary Table 4. $\mathrm{CNN}_{\mathrm{Lite}}$ still offers competitive performance (Fig. 4e), while being almost twice as fast as the RNN. Due to the complex architecture and large number of parameters, RoBERTa (GPU) is slightly slower than NUPACK, while

Model	# Params.	Batch	Hardware	Time (s)	Speedup
NUPACK 3	N/A	N/A	64-core VM	372.59	×1.00
RoBERTa	6.1M	1024	RTX 3090	388.44 ± 0.32	×0.96
RNN	249K	8192	RTX 3090	15.87 ± 0.10	×23.47
		4096	TPUv2	03.60 ± 0.11	×103.50
CNN	2.8M	512	RTX 3090	23.84 ± 0.08	×15.63
		4096	**TPUv2**	**01.23 ± 0.17**	**× 301.74**
CNN_{Lite}	470K	512	RTX 3090	09.01 ± 0.00	×41.34
		4096	TPUv2	01.28 ± 0.15	×290.21

Table 2. The execution of all deep learning methods is timed on the test dataset of 255,701 samples. The text in bold corresponds to the best model according to the time/speedup. The average execution time and the standard deviation are reported in seconds. Each deep learning method is run 10 times, after an initial warm-up run. The time elapsed to load the dataset into memory is not taken into account and the batch size was chosen to maximise inference time. All deep learning models use consumer hardware or openly-available hardware (the TPU platform is completely free to use).

also underperforming in prediction accuracy. Given these two observations, it is difficult to currently recommend Transformers.

The inference time increases proportionally with the dataset size (Supplementary Information 13, Supplementary Figure 1).

Integrating fast approximation into large-scale workflows. We concentrate on the task of designing a set of orthogonal sequences. A brute-force approach to this problem quickly becomes unfeasible: checking every pair of a set is a quadratic problem in the length of the set: for a set of 100,000 initial sequences, over 5 billion pairs need to be checked. No matter how efficient the yield computation, we cannot tackle this problem using just brute-force. We can, however, devise a method that reduces the use of heuristics as much as possible and utilises our efficient yield approximation.

Concretely, we determine an appropriate pairwise Longest Common Substring (not subsequence, i.e., the characters have to be consecutive)—the LCS. This is motivated by the fact that a sequence has to have at least *some* similarity to its reverse complement for them to anneal. In particular, consecutive regions are much more stable than individual, separated nucleobases that match. We demonstrate our workflow using a randomly generated 20-nucleobases sequence set of size 100,000. We use MMseqs2[25] to cluster sequences that achieve a minimum target LCS (in our case, 5). The full details are provided in the "Methods" section. After processing with MMseqs2, we arrive at a set of 20,033,335 pairs that passed the LCS threshold (i.e., have a consecutive overlap of at least 5 nucleobases) and that need to be further analysed by the yield computation.

Compared to the brute-force approach, we have reduced the number of checks needed from over 5 billion to just 0.004%. Note that we do not lose anything in terms of accuracy, since the sequence pairs that we eliminate are so structurally different that it is meaningless to include them in the hybridisation computation.

For NUPACK, we estimate a running time of over 62 h—two and a half days. Comparatively, based on our results from Table 2 and Supplementary Information 13 we estimate that the RNN prediction time is around 6 minutes for second generation TPUs, with the CNNs expected to be even faster. This kind of workflow was previously impossible, and earlier strategies relied only on human-designed heuristics to filter sequences. With our approach, we apply precise estimations at a considerably larger scale than before.

Discussion

We presented the first comprehensive study of machine learning techniques applied to the problem of predicting DNA hybridisation, aiming to improve DNA storage applications both qualitatively and quantitatively, and to encourage further research into this expanding area, especially by harnessing the power of machine learning for large, *near-data processing*-enabled systems. As part of this investigation, we introduced a carefully designed dataset of more than 2.5 million sequence pairs with hybridisation scores for 6 different temperatures. We evaluated a range of classical machine learning methods as a baseline and followed by introducing deep learning architectures capable of accurately predicting yields while reducing inference time by one to over two orders of magnitude. We believe that this combination of high-fidelity and massive throughput is necessary for scaling up emerging DNA storage applications. Our work can act as a building block for new neural architectures, or on their own, the deep learning models can be used for DNA database library design of a scope that was previously impossible. Future studies could focus on generative methods that produce pairs matching a given yield, predicting base-pair probabilities (i.e., at the character level) or applying Graph Neural Networks on DNA strands represented as graphs, for example for edge prediction (topology of the duplex).

