

Report for contest "Genetic Engineering Attribution Challenge"

I. INTRODUCTION

The task was to find TOP10 best lab matches for a given DNA-sequence. If the real lab of origin is in these TOP10 predictions then you got the points for this sample.

We have been provided with two types of data:

- DNA sequences;
- metadata about the plasmids.

Genetic engineering attribution is a very important problem nowadays and we proposed novel methods which solve it with very high accuracy.

II. DATA ANALYSIS

The metadata is valuable information but we did not find any insights there. The sequences are much more valuable and after analysis we found some interesting things.

Two DNAs generated in the same lab-of-origin often contain similar subsequences. These matched subsequences are the main features that can be extracted from DNA sequences. We decided to search for important subsequences instead of analyzing a sequence as a sentence. This insight helped us to follow the true way and build efficient architecture.

Some provided sequences are read 5'→3' direction but others are read 3'→5' direction. It was detected based on Blast output. This insight helped us to improve the quality of models.

III. MODEL DEVELOPMENT

3.1 Simple model architecture

Our main idea was to find popular subsequences which describe a lab-of-origin.

We tested 2 approaches:

- Use deep neural network (DNN) for automatic sequence analysis.

We build a DNN architecture which automatically extracts subsequences and classifies them. The architecture is presented in Figure 1.

- Approach based on Blast output.

If Blast finds a similar subsequence of any pair of DNAs we cut it out and add to the set of interesting subsequences. Then we build a matrix test + train size as first dimension and number of sequences as second dimension. It has size around 80K to 10^7 . This is too big for training. After filtering only popular subsequences and applying Principal Component Analysis (PCA) we extracted 1024 main components and successfully used them to build the lab-of-origin classifier.

There are issues in this approach. Blast doesn't find all matches in a sequence pair, it reports only the longest one. Analysis of 10^7 subsequences requires a lot of hardware resources so we have to reduce it significantly and so lose a lot of information. So while it's working approach we concentrated on the first one.

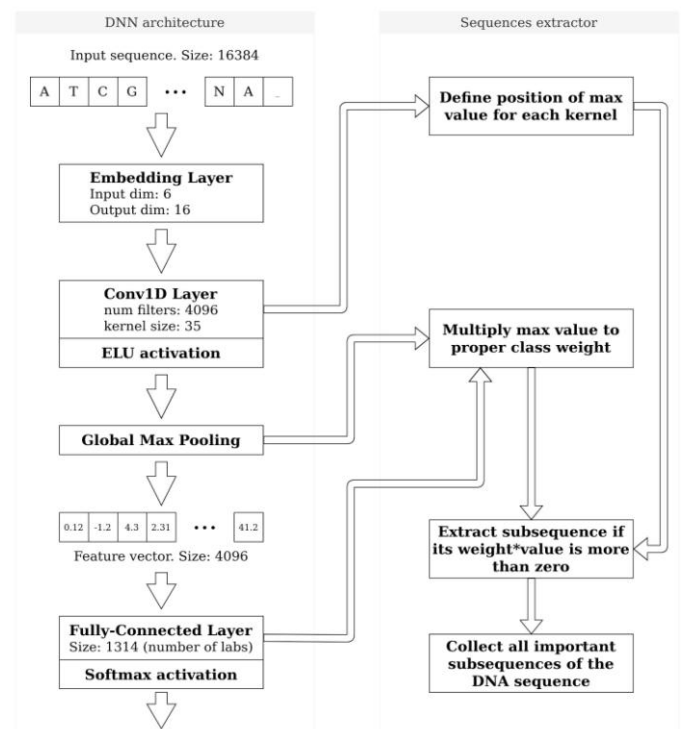


Figure 1. End-to-end DNN model which gets DNA sequence “as is” and returns the vector of lab-of-origin probabilities

On the left part of Figure 1 the DNN architecture is presented.

