Max Lifespan Research Log

Wyatt McCarthy

Weeks of: September 16th - September 30th

Focus of this period was to get re-situated with the project and caught up on the work that was done over the summer. For a quick recap, when I finished my research in the spring, we had just gotten to modeling, following a lengthy EDA period where we identified potentially “good” and “bad” data. For more info, see report (link to PDF).

Quick EDA Recap:

```
* 51 million DNA sequences orthologous to the human genome from 453 different mammalian genomes
* Added ‘lifespan‘ variable to data (according to the species which a sequence was from)
* Organized data into sets categorized by the gene from which a sequence was extracted
* For each gene:
    * used DNABERT-S embedding model to embed sequences (embeddings of dim 1 x 768)
    * reduced embedding dimensionality to 1 x 3 using PCA
    * k-means clustered reduced embeddings
        * idea here is that clustered embeddings represent similar DNA sequences
    * computed  lifespan statistics (mean, median, std deviation, z score) for each
    cluster to glean whether there is association between DNA similarity and lifespan in any
    clusters
        * if there is no association between DNA similarity and lifespan, data is definitely
        bad; if there is association, data has potential
* Glimpse of the data we’ve collected:
```

gene type	max seq len	min seq len	mean seq len	median seq len	wmc similarity score	species represented
E2F2	1374	1308	1327.752	1323	0.240	418
ARID4B	4131	3669	3819.180	3900	0.237	418
NFKBIE	1626	1083	1295.019	1101	0.139	415
MAZ	1557	1350	1432.163	1437	0.199	334
PID1	921	648	718.523	744	0.153	436

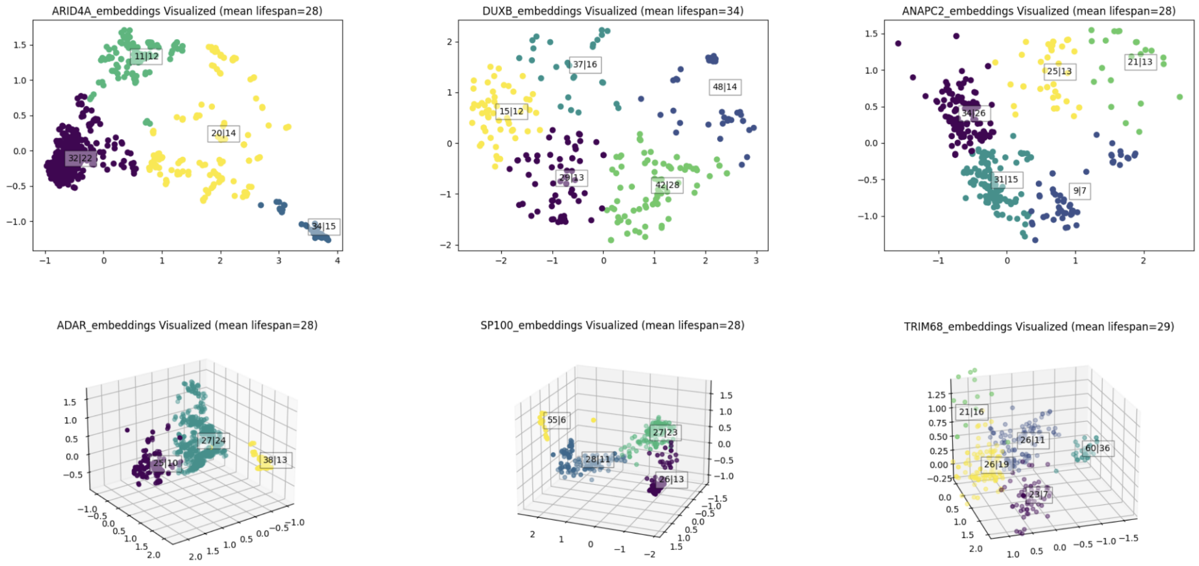
gene type	set mean lifespan	set std dev	set iqr bounds	cluster	cluster mean lifespan	cluster std dev	cluster iqr	cluster iqr bounds	#points in cluster	#species in cluster	modified z-score	statistical significance
PFN1	27.418	21.138	15-35	cluster0	23.8	14	17.55	14-32	315	181	-0.06	0.474
PFN1	27.418	21.138	15-35	cluster1	46.1	37	41.00	20-60	59	42	1.16	0.124
PFN1	27.418	21.138	15-35	cluster2	30.8	11	5.20	27-32	8	6	0.47	0.319
ZNF804A	27.600	20.872	15-36	cluster0	28.3	19	15.95	15-31	51	45	0.19	0.425
ZNF804A	27.600	20.872	15-36	cluster1	32.5	20	20.35	20-40	163	104	0.46	0.324

gene	avg cluster stat sig	most stat sig cluster	#species in most sig cluster	least stat sig cluster	avg # species per cluster
PFN1	0.306	0.124	42	0.474	76.333

