Hyperparameter Optimization

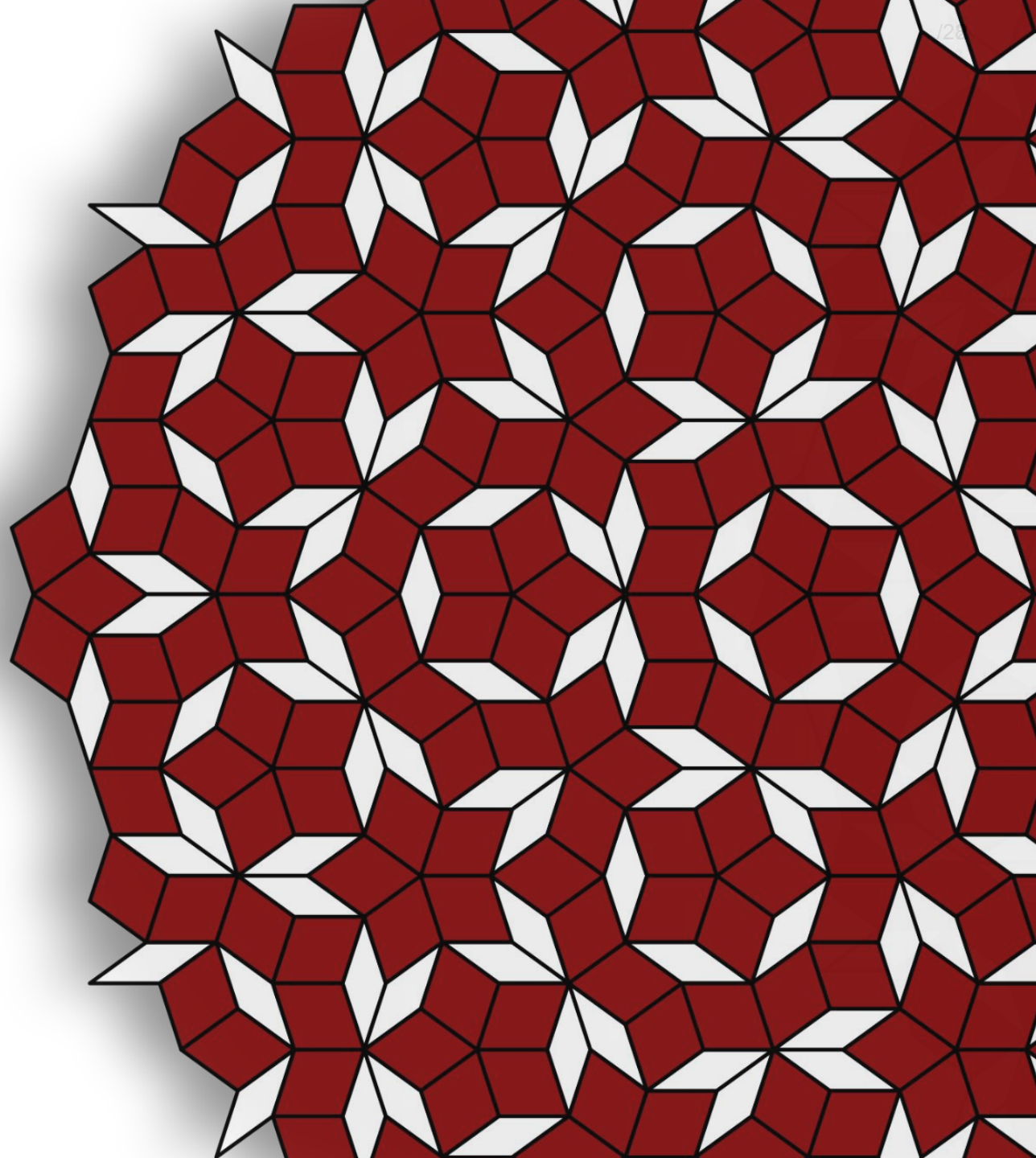
Beating Grid- and Randomsearch with alternative methods

Christian L. H. Rasmussen,
Jakob B. Frederiksen, Jonathan O. Melcher,
Rasmus A. Nielsen & Sune Halkjær

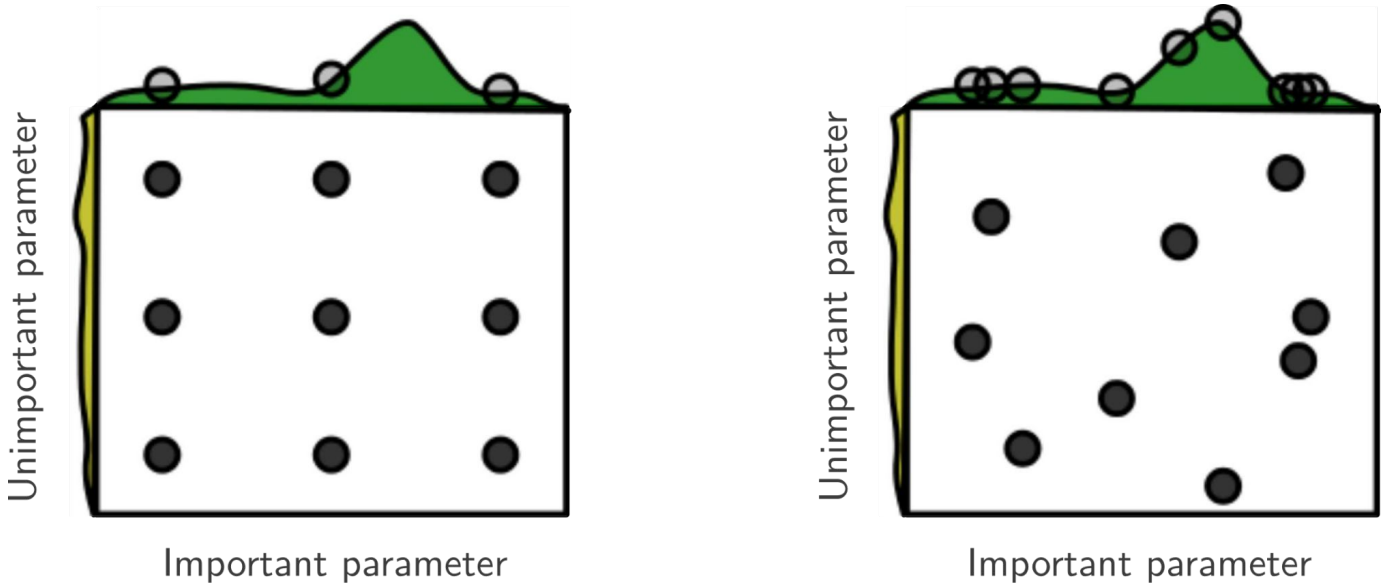
UNIVERSITY OF COPENHAGEN

Outline

- Motivation
- New tilings
- Simulation setup
- Results: CIFAR-10,
  Bjet & Housing data
- Discussion and further work
- Advice: What should ***you*** do?



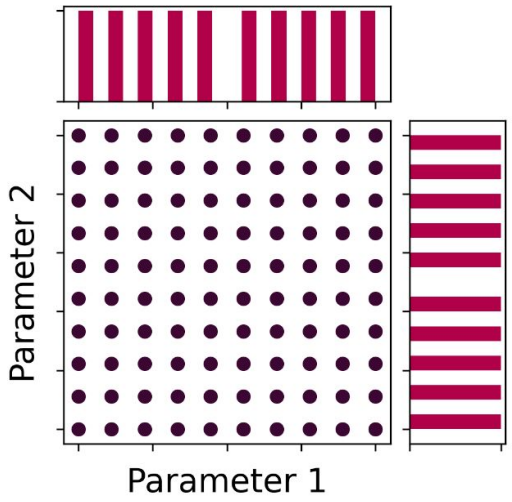
Motivation - Why not use Gridsearch and Randomsearch?



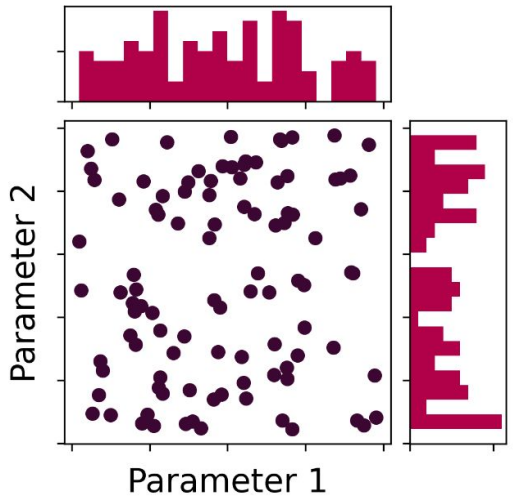
Bergstra, J. & Bengio, Y. Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research* **13**, 281–305 (2012).

Three new tilings

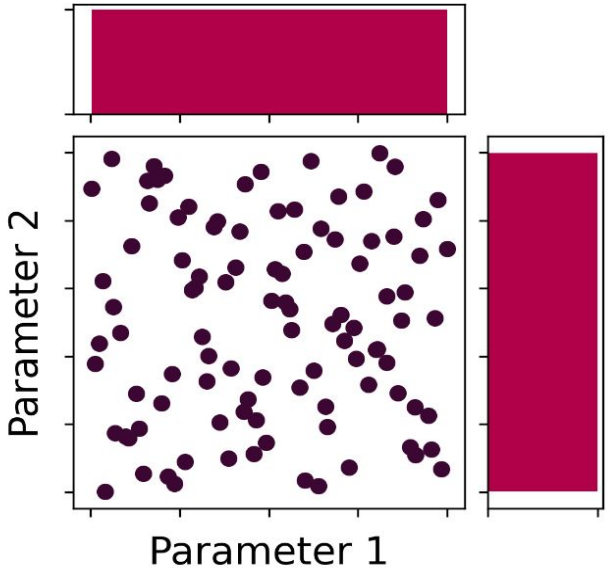
Regular



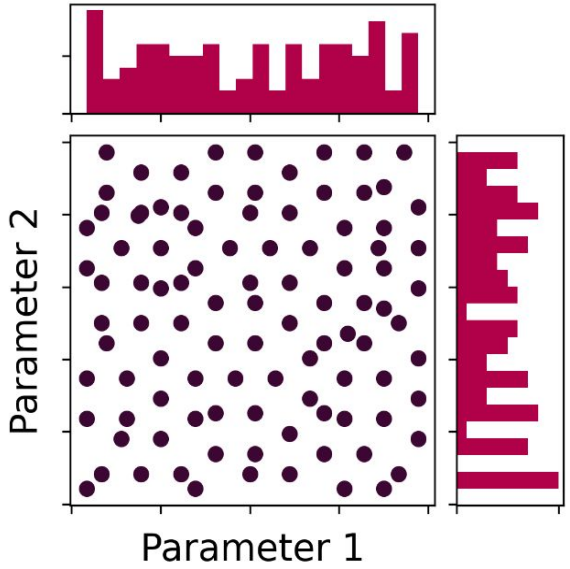
Random



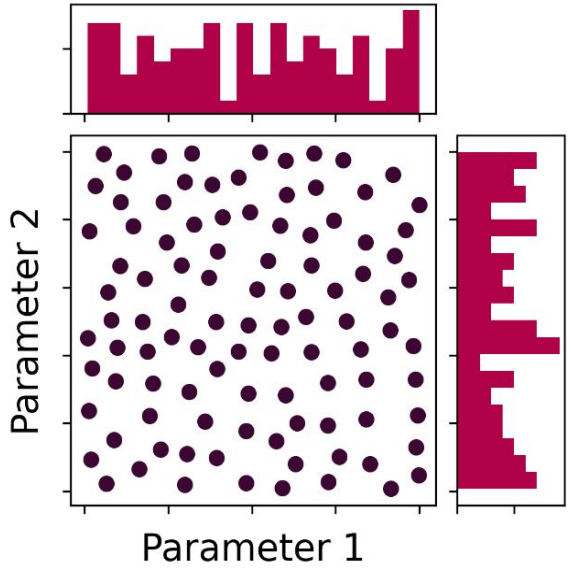
Latin hypercube sampling (LHS)



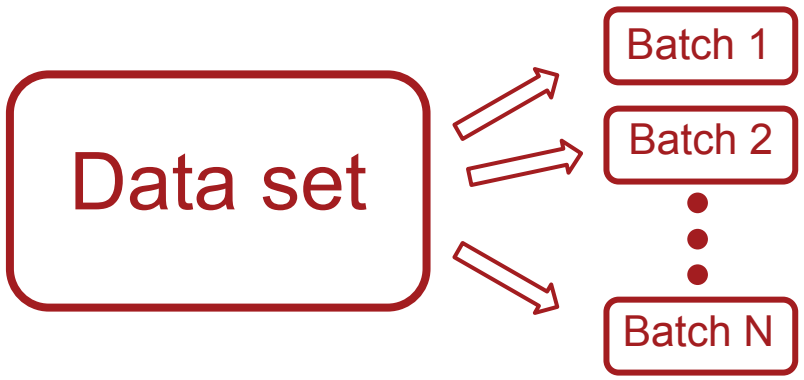
Hypertiling



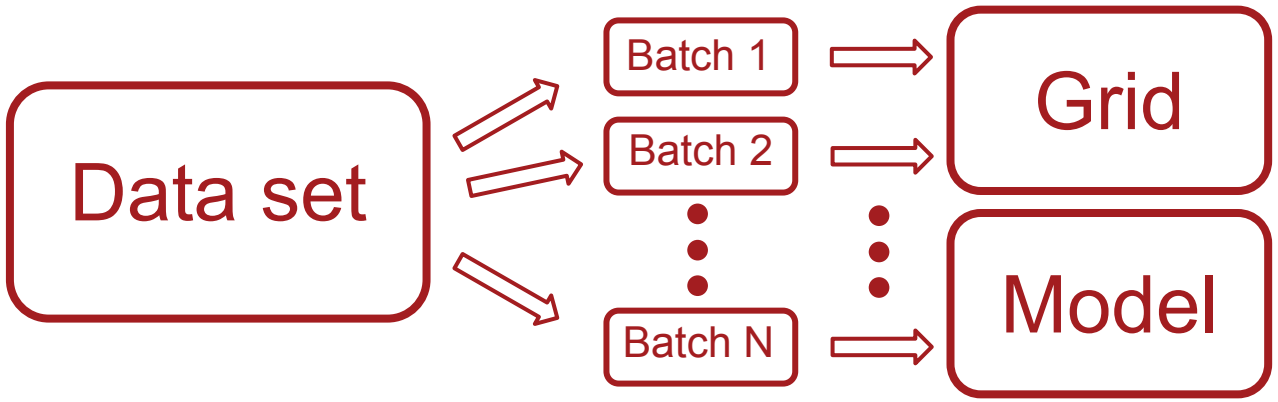
Random sequential addition (RSA)



