MicroDroP: Microfluidic Droplet Coalesence Prediction with Hamiltonian Neural Networks

Nihir Vedd [1] **Carles Balsells** [1] **Harry Coppock** [1] **Alex Spies** [1]

1. Introduction

Microfluidic Droplet Coalescence Prediction (MicroDroP) attempts to model whether two droplets will coalesce. The dataset consists of 48 "videos", with a max length of 948 frames. The task can be decomposed into 3 parts: 1) Using the first 100 frames predict the pixel wise dynamics of the remaining rollout; 2) Using the first 100 frames to predict whether droplets will coalesce; 3) Generate new synthetic data to be used to in part 2.

The primary challenges associated with this dataset are:

- Long horizon rollouts requiring precise dynamics to avoid compounding errors
- High-resolution images with noise
- Very limited high-dimensional data with sparse labels

We attempted all 3 challenges with a variety of different approaches (discussed below). We successfully completed parts 1 and 2, achieving an **Mean Squared Error (MSE) and accuracy of ¡¿ and 76.3%** respectively. Additionally, we successfully generated new synthetic data - both transformation-based data augmentation and via a generative model. Our model is currently training with this augmented data - thus we are currently unable to report metrics for this task.

We broke down the overall task into 5 different steps: 1) Visualising data; 2) Data Analysis; 3) Cleaning the data; 4) Augmenting data; 5) Modelling. The most expensive of these steps was the modelling process. We 'parallelised' this where possible with the team - building and testing a variety of different models. Our best model was a multi-task Hamiltonian network, which was trained to perform all 3 parts of the challenge. This network uses a Variational Autoencoder (VAE) as its generative backbone. However, we found that our EfficientNet-LSTM (Long Short Term Memory) performed at a higher performance than the Hamilitonian Network for part 2. The architecture is shown in Figure 1

2. Data

Visualising data. Before starting on the modelling process, we wanted to gain a better understanding of *why* droplets coalesce. We built a visualisation tool which takes a series of .npy files (frames of a trajectory), and plots an animation. This showed us 2 things: 1) Our hypothesis of speed being a factor of coalescence was wrong. By looking the frame number of when the droplets collided, we can work out a crude approximation of the speed of the droplet. For a small sample over both classes, the average frame of collision is almost indistinguishable. 2) There is "noise" in the image which interacts with the droplets.

Data Analysis. The data we were provided are a series of one-channel frames. We find that the average percent of non-black pixels is about 1%. Despite the challenge requirement of reporting the MSE metric, we posit that it is neither a suitable loss or metric to optimise for as an all black image will receive a low MSE.

[1]imperial College London. Correspondence to: Alex Spies <afspies@imperial.ac.uk>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

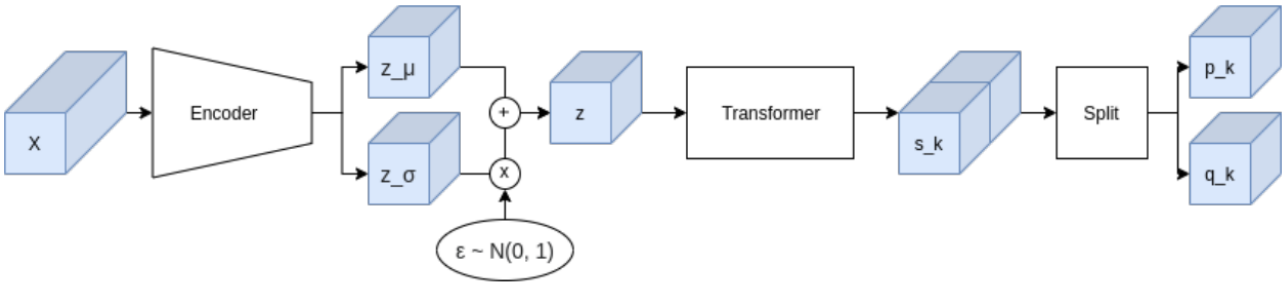

Figure 1. HGN network architecture to find the final abstract position and momentum (qk, pk) from the input sequence. Tensors are represented in blue and operations in black. The encoder takes as input a sequence of k + 1 frames concatenated along channels and samples the latent variable.

