# GENERATIVE ADVERSARIAL NETWORKS WITH FEEDBACK MECHANISM FOR NOVEL PROTEIN SEQUENCES

*Lada Nuzhna, Karina Zadorozhny*

Northwestern University

## ABSTRACT

Generative Adversarial Networks (GANs) can learn patterns from real samples and harnesses this information to create novel data. GANs have recently been applied to *in silico* design of protein and DNA sequence. This work extends a method that not only produces realistic sequences but allows for feature enrichment - Feedback GAN. Using this approach, we designed novel proteins optimized to contain multiple 3D structural properties.

***Index Terms***— GAN, Generative Models, Language GAN, Sequences, Protein Sequence

## 1. BACKGROUND

Designing novel protein structures is important both for protein-based drug development and genetic engineering. There has been an effort to generate protein and DNA sequences using generative neural networks [3, 1]. In this work, we extend the feedback-mechanism approach described in [1]. The generative model uses a separate network that grades generated sequences and passes the top-performing sequences to the discriminator as the real ones. This project extends from the previous effort to generate samples with an $\alpha$-helix (H) to optimization of more rare features and multiple features at once.

## 2. IMPLEMENTATION

### 2.1. Overview

We have implemented a Feedback-GAN architecture that consists of three main parts: a generator, a discriminator, and a feedback network. First, GAN is pre-trained to produce realistic-looking DNA sequences. Separately, Feedback Net is trained to recognize the presence of the structural properties and acts as a multi-label classifier. Overall, protein structure can be characterized by eight different states which correspond to the different 3D orientation of a protein: helix (G), $\alpha$-helix (H), $\pi$-helix (I), $\beta$-stand (E), bridge (B), turn (T), bend (S), and coil (C). After pre-training, batches of sequences produced by the generator are passed to the Feedback Net and receive eight scores that correspond to the probability of each of the eight structures being present in a sequence. If, for example, the desired features are E, H, C, then the generator's outputs that pass a threshold for E, H, C are getting added to the discriminator's input as the real sequences. The generator thus slowly learns the patterns of the top-performing results. Note that the sequences pass as exemplary only if scores for *every* desired feature is above the threshold.

### 2.2. Feedback Network

Formally, the task of predicting these eight structures from sequences is called the Q8-problem. Given its complexity, we omit the exact positioning of the 3D property for the purposes of this experiment and focus just on whether it appears in any given sequence or no.

Given the sequential nature of the protein data, we implemented an architecture that combines an embedding layer and bidirectional LSTMs. The network was optimized as a multi-label classifier. More specifically, the last layer of the network consists of eight nodes, each with a sigmoid activation. The network also utilizes the binary cross-entropy loss for its learning. Thus, Feedback Net outputs the probability of 8 labels being present in the input.

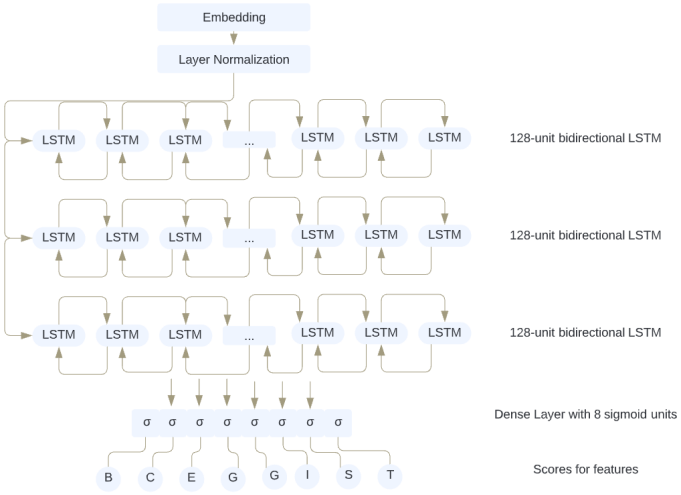


**Fig. 1**. The Feedback Net architecture.

### 2.3. GAN

The GAN architecture corresponds to the language WGAN with a gradient penalty described in [1, 2]. The generator and the discriminator consist of five residual blocks that perform the 1D convolution operation and up- or down- sample the inputs, respectively. The generator $G$ takes in a latent seed vector $z \in \mathbf{R}^D$. Its output is a matrix of logits $A = G(z) \in R^{L \times M}$ where $L$ is the length of a sequence and $M$ is the number of encoding letters. The matrix A represents probabilities of the position $l_i$ corresponding to each letter. This softmax output is passed directly to the discriminator. For the final output, the letter with the maximum probability for each position is chosen (argmax) and the output is an $L$-dimensional vector.

