Supplementary Information for

MOLE: Multi-Omics Learning to Extrapolate Proteome Expression

Table of Contents:

1. Experimental data

1.1 Experimental data and its normalization

The Tissue29 dataset is a systematic, quantitative and deep proteome and transcriptome abundance atlas from 29 paired healthy human tissues to serve as a molecular baseline to study human biology.

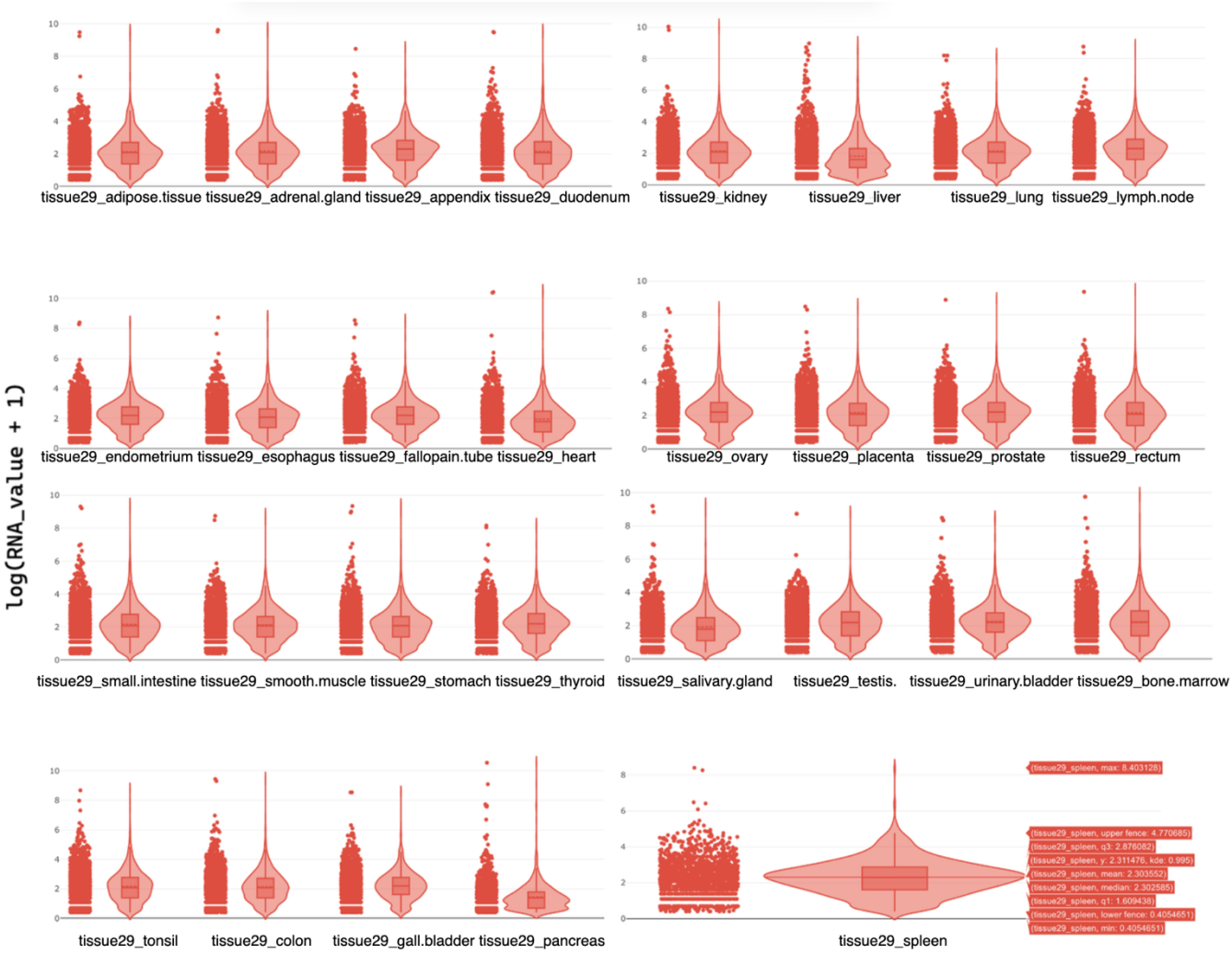

Figure S1. Human normal tissues dataset validation RNA expression distribution per sample.