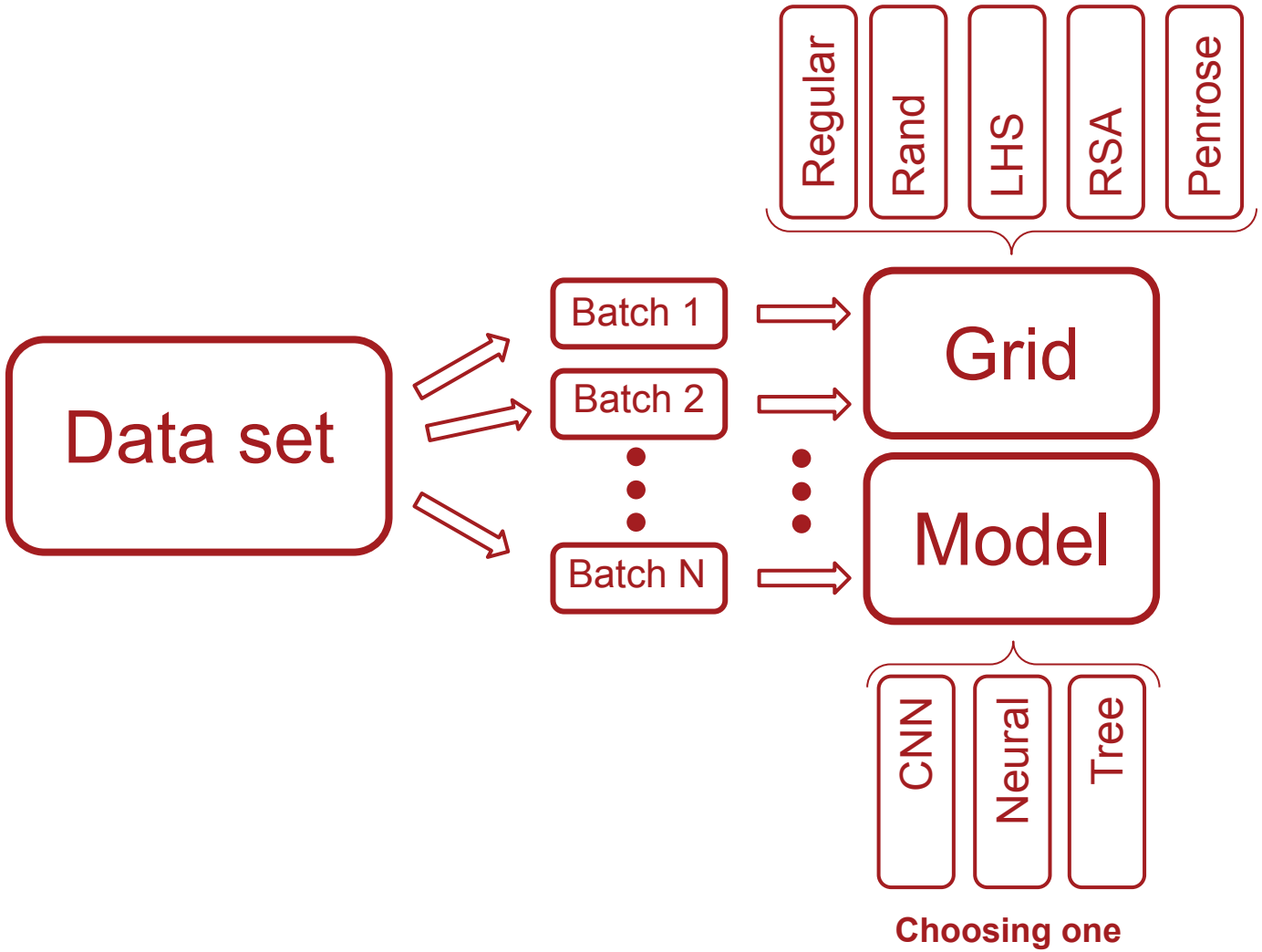
Simulation setup



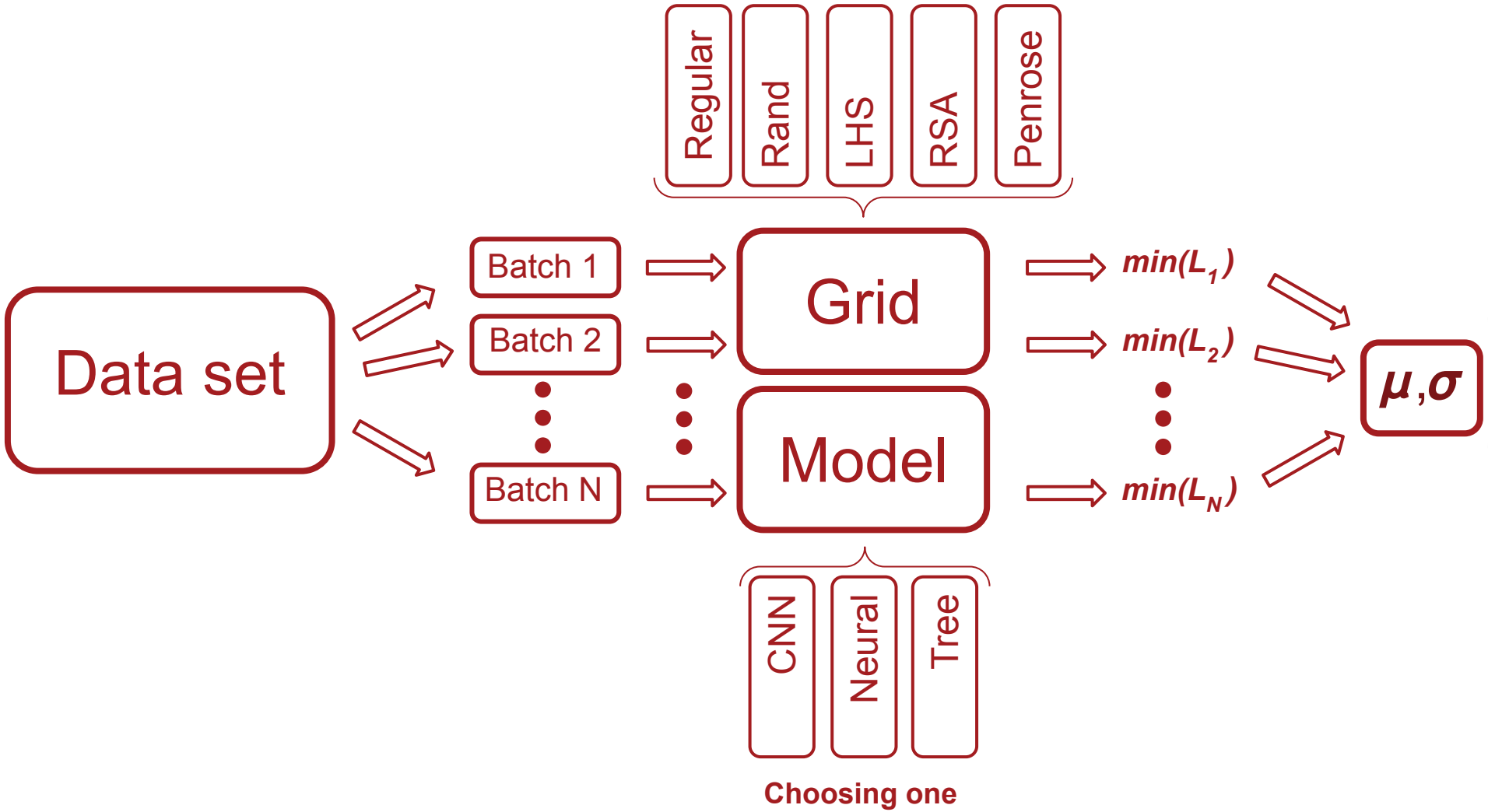
Simulation setup



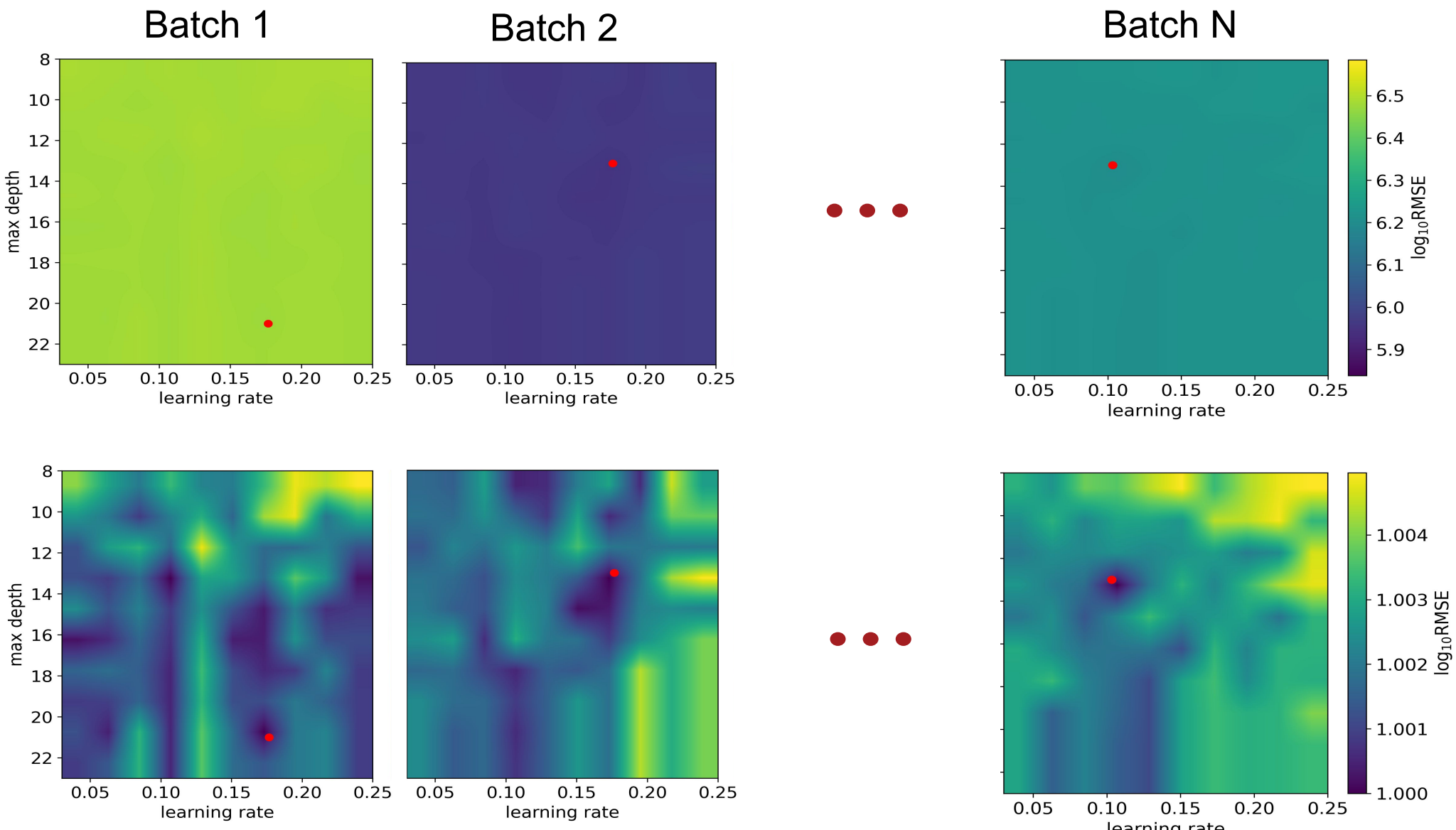
Simulation setup

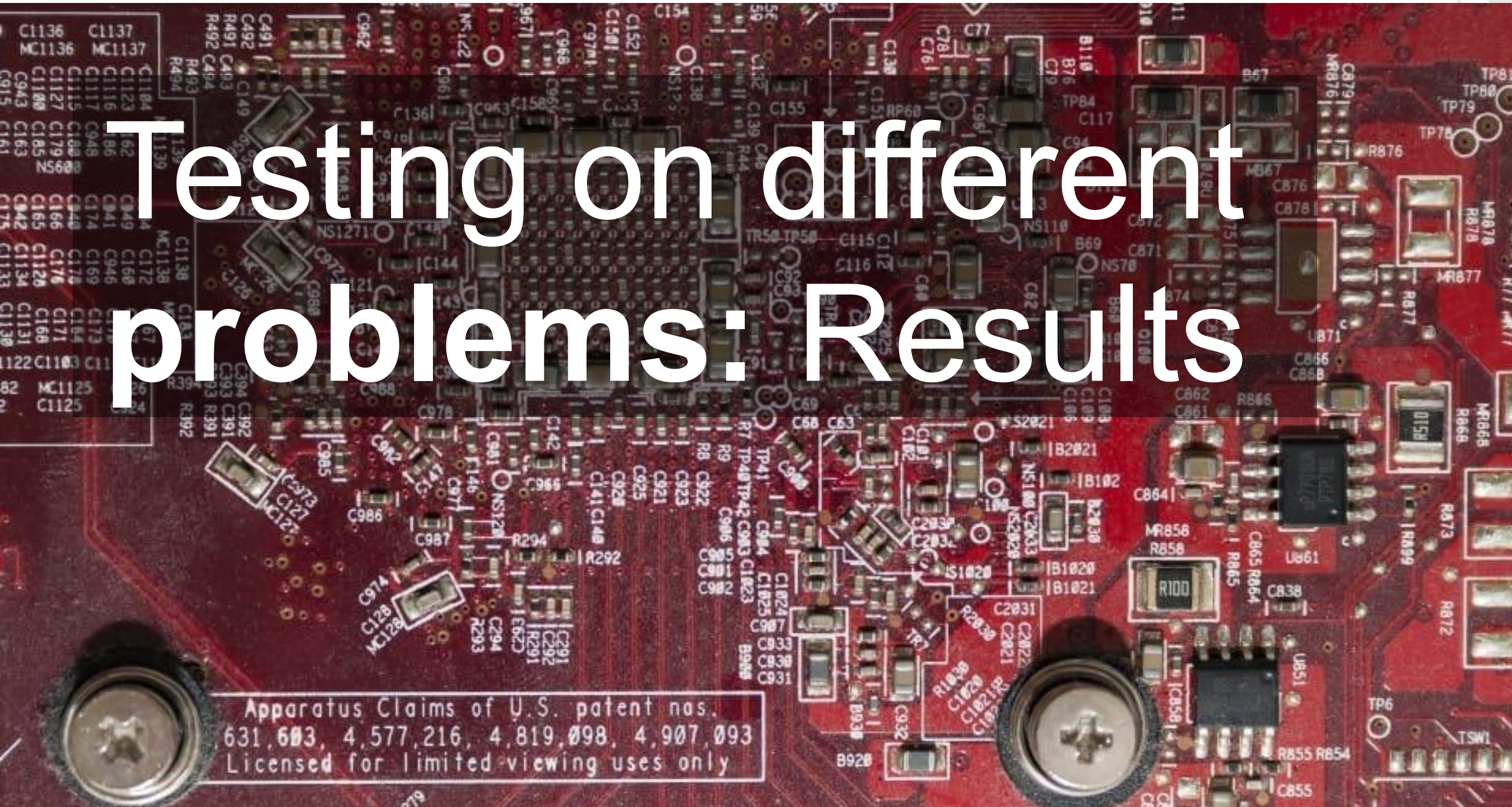


Simulation setup



Data processing



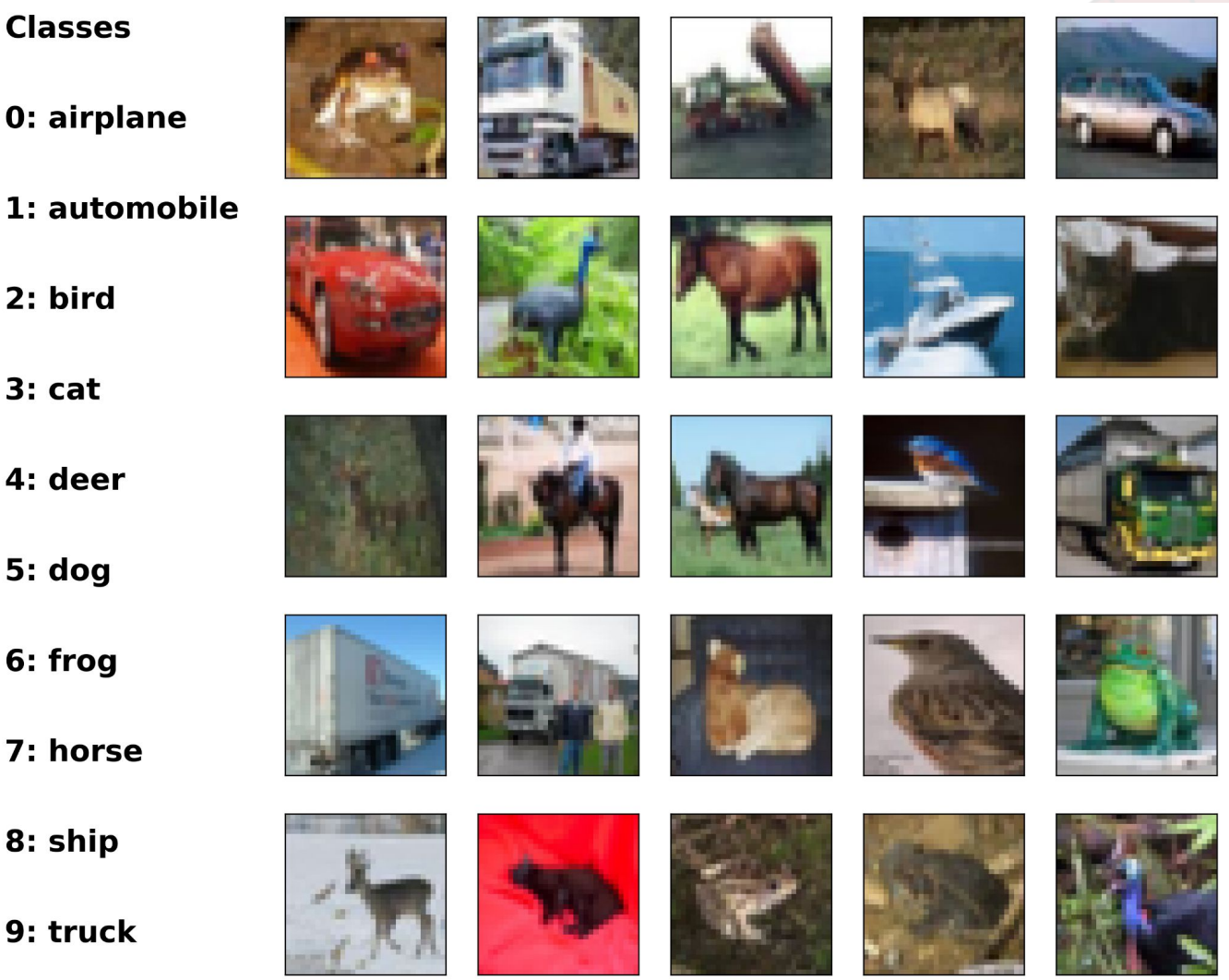


Testing on different problems: Results

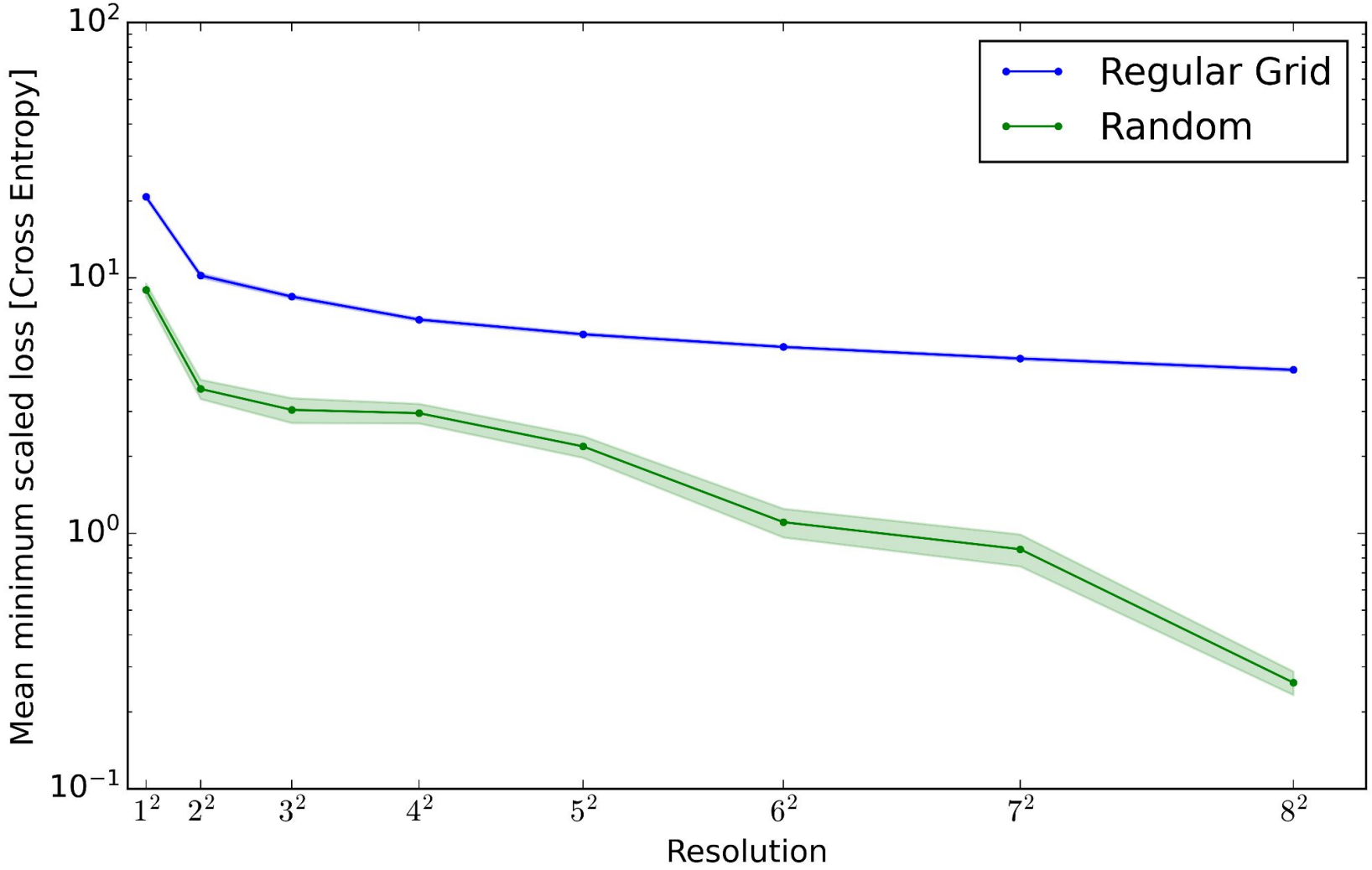
CIFAR10 with CNN

- DATA
  - 10 classes
  - 50.000 training
  - 10.000 testing

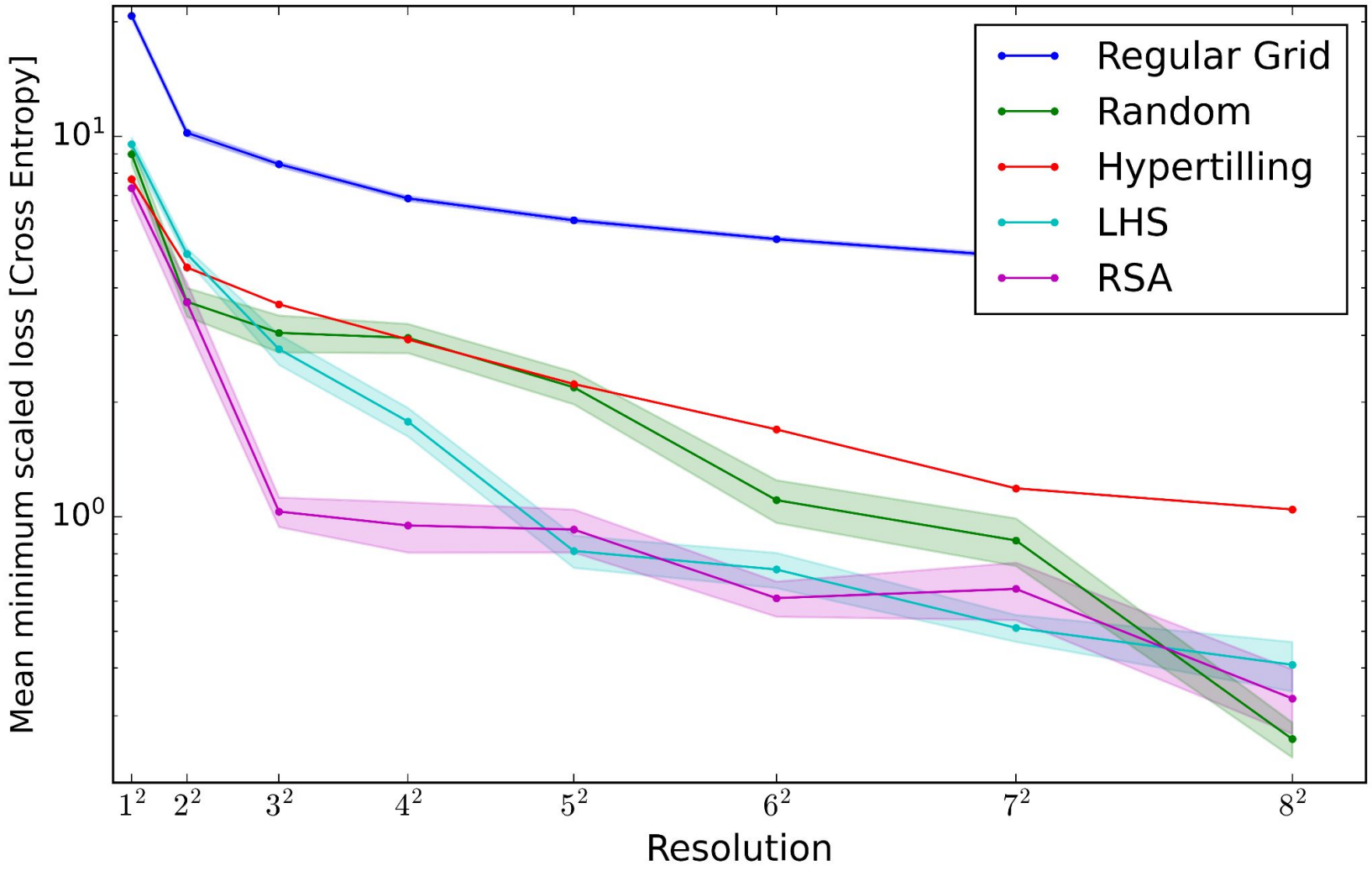
- MODEL
  - CNN, multiclass - classification
  - ResNet
    - New dense layer
  - Learning rate
  - Number of units in dense layer



CIFAR10 with CNN



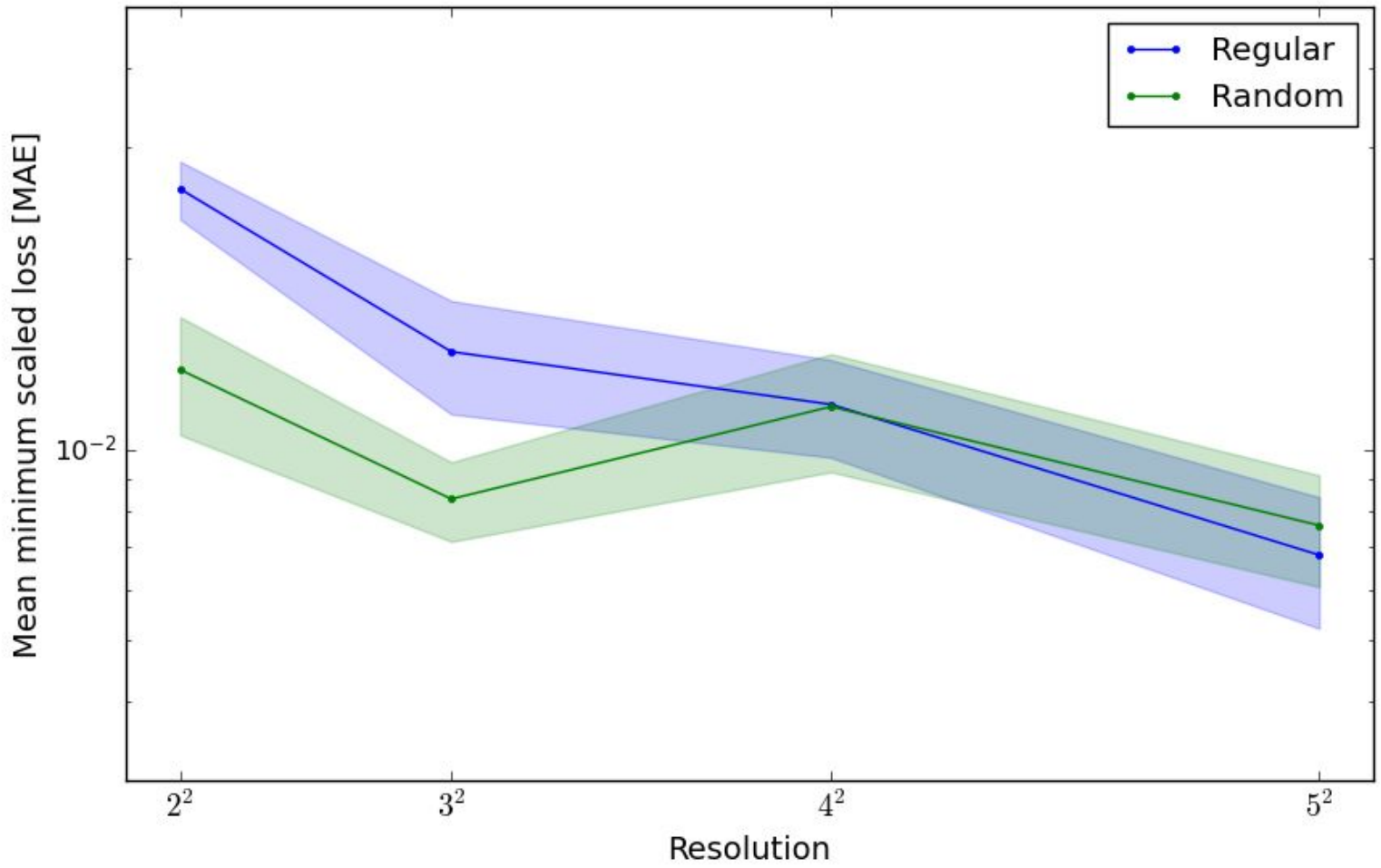
CIFAR10 with CNN



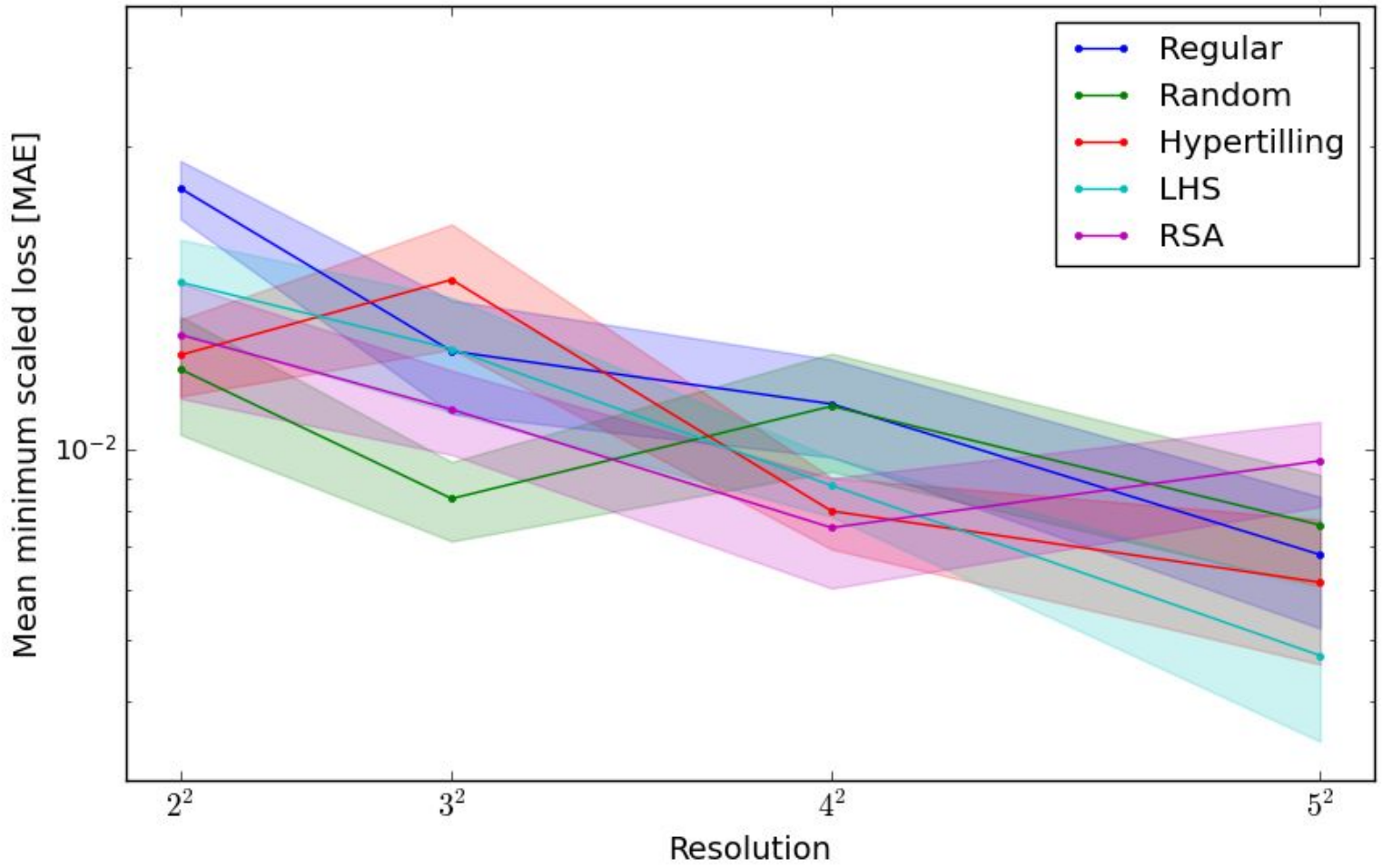
Aleph Bjet with Tensorflow NN

- MODEL
  - Regression
  - Hyperparameter
    - Learning rate
    - Epochs
- DATA
  - 10 features, 50.000 samples
  - 10 batches