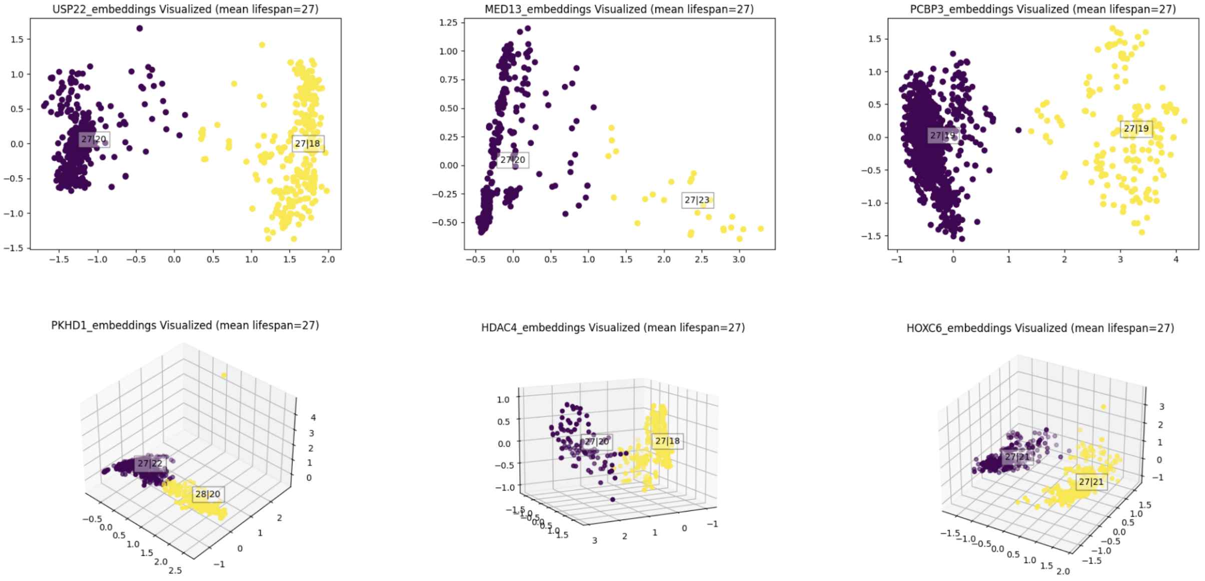
gene	avg cluster stat sig	most stat sig cluster	#species in most sig cluster	least stat sig cluster	avg # species per cluster
ZNF804A	0.393	0.324	104	0.433	64.750
IRAK3	0.442	0.340	61	0.494	68.500
INO80B	0.419	0.307	23	0.477	83.333
SIVA1	0.398	0.290	90	0.482	72.250

```
* And some examples of clustering results (and how they vary according to z-score)
```

Embedding visualizations on sets for which there are high z-scoring clusters


ARID4A_embeddings Visualized (mean lifespan=28)
DUXB_embeddings Visualized (mean lifespan=34)
ANAPC2_embeddings Visualized (mean lifespan=28)
ADAR_embeddings Visualized (mean lifespan=28)
SP100_embeddings Visualized (mean lifespan=28)
TRIM68_embeddings Visualized (mean lifespan=29)

Embedding visualizations on sets for which there are low z-scoring clusters

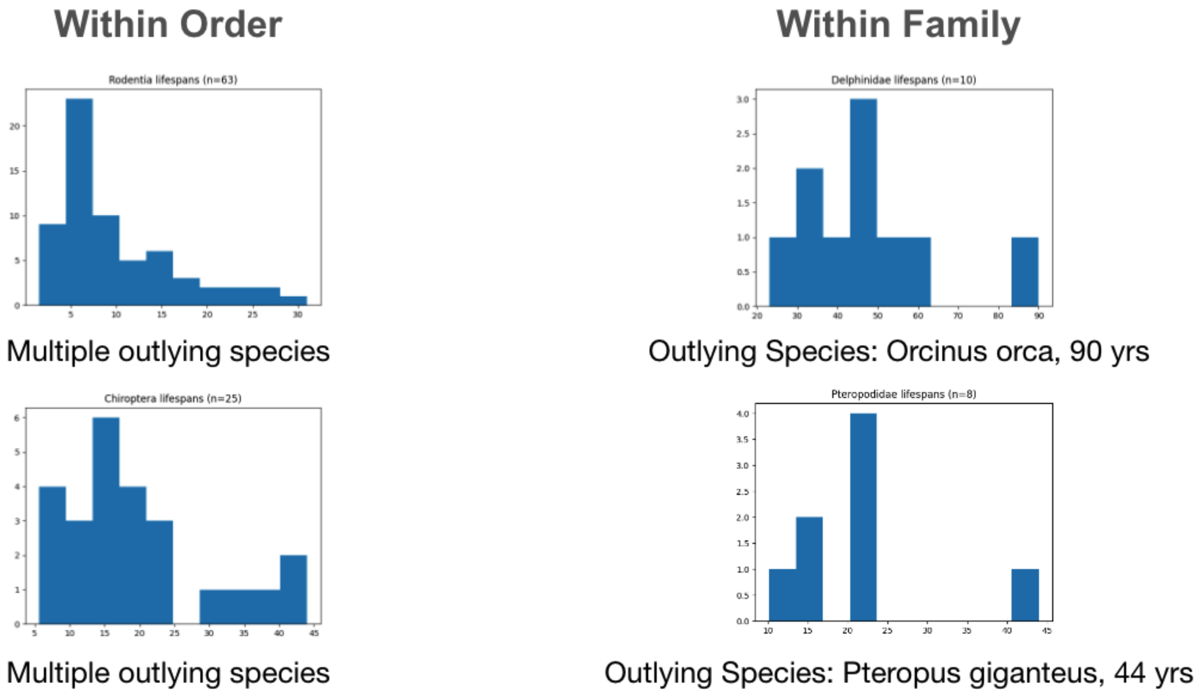

USP22_embeddings Visualized (mean lifespan=27)
MED13_embeddings Visualized (mean lifespan=27)
PCBP3_embeddings Visualized (mean lifespan=27)
PKHD1_embeddings Visualized (mean lifespan=27)
HDAC4_embeddings Visualized (mean lifespan=27)
HOXC6_embeddings Visualized (mean lifespan=27)

