Electronic Supplementary Material (ESI) for Soft Matter.

Electronic Supplementary Information: Tracking Janus microswimmers in 3D with Machine Learning

Maximilian R. Bailey,*[a], Fabio Grillo, and L. Isa*[a]

[a]Laboratory for Soft Materials and Interfaces, Department of Materials, ETH Zürich, Vladimir-Prelog-Weg 5, 8093 Zürich, Switzerland

*To whom correspondence should be addressed:

E-mail: maximilian.bailey@mat.ethz.ch;

E-mail: lucio.isa@mat.ethz.ch

The PDF file includes:

Traditional Machine Learning models: Pt-SiO_2 Janus particle system

Image Properties and Feature Extraction

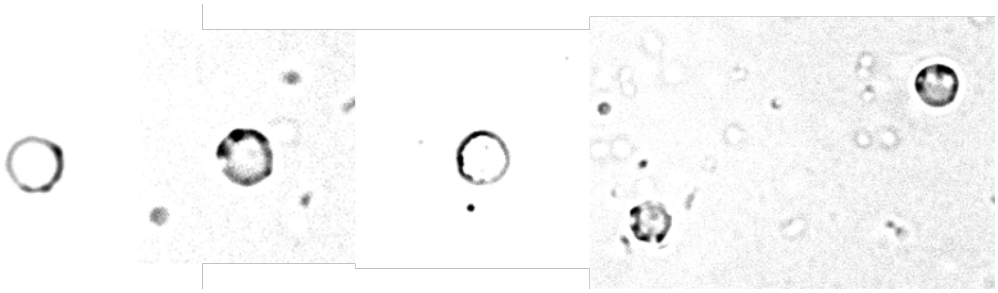

Figure. S 1: Optical asymmetries arising from the TiO_2 nanoparticles on a SiO_2 microparticle support. Contrast has been enhanced to highlight the nanoparticles (dark spots)

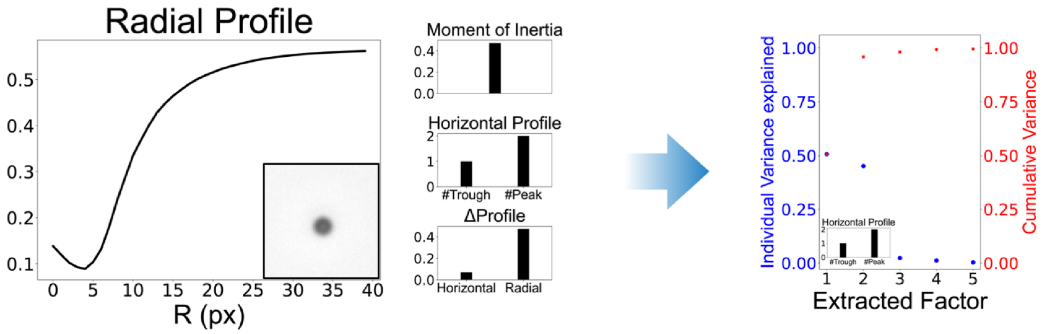

Figure. S 2: Extraction of numerical and categorical features from a single particle mask. From the mask, 40 values of the radial profile, the image moment of inertia, the difference between the largest and smallest values in the horizontal and radial profiles, and the number of maxima and minima in the horizontal profile are extracted. From the numerical features (radial profile, moment of inertia, and $\Delta(max - min)$) of the two profiles), exploratory factor analysis determines the 5 underlying factors with the highest explanatory power (99.5% of total variance)