Aleph Bjet with Tensorflow NN



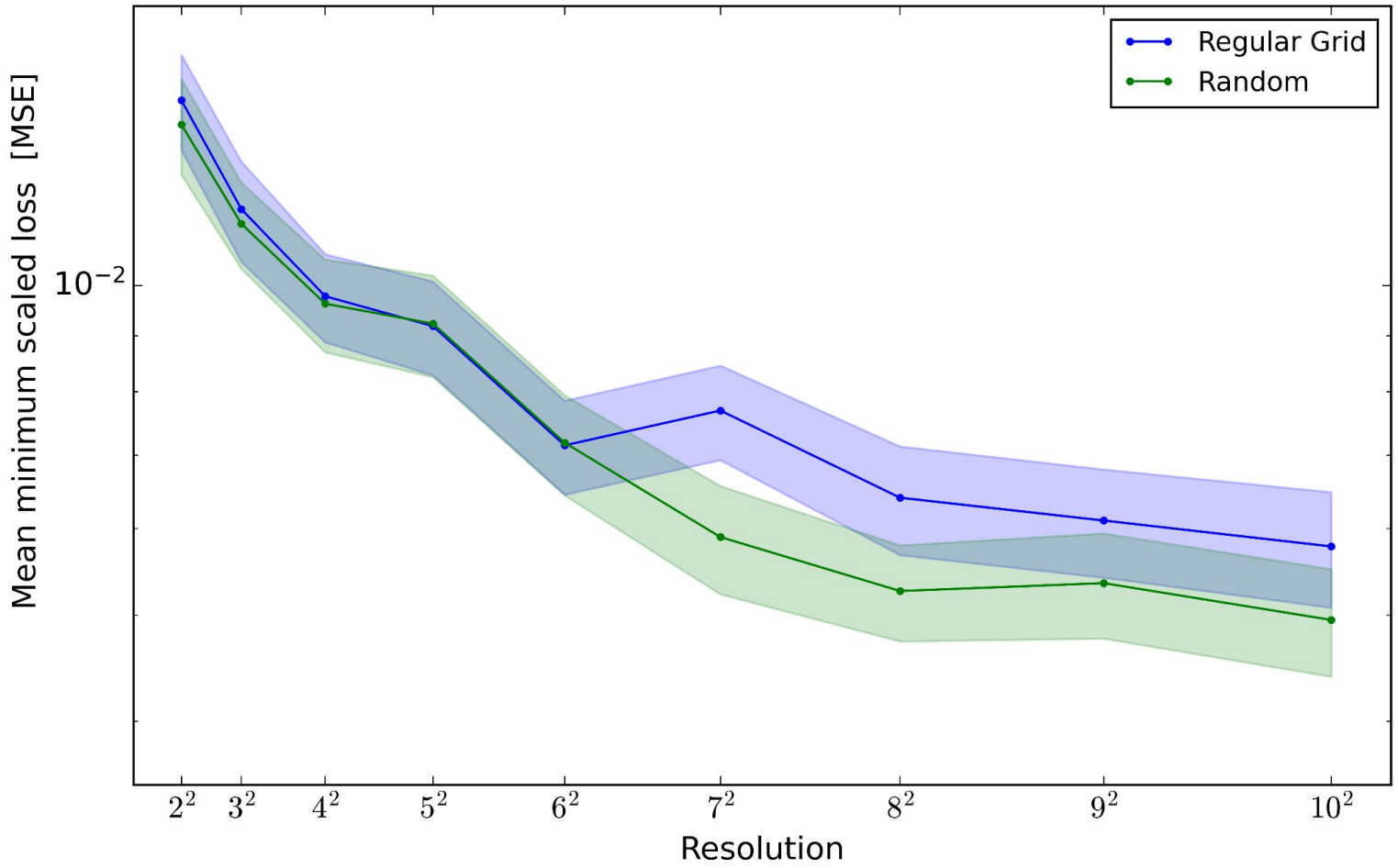
Aleph Bjet with Tensorflow NN



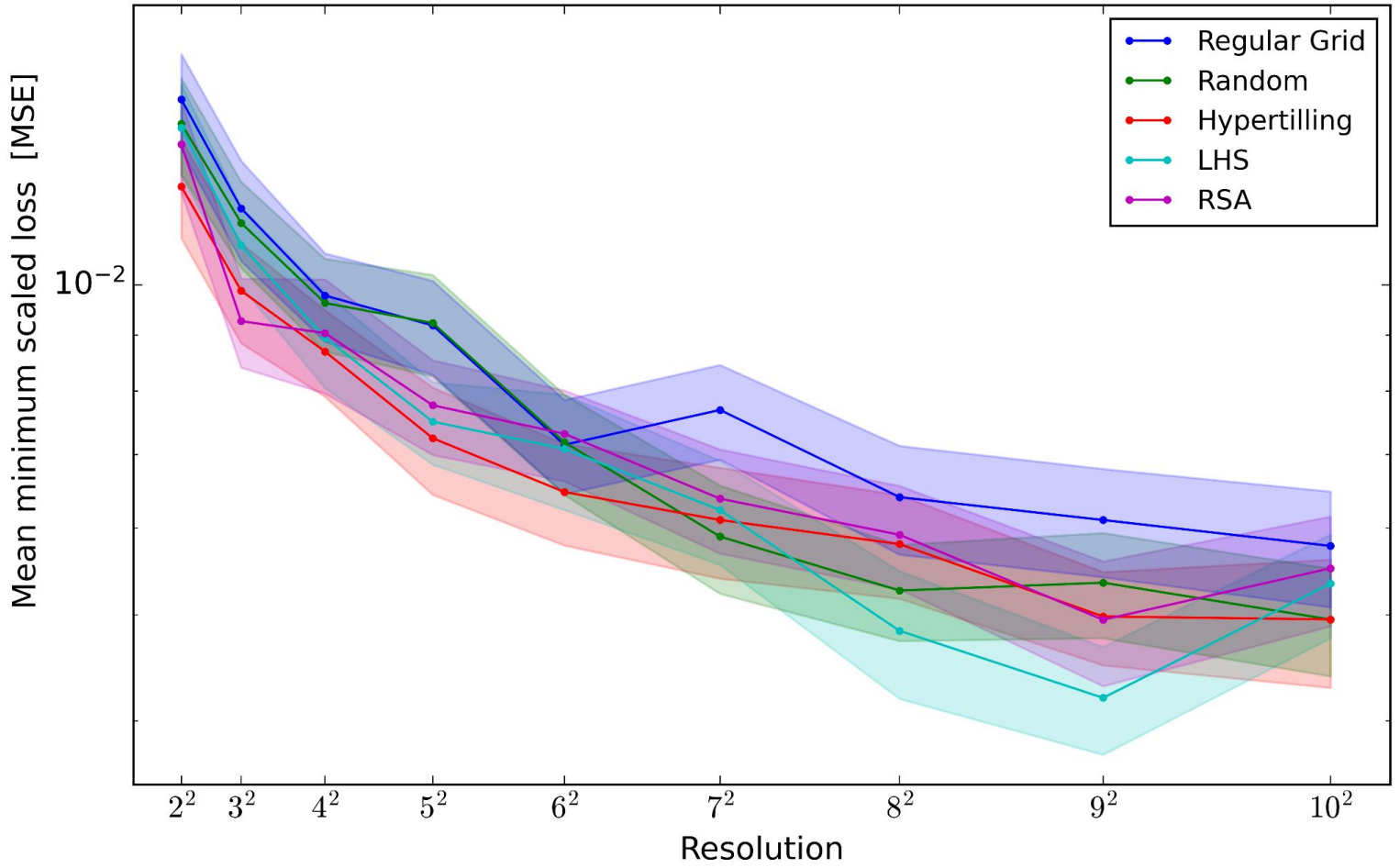
Housing price with LGBM tree based model

- MODEL
  - Regression LightGBM
    - Learning rate
    - Max depth
- DATA
  - ~500.000 data points
    - 44 Batches
    - ~10.000 pr. Batch
  - 118 features selected 20 using SHAP

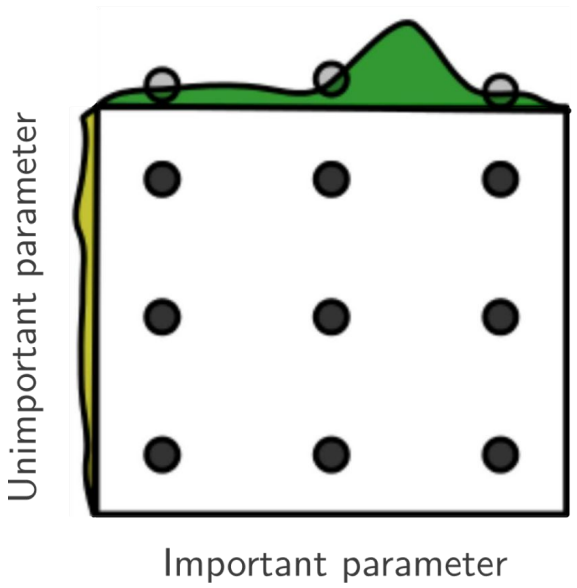
2D

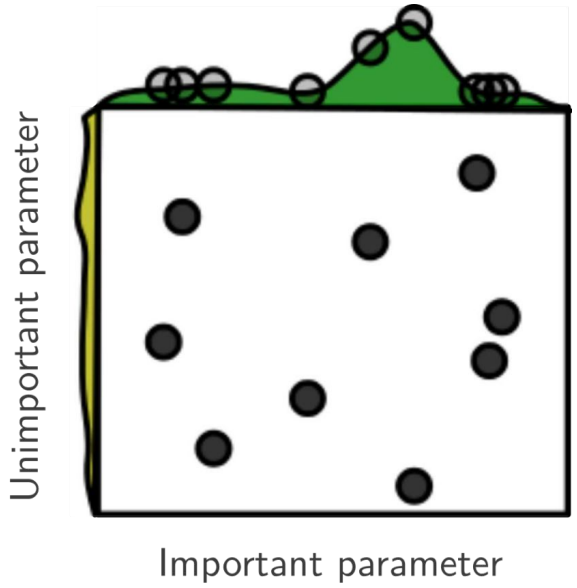

Regular Grid
Random
Mean minimum scaled loss [MSE]
10^{-2}
2^2 3^2 4^2 5^2 6^2 7^2 8^2 9^2 10^2
Resolution

2D


Regular Grid
Random
Hypertilling
LHS
RSA
Mean minimum scaled loss [MSE]
10^{-2}
2^2 3^2 4^2 5^2 6^2 7^2 8^2 9^2 10^2
Resolution

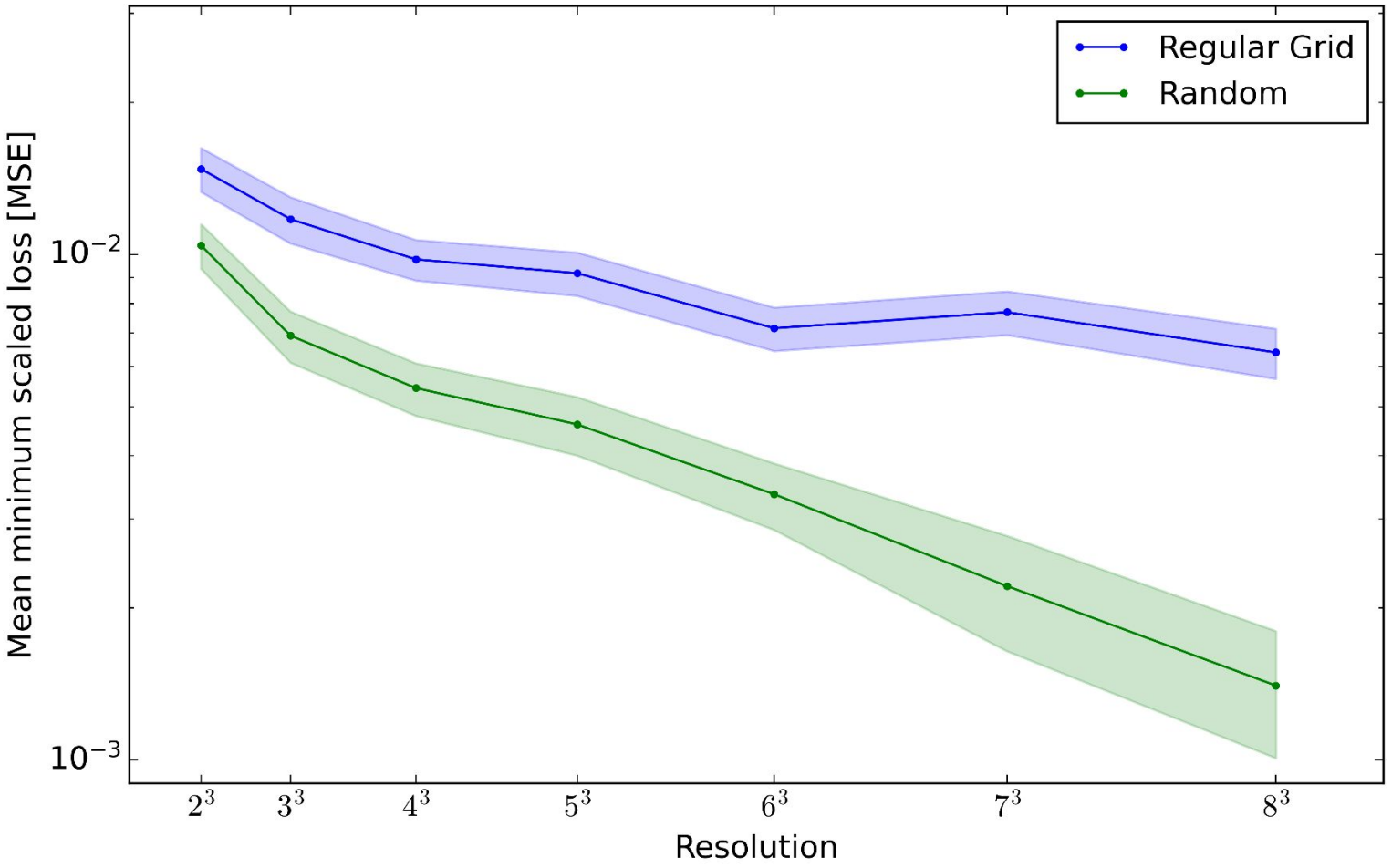
Adding a unimportant parameter test



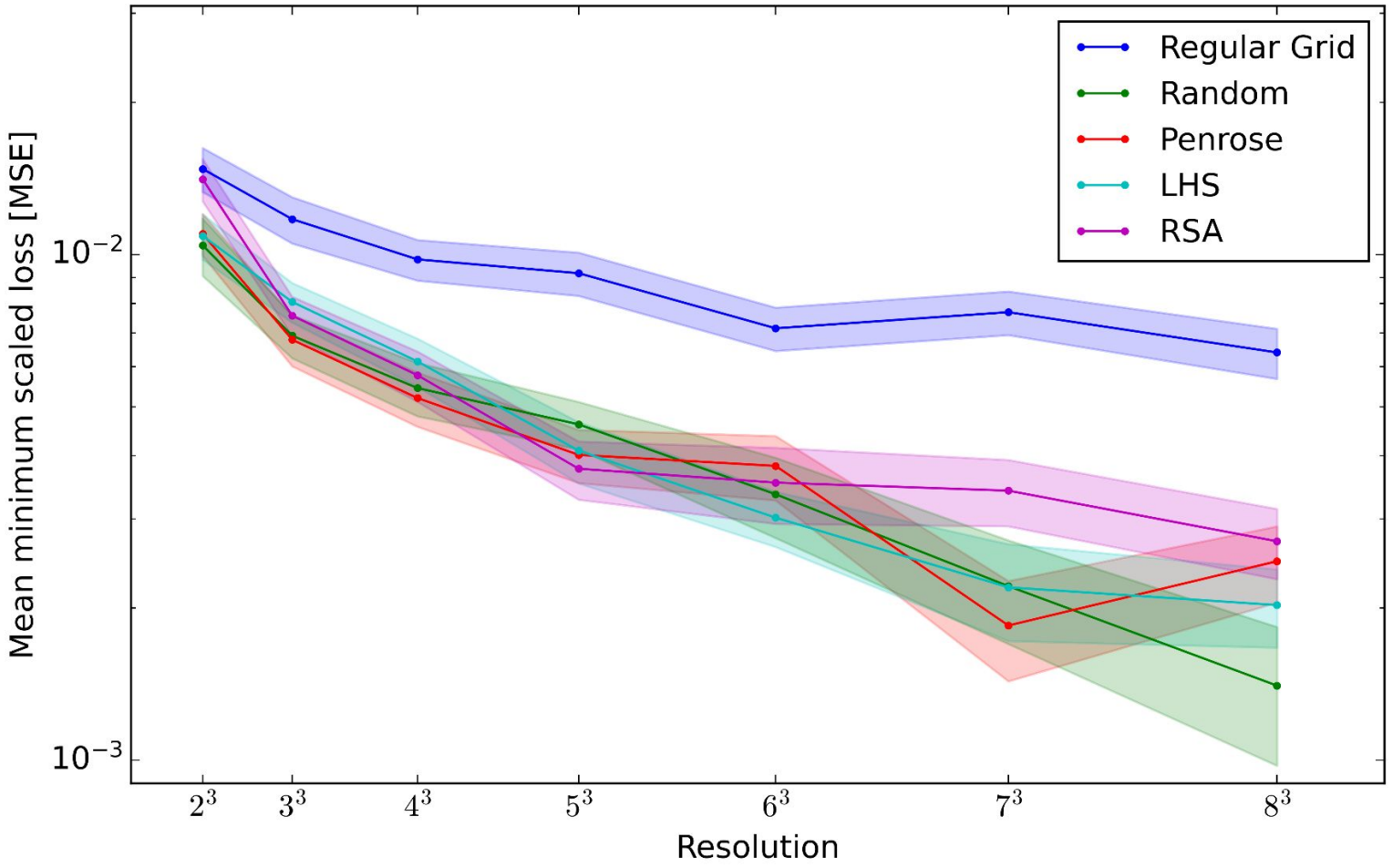


Bergstra, J. & Bengio, Y. Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research* **13**, 281–305 (2012).