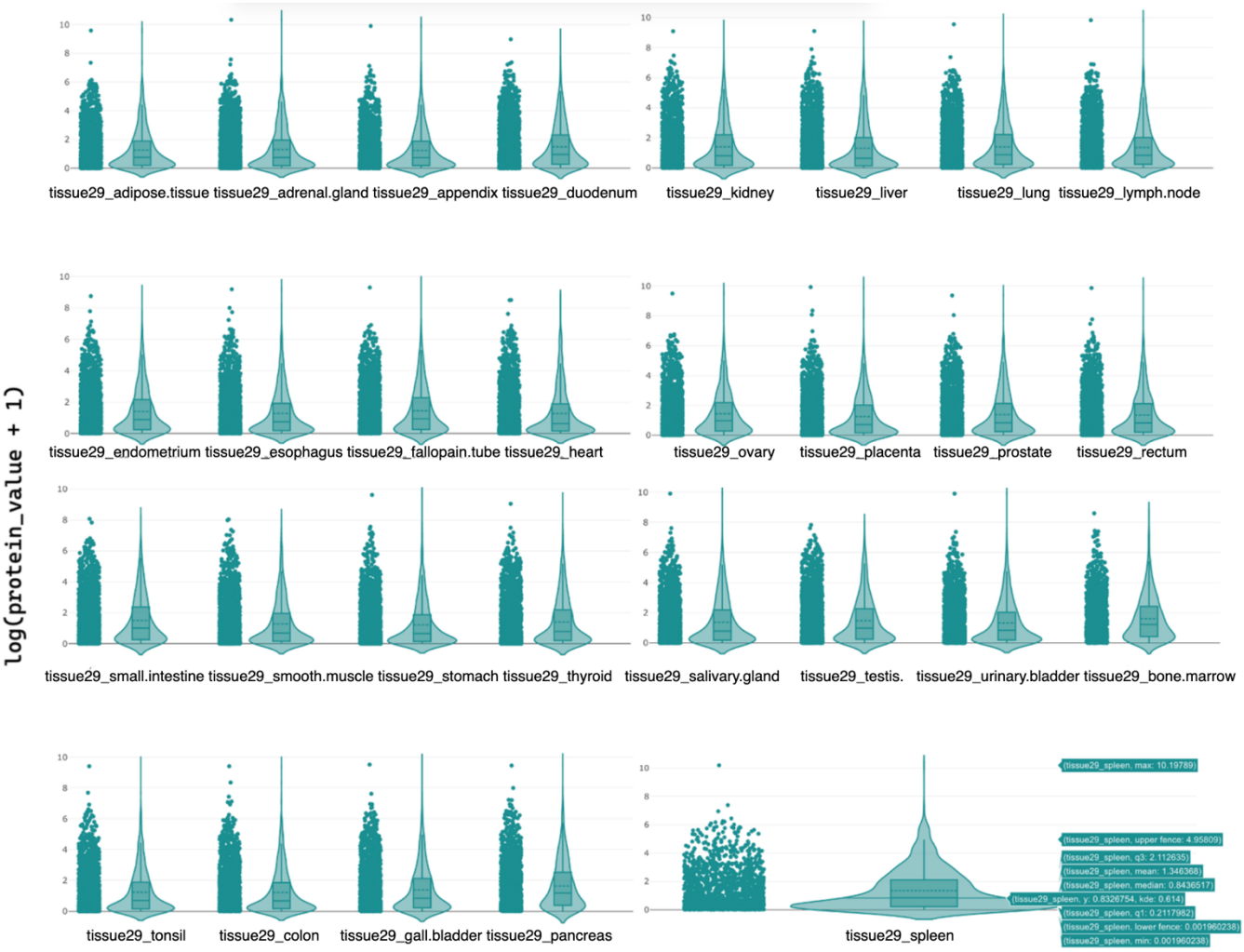

Figure S2. Human normal tissues dataset validation protein expression distribution per sample.

The NCI-60 cell line collection is a very widely used panel for the study of cellular mechanisms of cancer in general and *in vitro* drug action in particular. Was implemented in fully operational form in 1990 and utilizes 60 different human tumor cell lines to identify.