Cleaning data. Our data visualisation showed noise in the image. Namely, the full-scale 1024x1024 image had white 'specs' present, and one of the 4 walls was faintly visible. This can be seen in Figure 2-a. A full scale image is too big to feed directly into a neural network (even if we were to feed it into an encoder to get a latent representation). Thus, first resizing a frame is necessary.

However, as seen in 2-b, standard interpolated down-scaling faces 2 issues: 1) Specs and the wall are still visible; 2) The circle is now "broken".

To mitigate these issues, and create viable down-sampled images, we use 2 filters. Firstly, a median filter, which is responsible for preserving the edges of the image while still removing noise. Secondly, a 2d filter, which uses a kernel to sharpen and find edges in the image. The result of the resized and preprocessed images can be seen in Figure 2-c.

A new challenge now occurs with the aforementioned pre-processing strategy. How do we 'safely' evaluate MSE? A trivial option is to calculate the MSE against the smaller images, and then scale the MSE value by the image scaling factor squared. However, this strategy does not fully meet the requirements of part 1) because the generations are required to be 1024x1024. We can reconstruct the original image using 1 of 2 strategies: 1) 'classical' upscaling of the images. We acknowledge the classical upscaling will produce a worse MSE because of the information loss from upscaling. However, if were to use an image super-resolution model (2) instead of classical upscaling, our reported MSE will be closer to the aforementioned theoretical value.

Augmenting data. As mentioned in Section 1, we have 2 data augmentation strategies. The first is a custom transformation based augmentation that we built ourselves. The transformations used to create this augmented data were RandomRotate, RandomHoriztonalFlip and RandomVerticalFlip. The RandomRotate rotates an image by a multiple of 90 degrees (i.e. 0, 90, 180, 270 degrees), and the RandomXFlip flips an image horizontally or vertically. We opted against sheering or cropping because these will change the physics of the image/video, and thus will be harder to model.

We also synthesised new data using our generative Hamiltonian model (see Section 4).

3. Modelling and Experiments

Part 1. We attempted a variety of different models for the different tasks. For part 1) we experimented with 2 networks: an auto-regressive Recurrent Neural Network (RNN), and a generative Hamiltonian network (described in Section 4). The auto-regressive RNN used a Gated Residual Unit (GRU) cell at its core, and attempted to model the co-ordinate values. More specifically, we mapped the 64x64 image to a 4096-dimensional khot vector, where the 'hot' values were the indexes at which there was a nonzero value in the image. The network was modelled as a multi-label binary classification problem - predicting a 4096 dimension vector. This prediction can then be mapped back to 64x64. This method was severly outperformed by the Hamiltonian network.

Part 2. For part 2, we modelled the binary classification problem in 2 ways. Firstly, by incorporating multi-task learning into the Hamiltonian network. The second way was to use a feature-encoded representation of the image, and then feed a sequence of these features into an LSTM based RNN. The LSTM uses the temporal dependencies of the features to make a

binary classification at every N steps of the frames for a given trajectory. We then use the mode of these predictions for a given trajectory as the predicted label of the image. We use EfficientNet (Tan & Le, 2019) as the backbone image encoder.

4. Hamiltonian Networks

To accurately model long-term dynamics we can explicitly encode known physical laws into our models. This can be done differentially through the hamiltonian formulation of classical dynamics, as demonstrated by (Greydanus et al.). In this formalism an encoder is trained predict the hamiltonian, H of a system of particles with momenta $\vec{p}$ and positions $\vec{q}$, related by

$$\frac{d\mathbf{q}}{dt} = \frac{\partial \mathcal{H}}{\partial \mathbf{p}}, \quad \frac{d\mathbf{p}}{dt} = -\frac{\partial \mathcal{H}}{\partial \mathbf{q}} \tag{1}$$

This is then used with numerical euler integration (which can be back-propagated through) over the dynamics. This was extended into a variational generative framework by (Toth et al.). Here a variational auto-encoder is used, with $\vec{p}$, $\vec{q}$ sampled from distributions learned by an encoder. This should facilitate the sampling of new trajectories as required by Question 3. Based on an implementation by (Rodas et al., 2021). See Figure 1.

