

III. Model documentation and write-up

Information included in this section may be shared publicly with challenge results. You can respond to these questions in an e-mail or as an attached file. Please number your responses.

1. Who are you (mini-bio) and what do you do professionally?

If you are on a team, please complete this block for each member of the team.

Amalgam

Amalgam is one of the few companies in the market today that is capable of offering a complete suite of solutions for all Artificial Intelligence needs from Server, to Data Center, to Data Engineering, to Data Science, to Machine Learning, and beyond. Our goal is to make Artificial Intelligence easy, cheap, and accessible to companies of all sizes and industries.

Our solutions offer your business the edge required to harness the power of AI to disrupt your industry at a cost effective manner. Your industry is waiting to be disrupted by AI and with Amalgam you can achieve that in a cost-effective manner. Visit us at <http://www.amalgam.ai> to learn more about how Amalgam can help your business harvest the full power of Artificial Intelligence.

Adriano Marques

Adriano was the engineering team lead at crowdSPRING and is the founder of the Umit Project Open Source Organization where he had, for 6 years, mentored dozens of engineers around the world through the Google Summer of Code program. He is the former co-founder and CTO at Startup Foundry, and currently the CEO of Exponential Ventures, a dream company aimed at creating and launching new Startups and Senior Data Scientist at Loevy & Loevy.

A dedicated husband and daddy to two boys and two girls, Adriano still finds time to cook, watch movies, send a camera to the edge of space and record a major storm forming from above, work as a volunteer and read (or listen to) books while he's not sleeping or working.

Adriano graduated from the State University of Goias, in Brazil, where he earned a Bachelor degree, while traveling about 55,900 miles (more than earth's equatorial circumference). In his quest to travel more than Gerard Moss (the first South American to fly around the world), Adriano graduated, married, and traveled with his wife to live in Brussels and work for Belgacom, where he learned a bit of French and to appreciate the original French fries which are actually Belgian. Since 2012 he's been happily living in the United States with his Family, and in 2016 graduated as a M.Sc. in Artificial Intelligence from Georgia Tech.

Fernando Camargo

Fernando Camargo is a creative, data-driven, and well-rounded Machine Learning Engineer with in-depth machine learning expertise and a wide-ranging 10+ years of experience in software engineering. He is currently a Data Scientist at Exponential Ventures, Machine Learning Researcher at Deep Learning Brazil (Federal University of Goiás), and Co-Founder at DataLIFE.ai. He holds a Masters Degree and is currently pursuing his Ph.D. at Federal University of Goiás.

Igor Muniz

Igor Muniz is a Data Scientist specialized in Machine Learning Engineering. Passionate about puzzles and a competitive person, he has a hobby competing in data science challenges while learning new things every day. He is currently working as Data Scientist at Exponential Ventures, teaching deep learning class and playing chess in his free time.

2. What motivated you to compete in this challenge?

Everybody in this world has a passion that keeps them moving forward. Our passion is solving the most difficult problems using Artificial Intelligence that can benefit humanity in significant ways. This challenge checked all the boxes. We're very excited to have placed 10th in the leaderboard and 2nd in the innovation track.

3. High level summary of your approach: what did you do and why?

Instead of doing a common multi-class classification with a softmax layer, we used Triplet Networks in order to learn embeddings for the labs and learn an embedding extractor for DNA sequences. This way, we can use distance metrics to rank the labs-of-origin while having the embeddings as a useful sub-product.

4. Copy and paste the 3 most impactful parts of your code and explain what each does and how it helped your model.

```
def forward(
    self, sequences: torch.Tensor, extra_inputs: torch.Tensor, labs: torch.Tensor
) -> Tuple[torch.Tensor, torch.Tensor, torch.Tensor]:
    batch_size = sequences.size(0)

    anchors = self.normalize(
        self.extract_sequence_embedding(sequences, extra_inputs)
    ) # (B, E)
    positives = self.normalize(self.lab_embedding(labs)) # (B, E)

    all_labs = torch.arange(
        0, self.num_labs, dtype=torch.long, device=self.device
    ).repeat(
        batch_size, 1
    ) # (B, L)
    negative_labs_mask = torch.ones(
        batch_size, self.num_labs, device=self.device
    ).bool()
    negative_labs_mask[
        torch.arange(0, batch_size, device=self.device), labs
    ] = False
    all_negative_labs = all_labs[negative_labs_mask].reshape(
        batch_size, self.num_labs - 1
    )
    all_negative_labs_embedding = self.normalize(
        self.lab_embedding(all_negative_labs), dim=2
    ) # (B, L-1, E)

    similarity_between_anchor_and_negatives = (
        anchors.reshape(batch_size, 1, self.lab_embedding.embedding_dim)
        * all_negative_labs_embedding
    ).sum(
        2
    ) # (B, L-1)

    hardest_negative_labs = torch.argmax(
        similarity_between_anchor_and_negatives, dim=1
    ) # (B,)

    negatives = all_negative_labs_embedding[
        torch.arange(0, batch_size, device=self.device), hardest_negative_labs
    ]

    anchors, positives, negatives = self.embeddings_dropout(
        anchors, positives, negatives
    )
    anchors = self.normalize(anchors)
    positives = self.normalize(positives)
    negatives = self.normalize(negatives)
    return anchors, positives, negatives
```