Quick Modeling Recap: * The main purpose of the EDA described above was to identify and compile training data; the idea from there was that if we could train a model to accurately predict lifespan when given a DNA sequence from an “outlying” species, we were on to something * in this context, by “outlying” species we mean a species whose lifespan is uncharacteristically large relative to genetically similar species * to identify such species, we performed further analysis, briefly shown in the tables and plots below

family	median lifespan	IQR	outlying species	outlying max lifespan
Procyonidae	24.0	5.40	Potos_flavus	38.4
Lemuridae	36.2	1.80	Prolemur_simus	17.6
Hominidae	59.0	1.27	Pan_paniscus	55.0
Cervidae	22.0	4.70	Cervus_elaphus	31.5
Cervidae	22.0	4.70	Muntiacus_crinifrons	11.0

genus	median lifespan	IQR	outlying species	outlying max lifespan
Microtus	4.80	0.83	Microtus_oeconomus	2
Macaca	38.05	2.92	Macaca_mulatta	30
Pteropus	20.60	7.27	Pteropus_giganteus	44

order	median lifespan	IQR	outlying species	outlying max lifespan
Rodentia	7.3	8.00	Hystrix_cristata	28.0
Rodentia	7.3	8.00	Heterocephalus_glaber	31.0
Rodentia	7.3	8.00	Coendou_prehensilis	26.6
Ruminantia	22.0	8.60	Giraffa_camelopardalis	39.5
Whippomorpha	49.5	38.67	Balaena_mysticetus	211.0



Multiple outlying species

Outlying Species: Orcinus orca, 90 yrs

Multiple outlying species

Outlying Species: Pteropus giganteus, 44 yrs

Figure 1: Species Lifespan Histograms

```
* With the context gleaned from this EDA, we experimented with various modeling approaches:
  * The Perceiver Model
    * unsuccessful, model did not appear to learn well regardless of the configuration of
    training data tested; some results shown below
```

