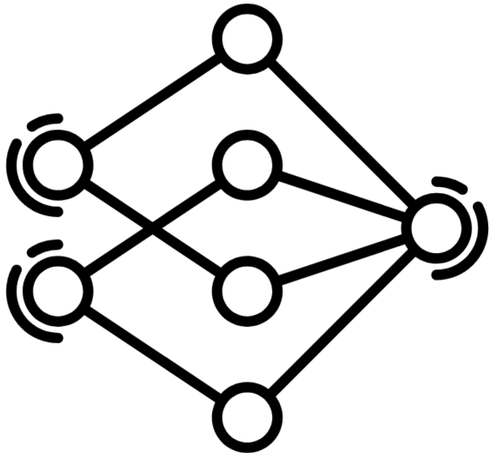


Kolmogorov-Arnold Networks

Supervised learning & Unsupervised learning

Shamanth Kuthpadi Seethakantha

How can we leverage KANs to perform **supervised** and **unsupervised** learning?

Agenda

05/30/25

Preliminaries

Experiment

Setup

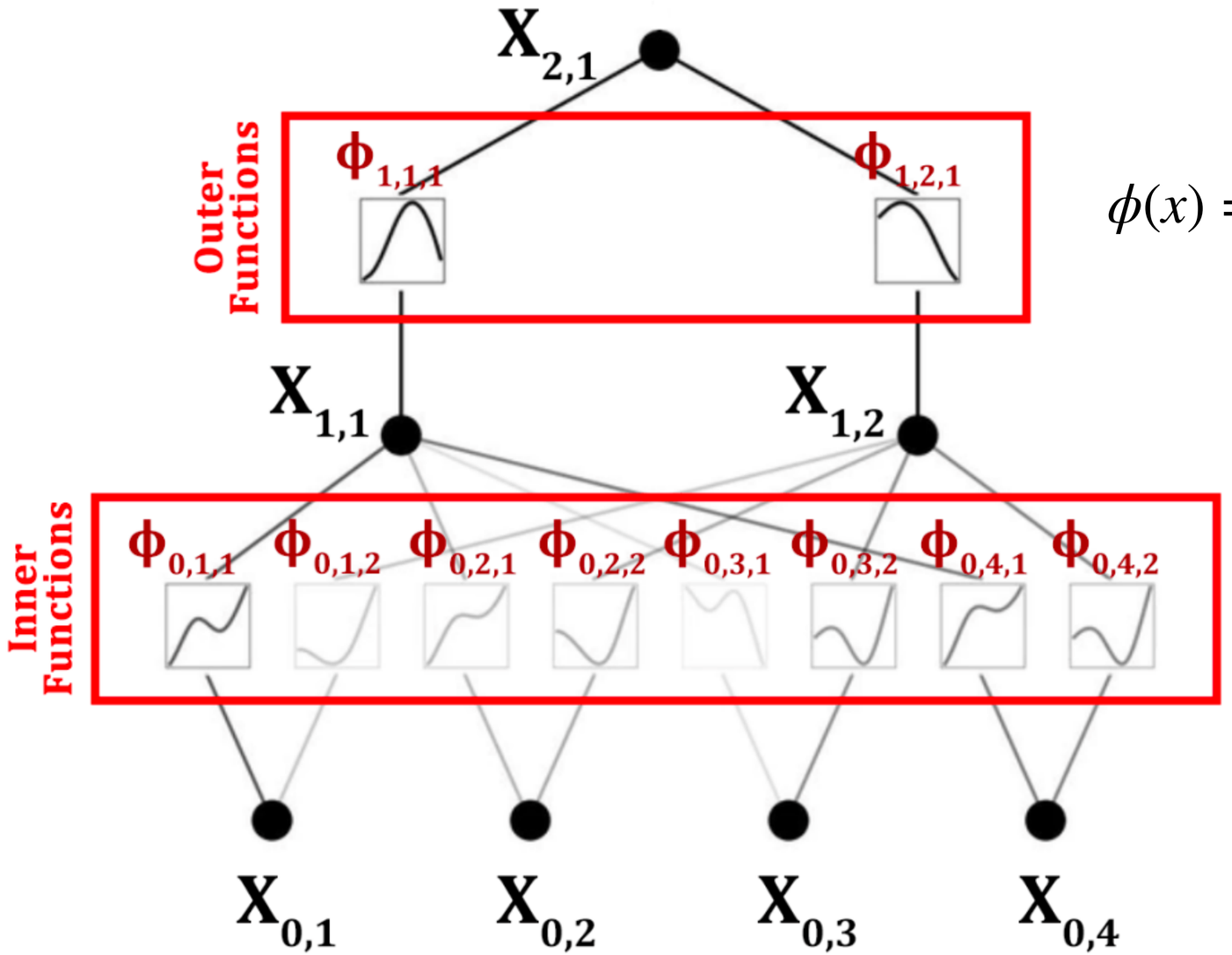
Code instrumentation

Results (highlight only)

Emerging Conclusions

Next Steps

Preliminaries



Activation Function on the Edges

$$\phi(x) = scale\_base * b(x) + scale\_sp * spline(x)$$

We fix ***scale_base***, ***b(x)***, and ***scale_sp***.

We learn the parameters relating to ***spline(x)***.

Cacciatore et al.
A Preliminary Study on Continual Learning in Computer Vision Using Kolmogorov-Arnold Networks

$$spline(x) = \sum_{i}^{G+k-1} c_i B_i(x)$$

$$spline(x) = \sum_{i}^{G+k-1} c_i B_i(x)$$

- ***x***: value at which spline funtion is being evaluated
- ***G***: number of grid intervals used to defined the B-spline basis, set during initialization as the `grid` parameter
- ***k***: polynomial order of the B-spline
- **c**: coefficients of the basis functions
- ***B(x)***: basis functions of B-spline
- ***G+k+1***: number of B-spline basis functions

$$spline(x) = \sum_{i}^{G+k-1} c_i B_i(x)$$

- ***x***: value at which spline funtion is being evaluated
- ***G***: number of grid intervals used to defined the B-spline basis, set during initialization as the `grid` parameter
- ***k***: polynomial order of the B-spline
- **c**: coefficients of the basis functions
- ***B(x)***: basis functions of B-spline
- ***G+k+1***: number of B-spline basis functions

API 4: Initialization

Initialization is the first step to gaurantee good training. Each activation function is initialized to be $\phi(x) = \text{scale\_base} * b(x) + \text{scale\_sp} * \text{spline}(x)$. 1. $b(x)$ is the base function, default: 'silu', can be set with base_fun

2. scale_sp sample from N(0, noise_scale^2)
3. scale_base sampled from N(scale_base_mu, scale_base_sigma^2)
4. sparse initialization: if sparse_init = True, most scale_base and scale_sp will be set to zero

Activation functions on the edges of the KAN are the building blocks of the learning process

Experiment

Experiment Design

- Employ the one-factor-at-a-time (**OFAT**) experiment design to elucidate the impact of each factor or hyperparameter on both the outcome and the learning process.
    - Select a hyperparameter (*hp)* to analyze.
    - Determine specific values for $hp = \{V\}$.
    - Execute the pipeline for each value in V and quantitatively assess the learning process.
- Repeat the experiment method after applying a **shock** to comprehend the influence of the **coefficients.**

Shock Introduction

There are a lot of plots that are generated and can be found in the GitHub repository.

For today's talk I will just focus on a few key plots that highlight some emerging conclusions.

- Midway through the training process apply shock (i.e. shock at 25th epoch if there are 50 epochs in all)

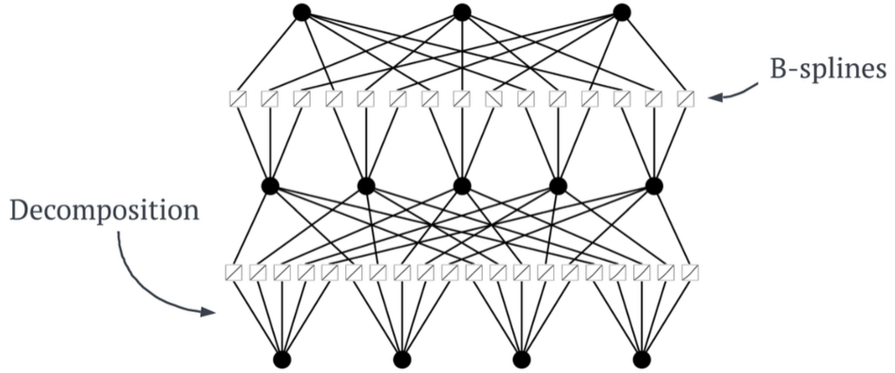
- Peturb all coefficients of all layers via random Gaussian noise

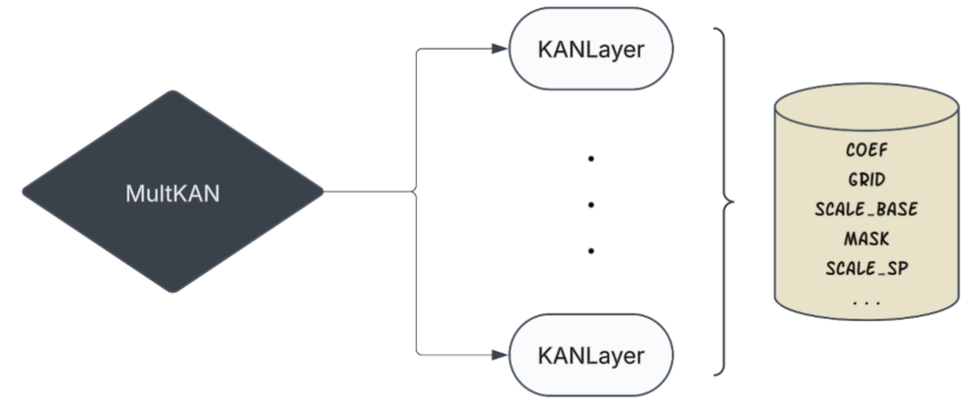
https://github.com/shamanth-kuthpadi/KANs-IOMICS