4.1. Results

Question 1.
MSE (on 128 x 128 images): 0.0139

Question 2 Preprocessed sequences as in Figure 2 Multimodal Hamiltonial Generative Network - auxillary loss on labels 47%

Achieved Accuracy 76.3%

Question 3 See Figure 5

References

Greydanus, S., Dzamba, M., and Yosinski, J. Hamiltonian Neural Networks. URL http://arxiv.org/abs/1906.01563.

Rodas, C. B., Canal, O., and Taschin, F. Re-hamiltonian generative networks. In *ML Reproducibility Challenge 2020*, 2021.

Tan, M. and Le, Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pp. 6105–6114. PMLR, 2019.

Toth, P., Rezende, D. J., Jaegle, A., Racanière, S., Botev, A., and Higgins, I. Hamiltonian Generative Networks. URL http://arxiv.org/abs/1909.13789.

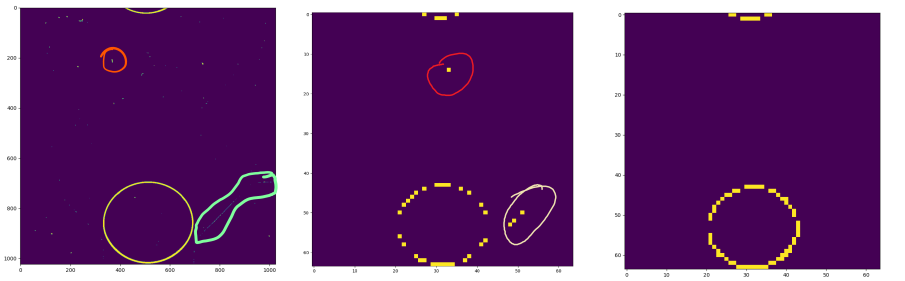

Figure 2. Visualising and preprocessing images

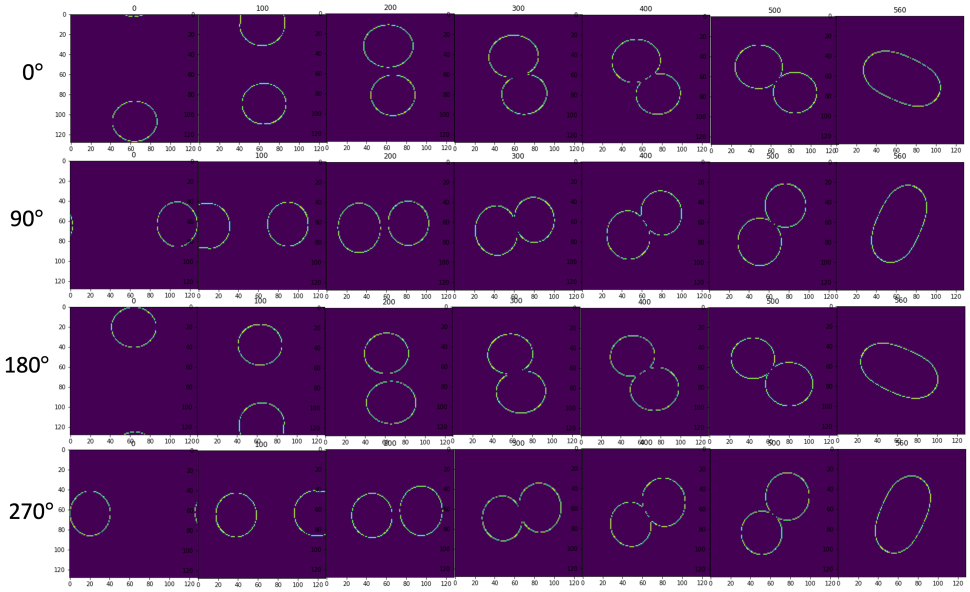

Figure 3. Examples of the random rotations applied across input sequences. Horizontal and vertical flipping were also implemented (as default PyTorch implementations do not support videos).