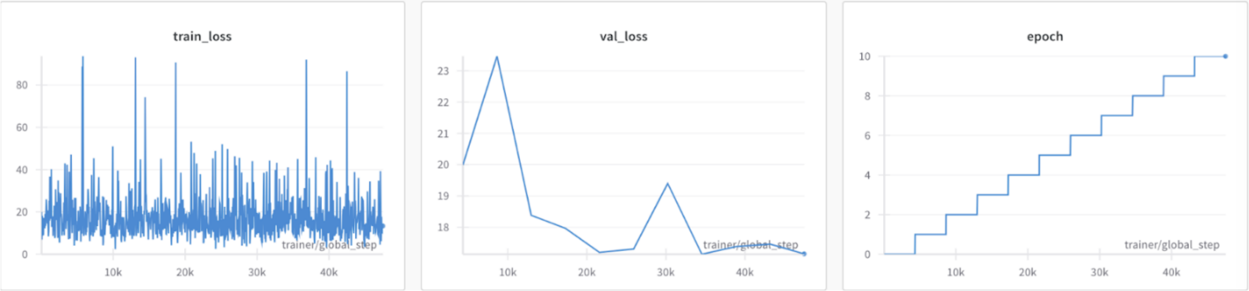
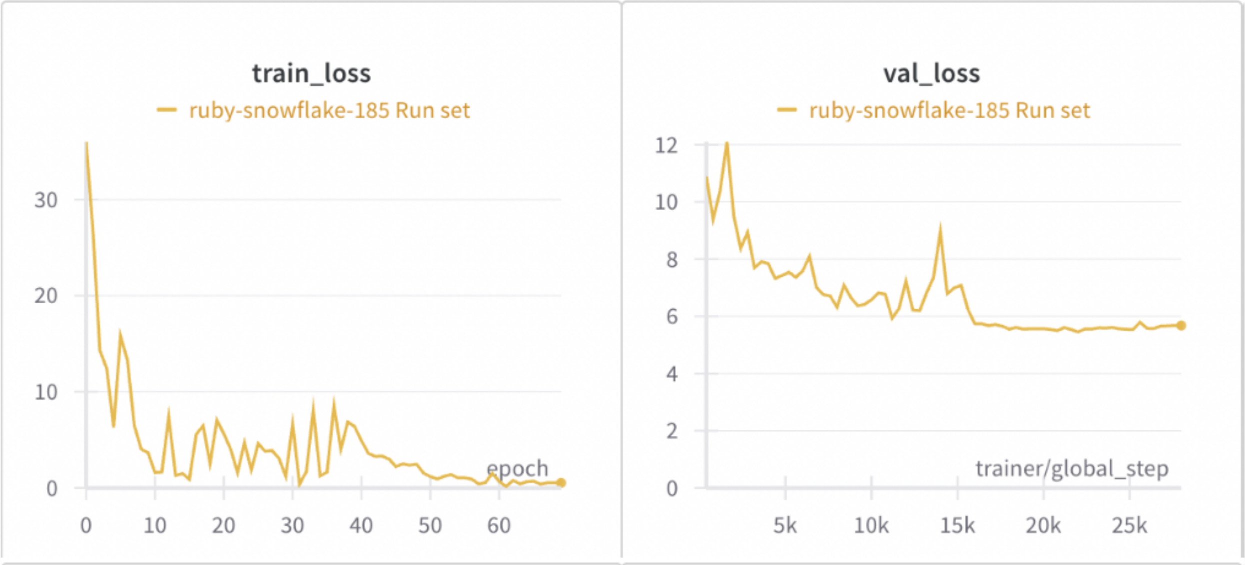
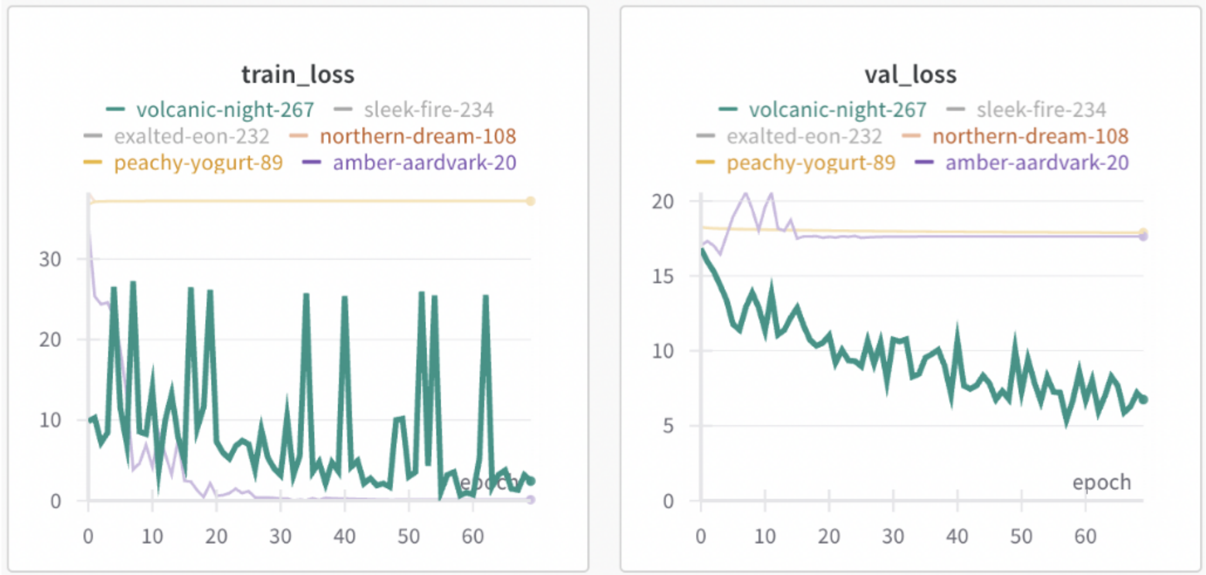
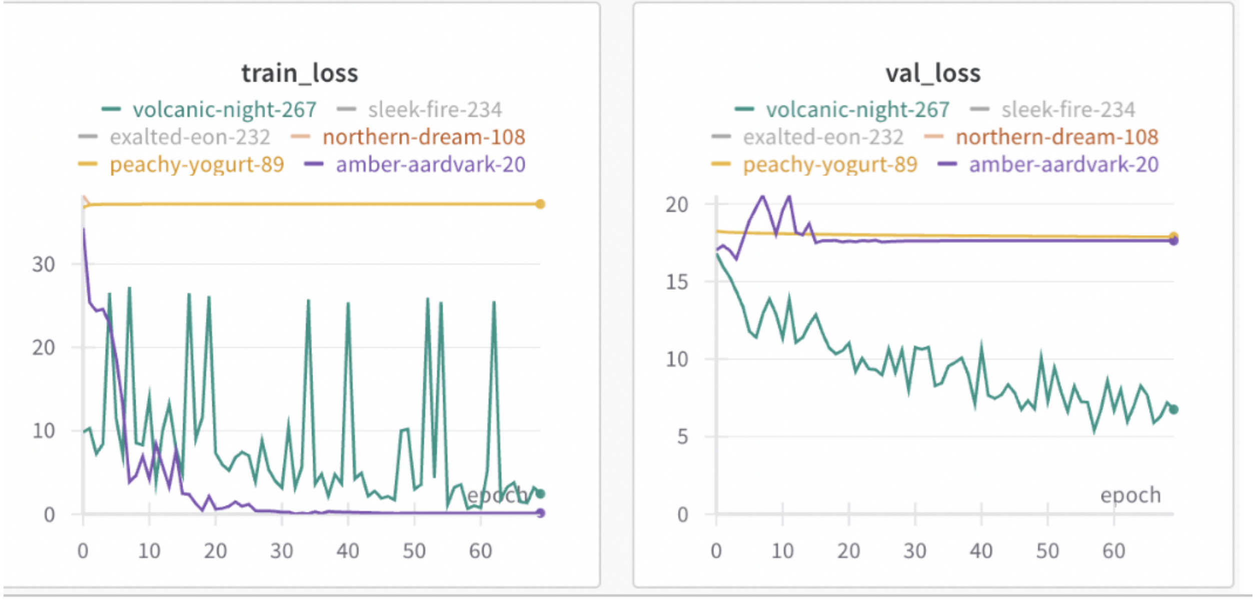