Source Code Instrumentation

pykan

- Each layer has some learnable activation parameters on the edge
- Find the average of each of those parameters for a given layer
- For each parameter find the mean across the layer averages
- Record the averages at each step of the fitting process
- Visualize as a plot





$$spline(x) = \sum_i^{G+k-1} c_i B_i(x)$$

$$\phi(x) = scale\_base * b(x) + scale\_sp * spline(x)$$

$$b(x) = silu(x) = \frac{x}{1+e^{-x}}$$

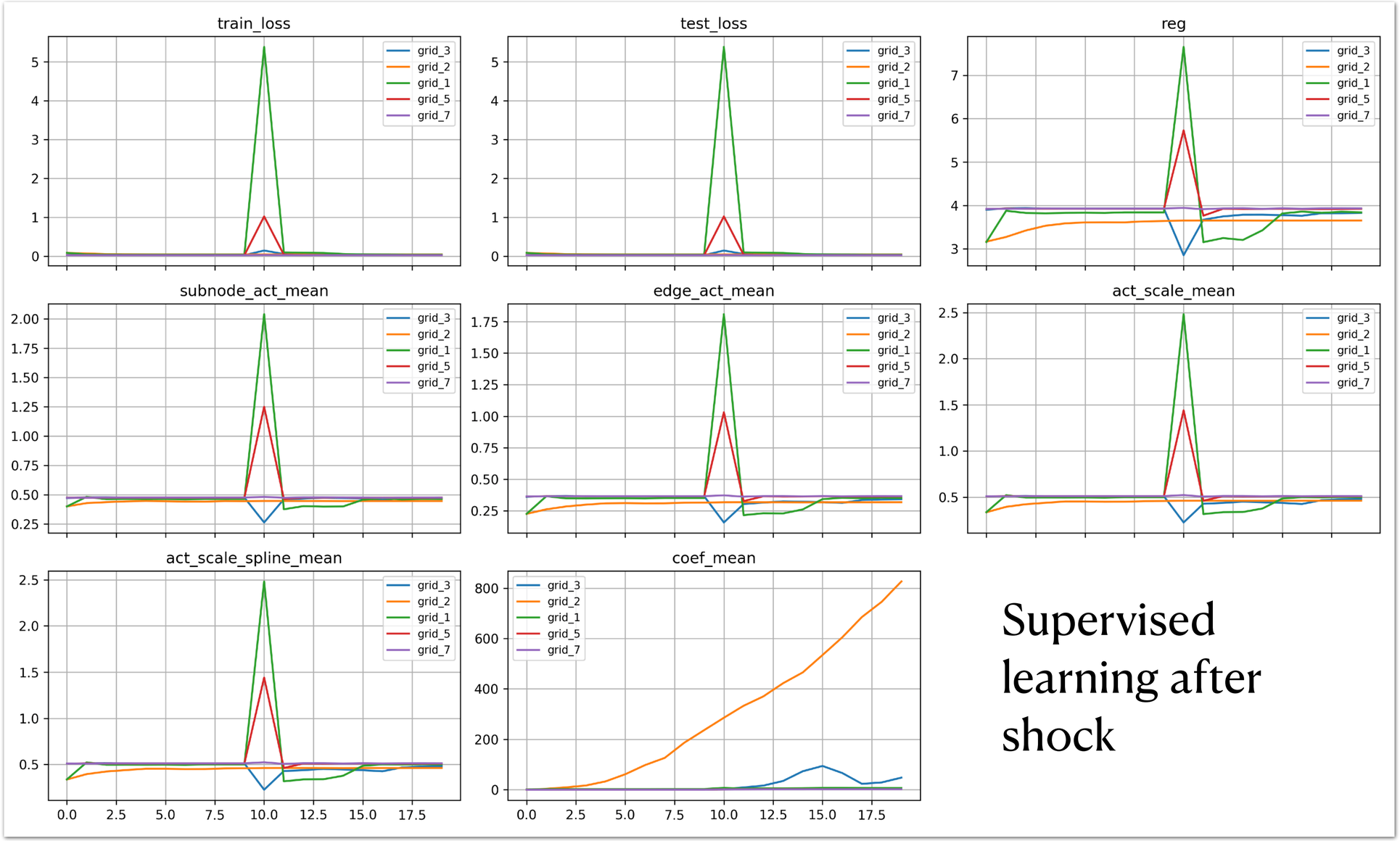
Emerging Conclusions

Impact of Shock

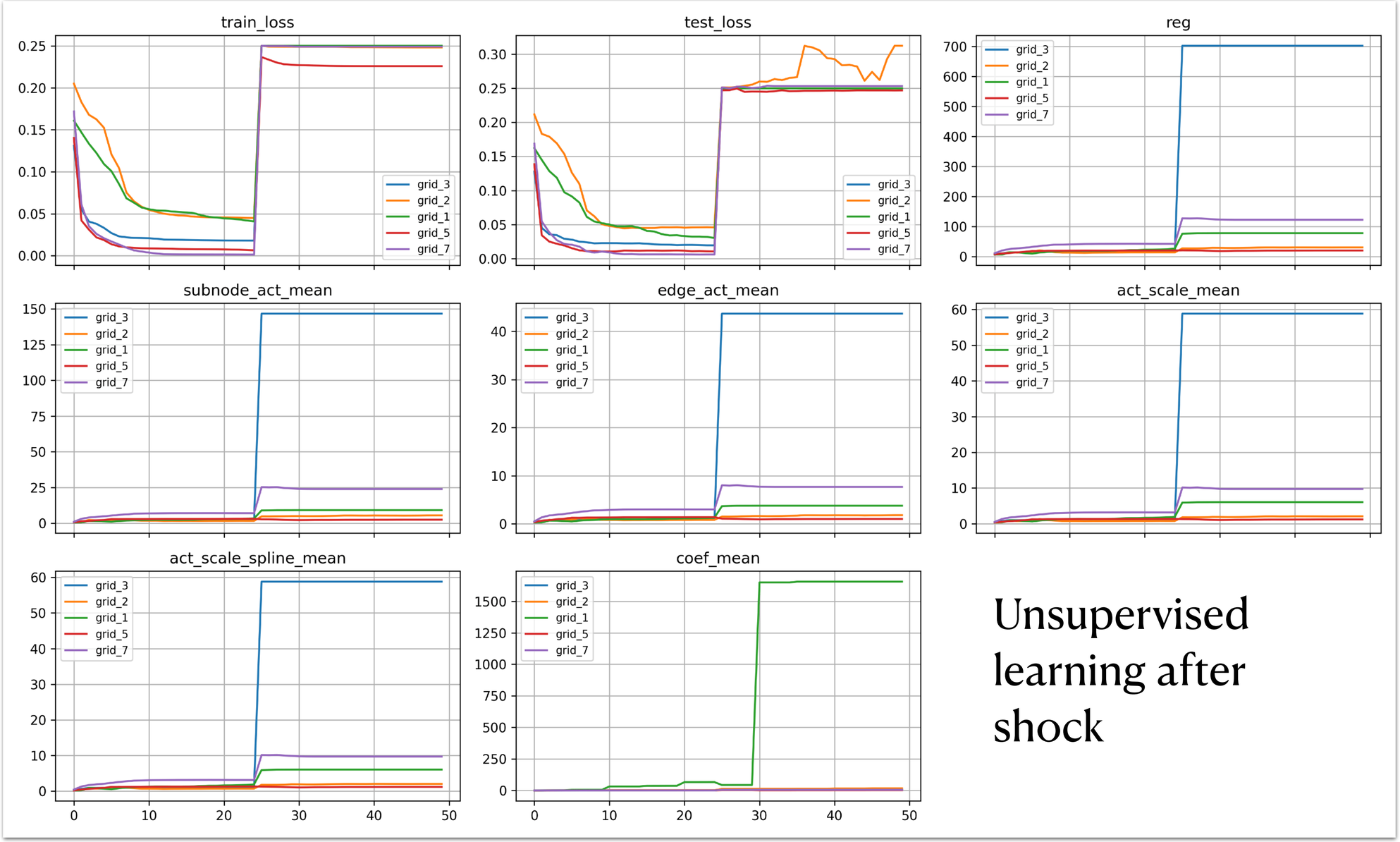
Unsupervised learning **has trouble** recovering after the shock

Supervised learning (classification based) usually has **issues** even finishing training after shock