## 3. DATASET ANALYSIS

The dataset of protein sequences used for both Feedback and GAN pre-training was obtained from an open-source Protein Secondary Structure Dataset from Kaggle. Dataset contains 95,915 samples of varying lengths with up to 800 amino acids in a sequence. We have used sequences with up to 75 amino acids for GAN pre-training (over 24,000 training samples) and with up to 128 amino acids for Feedback Net pre-training (88,751 training samples).

In the dataset, there are rare and common features. The frequency distribution of features in the dataset corresponds to the natural distribution in proteins. These numbers were taken into account during Feedback Net training and during feature optimization by fbGAN.

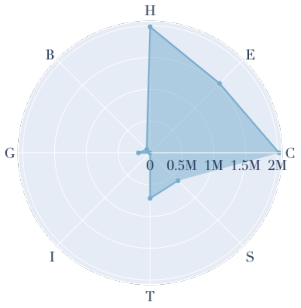


**Fig. 2**. Distribution of features in the dataset (note that sequence may contain more than one feature)

## 4. TRAINING

### 4.1. Feedback Net Training

We started by implementing a network with a single-label task (picks dominant feature as a label for sequence). However, this approach appeared to be non-suitable for our problem, as we later learned that the dataset is dominated by 3 labels (Fig 2).

Fig. 3 shows the accuracy, precision, recall, f-1 score, and support metrics for each class. Note that the low recall score for the I class corresponds to its representation in the training set (less that $0.3\%$). To our knowledge, Q8 problems focus on locating the features within sequences and thus cannot be directly compared with our binary classification network.

| feature | precision | recall | f-1 score | support |
|---|---|---|---|---|
| C | 1.00 | 1.00 | 1.00 | 26626 |
| E | 0.95 | 0.96 | 0.96 | 16950 |
| G | 0.88 | 0.71 | 0.78 | 11741 |
| H | 0.99 | 0.94 | 0.96 | 20003 |
| I | 0.82 | 0.46 | 0.59 | 71 |
| S | 0.98 | 0.96 | 0.97 | 23403 |
| T | 0.97 | 0.96 | 0.97 | 23488 |
| B | 0.88 | 0.69 | 0.77 | 10170 |

**Fig. 3**. Multi-label classification summary

### 4.2. GAN Training

We first implemented GAN for learning 21-characters (20 amino acids and one padding symbol). However, this approach did not lead to loss convergence and the produced sequences repeated only one or two character at random. To overcome this problem, we used the DNA encoding of protein sequences which is essentially a one-to-one mapping and allows for conversion without a substantial loss of information. Each amino acid is encoded by three DNA letters. We have trained the model to produce DNA sequences of length 225 (75 amino acids). The sequences were prep-ended and appended with start and end codons (ATG and TAG).

We have confirmed that the validity of generated sequences by comparing their biochemical properties to the real sequences (Fig 5). Given the GAN's susceptibility to mode collapse [4], we investigated the similarity of the generated sequences using a simple Levenshtein (edit) distance metric (Fig 6).

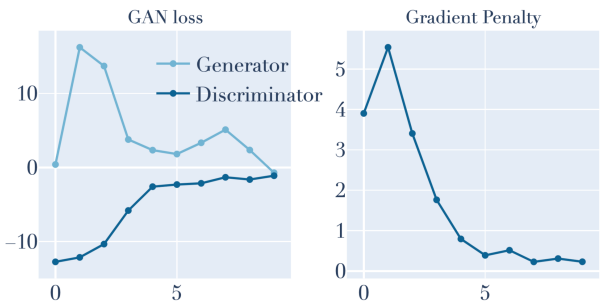


**Fig. 4**. GAN pre-training. The loss and gradient penalty curves over epochs. Hyperparameters: epochs = 10, gradient penalty = 5, learning rate = 0.002.

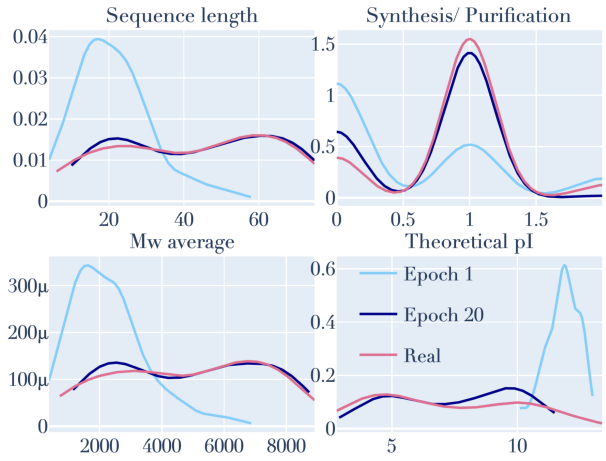


**Fig. 5**. GAN pre-training. Biochemical properties of sequences generated by the generator at the beginning (epoch 1) as compared to the end of training (epoch 20).