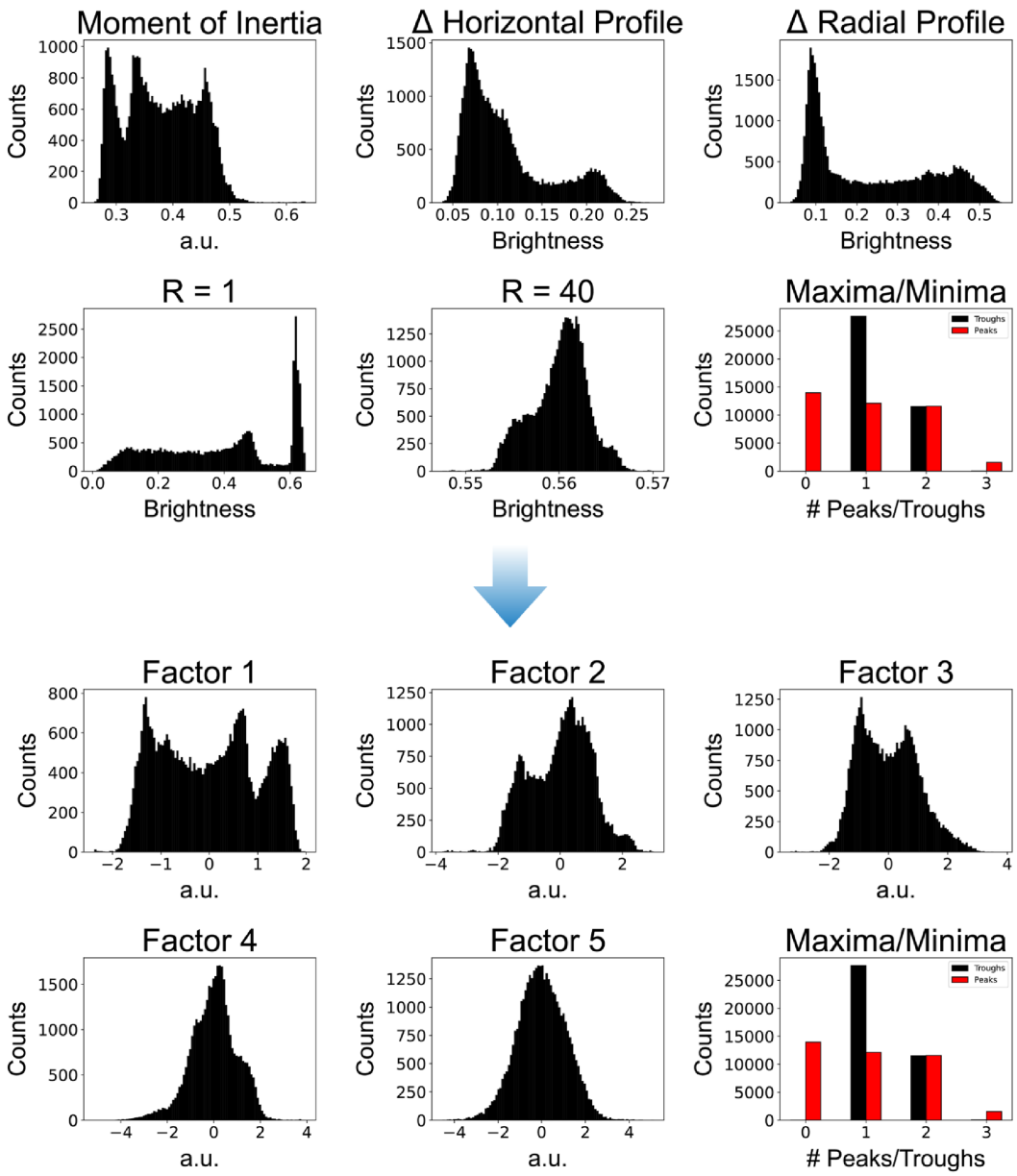

Figure. S 3: Extraction of numerical and categorical features from the full set of Z-slices across many particles (> 100). The 5 underlying factors with the highest explanatory power are extracted from the numerical features as described in Figure S2. The distributions across all observations (31834 total Z-slices) for selected features are shown.

