Article

https://doi.org/10.1038/s41592-022-01700-2

Deep embedding and alignment of protein sequences

Received: 30 November 2021

Accepted: 24 October 2022

Published online: 15 December 2022

Check for updates

Felipe Llinares-López, Quentin Berthet, Mathieu Blondel, Olivier Teboul & Jean-Philippe Vert

Protein sequence alignment is a key component of most bioinformatics pipelines to study the structures and functions of proteins. Aligning highly divergent sequences remains, however, a difficult task that current algorithms often fail to perform accurately, leaving many proteins or open reading frames poorly annotated. Here we leverage recent advances in deep learning for language modeling and differentiable programming to propose DEDAL (deep embedding and differentiable alignment), a flexible model to align protein sequences and detect homologs. DEDAL is a machine learning-based model that learns to align sequences by observing large datasets of raw protein sequences and of correct alignments. Once trained, we show that DEDAL improves by up to two- or threefold the alignment correctness over existing methods on remote homologs and better discriminates remote homologs from evolutionarily unrelated sequences, paving the way to improvements on many downstream tasks relying on sequence alignment in structural and functional genomics.

Sequence alignment is a key component of bioinformatics pipelines to study the structures and functions of proteins, and to annotate open reading frames in newly sequenced genomes and metagenomes[1]. Indeed, aligning two protein sequences allows identifying homologous sequences with known structure or function, and regions of similarity that may be the result of evolutionary, structural or functional relationships. Jointly aligning multiple sequences further gives information about evolutionary constraints and patterns of coevolution along the sequence, which is, for example, useful for three-dimensional (3D) structure inference[2–4]. It is remarkable that algorithms for pairwise sequence alignment have basically remained unchanged since their invention in the 1980s and 1990s: in particular, the Smith–Waterman (SW) algorithm to find the best local alignment between two sequences in quadratic time with dynamic programming[5] and faster heuristics such as BLAST[6] or FASTA[7], which are also the first steps of more advanced methods based on multiple sequence alignment such as PSI-BLAST[8]. However, in spite of their immense popularity and importance for downstream applications, these algorithms often produce erroneous alignments or fail to detect homology, particularly for sequences with low similarity[9]. This leaves a sizable fraction of predicted open reading frames in genomics and metagenomics projects without annotation[10], while alignment errors can in turn result in wrong structural or functional annotations. Improving pairwise sequence alignment algorithms, particularly for divergent sequences, could directly benefit a number of downstream tasks.

The SW algorithm formulates the problem of finding the correct alignment between two sequences as a search for the best-scoring path in a graph. Each candidate path represents a possible alignment and the score of a path depends on user-defined parameters. These are the costs of starting or extending a gap in the alignment, and the substitution score of aligning each position of the first sequence to each position of the second one, which usually depends on the amino acids at both positions. This formulation, which leads to efficient algorithms to find or approximate the best-scoring alignment, has at least two weaknesses that may lead to possible errors in the alignment found. First, it lacks flexibility in the formulation of the score of a path, by enforcing, for example, that the cost of opening or extending a gap is the same wherever in the sequences, or by enforcing that the substitution score of aligning two positions only depends on the amino acids at those positions. Allowing more flexible parameterizations may give more opportunities to the algorithm to optimize a relevant score; for example, ref. [11] shows the benefits

Brain Team, Google Research, Paris, France. e-mail: jean-philippe.vert@m4x.org

of making the substitution score between two positions in protein sequences depend not only on the amino acids at those positions, but also on predicted structural properties, at the cost of increasing the number of parameters of the model. Second, the best alignment returned by the algorithm strongly depends on the choice of the parameters[12–18], and even for simple models where the substitution score only depends on the amino acids to be aligned, it is well known that there are no universally 'good' substitution scores that lead to good alignments for all protein pairs[19,20]. This raises the question of how to optimally choose adequate scoring parameters for a given pair of proteins.

In this work, we propose DEDAL (deep embedding and differentiable alignment), an algorithm for pairwise sequence alignments that addresses both issues. DEDAL builds on top of the standard SW algorithm to efficiently find an optimal alignment between two sequences, but provides a flexible parameterization of the scoring function used by the SW algorithm that adapts to each sequence pair and each position in each sequence. The parameterization is automatically learned during a training phase from a set of sequence pairs with known alignments, and a large set of raw protein sequences. It relies both on recent advances in deep learning language models, which embed discrete sequences in a continuous space and are automatically trained on a large corpus of raw sequences, and also a parameterization of the SW algorithm (gap and substitution parameters) as a function of the continuous embedding. To train DEDAL, we propose a smoothed variant of the SW algorithm to make the alignment solution a continuous and differentiable function of the scoring parameters. Given a set of sequence pairs with known correct alignment, we then tune automatically the various parameters of the model by end-to-end gradient-based optimization to minimize the alignment errors. Once trained, DEDAL produces gap and substitution scoring matrices computed specifically for each new pair of sequences. In addition, the gap and substitution scores are contextual: for each pair of positions, they depend on the full sequences to be aligned. The optimal alignment is then computed with a standard SW algorithm using those parameters. We show that DEDAL can be trained efficiently on modern hardware with accelerators. Once trained, we demonstrate that DEDAL improves by up to two- or threefold the quality of the alignment predicted for remote homologs compared to standard SW, and produces an alignment score that more accurately detects remote homology.

Related to this work, the fact that the solution of the SW algorithm depends on the scoring parameters has been studied in detail in the context of the so-called parametric sequence alignment problem, which aims to describe the set of possible solutions as a function of the parameters used[12–18]. This approach, however, is only feasible for simple models with up to two or three parameters that vary. Conversely, the idea to search for parameters that reproduce a set of given, correct alignments, was tackled by refs. [11,21–23] using various optimization techniques, but these studies focus on simple parameterizations where a fixed-size substitution matrix and one or two gap parameters are optimized. Reference [24] proposes exploitation of known alignments to help train deep embedding models with a differentiable loss; however, this model does not produce a sequence alignment once trained. The closest study to ours is the DeepBlast model in ref. [25], which also proposes learning a deep language model for proteins with a differentiable alignment module; however, the model has a simpler model for scores (linear instead of affine) and only produces global alignments. More recently, concurrently and independently from this work, ref. [26] presented a differentiable version of SW for multiple sequence alignment, albeit with a simple convolution layer instead of a full language model to embed the sequences, and simpler position-independent gap penalty. Moreover, the authors focus on contact and structure prediction tasks, as opposed to directly investigating alignment and homology detection performance as done here.

Results

Pairwise local alignment of protein sequences with DEDAL

We introduce DEDAL, a trainable algorithm for accurate pairwise local alignment of protein sequences (Fig. 1). DEDAL aligns sequences by computing substitution scores and gap penalties that are specific to the sequences being aligned (Fig. 1, top). To this end, DEDAL depends on parameters that are automatically tuned before inference during a training phase (Fig. 1, bottom).

For sake of clarity, let us first describe how DEDAL works once it is trained and ready to be used to align sequences. To align two sequences x and y and score the resulting alignment, DEDAL simply uses the standard SW algorithm for pairwise local alignments, but with gap open, gap extend and substitution score matrices that are computed specifically from x and y. To do so, a deep learning-based transformer encoder network T_ϕ with parameters ϕ (ref. [27]) is first used to independently obtain a continuous representation of each of these sequences. In this representation, each residue of each sequence is mapped to a vector in a vector space of a fixed dimension (we use $d = 768$ in our experiments). Crucially, these embeddings are contextual: that is, the embedding of each residue encodes information not only about the amino acid present at that position, but also about all other residues in the sequence, as well as their relative arrangement. This allows DEDAL to be highly flexible in the way sequences are represented, opting for a data-driven approach toward incorporating contextual information over hard-coded rules. Next, DEDAL computes a substitution score as well as gap open and gap extend penalties for each pair of residues from the sequences to be aligned, computed from their respective vector representations by a parameterizer function P_β, which depends on parameters β. Finally, the standard SW algorithm is used to compute an optimal alignment and score it, using the substitution scores, gap open and gap extend penalties computed in the previous step. In other words, DEDAL relies on the SW algorithm to align sequences and score the alignment, but provides a very flexible framework to parameterize the SW algorithm; in particular, the substitution score, gap open and gap extend penalties are specific to each pair of positions in the two input sequences, and depend on the full sequences through the contextual embedding of the transformer encoder and the parameterizer.