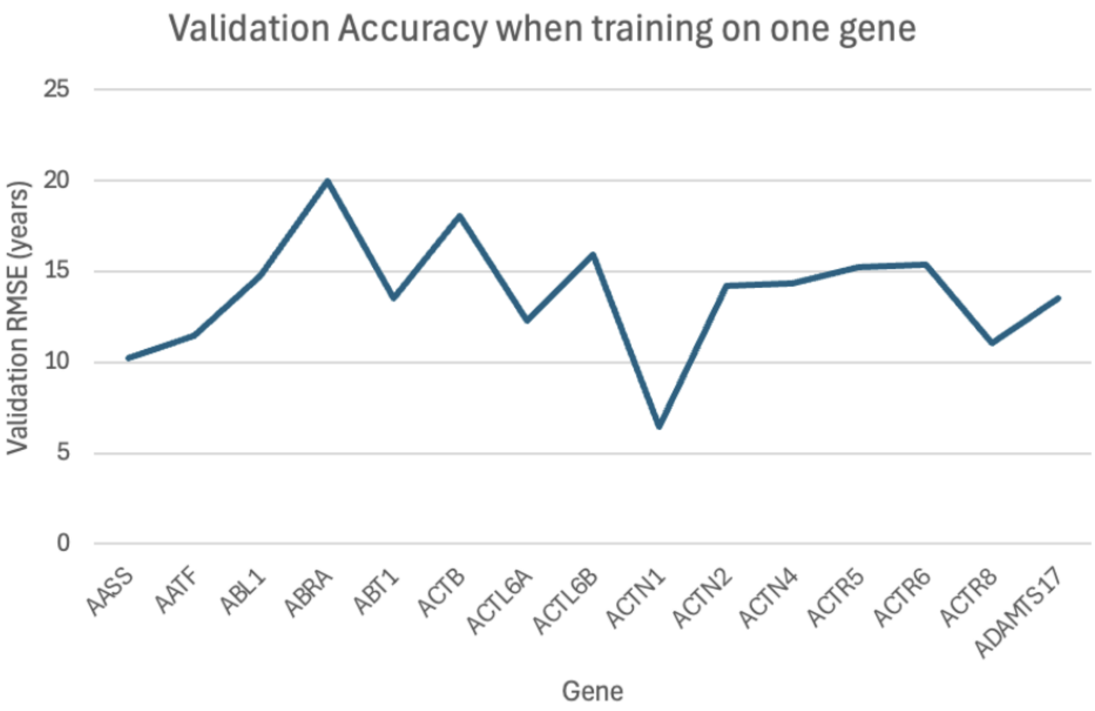
Figure 2: Perceiver Training Results

* The Enformer Model
  * some successful runs; able to get low validation loss on two sets of training data; one with 1000 entries selected from the top 6 genes (according to statistically significant clusters) and another with 100 arbitrarily selected entries across all data
  * positive results here inspired us to investigate training on isolated gene datasets; finding a relationship between DNA composition and lifespan within a certain gene would be particularly interesting
  * when trying to train on larger datasets, we ran into memory issues due to how we were loading in data. That is, we were trying to read in a 20gb dataset rather than processing in smaller chunks at a time (this explains why per-gene analysis cut-off after a very small subset of genes)




train_loss
volcanic-night-267
sleek-fire-234
exalted-eon-232
northern-dream-108
peachy-yogurt-89
amber-aardvark-20
0
10
20
30
0
10
20
30
40
50
60
epoch
val_loss
volcanic-night-267
sleek-fire-234
exalted-eon-232
northern-dream-108
peachy-yogurt-89
amber-aardvark-20
0
5
10
15
20
0
10
20
30
40
50
60
epoch