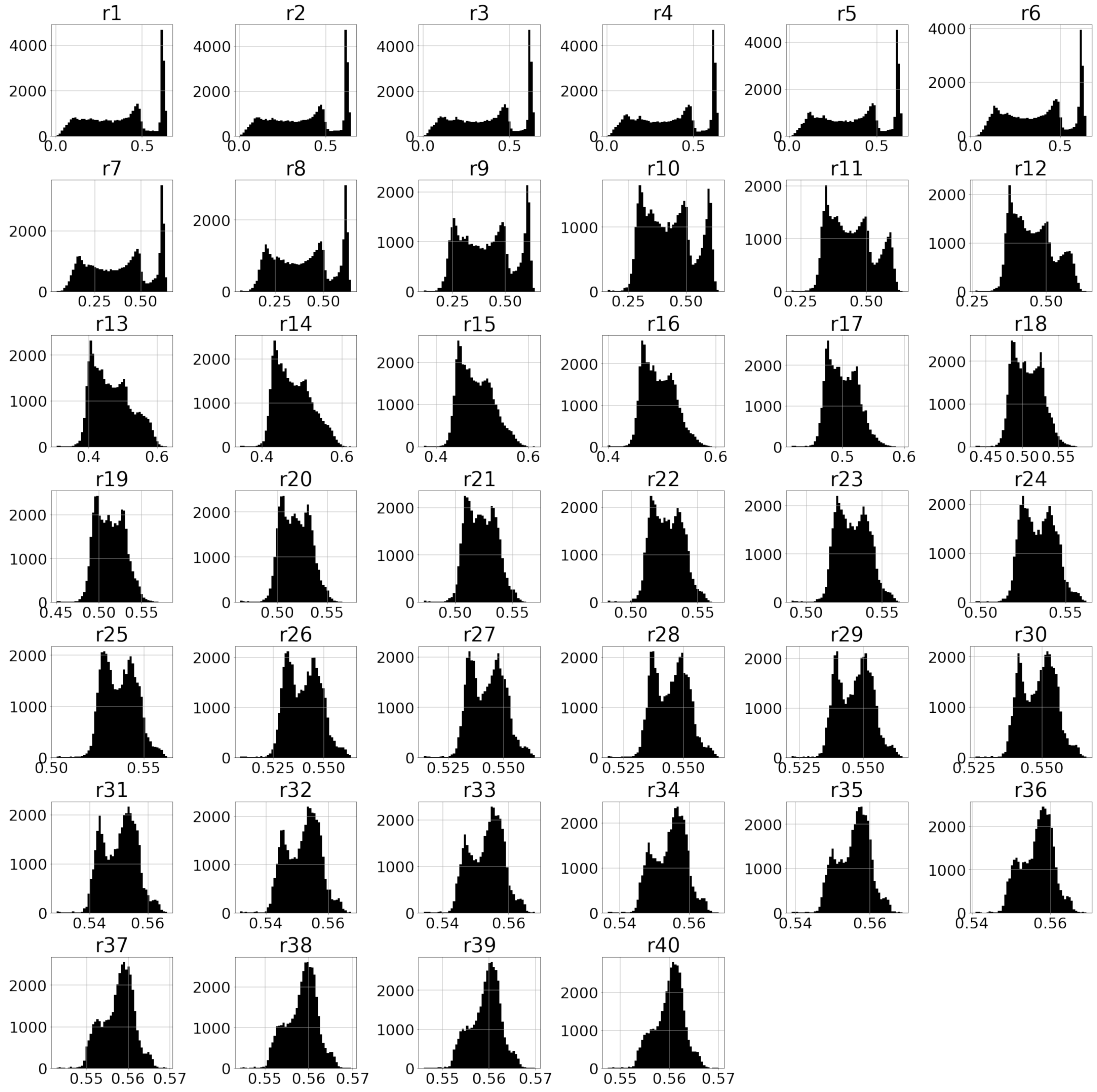

Figure. S 4: Extraction of all the radial features from the full set of Z-slices across many particles (> 100). The self-similarity between subsequent values of the radial profile can be observed from the distributions

