Plot2CSV: Automated Curve Digitization Using Synthetic Data and Deep Learning

Garrett Gmeiner & Tyler Ton

Fall 2025

GitHub Repository: https://github.com/ggmeiner22/plot2csv

Abstract

Plot2CSV is a deep learning–based system designed to automate the extraction of numerical values from scientific plot images. Traditional digitization requires manual clicking or semi-automated tools, which become infeasible for large-scale data analysis. Plot2CSV overcomes these limitations by generating synthetic datasets, training neural networks to regress curve values, and applying an end-to-end inference pipeline that outputs 20 predicted y-values for any input plot. This report presents the methodology, system design, experimental results, improvements, and implications for scientific and industrial workflows. The project's full codebase is available on GitHub.

1. Introduction

The extraction of quantitative information from plots is essential in scientific research, engineering analysis, and data-intensive fields. Despite advancements in automation, many researchers still manually digitize plots by clicking points along a curve. This approach is slow, error-prone, and impossible to scale when large datasets or publication corpora must be processed. Plot2CSV was developed to address these challenges by leveraging synthetic data, neural networks, and controlled rendering pipelines to automate plot digitization. The core objective of Plot2CSV is to provide a fully automated system capable of predicting 20 evenly spaced curve values directly from a single grayscale plot image.

A critical advantage of the system is its reliance on synthetic data, which allows complete control over the types of curves generated, their parameter ranges, and their noise characteristics. Unlike traditional datasets, synthetic data provides perfect ground

truth and
eliminates labeling errors. This enables efficient training and evaluation across a diverse set of curve families, ultimately improving model generalization. The following sections detail the motivation, methodology, architecture, experimental results, improvements, and broader implications of Plot2CSV.

2. Background and Motivation

Manual digitization has remained the default method for extracting curve data from published figures for decades. Although tools exist to semi-automate the process, they typically require substantial human supervision, introducing variability and limiting throughput. Plot2CSV
addresses these limitations by replacing human annotation with a deep learning regression model trained entirely on synthetic images.

Synthetic data provides several key benefits. First, it ensures the availability of large, high-quality datasets without the need for manual labeling. Second, it allows precise control over curve shapes, sampling resolutions, and stylistic variations. Third, by simulating noise, distortions, and parameter diversity, synthetic data enables the model to generalize effectively to real-world inputs. This approach is increasingly common in domains where labeled data is scarce or expensive to produce.

Plot2CSV builds upon these principles by generating six families of curves:linear, quadratic, exponential, logarithmic, sinusoidal, and sigmoid. Each represents a different functional behavior commonly seen in scientific literature. By training on a broad set of such functions, the model becomes capable of accurately reconstructing complex curves, including those containing noise or nonlinear behavior.

2.1 Curve families

Curve Type	Equation Form	Parameters
Linear	$y = a\,x + b$	a, b
Quadratic	$y = a\,x^2 + b\,x + c$	a, b, c
Exponential	$y = a\,e^{(b\,x)}$	a, b

Logarithmic	$y = a \log(b\,x + c)$	a, b, c
Sinusoid	$y = a \sin(b\,x + c)$	a, b, c
Sigmoid	$y = 1 / (1 + e^{(-a(x - b))})$	a, b

3. Methods

Synthetic data generation forms the foundation of Plot2CSV. Each sample consists of a mathematical curve rendered into a 256×256 antialiased image, later downsampled to 50×50 grayscale for neural network input. For each image, 20 evenly spaced x-values are sampled from the underlying curve function. These y-values are normalized using min-max scaling to provide stable training labels between 0 and 1.

Each curve family includes tunable parameters. For instance, linear curves vary in slope and intercept, while quadratics vary in curvature and orientation. Exponential and logarithmic curves are parameterized to ensure both positive and negative trends. Sigmoids differ in their steepness and midpoint, while sinusoids include tunable amplitude, phase, and frequency.

A major improvement was the introduction of controlled sinusoid period ranges. Early versions allowed random frequencies, producing overly rapid oscillations that were difficult for the model to learn. By constraining periods between 8 and 20 units, the model achieved significantly more stable performance on sinusoidal curves.

