First results on data augmentation

May 11, 2020

Ghent University: Marek Niedziela, Luka Lambrecht

Contents

Tested different methods of resampling of histograms:

- bin-per-bin noise
- bin-to-bin migrations
- smooth shape noise using fourier components
- randomized linear combinations of similar-looking histograms
- variational autoencoder

Pleased to hear from you:

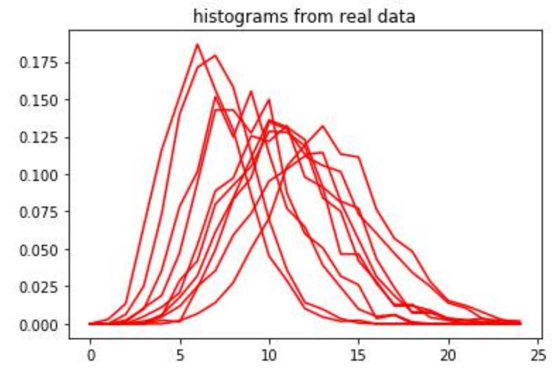
- advantages
- disadvantes
- possible improvements

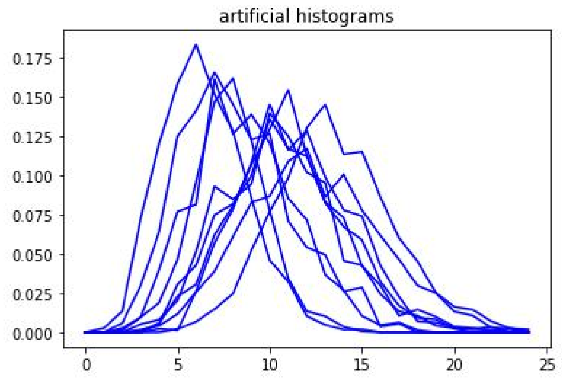
for each method!

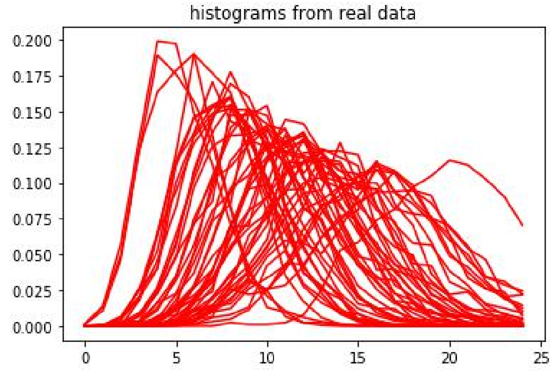
Note: in this presentation, x-axis = bin number, y-axis = arbitrary units