DEDAL thus depends on parameters ϕ for the transformer encoder T_ϕ, and β for the parameterizer P_β. These parameters are tuned automatically during a training phase, where we provide DEDAL with a large set of roughly 30 million nonredundant protein sequences from UniRef50 (ref. [28]) to train the transformer encoder, as well as a set of pairs of homologous sequences with curated correct alignment extracted from the set of roughly 1.2 million sequences organized in roughly 19,000 families in the Pfam-A seed database[29] to jointly train the transformer encoder and the parameterizer. More precisely, using these datasets, DEDAL tunes its parameters by end-to-end gradient-based optimization to minimize a loss function combining three tasks: (1) a so-called masked language modeling task on the UniRef50 dataset, which is a classical way to tune the parameters of a transformer encoder[30,31]; (2) an homology detection task, where we train DEDAL so that it can discriminate homologous from nonhomologous pairs in Pfam-A seed from the alignment score and (3) a learning to align task, where we train DEDAL to produce the correct alignment on homologous pairs from Pfam-A seed with known correct alignment. To solve the second and third tasks by gradient-based optimization, we implement a new differentiable variant of the SW algorithms that allows to backpropagate not only the alignment score, but also the error between the predicted and true alignments to the parameters ϕ and β. Since Pfam sequences represent protein domains, which therefore tend to be aligned from start to end within a given family, we perform data augmentation, extending these sequences by including flanks upstream and downstream of the domain so that DEDAL is trained to produce a correct local alignment. We refer the reader to the Methods section for more details about the model and the data. DEDAL is implemented in Python,

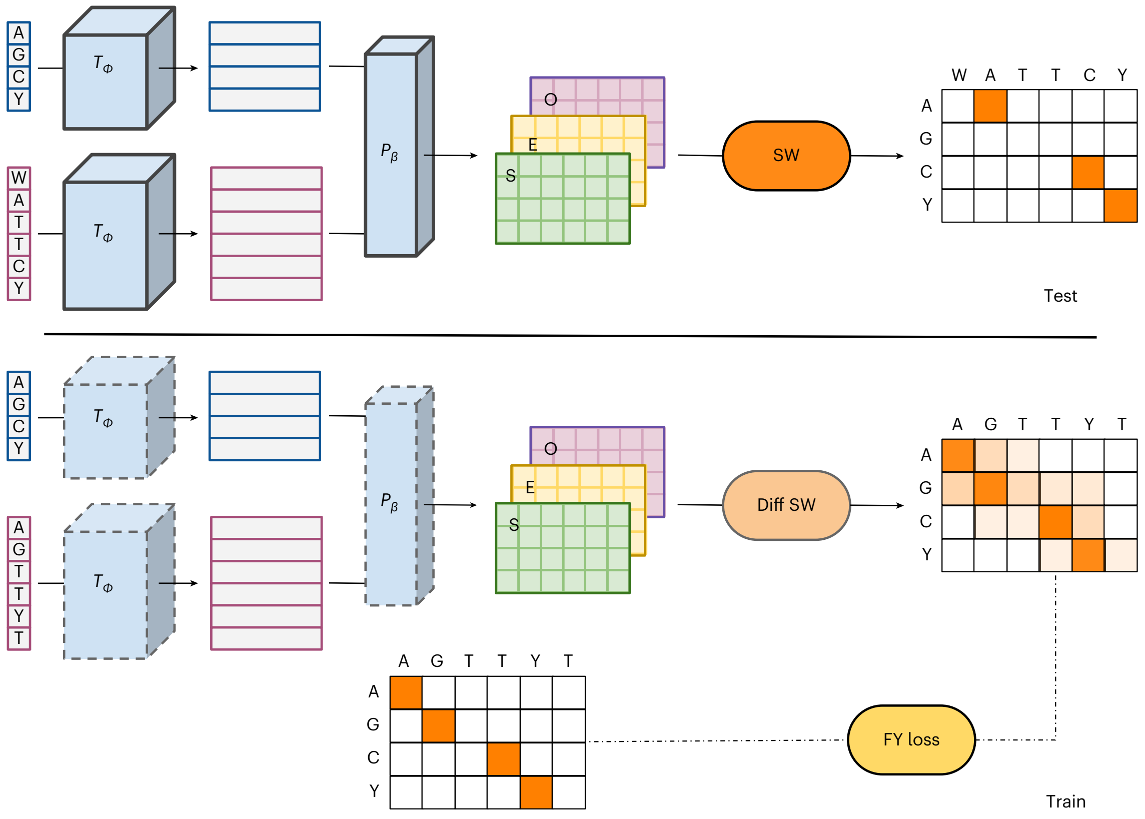


Fig. 1 | Overview of DEDAL. Top, at the test time, DEDAL aligns two sequences *x* and *y* by running the SW local alignment algorithm with position-specific parameters (gap open *O*, gap extend *E*, substitution scores *S*) that depend on the input sequences through a transformer encoding T_ϕ of each sequence followed by a parameterizer P_β that transforms the continuous representations into matrices of parameters needed by SW. Bottom, both T_ϕ and P_β depend on parameters ϕ and β that are learned during the training time from a large corpus of raw sequences to learn the language model T_ϕ, combined with pairs of sequences with known alignment to learn jointly T_ϕ and P_β by end-to-end optimization with a differentiable variant of SW and a specific alignment loss function.

using the TensorFlow[32] library for deep learning. It takes about 1 week using 32 tensor processing unit v.3 cores to train DEDAL.

DEDAL accurately aligns homologous sequences

We first assess the ability of DEDAL to accurately align homologous sequences. Since DEDAL is trained on a set of known correct alignments, we must evaluate its performance on sequences not seen during the training time. We therefore split the Pfam sequences into two nonoverlapping sets. The first one is used to train the model and to choose hyperparameters, and the second one to assess its performance. We consider two ways to make the split: (1) split sequences uniformly at random in Pfam, so that sequences in the test set come from families seen during training time (we call this setting the in-distribution setting), and (2) split clans between training and test sets, so that sequences in the test set come from clans that are not seen during training time (we call this setting the out-of-distribution setting). Obviously the out-of-distribution setting is more challenging, as it requires the model to generalize across clans. It mimics the situation where we want to align sequences from unknown domains. The in-distribution setting, on the other hand, mimics the common situation where we want to align sequences from known Pfam domains. In both cases, we keep the UniRef50 set of sequences used for the masked language model task, as we want to simulate the situation where a user wants to align a sequence from the 'protein universe' described by UniRef50 and known at training time, whether or not it is similar to sequences annotated in Pfam; in addition, in the Supplementary Information we provide more results where we only train the language model task on a subset of UniRef50 with no match to any of the Pfam families in the out-of-distribution test set to assess DEDAL's performance on sequences that are not only in Pfam clans not seen at training time, but are also substantially different from any sequence in the 'protein universe' seen at training time for the language model task. We measure the quality of a predicted alignment by the F_1 score of the prediction on the test set, and stratify the performance by percentage identity (PID) of the sequences in the correct alignment, as it is well known that the difficulty of aligning sequences increases when PID decreases. As a baseline, we compare DEDAL to the SW algorithm ran with a fixed PFASUM70 substitution matrix and an affine gap penalty model with gap open of 15 and gap extend of 1.5. We chose this configuration as the best among more than 1,400 combinations of substitution matrices in the BLOSUM[19], VTML[33,34] and PFASUM[20] families, and gap open and extend parameters (Methods).

Figure 2a,b summarizes the alignment performance of DEDAL and of the baseline, respectively in the in-distribution and out-of-distribution settings. A breakdown of the F_1 score into precision and recall is shown in the Supplementary Information, respectively. We see that DEDAL substantially outperforms the baseline both in-distribution and out-of-distribution. The difference is particularly strong in distribution (with an average relative improvement of 18% over baselines, from $F_1 = 0.744$ to $F_1 = 0.877$), which confirms the benefit of training DEDAL on Pfam families to then align new protein sequences within these families. The gap between DEDAL and baselines widens as the PID decreases, with a relative improvement of up to 286% in the hardest setting (PID ≤0.1), showing that DEDAL can provide good quality alignments ($F_1 = 0.587$) even between very remote homologs, while the baseline cannot ($F_1 = 0.152$). A similar pattern is visible on the out-of-distribution setting, where DEDAL outperforms the baselines by

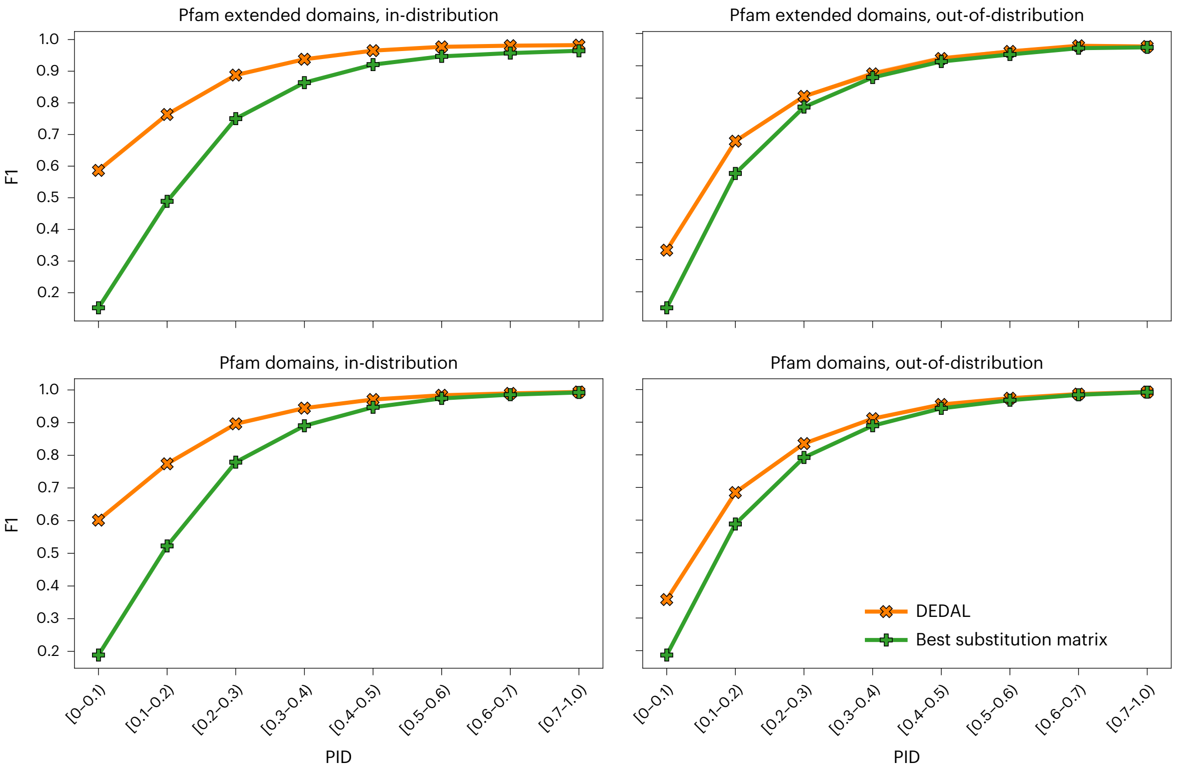


Fig. 2 | Alignment performance. These plots show the alignment F_1 score of DEDAL and the best-performing substitution matrix baseline, in the in- and out-of-distribution settings (respectively, left and right columns), and for Pfam extended or raw domains (respectively, top and bottom rows). Results for DEDAL were averaged over ten replicates, with error bars describing 95% confidence intervals.