We cut the first 16384 symbols of the sequence. In case the sequence is shorter we pad it with zeros. Some sequences are reversed so we implement reverse augmentation for training. We take the last 16384, reverse them and replace A with T and C with G.

Firstly the model replaces each symbol of an input sequence to a vector of length 16 (Embedding layer). This means that the input sequence of length 16384 will be encoded as a 2D-matrix 16384×16 .

Then it multiplies a kernel 2D-matrix of size 16×35 to each position of the sequence matrix and results in a vector of $16384 - 35 + 1 = 16350$ values (Conv1D layer). There are 4096 kernels so we have a 2D-matrix 16350×4096 .

Each kernel (2D-matrix 16×35) from the set of 4096 kernels optimized to have such weights which after multiplication to sequence embeddings provide large values in position of important subsequences. There are 16350 different values for each kernel and we are interested only in the maximum. This maximum value shows us how likely the sequence contains some particular subsequence. So we are not interested in all 16350 values. Each kernel describes a subsequence and we only want to know that the subsequence is located in the given sequence. So we extract only max values that each kernel found using GlobalMaxPooling layer.

The last layer is classifier based on fully-connected layer, which returns most probable labs-of-origin.

On the right part of the image presented subsequences extractor.

To prove that a particular sequence is classified correctly it is necessary to look inside the black box of DNN. For interpretation of the model we look at the output of GlobalMaxPooling values. If a value is bigger than 0 then it is an important value for the class (for simplicity in the description we suppose that all classifier weights are positive). Then we make one more step back and look at the matrix 16350x4096. Find the position of max value that GlobalMaxPooling extracted for the particular kernel and define position (of length 35) in sequence that leads to positive values in the final vector. We developed a script that can define important parts of sequences based on a pretrained model and provide a set of subsequences which describe the lab-of-origin. In this script we consider that classifier weights can be negative.

Ensemble of several architectures presented on Figure1 without plastic metadata gives TOP10 validation accuracy ~0.92. With metadata about the plasmids it is ~0.95.

Pros:

- Simple and interpretable model
- No any data preprocessing
- Fast inference (100+ sequences per second on Gpu)
- Simple and fast fine tuning for new labs

Cons:

- Top 10 accuracy of the model is not perfect (~0.95 vs 0.97+ for the best model)

3.2. Large but more accurate model

Except the subsequences extraction we generated other features based on Blast output which were included to the final model.

3.2.1 Matrices of similarity between each DNA in train and test datasets.

The feature matrix is a square matrix where each row and column is sequence id and value on the intersection of sequence ids is the value that Blast provides for that pair. Blast provides a set of values for each found match: identity, alignment length, number of mismatches, gap opens, query start, query end, subject start, subject end, evalue and bit score. We found which values contain the most useful information and ended up with 4 different matrices based on log(alignment length), log(bitscore), mismatches and gap opens. The resulting matrices are too big for DNN input and we reduced them to 1024 main PCA components but it's still possible to use them in raw form in independent models.

3.2.2 Blast class similarity matrix

From the Blast results, you can also get a matrix of a special kind. The matrix represents the ranking of lab-of-origins for each sequence based on Blast bitscores. The matrix can be used as a preliminary answer to the problem, as

additional features for the neural network or as a part of the final ensemble.

Form of the matrix is (N x 1314), where N is the number of sequences (in the case of this problem, (63017 x 1314) for train and (18816 x 1314) for test). When constructing the matrix, the target is used - therefore, it must be built according to the Out of Fold (OOF) scheme with dividing the training data into folds, or use the Leave One Out scheme as an extreme case.

We take a specific sequence, look for all its matches with other sequences (which are replaced with lab-of origin) from the train and sort matched lab-of origin by bitscore. We look at the cls class of the first match, and put the value 1 in position (i, cls). We look at the next match - if the class is the same, then skip and go to the next match, if the class is new cls1, then we put the value $\frac{1}{2}$ in position (i, cls1), where 2 means the number of unique matching classes found so far. That is, for the next found class it will be 1/3, for the next 1/4, and for K class $1 / K$.