Rendering is performed at high resolution to ensure smooth curves and clear markers. Each curve is drawn using antialiased polylines and 20 circular markers at the sampled x-positions. The high-resolution image is then downsampled to 50×50 pixels, preserving essential information while reducing computational complexity. Both the rendered image and its corresponding normalized y-values are saved into a dataset, ensuring reproducibility.

Two architectures were evaluated: a custom convolutional neural network and a modified ResNet-18. The custom CNN includes three convolutional stages followed by fully connected layers producing 20 regression outputs. While effective for simpler curves, it struggled with oscillatory and nonlinear curves.

ResNet-18 provided superior performance. To adapt it for this task, several modifications were made: the initial convolution was changed to single-channel input,

early max-pooling was removed to preserve spatial details, and the final classification layer was replaced with a 20-output regression head. These adjustments made ResNet-18 highly effective for low-resolution regression tasks.

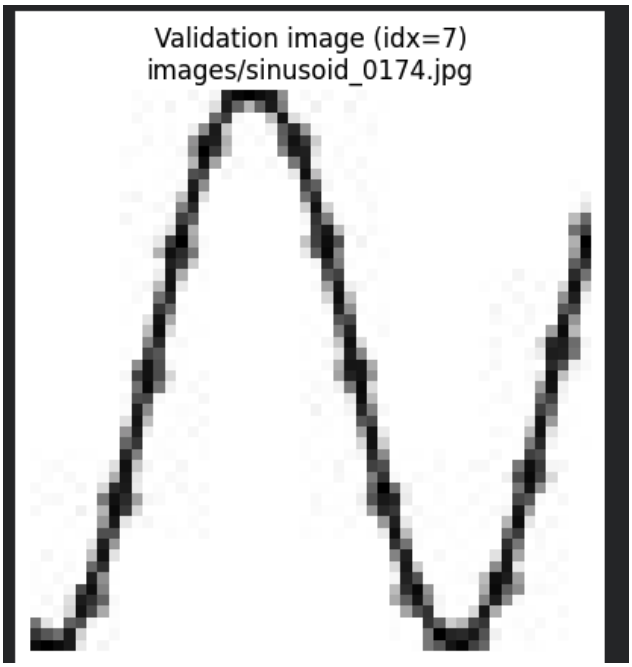
Training used an 85/15 train/validation split with MSE loss and the Adam optimizer at a learning rate of 3e-4. Across 20–50 epochs, the model demonstrated strong convergence for most curve families. Evaluation metrics included MAE, RMSE, R^2, and hit-rate thresholds for ±0.5, ±1.0, and ±2.0 axis units. These metrics provided a comprehensive picture of both point-wise accuracy and overall curve reconstruction fidelity.