Methods

Dataset design. We start with a small number (e.g. 1000 or 2000) of randomly generated sequences where long repeats (≥ 3) of the same nucleobase are forbidden. We then perform three types of mutations: insertions, deletions and substitutions on each sequence. We incorporate both *minor* mutations (generally limited to a single type of mutation out of the three and which is applied once or twice) and *severe* mutations (all three types of mutations occur in the same sequence, possibly five or more times). To cover a wide range of thermodynamic interactions, we also apply the mutation operations on the reverse complement of the sequence. Thus, for each sequence, the mutation procedure generates a set of related sequences. We form a set of the initial randomly-generated sequences and all of their mutations, remove duplicates and generate all possible pairings. We repeat the procedure until the target size and variety constraints are achieved. To automate the yield computation process, we wrote Python wrappers for the NUPACK stand-alone executables. Each pair of inputs is written to a `.in` file, which is then fed to the `complexes` executable. In this work the default temperature is 57 °C, instead of the NUPACK default 37 °C; also see the discussion in *Pre-trained models allow estimation on different temperatures*. As with the length criterion, this is based on the consensus in molecular biology that primers should be designed broadly in the 50–65 °C range. The output of `complexes` is then used as input for the `concentrations` executable. Finally, we can read the result as a float and incorporate it into our dataset. Based on this numerical value, we can assign labels to each record. A good threshold, as suggested by[18], is 0.2. Thus, pairs reported to be below 0.2 are labelled **Low**, otherwise **High**.

Alignment scores. When used as a proxy for DNA annealing, we have to reverse complement one of the sequences. A high alignment score indicates that a sequence is very similar to the other's reverse complement. In this case, it is reasonable to assume that they will anneal. On the other hand, if the alignment score is low, then the sequence is dissimilar to the other's complement and we would not expect them to bind. However, as DNA interactions are very sensitive, it is possible to encounter pairs that do not strictly follow these rules. A few examples are illustrated in Supplementary Information 1, Fig. 3a,b. To perform alignments, we use parasail[26], a fast implementation of the Smith-Waterman and Needleman-Wunsch algorithms for local, and respectively global and semi-global sequence alignment. Out of the three alignment techniques, the one that fits our task is semi-global alignment. This is because we would expect that two compatible sequences can be aligned from end to end, perhaps with mismatches and gaps in-between. It is also possible that considerable (>4 nucleobases) deletions and insertions happened at the end of the sequences. This means that just global alignment is too weak, while local alignment is not optimal as we are not interested in matching isolated fragments of the sequences. The command used to compute the semi-global alignment of two sequences is:

```
parasail.sg_scan_16(str(s1), str(s2), 5, 2, parasail.dnafull)
```

where the input sequences `s1` and `s2` are Biopython[27] `Seq` objects. Also note that we are using a DNA substitution matrix, a gap open penalty of 5 and a gap extend penalty of 2. The graphical representation of the alignment can be computed by enabling the traceback feature:

```
parasail.sg_trace_scan_16(str(s1), str(s2), 5, 2, parasail.dnafull)
```

The resulting alignments, however, are not always representative of the actual binding that NUPACK predicts, as can be seen when comparing Supplementary Information 1 to the NUPACK outputs of Fig. 3a,b.

Classification metrics. The overall classification performance can be expressed through two metrics, the Matthews Correlation Coefficient (MCC) and Area Under the Receiver Operating Characteristics (AUROC). The AUROC is well-known and widely used in classification tasks, while the MCC has been recently argued to be more informative and reliable in binary classification tasks compared to popular metrics like accuracy and F_1 score[28,29]. We do not reproduce the mathematical formulae here; however, they can be accessed through the cited studies.