If you send the matrix constructed this way to the LeaderBoard, you can immediately get a score of ~0.88 on Public. We used the matrices both for training and for mixing into the final ensemble.

The final architecture presented on Figure 2. The sequence analyzer is not changed but we added additional input data and concatenated all intermediate layers before final classification. Additional inputs are:

- Initial plasmids metadata without any changes. It contains unique information about the lab-of-origin so significantly improves metrics.
- 4 PCA from similarity matrices. Each has the same 2 layers architecture and concatenation with categorical features. This is not unique data because the information extracted with blast theoretically can be extracted from raw sequences with Embedding layer + Conv1D.
- Blast class similarity matrix

Pros:

- Perfect top10 accuracy ~0.97 on validation.
- All data processed with a single model.

Cons:

- Requires complex preprocessing (launch Blast, build matrix, apply PCA)
- Difficult interpretation because of complex architecture

3.3. Top10 accuracy metric improvements

Our experiments helped to find peculiarities of post processing that improve top10 accuracy metric.

- Hard smoothing. Positive class = 0.7 instead of 1.0.
- Use logist layers instead of softmax layers for ensembling.
- Shift the prediction distribution to the test distribution.

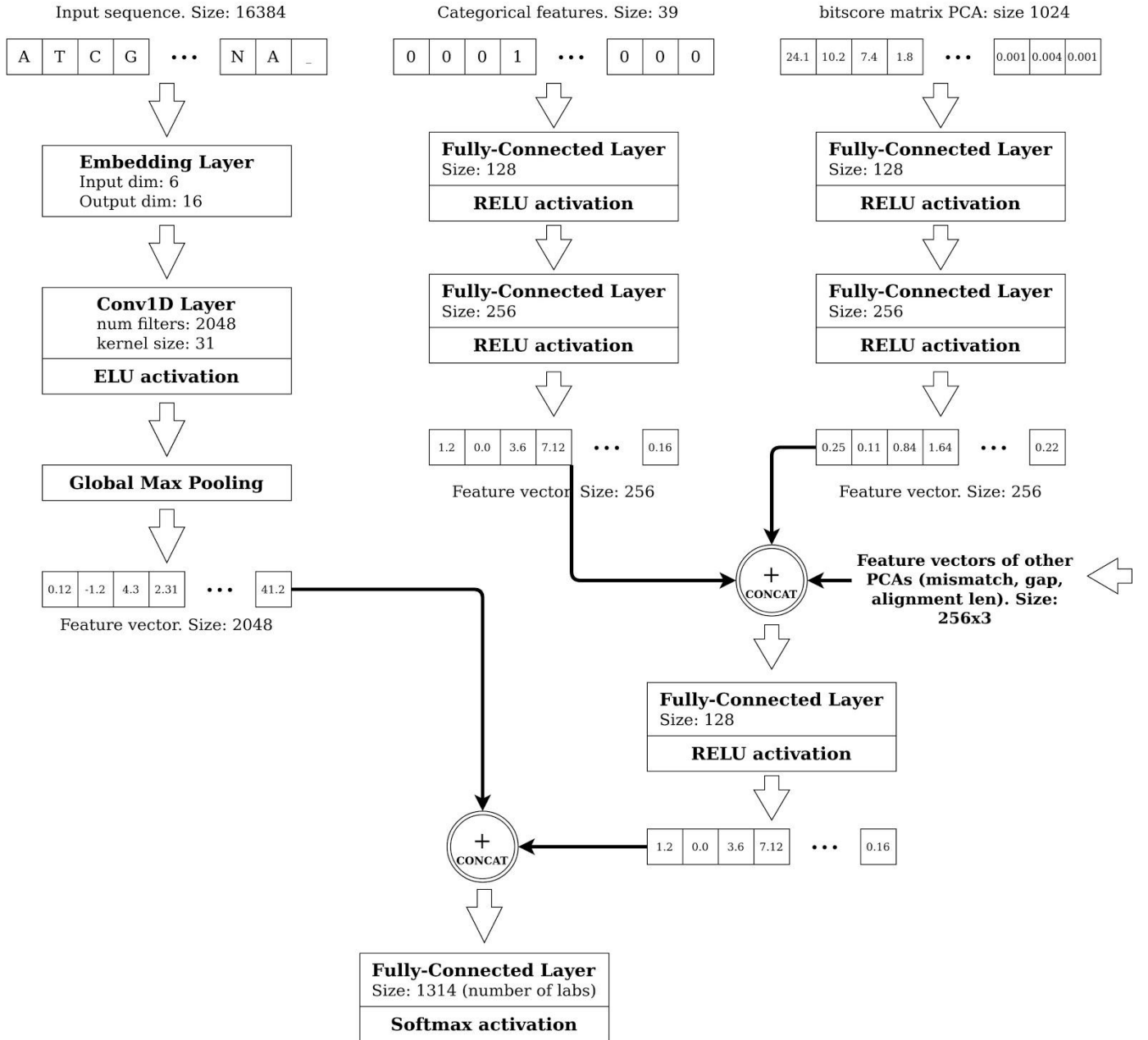


Figure 2. Structure of most precise DNN model

Interesting thing is that these approaches don't always improve quality in other problems and we believe that it works here because of the specific TOP10 metric. All these things together improve top10 accuracy on 0.01 while top1 accuracy almost not changed.

IV. MODEL ANALYSIS

4.1 Interpretation

If you train model without additional Blast features, e.g. on raw sequences then model is easy to interpret. We described it in section 3.1.

But there are more things we should analyze.

4.2 Uncertainty

Output of our model is a vector of probabilities. We do not use softmax because it often predicts values very close to 1 for most likely class and close to zero 0 for others. Instead we

use manually normalized $((x - \min) / \max)$ logits. We analyzed uncertainty based on scores. Most likely class always 1 because during normalization we divided by max. But values of second-ten positions scores show uncertainty of prediction. The smaller the value of the second position score the more confident the model that the first position class is correct. We tested this idea and removed 30% of data based top10 scores. This improves top10 accuracy from 0.97 to 0.99 on validation for left 70%.

