DREAM Challenge 2022

Predicting gene expression using millions of random promoter sequences

by autosome.org team

Dmitry Penzar*, Daria Nogina*, Georgy Meshcheryakov,
Andrey Lando, Arsenii Zinkevich, Ivan V. Kulakovskiy

* - equal contribution

Abstract

We present NoGiNet, a deep learning solution for the DREAM 2022 promoter expression challenge. Our network is based on EfficientNetV2 with residual concatenation instead of residual summation blocks. The One Cycle Policy is used for training with AdamW resulting in the so-called superconvergence of the model, with reduced training time and improved model performance.

Further improvement of the model convergence and validation performance was achieved by re-formulating the initial regression task as a soft-classification problem. Additionally, during training, we use an additional binary channel to explicitly mark the object with integer and thus likely imprecise expression measurements. The information from the second strand of each promoter is provided in a similar way, by augmenting the dataset with the reverse complementary sequences and then explicitly marking the orientation in a separate binary channel.

Our approach does not include any attention-based mechanics but reached high internal validation metrics and competitive performance on the public leaderboard. This agrees with the recently published studies showing that properly designed convolutional neural networks can outperform attention-based architectures in image analysis.

1. Description of data usage

1.1. Input vector structure

The training sequences were padded on the 5' end with nucleotides from the corresponding plasmid to the uniform total length of 150 base pairs and encoded into 4-dimensional vectors using **one-hot encoding**.

We have considered the sequences with integer scores as **singletons**, *i.e.* they likely have been observed only once, while non-integer scores were obtained by averaging two or more observations. To supply this information to the model, we introduced a binary `is_singleton` **channel** (1 for singletons, 0 for other training sequences). The final predictions for evaluation were made by specifying `is_singleton=0`.

Since the regulatory elements are often asymmetric relative to the transcription start sites, different scores were expected for direct and reverse complementary strands of a

particular sequence. Thus, the data was augmented by providing each sequence twice in native and reverse complementary form, specifying 0 and 1, respectively, in an additional `is_reverse` **channel**. The test-time augmentation was to average the predictions made for direct (`is_reverse=0`) and reverse complementary (`is_reverse=1`) input.

The resulting input of a neural network has the shape of 6x150.

1.2. The formal description of the challenge problem

We reformulated the problem of the challenge as a soft classification task by transforming expression scores to class probabilities.

1) Given a measured expression e, assume that the real expression is normally distributed: $\rho \sim N(\mu = e + 0.5,\ sd = 0.5)$.
2) For each class i from 1 to 16 defined by an original measurement bin, a probability of the class is the probability of $\rho \in [i,\ i + 1)$, where 0 and 17 bins are special cases with $\rho \in (-\inf, 1)$ and $\rho \in [17, +\inf)$, respectively.

Thus, the output of the model will be a vector containing 18 probabilities that correspond to each class (bin).

1.3 Train-validation split

To assess the model performance during internal evaluation, the training dataset was divided into 10 folds with mmh3 python library, where fold id for a sequence was calculated as MurMurHash3 hash sum of a sequence divided by 10, see the implementation in the Github repo. The last fold was used for internal validation during the leaderboard stage.

2. Description of the model

Our model (Fig. 1A) is based upon a fully-convolutional neural network architecture inspired by EfficientNetV2 [1] with selected features from DenseNet [2] and additional custom blocks. For all convolutions, we used `padding="same"`. For all convolutions followed by BatchNorm, we used `bias=False`.

The first block of our network is a standard convolution with `kernel_size=7`, followed by BatchNorm and SiLU activation (Fig. 1B). The output of the first block is passed to the sequence of six convolution blocks of EfficientNet-like structure (Fig. 1C) but using the grouped convolution instead of the depthwise of the original EfficientNetV2. Another performance-improving modification was to substitute the standard residual blocks with residual channel-wise concatenation.

The resize block is of the same structure as used at the start of the network (Fig 1B). The Squeeze and Excitation (SE) block is a modification of that of the original EfficientNetV2 (Fig. 1E). The bilinear block is re-implemented using `tensorly-pytorch` instead of using default PyTorch version, see the respective model code. The final block consists of a single convolutional layer with `kernel_size=1` followed by per-channel Global Average Pooling and SoftMax activation (Fig. 1D).