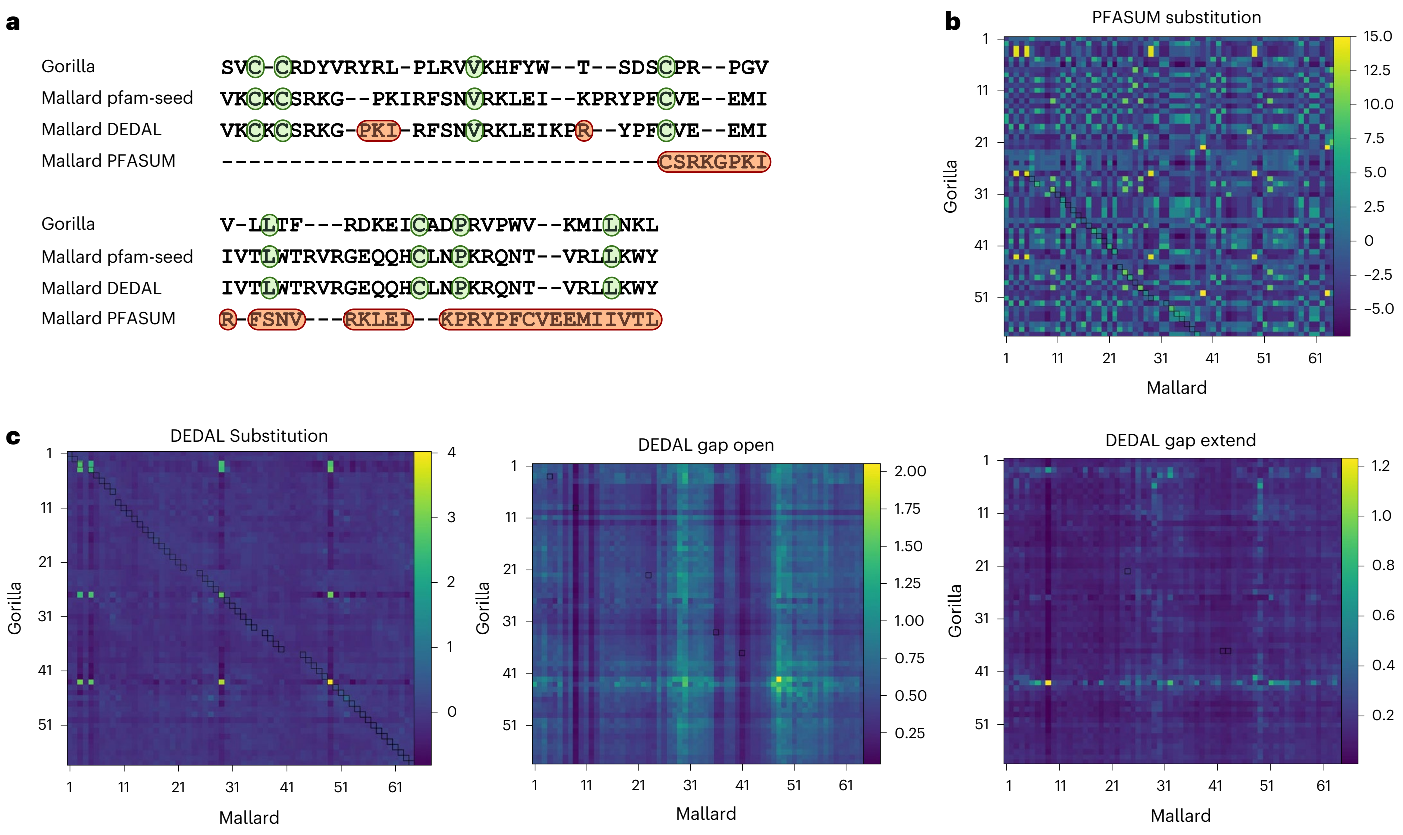
4% on average (from $F_1 = 0.782$ to $F_1 = 0.816$) and by 119% on the remote homologs with PID < 0.1 (from $F_1 = 0.150$ to $F_1 = 0.329$). This suggests that DEDAL learns a model of protein sequence similarity that extends beyond the biological subspace it sees during training time, and that DEDAL is particularly useful is difficult situations of remote homology, where standard methods perform poorly.

We then compare DEDAL and the baseline on their ability to align Pfam domains, as opposed to the extended domains that we use to train DEDAL. While the domain sequences are the same in both cases, the alignments are very different since two domain sequences in a Pfam family can generally be aligned from the beginning to the end, thus making the correct alignment closer to a global alignment than a local one. We show the results in this setting in Fig. 2c,d. Compared to the previous setting, we see that the performances of all methods tend to be better, which can be explained by the fact that the number of wrong candidate alignments is reduced when we focus on the domain sequence that must be aligned. Besides the overall improvement of all methods, DEDAL still substantially outperforms the baseline, particularly in-distribution and particularly in the low PID regime (with, for example, a relative F_1 increase of 218% for in-distribution domains with PID ≤0.1, from $F_1 = 0.189$ to $F_1 = 0.602$).

To clarify why DEDAL behaves so differently from a standard SW algorithm with a fixed substitution matrix and gap parameters, we now focus on a particular pairwise alignment between two domain sequences of Pfam-A seed (accession no. PF00048): a C–C motif chemokine from western lowland gorilla (UniProt A0A2I2ZUR5) and a C-X-C motif chemokine ligand 14 from northern mallard (UniProt U3IBZ2). Figure 3 summarizes the alignments produced by DEDAL and the baseline (using a PFASUM substitution matrix), and the parameters of both models. Note that this alignment is not part of DEDAL's training set. As seen in Fig. 3a, the sequences have only eight conserved residues in the Pfam-A seed curated alignment, meaning that this is a difficult case (PID = 0.12). Indeed, using SW with a PFASUM substitution matrix gives an alignment where no residue is correctly aligned ($F_1 = 0$), however, we see that using DEDAL gives an almost perfect alignment where only four residues are wrongly aligned, and in all cases with an error of only one residue in distance ($F_1 = 0.94$). To better understand the difference between PFASUM and DEDAL, we plot in Fig. 3b,c, respectively, the SW parameters for PFASUM (substitution scores) and for DEDAL (substitution scores, gap open and gap extend penalties). We see that the substitution score matrix produced by DEDAL is much smoother spatially than the one of PFASUM, highlighting the benefit of context-dependent embeddings of residues. We also see that DEDAL clearly picks the importance of aligning cysteine (C) residues, which are very conserved since they form disulfide bonds that stabilize the 3D structure of the domain (there are four cysteines in each sequences, respectively, at positions 3, 4, 27, 43 in gorilla and 3, 5, 29, 49 in mallard, corresponding to the 16 positions in DEDAL's substitution matrix with highest score). PFASUM, on the other hand, has high scores not only to match the cysteines, but also to match the two tryptophans (W) at respective positions 21 and 49 in the gorilla sequence, and 38 and 63 in the mallard sequence. While tryptophans are usually very conserved, and have therefore a large substitution score in standard substitution matrices such as PFASUM, it is interesting that in this particular case they should not be aligned according to the Pfam-A seed curated alignment and DEDAL correctly predicts it by producing a small substitution score. Finally, we note that DEDAL's gap open and extend penalties are far from uniform in the matrix: in particular, with strong penalties to open and extend gaps near the cysteines suggesting that DEDAL has learned that mutations are more likely than insertions or deletions near a cysteine involved in a disulfide bond.