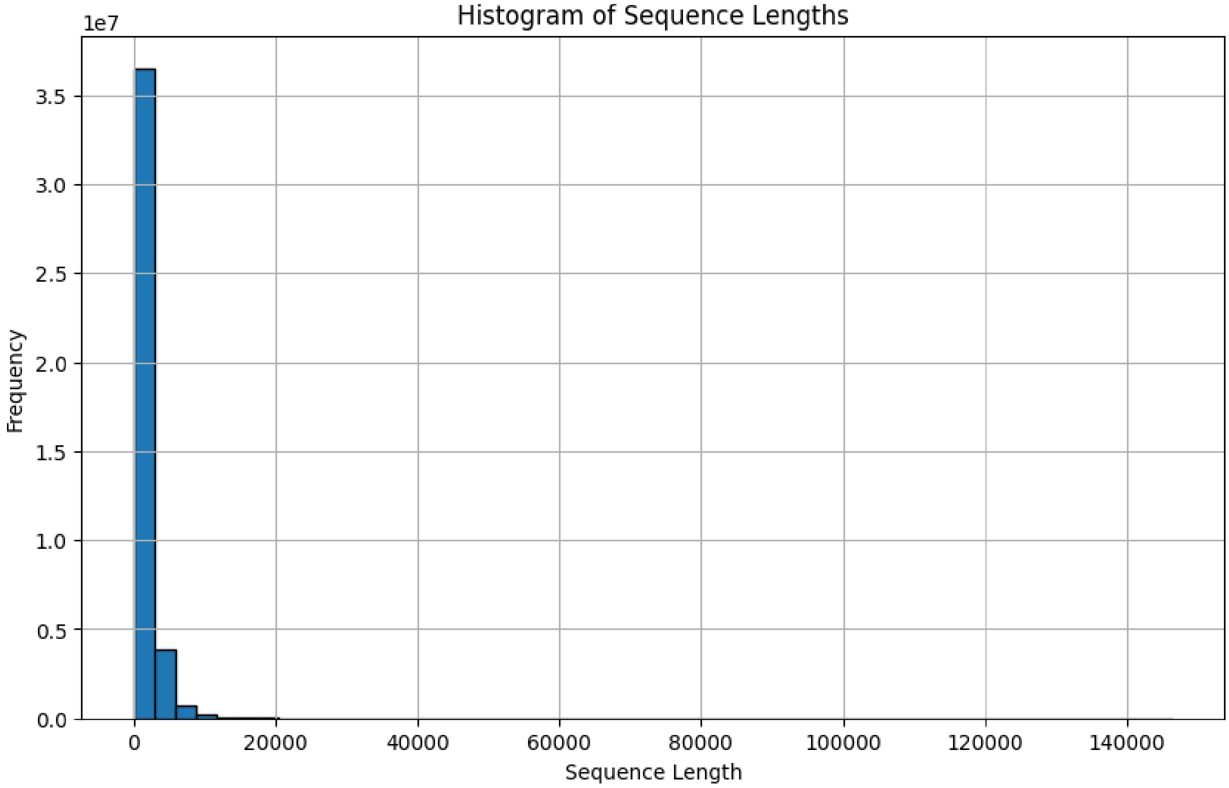

train_loss
volcanic-night-267
sleek-fire-234
exalted-eon-232
northern-dream-108
peachy-yogurt-89
amber-aardvark-20
0
10
20
30
0
10
20
30
40
50
60
epoch
val_loss
volcanic-night-267
sleek-fire-234
exalted-eon-232
northern-dream-108
peachy-yogurt-89
amber-aardvark-20
0
5
10
15
20
0
10
20
30
40
50
60
epoch

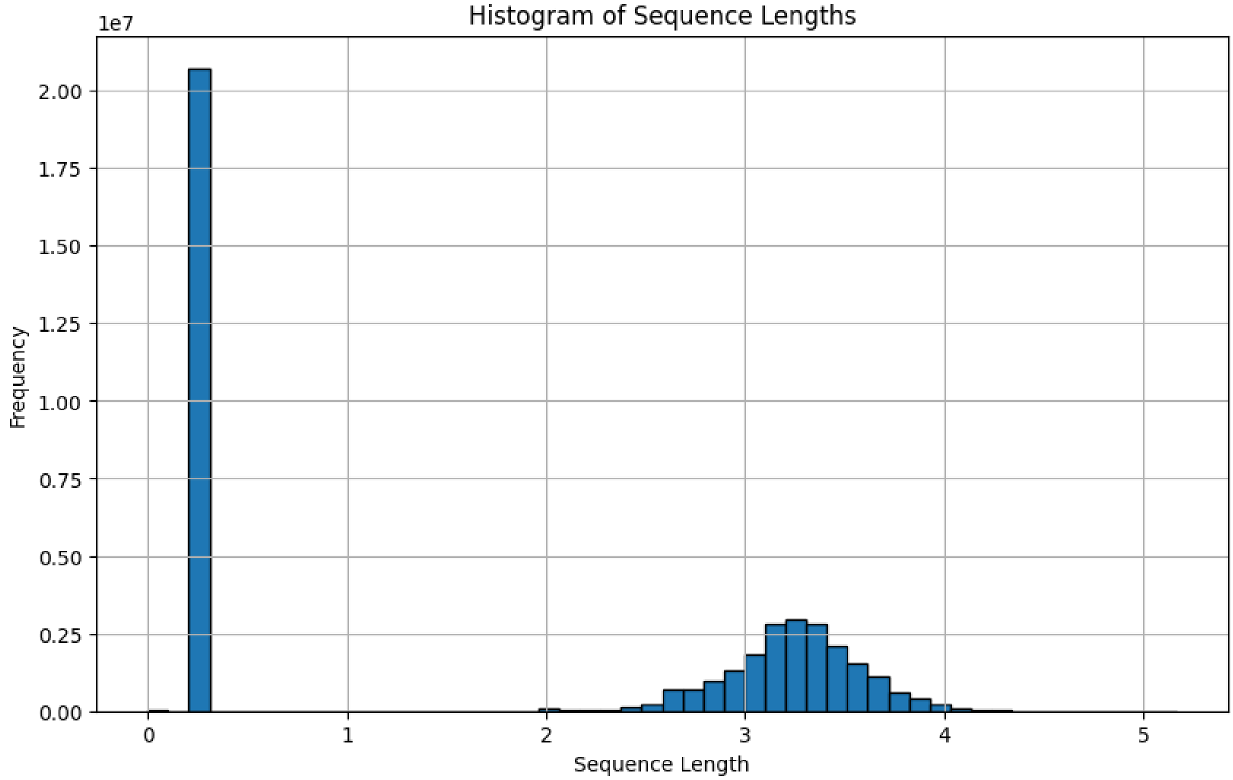


Given our failures to analyze larger quantities of data, we had to re-evaluate our approach to training.