Convolutional neural networks. As exemplified in Fig. 3a,b, duplex interactions are complex and often do not correspond to a simple end-to-end alignment. To effectively incorporate this property, we used semi-local alignment for the baseline models. For CNNs, we first perform 2D convolutions with large filters of size 4×9, covering slightly under 35% of the maximum sequence length. We follow with 1D convolutions of size 9, 3, 3 and finally 1. The large filters in the first two layers ensure that spatial information is captured from a wide area, while the following layers, decreasing in size, can learn more specialised features. Our experiments confirm this choice, as this architecture is superior to stacked convolutions of the same size, e.g. 3×3 (2D) then 3 (1D) on our regression task. We also note that the small input size of $4 \times N \times 2$ limits the use of very deep models or alternative layers such as residual blocks.

Similar encodings have been used in the literature, although not for pairs of sequences and mostly in different settings, for example in[13,30,31]. In principle, an entire architecture can be devised using only 2D convolutions, by applying (zero) padding on the output of the first convolution operator (we used this approach in our work at DNA25). For this study, 1-dimensional convolutions follow the first 2D operation. This change saves millions of learnable parameters while reaching very similar performance. All convolutions are followed by ReLU activation and batch normalisation, with dropout interspersed throughout. Following the convolutional operations, fully-connected layers predict a single numerical value, the hybridisation yield. The entire network is trained under a supervised learning setting with NUPACK yields as ground truth.

Recurrent neural networks. Our sequential pair encoding requires the concatenation of two single-stranded sequences, not before inserting a special *separator* token ($< s >$ in Fig. 4b). Additionally, a special *beginning* token, represented by $< \texttt{b} >$ in Fig. 4b is prepended to the resulting sequence. Symmetrically, the separator token also marks the ending (last character). Each character is indexed into a vocabulary and then fed through an embedding layer. At this point, multiple recurrent layers can be stacked. The hidden states of the special beginning token ($< \texttt{s} >$) are extracted from the last layer and ran through a dense layer that outputs the hybridisation yield. As LSTM layers do not possess the same interpretable hyperparameters as CNNs (e.g. filter sizes), we perform hyperparameter optimisation (see Supplementary Information 5).

Transformers. A short description of a general Transformer architecture and the standard pre-training schemes is provided in Supplementary Information 6. We perform pre-training using the collection of all unique single-stranded DNA sequences in our dataset (totalling 2,579,555 samples). As we formulate our task at the character level, we have no appropriate equivalent for longer sequences of text and thus for the NSP task. Consequently, only MLM is used during pre-training, which we consider useful as predicting nucleobases based on the surrounding context and position is important for learning the structure of DNA sequences. The purpose of the pre-training scheme is learning to model DNA at the primary structure level. During fine-tuning, the pre-trained model is adjusted for the sentence-pair regression task by appending a fully-connected layer and propagating gradients throughout the entire model. We follow our intuition that modelling DNA interactions is not as complex as modelling natural language and since training Transformer models is extremely computationally demanding, we do not perform hyperparameter optimisation; instead, we base our architecture on documented models such as `roberta-base`. The particular framework we employ is HuggingFace's Transformers[32].

Measuring inference time. For each platform, we perform 10 trials (repetitions) after one warm-up trial and report the results. We took all precautions we are aware of to ensure that measuring the inference time is as precise and fair as possible. NUPACK was executed on a 64-core Azure VM equipped with the premium SSD option (an important observation since NUPACK outputs must be written and read from disk). Since NUPACK is a single-threaded application, we wrote Python wrappers around NUPACK and employed a `ProcessPool` from the `pathos` multiprocessing library to distribute NUPACK logic, reads and writes to disk across all available cores. This dramatically improved NUPACK running time from more than 5 h when running (in single-threaded mode) on an AMD Ryzen 5950X processor and a PCI Express 4.0 SSD to slightly more than 6 minutes on the Azure VM. This is a more than $\times 50$ improvement over the out-of-the-box NUPACK performance. The execution of the code was timed using the `time` shell command preceding the script invocation and the wall clock time was read and reported.