DEDAL accurately detects remote homologs

Next, we seek to determine whether DEDAL's ability to align homologous sequences accurately also indicates that the alignment score it computes is effective to detect homology. To this end, we follow the



```
Gorilla             SVC-CRDYVRYRL-PLRVVKHFYW--T--SDSCPR--PGV
Mallard pfam-seed   VKCKCSRKG--PKIRFSNVRKLEI--KPRYPFCVE--EMI
Mallard DEDAL       VKCKCSRKG-PKI-RFSNVRKLEIKPR--YPFCVE--EMI
Mallard PFASUM      --------------------------------CSRKGPKI

Gorilla             V-LLTF---RDKEICADPRVPWV--KMILNKL
Mallard pfam-seed   IVTLWTRVRGEQQHCLNPKRQNT--VRLLKWY
Mallard DEDAL       IVTLWTRVRGEQQHCLNPKRQNT--VRLLKWY
Mallard PFASUM      R-FSNV---RKLEI--KPRYPFCVEEMIIVTL
```

Fig. 3 | Example of pairwise alignment of two protein domain sequences from Pfam-A seed. This figure illustrates how DEDAL differs from a standard alignment algorithm, on the case of a C–C motif chemokine from western lowland gorilla (UniProt A0A2I2ZUR5) and a C-X-C motif chemokine ligand 14 from northern mallard (UniProt U3IBZ2). **a**, Alignment respectively from the curated Pfam-A seed database (second row), predicted by DEDAL (third row) and predicted with a PFASUM70 substitution matrix (fourth row). We show all residues in both sequences for the Pfam-A seed and DEDAL alignments, but not the unaligned residues upstream and downstream of the alignment in the mallard sequence for PFASUM. Residues highlighted in green correspond to correctly aligned conserved residues, while those in red correspond to discrepancies between a predicted alignment and the Pfam-A seed one. **b**, Substitution scores between all pairs of residues from the PFASUM substitution matrix. **c**, SW parameters predicted by DEDAL.

same experimental setup as when probing alignment performance, namely considering pairs of Pfam extended domains or pairs of Pfam domains on the one hand, and considering pairs of candidate homologous sequences from families seen during training time (in-distribution setting) or from clans not seen in training time (out-of-distribution setting) on the other hand. For each of these settings, we measure the area under the receiver operating characteristic curve (AUROC) when predicting whether a pair of sequences belongs to the same Pfam clan or not based on the SW score, correcting the alignment score for sequence length as usually done when computing alignment E values to assess homology (Methods). To rank true homologs according to the 'difficulty' of detecting their relatedness, those originating from the same Pfam family are again stratified by PID whereas (remote) homologs belonging to the same Pfam clan but different Pfam families, whose ground-truth PID is unknown, are all assigned to a special 'Clans' bin. Unrelated pairs (negative class), whose ground-truth PID is also unknown, are included in all bins.

The results of this experiment are summarized in Fig. 4. We also show in the Supplementary Information an alternative method to evaluate performance using the area under the precision-recall curve. We see that all methods are largely successful in detecting homology for sequence pairs with moderate-to-large PID (≥0.2) in all settings. DEDAL is markedly better at detecting homology of highly divergent sequences across all four settings. For the smallest PID bin (<0.1), DEDAL improves in-distribution performance by 25% (from AUROC of 0.797 to 0.997) and 42% (from AUROC of 0.696 to 0.992) for Pfam domains and extended domains, respectively. Once again, the performance of DEDAL degrades in the out-of-distribution split while remaining substantially superior to the baselines, boosting AUROC by 23% (from AUROC of 0.770 to 0.948) and 30% (from AUROC of 0.698 to 0.910) for domains and extended domains, respectively. When detecting homology for sequences belonging to the same Pfam clan but different Pfam families ('Clans' bin), the baselines perform only slightly better than random guessing. In the in-distribution split, their AUROC reaches 0.611 and 0.550 for Pfam domains and extended domains, respectively. In contrast, DEDAL's performance in the in-distribution split for the 'Clans' bin is comparable to that of homologs from the same Pfam family in the smallest PID bin (<0.1), with corresponding AUROCs of 0.992 and 0.987 for domains and extended domains, respectively. In the out-of-distribution split, DEDAL exhibits a marked performance drop in the 'Clans' bin compared to the PID < 0.1 bin, suggesting a certain degree of memorization of clan-specific motifs. Still, its performance remains well above that of the baselines, with AUROCs of 0.811 and 0.725 for Pfam domains and extended domains, respectively.

Discussion

Using recent advances in deep language models with transformers and new differentiable alignment modules, we showed that DEDAL learns a continuous representation of protein sequences that, combined with the SW algorithm, leads to more accurate pairwise sequence alignment and homology detection than using the SW algorithm with fixed substitution matrices and gap penalties. The improvement is particularly striking in hard cases where we want to align remote homologs with limited sequence identity, and we showed that the model learned by

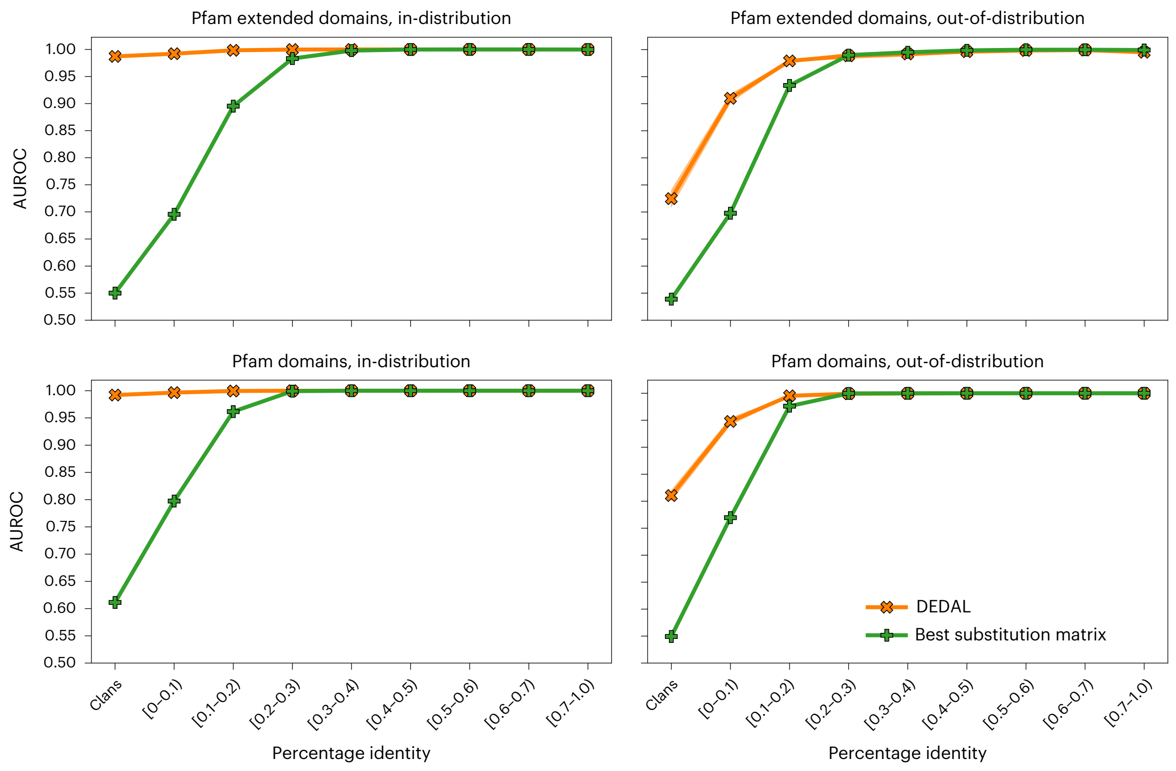


Fig. 4 | Homology detection performance. These plots show the homology detection AUROC of DEDAL and the best-performing substitution matrix baseline, in the in- and out-of-distribution settings (respectively, left and right columns) and for Pfam extended or raw domains (respectively, top and bottom rows). Results for DEDAL were averaged over ten replicates, with error bars describing 95% confidence intervals.

DEDAL on some sequence space generalizes well to new clans, suggesting that DEDAL learns universal biological properties not easily captured by standard substitution matrices and affine gap penalties.

Evaluating the performance of methods for remote homology detection is challenging, because by construction our current knowledge of homology relationships is mostly derived from existing bioinformatics tools, which have their own limitations. For the purpose of this study, we consider sequences in different Pfam clans (and with no nested domains) as nonhomologous, which is only a proxy to the biological definition of homology. Whereas there are arguably many unknown homologous pairs with very low sequence identity that we may be missing in current databases, we note that our definition of homology already creates many very challenging remote homology detection problems when sequences have low sequence identity, giving a useful benchmark to make progress on homology detection.

Since DEDAL incorporates several design choices, such as the architecture of the model or the strategies to train it, one may wonder which aspects are the most critical to explain its success. To address that question, we performed an ablation study where we systematically evaluated the effect of various design choices on the performance of DEDAL (Supplementary Information). In short, we found that replacing our rich parameterization of position-specific gap open and extend parameters by a simpler model where the gap open and extend parameters are position independent does not affect the performance of DEDAL much, achieving slightly superior alignment F_1 scores in the in-distribution split but marginally inferior results for out-of-distribution sequence pairs. Another simplification would be to replace our position-specific affine gap penalty by a position-specific linear gap penalty, as for example used in DeepBLAST[25], which, however, we found leads to a small performance drop for remote homologs that is most pronounced in the out-of-distribution split.