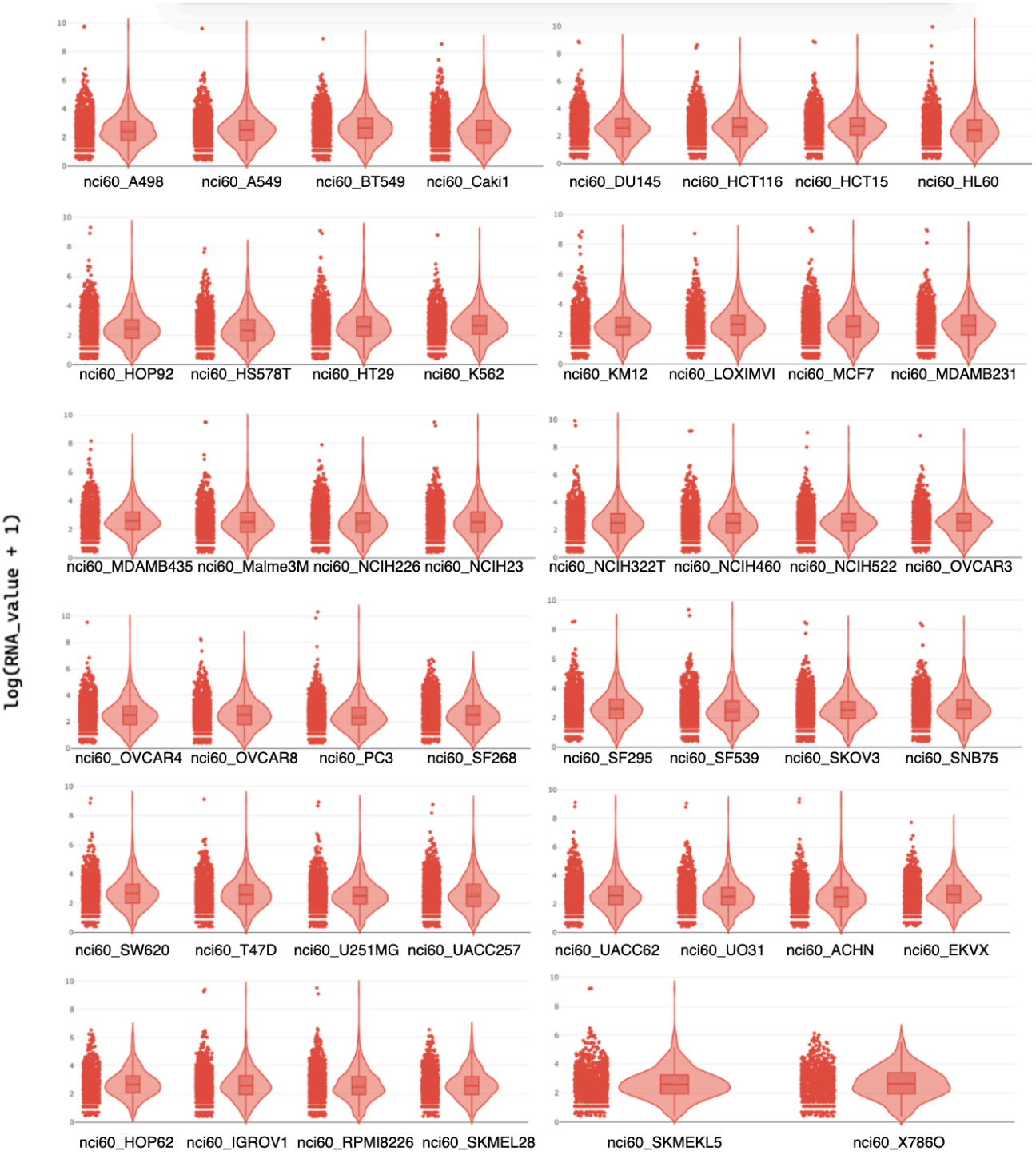

Figure S3. Human tumor cell lines dataset validation RNA expression distribution per sample.