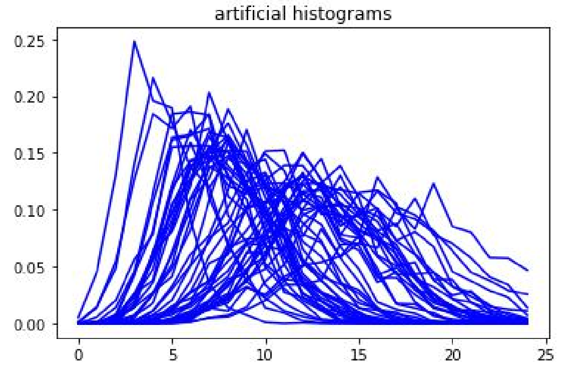
Method 1: bin-per-bin noise

Method 1a: gaussian noise



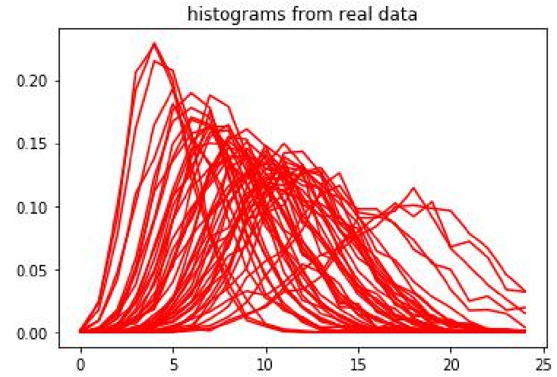


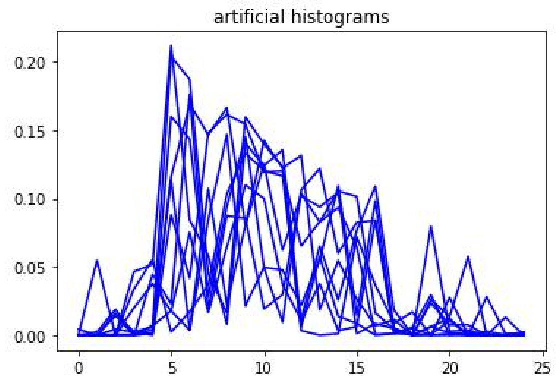




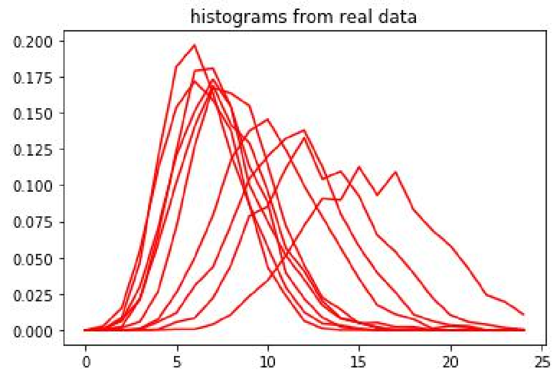
Bins are treated as independent from each other
→ maybe not ideal for histograms with smoother shape.
Amplitude of noise is a free parameter.

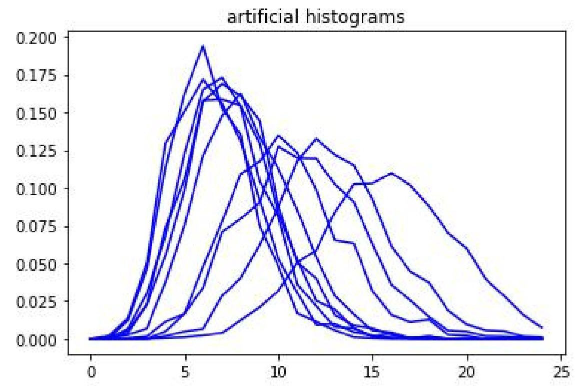
Method 1b: sample from bin-per-bin probability distribution





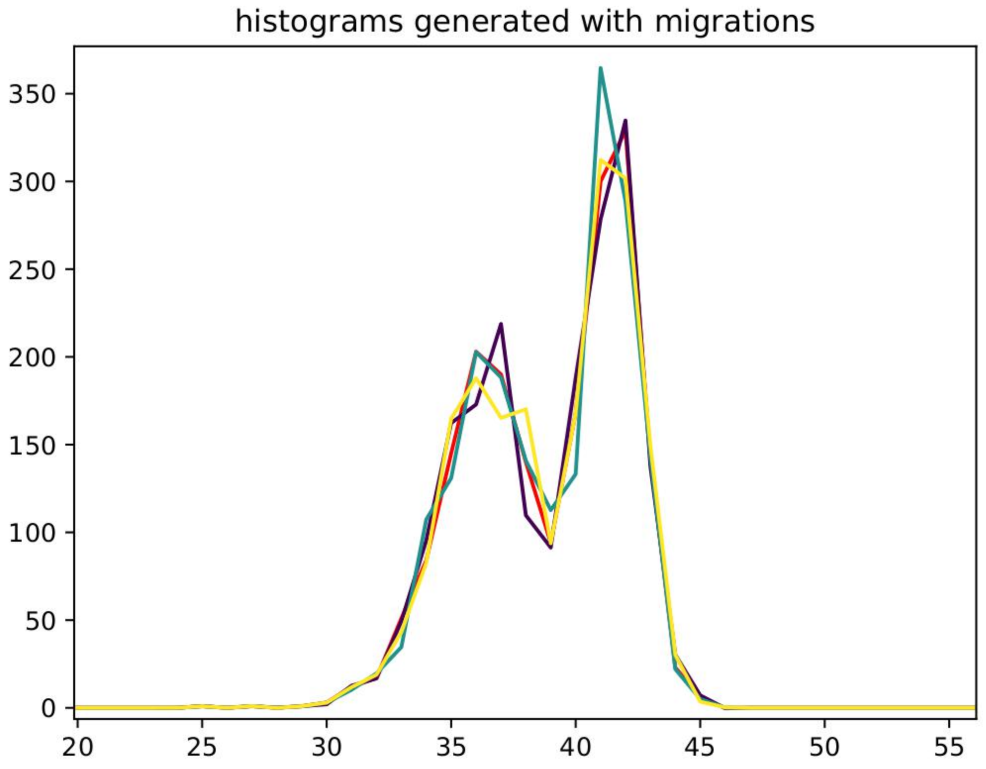
naive: does not take into account possible systematic shifts in histogram shape