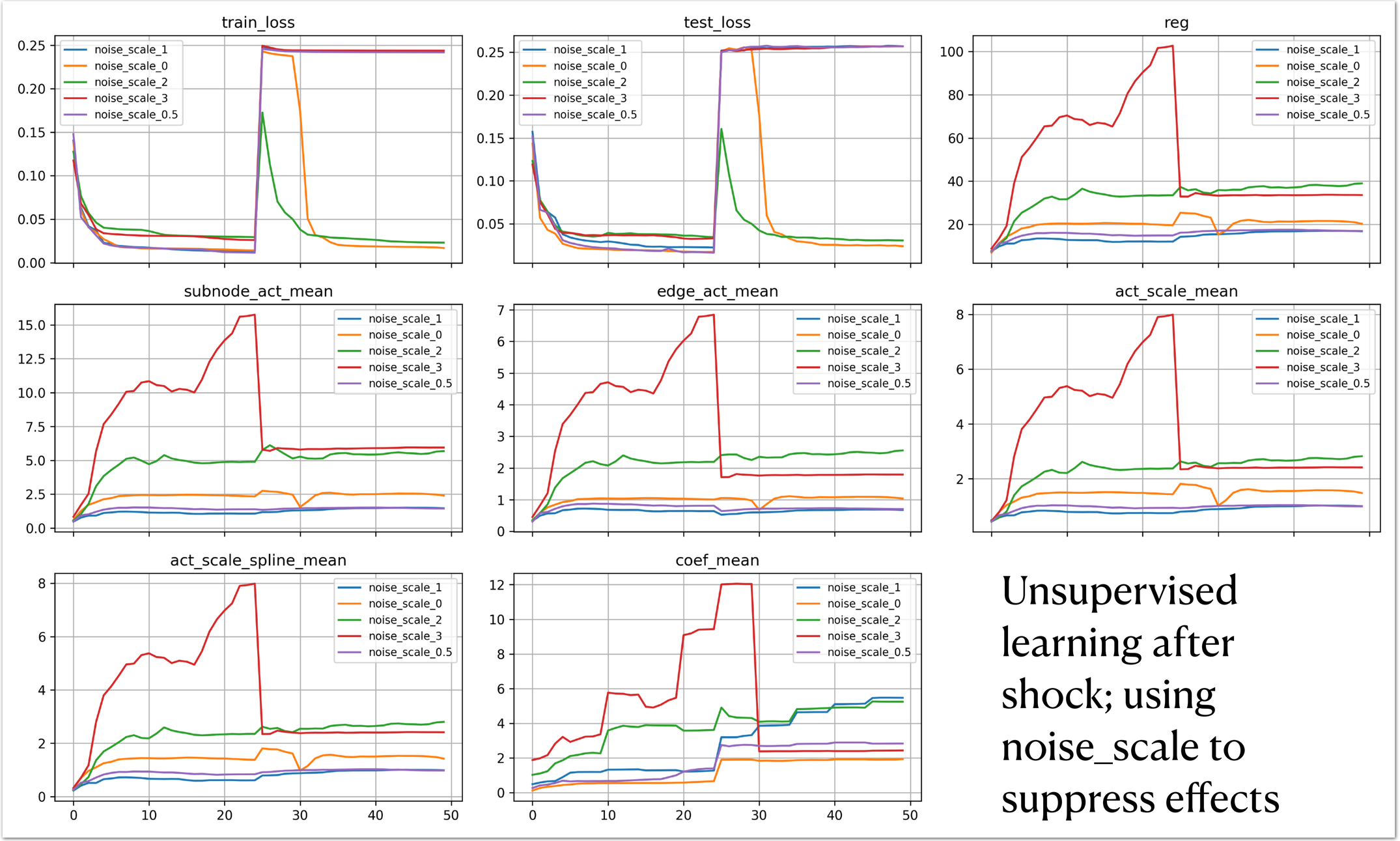
Supervised learning (regression based) is able to **recover** from the shock