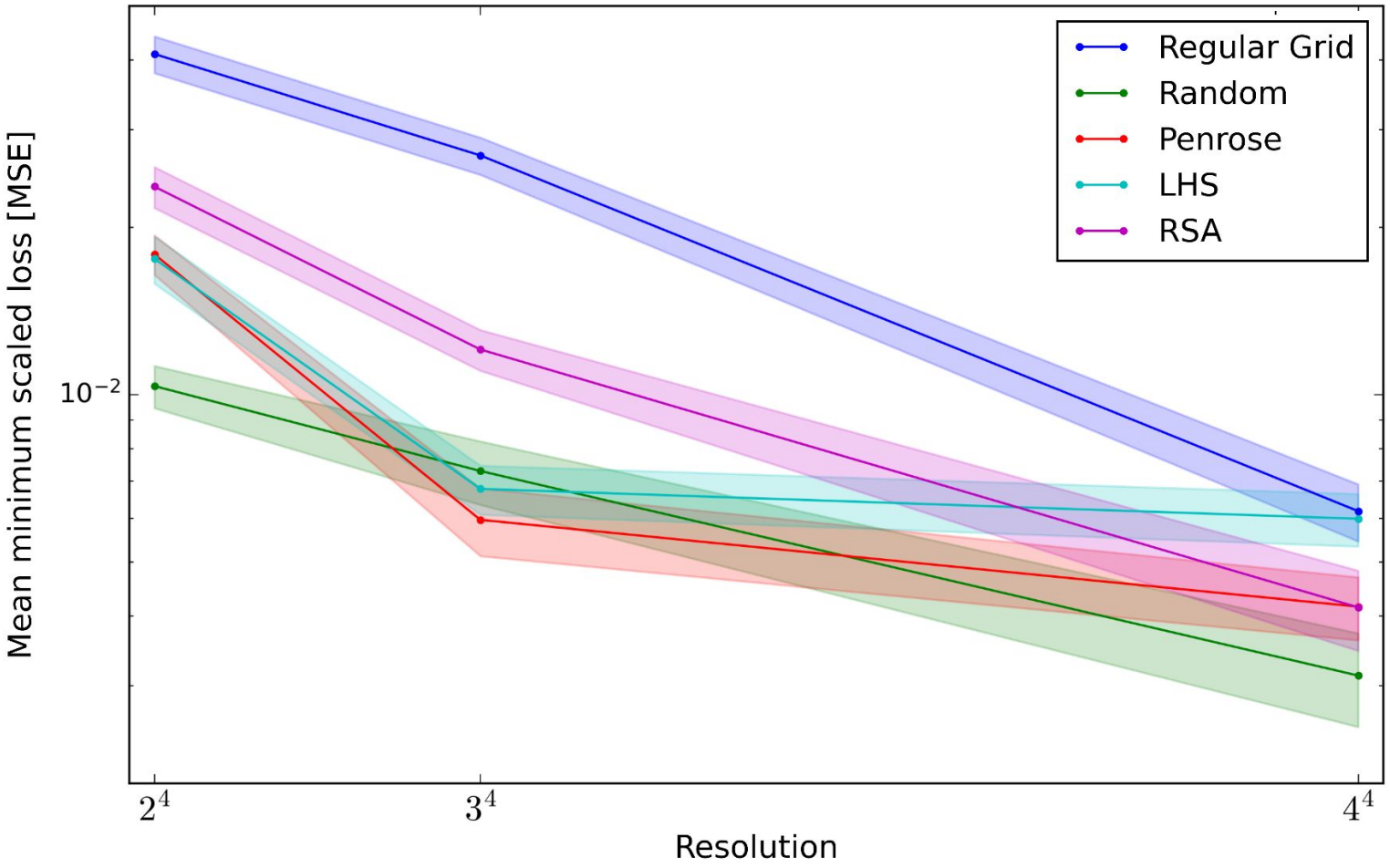
3D - with unimportant 3rd parameter



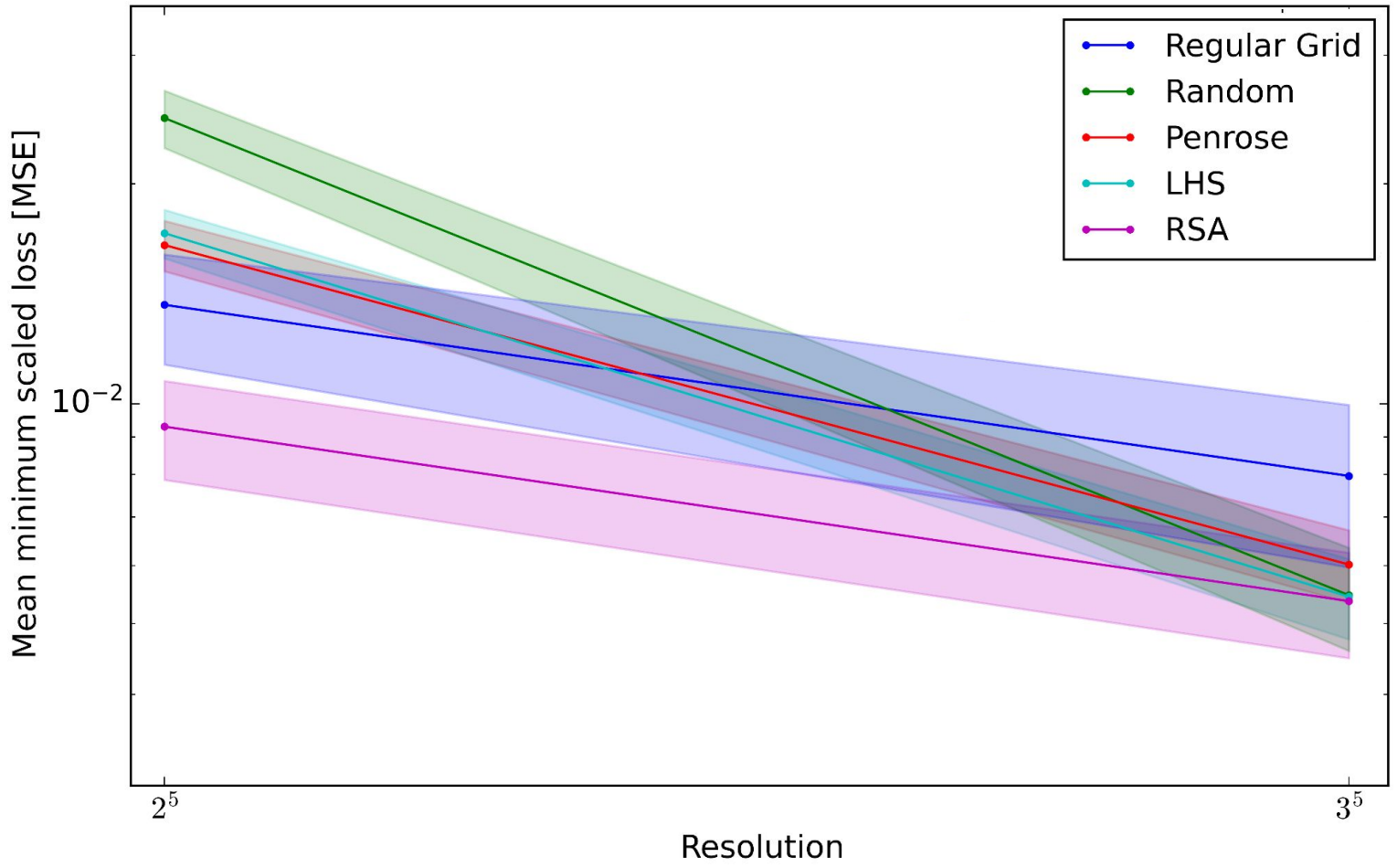
3D - with unimportant 3rd parameter



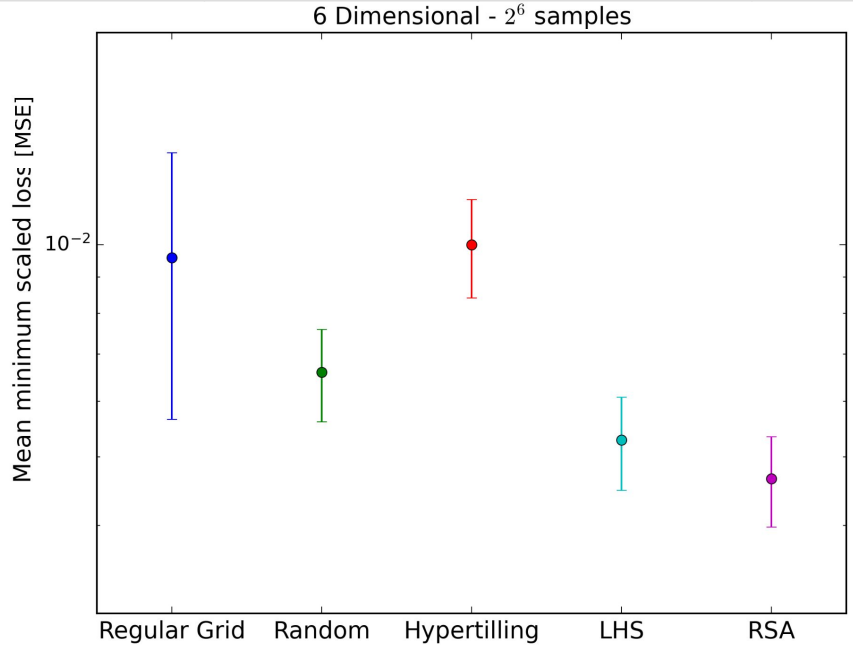
4D

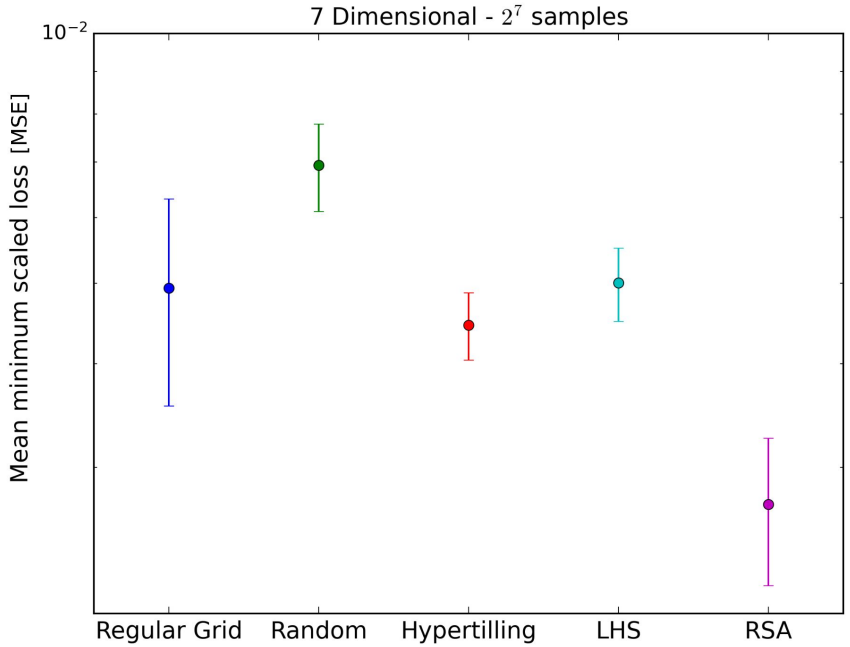

Regular Grid
Random
Penrose
LHS
RSA
Mean minimum scaled loss [MSE]
10^{-2}
2^4
3^4
4^4
Resolution

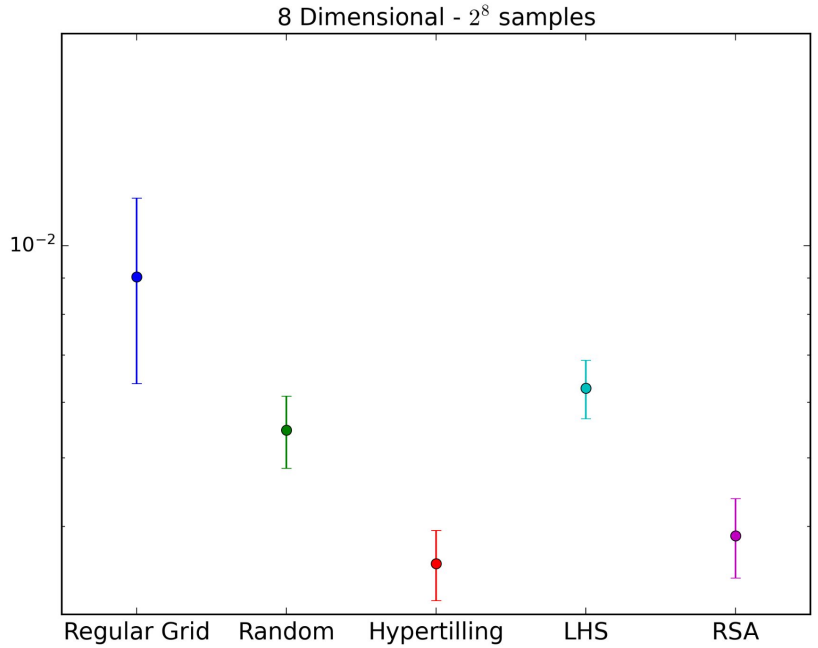
5D


Regular Grid
Random
Penrose
LHS
RSA
Mean minimum scaled loss [MSE]
10^{-2}
2^5
3^5
Resolution

Even higher dimensions








558.450
models trained

Discussion

- Low batch size
- Resolution and dimensions
- Only N^d grids
- Further work: More ML models and grid types

Resolution

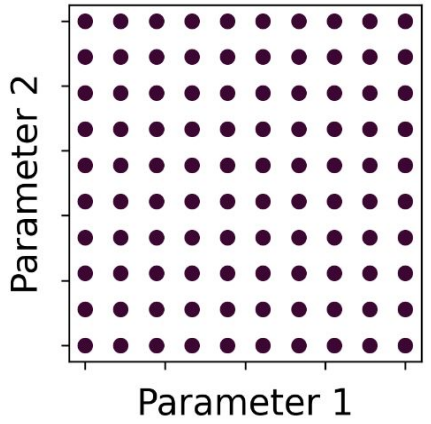
Dimensions

	2^d	3^d	4^d	5^d	6^d	7^d	8^d	9^d	10^d
2	T,C,N	T,C,N	T,C,N	T,C,N	T,C	T,C	T,C	T	T
3	T	T	T	T	T	T	T		
4	T	T	T						
5	T	T							
6	T								
7	T								
8	T								

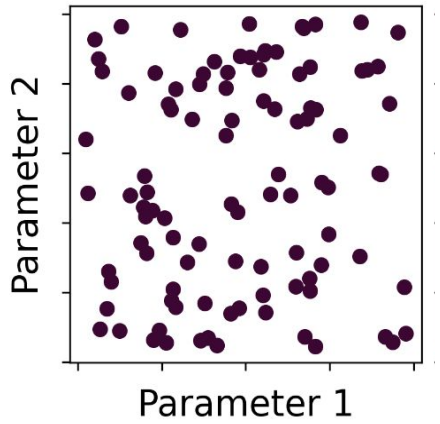
T: Tree-based, C: Convolutional NN, N: Neural Network

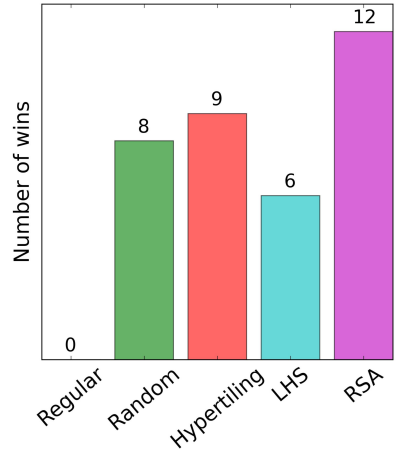
What should *you* do?

Regular

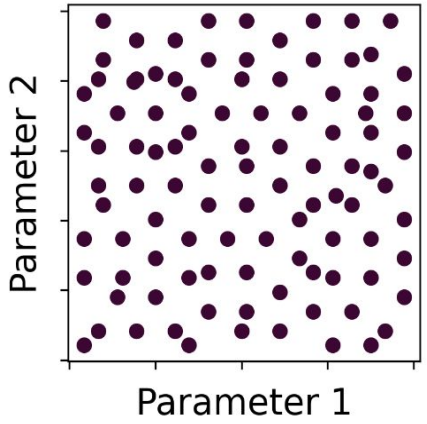


Random

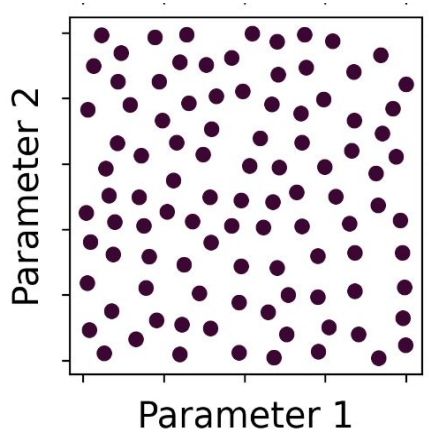




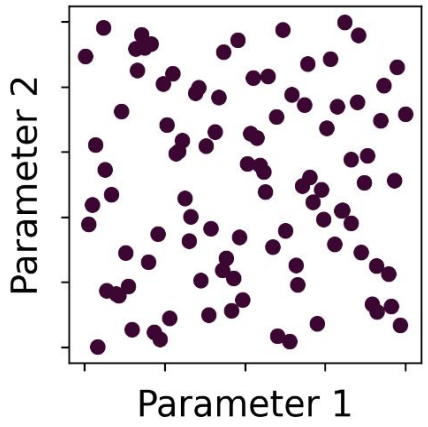
Hypertiling

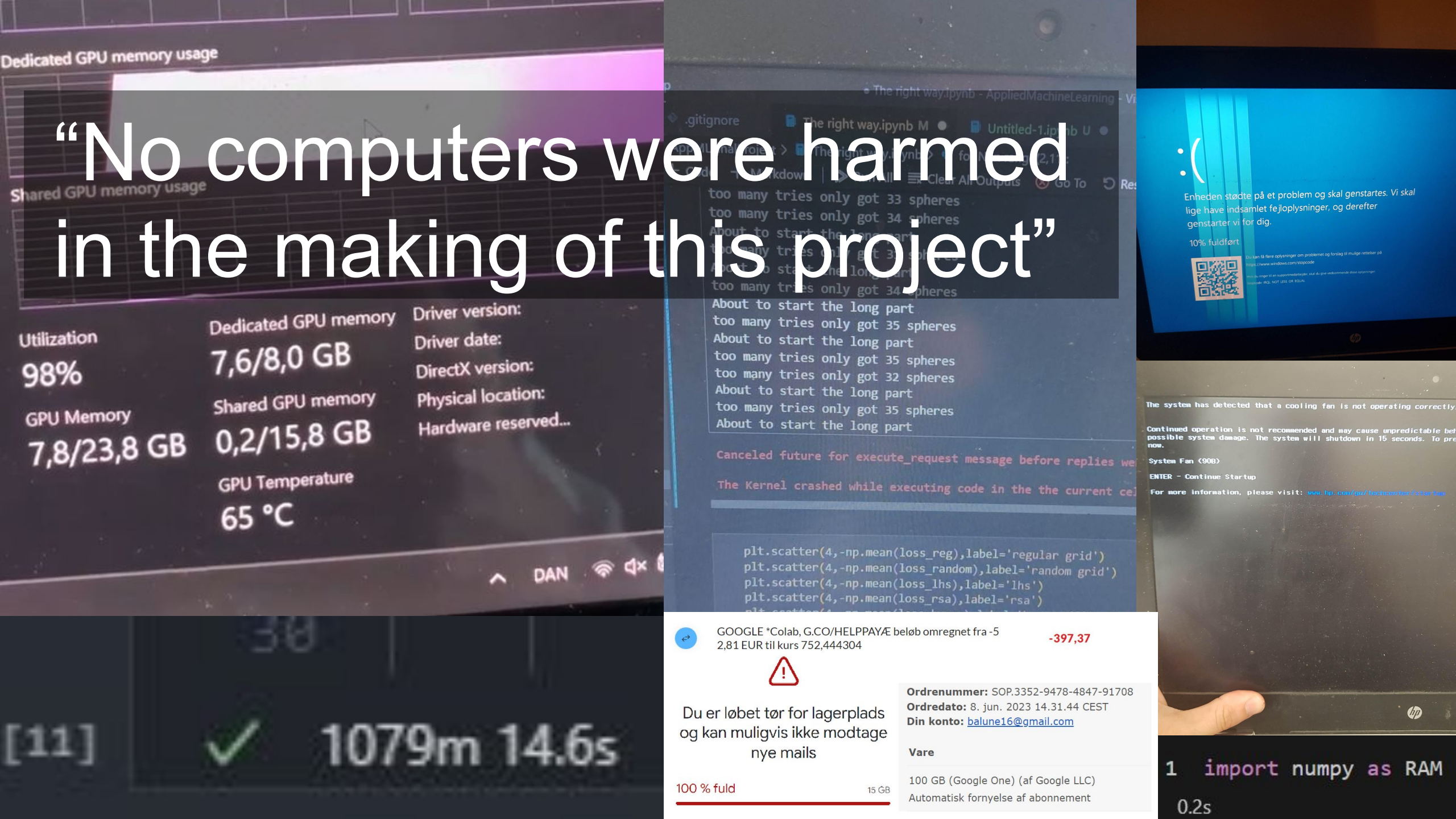


Random sequential addition



Latin hypercube sampling





“No computers were harmed in the making of this project”

Appendix overview

Moving the minimum and edge effects	Appendix M1
Data and standard deviation	Appendix M2
Normalization	Appendix M3

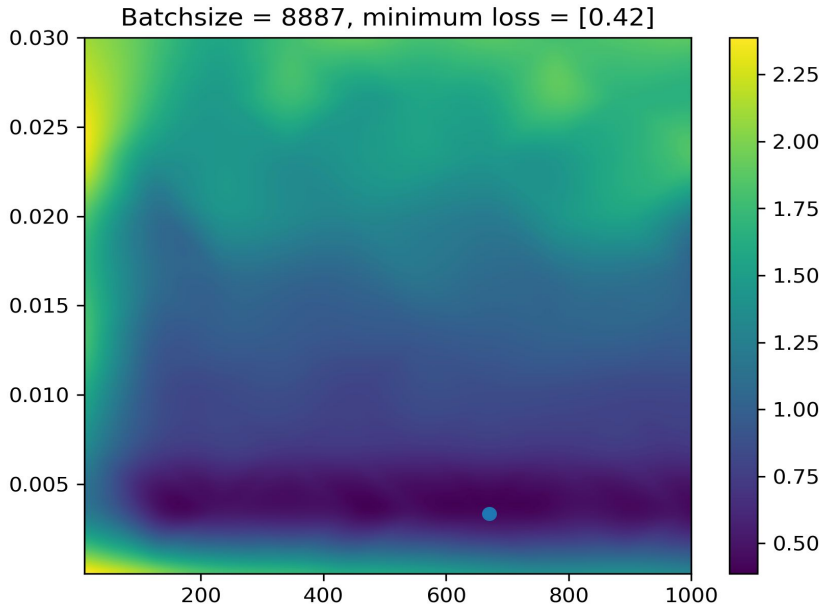
Data	ML Model	Succes?	Comment	Appendix
Housing data	**Tree-based:** Regression with LightGBM	Yes		Appendix A
CIFAR10	**CNN:** Labeling with Convolutional Neural Network	Yes	Long computation times.	Appendix B
Bjet Energy	**NN:** Regression with Neural Network	Yes	Long computation times.	Appendix C
Insolubles	**CNN:** Labeling with Convolutional Neural Network	No	Too long run time, single-batch data collected for sample size 4, 9 & 16	Appendix D
Ice cube	**GNN:** Graph Neural Network	No	Too long run time and complications with GPUS	Appendix E

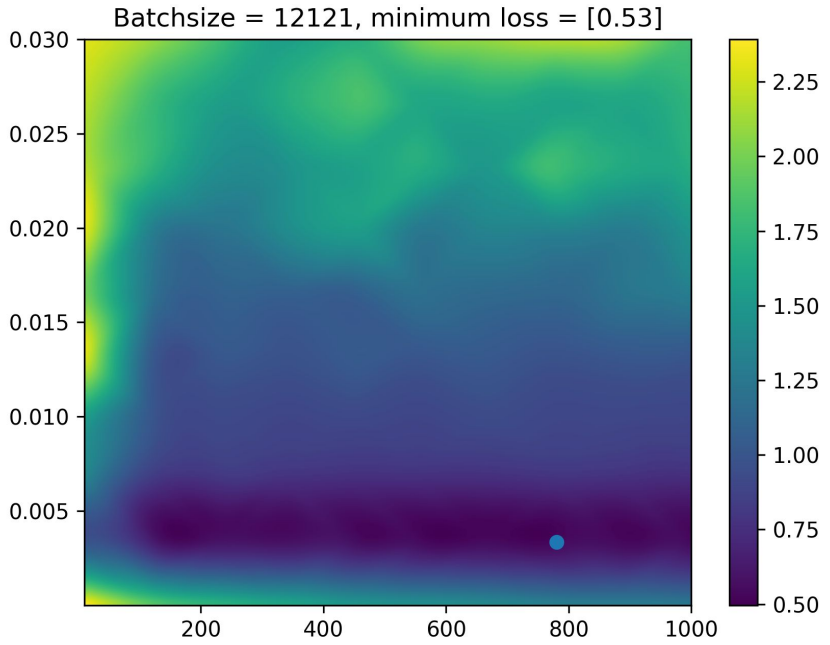
Appendix M1 - Moving the minimum and Edge effects

From data preprocessing it was possible to have minimas from each batch on the edge of the grid. If the minimum lays on the edge, then the real minimum might be outside of the range. We wish for at least a local minimum to be in the hyperparameter domain and not on the edge.

To avoid this several attempts were made. The most efficient way was to search another space by expanding the hyperparameter domain.

Another reason to avoid minimas on the edge is the edge effects. Both RSA and Hypertiling have edge effects, therefore to have the models to compete equally we will avoid batches with global minimums on the edge.

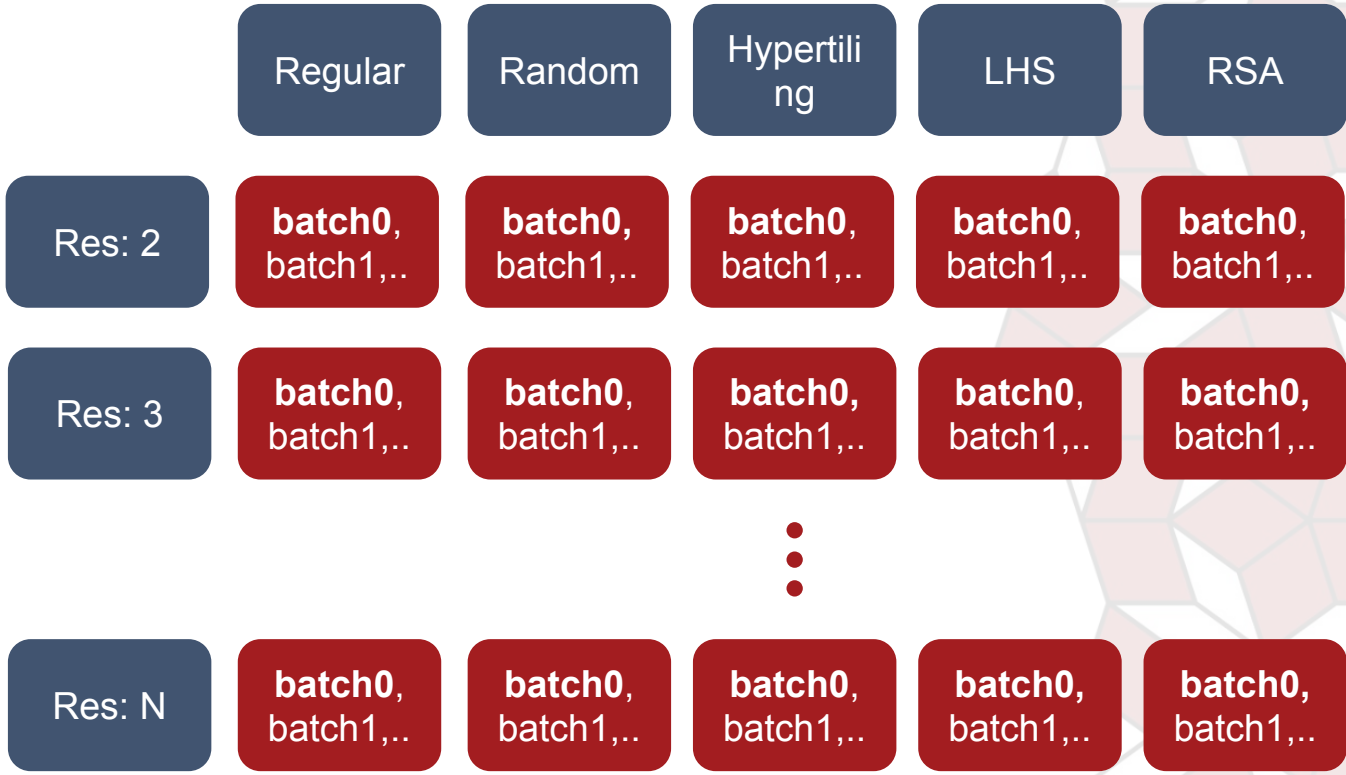




Appendix M2 - Normalization

The normalization is done to compare the batches from different models. For each batch we find the minimum for that batch across models and resolutions. Then divide all the loss values from batches with the global minimum for the respective batch. Ex. finding global minimum for batch0 in the figure and then divide all the batch0 loss values with the global minimum for batch0. Now it is possible to compare batches across models.

Now each batch is normalized and the difference across batches has been removed. Therefore we can compare across batches, for a given resolution, dimension and model. We therefore take the mean and standard deviation in the batch dimension. This gives us one value in each of the red squares on the figure to the right, these are the ones shown on the result plots in the slides.