To obtain a predicted expression value for a sequence during the validation step, the predicted probabilities were multiplied by the corresponding bin numbers (Fig. 1D). This layer, if joined with softmax, is called **soft-argmax** [3]: $expression = \sum_{i=0}^{17} i \cdot p_i$.

We used 256 channels for the initial block and [128, 128, 64, 64, 64, 64] channels for EfficientNetV2-like blocks. The total number of parameters of our model is 1,852,846.

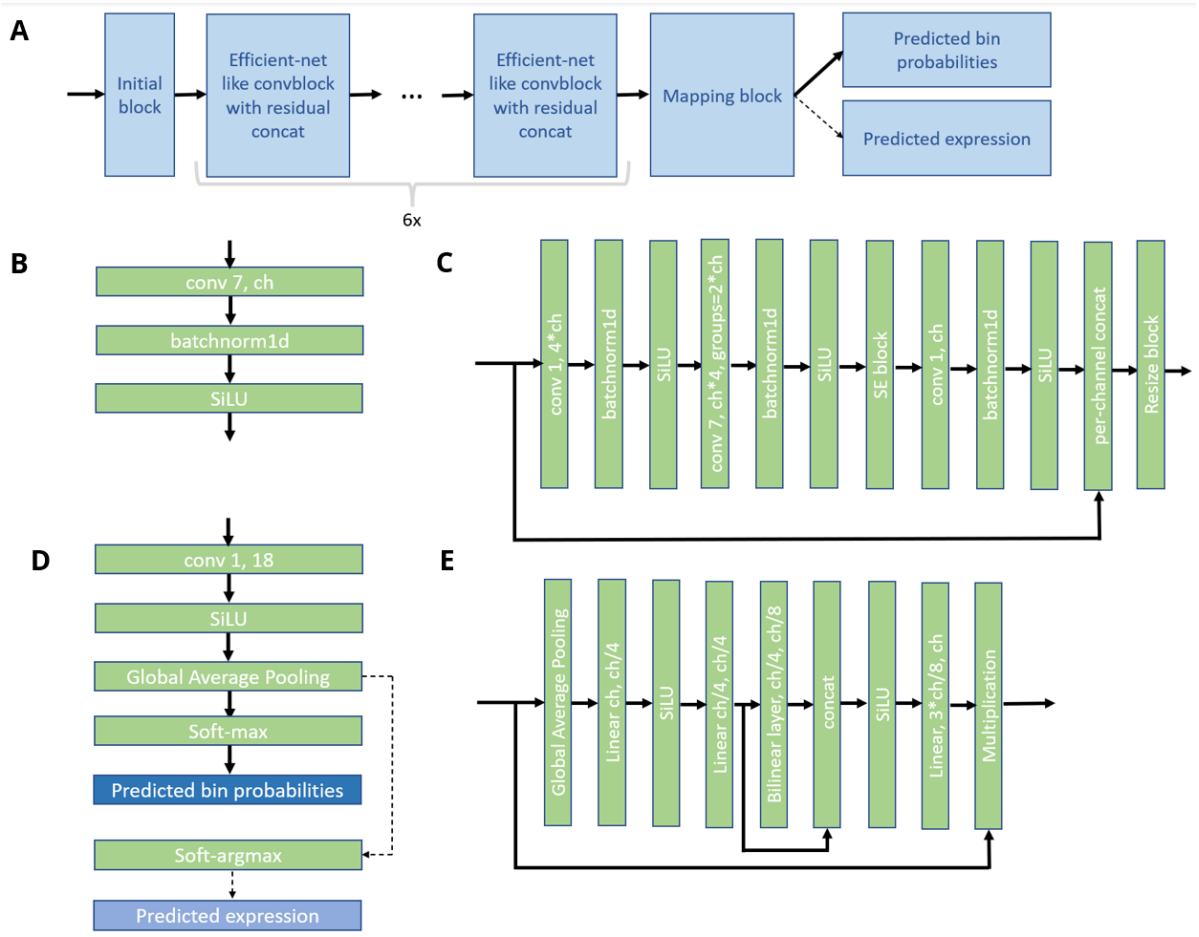

Figure 1. Architecture of NoGiNet. **A** - overall architecture. **B** - the structure of the initial and resize blocks. **C** - EfficientNetV2-like convolutional block. **D** - final layer of the network. Dashed lines denote the procedures used during the validation and evaluation only. **E** - SE-block structure.