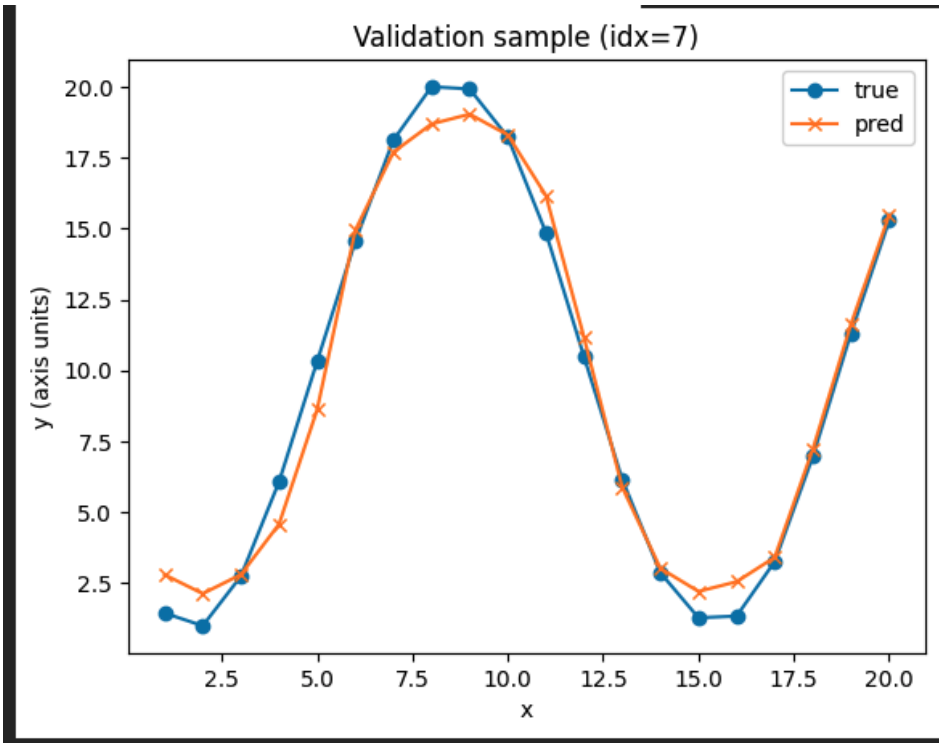
4. Results

Overall performance was strong across all six curve families. Linear and logarithmic curves achieved the lowest errors due to their monotonic nature. Sigmoid curves also performed well. Quadratic curves required parameter tuning to avoid degenerate shapes but ultimately showed good performance. Sinusoidal curves, initially the most difficult, improved dramatically after period constraints were introduced. The R^2 scores indicated strong alignment between predicted and ground-truth values, and hit-rate metrics exceeded 95% for ±1.0 thresholds in most curve families. For further results, view Appendix B: Metrics.

Increasing the synthetic dataset size also improved generalization by reducing overfitting. With approximately 400 samples per curve type, the model achieved greater robustness and stability.

The images below show an example of a synthetic data image with a sinusoidal line along with a validation sample comparing the predicted line with the true line values. Additional images for the other 5 families can be found in Appendix A: Synthetic Data Compared to Predicted Values.





5. Discussion

Plot2CSV demonstrates the effectiveness of synthetic data paired with deep regression models for scientific digitization tasks. The use of controlled rendering, normalization, and parameter tuning significantly improved the model's ability to generalize. One notable finding is that architectural choices strongly influence performance on nonlinear curves. The ResNet-18 backbone provided improved feature extraction, especially for oscillatory behavior in sinusoidal curves.

Limitations remain. The model currently assumes a single curve per image and fixed axis alignment. Real-world plots may contain multiple curves, legends, gridlines, or noise. Expanding Plot2CSV to handle such complexities is a promising direction for future work. Additional
improvements could include domain adaptation to real scientific figures, OCR integration for axis labels, and multi-curve extraction.

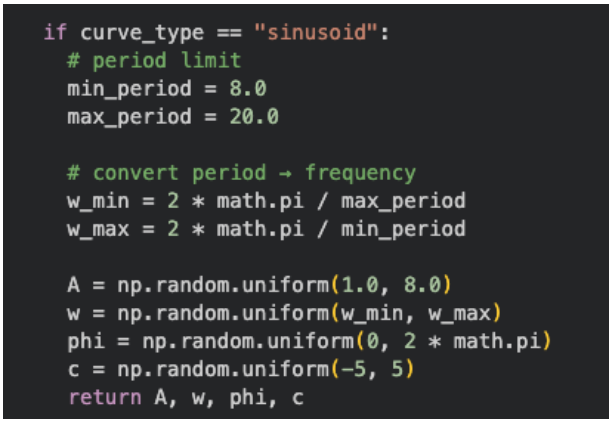
6. Improvements

Throughout the development of Plot2CSV, several key improvements were implemented to enhance data quality, model accuracy, and overall system robustness. These refinements were informed by observed error patterns, validation metrics, and early prototype limitations.