improved: resample from only similar-looking histograms → better results

Method 2: migrations



(Examples from histogram type MainDiagonalPosition)

Strategy:

- Start from a single given histogram.
- For each bin:
    - draw a standard normal random value.
    - multiply by transmigration rate (tunable parameter) and bin content.
    - subtract this number from bin, add to next (or previous) bin.

Method 3: “fourier noise”

Strategy:

- input = set of histograms of arbitrary size and number.

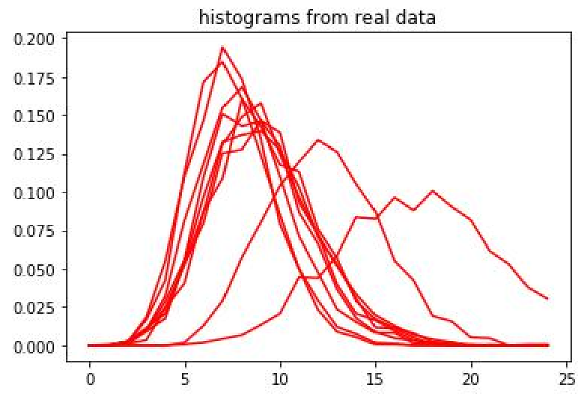
- for the requested number of resampled histograms:

    - pick a random histogram from the input set.
    - scan the entire input set for histograms that look ‘similar’ (based on MSE or momenta).
    - average out these similar histograms.
    - apply ‘noise’ on averagein the form of a random linear combination of sines and cosines + white noise as an extra option.

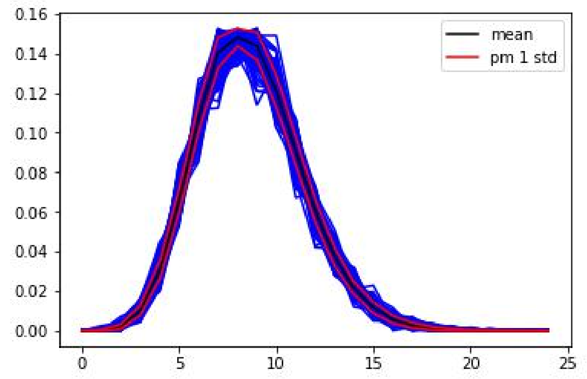
- output = new histogram or new set of histograms with same distribution as input histograms.

Note: very loosely based on the idea that each sufficiently smooth variation can be decomposed in low-frequency fourier components. (More details in backup.)