4.3 Robustness

Our current model is a standard classification model, so there are some difficulties if new labs will appear. To add support for new labs you will need to retrain the model with changes in the last fully-connected layer. Good thing is that you still can start from the old weights and even froze weights on the first layers, e.g. training will be much faster. But after some large number of labs e.g. 50-100K it will be hard to classify them. So we can use a metric learning approach here. In metric learning each sequence will be associated with a

vector of fixed length. All vectors of one laboratory will lie close to each other in high dimension vector space (according to the Cosine Similarity metric). Then, by comparing the vector of the new sequence with all other vectors, it will be possible to select the cluster of vectors that lies close to the given vector. The PROs of this method that you can add new labs without retraining the actual model.

V. FUTURE DEVELOPMENT

Quality of DNN has been rapidly growing in different areas. The main breakthrough was in images and texts processing. In this work we developed DNN based on state of the art approaches in other areas. We believe that the progress will continue and cover other specific areas.

Our simple architecture has many advantages but it does not provide the quality that we can achieve after manual feature extraction. We believe that progress of DNN will reduce the gap of quality. Thus, we would not continue research in manual feature extraction and quality improvements but develop a framework based on the simple architecture and wait for improvements of base DNN training features (optimizers, loss functions, layers). So we don't need BLAST or other external programs for feature extraction anymore.

The idea of the framework is to make it simple to interpret results of trained models and simplify fine tuning for new sequences and labs. In parallel it makes sense to continue research of specific optimizations of the architecture. For example develop loss function which better optimizes interested metric (top10 accuracy).

Following this way we would have a framework which meets the requirements of various specialists. Quality will be automatically improved from time to time together with DNNs progress in general.