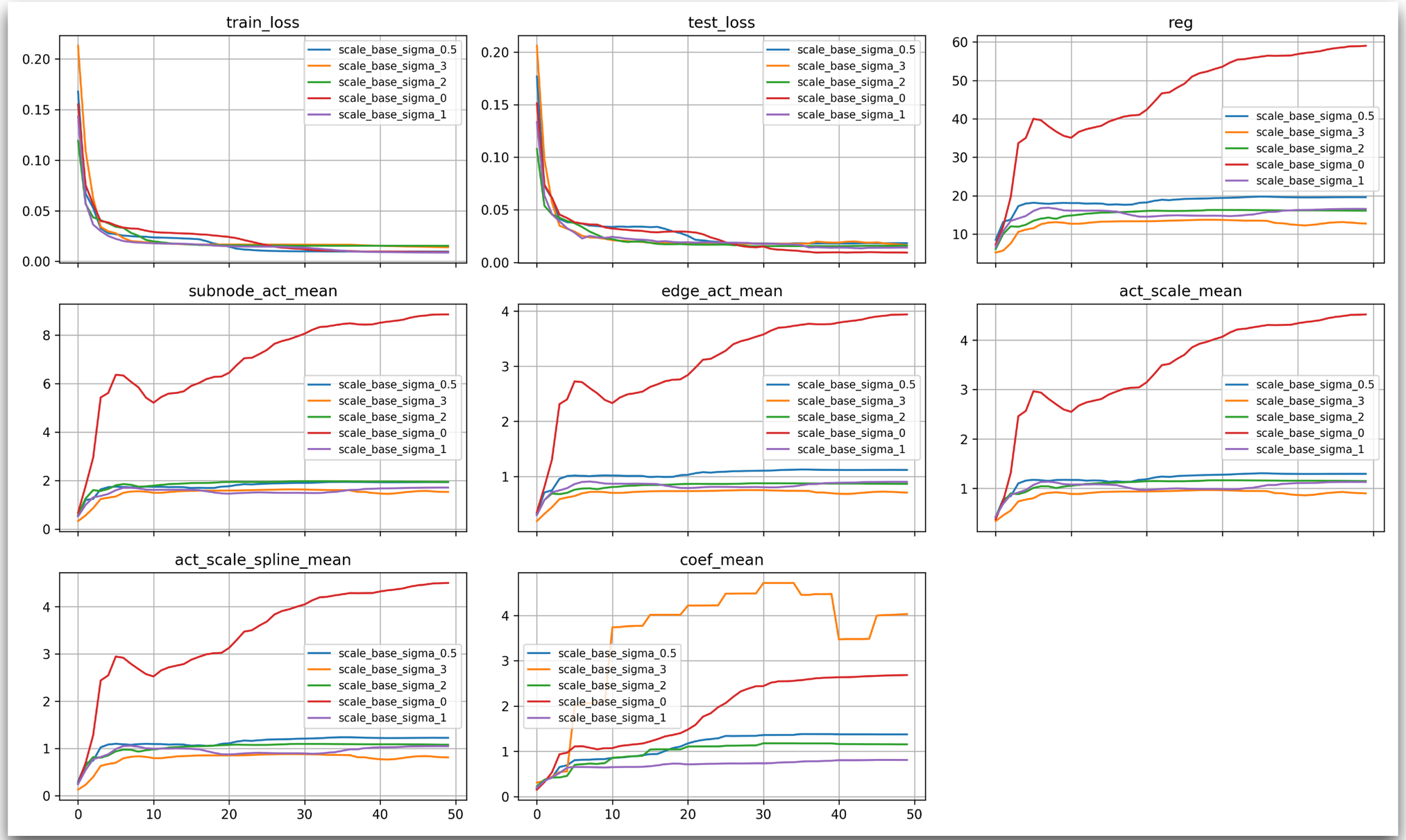
Supervised learning after shock

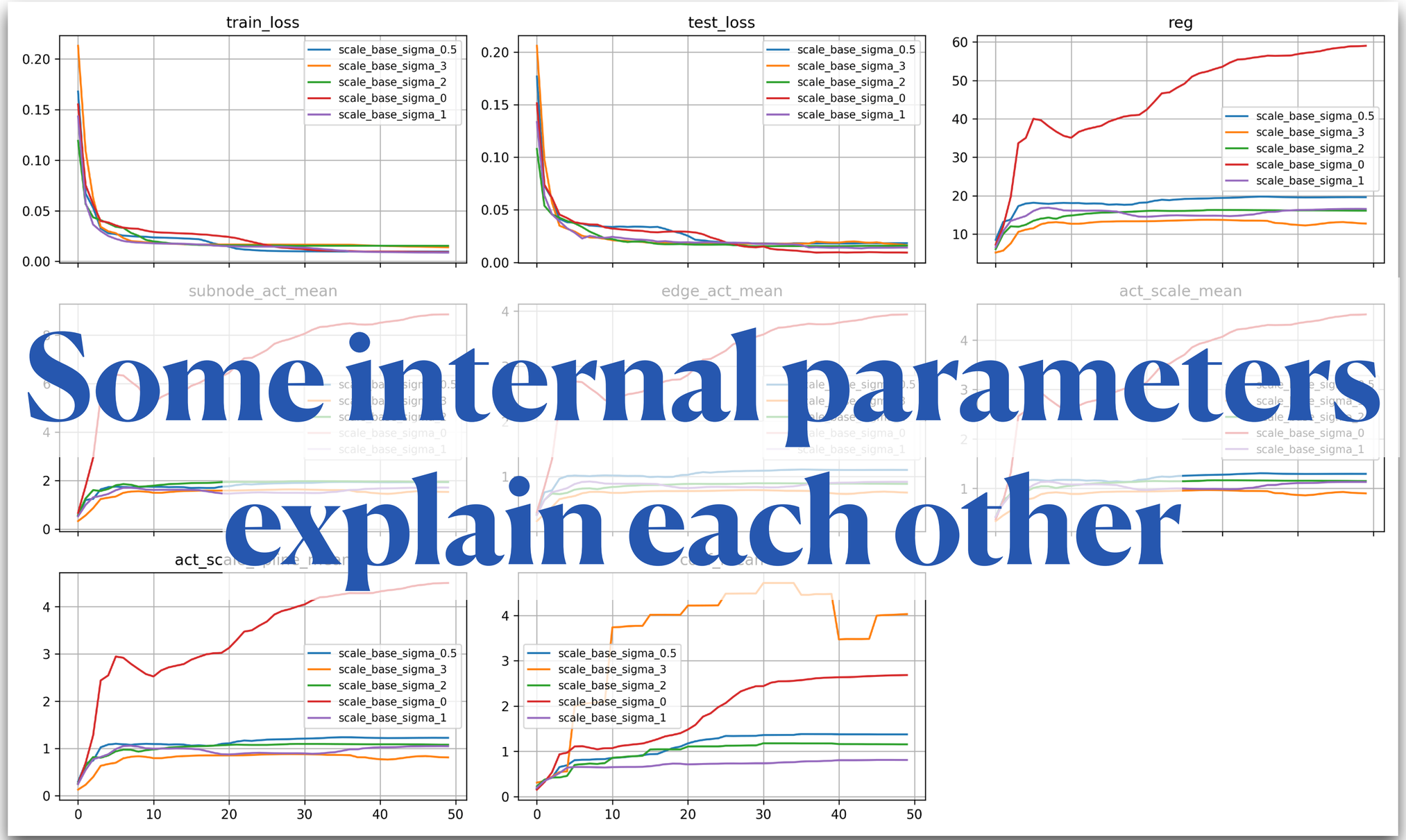


Unsupervised learning after shock

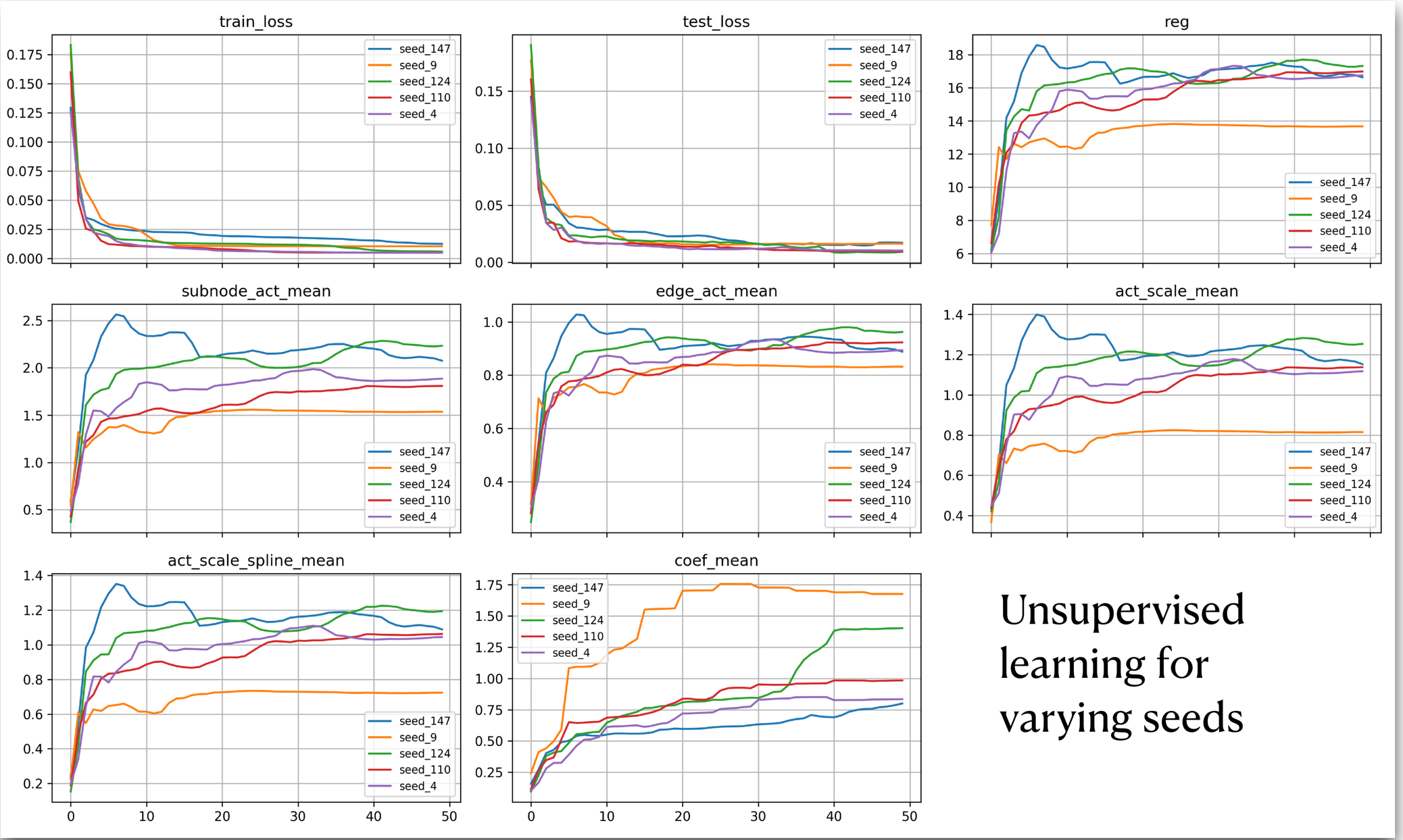


Unsupervised learning after shock; using noise_scale to suppress effects

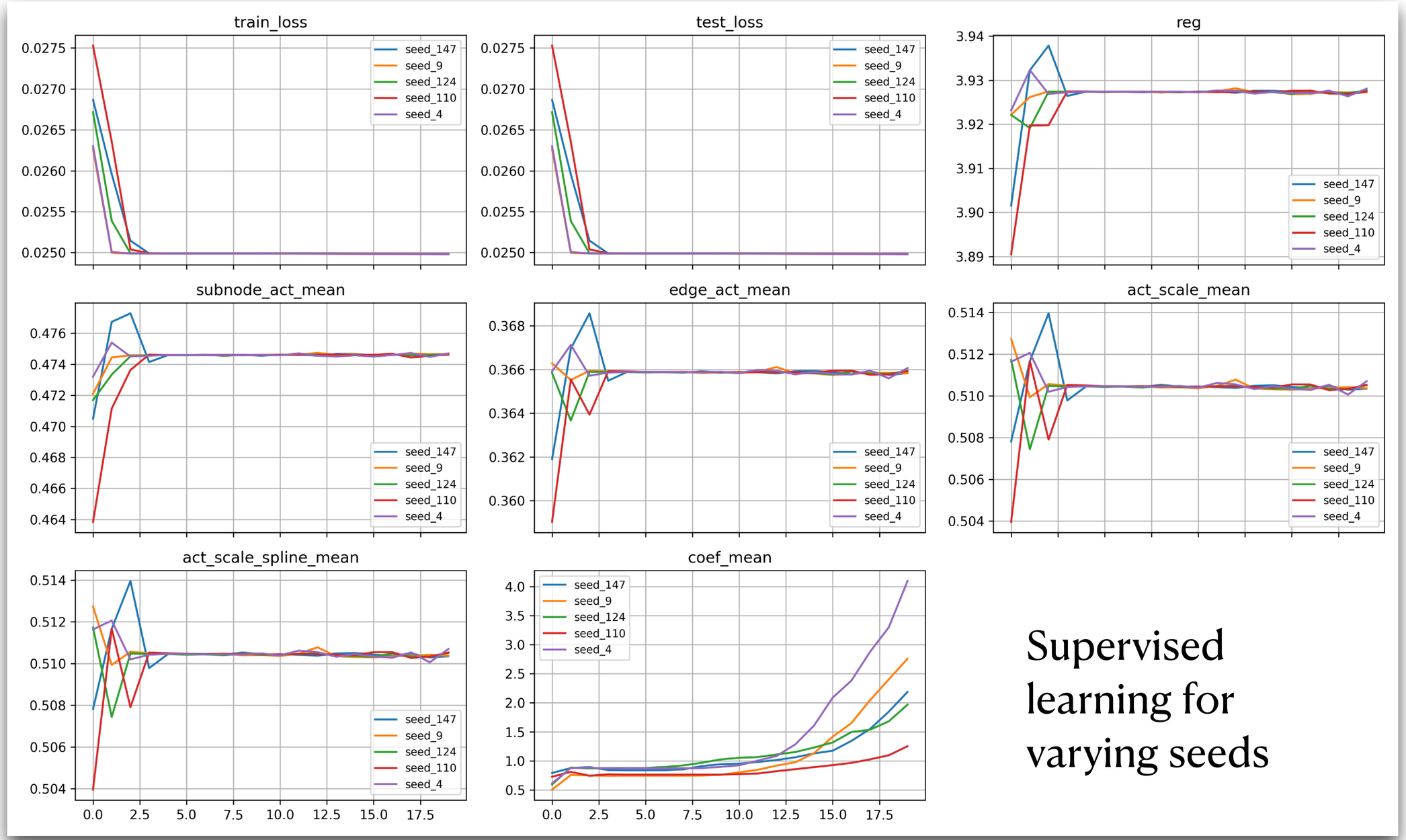

train_loss
test_loss
reg
subnode_act_mean
edge_act_mean
act_scale_mean
act_scale_spline_mean
coef_mean
scale_base_sigma_0.5
scale_base_sigma_3
scale_base_sigma_2
scale_base_sigma_0
scale_base_sigma_1