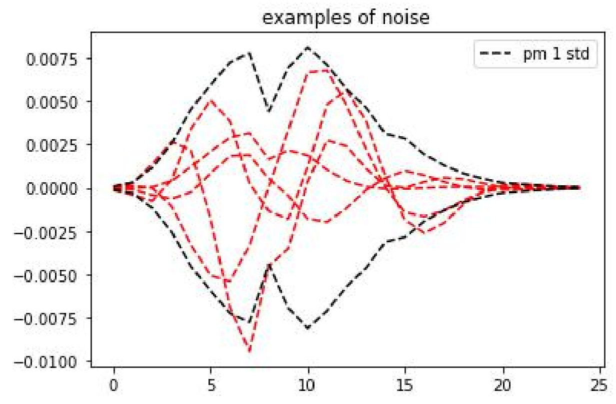
Example: NumberOfClustersInPixel



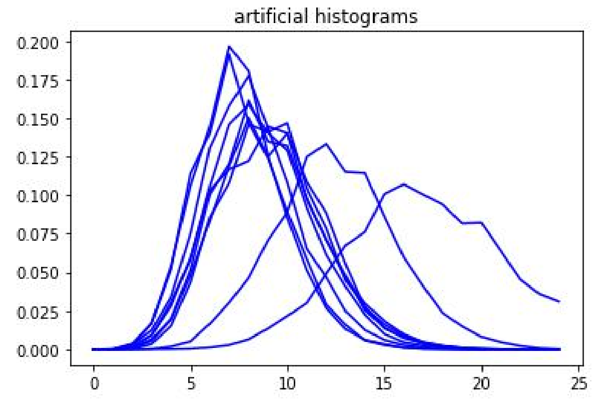
Some example histograms from data.



Pick one histogram and scan the entire set for similar histograms, determine mean and standard deviation.

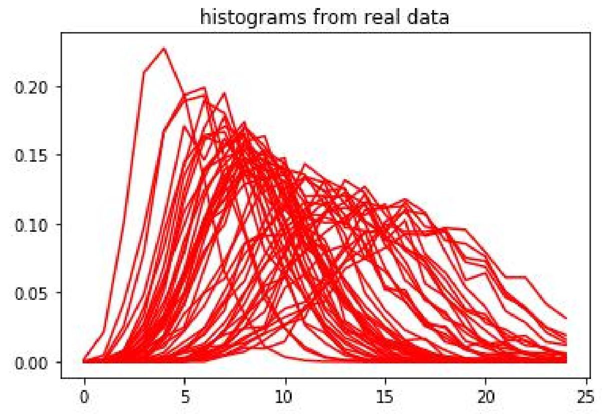


Generate smooth noise and scale amplitude with standard deviation.

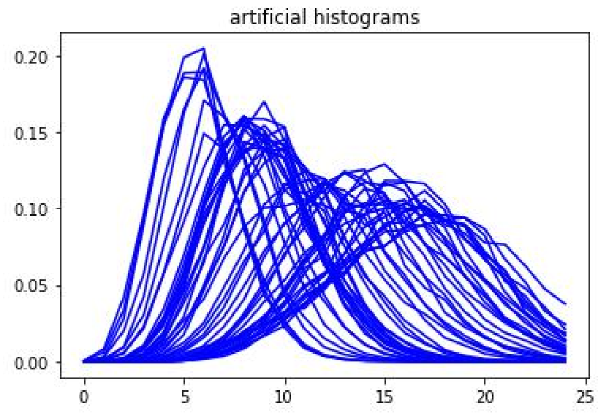


Apply noise on mean. Repeat this procedure for all histograms in the left plot.

Some more examples:



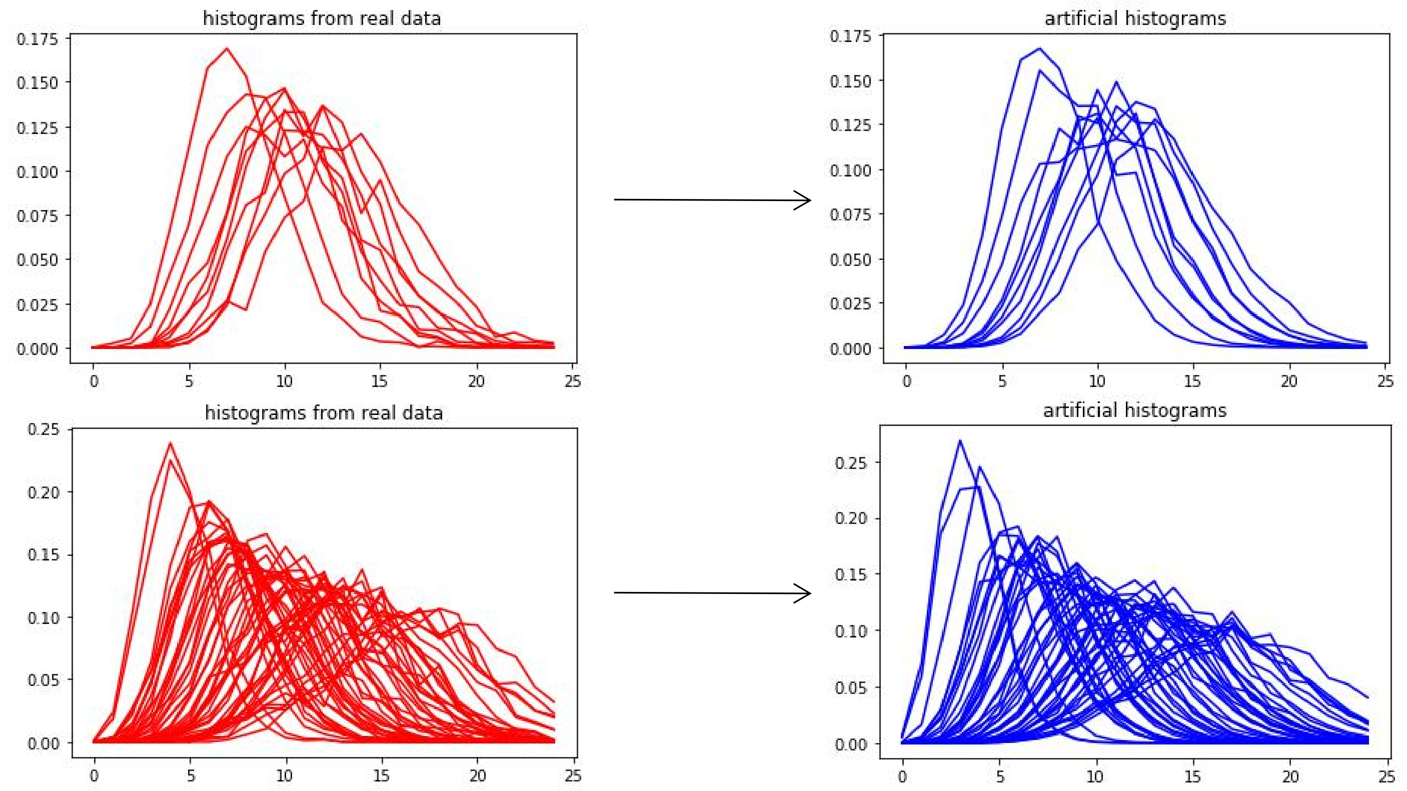
$\longrightarrow$



Method 4: linear combinations

Strategy:

- input = set of histograms of arbitrary size and number
- pick a histogram, look for similar histograms, make a linear combination with randomized coefficients



Method 5: variational autoencoder

Strategy:

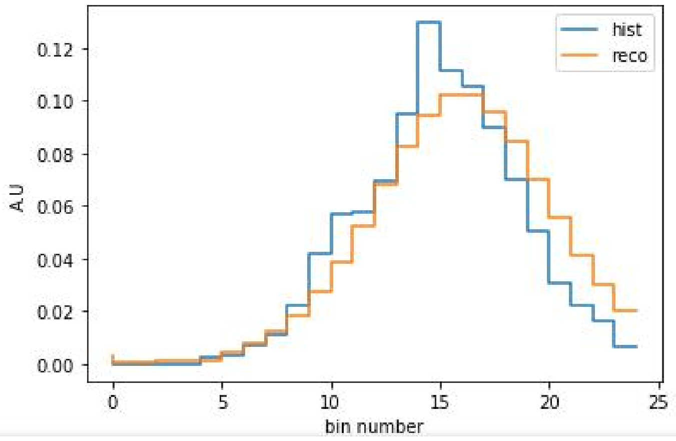
- input = set of histograms of arbitrary size and number.