6.1 Sinusoid Period Control

Originally, sinusoidal curves were generated with fully random frequencies, resulting in extreme variations in oscillation rate. This caused the model to struggle with learning consistent sinusoid features. To address this, we:

- Introduced a controlled period range (8–20 units)
- Converted period constraints into a frequency interval
- Ensured all sinusoids have realistic, learnable oscillation behavior
- This led to a significant reduction in MAE and improved hit rates for the sinusoid family.



```
if curve_type == "sinusoid":
  # period limit
  min_period = 8.0
  max_period = 20.0

  # convert period → frequency
  w_min = 2 * math.pi / max_period
  w_max = 2 * math.pi / min_period

  A = np.random.uniform(1.0, 8.0)
  w = np.random.uniform(w_min, w_max)
  phi = np.random.uniform(0, 2 * math.pi)
  c = np.random.uniform(-5, 5)
  return A, w, phi, c
```

6.2 Parameter Range Refinement

Several curve families produced shapes that overlapped visually or contained pathological parameter combinations. To fix this, we:

- Reduced the curvature range of quadratics
- Limited exponential growth rates
- Stabilized sigmoid steepness and horizontal shift
- Ensured logarithmic curves remain monotone within the image bounds

These changes reduced ambiguity between families and improved classification-like separability.

6.3 Normalization Clarification and Alignment

A clearer separation of normalization roles was implemented:

- [1,20] axis scaling for visual rendering
- [0,1] normalization for neural network training

This ensures consistent plotting while guaranteeing numerical stability during training.

6.4 Transition to ResNet-18 Backbone

The custom CNN initially underfit complex curves. We improved model performance by:

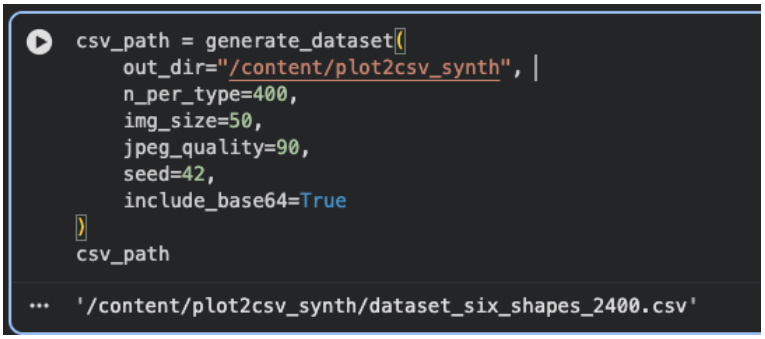
- Integrating a modified ResNet-18 for 50×50 grayscale images
- Removing early maxpooling layers to preserve spatial resolution
- Replacing the final FC layer with a 20-output sigmoid regression head

ResNet-18 offered better feature extraction and robustness across curve families.

6.5 Larger Dataset Size

We improved model performance by generating a larger synthetic dataset. The original dataset (about 200 per sample) was too small to capture the full variety of curve shapes, especially for harder families like sinusoids. By increasing the number of samples to 400 samples per curve type, the model benefited in several ways:

- Better generalization to unseen plots
- Less overfitting during training
- More coverage of different curve shapes and parameter combinations
- Higher accuracy, especially for sinusoidal curves



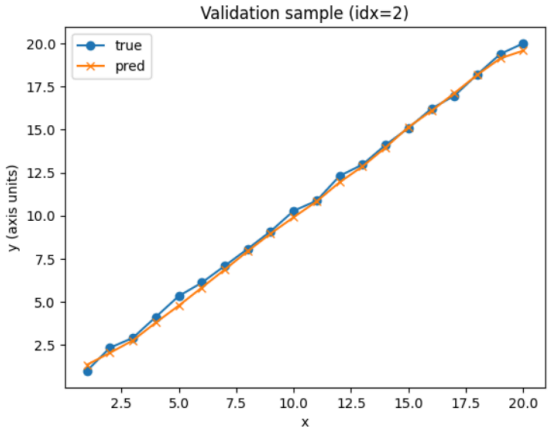
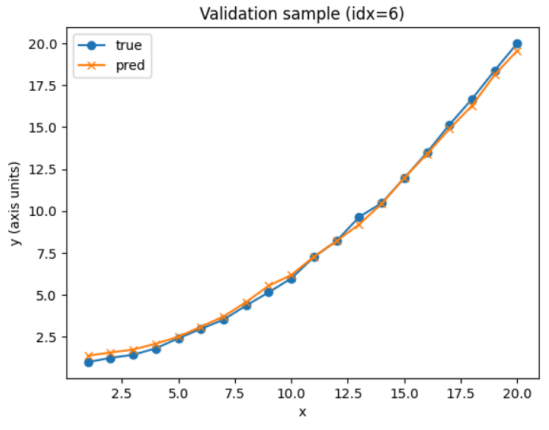
7. Conclusion