This code is the most important part of our model. Basically, our Dataset gives us the DNA sequences, some extra inputs and the lab-of-origin of each of these sequences. We use the

DNA sequences and extra inputs to extract an embedding for each sequence and we have an embedding layer for the labs. The labs we got from the Dataset is our positive example and we do hard negative mining to generate the negative examples. In this case, the negative example is a lab that is the closest to the DNA sequence and is not the lab-of-origin. This code does this hard negative mining online in a vectorized way that can run inside the GPU, making it much faster.

```
def predict_lab_scores(
    self, sequences: torch.Tensor, extra_inputs: torch.Tensor
) -> torch.Tensor:
    sequence_embedding = self.normalize(
        self.extract_sequence_embedding(sequences, extra_inputs)
    ) # (B, E)

    all_labs_embedding = self.normalize(self.lab_embedding.weight) # (N, E)

    x = sequence_embedding @ all_labs_embedding.T # (B, N)
    x = torch.softmax(x, dim=1)

    return x
```

This code generates the scores for each lab using a dot product between the extracted embeddings of the DNA sequences and the embeddings of all labs. It then passes it through a softmax.

```
def _preprocess_sequence(self, sequence: np.ndarray) -> torch.Tensor:
    return torch.tensor(
        self._get_random_piece(
            self.transform_sequence_fn(sequence), self.piece_size
        ),
        dtype=torch.int64,
    )
```

We used random roll (rolling the sequence circularly) and cutting it to a fixed size before passing it through our model. So, the model learned to deal with different pieces of the same DNA sequence.

5. Please provide the machine specs and time you used to run your model.
- CPU (model): AMD EPYC 7401, i7-6950X, i7-5960X
 - GPU (model or N/A): GTX 1080, GTX 1080 Ti, GTX 1070, RTX 2080 Ti, Titan V
 - Memory (GB): All nodes had between 64Gb and 128Gb of RAM
 - OS: Ubuntu Linux Server 18.04 LTS
 - Train duration: 1 day, 4:13:39.354372 (4 folds)
 - Inference duration: ~5 minutes

6. What are some other things you tried that didn't necessarily make it into the final workflow (quick overview)?

We tried to use PlasmidHawk (<https://gitlab.com/treangenlab/plasmidhawk>) as a feature engineer but it didn't work as we expected.

7. Did you use any tools for data preparation or exploratory data analysis that aren't listed in your code submission?

No

8. How did you evaluate performance of the model other than the provided metric, if at all?

1 – Accuracy

2 – Top 10 Accuracy

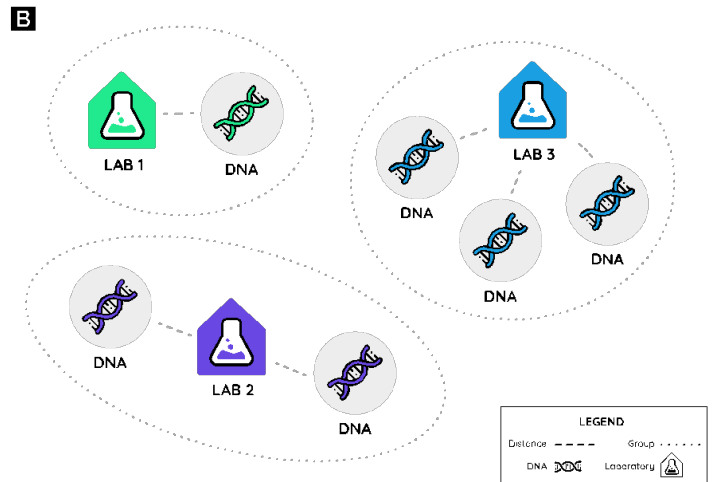
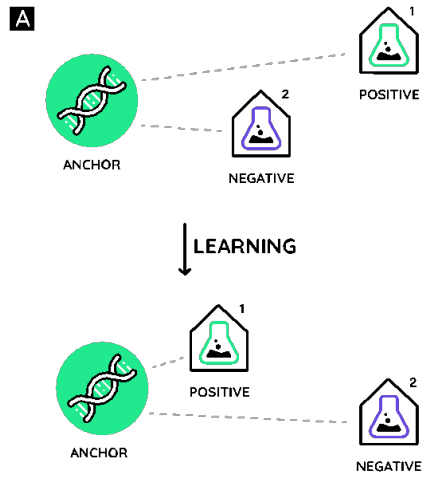
3 – Triplet Accuracy: The number of times in which the lab-of-origin was the most similar to the DNA sequence divided by the number of examples.

9. Anything we should watch out for or be aware of in using your model (e.g. code quirks, memory requirements, numerical stability issues, etc.)?

You may notice that instead of using the 'train_values.csv' dataset, we used the 'train_values_grouped.csv' dataset instead. It is just a preprocessed version of the dataset in which we grouped sequences of each lab by their Levenshtein distance and the code to generate it is in 'utils/group_sequences.py'.

In order to run this script you'll need at least 64GB RAM and due to the long run time you may want to split the processing by running the same script for different labs in parallel.

10. Do you have any useful charts, graphs, or visualizations from the process?



11. If you were to continue working on this problem for the next year, what methods or techniques might you try in order to build on your work so far? Are there other fields or features you felt would have been very helpful to have?

We would improve the part of the model that extracts features to generate the embeddings for the DNA sequences.