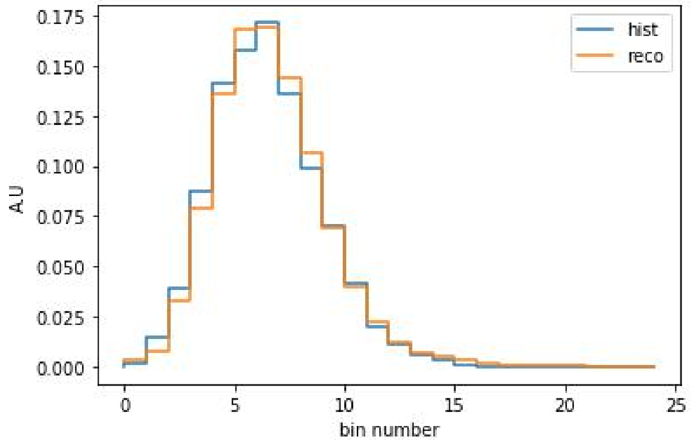
- train autoencoder with additional constraints on points in latent space to ensure smoothness of encoding (e.g. similar histograms are encoded to nearby points).

- sample random points from latent space and apply decoder part to make new histograms.

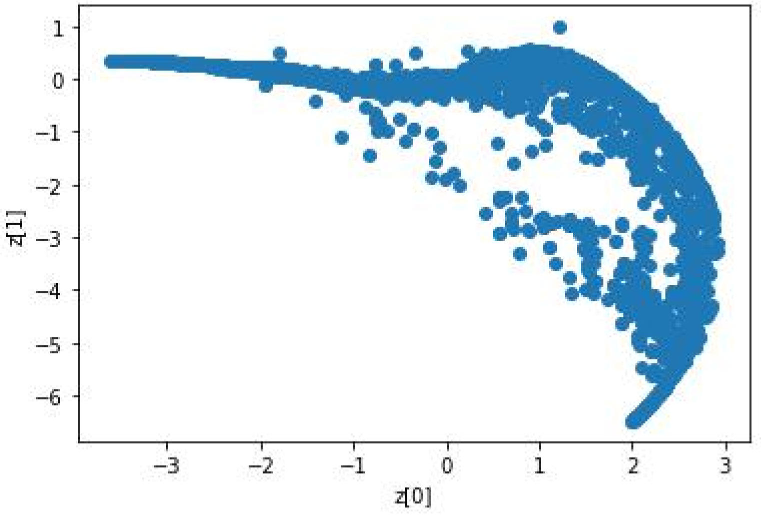
- DISADVANTAGES:
    - so far, no hyperparameter combination was found that works for multiple types of histograms, would need to be retuned for all types.
    - much more complicated and slower than previous method.
    - resampled set is not fully representative of input set. Results are very sensitive to small changes in hyperparameters.