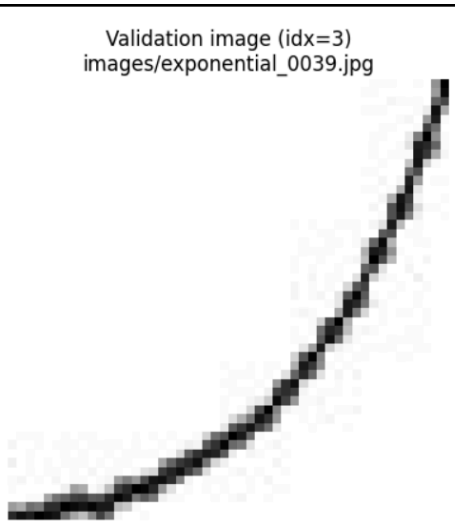
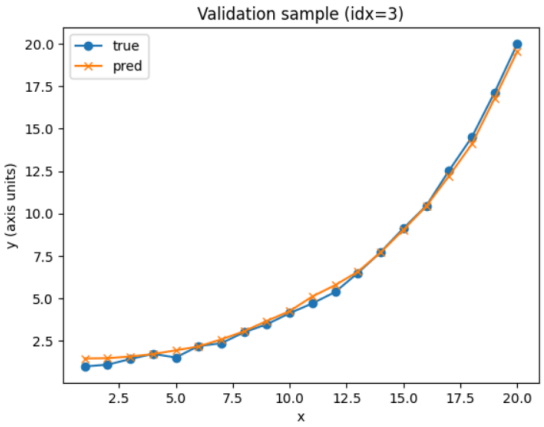
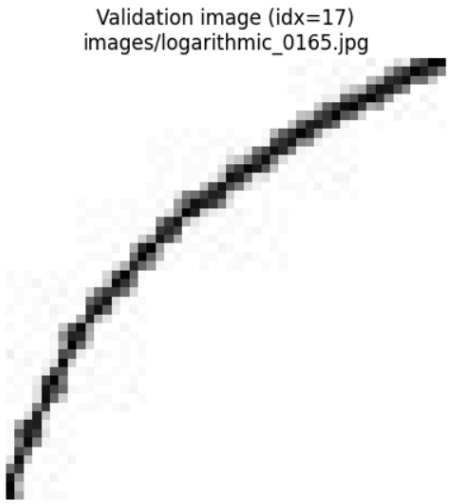
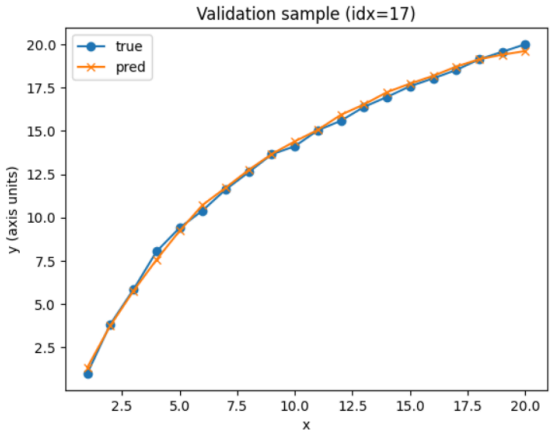
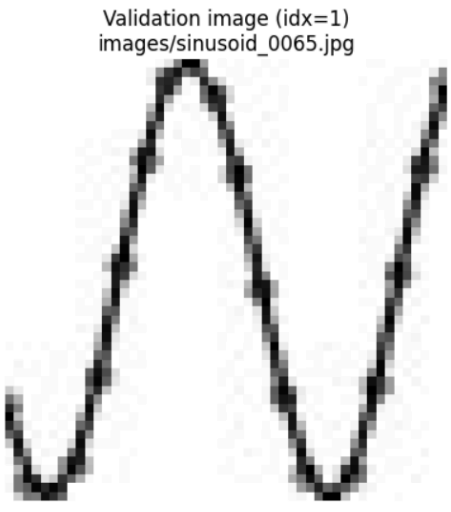
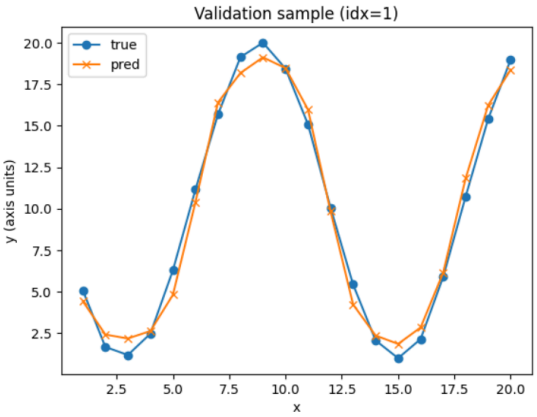
Plot2CSV provides a scalable, accurate, and completely automated approach to extracting numerical data from plot images. By utilizing synthetic data generation, robust neural architectures, and a well-designed normalization and rendering pipeline, the system achieves high performance across diverse curve families. This work establishes a strong foundation for automated figure digitization and highlights the potential of synthetic data for scientific AI applications. Future extensions could broaden its applicability to real-world literature mining, engineering workflows, and automated data ingestion systems.