When measuring execution times on the GPU, additional measures were taken. More specifically, we overrode the PyTorch Lightning method `test_step` to perform only the forward pass and no additional computations (such as the loss function), and surrounded the call to the `test` method of the PyTorch Lightning `Trainer` object with two `torch.cuda.Event(enable_timing=True)` objects calling their `record` method. Before calculating the elapsed time, we wait for all kernels in all streams on the CUDA device to complete (`torch.cuda.synchronize`). A representative snippet of timing code for GPUs is provided in Supplementary Information 10. For GPUs, the PyTorch `DataLoader` object does not use multiple workers (`num_workers=0`). For consistency, we use the same setting for TPUs.

Finally, to the best of our knowledge, there currently is no established way to measure execution time on Tensor Processing Units (TPUs). To combat this, we implemented our own timing code which is consistent with the epoch time reported by PyTorch Lightning (note that the epoch time only reports elapsed seconds, i.e., low temporal resolution). The PyTorch Lightning method `test_step` is subjected to the same treatment as above. Our code defines a boolean flag inside the `LightningModule` to be timed; the first time the `forward` method is entered, the flag is set and the timestamp is recorded. The PyTorch Lightning method `on_test_epoch_end` is also overridden to capture a final timestamp and compute the elapsed time. A representative snippet of timing code for TPUs is provided in Supplementary Information 10.

Finding an appropriate Longest Common Substring. We start by investigating our 2.5 million pairs dataset. In particular, we looked only at pairs with high yield (above 0.2), which correspond to about 56% of the dataset, or more exactly, 1,437,758 elements. Out of the 1,437,758 high yield pairs, the shortest LCS among pairs is 4, which happens for only 29 pairs, or about 0.00002% of the high yield set. Also, the 29 pairs have an average yield of only 0.37, which is overall still quite low, even though it passes the threshold of 0.2. The pairs with an LCS of 5 are more numerous, at 349 samples, with an average yield of 0.47. We have thus decided that for our next experiment, working with a *minimum* LCS of 5 is the most appropriate. The running time performance difference between 4 and 5 is significant enough in the clustering stage described next to justify the trade-off. When working with sets of sequences of length 20, this means a minimum 25% identity match between sequences, which we find reasonable. The problem is now to efficiently process a large dataset such that sequences with shared regions of 25% or longer are associated. From our experience, the most appropriate tool for this task is the recently-developed MMseqs2. Briefly, MMseqs2 is described as an "*ultra fast and sensitive search and clustering suite*", and its performance is said to scale almost linearly with the number of processor cores.

Clustering with MMseqs2. Assuming a set of single stranded DNA sequences in FASTA format, the first step is to build a MMseqs2 database, with the following command:

```
mmseqs createdb <filename>.fasta <dbname> --dbtype 2 --max-seq-len <N>
```

The dbtype option indicates a nucleotide database. For example, in practice the command might look like:

```
mmseqs createdb init.fasta seqsDB --dbtype 2 --max-seq-len 20
```

The crucial step is the search. A fully configured search command looks like the following:

```
mmseqs search <dbname> <dbname> <alignname> <tmp> -s 7.5 -k 5
--max-seqs 10000 --exact-kmer-matching 1 --spaced-kmer-mode 0
--min-seq-id 0.25 --alignment-mode 4 -e inf
--min-aln-len 5 --sub-mat dnafull --max-seq-len 20
--strand 0 --search-type 3
```

A short description is in order. The dbname is the same as used in the database creation step, while alignname and tmp are names for the output. The sensitivity is set to the maximum (7.5). The search is to be done using *k*-mers of size 5. We allow a maximum of 10,000 related sequences to be returned for each individual query sequence. Exact *k*-mer matching is enabled, and spaced *k*-mer mode is disabled to ensure alignment only of ungapped regions. The minimum sequence similarity is defined to be 0.25. Alignment mode 4 is faster, and only considers ungapped alignments. The -e inf option targets similarity, and we want to consider all possibilities. The minimum alignment length is set to 5. The substitution matrix is set to dnafull, the one used for DNA sequences. We again specify a maximum sequence length of 20. The option -strand 0 is important, as it only searches for similar regions in the reverse complements of the set. Finally, a nucleotide search is specified by -search-type 3.