A figure shown how there will be a distribution of batches in each model and each resolution. Here we normalize by finding the global minimum for each batches

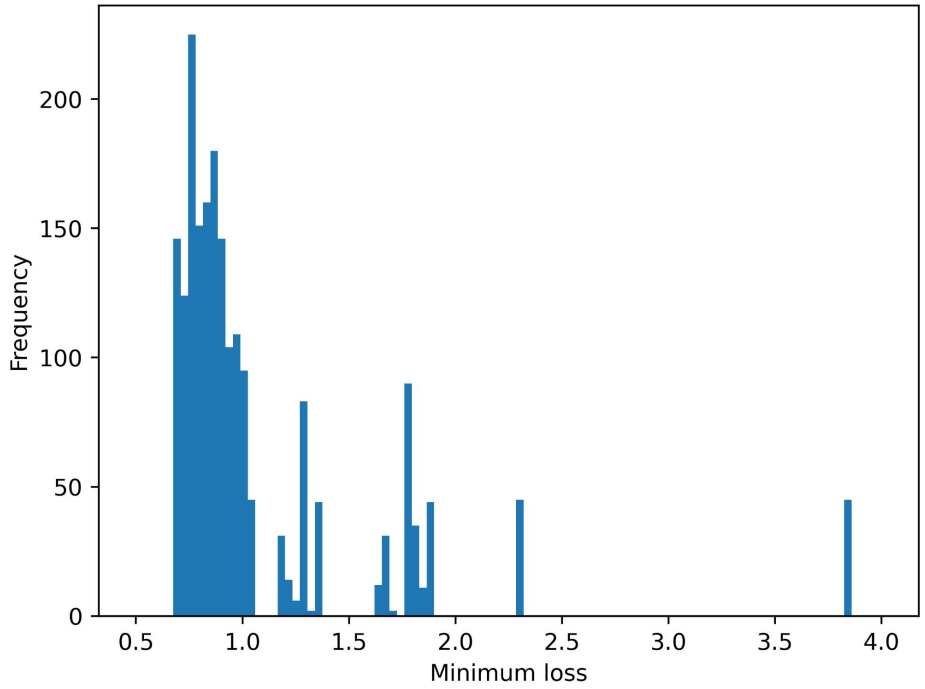
Appendix M2 - Normalization

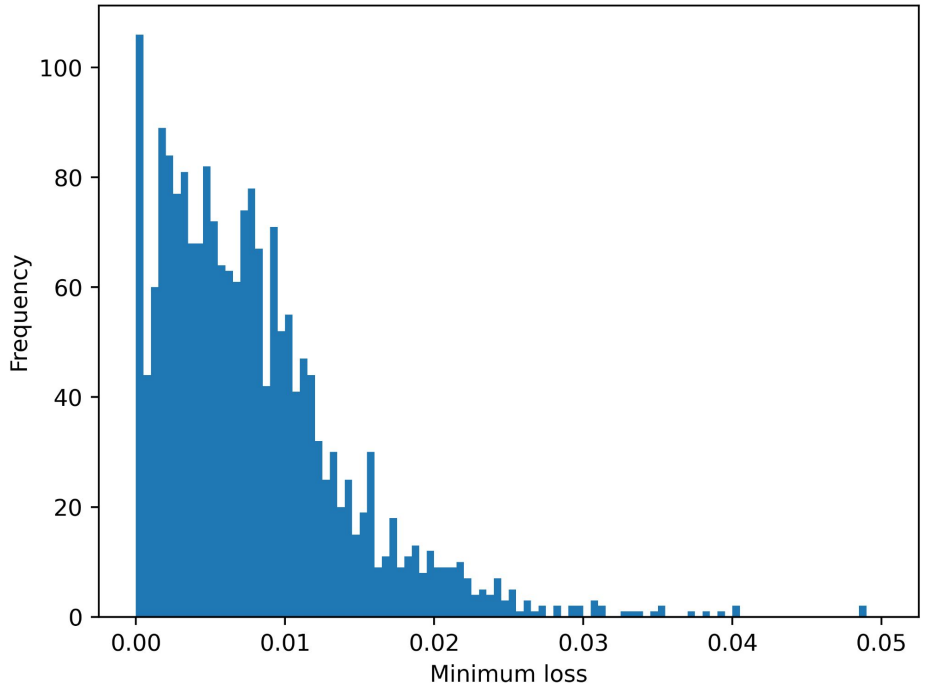
Here we show the distribution of all the minimum losses of the housing data before and after normalization to illustrate the effect of the normalization.

The top figure shows before normalization.
It is clear to see that the data is hard to compare as the variation is huge.

The lower figure shows after normalization.
Now the cross batch variation is removed. Therefore it is possible to compare the batches as the minimum loss is scaled.

There is a similar distribution of losses for the other benchmark problems.





Appendix M3 - How to get uncertainties

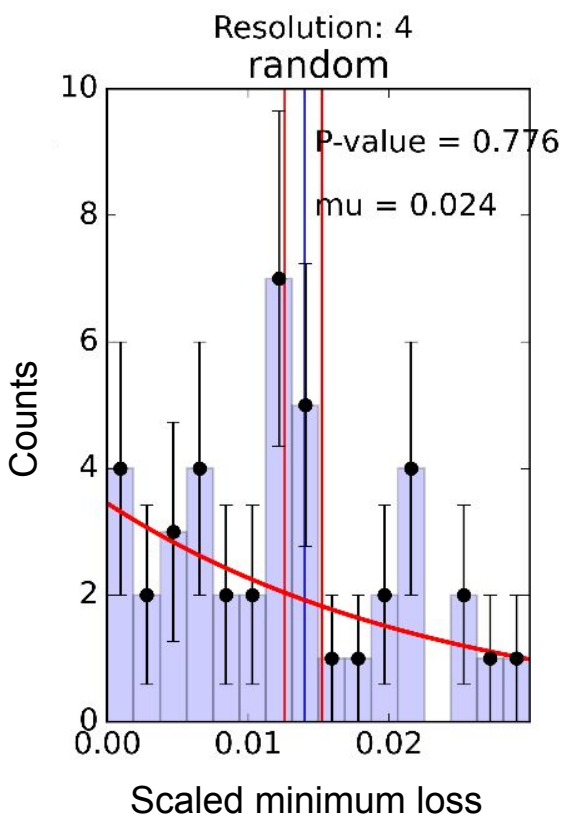
We want to find the mean and uncertainty of distributions similar to the examples seen on the right.

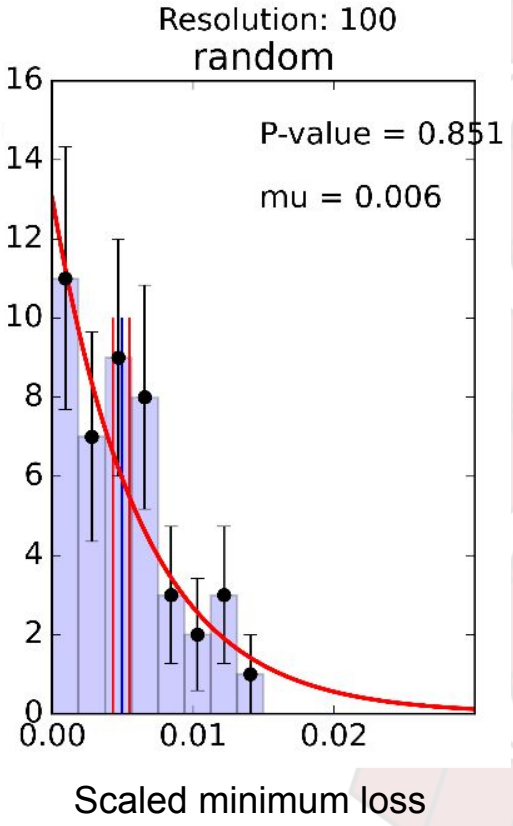
We found the uncertainties by assuming they where gaussian, thereby taking the standard deviation and dividing by the square root of the number of samples.

We recognize that we should have used some more sophisticated statistics based on an exponential distribution, as this would be correct for the right most plot. However the gaussian assumption is more correct for the lower resolution and therefore some cutoff should be chosen where we switch between the two.

Here we propose looking at the P-value of the chi^2 fit for the exponential distribution as the cutoff value.

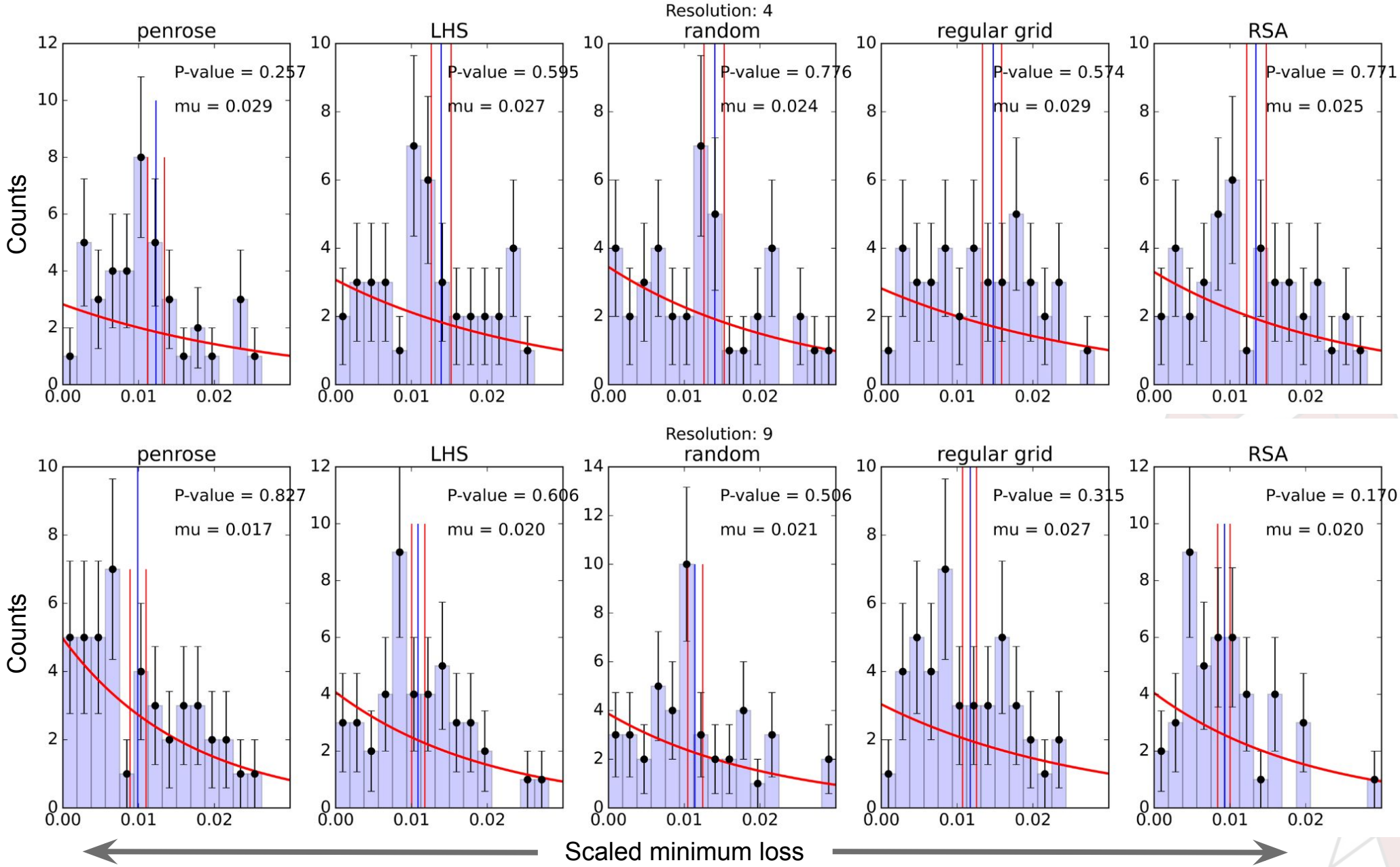
To see all the distributions see the next slides

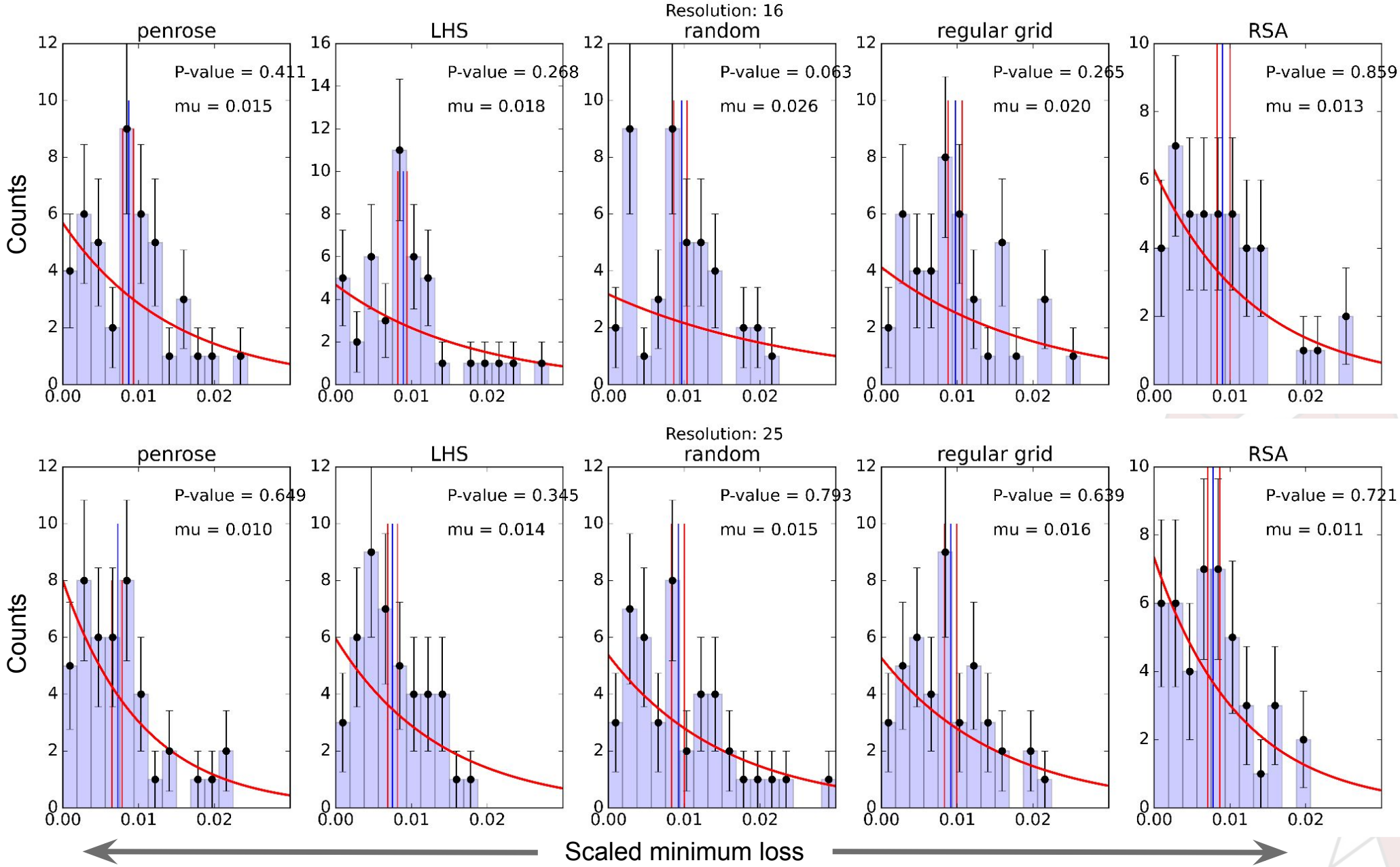


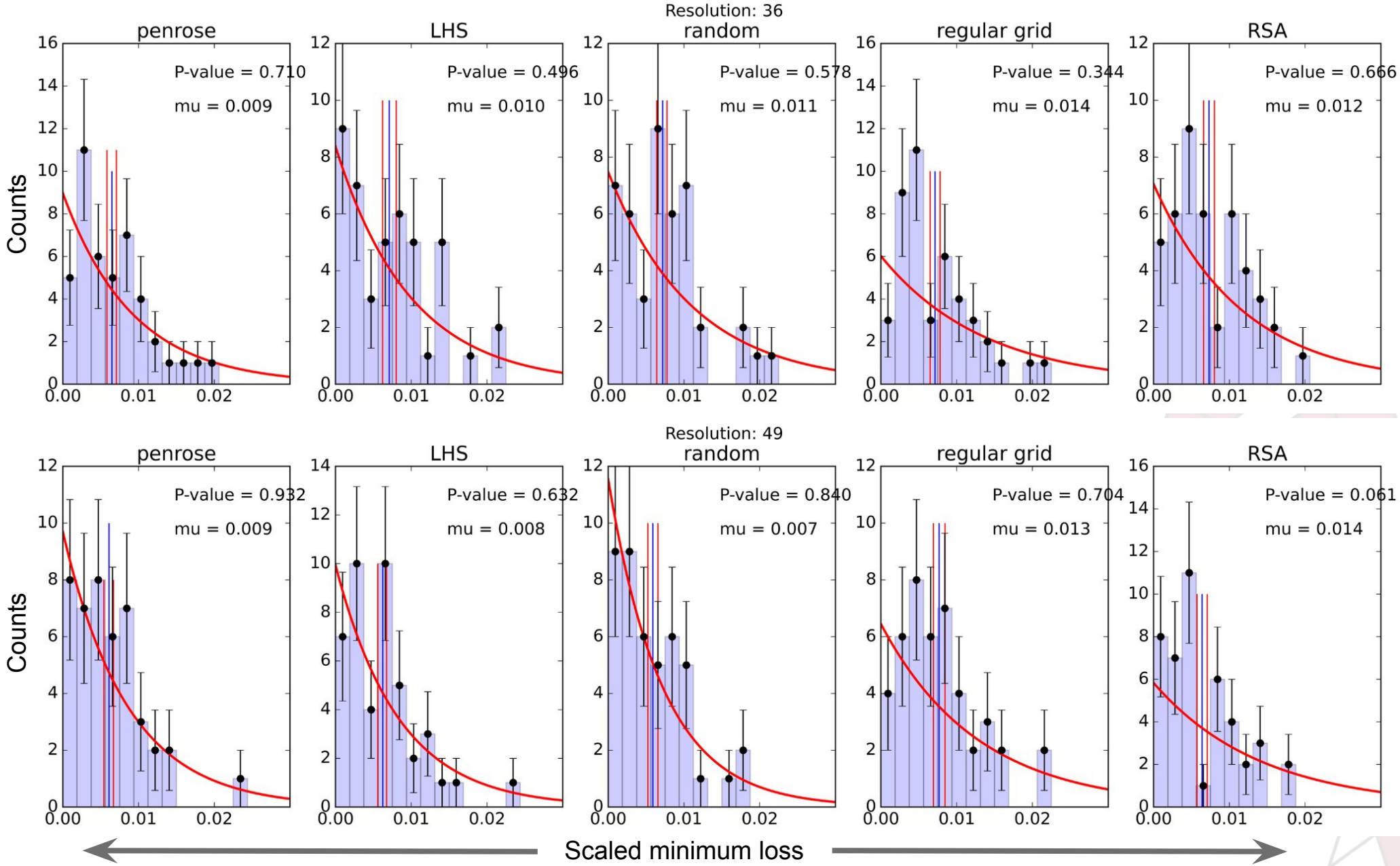


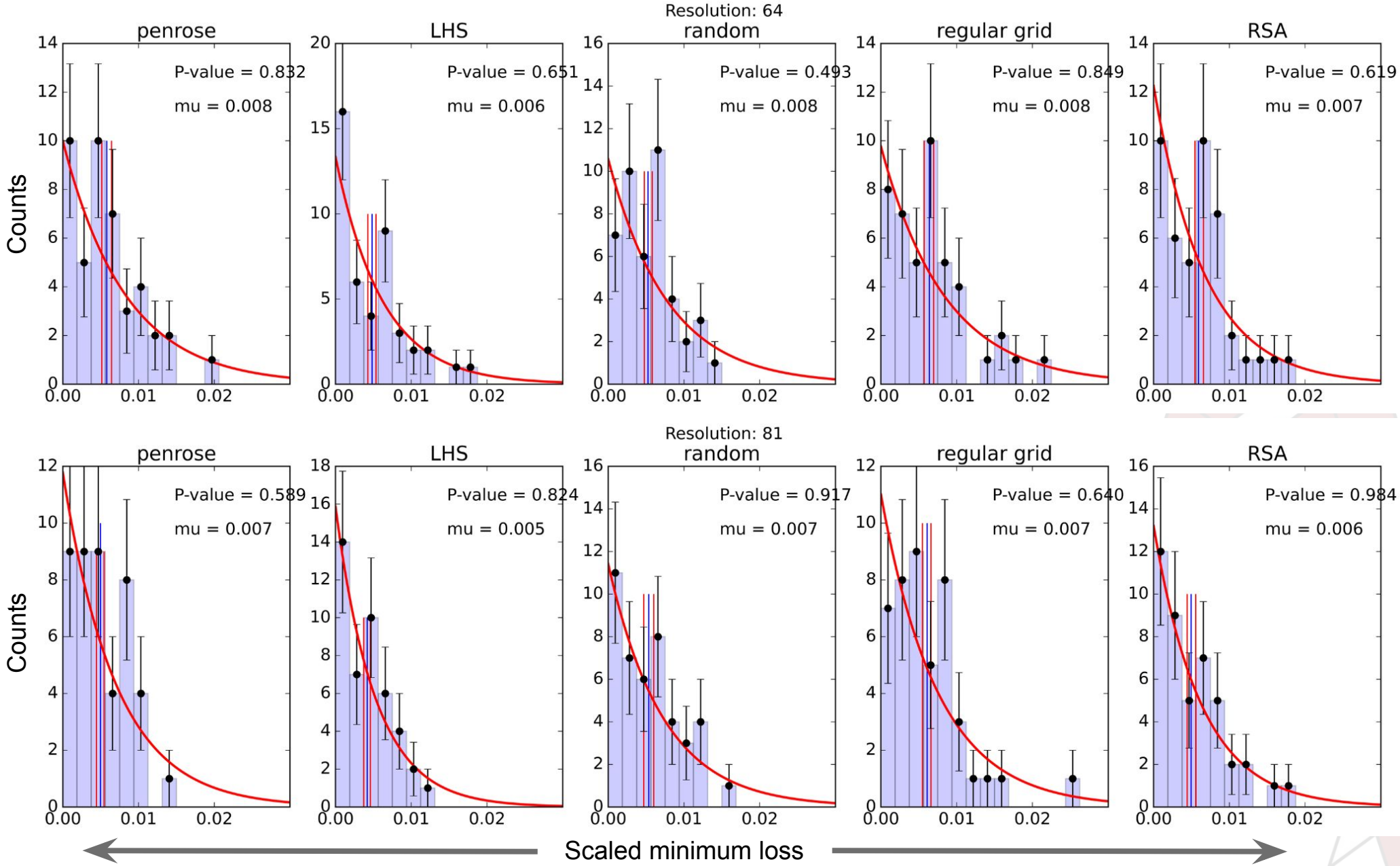
Fit of exponential distribution to minimum scaled loss for each sampling method and resolution. Blue vertical line is the mean of the distribution. The two red vertical lines indicate the error on the mean. The P-value is for the exponential distribution chi^2 fit.

We expect that with increasing resolution and thereby samples the distribution will become more and more exponential. This is because min scaled loss cannot go below 0 and with more trials per batch the best guess gets closer to the true value for each batch - meaning closer to 0.

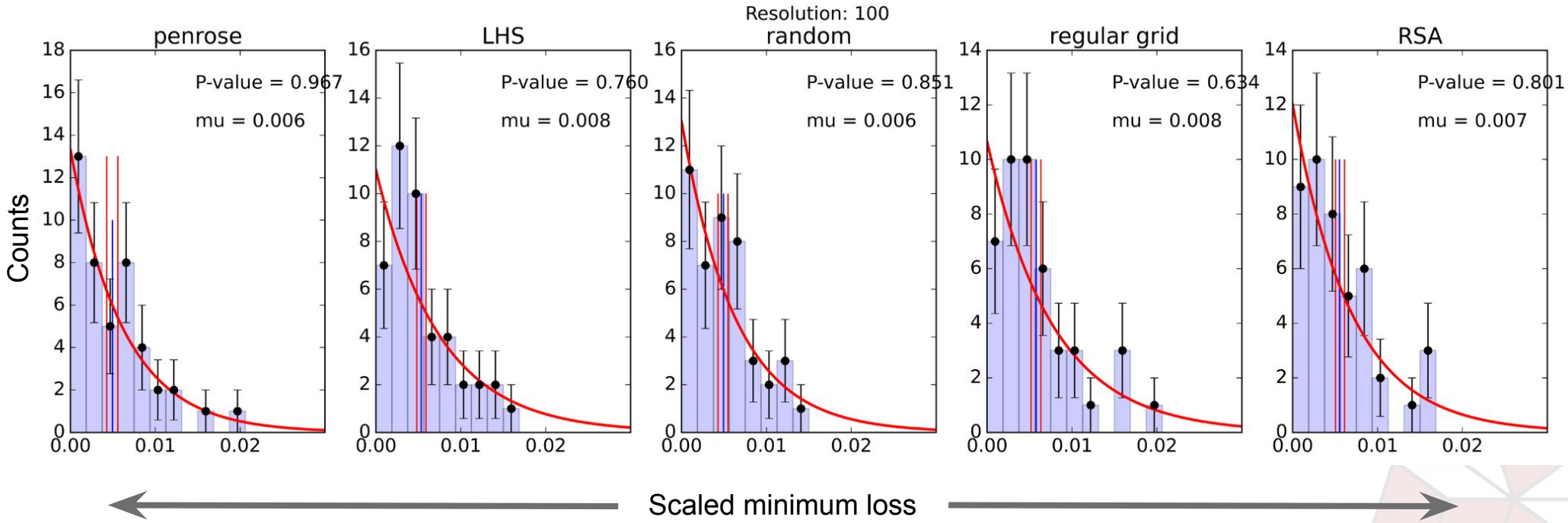



Resolution: 16
penrose
P-value = 0.411
mu = 0.015
LHS
P-value = 0.268
mu = 0.018
random
P-value = 0.063
mu = 0.026
regular grid
P-value = 0.265
mu = 0.020
RSA
P-value = 0.859
mu = 0.013
Resolution: 25
penrose
P-value = 0.649
mu = 0.010
LHS
P-value = 0.345
mu = 0.014
random
P-value = 0.793
mu = 0.015
regular grid
P-value = 0.639
mu = 0.016
RSA
P-value = 0.721
mu = 0.011
Counts
Counts
Scaled minimum loss


Resolution: 36
penrose
P-value = 0.710
mu = 0.009
LHS
P-value = 0.496
mu = 0.010
random
P-value = 0.578
mu = 0.011
regular grid
P-value = 0.344
mu = 0.014
RSA
P-value = 0.666
mu = 0.012
Resolution: 49
penrose
P-value = 0.932
mu = 0.009
LHS
P-value = 0.632
mu = 0.008
random
P-value = 0.840
mu = 0.007
regular grid
P-value = 0.704
mu = 0.013
RSA
P-value = 0.061
mu = 0.014
Counts
Counts
Scaled minimum loss


Resolution: 64
penrose
P-value = 0.832
mu = 0.008
LHS
P-value = 0.651
mu = 0.006
random
P-value = 0.493
mu = 0.008
regular grid
P-value = 0.849
mu = 0.008
RSA
P-value = 0.619
mu = 0.007
Resolution: 81
penrose
P-value = 0.589
mu = 0.007
LHS
P-value = 0.824
mu = 0.005
random
P-value = 0.917
mu = 0.007
regular grid
P-value = 0.640
mu = 0.007
RSA
P-value = 0.984
mu = 0.006
Counts
Counts
Scaled minimum loss

As we expected the distributions have become exponential.