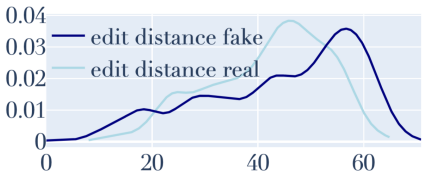


**Fig. 6**. Edit (Levenshtein) distance of the real sequences (100 samples) and the distance of the generated sequences (100 samples).

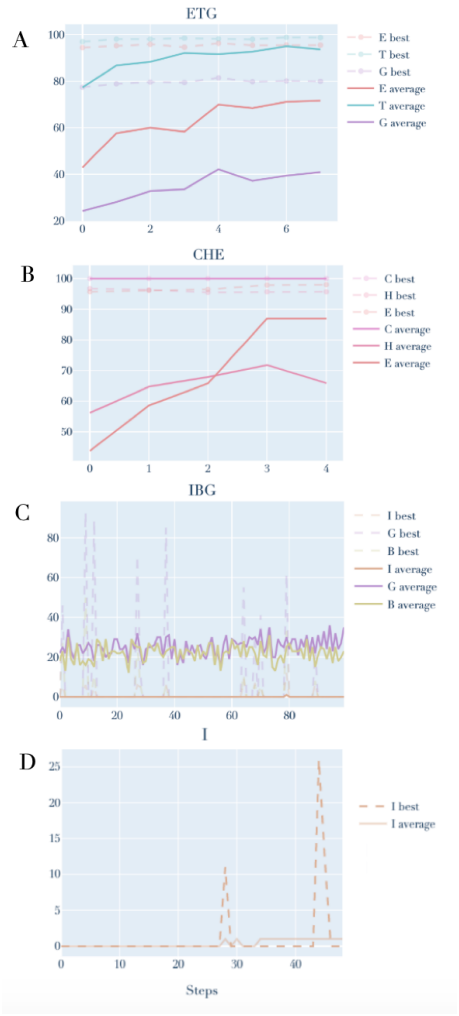


**Fig. 7**. Scores history during fbGAN training. Scores over steps plotted. Best scores indicate the scores of the samples at passed a threshold during training.

## 5. RESULTS

To test the approach, we ran two types of experiments: optimizing for single features and for a combination of different features at once. The network was able to separately enrich for the common features by up to 30-40% (see external link). However, for very rare features such as I ($\pi$-helix), only a very small enrichment was observed, after lowering the threshold (the score generated sequences needed to pass to be passed to the discriminator) to 0.1 and substantially extending the training time (Fig. 7D).

Next, we tested how the network performs optimization for combination of features. We optimized for three common features (C,H,E), 1 common and 2 rare (E,T,G), and 3 rare (B,G,I). Interestingly, the feature G achieved a 20% improvement in combination with E,T (Fig. 7A) but not B,G (Fig. 7C) which suggests that certain structures are not easily combined together in one sequence. This is in an agreement with their natural conditional distribution. (see supplemental data).

## 6. FUTURE WORK

Given recent advancements with AlphaFold2, incorporating protein distance maps into feedback mechanism might lead to a more precise optimization of features. This would allow us to monitor the novelty and similarity of the generated sequences more precisely, by taking the relative location of the features into consideration. Introducing similarity penalty during training might also force the generator to produce more diverse sequences.

For rare features optimization, we plan to incorporate a dynamic, gradually increasing threshold, allowing the Discriminator to have more sequences to learn from at the very beginning and becoming more ”selective” once the scores improve.

# References


[1] James Zou Anvita Gupta. “Feedback GAN for DNA optimizes protein functions”. In: *Nature Machine Intelligence* (2019). DOI: https://doi.org/10.1038/s42256-019-0017-4.

[2] Ishaan Gulrajani et al. “Improved Training of Wasserstein GANs”. In: *CoRR* abs/1704.00028 (2017). arXiv: 1704.00028. URL: http://arxiv.org/abs/1704.00028.

[3] Andrew Delong David Duvenaud Brendan J. Frey Nathan Killoran Leo J. Lee. “Generating and designing DNA with deep generative models”. In: (2017).

[4] Akash Srivastava et al. “VEEGAN: Reducing Mode Collapse in GANs using Implicit Variational Learning”. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017, pp. 3308–3318. URL: https://proceedings.neurips.cc/paper/2017/file/44a2e0804995faf8d2e3b084a1e2db1d-Paper.pdf.