On the technical side, we explored two approaches to create a differentiable SW alignment module, needed to train the parameters of DEDAL in the 'learning to align' task, using either a smoothing technique[35] or a perturbation technique[36]; we found no significant difference between both in terms of performance, and implemented the one based on perturbations in the final DEDAL model. Regarding the set of alignments used to train DEDAL, we found that it is beneficial to use Pfam extended domains instead of Pfam domains when we want DEDAL to be able to predict accurate local alignments. Excluding sequences related to the out-of-distribution families from the 'protein universe' seen when pretraining DEDAL on the masked language modeling task led to a slight performance drop for remote homologs, albeit insignificant relative to the performance gap with respect to the baselines. Regarding the strategy to train end-to-end jointly the transformer and parameterizer, we found that this is indeed significantly better than a more classical, two-step strategy that would first train a transformer encoder on the masked language modeling task, and second the parameterizer on the 'learning to align' task by keeping the transformer fixed. This suggests that a universal language model, such as the one presented by ref. [31], is not sufficient and should be at least fine-tuned for optimal performance in alignment. Finally, we assessed the benefit of context-dependent embeddings by simply training a model where the substitution cost is constrained to depend only on the amino acids to be aligned; we observed a strong drop in performance for this model, reaching about the same performance as the best-performing substitution matrix in the literature. All in all, our ablation study indicates that DEDAL is robust to changes in several technical aspects of the model, but that the main idea to jointly train a deep language model and a flexible parameterizer for SW is key to its performance.

This work opens several research directions for the future. First, as pure language models are increasingly used for a variety of downstream tasks, it is interesting to explore whether adding alignment information to supervise language models, as DEDAL does, may also benefit other downstream tasks beyond sequence alignment. In preliminary experiments, we did not observe significant differences between the transformer encoder trained by DEDAL and one purely trained on a language modeling task on the TAPE benchmark[37], a collection of problems to assess the relevance of protein embedding models for various downstream tasks (Supplementary Information), but believe there is room for improvement. Second, it will be interesting to extend DEDAL to the multiple alignment setting (MSA), either by combining pairwise alignment operations, as proposed by ref. [26], which could directly benefit from DEDAL, or by directly embedding multiple nonaligned sequences and creating a differentiable MSA module. Third, given the accuracy of DEDAL's alignment and ability to capture homology, it will be interesting to develop fast approximations based, for example, on fast similarity search in continuous spaces, to allow accurate homology search and alignment in large databases.

Online content

Any methods, additional references, Nature Portfolio reporting summaries, source data, extended data, supplementary information, acknowledgements, peer review information; details of author contributions and competing interests; and statements of data and code availability are available at https://doi.org/10.1038/s41592-022-01700-2.

References

1. Prakash, T. & Taylor, T. D. Functional assignment of metagenomic data: challenges and applications. *Brief Bioinform.* **13**, 711–727 (2012).
2. Lockless, S. W. & Ranganathan, R. Evolutionarily conserved pathways of energetic connectivity in protein families. *Science* **286**, 295–299 (1999).
3. Marks, D. S. et al. Protein 3D structure computed from evolutionary sequence variation. *PloS ONE* **6**, e28766 (2011).
4. Jumper, J. et al. Highly accurate protein structure prediction with alphafold. *Nature* **596**, 583–589 (2021).
5. Smith, T. F. & Waterman, M. S. et al. Identification of common molecular subsequences. *J. Mol. Biol.* **147**, 195–197 (1981).
6. Altschul, S., Gish, W., Miller, W., Myers, E. & Lipman, D. Basic local alignment search tools. *J. Mol. Bol.* **215**, 403–410 (1990).
7. Pearson, W. R. Rapid and sensitive sequence comparisons with FASTP and FASTA. *Meth. Enzymol.* **183**, 63–98 (1990).
8. Altschul, S. et al. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.* **25**, 3389–3402 (1997).
9. Landan, G. & Graur, D. Characterization of pairwise and multiple sequence alignment errors. *Gene* **441**, 141–147 (2009).
10. Lobb, B., Kurtz, D. A., Moreno-Hagelsieb, G. & Doxey, A. C. Remote homology and the functions of metagenomic dark matter. *Front Genet.* **6**, 234 (2015).
11. Yu, C.-N. J., Joachims, T., Elber, R. & Pillardy, J. Support vector training of protein alignment models. *J. Comput. Biol.* **15**, 867–880 (2008).
12. Fitch, W. M. & Smith, T. F. Optimal sequence alignments. *Proc. Natl Acad. Sci. USA* **80**, 1382–1386 (1983).
13. Waterman, M. S., Eggert, M. & Lander, E. Parametric sequence comparisons. *Proc. Natl Acad. Sci. USA* **89**, 6090–6093 (1992).
14. Gusfield, D., Balasubramanian, K. & Naor, D. Parametric optimization of sequence alignment. *Algorithmica* **12**, 312–326 (1994).
15. Waterman, M. S. Parametric and ensemble sequence alignment algorithms. *Bull. Math. Biol.* **56**, 743–767 (1994).
16. Vingron, M. & Waterman, M. S. Sequence alignment and penalty choice. review of concepts, case studies and implications. *J. Mol. Biol.* **235**, 1–12 (1994).
17. Gusfield, D. & Stelling, P. Parametric and inverse-parametric sequence alignment with xparal. *Methods Enzymol.* **266**, 481–494 (1996).
18. Pachter, L. & Sturmfels, B. Parametric inference for biological sequence analysis. *Proc. Natl Acad. Sci. USA* **101**, 16138–16143 (2004).
19. Henikoff, S. & Henikoff, J. G. Amino acid substitution matrices from protein blocks. *Proc. Natl Acad. Sci. USA* **89**, 10915–10919 (1992).
20. Keul, F., Hess, M., Goesele, M. & Hamacher, K. Pfasum: a substitution matrix from pfam structural alignments. *BMC Bioinform.* **18**, 293 (2017).
21. Sun, F., Fernández-Baca, D. & Yu, W. Inverse Parametric Sequence Alignment. In *Computing and Combinatorics* (eds Ibarra, O. H. & Zhang, L.) 97–106 (Springer, 2002).
22. Saigo, H., Vert, J.-P. & Akutsu, T. Optimizing amino acid substitution matrices with a local alignment kernel. *BMC Bioinform.* **7**, 246 (2006).
23. Kececioglu, J. & Kim, E. Simple and Fast Inverse Alignment. In *Research in Computational Molecular Biology* (eds Apostolico, A. et al.) 441–455 (Springer, 2006).
24. Bepler, T. & Berger, B. Learning protein sequence embeddings using information from structure. In *7th International Conference on Learning Representations (ICLR)* (Openreview.net, 2019).
25. Morton, J. T. et al. Protein structural alignments from sequence. Preprint at *bioRxiv* https://doi.org/10.1101/2020.11.03.365932 (2020).
26. Petti, S. et al. End-to-end learning of multiple sequence alignments with differentiable Smith-Waterman. *Bioinformatics* https://doi.org/10.1093/bioinformatics/btac724 (2022).
27. Vaswani, A. et al. Attention is all you need. In *Proc. of the 31st International Conference on Neural Information Processing Systems* (eds Guyon, I. et al.), 5998–6008 (Curran Associates, Inc., 2017).
28. Suzek, B. E. et al. Uniref clusters: a comprehensive and scalable alternative for improving sequence similarity searches. *Bioinformatics* **31**, 926–932 (2015).
29. Mistry, J. et al. Pfam: the protein families database in 2021. *Nucleic Acids Res.* **49**, D412–D419 (2021).
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, NAACL-HLT 2019* Vol. 1 (Long and Short Papers) (eds Burstein, J. et al.) 4171–4186 (Association for Computational Linguistics, 2019).
31. Rives, A. et al. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proc. Natl Acad. Sci. USA* **118**, e2016239118 (2021).
32. Abadi, M. et al. TensorFlow: large-scale machine learning on heterogeneous systems (tensorflow.org, 2015); https://www.tensorflow.org/
33. Müller, T. & Vingron, M. Modeling amino acid replacement. *J. Comput. Biol.* **7**, 761–776 (2000).
34. Müller, T., Spang, R. & Vingron, M. Estimating amino acid substitution models: a comparison of dayhoff's estimator, the resolvent approach and a maximum likelihood method. *Mol. Biol. Evol.* **19**, 8–13 (2002).
35. Mensch, A. & Blondel, M. Differentiable dynamic programming for structured prediction and attention. In *Proc. 35th International Conference on Machine Learning* Vol. 80 (eds Dy, J. & Krause, A.) 3462–3471 (PMLR, 2018).
36. Berthet, Q. et al. Learning with differentiable perturbed optimizers. In *Proc. of the 34th International Conference on Neural Information Processing Systems* (eds Larochelle, H. et al.) 9508–9519 (Curran Associates, Inc., 2020).
37. Rao, R. et al. Evaluating protein transfer learning with TAPE. *Adv. Neural Inf. Process. Syst.* **32**, 9689–9701 (2019).

Methods

Data

To train the transformer encoder with a masked language modeling task, we used 30,162,111 cluster representative sequences from the March 2018 release of the UniRef50 database[28]. These representatives provide a nonredundant yet complete coverage of UniProtKB[38], obtained by clustering sequences with at least 50% identity and 80% overlap with the cluster's longest sequence using MMseqs2 (ref. [39]). A representative for each cluster was chosen accounting for factors such as quality of the UniProtKB entry and availability of annotations, among others. Given the relatively low redundancy between UniRef50 representative sequences, sequences were allocated to train or test uniformly at random as in ref. [31], resulting in training and test sets of size 27,052,653 and 3,019,006, respectively. We used the training set to define the loss of the masked language model task, and the test set to monitor its convergence during optimization of the training loss. A small validation set of 90,452 sequences was further reserved for hyperparameter tuning. For computational efficiency, we limit the length of sequences fed to the model for this task to at most 1,023 amino acids. Longer sequences are cropped to length 1,023 with a uniformly distributed offset sampled on-the-fly at training and evaluation time.