Appendix A

To reduce the amount of features we trained a model on the whole dataset using optuna to find the best hyperparameters. From here we used SHAP values to get the 20 most important features, seen on the right. This makes the training computationally easier.

Then we split the dataset into batches with the size given by a gaussian with mean 10 000 and spread 1 000 to get different sizes. This is to move the minimum in the hyperparameter domain as explained in Appendix M1. The hyperparameter domain is learning rate (range 0.03 to 0.25) and max depth (range 8 to 22).

In the model we introduced early stopping to eliminate the n-estimators as a parameter and thereby making the training more effective. This also ensures that we will not overtrain.

20 BEST FEATURES

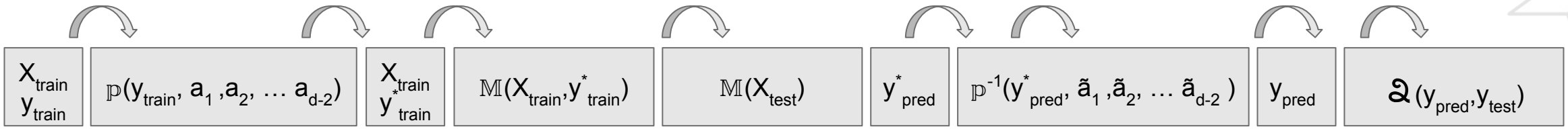
GeoKommuneNr
ArealBolig
HisSalgsPris1
ByggeAAr
ArealGrund
Hoejspaendingsledning
Kyst
GeoPostNr
Vindmoelle
EnergiMaerke
BeregnetAreal
Motorvej
JernbaneSynlig
EnhedAntalToilet
OprettetDato
HisSalgsDato1
Rigsgraense
RekreativtOmraade
MotorvejTilFraKoersel
AnnonceretDato

Appendix A - Adding dimensions

In 2d we had learning rate and max depth but as seen in the edges appendix slide, it is very time consuming to find new hyperparameters that is inside the searched space.

In order to save time on searching for the hyperparameter domain, we take the y values (housing prices) and mangle them through a purely odd power polynomial with positive coefficients as that is a bijective transformation. The coefficients of these polynomials are our new parameters for the tilings to search through. We can generate these true parameters and thereby be certain that they are random but inside a specified range.

How did we do it? - please follow the below diagram.
We take our y train data (y_{train}) and run it through a polynomial with *d-2* coefficients ($\mathbb{p}$) only using ord power terms, this is because we want the transformation to be bijective. The -2 comes from the fact that we already have learning rate and max depth as our primary hyperparameters. The mangled y train data (y^*_{train}) is then used to train our model ($\mathbb{M}$). The model is then used on the X test data (X_{test}) to get a mangled y prediction (y^*_{pred}). The y data (y^*_{pred}) is then unmangled using the inverse polynomial ($\mathbb{p}^{-1}$) with coefficients given by the hyperparameter sampling method to get the true prediction (y_{pred}). This final prediction is then evaluated using the lossfunction ($\mathfrak{L}$) against the true y test data (y_{test}). The polynomial coefficients are then our added dimensions.



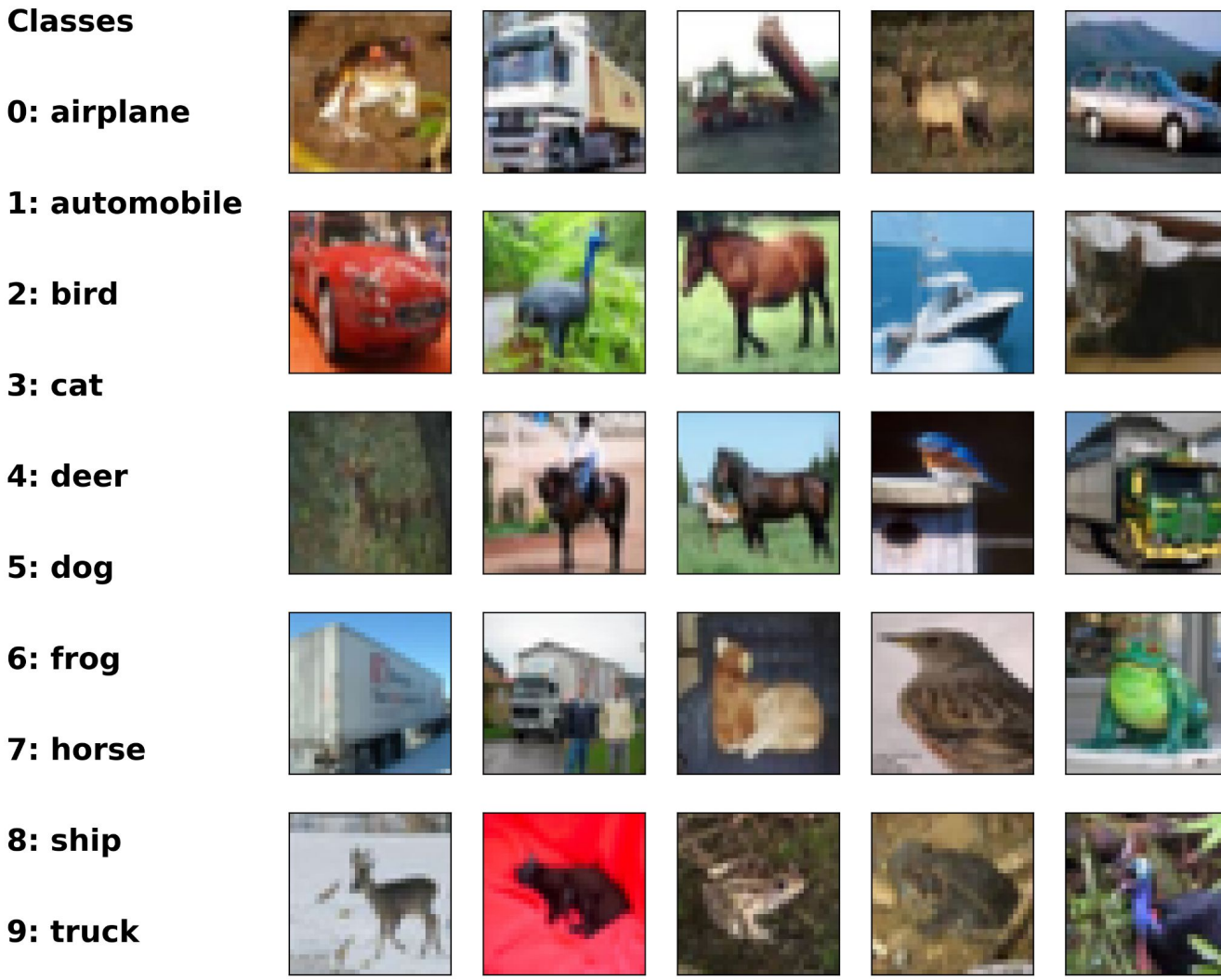
Appendix B

To do the analysis of the CIFAR10 dataset, the dataset was split into 10 batches of random size, uniformly chosen between 7500 - 12500 points, to have changes in the loss-landscape.

The multi-class classification was then done utilizing ResNet50, a deep learning network already trained using the imagenet weights.

We plastered a fully connected dense layer on the ResNet50 network used for the deep learning to our pictures, and generating a tuneable hyperparameter for the searches.

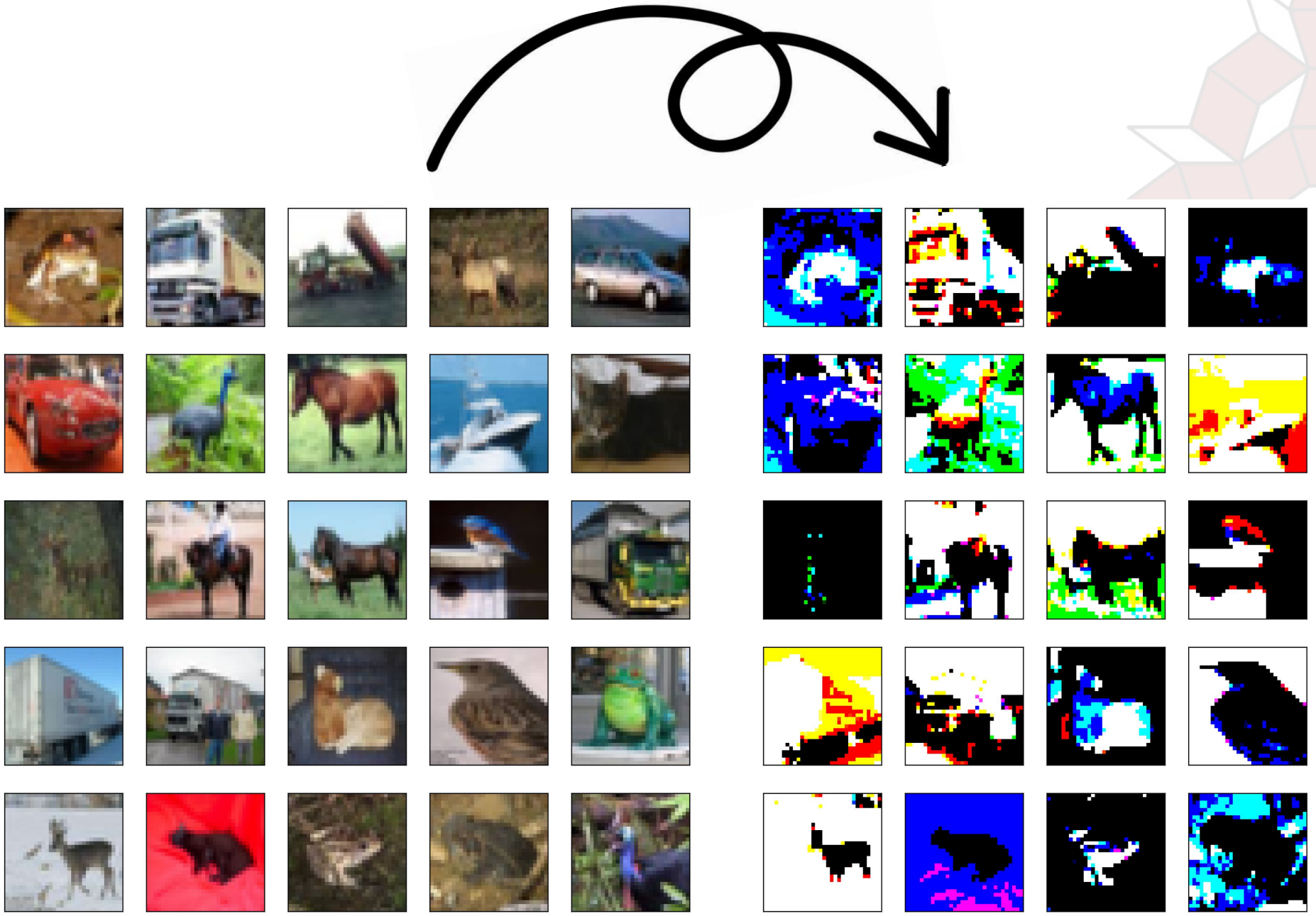
Then individual searches for the the minimum of the batches was made, to ensure a uniform parameter space, where minima did not lie on the edges for any of the batches, as explained in appendix M1.



Appendix B

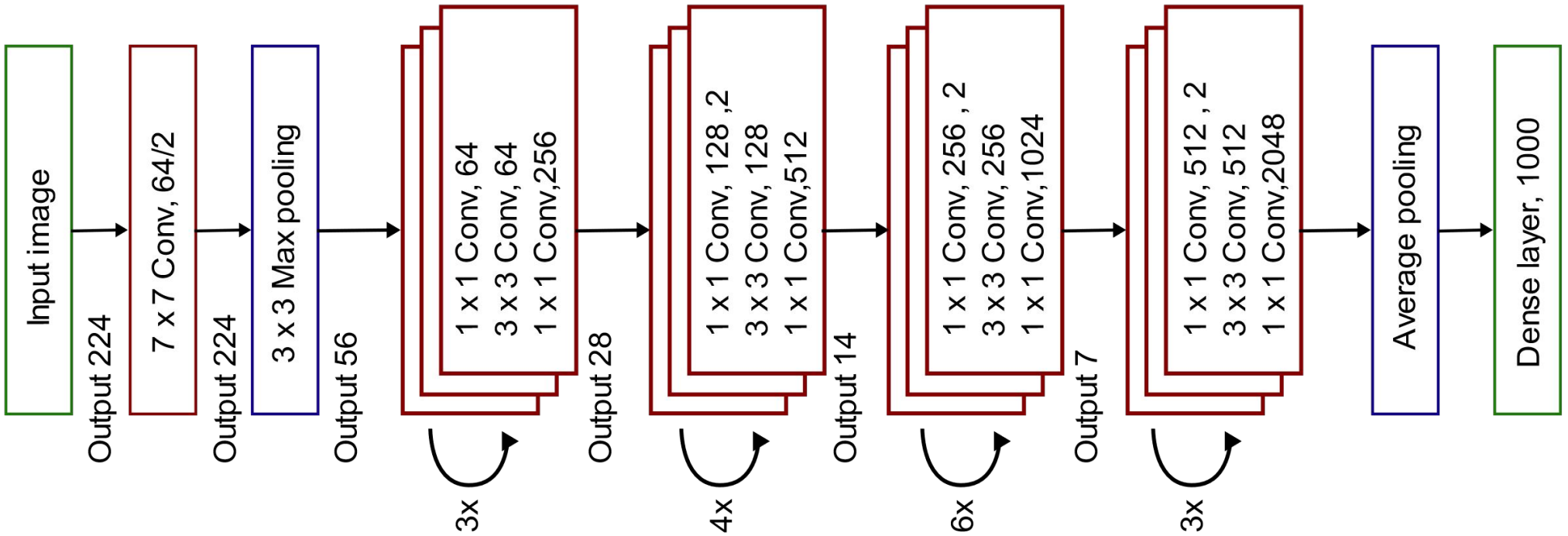
For the CIFAR10 dataset to work with ResNet50, the data is preprocessed by changing from RGB color scale to BGR, and each color channel is zero-centered with respect to the ImageNet dataset, omitting scaling.

The test data is converted from a class vector to a binary class matrix to prepare for the classification.



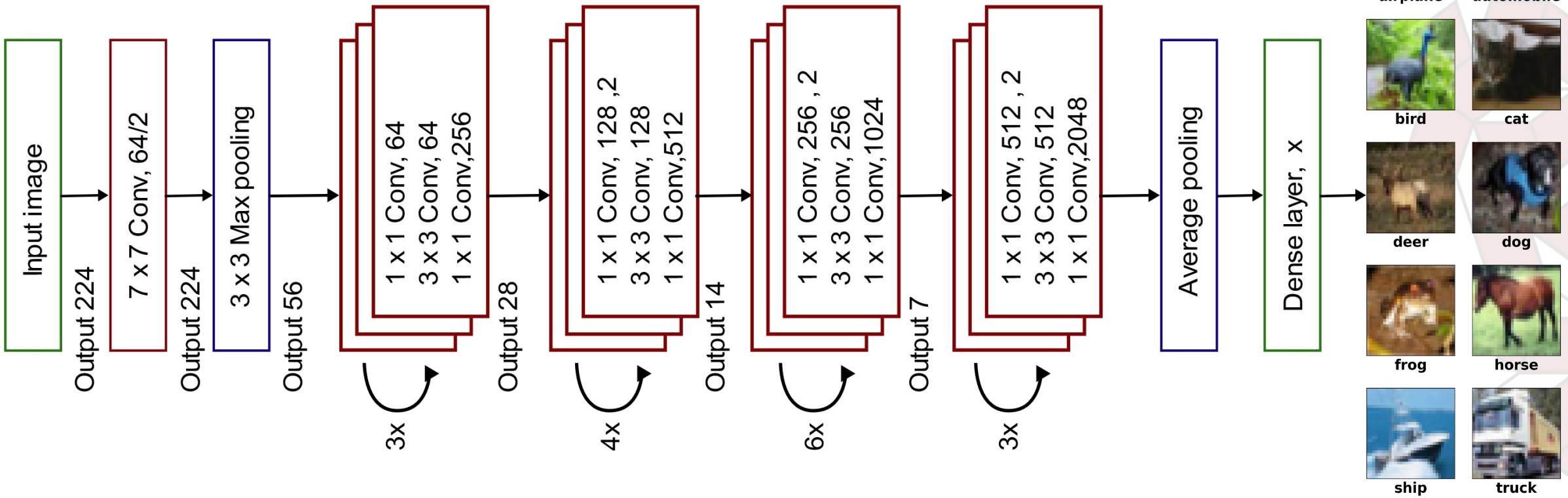
Appendix B

The data is then run through the ResNet50 network. The architecture of which is shown here, ResNet50 is composed of several large convolution layers, which is noted as “m x m Conv, n, o”, meaning a Convolutional layer, with n kernels of size m by m, with stride o. The middle convolutional layers are repeated, denoted by the arrows. Lastly, there is a fully connected layer corresponding to the 1000 classes in the original Imagenet dataset.



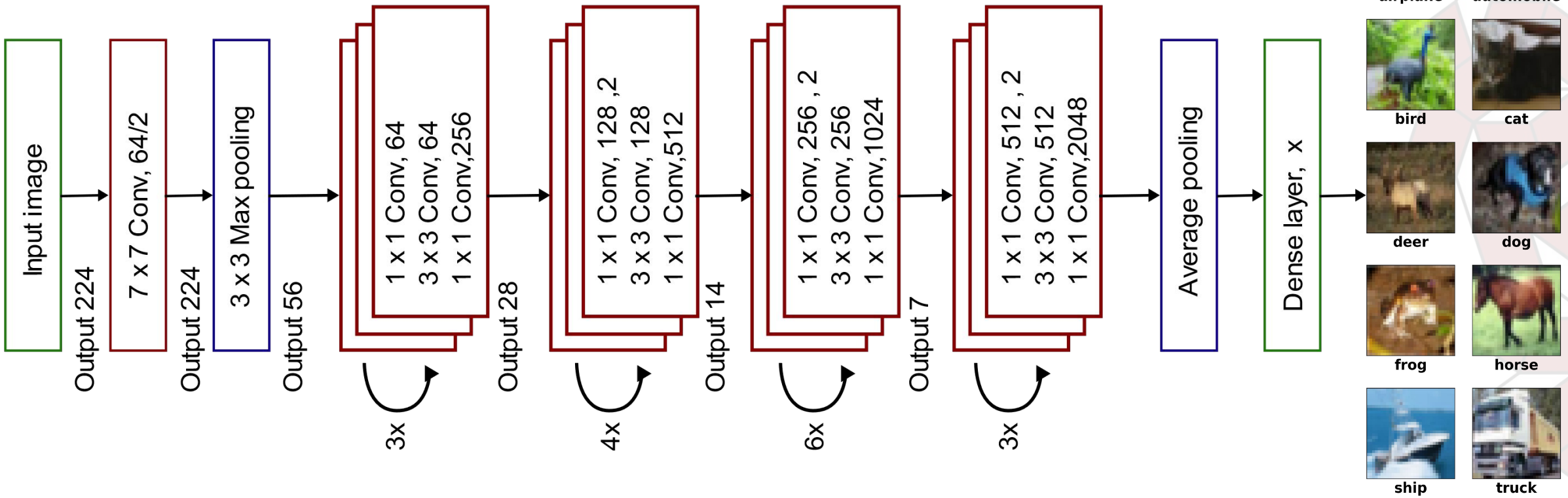
Appendix B

We modify the ResNet architecture by adding another fully connected layer, which will be used to tune the network to our images, instead of the original Imagenet pictures. The size of this dense layer is a tuneable hyperparameter for the searches. After this, another dense layer with 10 units, is used for the classification to the 10 classes, using a softmax activation function, which is great for multiclass classification. All the layers before the tuneable dense layer is frozen, so the weights are not affected when training on the CIFAR10 dataset, only the dense layer.



Appendix B

ResNet50, was chosen since the pretrained layers allows the CNN to achieve a high accuracy (low loss) from the network complexity, but since the original network is trained on wildly different pictures, the new model is very sensitive to the chosen hyperparameters (fluctuating between a validation accuracy of 10% - 95%), which we saw as favorable when comparing the searches, since this penalises bad parameters severely.



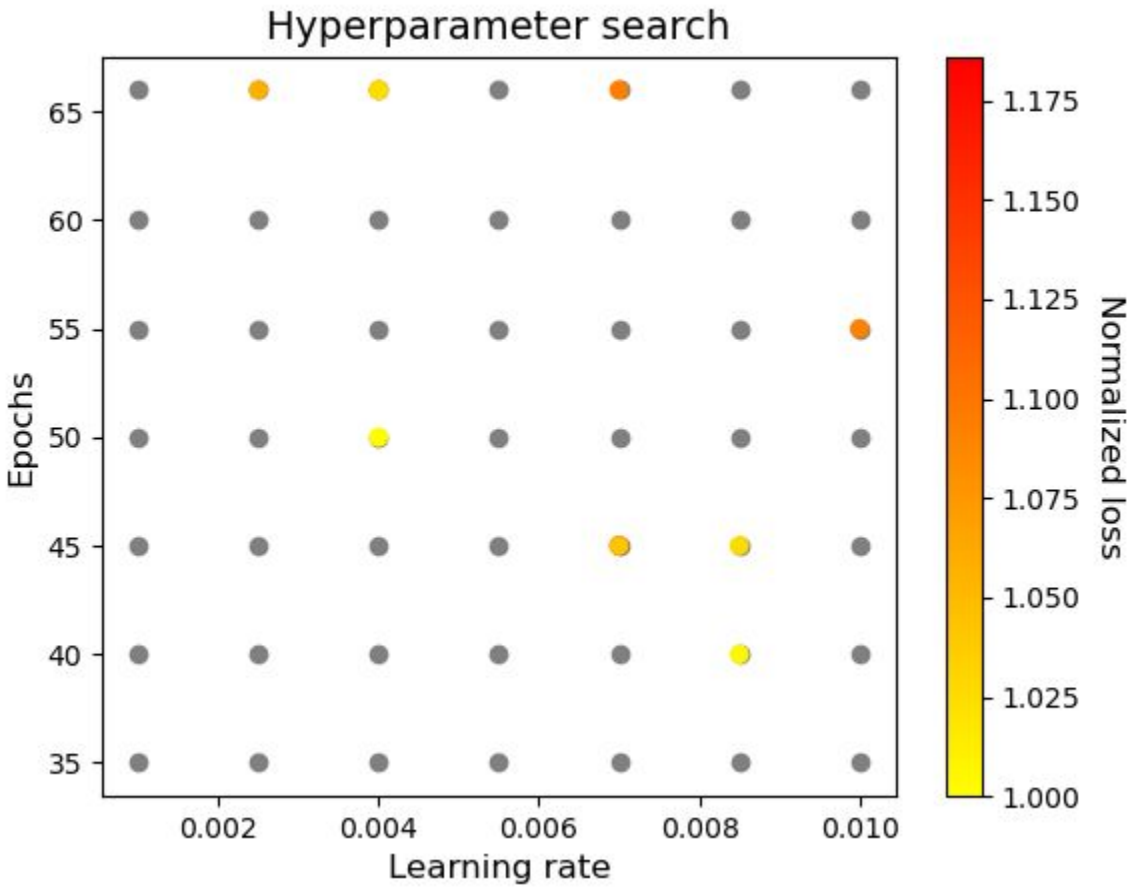
Appendix C

Energy regression on Aleph Bjet with Neural Network with tensorflow. Here we used Optuna to find the number of neurons in each of the two layers. We also used Optuna to find a range to search for the minimum.