Performance of Ensemble Tree-based Models

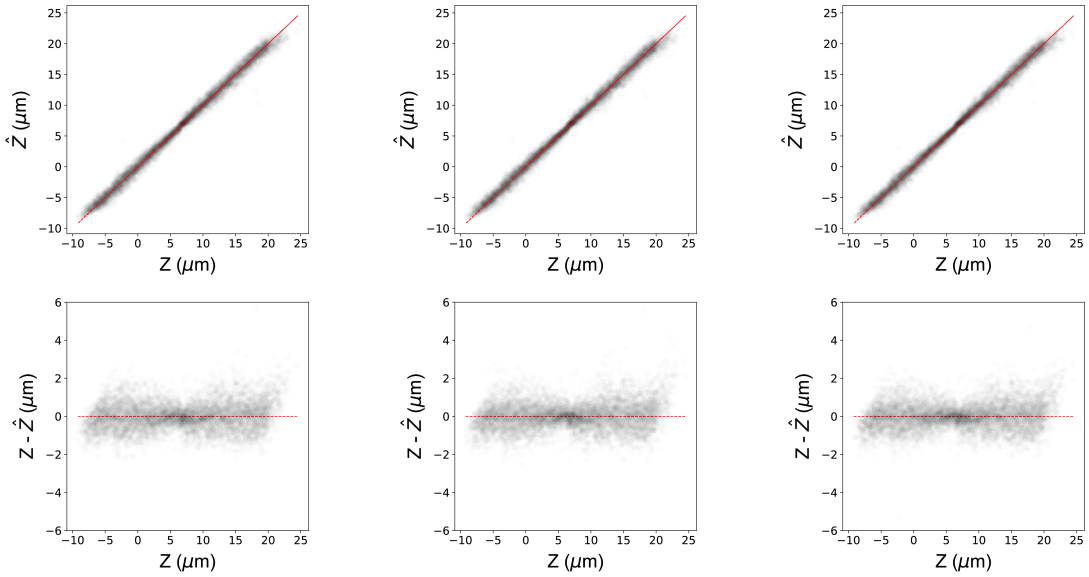

Figure. S 5: Comparison of the model predictions (top row) and residuals (bottom row) obtained against the ground-truth values from labelled Z-slices, for different decision tree based ensemble models. Left: Predictions obtained using a Random Forest (*1, 2*). Middle: Predictions obtained using XGBoost (*3*). Right: Predictions obtained using a voting regressor combining the predictions of the XGBoost model (middle) and Extremely Randomised Trees model discussed in the main text. For comparison, all models were trained and tested on the same randomly shuffled datasets. We find no qualitiative difference in the performance of the respective models (all with normalised errors $\epsilon = s(z_{measured} - z_{label})/Z_{total} = 0.021$ (where ϵ is the normalised error, $s(z_{measured} - z_{label})$ is the sample standard deviation of the residuals, and Z_{total} is the total valid range of tracking, from (*4*))). We opt for the Extremeley Randomised Trees in the main text principally by visual inspection of the residuals and the significantly reduced model training time (due to the random selection of thresholds at the decision tree nodes).

Text S1: Deep Learning (Convolutional Neural Networks)

The recent hype surrounding Machine Learning (ML) models is in large part due to the renewed interest in Deep Learning (DL) architectures. The concept of artificial neurons is not new (*5*), but the rise in computing power, demonstrated by Moore's law, coupled with the accessibility of GPU processors and modifications to training algorithms, has made the training of Neural Networks far more feasible. One of the main advantages of DL models for the 3D tracking of the Janus microswimmers compared to traditional ML models described in this manuscript is that they can directly accept image inputs, and the extensive feature extraction and engineering steps described for the traditional ML models are no longer necessary. This makes the training pipeline of DL models more generalisable to different particle systems and optical configurations compared to traditional ML methods, as we will now demonstrate.

Amongst standard Deep Learning architectures, Convolutional Networks (CNNs) are the best performers for computer vision tasks such as image classification (*6*). The introduction of convolutional and pooling layers into the architecture of neural networks (*7, 8*), was in-part inspired by studies into the neurons of the visual cortex which demonstrated the presence of small, local receptive fields recognising patterns of varying complexity (*9*). Unlike a dense layer, each neuron in a convolutional layer is connected to a limited subset of pixels in the previous layer. This allows the network to first focus on smaller features before assembling them into higher-level features in subsequent layers. Each convolutional layer will consist of a defined number of filters performing linear operations, outputting different feature maps. Thus, the output of each convolutional layer can be represented as a 3D set of 2D feature map slices representing each linear operation. An activation function, typically the rectified linear activation function (*10*), is then usually applied to the outputs to introduce non-linearities into the transformed feature maps. Pooling layers, which sub-sample the input feature map by aggregating the values and

return a feature map of reduced size, are frequently used in CNNs to minimise the memory requirements during training and inference. At the cost of image information, they reduce the number of parameters, computations and memory usage in subsequent layers, and can also introduce a small degree of invariance to different transformations of the image. They are typically used between convolutional layers, particularly in deep CNNs, to prevent an explosion of computational load. The final layers of a CNN are a dense layer with outputs appropriate for the desired regression or classification task.