To create pairs of homologous sequences with known alignment, we collected 19,179 manually curated MSAs from release 34.0 of the Pfam-A seed database[29], grouped into 12,452 evolutionarily related clans (including 645 Pfam clans that altogether comprise 7,372 families, and the remaining 11,807 families that we assigned to their own clan). Together, these span a total of 1,223,021 sequences from which we kept only the 1,198,870 sequences (98% of the total) of length smaller than 512. We define a pair of homologous sequences as a pair of sequences in the same clan, from which we extract a correct pairwise alignment when both sequences are in the same family. Any two sequences not in the same clan are deemed as not homologous in the homology detection task if they share no other annotations at the clan level. Sequences not in the same clan but with common annotations (as could be the case, for example, for nested domains) are deemed ambiguous and excluded from the analysis. These represent less than 0.1% of nonhomologous sequence pairs. From this set of Pfam domain sequence pairs, we also build a set of Pfam extended domain sequence pairs by adding flanking sequences with random lengths upstream and downstream of each domain sequence. For each pair of sequences, these flanks are randomly chosen (sub)sequences from the UniProtKB protein sequence database sharing no clan annotations with either the domain sequence or flanks of the other Pfam extended domain sequence in the pair (Supplementary Information). As a result, the correct alignment between two homologous extended domains usually does not span the full sequences. For both Pfam domains and extended domains, we split the full dataset into five disjoint splits: train (80%), in-distribution validation (2.5%), out-of-distribution validation (2.5%), in-distribution test (7.5%) and out-of-distribution test (7.5%). To this end, we first select 255 and 955 out of the 12, 452 Pfam clans uniformly at random to form the out-of-distribution validation and out-of-distribution test, respectively. The number of clans was chosen so that these splits contain approximately 2.5 and 7.5% of the total number of sequences. Finally, we divide the set of sequences in the remaining 10,493 clans uniformly at random to form the train, in-distribution validation and in-distribution test splits so that a single family may have sequences in all three sets.

The alignment task is then defined by all pairs of sequences in each split belonging to the same Pfam family, resulting in 126,176,858 training, 111,031 in-distribution validation, 1,101,061 in-distribution test, 5,059,395 out-of-distribution validation and 11,558,433 out-of-distribution test samples, respectively. For the homology task, we complement these homologous pairs (positive class) with pairs of sequences belonging to different Pfam clans (negative class) as well as pairs of sequences belonging to the same Pfam clan but different Pfam families ('hard' cases in the positive class). These are sampled uniformly at random from their respective sets so that they make 50 and 11% of the total data in each split, respectively. These proportions were motivated by having a balanced class ratio as well as a moderate number of remote homologs in the positive class (roughly 25% including sequence pairs from the same Pfam family with PID < 0.1).

DEDAL model

DEDAL is a parametric model that transforms a pair of sequences onto three matrices of gap open, gap extend penalties and substitution costs, respectively, and then uses the SW algorithm to align the two initial sequences using these matrices of parameters. Each protein sequence is seen as a sequence of tokens in a 27-letter alphabet (22 amino acids including pyrrolysine (O) and selenocysteine (U), three ambiguous codes B, Z and X, and two special tokens <EOS>, which we append to each sequence, and <MASK>, which is used for masked language modeling only). Each such sequence is mapped to a sequence of vectors by a transformer encoder neural network similar to the smallest transformer-based model of ref. [31] that uses $L = 6$ transformer encoder layers with $h = 12$ heads per layer and embeddings of dimension $d = 768$. Each transformer encoder layer combines multi-head self-attention, position-wise multilayer perceptron, dropout and layer normalization. From the output of the transformer, we compute the substitution score between two positions as a bilinear function of their vector representations, and the gap and open penalties as the softplus function applied to a bilinear function of their vector representations to ensure they are nonnegative. An in-depth description of the DEDAL model as well as of its architecture and hyperparameters can be found in the Supplementary Information.

Differentiable SW alignment

Given two sequences $x = x_1 \ldots x_n$ and $y = y_1 \ldots y_m$ of amino acids, an alignment π is a subset of distinct positions in x and y that are matched in increasing order. Let us denote by $\Pi(x, y)$ the set of possible alignments between x and y, including the empty one. Given three $n \times m$ matrices S (substitution scores), O (gap open) and E (gap extend penalties), the local alignment score $s(\pi)$ of a candidate alignment π is defined as the sum of the substitution scores over matched positions, and gap penalties defined as the gap open penalty where a gap starts and gap extend penalties where it continues. The SW algorithm finds the maximum scoring alignment with the dynamic programming recursion over the $(n + 1) \times (m + 1)$ matrices M, X and Y defined for any $i = 1, \ldots, n$ and $j = 1, \ldots, m$ by $M_{i,0} = M_{0,j} = X_{i,0} = X_{0,j} = Y_{i,0} = Y_{0,j} = -\infty$ and:

$$\begin{cases} M_{i,j} = S_{i,j} + \max(0, M_{i-1,j-1}, X_{i-1,j-1}, Y_{i-1,j-1}), \\ X_{i,j} = \max(M_{i,j-1} - O_{i,j}, X_{i,j-1} - E_{i,j}), \\ Y_{i,j} = \max(M_{i-1,j} - O_{i,j}, X_{i-1,j} - O_{i,j}, Y_{i-1,j} - E_{i,j}). \end{cases} \quad (1)$$

The score of the best alignment is then $s^*(x, y) = \max_{\pi \in \Pi(x,y)} s(\pi) = \max\left(0, \max_{i,j}(M_{i,j})\right)$, and the best-scoring alignment $\pi^*(x, y) = \mathrm{argmax}_{\pi \in \Pi(x,y)} s(\pi)$ can be recovered by backtracking the argmax in the recursion. Both operations can be implemented in $O(nm)$ complexity in time and space; to benefit from modern computational infrastructure, we implemented a version optimized for hardware accelerators such as graphical and tensor processing units, which practically runs in $O(\max(m, n))$ time and in parallel across a batch of sequences to align, using a wavefront algorithm (Supplementary Information). We note that the score of any alignment $\pi \in \Pi(x, y)$ is a linear function of the parameters $\Theta = (S, O, E)$ seen as a $3nm$-dimensional vector, which we write as $s(\pi) = \Theta^\top \Phi(\pi)$ for some mapping $\Phi : \Pi(x, y) \to \mathbb{R}^{3nm}$. This shows that $s^*(x, y)$ is almost everywhere differentiable in Θ, with gradient $\Phi(\pi^*(x, y))$, which potentially allows to backpropagate to Θ any differentiable objective function that depends on the optimal alignment score, such as the one used in the homology detection task. On the other hand, the best alignment $\pi^*(x, y)$ is a piecewise constant function of Θ and any loss that only depends on $\pi^*(x, y)$, such as how different it is from the correct alignment, cannot be backpropagated to the parameters. In the 'learning to align' task, however, we need to have a differentiable loss to minimize with respect to Θ when SW finds an alignment $\pi^*(x, y)$ but the correct alignment is $\bar{\pi}$, potentially different from $\pi^*(x, y)$.

To overcome this issue, a first idea is to consider the so-called structured perceptron loss[40], which penalizes the difference between the score of the best alignment and the score of the true alignment: $\ell_{\text{Perceptron}}(\ ) = s(\bar{\pi}) - s^*(x, y)$. Its gradient is simply $\Phi(\bar{\pi}) - \Phi(\pi^*(x, y))$, which can be computed with the SW forward and backward recursions. To create a smoother loss function, we consider two natural extensions to the perceptron loss. The first one is the so-called conditional random field (CRF) loss[41] given by $\ell_{\text{CRF}}(\ ) = s(\bar{\pi}) - s^*_\tau(x, y)$ for $\tau \geq 0$. Instead of being the maximum score, $s^*_\tau(x, y) = \tau \log \sum_{\pi \in \Pi(x,y)} e^{s(\pi)/\tau}$ is the soft-maximum score. The CRF loss can also be interpreted as a negative log-likelihood model under a Gibbs distribution and, as shown in ref. [35], its value and gradient can be computed as efficiently as for the perceptron loss, by simply replacing the max operation in the SW recursion (1) by a softmax of the form $\max_\tau(\{u_1, \ldots, u_k\}) = \tau \log \sum_{i=1}^{k} e^{u_k/\tau}$, and adapting the backtracking. The CRF loss thus provides a smooth approximation to the perceptron loss, which converges to it when τ goes to zero.

The second extension creates smoothness by random perturbation of Θ, as proposed by ref. [36]. More precisely, we consider a random perturbation $\Theta + \tau Z$ of the parameters Θ, where $\tau \geq 0$ and Z is a standard multivariate normal random vector; then the so-called Fenchel–Young loss[42] associated to the perturbation according to ref. [36] is a valid loss with gradient $\nabla \ell_{\text{FY}}(\ ) = \Phi(\bar{\pi}) - E_Z \Phi(\pi^*(x, y))$. In other words, ℓ_{FY} is a valid loss whose gradient is the expectation of the perceptron loss gradient under random perturbation of the SW parameters. In practice, we compute a stochastic gradient (which is enough to optimize the loss function by stochastic gradient descent) by taking the perceptron loss gradient after a single random perturbation of the parameters by τZ. We note, again, that the Fenchel–Young loss is a smooth approximation to the perceptron loss, which converges to it when τ goes to 0.