Weeks of: October 1st - October 25th

Focus of this period was to retrace our data collection (from months and months ago) to ensure it's validity, recompile and trim data where necessary and devise a modeling approach in which we could effectively train on these smaller, trimmed datasets. Our first task was to trim our cumulative data. Given the memory issues we encountered earlier, as well as the performance issues incurred by pre-processing extremely long DNA sequences before training a model, we wanted to trim data with outlying sequence lengths. To define a valid range of sequence lengths, we collected metrics on the distribution of sequence lengths across all gene data:

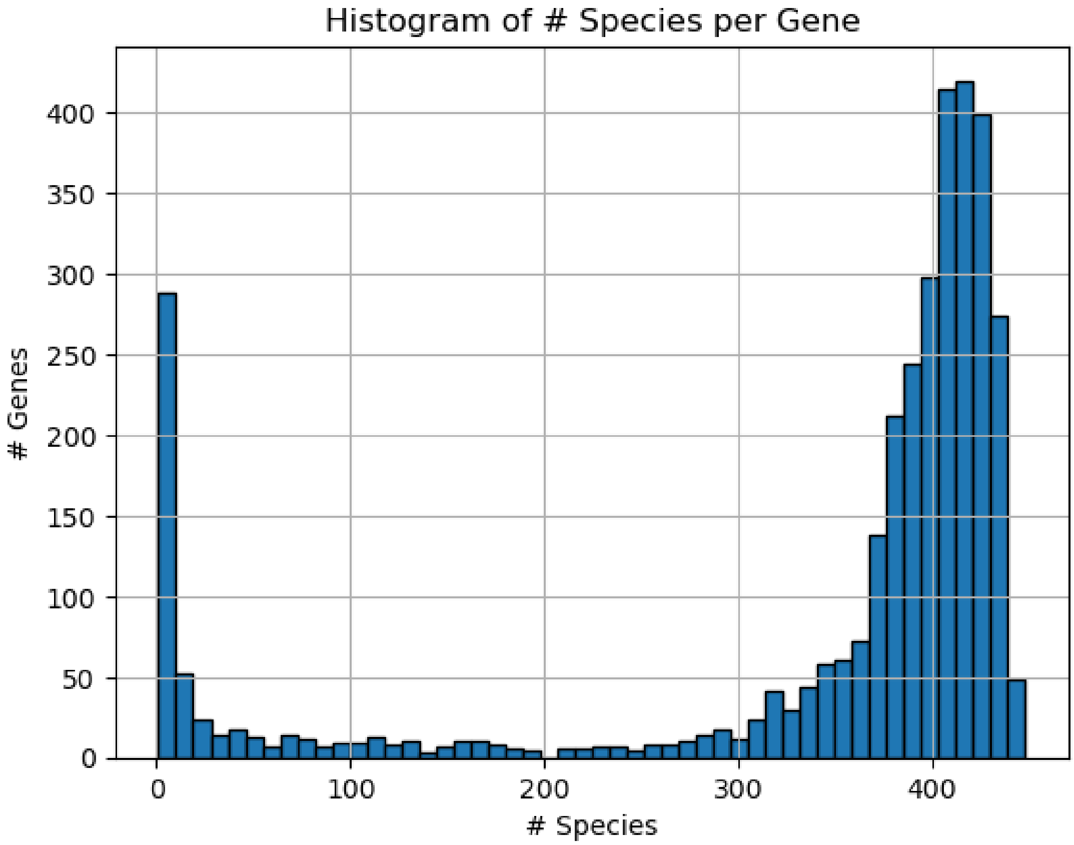




Given the massive range of sequence lengths, we decided an appropriate range of sequence lengths to incorporate in analysis was between 100 (inverse log10(2)) and 31622 (inverse log10(4.5)), where the vast majority of the distribution falls. With that, we decided to trim data such only we only kept sequences with lengths in this range and that were annotated as "one2one" (given our gene annotations come from TOGA, a ML classifier, we want sequences that are most likely to be orthologous). Below you can see the code used for one of our trimming methods (this chunk also outputs some stats on the trimmed data).