The low running time of this search allows this procedure to be carried out even on a laptop. On our portable experimental platform with 12 CPU threads, the search takes under 15 minutes. We have also verified the search speed on an Azure VM with 64 CPUs and the search finishes in under 5 seconds. The generated output file has 39,432,713 pairs, with a small sample reproduced in Supplementary Table 5.

We notice that it is possible for some sequences to be associated with themselves, i.e., have a pair of identical IDs. We eliminate these from the set, discovering that there are just 7,056 such pairs. We also notice that it is possible for symmetric pairs to occur, i.e., a pair such as (seq10, seq20) and (seq20, seq10). By eliminating one of these duplicates, we are left with 20,033,335 pairs out of the 39,425,657 from the previous filtering step, which is a 51% smaller set.

Data Availability

The author declares that the dataset and the source code supporting the findings of this study are hosted on GitHub at https://github.com/davidbuterez/dna-hyb-deep-learning.

Received: 20 April 2021; Accepted: 23 August 2021
Published online: 15 October 2021

References

1. Adleman, L. M. Molecular computation of solutions to combinatorial problems. *Science* **266**, 1021–1024 (1994).
2. Reinsel, D., Gantz, J. & Rydning, J. *The Digitization of the World From Edge to Core* tech. rep. (2018). https://www.seagate.com/files/www-content/our-story/trends/files/idc-seagate-dataage-whitepaper.pdf. Accessed 1 June 2019.
3. Carmean, D. *et al.* DNA data storage and hybrid molecular-electronic computing. *Proc. IEEE* **107**, 63–72 (2019).
4. Allentoft, M. E. *et al.* The half-life of DNA in bone: Measuring decay kinetics in 158 dated fossils. *Proc. R. Soc. B: Biol. Sci.* ISSN: 14712954 (2012).
5. Grass, R. N., Heckel, R., Puddu, M., Paunescu, D. & Stark, W. J. Robust chemical preservation of digital information on DNA in silica with error-correcting codes. *Angewandte Chemie - International Edition*. ISSN: 15213773 (2015).
6. Jain, M., Olsen, H. E., Paten, B. & Akeson, M. The Oxford Nanopore MinION: Delivery of nanopore sequencing to the genomics community. *Genome Biol.* **17**, 239. https://doi.org/10.1186/s13059-016-1103-0 (2016).
7. Appuswamy, R. *et al.OligoArchive: Using DNA in the DBMS storage hierarchy* in *Conference on Innovative Data Systems Research (CIDR)* (2019).
8. Goldman, N. *et al.* Towards practical, high-capacity, low-maintenance information storage in synthesized DNA. *Nature* **494**, 77–80. https://doi.org/10.1038/nature11875 (2013).
9. Rychlik, W. in *The Nucleic Acid Protocols Handbook* (ed Rapley, R.) 581–588 (Humana Press, 2000). ISBN: 978-1-59259-038-4. https://doi.org/10.1385/1-59259-038-1:581.
10. Yang, X., Scheffler, B. E. & Weston, L. A. Recent developments in primer design for DNA polymorphism and mRNA profiling in higher plants. *Plant Methods* **2**, 4. https://doi.org/10.1186/1746-4811-2-4 (2006).
11. Khabar, K. S., Dhalla, M., Bakheet, T., Sy, C. & al Haj, L. An integrated computational and laboratory approach for selective amplification of mRNAs containing the adenylate uridylate-rich element consensus sequence. *Genome Res.* **12**, 985–995 (2002).
12. Bustin, S. & Huggett, J. qPCR primer design revisited. *Biomol. Detect. Quant.* **14**, 19–28 (2017).
13. Stewart, K. *et al.A content-addressable DNA database with learned sequence encodings in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (2018). ISBN: 9783030000295.
14. Zhang, J. X. *et al.* Predicting DNA hybridization kinetics from sequence. *Nat. Chem.* **10**, 91–98. https://doi.org/10.1038/nchem.2877 (2018).
15. Bee, C. *et al.* Content-Based Similarity Search in Large-Scale DNA Data Storage Systems. bioRxiv. Eprint: https://www.biorxiv.org/content/early/2020/05/27/2020.05.25.115477.full.pdf. (2020).
16. Zadeh, J. N. *et al.* NUPACK: Analysis and design of nucleic acid systems. *J. Comput. Chem.* **32**, 170–173 (2011).