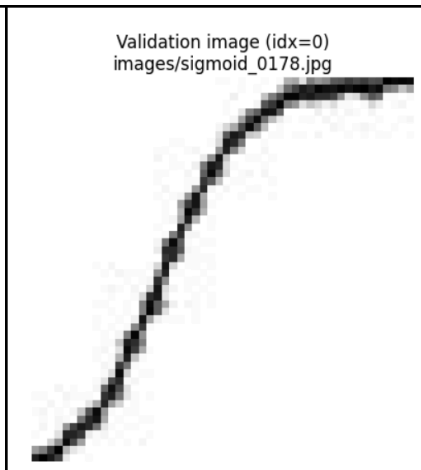
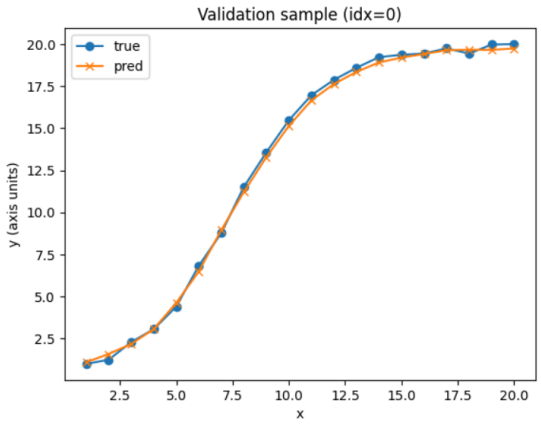
References

[1] ResNet-18 Architecture: K. He, X. Zhang, S. Ren, J. Sun, “Deep Residual Learning for Image Recognition,” 2015.
[2] Synthetic Data in Machine Learning: N. Nikolaidis, “The Role of Synthetic Data in Training Deep Models,” IEEE, 2022.
[3] GitHub Repository for Plot2CSV: https://github.com/ggmeiner22/plot2csv

Appendix A: Synthetic Data Compared to Predicted Values

Curve Type	Synthetic Data Image	Validation Sample
Linear		
Quadratic (We are modeling one quadrant only so quadratic resembles exponential)		