Example: NumberOfClustersInPixel

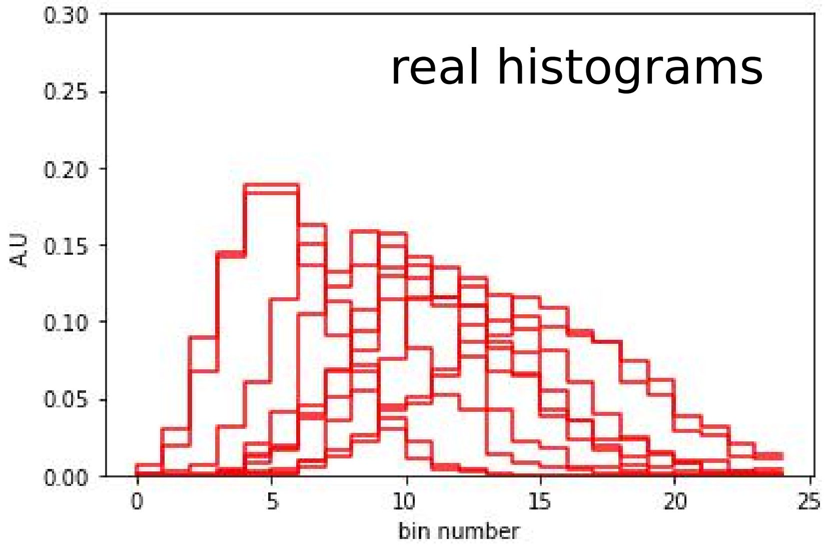


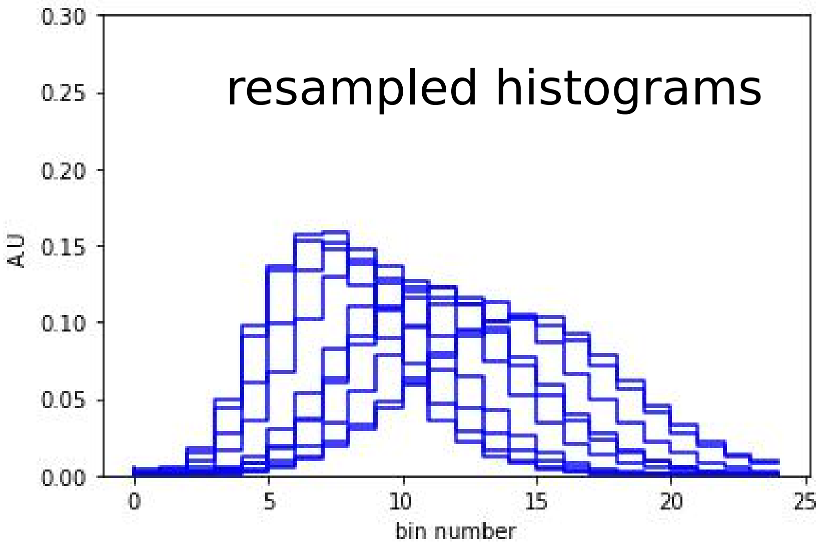


→ compromise between reconstruction accuracy and smoothness in latent space.