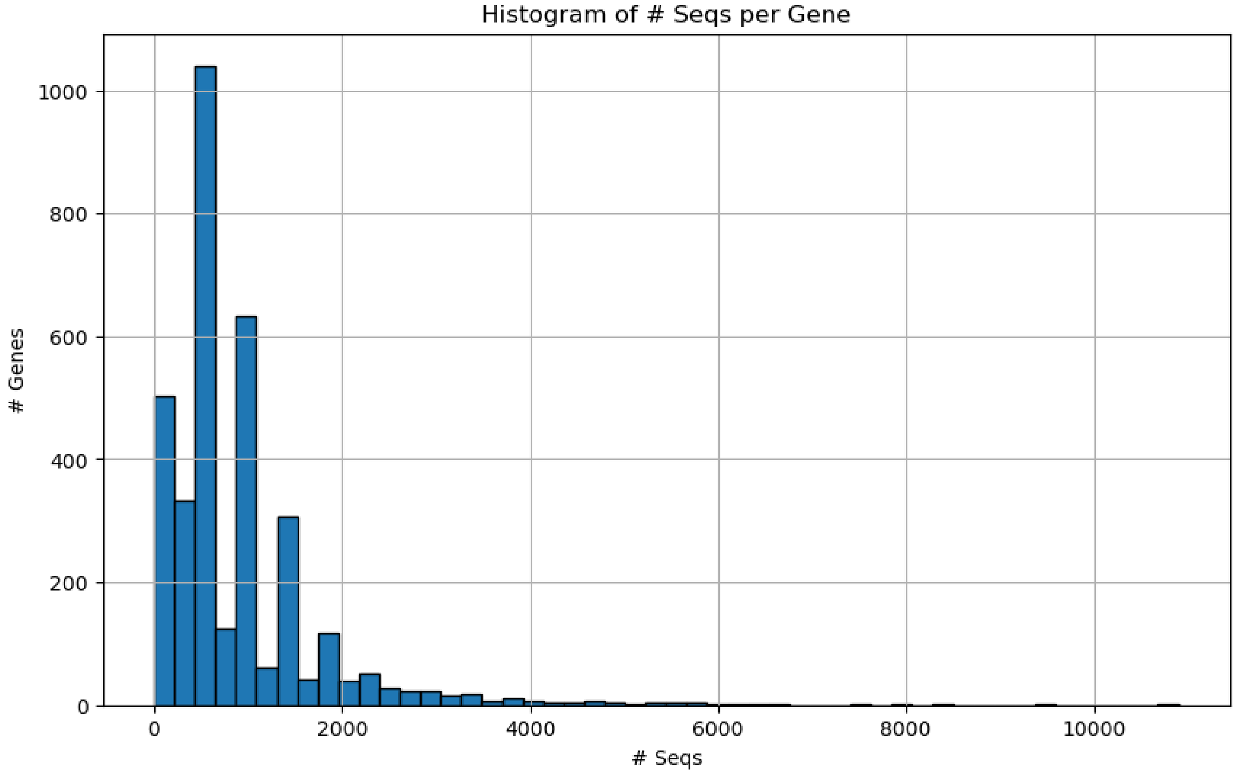
```
# Demo Script for trimming data
"""
method to trim a given gene's csv file path
(expected format:
['organism','max_lifespan', 'gene_id',
'orthologType','chromosome','start','end',
'direction (+/-)','intactness','sequence'])
generates new file only including one2one sequences btwn lengths of 104 and 31620
computes stats of # of sequences and # of species represented per gene
"""
def create_one2one_gene_sets():
    gene_stats = {} #dict to hold gene:(num_data_entries, num_species) pairs
    num_genes = 0
    for file in os.listdir(gene_datasets_path):
        file_path = gene_datasets_path + file
        new_file_path =
          "/".join(file_path.split("/")[:-2]) +
          "/regulatory_one2one/" + "".join(file.split(".")[:-1]) +
          "_one2one.csv"
        #only execute on most up-to-date 'trimmed' gene files
        if file_path[-21:] != "orthologs_trimmed.csv": continue
        num_genes += 1
        cur_gene = file.split("_")[0]
        num_data = 0
        organisms = set()

        with open(new_file_path, "w") as write_to:
            writer = csv.writer(write_to)
            writer.writerow(
              ['organism','max_lifespan', 'gene_id',
              'orthologType','chromosome','start','end',
              'direction (+/-)','intactness','sequence']
            )
            with open(file_path) as read_from:
                for line in read_from:
                    line = line.split(",")
                    if len(line) < 10: continue
                    seq = line[-1].strip()
                    length = len(seq)
                    if length < 104 or length > 31620:
                      continue #only include seqs of length in this range
                    ortholog_type = line[3]
                    if ortholog_type != "one2one":
                      continue #only include one2one seqs
                    num_data += 1
                    print(line[0])
                    organisms.add(line[0])
                    writer.writerow(line)

        #fill dict entry for current gene
        gene_stats[cur_gene] = (num_data, len(organisms))

        write_to.close()
        os.system(f'rm -rf {file_path}')

    # after iterating thru all genes s.t we've generated one2one, trimmed, gene sets
    # create new file that outputs stats per gene
    # (csv of format [gene | num_entries | num_species])
    # we can use this file to generate histograms thereafter

    with open(
      "/data/rbg/users/wmccrthy/chemistry/Everything/EDA/regulatory_one2one_sets_metadata.csv",
      "w") as write_to:
        writer = csv.writer(write_to)
        writer.writerow(["gene", "# seqs", "# species"])
        for gene in gene_stats:
            num_seqs, num_species = gene_stats[gene]
            writer.writerow([gene, num_seqs, num_species])
    write_to.close()
```

For the time being, we want to experiment with training on individual genes (which can be individually pre-processed and tokenized) to avoid the memory issues we were encountering earlier. Thereafter, we will circle back to training on larger datasets. Once our data was trimmed according to the above constraints, we collected the following to inform this training: * distribution of number of species represented in each gene



dataset *

distribution of number of sequences in each gene dataset



Given the distribution of species in each gene dataset, we decided it makes sense to train on gene sets with at least 300 species represented.

Weeks of: October 28th - . . .

Focus of this period is to run scripts to train Enformer model on each of our gene datasets, evaluate results, and move forward accordingly. . .

3.1 Training Results