Both extensions not only encourage the correct alignment to have a 'large' score, but they also encourage it to have a 'better' score than other candidate alignments. One can see this by noticing that the CRF loss, for example, is the negative log-probability of the correct alignment among all possible alignments under a Gibbs distribution where the probability of each alignment π is proportional to $e^{s(\pi)/\tau}$; since the probabilities of all alignments sum to 1, encouraging the probability of the correct alignment to increase (by minimizing the CRF loss) can only be achieved by not only increasing the score of the correct alignment, but also decreasing the scores of other, alternative alignments, hence encouraging the correct alignment to be the unique maximizer of the alignment score.

Homology detection classifier. We treat homology detection as a binary classification problem, aiming to explain whether a pair of sequences x and y belong to the same Pfam family or not, given their alignment score $s^*(x, y)$. Since it is well known that the distribution of alignment scores on random sequences strongly depends on their length[43], we must correct for the length effect on the score and pose the following model, inspired by how E values are defined to assess the significance of alignment scores, for the probability P that x and y are related given their alignment score and lengths n and m:

$$P = \text{Sigmoid}(w_1 s^*(x, y) - w_2 \log(nm) + b), \tag{2}$$

where $w_1, w_2 \in \mathbb{R}_+$ and $b \in \mathbb{R}$ are parameters. These parameters are jointly trained end-to-end alongside the sequence encoder and the differentiable alignment layer using a binary cross-entropy loss. The differentiable alignment layer is thus explicitly trained to output small scores for unrelated sequence pairs, preventing the model from acquiring biases caused by it only being trained on related sequences by the alignment task.

Output head for masked language modeling. We follow refs. [30,31] and define a masked language modeling task to train the transformer by randomly selecting 15% of the residues in each sequence to be prediction targets: 80% of these are substituted for a special <MASK> token, 10% by a randomly chosen residue and the remaining 10% is left unchanged. For a sequence x of length n, we denote by $\mathcal{T} \subseteq [\![1, n]\!]$ the set of target residues, chosen uniformly at random, and by $\tilde{x} \approx \text{perturb}(x; \mathcal{T})$ the resulting randomly perturbed sequence. The goal of the masked language modeling output head is to predict the original value of each perturbed token x_i from the embeddings of the (perturbed) sequence $\phi(\tilde{x})$ as

$$\hat{P}(x_i|\tilde{x}) = (\text{Softmax} \circ W^{\text{LM}})(\phi_i(\tilde{x})), \tag{3}$$

where $W^{\text{LM}} : \mathbb{R}^d \to \mathbb{R}^k$ is a learnable, affine mapping that is not tied to F in the input embedding layer (Supplementary Information). These predicted probabilities $\hat{P}(x_i|\tilde{x}) \in \Delta^{k-1}$ are then compared to the ground-truth using a cross-entropy loss,

$$L_{\text{LM}} := \mathbb{E}_x \mathbb{E}_{\mathcal{T}} \mathbb{E}_{\tilde{x}|x,\mathcal{T}} \left[\sum_{i \in \mathcal{T}} -\log \hat{P}(x_i|\tilde{x}) \right], \tag{4}$$

where we omitted the dependence of L_{LM} on the parameters ϕ of the sequence encoder and W^{LM}.

Training DEDAL. We first pretrain the sequence encoder on the masked language modeling task for 1 million steps. After that, we jointly train the sequence encoder, differentiable alignment layer, homology detection classifier and masked language modeling output head for 1.3 million additional steps to minimize a multi-task objective:

$$L_{\text{MT}} := L_\tau(x_{\text{AL}}, y_{\text{AL}}) + \lambda_{\text{HD}} L_{\text{HD}}(x_{\text{HD}}, y_{\text{HD}}) + \lambda_{\text{LM}} L_{\text{LM}}(\tilde{x}_{\text{LM}}, x_{\text{LM}}), \tag{5}$$

where L_τ is a loss for the alignment task dependent on the relaxation scheme, L_{HD} the binary cross-entropy loss for the homology detection task, L_{LM} the cross-entropy loss for the masked language modeling task and $\lambda_{\text{HD}}, \lambda_{\text{LM}} \in \mathbb{R}_+$ are hyperparameters controlling the relative weight of each loss on the multi-task objective. The batches $(x_{\text{AL}}, y_{\text{AL}})$, $(x_{\text{HD}}, y_{\text{HD}})$ and $(\tilde{x}_{\text{LM}}, x_{\text{LM}})$ for the alignment, homology and masked language modeling tasks are sampled independently of each other at each training step. We found it advantageous to disable the homology task (effectively using $\lambda_{\text{HD}} = 0$) for the first 300,000 steps. In all three training phases (pretraining, fine-tuning without homology and fine-tuning with homology), we use the Adam[44] optimizer with a linear warm-up of 8,000 steps until reaching a maximum learning rate lr_{max} that we treat as a hyperparameter, followed by an inverse square root decay schedule.

Metrics

Given two homologous sequences x and y from an evaluation set $\mathcal{D}_{\text{eval}}$ with known correct alignment, let us denote by $m_{\text{true}}(x, y)$ the set of pairs of indices aligned to each other in the correct alignment, and by $m^*(x, y)$ the set of pairs of indices predicted to be aligned by a SW prediction. We assess the performance of the SW prediction in terms of precision, recall and F_1 score defined as:

$$\begin{aligned}
\text{Precision} &:= \frac{\sum_{(x,y,m_{\text{true}}(x,y)) \sim \mathcal{D}_{\text{eval}}} |m^*(x,y) \cap m_{\text{true}}(x,y)|}{\sum_{(x,y,m_{\text{true}}(x,y)) \sim \mathcal{D}_{\text{eval}}} |m^*(x,y)|}, \\
\text{Recall} &:= \frac{\sum_{(x,y,m_{\text{true}}(x,y)) \sim \mathcal{D}_{\text{eval}}} |m^*(x,y) \cap m_{\text{true}}(x,y)|}{\sum_{(x,y,m_{\text{true}}(x,y)) \sim \mathcal{D}_{\text{eval}}} |m_{\text{true}}(x,y)|}, \\
\text{F1} &:= 2 \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}.
\end{aligned} \tag{6}$$

To evaluate homology detection performance, we use the AUROC and the area under the precision-recall curve.

To better reflect the relative difficulty of each task, we stratify all alignment and homology detection metrics by PID[45], using the definition of PID from[46], namely,

$$\text{PID} := \frac{\text{IdenticalResidues}}{\text{AlignedResidues} + \text{InternalGaps}}, \qquad (7)$$

where the number of identical residues and internal gaps is computed for each pair of related sequences using the ground-truth alignments. For homology detection, only a subset of the positive class (sequence pairs belonging to the same Pfam family) can be stratified in this manner. Negative pairs, for which no ground-truth alignments are available, are shared across all PID bins while positive pairs belonging to different Pfam families, whose curated PIDs are also unknown, are stratified as their own 'Clans' bin. Since we chose a version of PID that disregards external gaps, it is largely unaffected by our data augmentation process where we add flanking regions to Pfam domains (Supplementary Fig. 7).

All reported metrics were evaluated in the entirety of the (in- and out-of-distribution) test splits for both the Pfam extended and raw domain settings.

Baselines
We compared the performance of DEDAL on the alignment and homology detection tasks to three widely used families of substitution matrices: BLOSUM[19], VTML[33,34] and PFASUM[20]. For each family, we exhaustively optimized over the following hyperparameters: (1) the matrix number in the family, where we considered four matrix numbers for the BLOSUM family (45, 50, 62, 80), six for the VTML family (10, 20, 40, 80, 160, 200) and, finally, seven for the PFASUM family (31, 43, 51, 60, 70, 80, 90); (2) the gap open penalty, taking any integer value between 5 and 18 and (3) the gap extent penalty, taking values in {0.5, 0.75, 1.0, 1.25, 1.5, 2.0}. We selected the best-performing configuration over these 1,428 combinations as the baseline representing substitution matrices, following the protocol described in the Supplementary Information.

Reporting summary
Further information on research design is available in the Nature Portfolio Reporting Summary linked to this article.

Data availability
The Uniref50 dataset is freely available under the Creative Commons Attribution (CC BY 4.0) License from https://www.uniprot.org. We used the 2018-03 release available at ftp.uniprot.org/pub/databases/uniprot/previous_major_releases/release-2018_03/uniref/. Pfam is freely available under the Creative Commons Zero ('CC0') licence from https://pfam.xfam.org. We used the Pfam-A seed dataset available at http://ftp.ebi.ac.uk/pub/databases/Pfam/releases/Pfam34.0/Pfam-A.seed.gz. Source data are provided with this paper.

Code availability
The code used in this study is freely available under an Apache v.2.0 license at https://github.com/google-research/google-research/tree/master/dedal. All analysis was carried out in Python v.3.6, with the following libraries: absl-py 0.7, gin-config 0.4, numpy 1.18.4, tensorflow 2.3, tensorflow_datasets 3.0, tensorflow_probability 0.1 and tf-models-official 2.6.