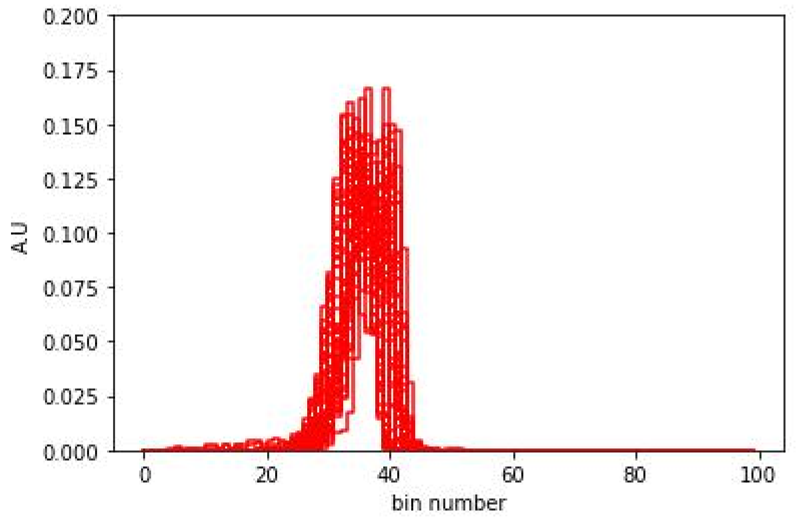
→ systematic shift seems to be present



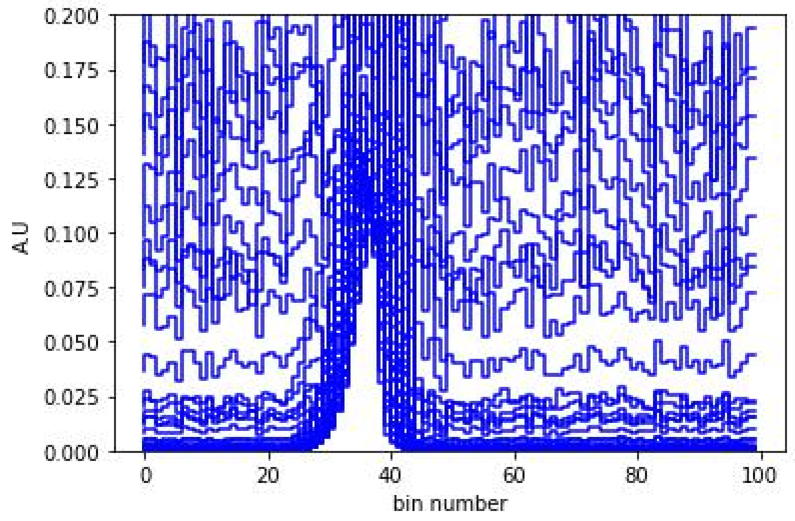


Example: MainDiagonal_Position

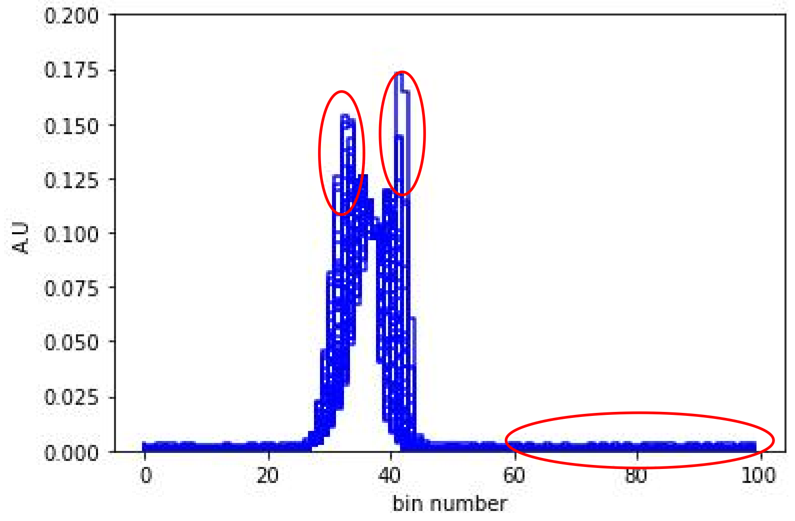
real histograms



resampled histograms



→ unstable, small hyperparameter change can lead to nonsensical results.



→ even with 'best' hyperparameters, resampling is not very accurate.

Conclusions

- Variational autoencoder does not seem a viable option, too much tuning needed.
- All other options give similar results qualitatively.
- First step: given input set of histograms, get output set of similar histograms.
  → seems to be more or less achieved. However, happy to hear your comments or suggestions for improvement.
- Problem: How to go from simple resampling to resampling only good or bad?
  Start from manually picked good and bad histograms?

Preliminary version of code will be put here:
https://github.com/mniedzie/ML4DQM-DC_SharedTools

Backup: some technicalities on fourier noise

- In principle, any discrete function on n points can be rewritten as a sum of frequency components of the form
$$A\ sin(kx + \varphi)$$
where A is the amplitude, k is related to the frequency and φ is the phase.

- φ is chosen randomly between 0 and 2π.
- A is chosen randomly between 0 and 2 (i.e. mean amplitude is 1)
- k is chosen randomly between 0 (constant) and 0.25π
(maximum physically allowed k is π, but scale by arbitrary factor to have only low-frequency components)

- Rescale using a histogram that you can give as an argument, e.g. the bin-per-bin average magnitude of variation that you expect.

- Sum a number of these components (also renormalize so amplitude does not depend on this)
Note: do not use too few or you will get too little variation in shape of noise.
do not use too many or the variations will average out to zero.