Given our relatively simple image processing task, we build our CNN sequentially using the Keras backend of the TensorFlow API (*11, 12*), analysing our TiO_2-SiO_2 Janus particle system. In this manner, we find that the best performance can be achieved with a CNN architecture consisting of 3 convolutional layers and 2 fully connected dense layers (see Figure S 6). The first convolutional layer is constructed with 3x3 kernels outputting 64 feature maps, using a stride of 2. This reduces the dimensionality of the input image (87x87 pixels), allowing an increase in the number of filters in the second convolutional layer (128) without an explosion of parameters. The first two convolutional layers are followed by the ReLU activation function, and no pooling is used. Interestingly, we obtain better performances by introducing a third convolutional layer (128 feature maps) without an activation function. The output of the third convolutional layer is then fully connected to a dense layer, with 64 output neurons and again activated with the ReLU function. The output of this dense layer is finally connected to a dense layer with a single output neuron representing the target Z-value. This output has a linear activation function for regression, which is regularised using L2 (Ridge) penalty terms. The model is trained using Mean Absolute Error (MAE) as the cost function (*13*). Mini-batch gradient descent with early stopping is used to optimise the weights (100 observations), which was the batch size identified as optimal for training and which also easily fit into the memory without significantly extending training times.

Remarkably, this simple CNN architecture achieves similar results as the Decision tree based ensemble methods that we study, despite the essentially raw particle masks with minimal pre-processing used as inputs. However, in the application to our 3D swimming trajectories, we note significant noise in the axial tracking, specifically during the 2D segments of motion. On closer visual inspection of the model predictions on the test data, we note an underlying structure in the residuals, which has a significant negative impact on our Z-tracking. This highlights the importance of validating the model predictions on the out-of-test data to be analysed. Nevertheless, the performance of our trained CNN on our particle masks during the training-validation-test stage demonstrates that even simple neural network architectures can be promising for the 3D tracking of microswimmers with optical asymmetry, and further work in this direction might yield better outcomes than that achievable with traditional ML models.