Exponential		
Logarithmic		
Sinusoid		

Sigmoid		

Appendix B: Metrics

Below, we show 20 epochs of training and the metrics calculated throughout.

```
...
=== Overall (from your evaluate_val) ===
        MAE_axis: 0.3155
       RMSE_axis: 0.4714
         R2_axis: 0.9951
       hit@±0.5: 0.8459
       hit@±1.0: 0.9427
       hit@±2.0: 0.9943
  curve_hit@±0.5: 0.3837
  curve_hit@±1.0: 0.8062
  curve_hit@±2.0: 0.9207

=== Per-family performance (means) ===
        family  MAE_axis  RMSE_axis   R2_axis  hit@±0.5  hit@±1.0  hit@±2.0  \
2  logarithmic  0.177098   0.219219  0.997789  0.992308  0.998077  1.000000
1       linear  0.182234   0.227646  0.998408  0.973529  1.000000  1.000000
0  exponential  0.214305   0.257650  0.997847  0.948148  1.000000  1.000000
4      sigmoid  0.218050   0.261608  0.998633  0.960714  1.000000  1.000000
3    quadratic  0.342549   0.406703  0.993540  0.809211  0.965789  0.994737
5     sinusoid  0.773111   0.933276  0.979470  0.387037  0.661111  0.972222

   curve_hit@±0.5  curve_hit@±1.0  curve_hit@±2.0
2        0.846154        0.961538        1.000000
1        0.588235        1.000000        1.000000
0        0.259259        1.000000        1.000000
4        0.642857        1.000000        1.000000
3        0.052632        0.815789        0.921053
5        0.000000        0.000000        0.629630
```

```
epochs = 20
for ep in range(1, epochs+1):
    tr = run_epoch(train_loader, train=True)
    va = run_epoch(val_loader,   train=False)
    val_metrics = evaluate_val(val_loader)
    print(f"epoch {ep:02d} | train MSE {tr:.5f} | val MSE {va:.5f} | "
          f"MAE_axis {val_metrics['MAE_axis']:.3f} | hit@±1 {val_metrics['hit@±1.0']:.2%} | "
          f"curve_hit@±1 {val_metrics['curve_hit@±1.0']:.1%}")
```

```
epoch 01 | train MSE 0.05709 | val MSE 0.04325 | MAE_axis 3.171 | hit@±1 17.44% | curve_hit@±1 0.0%
epoch 02 | train MSE 0.01552 | val MSE 0.01382 | MAE_axis 1.607 | hit@±1 41.96% | curve_hit@±1 0.0%
epoch 03 | train MSE 0.00472 | val MSE 0.00406 | MAE_axis 0.878 | hit@±1 69.11% | curve_hit@±1 6.7%
epoch 04 | train MSE 0.00254 | val MSE 0.00197 | MAE_axis 0.666 | hit@±1 80.91% | curve_hit@±1 15.7%
epoch 05 | train MSE 0.00165 | val MSE 0.00125 | MAE_axis 0.483 | hit@±1 91.21% | curve_hit@±1 70.0%
epoch 06 | train MSE 0.00136 | val MSE 0.00105 | MAE_axis 0.431 | hit@±1 91.21% | curve_hit@±1 77.9%
epoch 07 | train MSE 0.00132 | val MSE 0.00104 | MAE_axis 0.435 | hit@±1 91.54% | curve_hit@±1 78.2%
epoch 08 | train MSE 0.00142 | val MSE 0.00114 | MAE_axis 0.485 | hit@±1 89.60% | curve_hit@±1 50.1%
epoch 09 | train MSE 0.00134 | val MSE 0.00107 | MAE_axis 0.425 | hit@±1 91.02% | curve_hit@±1 74.0%
epoch 10 | train MSE 0.00119 | val MSE 0.00078 | MAE_axis 0.367 | hit@±1 93.17% | curve_hit@±1 77.2%
epoch 11 | train MSE 0.00108 | val MSE 0.00067 | MAE_axis 0.313 | hit@±1 94.01% | curve_hit@±1 80.5%
epoch 12 | train MSE 0.00102 | val MSE 0.00066 | MAE_axis 0.343 | hit@±1 94.20% | curve_hit@±1 73.0%
epoch 13 | train MSE 0.00108 | val MSE 0.00071 | MAE_axis 0.366 | hit@±1 93.50% | curve_hit@±1 66.0%
epoch 14 | train MSE 0.00098 | val MSE 0.00067 | MAE_axis 0.336 | hit@±1 94.32% | curve_hit@±1 69.4%
epoch 15 | train MSE 0.00089 | val MSE 0.00051 | MAE_axis 0.289 | hit@±1 94.93% | curve_hit@±1 80.5%
epoch 16 | train MSE 0.00088 | val MSE 0.00049 | MAE_axis 0.291 | hit@±1 95.67% | curve_hit@±1 79.4%
```

```
model.eval()
```