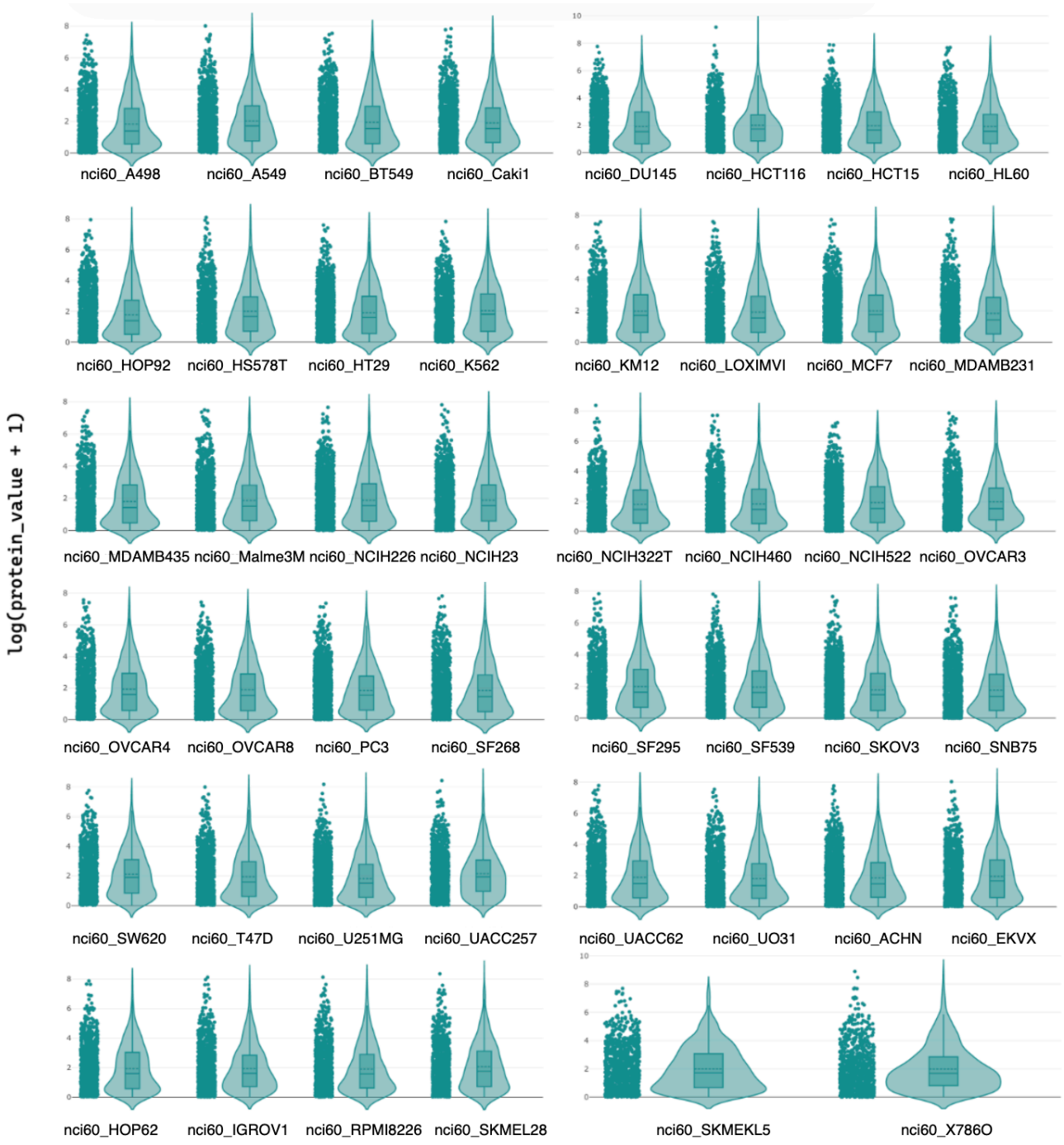

Figure S4. Human tumor cell lines dataset validation protein expression distribution per sample.

From the UniProt database website was downloaded a file with mappings to 22 databases

https://ftp.uniprot.org/pub/databases/uniprot/current_release/knowledgebase/idmapping/by_organism/HUMAN_9606_idmapping_selected.tab.gz [26]. The file contains UniProtKB-AC, UniProtKB-ID, GeneID (EntrezGene), RefSeq, GI, PDB, GO, UniRef100, UniRef90, UniRef50, UniParc, PIR, NCBI-taxon, MIM, UniGene, PubMed, EMBL, EMBL-CDS, Ensembl, Ensembl_TRS, Ensembl_PRO and additional PubMed databases mappings. Some of them repeat another or have one-to-one mappings.

1.2 Data Caching

Each gene tensor could be cached to speed up work with genecentric databases and annotations. One cached tensor representing gene databases mappings and amino acids vectorized sequence prepared to train or inference weights 1.8 MB. After loading the gene tensor the channels corresponding to protein tissue id, RNA tissue id and normalized RNA value are modified by data loader depending on the current experiment.

For instance, cached genes tensors for the experiment of 12421 unique genes occupy 20.5 GB (the full non-zero experiments Tissue29 dataset).

UniProt API responses were cached to avoid wasting time waiting for an answer from the database. All data preparation before training start steps (we could call it the first epoch for the dataset) can be processed without GPU-acceleration.

2. Models

2.1 ResNet34V1.3

Our experiments with training of different model settings led to using the ResNetV1 model with depth of 34 stacked layers with each stacked layer convolutional kernels channels modification factor of 3. First usually used convolution with the 7x7 kernel (to reduce the dimension space of big image tensors) was changed to 3x3 kernel and max pooling before layers stacking was removed.