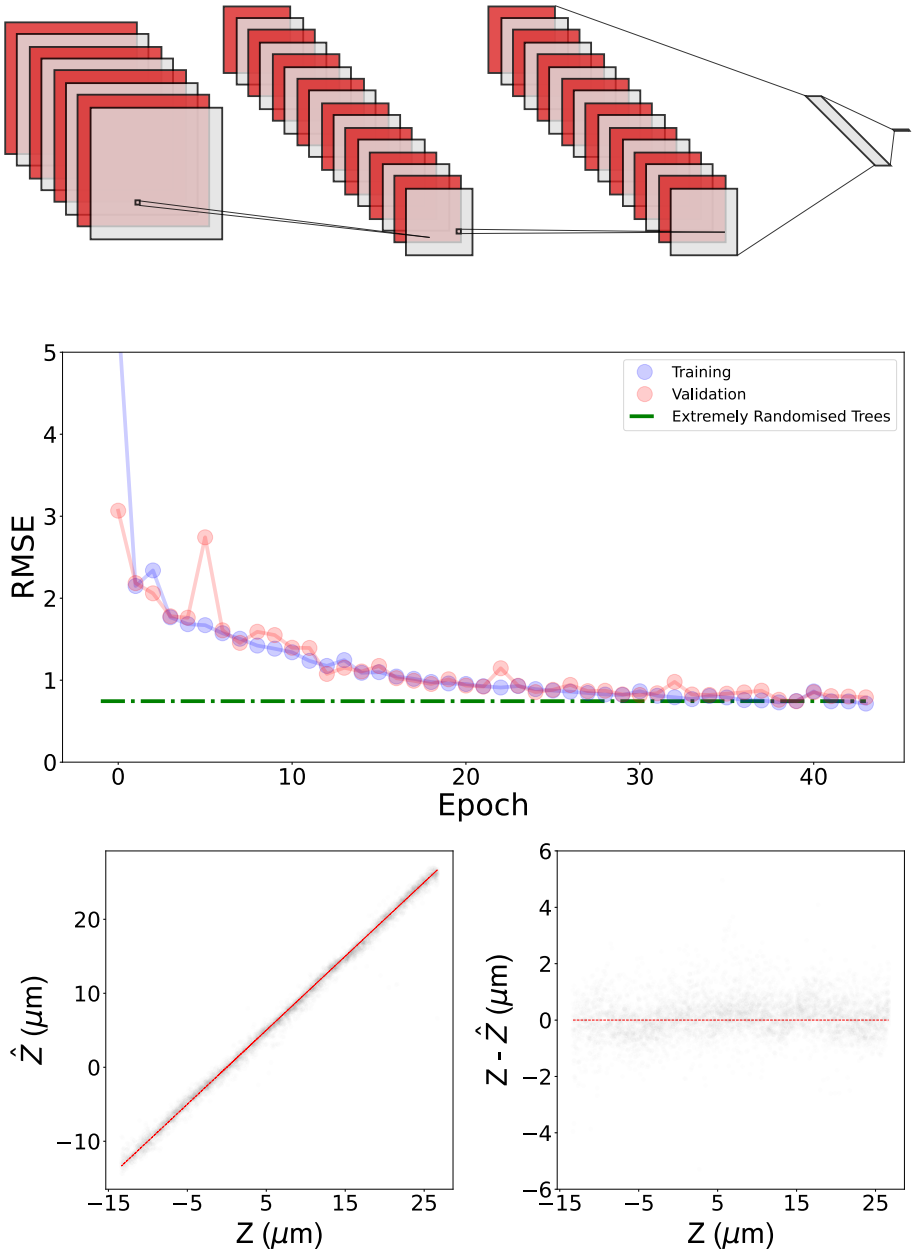

Figure. S 6: Top row: Schematic of the architecture of our CNN with 3 Convolutional layers and 2 Dense layers (single output for the Z-label regressed), created using software from (*14*). Middle row: Training-Validation performance of the CNN with Epochs. The root-mean-squared error (RMSE) saturates at a value similar to that achieved using traditional decision-tree based ensemble ML models. Bottom row: Performance of our trained CNN on the test data-set. We note a slight underlying structure in the residuals (right), which could explain the poor performance on real 3D trajectories despite good performance on the shuffled test data.

References

1. Tin Kam Ho, *Proceedings of 3rd International Conference on Document Analysis and Recognition* pp. 278–282 (1995).

2. L. Breiman, *Statistical Science* **16**, 199 (2001).

3. T. Chen, C. Guestrin, *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* pp. 785–794 (2016).

4. R. Barnkob, C. J. Kähler, M. Rossi, *Lab on a Chip* **15**, 3556 (2015).

5. W. S. McCulloch, W. Pitts, *Bulletin of Mathematical Biophysics* **5**, 115 (1943).

6. R. Yamashita, M. Nishio, R. K. G. Do, K. Togashi, *Insights into Imaging* **9**, 611 (2018).

7. K. Fukushima, *Biological Cybernetics* **36**, 193 (1980).

8. Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, *Proceedings of the IEEE* **86**, 2278 (1998).

9. D. H. Hubel, T. N. Wiesel, *The Journal of Physiology* **195**, 215 (1968).

10. V. Nair, G. E. Hinton, *Proceedings of the 27th International Conference on Machine Learning* pp. 807–814 (2010).

11. F. Chollet, Keras (2015).

12. M. Abadi, *et al.*, *arXiv* (2016).

13. B. Midtvedt, *et al.*, *Applied Physics Reviews* **8** (2021).

14. A. LeNail, *Journal of Open Source Software* **4**, 747 (2019).