17. Fornace, M. E., Porubsky, N. J. & Pierce, N. A. A unified dynamic programming framework for the analysis of interacting nucleic acid strands: Enhanced models, scalability, and speed. *ACS Synth. Biol.* **9**, 2665–2678. https://doi.org/10.1021/512acssynbio.9b00523 (2020).
18. Beliveau, B. J. *et al.* OligoMiner provides a rapid, flexible environment for the design of genomescale oligonucleotide in situ hybridization probes. *Proceedings of the National Academy of Sciences*. ISSN: 0027-8424 (2018).
19. Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural Comput.* **9**, 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735 (1997).
20. Vaswani, A. *et al.* Attention Is All You Need. arXiv:1706.03762. (2017).
21. Tay, Y., Dehghani, M., Bahri, D. & Metzler, D. *Efficient Transformers: A Survey* 2020. arXiv:2009.06732 [cs.LG].
22. Liu, Y. *et al.* RoBERTa: A Robustly Optimized BERT Pretraining Approach. CoRR arXiv:1907.11692. (2019).
23. Irie, K., Tüske, Z., Alkhouli, T., Schlüter, R. & Ney, H. *LSTM, GRU, Highway and a Bit of Attention: An Empirical Overview for Language Modeling in Speech Recognition in Interspeech 2016* (ISCA, Sept. 2016). https://doi.org/10.21437/interspeech.2016-491.
24. Chung, J., Gulcehre, C., Cho, K. & Bengio, Y. *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling* 2014. arXiv:1412.3555 [cs.NE].
25. Steinegger, M. & Söding, J. MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nature Biotechnol.*. ISSN: 1087-0156. http://www.nature.com/doifinder/10.1038/nbt.3988 (Oct. 2017).
26. Daily, J. Parasail: SIMD C library for global, semi-global, and local pairwise sequence alignments. *BMC Bioinform.* **17**, 81. https://doi.org/10.1186/s12859-016-0930-z (2016).
27. Cock, P. J. A. *et al.* Biopython: Freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics* **25**, 1422–1423. https://doi.org/10.1093/bioinformatics/btp163 (2009).
28. Chicco, D. & Jurman, G. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genom.* **21**, 6. https://doi.org/10.1186/s12864-019-6413-7 (2020).
29. Chicco, D. Ten. quick tips for machine learning in computational biology. eng. *BioData Min.* **10**(29234465), 35–35 (2017).
30. Min, X. *et al.* Predicting enhancers with deep convolutional neural networks. *BMC Bioinform.* https://doi.org/10.1186/s12859-017-1878-3 (2017).
31. Zhang, Z. *et al.* Deep learning in omics: A survey and guideline. *Brief. Funct. Genom.* **18**, 41–57. https://doi.org/10.1093/bfgp/ely030 (2018).
32. Wolf, T. *et al.* HuggingFace's Transformers: State-of-the-art Natural Language Processing. *CoRR* https://arxiv.org/abs/1910.03771. (2019).

Acknowledgements
Work partially conceptualised and implemented at Imperial College London.

Author contributions
D.B. conceptualised the study, ran the experiments, wrote the manuscript text and prepared all the figures.

Competing interests
The author declares no competing interests.

Additional information
Supplementary Information The online version contains supplementary material available at https://doi.org/10.1038/s41598-021-97238-y.

Correspondence and requests for materials should be addressed to D.B.