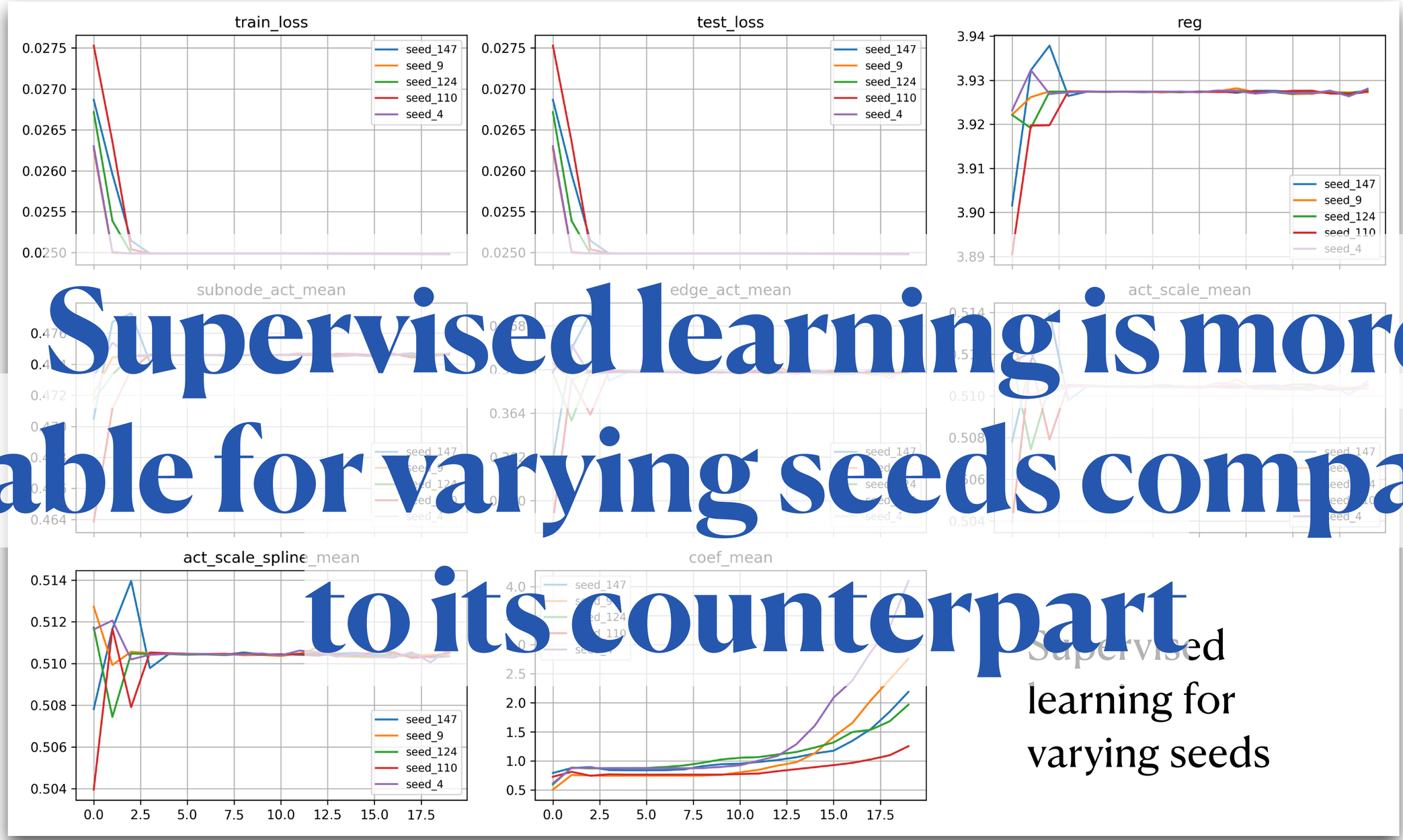


Some internal parameters explain each other



Unsupervised learning for varying seeds


train_loss
test_loss
reg
subnode_act_mean
edge_act_mean
act_scale_mean
act_scale_spline_mean
coef_mean
seed_147
seed_9
seed_124
seed_110
seed_4
Supervised learning for varying seeds



Supervised learning is more stable for varying seeds compared to its counterpart

Supervised learning for varying seeds

Next Steps

- Performing topology experiments with the following set-up is quite difficult
    - Settings for topology are far too arbitrary — as long as the input dimension and output dimension follow protocol, the hidden dimensions can be anything!
- Symbolification is a step after the learning process to gain interpretability of our learned model — an optional step
    - In the process of setting up an experiment for this — *how much noise can we add to a function before the symbolification process begins to wane in performance?*
- For HPO, we need some form of quantitative signal from the pipeline to optimize (specifically for unsupervised learning)

Relevant Papers

A Preliminary Study on Continual Learning in Computer Vision Using Kolmogorov-Arnold Networks

Deterministic Global Optimization over trained Kolmogorov Arnold Networks

A Comprehensive Survey on Kolmogorov Arnold Networks (KAN)

Agenda

06/06/25

Reminder: Unsupervised Learning Setup

Post-Shock

Graph Analysis

[NEW] Experiment Setup

Results

Next Steps

Unsupervised Learning Setup

Recall...

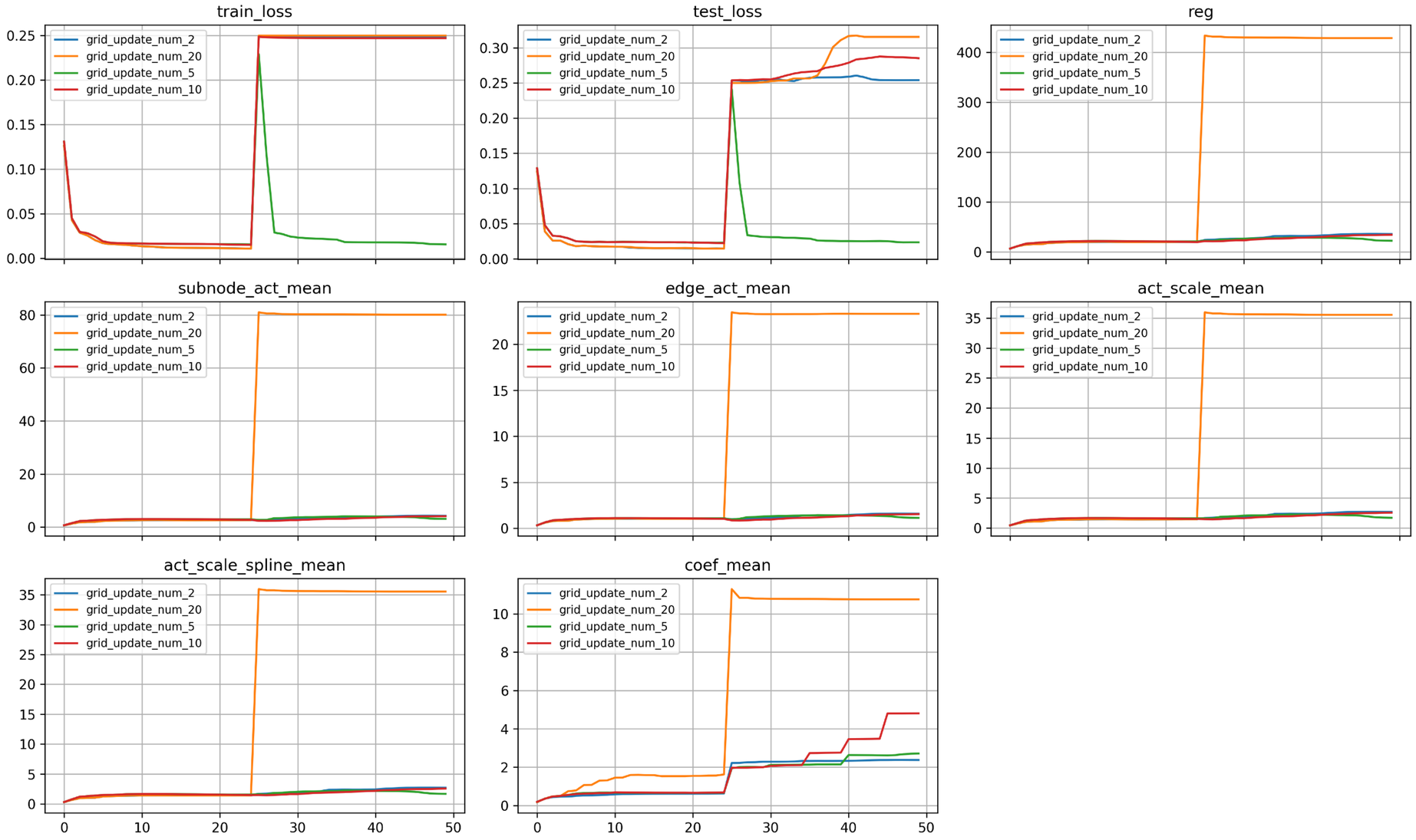
- Unsupervised learning involved
    - **6** input variables $x_1, x_2, x_3, x_4, x_5, x_6$
    - **3** relation sets $(x_1, x_2, x_3), (x_4, x_5), (x_6)$
- Task was designed as a classification problem where
    - Data abiding by any of the relation sets was considered *positive* samples
    - Data randomly permuted across the features was considered *negative* samples
- The model then converges to a relation at a given run and random seed

Post-shock Graph Analysis

- One primary goal was to improve the original KAN design for unsupervised learning
- We noticed that extracting/learning the right relations proved to be **hard**
- Recently, an experiment involving shock was performed
    - Randomly peturbe spline coefficients midway through training
    - Results for the unsupervised setting showed that certain hyperparameters (next slide) were useful in stabilizing the model post shock