In this neural network we looked at two parameters, learning rate (range 0.001 to 0.1) and epochs (range 35 to 66). The dataset was split in 10 batches of equal size and each batch was splitter in train and validation. From here we used the gridsearch to see, where the absolute minimum in the batch is.

On the figure we see that the minimas are moving around confirming that each batches have a different minimum.

Now the simulation was then done for 2x2, 3x3, 4x4 and 5x5 where all parameters for each grid was tested.

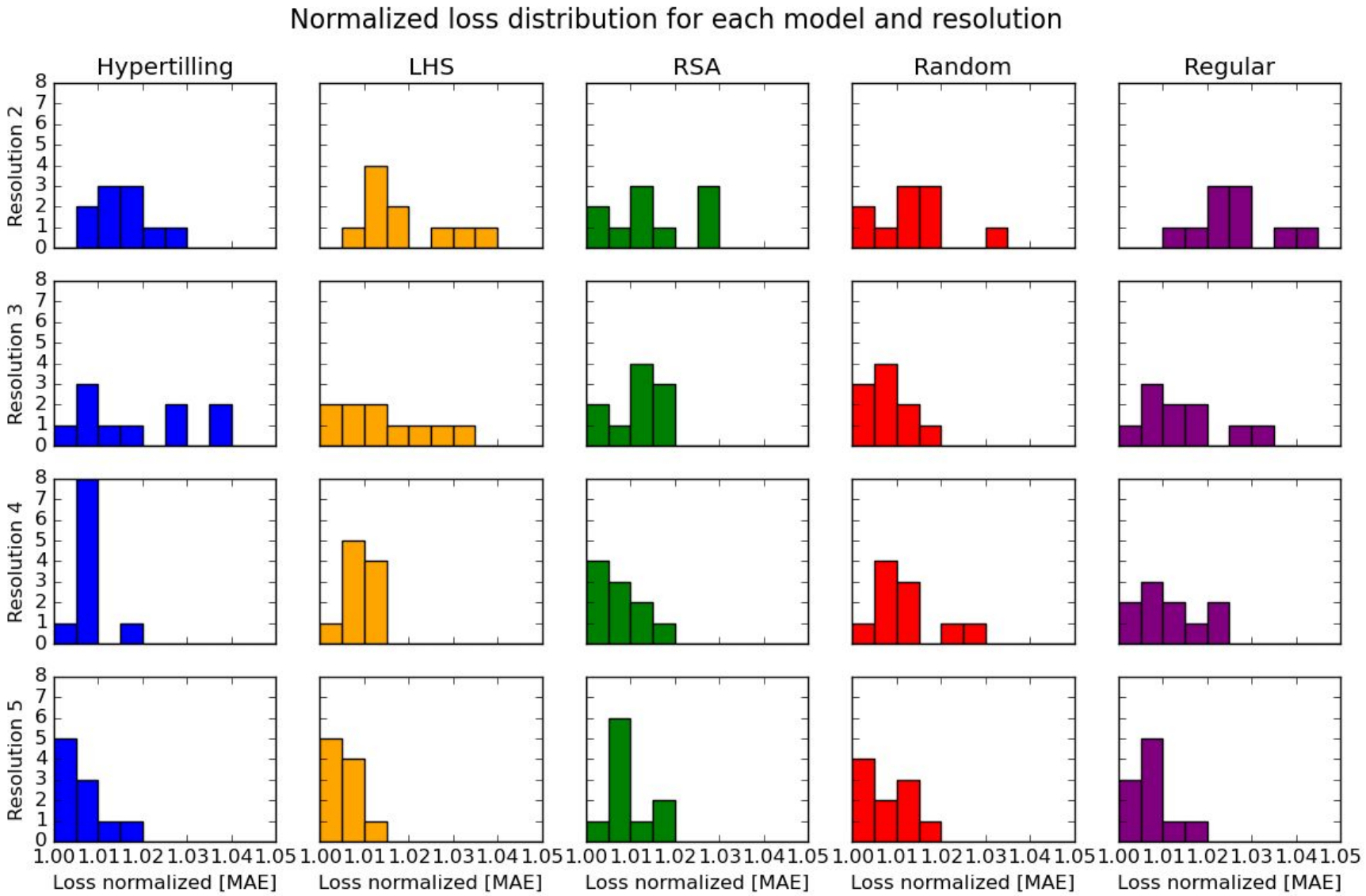


Appendix C

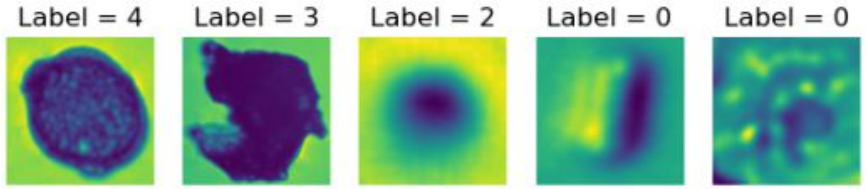
Here is an overview of the distribution of the normalized data. They are normalized with respect to the global minimum for each batch.

The figure shows the distribution for each model and each resolution. Here we used standard deviation even though it is not gaussian distributed.

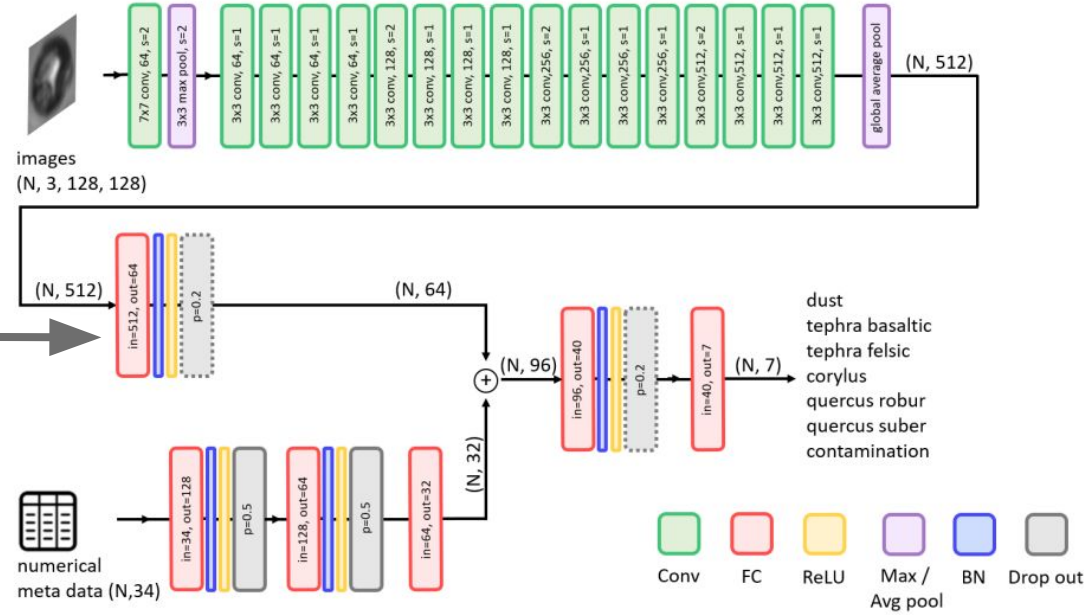
To make the figure in the presentation we took the mean in each subplot and plotted the resolution against the normalized loss and the error given from the standard deviation.



Appendix D

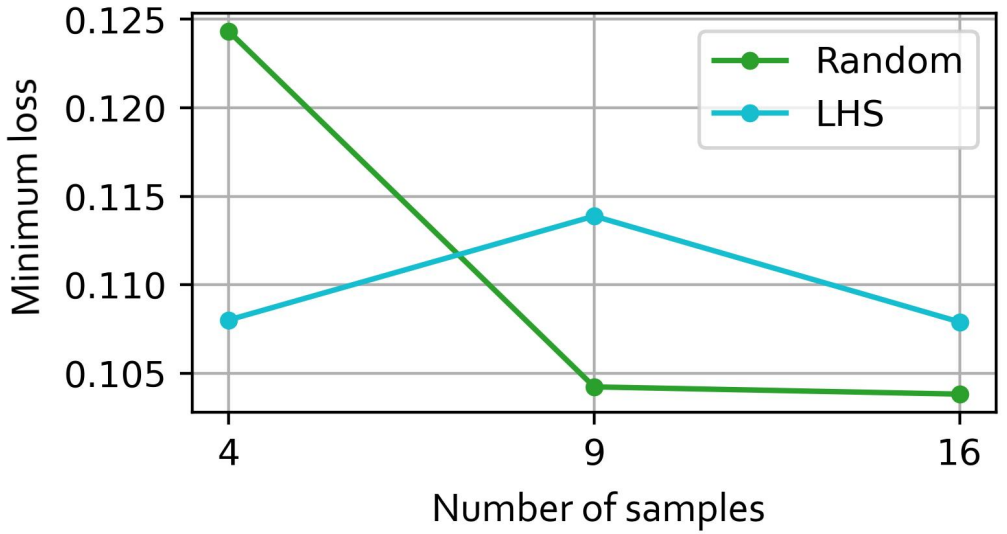


The insolubles dataset consists of 136.021 grayscale images of ice core impurities in 128x128 resolution. The convolutional neural network used for hyperparameter optimization follows that of the master's thesis *Machine Learning Methods for Autonomous Detection of Impurities in Ice Cores* (2022) by Amalie F. Mygind.



Hyperparameters examined were learning rate (LR) and LR decay (γ). Run times were extremely long, around 1 hour for *each* grid point, meaning that a single-batch run for the 5 sampling methods in a 3x3 grid would take around 45 hours (and we would need e.g. 10 batches to get an idea of means and variances).

Single-batch data for a 2x2, 3x3 and 4x4 grid was collected using random sampling and latin hypercube sampling (LHS). These batches did not show clear improvement in using latin hypercube sampling compared to random sampling.



Naturally, with this few data points randomness determines which model performs better. Therefore, more data collection is needed to quantify the performances. In further work, it might be a good idea to choose another hyperparameter over LR decay, since it is correlated with the LR itself.

Appendix E

We tried making a GNN and run it on the ice cube dataset. This send us on a week long quest through Windows Subsystem Linux, which turned out to be very useful in other parts of the project, over downloading huge mountains of datasets, and into the gloomy grottos of our gpus.

We were unsuccessful in getting GraphNet to run on the data samples and therefore scraped the data and model idea. However in hindsight we would not have been able to run our tests on the GraphNet model as it would have been to computationally heavy.

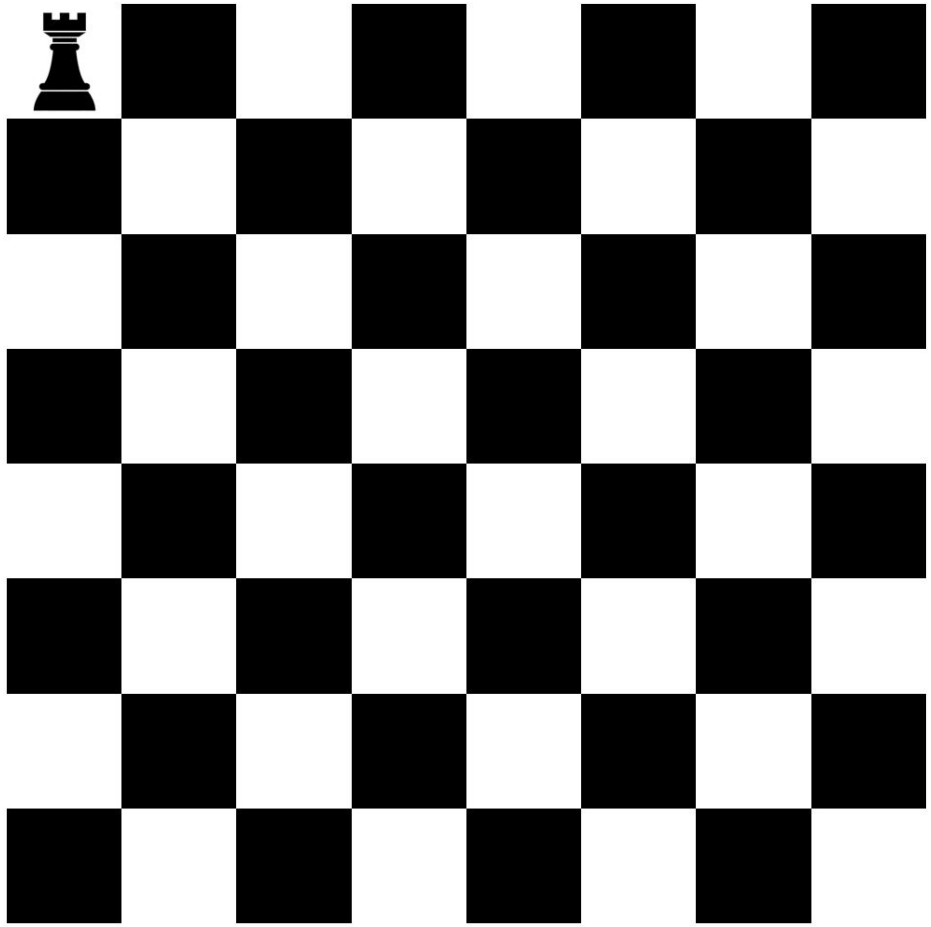
GraphNeT

Graph Neural Networks for
Neutrino Telescope Event Reconstruction

Appendix Chess analogy:
Or how I learned to stop worrying and learned to hyperparameter optimize

Latin Hypercube Sampling (LHS)

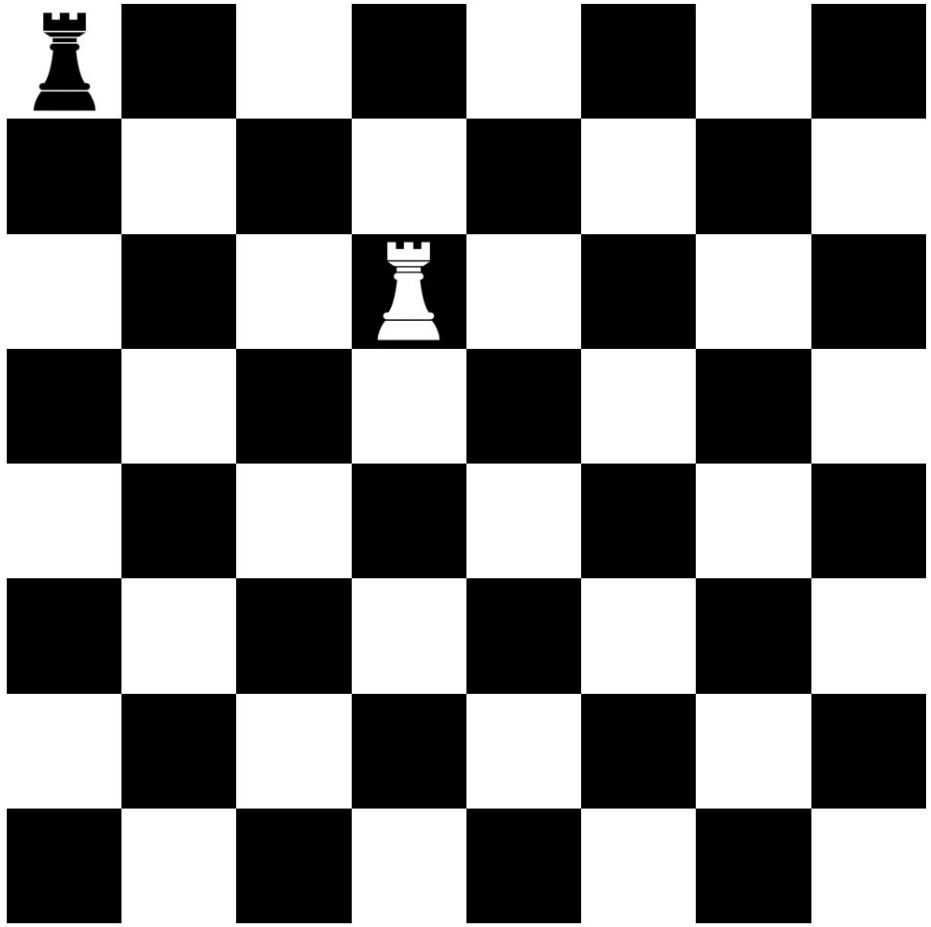
Easiest way of explaining LHS, is imagining a chess board, where you place a rook on some tile. This tile represents your parameter grid point.



Latin Hypercube Sampling (LHS)

Easiest way of explaining LHS, is imagining a chess board, where you place a rook on some tile. This tile represents your parameter grid point.

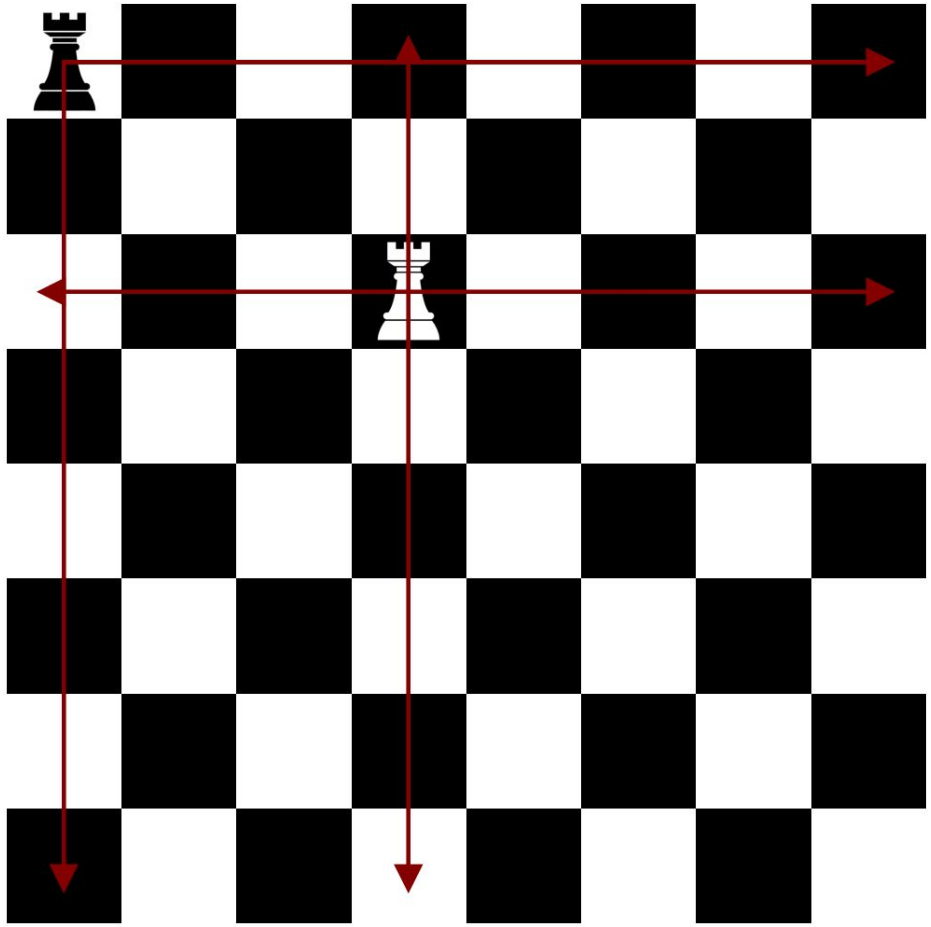
You then place another rook somewhere on the chess board.



Latin Hypercube Sampling (LHS)

Easiest way of explaining LHS, is imagining a chess board, where you place a rook on some tile. This tile represents your parameter grid point.

You then place another rook somewhere on the chess board.

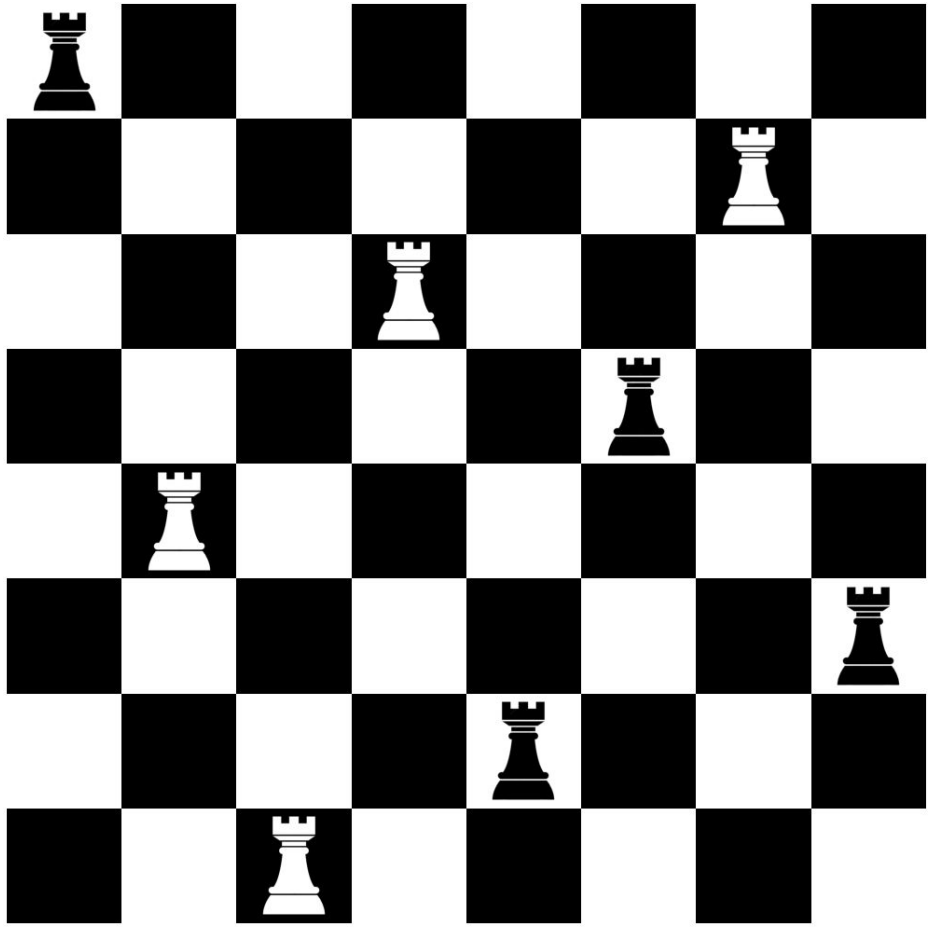


These rooks are <u>NOT</u> allowed to be able to capture each other

Latin Hypercube Sampling (LHS)

Easiest way of explaining LHS, is imagining a chess board, where you place a rook on some tile. This tile represents your parameter grid point.

You then place another rook somewhere on the chess board.

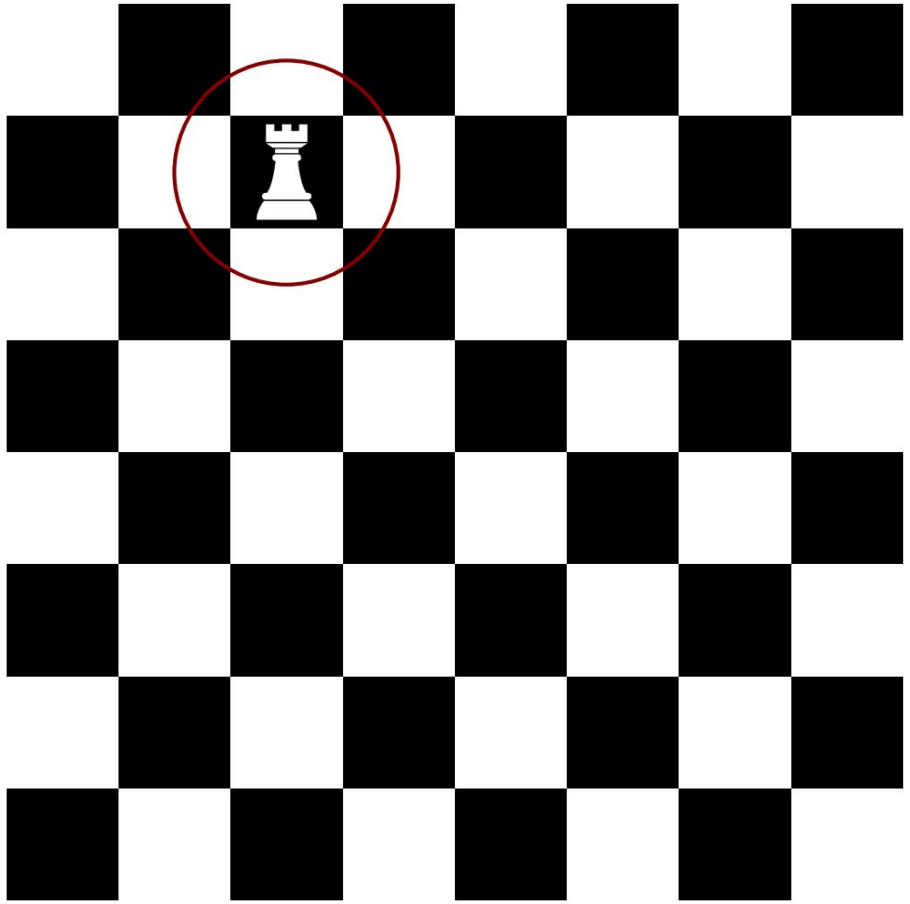


These rooks are <u>NOT</u> allowed to be able to capture each other

Continue until desired number of points has been sampled.