References
38. The UniProt Consortium. UniProt: the universal protein knowledgebase in 2021. *Nucleic Acids Res.* **49**, D480–D489 (2021).
39. Steinegger, M. & Söding, J. Mmseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nat. Biotechnol.* **35**, 1026–1028 (2017).
40. Collins, M. Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. In *EMNLP '02: Proc. ACL-02 Conference on Empirical Methods in Natural Language Processing* Vol. 10, 1–8 (Association for Computational Linguistics, 2002).
41. Lafferty, J., McCallum, A. & Pereira, F. Conditional random fields: probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning* (eds Brodley, C. & Danyluk, A.) 282–289 (Morgan Kaufmann, 2001).
42. Blondel, M., Martins, A. F. & Niculae, V. Learning with Fenchel–Young losses. *J. Mach. Learn. Res.* **21**, 1–69 (2020).
43. Karlin, S. & Altschul, S. F. Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. *Proc. Natl Acad. Sci. USA* **87**, 2264–2268 (1990).
44. Kingma, D. P. & Ba, J. Adam: a method for stochastic optimization. In *Proc. 3rd International Conference on Learning Representations, ICLR 2015, Conference Track Proceedings* (eds Bengio, Y. & LeCun, Y.) (ICLR, 2015).
45. Raghava, G. P. & Barton, G. J. Quantifications of the variation in percentage identity for protein sequence alignments. *BMC Bioinform.* **7**, 415 (2006).
46. Doolittle, R. F. Similar amino acid sequences: chance or common ancestry? *Science* **214**, 149–159 (1981).

Acknowledgements
We thank D. Belanger, D. Dohan and A. Gane for preprocessing the data files and implementing the input pipelines for UniRef50 and M. Bileschi for providing feedback to improve the manuscript and guidance for navigating the Pfam database.

Author contributions
F.L.-L. contributed to the development of the method, implemented most of the code, designed and ran most of the experiments and drafted the manuscript. Q.B. contributed to the development of the method, to the code and to the experiments. M.B. contributed to the development of the method and to the code. O.T. designed and implemented large parts of the code and contributed to the experiments. J.P.-V. designed the initial project, contributed to the method development and drafted the manuscript. All authors provided regular feedback to all aspects of the work, reviewed code and contributed to the writing of the final manuscript.

Competing interests
This study was funded by Google LLC. F.L.-L., Q.B., M.B., O.T. and J.-P.V. were employees of Google LLC and owned Alphabet stock as part of the standard compensation package during the making of this study.

Additional information
Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41592-022-01700-2.

Correspondence and requests for materials should be addressed to Jean-Philippe Vert.

Peer review information *Nature Methods* thanks Sean Eddy and Rama Ranganathan for their contribution to the peer review of this work. Primary Handling Editor: Arunima Singh, in collaboration with the *Nature Methods* team. Peer reviewer reports are available.

nature portfolio

Corresponding author(s): Jean-Philippe Vert

Last updated by author(s): Sep 24, 2022

Reporting Summary

Nature Portfolio wishes to improve the reproducibility of the work that we publish. This form provides structure for consistency and transparency in reporting. For further information on Nature Portfolio policies, see our Editorial Policies and the Editorial Policy Checklist.

Statistics

For all statistical analyses, confirm that the following items are present in the figure legend, table legend, main text, or Methods section.

n/a	Confirmed	
☐	☒	The exact sample size (n) for each experimental group/condition, given as a discrete number and unit of measurement
☒	☐	A statement on whether measurements were taken from distinct samples or whether the same sample was measured repeatedly
☒	☐	The statistical test(s) used AND whether they are one- or two-sided *Only common tests should be described solely by name; describe more complex techniques in the Methods section.*
☒	☐	A description of all covariates tested
☒	☐	A description of any assumptions or corrections, such as tests of normality and adjustment for multiple comparisons
☐	☒	A full description of the statistical parameters including central tendency (e.g. means) or other basic estimates (e.g. regression coefficient) AND variation (e.g. standard deviation) or associated estimates of uncertainty (e.g. confidence intervals)
☒	☐	For null hypothesis testing, the test statistic (e.g. F, t, r) with confidence intervals, effect sizes, degrees of freedom and P value noted *Give P values as exact values whenever suitable.*
☒	☐	For Bayesian analysis, information on the choice of priors and Markov chain Monte Carlo settings
☒	☐	For hierarchical and complex designs, identification of the appropriate level for tests and full reporting of outcomes
☒	☐	Estimates of effect sizes (e.g. Cohen's d, Pearson's r), indicating how they were calculated

Our web collection on statistics for biologists contains articles on many of the points above.

Software and code

Policy information about availability of computer code

Data collection	No software was used to collect data.
Data analysis	The code used in this study is freely available under an Apache 2.0 license at https://github.com/google-research/google-research/tree/master/dedal. All analysis was performed in Python 3.6, with the following libraries: absl-py 0.7, gin-config 0.4, numpy 1.18.4, tensorflow 2.3, tensorflow_datasets 3.0, tensorflow_probability 0.1, tf-models-official 2.6. We used HMMER v3.3.2 to run additional ablation studies shown in the Supplementary Information.

For manuscripts utilizing custom algorithms or software that are central to the research but not yet described in published literature, software must be made available to editors and reviewers. We strongly encourage code deposition in a community repository (e.g. GitHub). See the Nature Portfolio guidelines for submitting code & software for further information.

Data

Policy information about availability of data

All manuscripts must include a data availability statement. This statement should provide the following information, where applicable:

- Accession codes, unique identifiers, or web links for publicly available datasets
- A description of any restrictions on data availability
- For clinical datasets or third party data, please ensure that the statement adheres to our policy

The Uniref50 dataset is freely available under the Creative Commons Attribution (CC BY 4.0) License from https://www.uniprot.org . We used the 2018-03 release available at ftp.uniprot.org/pub/databases/uniprot/previous_major_releases/release-2018_03/uniref . Pfam is freely available under the Creative Commons Zero (``CC0'') licence from https://pfam.xfam.org . We used the Pfam-A seed dataset available at http://ftp.ebi.ac.uk/pub/databases/Pfam/releases/Pfam34.0/Pfam-A.seed.gz .

Human research participants

Policy information about studies involving human research participants and Sex and Gender in Research.

Reporting on sex and gender	n/a
Population characteristics	n/a
Recruitment	n/a
Ethics oversight	n/a

Note that full information on the approval of the study protocol must also be provided in the manuscript.

Field-specific reporting

Please select the one below that is the best fit for your research. If you are not sure, read the appropriate sections before making your selection.

☒ Life sciences ☐ Behavioural & social sciences ☐ Ecological, evolutionary & environmental sciences

For a reference copy of the document with all sections, see nature.com/documents/nr-reporting-summary-flat.pdf

Life sciences study design

All studies must disclose on these points even when the disclosure is negative.

Sample size	We analyze n=1,223,021 sequences from the Pfam-A seed database, which we split into training (80%), validation (5%) and test (15%) sets as explained in the manuscript. The sample size corresponds to >98% of the totality of Pfam-A seed database and was not otherwise optimized. It is sufficiently large to ensure that all performance measures we estimate are sufficiently accurate to draw statistically significant conclusions. For example, for the 'homology detection" tasks where we estimate AUROC of a binary classification problem to discriminate homologous pairs from non-homologous ones (Figure 4), the test set used to estimate AUROC has respectively >1 million and >5 million positive pairs, and the same amount of negative pairs; using the se_auc() method of R's auctestr package, we estimate that the standard error of the AUC estimate is <0.0005, which is more than two orders of magnitude smaller than the differences in AUROC between methods we study in Figure 4.
Data exclusions	We excluded from the analysis 24,151 sequences (about 2% of the total) with more than 512 amino acids, since the software used is optimized to process sequences of up to 512 amino acids.
Replication	Model training requires significant time and computation (several weeks on a large computer cluster), so we did not replicate the train/validation/test set split as explained above. The DEDAL model itself is trained by stochastic gradient optimization, so for a given training set the model obtained is stochastic. We replicated the training of DEDAL ten times with a different random seed to study the fluctuations of the performance of the model due to the stochasticity of the training, and plot the corresponding error bars in all figures.
Randomization	For the construction of the training/validation/test sets from the set of all sequences in Pfam-A seed, which are organized by clans, we first randomly selected 255 and 955 clans to form the out-of-distribution validation (2.5%) and test (7.5%) sets, respectively. From the sequences in the remaining 10,493 clans, we selected uniformly at random the in-distribution validation (2.5%) and test (7.5%) sets, respectively. The remaining sequences form the training set (80%).
Blinding	Blinding is not relevant in this study, since there is no group allocation to assess a treatment effect.

Reporting for specific materials, systems and methods

We require information from authors about some types of materials, experimental systems and methods used in many studies. Here, indicate whether each material, system or method listed is relevant to your study. If you are not sure if a list item applies to your research, read the appropriate section before selecting a response.

Materials & experimental systems

n/a	Involved in the study
☒	☐ Antibodies
☒	☐ Eukaryotic cell lines
☒	☐ Palaeontology and archaeology
☒	☐ Animals and other organisms
☒	☐ Clinical data
☒	☐ Dual use research of concern

Methods

n/a	Involved in the study
☒	☐ ChIP-seq
☒	☐ Flow cytometry
☒	☐ MRI-based neuroimaging