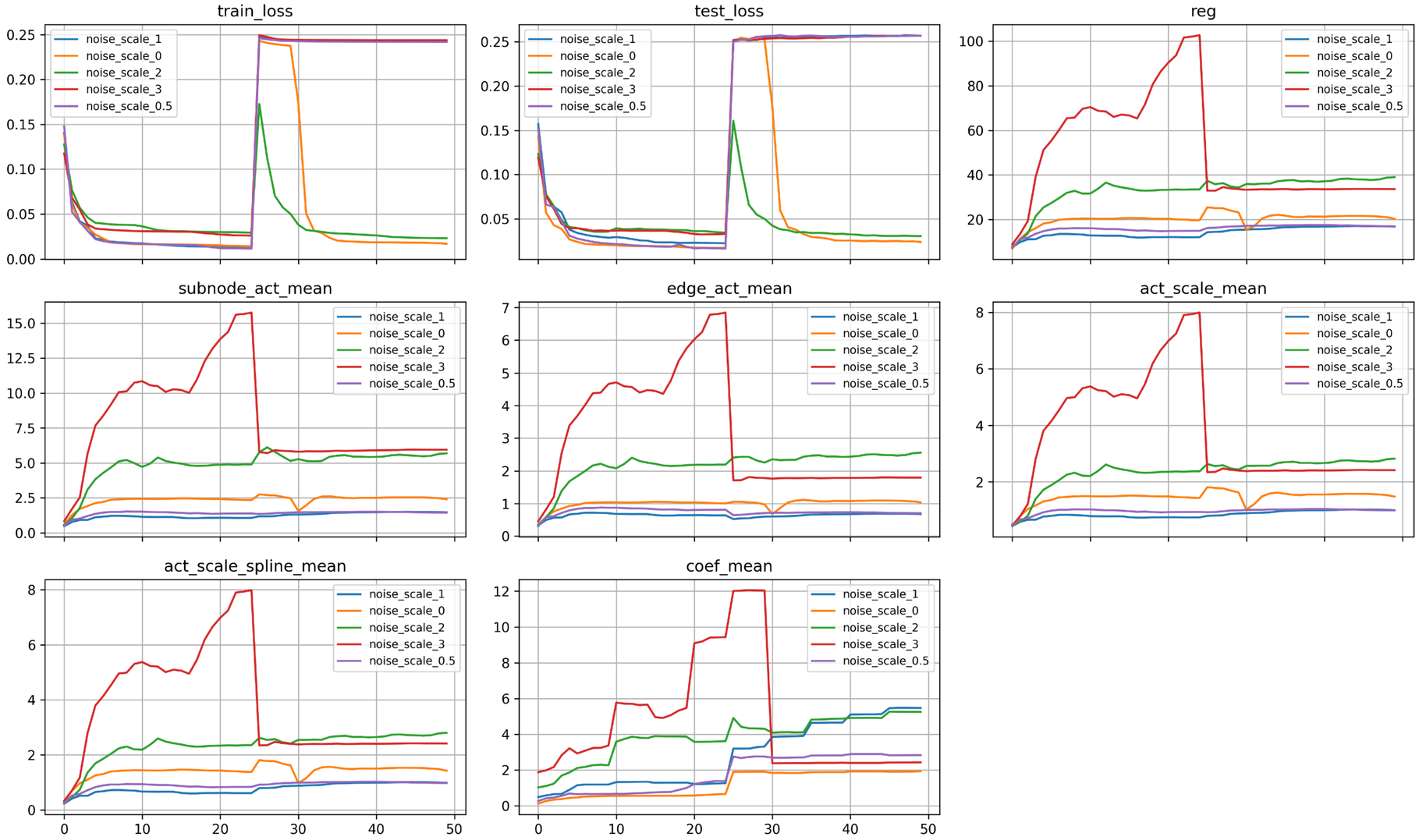
For a given set of random seeds, we rarely found the right relations

grid_update_num



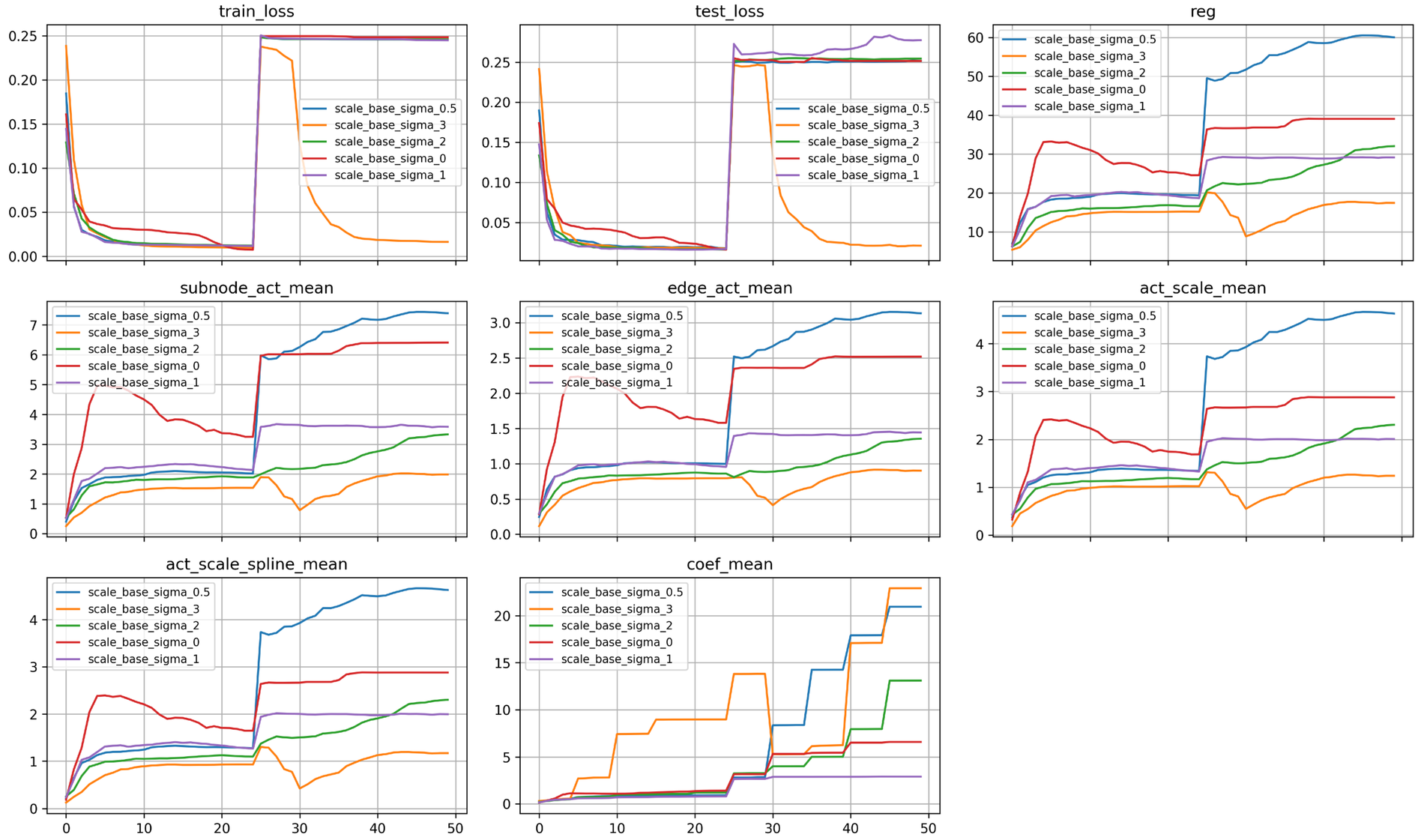
For grid_update_num=5, the model was able to stabilize post shock