Layers stacking was done in order (3, 4, 6, 3) (Table 7). The designed model architecture called ResNet34V1.3 has 3.1 million parameters (Table 7). This model is the smallest in our experimental cohort.

2.2 ResNet50V2

Models from ResNetV2 usually started from 50-layers depth due to changes in bottleneck and no basic block realization. The main idea is in rearranging block convolutions and using normalization and activation before 2D convolutions in block, so-called, pre-activation blocks (Figure 9 in main text). Another important difference of the used model variance is in replacing usual Conv2d layers with 2D convolution layers with weight standardization [21] and using group normalization [19] instead of batch normalization [20].

In used modification 3 and 4 network layers were stacked to have smaller reducing of features space, because of that layers stacking in the 50-depth version looks like

(3,4,9) not (3,4,6,3) like in classic ResNet. This model version has 11.92 million trainable parameters with exported model file weights 45.5 MB (Table 7).

2.3 ResNet26V2

In the first version this depth could be done by basic blocks stacked in (3, 4, 6, 3) order and the 18-depth version is created by the same basic block in (2, 2, 2, 2) stacking order. However, the ResNetV2 version doesn't have a basic block realization due to lighter bottleneck blocks. Because of that we had modified the 50-depth model by stacking bottleneck layers in (2, 2, 4) order and had got a robust training but light model with 5.7 million params and file weights 21.8 MB (Table 7) named ResNet26V2.

3. Learning rate scheduling

Setting of learning rate hyperparameter scheduling plays a significant role in model training convergence and in model finetuning. This parameter impacts directly on trainable parameters shift between batches inference and with the use of static value it is hard to get a robust model. In the work we have used two schedulers.

3.1 Scheduling by validation metric plateau

Scheduling learning rate by validation metric plateau is one of classic approaches, it's done by setting the min or max mode of the metric with thresholds and cooldowns based on the metric state. We have started training from 0.003 value reducing by 0.3 factor it further based on model convergence.

3.2 Cosine scheduling with warmup restarts

For a stable model convergence with bigger parameters count we have used cosine learning rate schedulers with warmup restart [28]. This approach allows iterating from min to max learning rate intervals through it based on inferenced training batches in a non-linear way (Figure S5).

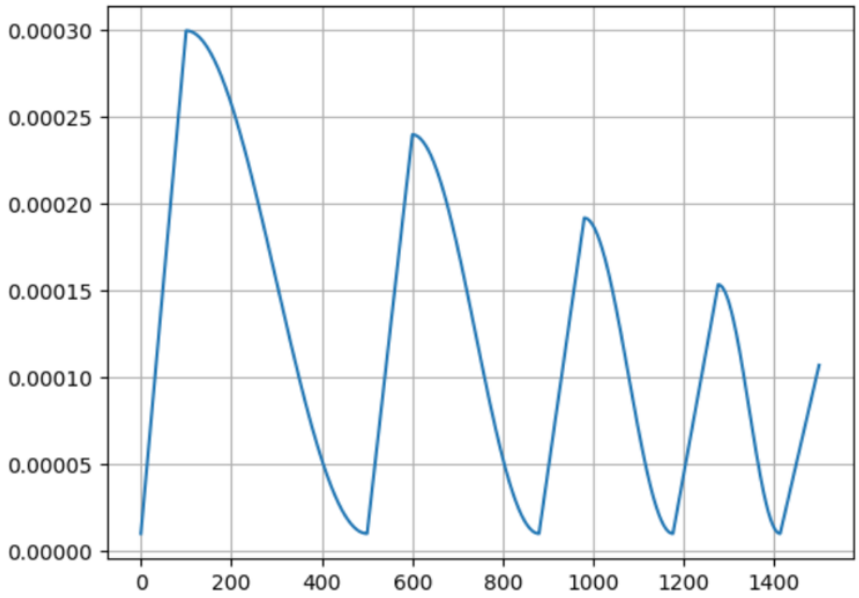

Figure S5. Example of learning rate changes by cosine scheduler with warmup restarts in interval from 0.00001 to 0.0003: one cycle contains 500 steps, 100 of them is warmup, each cycle length scaled by 0.8 and max learning rate scaled by 0.7. Iteration step is done when each batch is passed, not once at epoch.

Warmup restarts allow to start model training from minimal learning rate and give great results in model convergence. Another interesting thing is using parameters that scale max learning rate or cycle length. Using this approach helps us to train bigger

models in several epochs, further finetuning was done by setting learning rate by plateau.