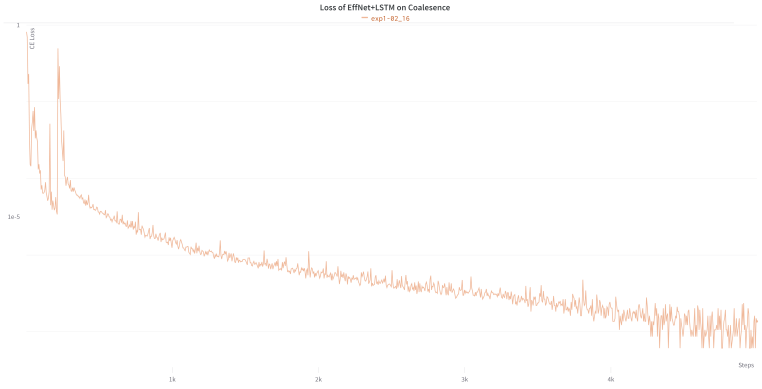

Figure 4. Loss of the EfficientNet + LSTM on the Coalesence Prediction task.

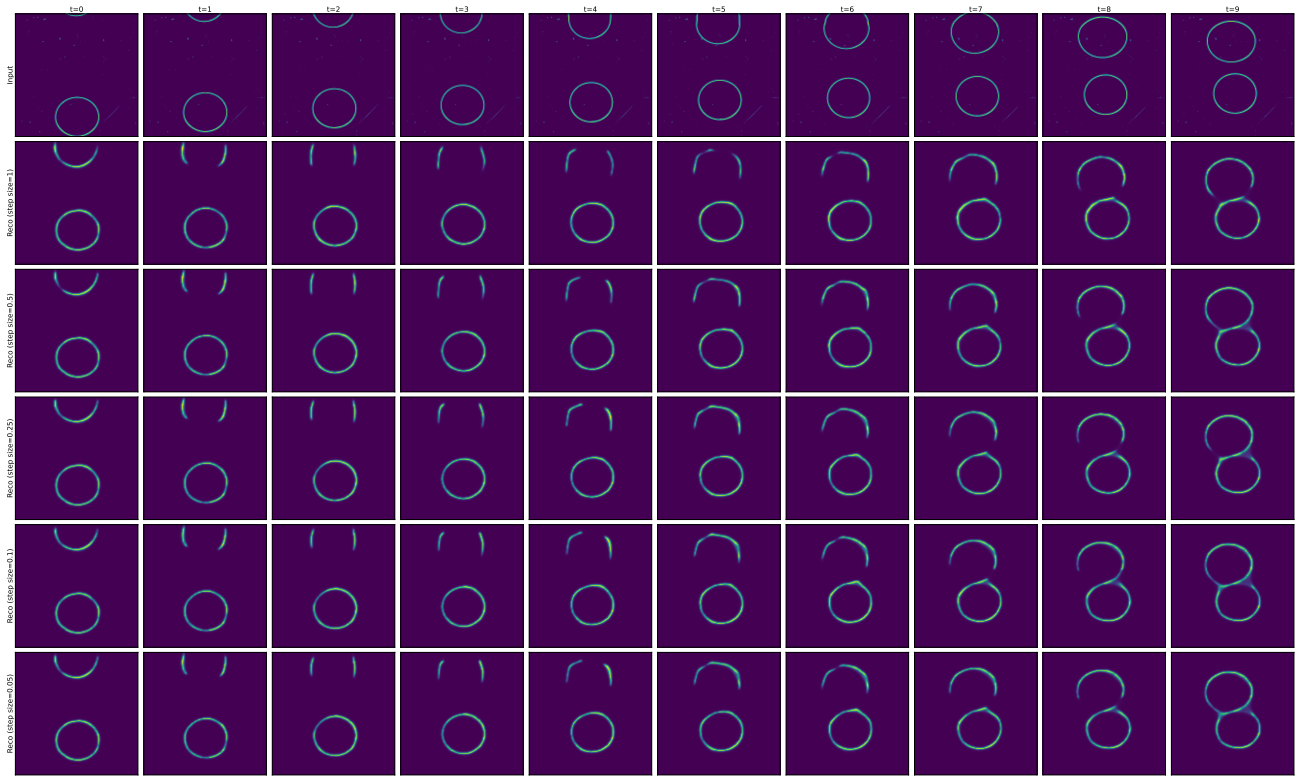

Figure 5. Rollouts from the generative hamiltonian network which takes 5 input frames (time steps 0, 20, 40, 60, 80) from the data. Different step sizes are used in the integration stage. As expected, even when training on frame-skips of 20, the acuity of dynamics with much smaller step sizes (0.05 to match the original frame rate) is essentially identical to the coarses one.