noise_scale



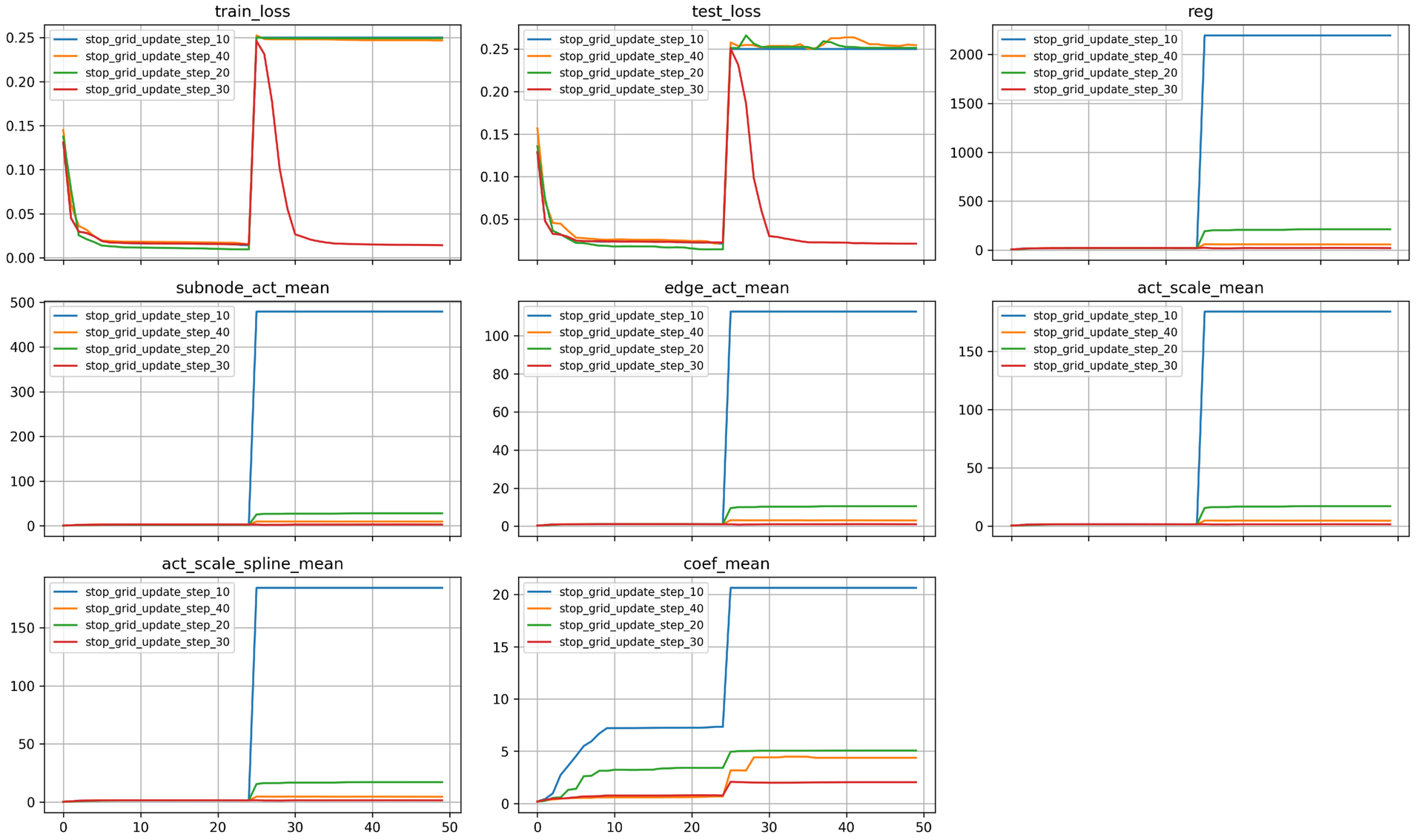
For noise_scale=2, the model was able to stabilize post shock

scale_base_sigma



For scale_base_sigma=3, the model was able to stabilize post shock

stop_grid_update_step



For stop_grid_update_step=30, the model is able to stabilize post shock

[NEW] Experiment Setup

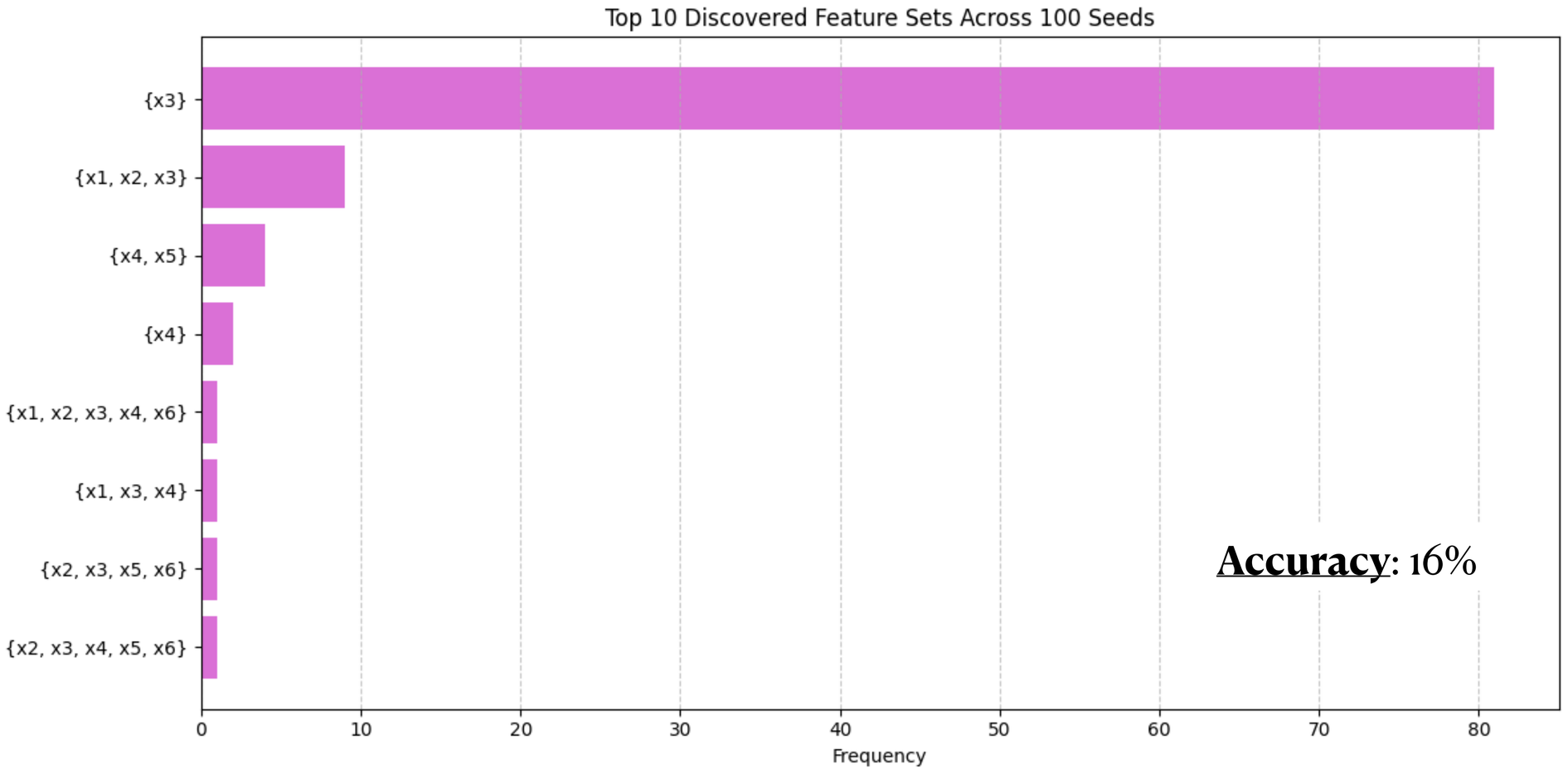
- This week, I mainly focused on one key question: *are the shock results helpful for our analysis and improving the KAN architecture?*
- Experiment design
  - Iterate over a set of random seeds to find the number of times a right relation is produced
  - Heuristic used for extracting relation — edge scores
    - activation functions are on the edge so these scores also tell us the significance of a given activation function

Pre-shock
Same as old code

Post-shock
Utilizes the hyperparameter settings that could stabilize the model after shock

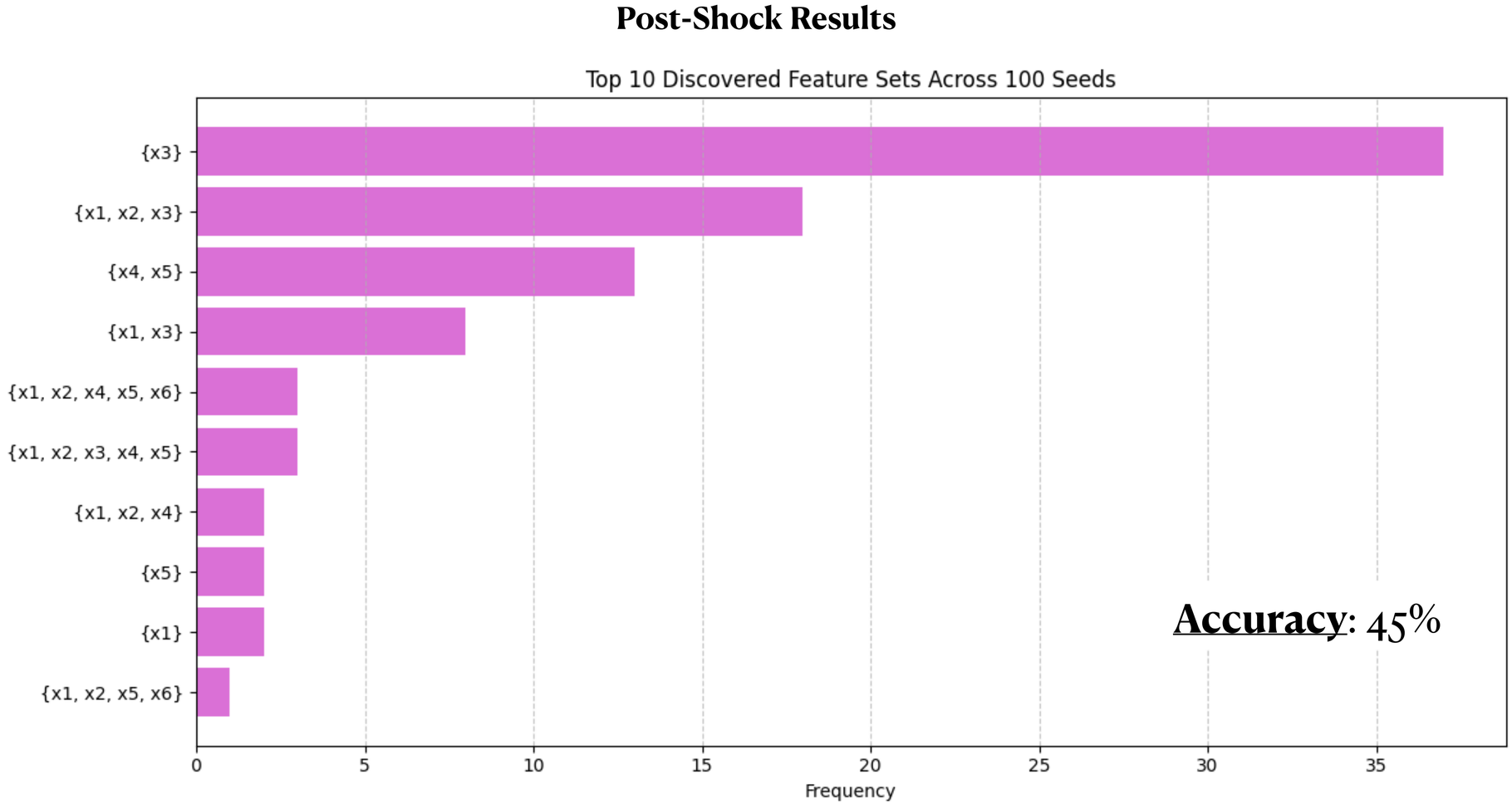
Results

Pre-Shock Results



Right relations were found rarely; model converges to {x3} more than all the other relations **combined**

A huge boost in accuracy — this definitely means we are heading in the right direction!