Random sequential addition (RSA)

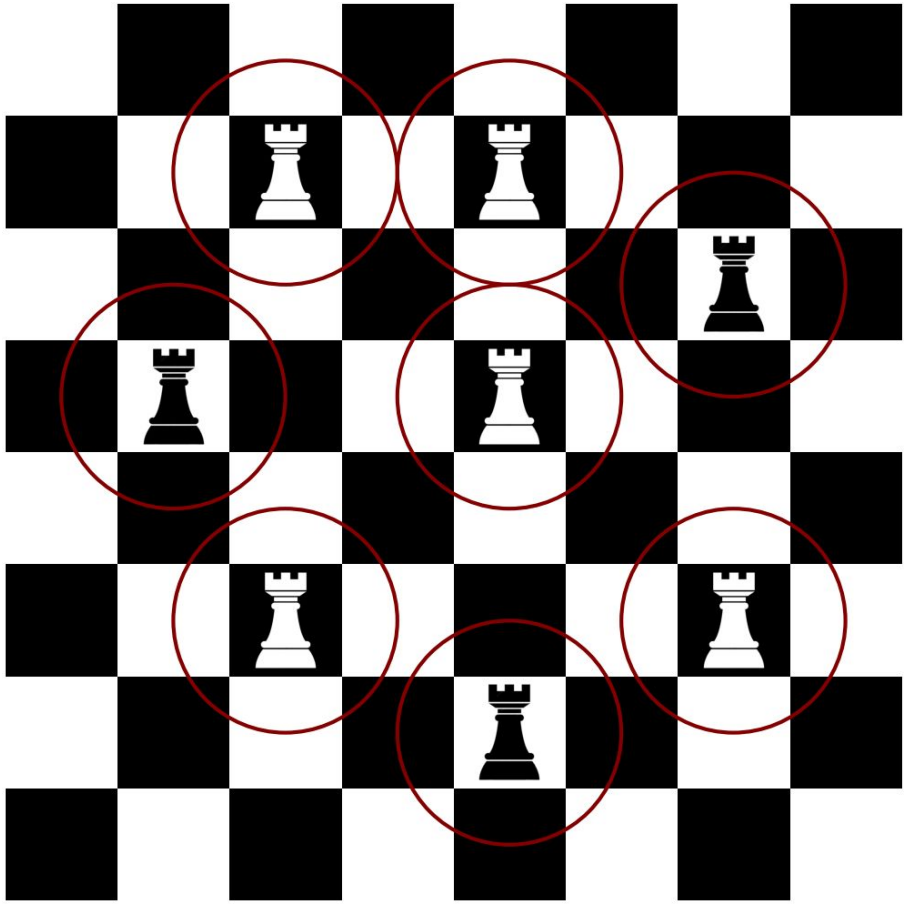
Similarly RSA is done by randomly setting a rook and then drawing a circle (sphere) around.



Random sequential addition (RSA)

Similarly RSA is done by randomly setting a rook and then drawing a circle (sphere) around.

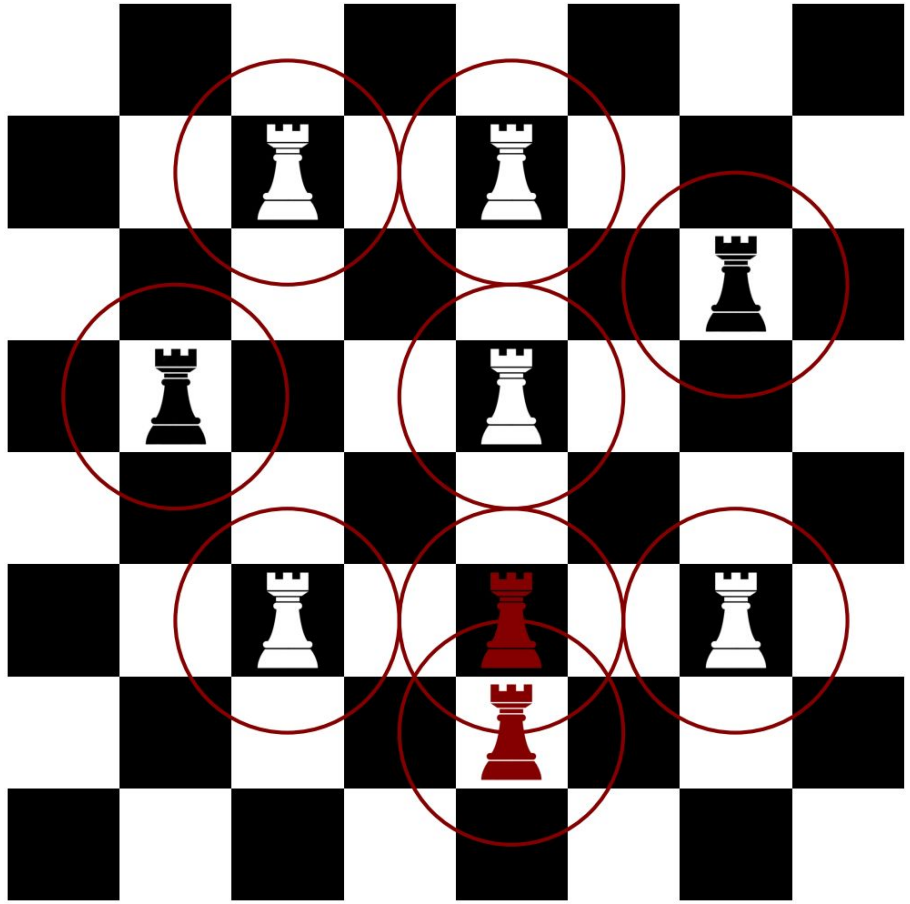
I then maxmizes the radius of the spheres for the number of samples wanted, to maximize the separation.



Random sequential addition (RSA)

Similarly RSA is done by randomly setting a rook and then drawing a circle (sphere) around.

RSA then maxmizes the radius of the spheres for the number of samples wanted, to maximize the separation.

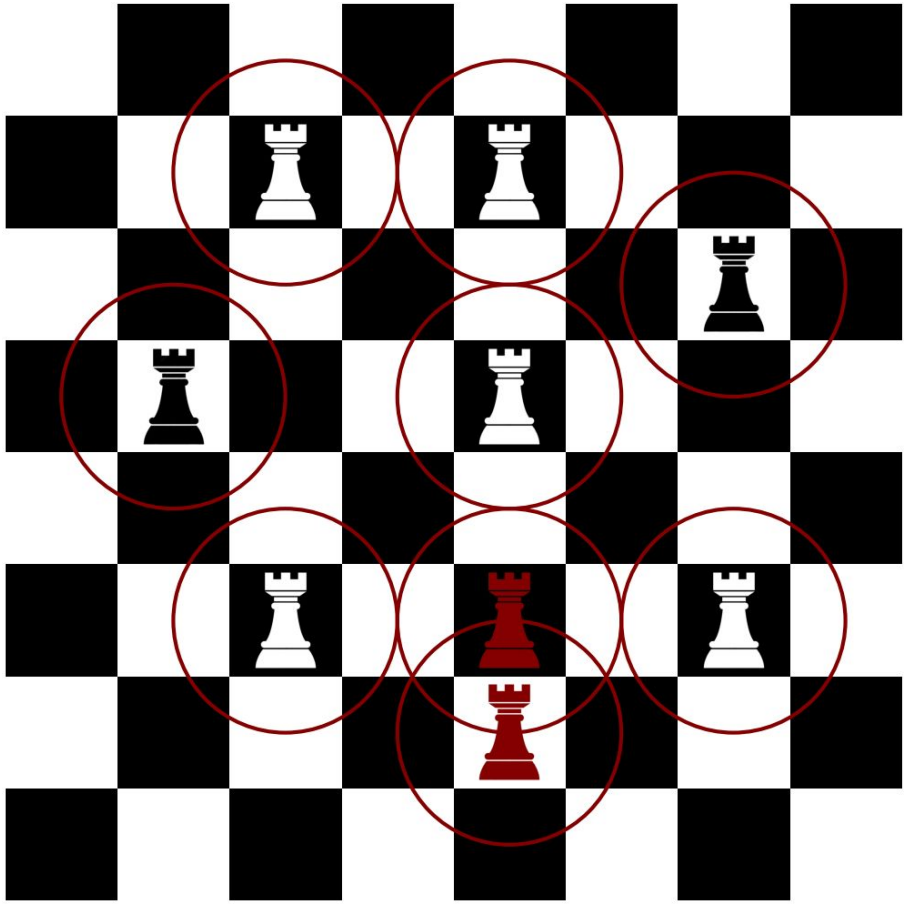


No points can have overlapping spheres!

Random sequential addition (RSA)

Similarly RSA is done by randomly setting a rook and then drawing a circle (sphere) around.

RSA then maxmizes the radius of the spheres for the number of samples wanted, to maximize the separation.



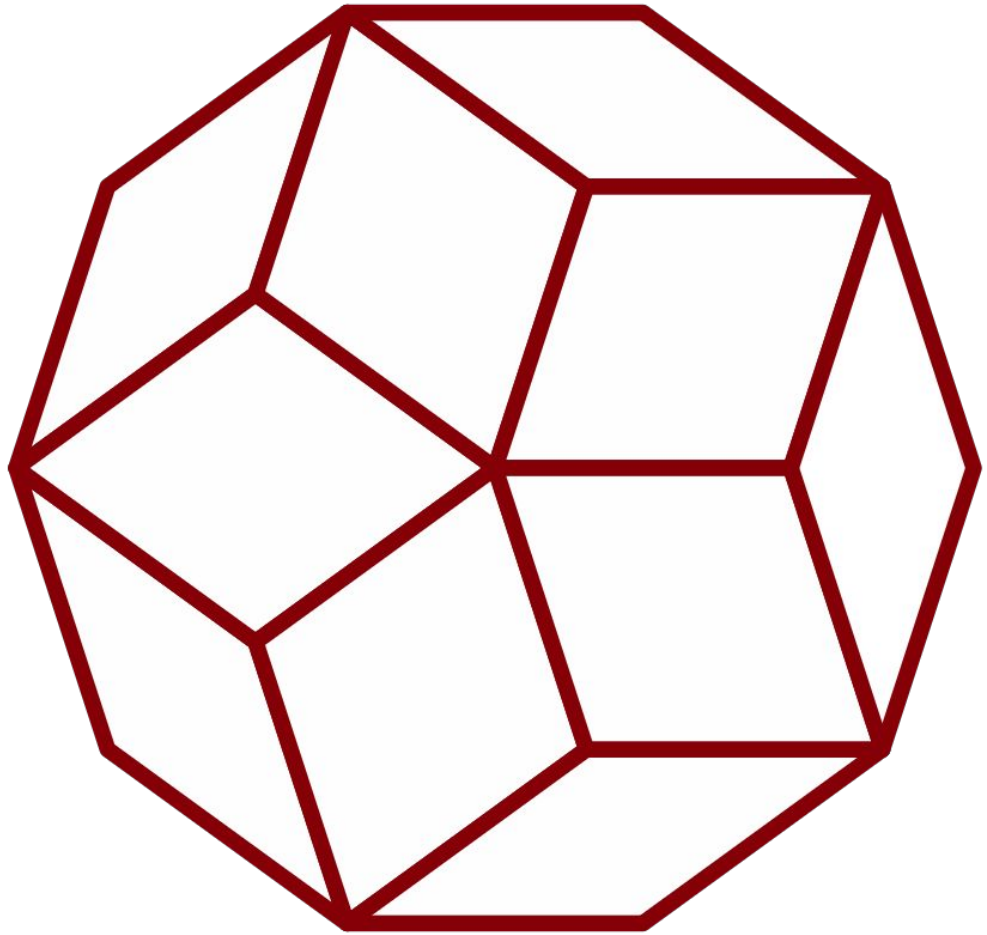
No points can have overlapping spheres!

The RSA methods implemented in Zhang and Torquato 2013 runs more efficiently by removing sampled regions, from the space instead of just rejecting points if sampled in region, can cut down on computing time massively

Zhang, G. & Torquato, S. Precise Algorithm to Generate Random Sequential Addition of Hard Hyperspheres at Saturation. Phys. Rev. (2013).

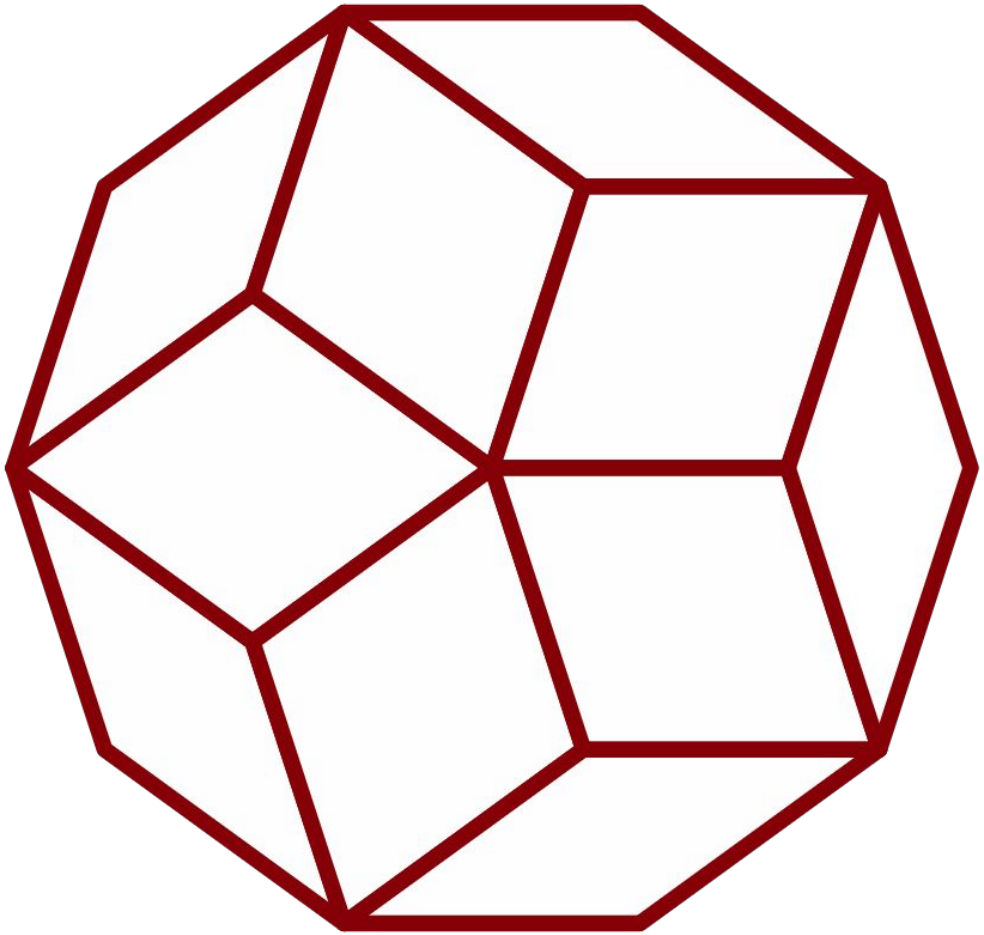
Hypertiling (Penrose tiling)

A Penrose tiling is a type of aperiod tiling. Penrose tiling ensures a diverse sampling of the parameter space, especially in higher dimensions.



Hypertiling (Penrose tiling)

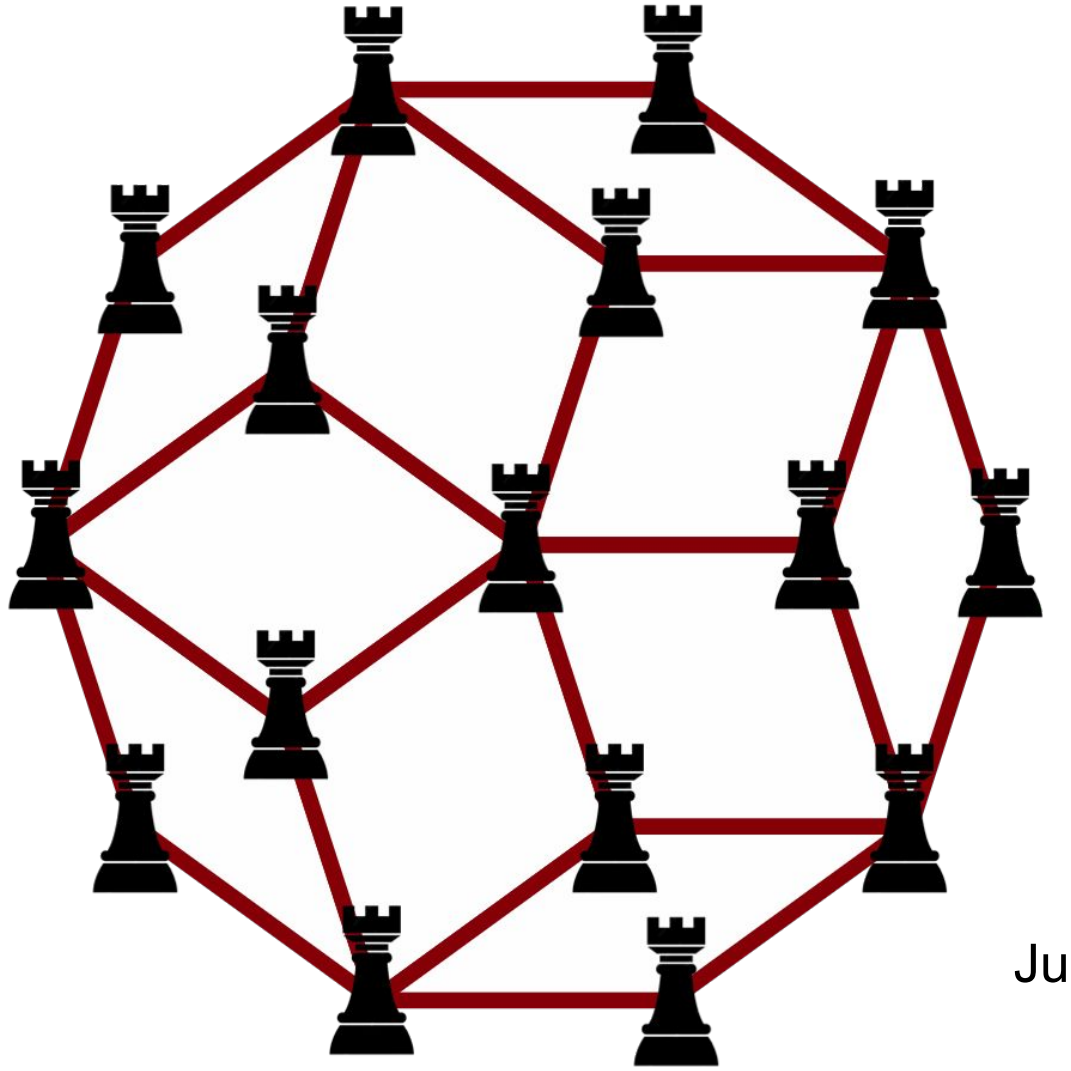
A Penrose tiling is a type of aperiod tiling. Penrose tiling ensures a diverse sampling of the parameter space, especially in higher dimensions.



And for the sampling?

Hypertiling (Penrose tiling)

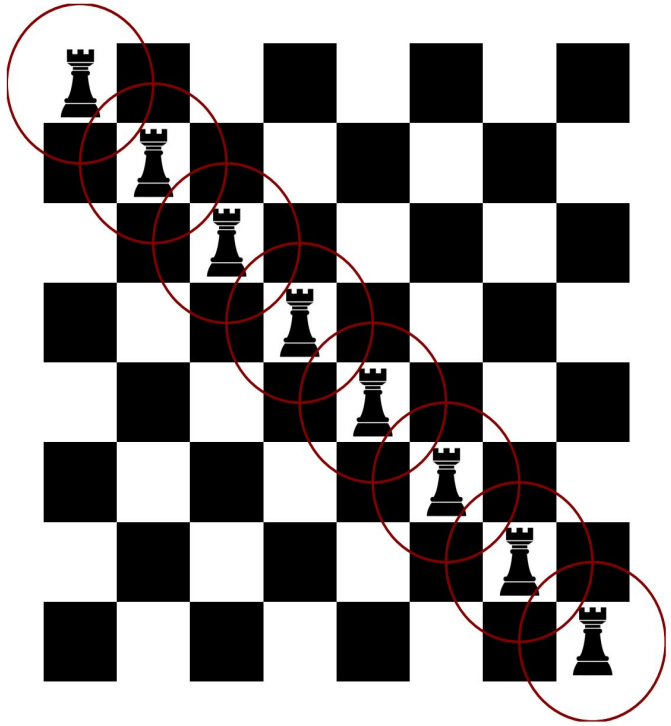
A Penrose tiling is a type of aperiod tiling. Penrose tiling ensures a diverse sampling of the parameter space, especially in higher dimensions.



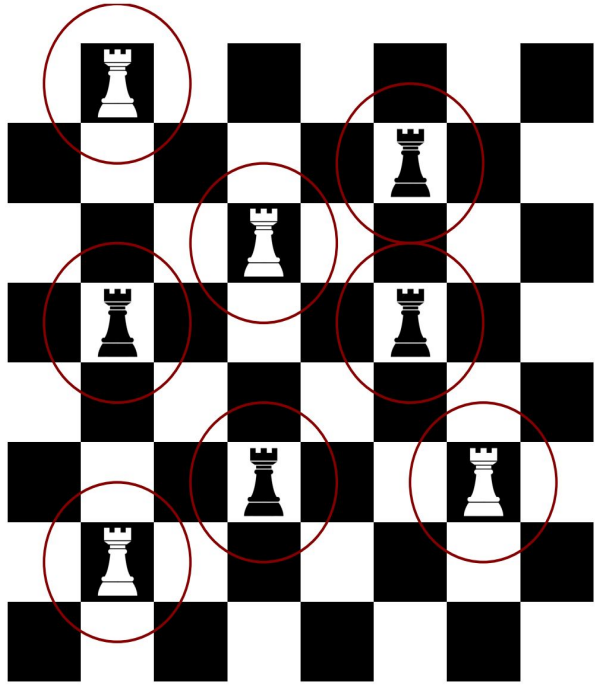
And for the sampling?

Just find the intersections!

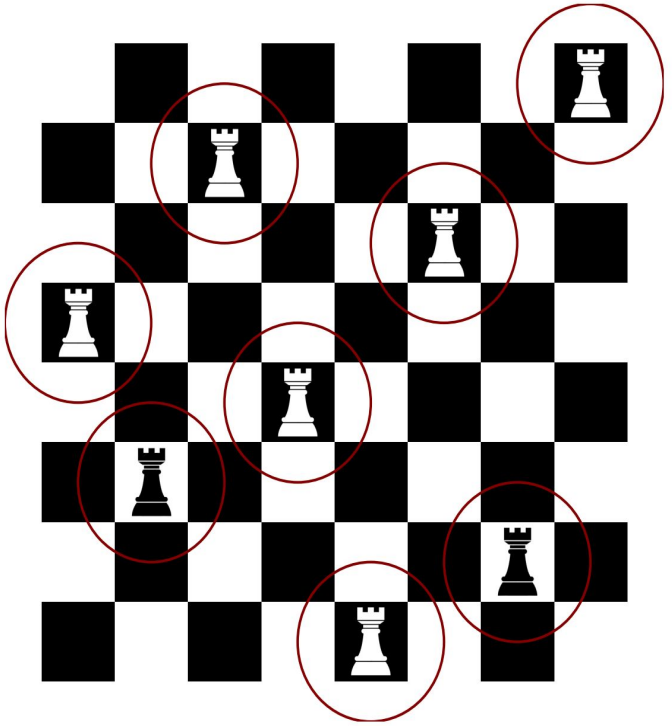
Combining RSA and LHS - Furtherwork



LHS - abiding
RSA - defying



LHS - defying
RSA - abiding



LHS - abiding
RSA - abiding