3. Training procedure

To train our neural network, we used One Cycle Policy [4] with FastAI [5] modifications: two phases (instead of the original three), the cosine annealing strategy instead of the linear one, AdamW optimizer (`weight_decay=0.01`) instead of the SGD with momentum.

Each epoch consisted of 1000 batches of size 1024. The model was trained for 80 epochs as a reasonable trade-off between training time and validation variance. To select the max learning rate (0.005) for the One Cycle Policy, we used the LR-range test suggested in [6]. For the final model, we used the hyperparameters based upon the validation on the 10th fold, but the model was trained from scratch on the whole training dataset.

4. Other important features

We've used the same weight initialization as is used in EfficientNetV2. The training of the final model takes about 8 hours using the NVIDIA RTX A5000 and PyTorch version 1.11.0+cu113. For further technical details please check the project GitHub repository.

5. Contributors and Acknowledgments

5.1. Contributors

Name	Affiliation	Email
Dmitry Penzar*	Vavilov Institute of General Genetics, Russian Academy of Sciences, Moscow, Russia Faculty of Bioengineering and Bioinformatics, Lomonosov Moscow State University, Moscow, Russia	dmitrypenzar1996@gmail.com
Daria Nogina*	Faculty of Bioengineering and Bioinformatics, Lomonosov Moscow State University, Moscow, Russia	nogina_daria@mail.ru
Georgy Meshcheryakov	Institute of Protein Research, Russian Academy of Sciences, Pushchino, Russia	iam@georgy.top
Andrey Lando	Private researcher	dronte.l@gmail.com
Arsenii Zinkevich	Vavilov Institute of General Genetics, Russian Academy of Sciences, Moscow, Russia Faculty of Bioengineering and Bioinformatics, Lomonosov Moscow State University, Moscow, Russia	arsenii.z.work@gmail.com
Ivan Kulakovskiy	Institute of Protein Research, Russian Academy of Sciences, Pushchino, Russia Vavilov Institute of General Genetics, Russian Academy of Sciences, Moscow, Russia	ivan.kulakovskiy@gmail.com

* - equal contribution

5.2. Acknowledgments

We personally thank Ilya Dugay (Skolkovo Institute of Science and Technology, Center of Life Sciences) and Alexander Poslavsky (Faculty of Computational Mathematics and Cybernetics, Lomonosov Moscow State University) for valuable comments and suggestions.

This study was supported by the Russian Science Foundation [20–74-10075 to I.V.K.]

6. References

1. https://arxiv.org/abs/2104.00298
2. https://arxiv.org/abs/1608.06993
3. https://arxiv.org/abs/1710.02322
4. https://arxiv.org/abs/1708.07120
5. https://www.fast.ai/
6. https://arxiv.org/abs/1506.01186

7. Feedback

(1) Explicitly providing information on 'singletons' to the model allowed for significant improvement in the model performance both in internal evaluation and at the leaderboard. Hence, we believe that the raw data on expression bins or, at least, the explicit information on the number of sequence occurrences used to calculate the 'expression score' may improve the performance even further. Lack of this information in the challenge setup seems an essential limiting factor that may also impair the overall conclusion of the challenge in terms of selecting the genuinely best neural network architecture.
   (a) A related issue is the weighting procedure for averaging the measured expression intensity values. The distribution of non-integer expression scores found between neighboring integer values (singletons) in the training data is always skewed, suggesting there is a component of noise introduced in the training data.
   (b) The non-integer values were not rounded, which is leading to indirect data leakage. For instance, it is possible to deduce the relative bin weights by using the most frequent non-integer values in the dataset and, consequently, extract the frequencies of the bins that were observed for a given sequence. Although we refrained from using such leaked information in our model, a brute-force estimate suggests that this is possible for approximately 95% of the dataset.

(2) We consider the DREAM challenge as competition in both machine learning and data processing (compared to e.g. Kaggle). In this regard, we believe that the scope could be widened, especially as there is only one dataset under study. A separate discipline to utilize any external data and/or any complex solution (e.g. ensembles) would also be of interest, to estimate the gap between 'pure' and 'everything-allowed' solutions.

(3) Overall, we would like to thank the competition organizers for generating truly valuable and machine-friendly data that will certainly be further useful outside the scope of the contest. We were truly inspired by the friendly atmosphere, amicable community management, and careful and lightning-fast feedback. Ultimately, it was a real pleasure to participate in the challenge and we truly hope to be able to contribute to the manuscript preparation and, in the years to come, participate in the next DREAM event or another contest organized by the same team.