Right relations were found for a large number of seeds

Next Steps

- Performing topology experiments with the following set-up is quite difficult
    - Settings for topology are far too arbitrary — as long as the input dimension and output dimension follow protocol, the hidden dimensions can be anything!
- Symbolification is a step after the learning process to gain interpretability of our learned model — an optional step
    - In the process of setting up an experiment for this — *how much noise can we add to a function before the symbolification process begins to wane in performance?*
- Can our new heuristic be scaled in a non-qualitative fashion?
    - i.e. Right now, we look at the graphs and decide the parameter settings
    - Is it possible to evaluate — quantitatively — if a hyperparameter is able to stabilize the model after shock?
- Finish the tables!
- Add more documentation for source code instrumentation

Still to do from last week

Added new

Tables

HPO Parameter	Purpose	Where Used	Observed Impact	Advisable RANGES	GUIDELINES	HEURISTICS
Topology	Defines the architecture of the KAN	Model initialization width	Affects capacity, overfitting, and interpretability More complex networks are harder to train (more latency)	As long as input dimension and output dimension is correct, hidden layer dimension can be anything.	Start small and simple (maybe 2 layers at most), increase if underfitting; deeper for complex data	
Coefficient Penalty	Penalizes the magnitude of spline coefficients to control complexity and encourage simpler, more linear functions	Regularization term used in the fitting process. lamb_coef	TASK 1A: No observed impact TASK 1B: No observed mpact TASK 2: No observed impact	Any float	Start small and increase as needed, high values regularize more.	
Entropy Penalty	"entropy term encourages the learned activation functions (splines) to be either sparse (few are active) or diverse (different splines are used for different edges), preventing" collapse to trivial or redundant solutions	Regulariztion term used in the fitting process. lamb_entropy	TASK 1A: No observed impact TASK 1B: No observed mpact TASK 2: No observed impact	TASK 1A: Any float TASK 1B: Any float TASK 2: Any float		
Penalty Strength	Controls magnitude of regularization (L1/L2 on coefs, smoothness, etc.)	Regularization term used in the fitting process. lamb	TASK 1A: high penalty strength discourages learning TASK 1B: high penalty strength discourages learning TASK 2: high penalty strength discourages learning	TASK 1: 0-0.1 TASK 2A: 0-0.1 TASK 2B: 0-0.1	Ensure to keep the penalty strength as low as possible — high values discourages learning and thereby decreases model accuracy	To judge we can look whether the coefficients of the B-spline basis function are changing and we can observed the accuracy.
Grid Number	Number of intervals for spline basis. Higher grid values indicate more control points in the B-spline and thus directly corresponds to higher complexity	Model initialization grid	TASK 1A: Model starts to learn complex activation functions. Too high values of grid is detrimental to accuracy. TASK 1B: Model starts to learn complex activation functions. Too high values of grid is detrimental to accuracy. TASK 2: Model starts to learn complex activation functions. For high values of grid, model needs to run for more epochs.	TASK 1A: 3-5 TASK 1B: 3-5 TASK 2: 3-5	Keep grid number low and then increase as needed.	We can look at accuracy of the model as well as the complexity of the functions learned. Too high values of grid will cause overfitting and too low values of grid will cause underfitting.
Piecewise Polynomial Order	Polynomial order of the B-spline basis functions. Higher polynomial orders indicate more complex basis functions and thus overall higher complexity.	Model initialization. k	TASK 1A: too low or too high order will impacts accuracy of the model negatively and can explode the coefficients TASK 1B: too low settings of the order negatively impacts the model accuracy TASK 2: too low settings of the order negatively impacts the model accuracy	TASK 1A: 3-5 TASK 1B: 3-5 TASK 2: 3-5	Keep k=3 in the beginning and increases as needed.	We can look at accuracy of the model as well as the complexity of the functions learned. Too high values of k will cause overfitting and too low values of k will cause underfitting.
Random Seeds	Randomly splitting the dataset into a training set and testing set. Introducing random noise into the coefficients of the splines and is scaled by noise_scale. Introducing variability in the initialization of scale_base	Model initialization. seed	TASK 1A: Not much observed impact TASK 1B: Not much observed impact, but the learning of the coefficients is altered TASK 2:different seeds have different convergences for the learned coefficient	TASK 1A: any TASK 1B: any TASK 2: any	TASK 1A: you can just choose any value for the seed TASK 1B: you can just choose any value for the seed TASK 2: different seeds are meant to produce different relations so choose a set of seeds to run the model	TASK 1A: accuracy of the model TASK 1B: accuracy of the model TASK 2: Check the number of correct relations (consistency) and also the number of different relations (coverage) for a given set of seeds

HPO Parameter	Purpose	Where Used	Observed Impact	Advisable RANGES	GUIDELINES	HEURISTICS
scale_base_mu	determines the mean of the initial scaling factor applied to the residual (base) function	Model initialization **`scale_base_mu`**	TASK 1A: n observed effect TASK 1B: no observed effect TASK 2: large values are detrimental to learning process and hinders model accuracy	TASK 1A: any float TASK 1B: any float TASK 2:	TASK 1A: TASK 1B: TASK 2:	TASK 1A: TASK 1B: TASK 2:
scale_base_sigma	determines the standard deviation of the initial scaling factor applied to the residual (base) function	Model initialization **`scale_base_sigma`**	TASK 1A: setting sigma too high will amplify model loss TASK 1B: mean of spline coefficients grow with sigma setting TASK 2: choosing sigma=0 will cause exploding regularization terms and will also explode the internal parameters	TASK 1A: <= 1 TASK 1B: TASK 2:	TASK 1A: TASK 1B: TASK 2:	TASK 1A: TASK 1B: TASK 2:
noise_scale	magnitude of random noise injected into the spline parameters at initialization	Model initialization **`noise_scale`**	TASK 1A: noise_scale=0 works best TASK 1B: learning of spline coefficients begin to become unstable with large values of noise_scale TASK 2: too much noise explodes regularization and also the internal parameters	TASK 1A: 0<noise_scale<2 TASK 1B: TASK 2:	TASK 1A: TASK 1B: TASK 2:	TASK 1A: TASK 1B: TASK 2:
Learning Rate	specifies the learning rate of the optimizer used (LBFGS or Adam)	Set during the fitting process in accordance with a selected optimizer. **`lr`**	TASK 1A: low learning rates work best and show better loss convergence TASK 1B: mean of spline coefficients grow with learning rate setting TASK 2: carefully calibrated learning rate allows for smooth learning curves	TASK 1A: <= 0.01 TASK 1B: TASK 2:	TASK 1A: TASK 1B: TASK 2:	TASK 1A: TASK 1B: TASK 2:
Batch size	specifies the number of training samples processed together in one forward and backward pass during model training	Set during the fitting process. **`batch`**	TASK 1A: setting a batch size not equal to the full dataset has significant and unpredictable impact on the learning process TASK 1B: setting a batch size not equal to the full dataset has significant and unpredictable impact on the learning process TASK 2: setting a batch size not equal to the full dataset has significant and unpredictable impact on the learning process	TASK 1A: -1 TASK 1B: TASK 2:	TASK 1A: TASK 1B: TASK 2:	TASK 1A: TASK 1B: TASK 2:
L1 Regularization (Lasso) Penalty	controls the strength of L1 regularization applied to the model's parameters during training	Regularization term used in the fitting process. **`lamb_l1`**	TASK 1A: no observed impact TASK 1B: no observed impact TASK 2: no observed impact	TASK 1A: any float TASK 1B: any float TASK 2: any float	TASK 1A: TASK 1B: TASK 2:	TASK 1A: TASK 1B: TASK 2:
grid_eps	controls how the grid for spline basis functions is constructed—specifically, how adaptive or uniform the grid is relative to the data distribution *"when grid_eps = 1, the grid is uniform; when grid_eps = 0, the grid is partitioned using percentiles of samples. 0 < grid_eps < 1 interpolates between the two extremes"*	Model initialization **`grid_eps`**	TASK 1A: choosing a setting less than or equal to 1.0 works best TASK 1B: no observed impact TASK 2: choosing a setting that allows for interpolation between the two extremes works best (0< grid_eps < 1)	TASK 1A: <=1.0 TASK 1B: <= 1.0 TASK 2: 0< grid_eps < 1	TASK 1A: TASK 1B: TASK 2:	TASK 1A: TASK 1B: TASK 2: