Soft Matter

ARTICLE TYPE

Cite this: DOI: 00.0000/xxxxxxxxxx

End-to-End Machine Learning for Experimental Physics: Using Simulated Data to Train a Neural Network for Object Detection in Video Microscopy†

Eric N. Minor,[a] Stian D. Howard,[a] Adam A. S. Green,[a] Cheol S. Park,[a] and Noel A. Clark[a]

Received Date
Accepted Date

DOI: 00.0000/xxxxxxxxxx

We demonstrate a method for training a convolutional neural network with simulated images for usage on real-world experimental data. Modern machine learning methods require large, robust training data sets to generate accurate predictions. Generating these large training sets requires a significant up-front time investment that is often impractical for small-scale applications. Here we demonstrate a 'full-stack' computational solution, where the training data set is generated on-the-fly using a noise injection process to produce simulated data characteristic of the experimental system.We demonstrate the power of this full-stack approach by applying it to the study of topological defect annihilation in systems of liquid crystal freely-suspended films. This specific experimental system requires accurate observations of both the spatial distribution of the defects and the total number of defects, making it an ideal system for testing the robustness of the trained network. The fully trained network was found to be comparable in accuracy to human hand-annotation, with four-orders of magnitude improvement in time efficiency.

1 Introduction

Current generation physics experiments often produce data of such complexity and volume that belie efficient analysis by classical algorithms[1,2]. The modern renaissance in machine learning[3–5] provides a potentially superior method to analyze such complex data. Indeed, machine learning methods have been successfully implemented in many scientific disciplines; from high-energy physics[6,7], and condensed matter physics[8–12] to biological systems[13]. These machine-learning algorithms often preform more robustly than previous state-of-the-art solutions[14].

A characteristic realization of a data set with both volume and complexity is that of an image sequence produced by video microscopy, a ubiquitous method of dynamical systems analysis that occurs in fields as disparate as biological systems[15,16] and hydrodynamics[17,18].

The analysis of video microscopy has long been stymied by the inherent difficulty of extracting quantitative information from images[19]. Except for very simple tasks, the classical algorithms to extract quantitative information from these videos require very specific conditions[20], requiring significant pre-processing and strict human oversight. Often these ideal conditions cannot be realized in an experimental setting, necessitating the bespoke analysis of individual image frames, where manual extraction of quantitative data is required. This results in a significant bottleneck in experimental analysis.

One common objective in the analysis of video microscopy is the extraction of the spatial position of a defined target. Machine learning methods have already been deployed to great effect in this regard, where they prove capable of both spatially labelling and categorizing human-defined objects[21,22]. However, these models are reliant on large, previously analyzed training sets, where the target has already been identified and annotated. The generation of these training sets requires a large, upfront time investment that make them impractical for small-scale applications.

Here we report on an 'end-to-end' method, where the training data and annotations, the list of spatial coordinates of targets, are procedurally generated through computer simulation, allowing for fast deployment of machine learning methods to small-scale applications.

We demonstrate the robustness of our approach through the analysis of defect-defect interactions in freely-suspended films of smectic-C liquid crystal.

1.1 Background

A smectic phase is a liquid crystalline mesophase composed of elongated molecules, with general orientational order between between the molecules and crystalline order along one axis. The crystalline order segregates the phase into stacked sheets of

[a] Department of Physics and Soft Materials Research Center, University of Colorado, Boulder, Colorado, 80309, USA

molecules that can flow freely in the plane, making smectic liquid crystals ideal realizations of two-dimensional hydrodynamic systems. Additionally, the molecules in the phase can be oriented co-linearly with the smectic-plane normal vector (SmA) or can be tilted with respect to the smectic-plane normal vector (SmC). In the latter case, the molecular tilt breaks the isotropic nature, giving the SmC phase a rich topological structure.

To first order, the Frank free energy that describes a single smectic-c layer is well approximated by the continuous XY model, which supports as ground-state solutions stable topological defects. The theoretical[23–27] and experimental[28–31] dynamics of defects in liquid crystal systems has been studied since the early 90's. However, it is an open question how well the non-hydrodynamic XY model describes the interaction of these topological defects in fluidic systems of liquid crystal materials.

Direct tests of the XY model could be made by observing the total number of and nearest-neighbor distance of defects in the coarsening dynamics of a quenched SmC film, but, as there currently exists no robust way to spatially track or label the defects in these textures, this analysis must be done manually– severely limiting the temporal resolution.

Machine-learning methods have already been successfully deployed in studies of the XY model, with previous work demonstrating the viability of using basic neural networks to identify whether a given simulated data set contains a topological defect[12]. However, the work was focused on simulated system states, consisting of molecule locations and orientations, rather than experimental image analysis. Furthermore, the algorithm that was utilized is purely for classification and did not give defect counts or locations, limiting uses in experimental data analysis.

2 Experimental System

In order to confirm the veracity of our system, we collected physical data from a typical topological defect experiment. To generate data for training the machine learning system, we used a simulation to generate perfectly annotated images and then ran those images through a data enhancement pipeline.

2.1 Experimental Defect Data

At the center of our experimental setup, shown in Figure 1, is a pressure chamber with an open aperture on the top to draw a film over. Air is pumped into the chamber, increasing pressure and causing the film to bulge outward. A valve on the tube is then opened to the atmosphere, rapidly equalizing the pressure in the chamber. The valve is controlled by a computer program that, upon reaching a predetermined pressure differential between the chamber and atmosphere, opens the valve, triggers the high-speed camera recording, and starts saving pressure readings. The collapse of the film results in a mechanical quench, which creates a high-energy state resulting in a large density of defects in the film which rapidly annihilate.

SmC liquid crystal defects can be visualized with partially or fully crossed polarizers. In our setup, polarized light is shined perpendicularly onto the film. The reflected light is collected into a microscope where it passes through a partially crossed polarizer.

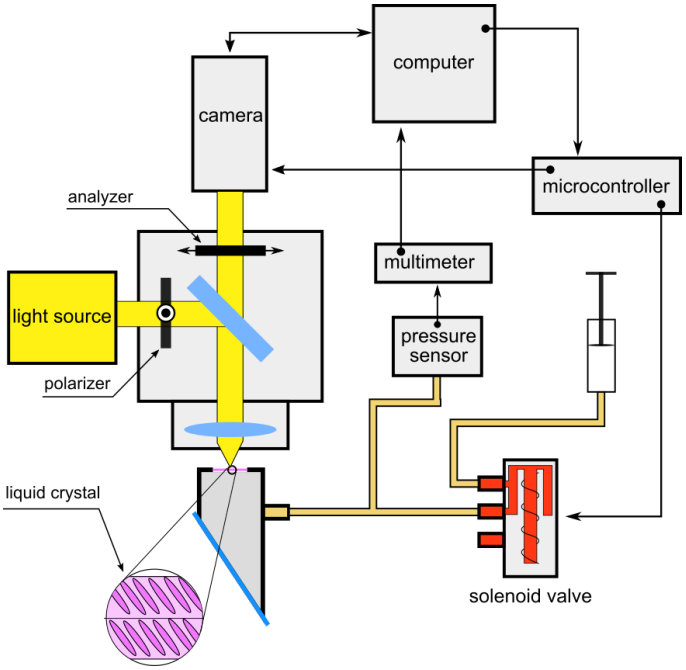

Fig. 1 Schematic of mechanical quench experiment. The syringe is coupled to a translation stage, allowing for fine control. When depressed, the compression increases the pressure in the quench chamber, causing the film to bulge outwards. The pressure is monitored by a pressure sensor (Honeywell SDX010IND4) which outputs a proportional voltage read by a digital multimeter connected serially to a computer. The user can enter a target threshold pressure in a program running on the computer, which automatically triggers a quench when said threshold is reached. The quench is triggered by sending a digital signal to a connected microcontrolled (Arduino Uno) which first instructs the high-speed (Phantom V12.2) camera to begin recording and then opens the solenoid valve, venting the system to atmosphere. The resulting dynamics on the film are captured by the camera as a video, which is then transferred to the computer for further analysis.

Because of the birefringent nature of SmC liquid crystals, when viewed under crossed polarizers the orientation of the molecule is mapped to a reflected intensity as $I \propto \sin(2\theta)^2$, where θ is the angle between the in-plane projection of the molecule (refered to as the c-director) and the polarizer. However, working with fully crossed polarizers dramatically decreases the reflected light, acting as a significant limiting factor for the exposure time needed to get viable images. Therefore, we work in a regime of decrossed polarization, which reflects more light. In this regime, the reflected intensity is well approximated by $I \propto \cos(2\theta)$, giving a characteristic 'bowtie' structure, as seen in Figure 2[32]. A high speed camera (Phantom V12.1) records the reflected light in gray-scale at 500 frames per second with an exposure time of 1900 μs, allowing us to directly view the coarsening dynamics of the film.

Each video lasts 12.2 seconds, capturing the entirety of the short term dynamics. The images, with 1104x800 resolution and 12 bit pixel precision, allow for high contrast to be gained in post processing. We used PM2[33] to form a film that exists in a Smectic C phase at room temperature. Figure 2 provides snapshots of the data collected over a range of times.

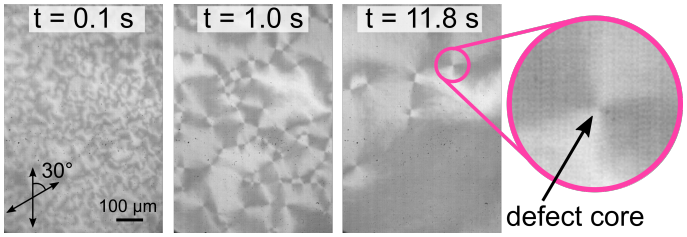

Fig. 2 Experimental data showing time evolution of a film with topological defects viewed under reflection microscopy. Images are labelled with the time elapsed since quenching. The topological defect appears as a 'bowtie' structure.

2.2 Simulation Data

Two methods were used to generate images for training the machine learning models to detect topological defects. The first method procedurally generates textures by linearly adding a random number of defects at random locations to an initially aligned XY grid. In the XY model, a stable, zero-temperature solution of the XY Hamiltonian is given by the plus/minus defect director configuration:

$$\varphi(x,y) = \pm \arctan\left(\frac{y-y_0}{x-x_0}\right) + \phi_0, \tag{1}$$

where ϕ_0 is a phase offset which was also randomized, and (x_0, y_0) is the location of the defect.

In this way, arbitrarily complex defect configurations can be generated. The c-director can be mapped to a scalar intensity value using the Schlieren mapping $I = \cos(2\varphi)$, giving a reasonable facsimile of experimental observations of defect configurations in freely-suspended films. Because the method involves linear superpositions of zero-temperature solutions, it produces very clean images that are free of thermal noise, as seen in Figure 3 (a). This simulation method will be referred to as the 'random defect' model.

The second method is based on directly simulating the dynamics of the XY model at a finite temperature as it evolves from a high-density defect configuration[23,34,35]. The angle of the c-director at each lattice site i evolves according to the discretized Ginzburg-Landau model through the Euler update scheme as:

$$\varphi_i(t+\Delta t) = \varphi_i(t) - \Delta t\left(\eta_i(t) + \kappa \sum_{\langle i,j\rangle} \sin(\varphi_i(t) - \varphi_j(t))\right) \tag{2}$$

where κ is a visco-elastic constant, and η_i is a random number with moments that correspond to the temperature through the fluctuation-dissipation theorem:

$$\langle \eta_i(t)\eta_j(t')\rangle = 2T\delta_{i,j}\delta(t-t') \tag{3}$$

Using the Chester-Tobochnik method[36], the defect locations can be extracted by calculating the winding number around each lattice plaquette. This is done by computing the successive differences in angles as the plaquette is circumnavigated, defined as: $\Delta\varphi = 2\pi n_i + \theta$, where θ is restricted to the range $(-\pi, \pi)$. The vorticity of the square is equal to the sum of the n_i's, and can be either positive or negative. In this way, the locations and number of the defects for each time-step in the system can be extracted in an automated annotation process. This method of directly simulating the XY model is capable of producing a wide variety of textures with different amounts of director fluctuations, shown in Figure 3 (b). This method will be refered to as the 'thermal-defect' model. The inclusion of temperature means these simulations more strongly conform to the experimental observations, making the thermal-defect model the preferred method.

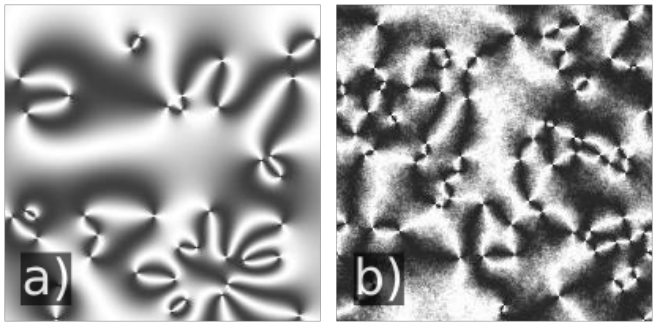

Fig. 3 Typical image from random-defect simulation (a) and thermal-defect simulation (b).

3 Topological Defect Tracking

In order to make a system viable for usage with real experiments, we developed a pipeline that makes use of modern deep-learning object detection and image enhancement techniques to train a model capable determining both the location and count of objects in an image.

3.1 Pipeline and Image Enhancement Motivation

For object detection we used darkflow[37], a TensorFlow implementation of the YOLOv2 algorithm[38] that offers improved performance when compared to the original YOLO algorithm[39]. YOLOv2 learns to perform both region proposal and region classification using the darknet-19 architecture. Training the region proposal mechanism is important for defect identification as detection algorithms that rely on traditional heuristic searches, such as R-CNN[40] and its successors, Fast R-CNN[41] and Faster R-CNN[42], would likely fail to identify defects as objects. Training and testing darkflow on a set of 100 simulated 200x200 images, with each image containing 20 defects, showed that this network is viable for detecting the locations of defects in simulation data. However, data enhancement techniques are necessary to make a network trained on simulated images viable for application to real data.

Machine learning algorithms, by nature, optimize themselves to perform as well as their architecture allows on the given training data. While this can lead to highly effective systems, it is the primary reason why training on a set of simulated data often makes the final model non-viable in the real world; simulated data is highly predictable and clean while real-world data can have significant noise and variance in how key objects appear. By training on the simulated data, the system will over-fit[43,44] on the very specific shapes, textures, and gradients produced by the

simulation. Our solution for training a model on simulated data to analyze real data is to introduce various artifacts that mimic real-world inaccuracies into the simulated images.

3.2 Standardization and Simulated Image Enhancement

The first issue that needs to be dealt with is lighting and contrast. In simulated defect images, the intensity of a pixel ranges from perfectly black to perfectly white depending on the director orientation, maximizing the gradients and contrast in the image. The mean intensity of the image will also generally be around 0.5 on a scale from 0 (black) to 1 (white) since there is no offset to the image brightness. When using an experimental image, the difference in brightness between perfectly aligned and perfectly misaligned directors is much smaller than the full dynamic range of the image, which causes smaller gradients. The average brightness of the experimental data is rarely 0.5, so what constitutes bright and dark pixels is more complex than just the intensity of the pixel. To make the simulation and experimental images as similar as possible in regards to average intensity and dynamic range, a variant of the basic feature standardization procedure[45] is used. Each pixel's intensity value is set according to the feature standardization formula

$$x' = \frac{x - \bar{x}}{6\sigma} + 0.5 \tag{4}$$

where x' is the output pixel intensity, x is the input intensity of each pixel and σ is the standard deviation of the global image pixel intensities. The output images have a mean pixel intensity of 0.5 and a dynamic range of six standard deviations. This procedure reasonably standardizes the lighting and contrast of the images regardless of the actual lighting and camera conditions, providing consistency across multiple data sets.

Adding imperfections to the simulated images emulates the experimental data and improves the robustness[46,47] of the neural network model. Gaussian blurring, Fourier noise, randomized image variance, randomized lighting boundaries, and arbitrary objects each target identified inconsistencies between simulation and experimental data. The alterations increase the image variety in our training data-set and teach the model that these imperfections are to be ignored when attempting to detect defects.

Due to the relatively low lighting of the experimental images, the camera read noise, generated by the camera hardware when reading information from the CCD (Charge-Coupled Device), is significant relative to the signal size. Applying a 2-D discrete Fourier transform, the composition of the image is extracted in the frequency domain[48], shown in Figure 4, where the periodic read noise appears as regularly-spaced lines. This characteristic camera noise is added as low-frequency noise to the simulated data set to increase similarity to experimental data, shown in Figure 6 c.

When collecting experimental data, it is rare that perfect focus is consistently achieved. In the quenching experiment, there are additional film fluctuations as the pressure on the two sides of the film equalize, resulting in a shifting focus point as the film moves. The lighting conditions can change significantly for different experimental setups. In particular, film thickness and camera

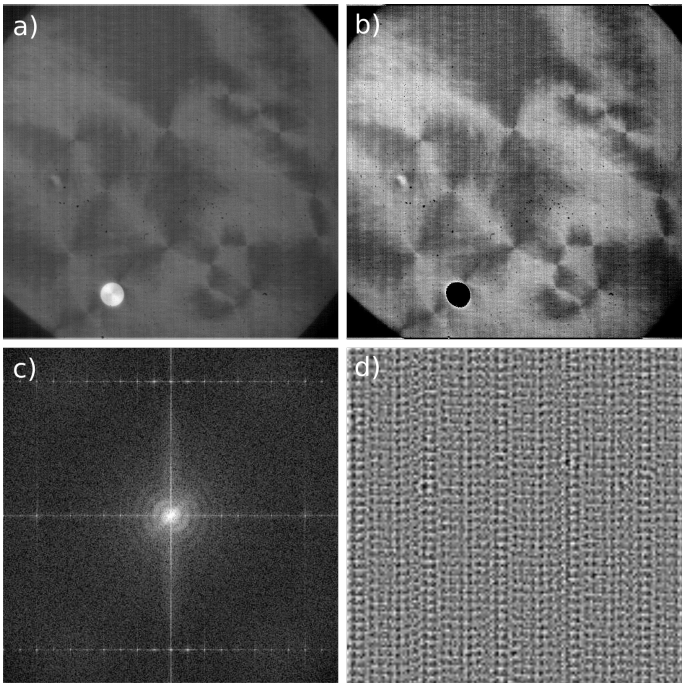

Fig. 4 Image standardization and periodic noise extraction. a) Raw image with an island. b) Standardized image. c) Magnified Fourier transform of standardized image. d) Magnified noise extracted from the image.

settings have an impact on the intensities of light captured by the camera. Randomized Gaussian blurring emulates the non-perfect and variable focus of the experimental data, shown in Figure 6 (b). The overall brightness and dynamic range of the simulated images is randomized to prevent the model from being dependant on specific light intensities or gradient magnitudes unique to the simulation.

The final additions to the simulated images are randomized lighting domains and circular artifacts. Observing the pattern of defect detections from previous models, it was discovered that detections would be made along lines where the lighting abruptly shifts. The microscope aperture and the boundaries between CCD sectors, which consistently read slightly different pixel intensities, generate the light shifts. To prevent this, the simulation images were broken into four quadrants of randomized size, with each quadrant having slightly different brightness. False detections were also made around the boundaries of islands, which are regions in liquid crystal films with additional layers of material. To prevent this, circles of random brightness were added to the simulation data to provide neutral examples[49] of non-defect objects that should not affect detections, shown in Figure 6(c).

3.3 Effects of Simulated Image Enhancements

To evaluate the effectiveness of each component in the pipeline, several models were trained on simulated images enhanced by various combinations of pipeline components. The models were validated using a hand-annotated set of experimental images to determine how well they performed on real data relative to human performance. The efficacy of the machine learning can be quantified through the *precision* and the *recall*. Precision describes

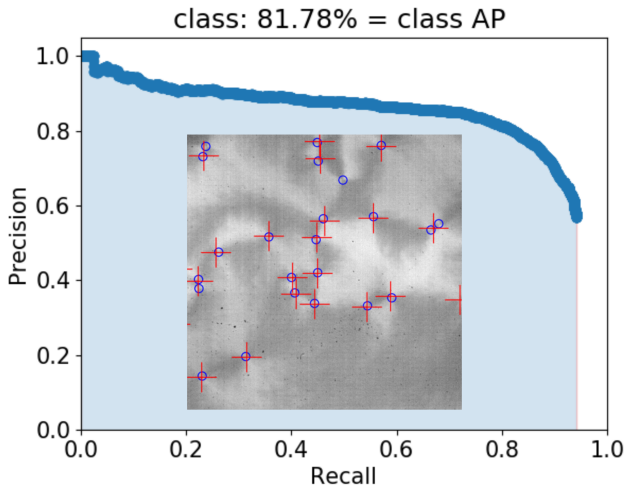

Fig. 5 Precision-Recall plot for the highest scoring model, with inset showing a characteristic annotated video still, with blue circles showing hand labelled defects and red crosses showing YOLO detections. mAP score is the percentage of area shaded between (0,0) and (1,1). Conceptually, this means that an average precision of 81.78% was achieved across all possible recall values

the accuracy of object detection. Recall describes how many of the objects in the image we detect. Rigoursly, these quantities are defined as follows, where TP represents the true positive detections, FP represents the false positive detections, and FN represents the false negative detections.

$$Precision = \frac{TP}{TP+FP} \tag{5}$$

$$Recall = \frac{TP}{TP+FN} \tag{6}$$

The model provides a confidence score for each detection. A threshold is set to drop low confidence detections. A low threshold will increase recall while lowering precision; a high threshold will have the opposite effect by only using the few detections that have a high likelihood of being correct. mAP and peak F1 scores are used to characterize the model across all thresholds. AP is the Average Precision across all recall scores and is a general measure for the effectiveness of the model across all thresholds. mAP is the mean AP across all detected classes, however since we are only training to identify defects, mAP is effectively equivalent to AP. The F1 score is the harmonic mean of the precision and recall, and it provides an overall 'goodness' measure in regards to recall and precision at a specific threshold. We recorded the maximum F1 score of each model to provide a measure of the peak model performance when choosing an ideal threshold. The mAP[50] score can be thought of as measuring average performance without setting a minimum confidence threshold while peak F1[51] score measures the highest obtained performance over all thresholds. Using $p(r)$ to represent the precision at a given recall value and n to represent the total number of detections, AP and F1 can be defined as follows:

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{7}$$

$$AP = \int_0^1 p(r)\,dr$$

however, in order to use AP for real data the formula must be discretized into

$$AP = \sum_{r=1}^{n} p_{interp}(r) \tag{8}$$

where

$$p_{interp}(r) = max_{\tilde{r}:\tilde{r}>r} p(\tilde{r})$$

Using the max function smooths the AP curve, preventing the local dips at each false detection from affecting the global score. In Fig. 5, the mAP score is represented by the area of the shaded region.

Simulated training images enhanced with only blurring, random islands, random lighting quadrants, or randomized image brightness and contrast produced models that performed poorly on experimental images. On validation, these models received mAP scores of $< 30\%$. Simulated images enhanced only with noise extracted using the Fourier transform produced a viable model, however the mAP score only reached 53.3%.

Adding multiple types of noise to the simulated images produced greatly improved models. Combining all types of artifacts produced a model with a mAP score of 59.0%, however this was outperformed by a model trained using only Fourier noise and randomized blurring which achieved a mAP score of 69.3%. Adding all forms of noise at a lower intensity to the simulated images produced a model with a mAP score of 74.9% [Fig. 6 d]. This suggests that a balance must be struck between making modifications and maintaining enough clarity to identify objects when training.

The original YOLOv2[38] paper reported an average mAP score of 73.4% when tested using the Pascal VOC2012 test set, which puts the detection accuracy of our model trained with simulated images on par with models trained using real images. This supports the viability of training YOLO object detection models with simulated data for use on experimental data.

3.4 Improvements Using the XY Model

Models trained with data produced from the simulation using a Landau-Ginzberg implementation of the XY model yielded improved accuracy. Landau-Ginzberg simulations provide thermal noise, as we see in our real data, and emulate natural defect systems. With no image modifications, a model trained on simulated images from the XY simulation attained a 47.4% mAP score, a significant improvement over the 2% achieved with the model trained on the raw random defects data.

The Landau-Ginzberg simulations are highly time-dependent with defect counts following a power law. This means that a linear reduction in defect number requires an exponential amount of time. If we train a model with only early time simulation images, where there is a high density of defects in the training data, the model will perform worse on images with a lower defect density. As such, the best performing models require long simulation runs to generate training data with a wide variety of defect densities.

Similar to the random defect simulation data, the best results

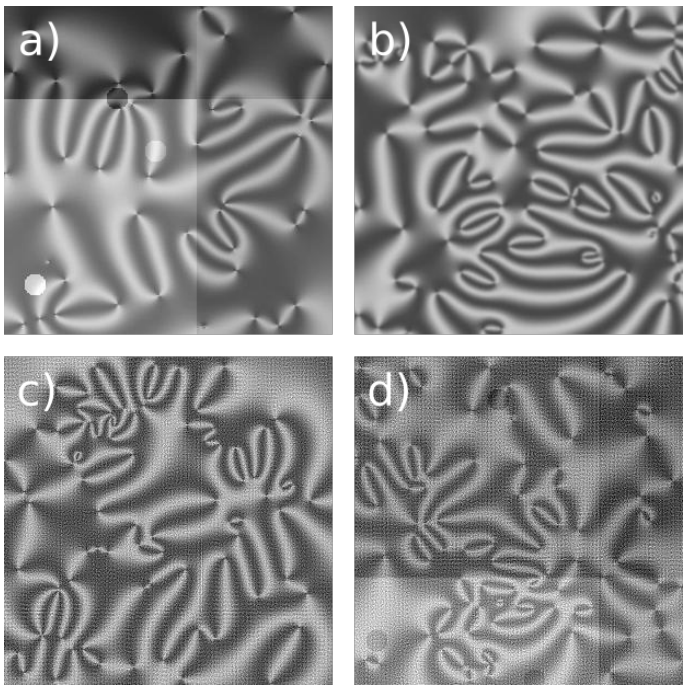

Fig. 6 Simulated images enhanced with various types of noise. (a) Image with dramatic lighting shifts and added circles (islands) (b) Image with Gaussian blurring (c) Image with Fourier noise (d) Image with all three

Table 1
Scoring of Models Trained on Simulated Images with Various Artifacts

Artifacts Added	Peak F1	mAP
LG, L-FN, L-RV, L-RB, L-GB, L-DC, RD, LR	0.811	0.818
LG, L-FN, L-RV, L-RB, L-GB, L-DC, RD	0.817	0.808
LG, FN, RV, RB, GB, DC, RD	0.806	0.783
L-FN, L-RV, L-RB, L-GB, L-DC, RD	0.754	0.749
FN, RB	0.744	0.738
FN, RV	0.726	0.725
FN, RV, RB, GB, DC, LT	0.740	0.700
FN, GB	0.707	0.693
FN, RV, RB, GB, DC, RD	0.683	0.663
FN, RV, H-RB, GB, DC, RD	0.661	0.643
FN, RV, RB, H-GB, DC, RD	0.635	0.626
FN, RV, RB, GB, H-DC, RD	0.632	0.620
H-FN, RV, RB, GB, DC, RD	0.640	0.617
FN, RV, RB, GB, DC	0.613	0.590
FN, H-RV, RB, GB, DC, RD	0.587	0.579
FN	0.569	0.533
LG	0.513	0.474
RB	0.442	0.270
RV	0.423	0.260
GB	0.296	0.116
Raw Random Defect Sim	0.099	0.020

Key
FN: Fourier Noise
RV: Random Variance
RB: Random Boundaries
LR: Long Simulation Run
GB: Random Gaussian Blurring
DC: Randomized Decross Angle
RD: Randomized Defect Number
LG: Used LandauGin Simulation
LT: Longer Training Time
L-XX: Lowered randomization of XX
H-XX: Higher randomization of XX

are attained with a lower intensity of many different forms of noise, achieving a mAP score of 81.8%. A full account of the artifacts added when training models and the evaluation metrics for each model can be found in Table 1.

3.5 Model Applications

When applying the model to data, a threshold needs to be set to eliminate low-confidence detections. To maximize the trade-off between precision and recall, the threshold corresponding to the model's peak F1 score is used. To evaluate the applied performance of the system, we use the top-scoring model that employed Landau-Ginzberg simulation and moderate levels of image enhancement.

A straightforward application of the model is counting the defects per frame in a video. Accurately resolving the defect number as a function of time would allow direct experimental probes of the applicability of the XY model in these systems of SmC liquid crystals. The model results, as seen in Figure 7 (a, c, and e), show broad agreement with results obtained from human annotations. Furthermore, it should be noted the human annotations started when annotators judged that defects could be reliably marked. However, the model was capable of producing defect counts at significantly earlier times consistent with the observed scaling.

The spatial distribution of the defects can also be studied. The XY model makes definitive predictions for the spin-spin correlation length[23]. If the YOLO detections can accurately resolve the spatial distribution of the defects, then measuring the average defect nearest-neighbor distance would allow for a high-resolution test of the XY predictions. The accuracy of the nearest neighbor distance is demonstrated in Figure 7 (b, d, and f). Though there appears to be a systematic bias, where the YOLO detections are, on average, farther apart than the hand-labelled defects, the important dynamics are captured by the *scaling* of the nearest neighbor distance with time, which is resolved by the slope. As the the slope of both methods are consistent, this gives confidence for using the YOLO method for spatial analysis.

Another application of the YOLO model to these systems is in measuring the dynamics of isolated defects. Reliable defect tracking requires the model to be consistently capable of precisely locating defects over a larger number of frames – a common yet challenging goal in the machine learning paradigm. We make use of the Trackpy Python module, a package of functions specializing in particle tracking, to link identified objects through consecutive images over time. The end result is a linked path for each defect, as seen in Figure 8.

To numerically rate the performance of our defect tracking, we compare the track to human annotated defect locations. Error is calculated by taking the root mean square of the distance in pixels between each human annotated defect location and the nearest neighbour path. For the test case in Fig. 8, using the best performing model, we found the error to be 1.03 pixels. This shows that our machine learning pipeline is capable of tracking objects to a similar quality as a human, making it a viable method for high precision automation.

3.6 Computational Performance

Running the model on 1104x800 images takes approximately 0.07 seconds per image with an 8 second startup overhead on

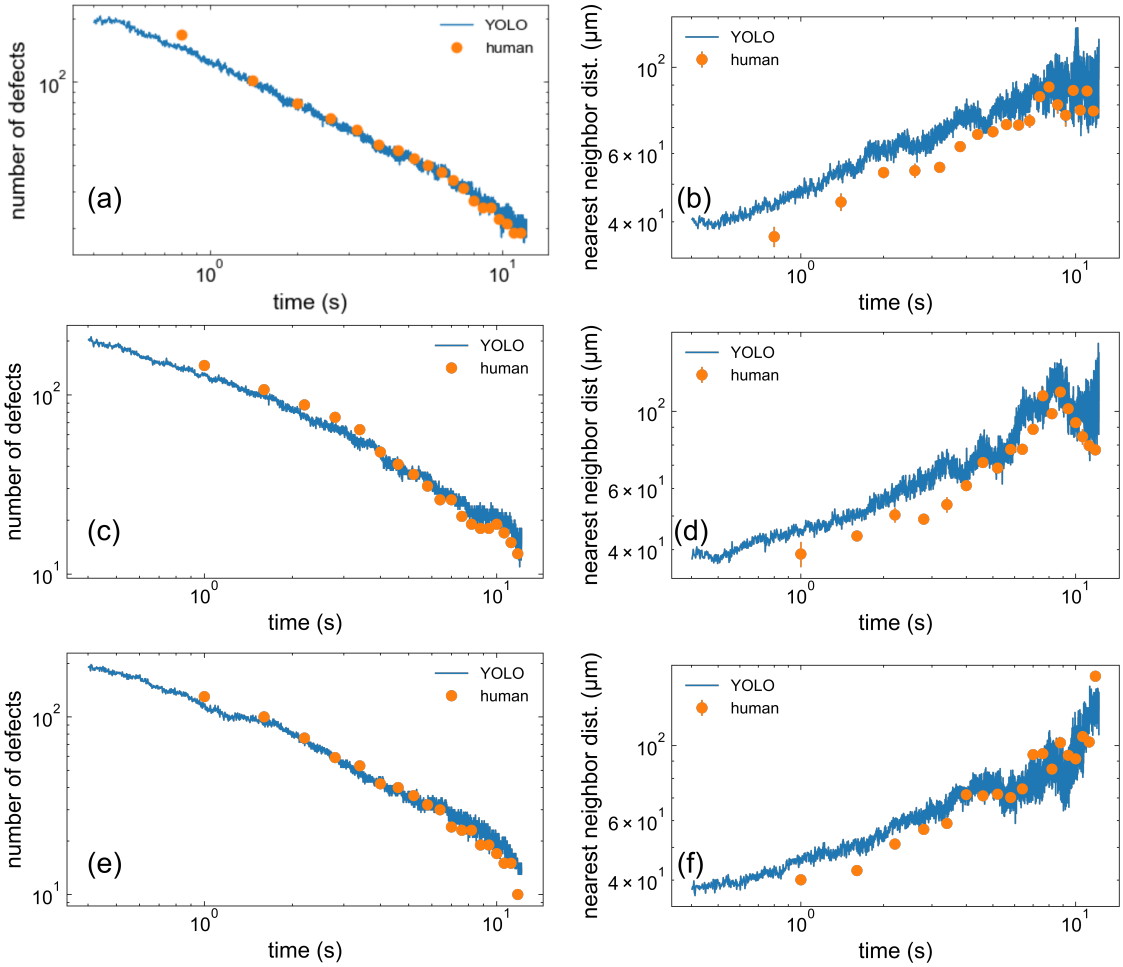

Fig. 7 Validating YOLO defect detections. (a,c,e) show number of defects vs. time for three separate videos, comparing YOLO results with hand annotated results. (b,d,f) show defect nearest neighbour distance vs. time for three separate videos, comparing YOLO results with hand annotated results.

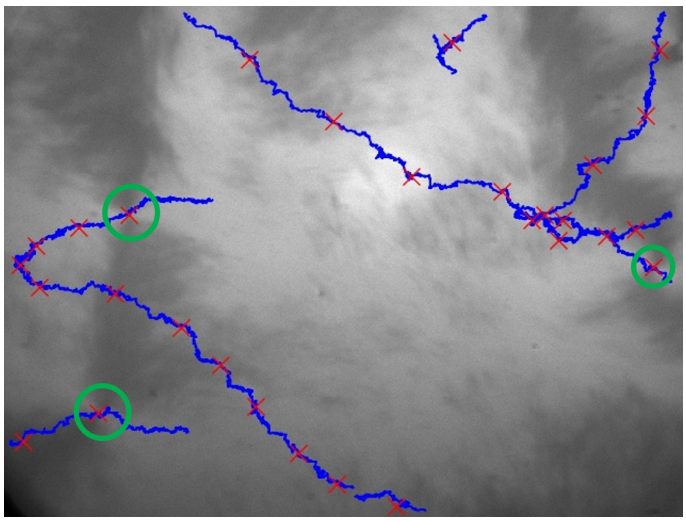

Fig. 8 Computer tracked paths are represented by lines. Human annotations are represented by X's. The defects being tracked are circled. Not all tracked defects were present in a single frame, resulting in there being more tracks than circled defects. Some false detections were made, however tracks with few defect detections were omitted for clarity.

a 2017 GeForce® GTX 1080 GPU. Using an i7-7700K CPU, the model took 4.62 seconds per image with a 10 second startup overhead. When trained on the aforementioned GPU, it took 0.51 seconds per iteration using a batch size of 8 with a 12 second startup overhead, or approximately 0.064 seconds per image. When trained on the CPU, the time per image was approximately 3 seconds. This demonstrates the viability of using a YOLO model for rapid image data analysis, especially when used in conjunction with a modern GPU. Using a GPU, a model trained for 40 epochs on a training-set of 1000 images takes approximately 1-1.2 hours. Training an identical model on the CPU is estimated to take between 20-40 hours, however this was not explicitly measured.

4 Results and Discussion

We examined the viability of using simulated images to train a modern machine learning, object detection algorithm for use in small scale applications. We demonstrate a general methodology for creating diverse training data that results in viable models with predictive power. By pairing a randomization process with the injection of characteristic experimental noise, we were able to

build viable training data from simple computational simulations.

It was found that a model trained on unmodified simulated images produced a model that performed poorly on experimental images. After increasing the diversity of simulated images via our general modification pipeline, it was found that model performance was greatly improved on experimental images, with mAP score peaking at 0.818 from a raw score of .02, with a corresponding peak F1 score of 0.811. The model resulted in comparable spatial and number resolution to the human annotations, with significant decrease in the time-per-frame (faster analysis), resulting in a dramatic increase in the time-resolution (more frames analyzed). Additionally, the model was able to out-perform human analysis in high defect density images, which significantly supplemented the usable data.

When used in conjunction with Trackpy, the model was able to track defects with an error of 1.03 pixels compared to human annotations. This could potentially be generalized to other non-trivial targets, such as active-matter nematic defects[52,53] or even tracking biological systems such as cells[54]. This method is fast, accurate, and easily trainable on new object types, making it a useful and versatile method for video data analysis.

Conflicts of interest

There are no conflicts to declare.

5 Acknowledgments

This work was supported by the Soft Materials Research Center under NSF MRSEC Grant DMR-1420736 and by NASA Grant NNX-13AQ81G.

Notes and references

1 C. Adam-Bourdarios, G. Cowan, C. Germain-Renaud, I. Guyon, B. Kégl and D. Rousseau, *Journal of Physics: Conference Series*, 2015, **664**, 072015.

2 A. Radovic, M. Williams, D. Rousseau, M. Kagan, D. Bonacorsi, A. Himmel, A. Aurisano, K. Terao and T. Wongjirad, *Nature*, 2018, **560**, 41–48.

3 A. Dey, *International Journal of Computer Science and Information Technologies*, 2016, **7**, 6.

4 C. Bishop, *Pattern Recognition and Machine Learning*, Springer-Verlag, New York, 2006.

5 K.-H. Tan and B. P. Lim, *APSIPA Transactions on Signal and Information Processing*, 2018/ed, **7**, e6.

6 P. Baldi, P. Sadowski and D. Whiteson, *Nature Communications*, 2014, **5**, 4308.

7 K. Albertsson, P. Altoe, D. Anderson, J. Anderson, M. Andrews, J. P. A. Espinosa, A. Aurisano, L. Basara, A. Bevan, W. Bhimji, D. Bonacorsi, B. Burkle, P. Calafiura, M. Campanelli, L. Capps, F. Carminati, S. Carrazza, Y.-f. Chen, T. Childers, Y. Coadou, E. Coniavitis, K. Cranmer, C. David, D. Davis, A. De Simone, J. Duarte, M. Erdmann, J. Eschle, A. Farbin, M. Feickert, N. F. Castro, C. Fitzpatrick, M. Floris, A. Forti, J. Garra-Tico, J. Gemmler, M. Girone, P. Glaysher, S. Gleyzer, V. Gligorov, T. Golling, J. Graw, L. Gray, D. Greenwood, T. Hacker, J. Harvey, B. Hegner, L. Heinrich, U. Heintz, B. Hooberman, J. Junggeburth, M. Kagan, M. Kane, K. Kanishchev, P. Karpiński, Z. Kassabov, G. Kaul, D. Kcira, T. Keck, A. Klimentov, J. Kowalkowski, L. Kreczko, A. Kurepin, R. Kutschke, V. Kuznetsov, N. Köhler, I. Lakomov, K. Lannon, M. Lassnig, A. Limosani, G. Louppe, A. Mangu, P. Mato, N. Meenakshi, H. Meinhard, D. Menasce, L. Moneta, S. Moortgat, M. Neubauer, H. Newman, S. Otten, H. Pabst, M. Paganini, M. Paulini, G. Perdue, U. Perez, A. Picazio, J. Pivarski, H. Prosper, F. Psihas, A. Radovic, R. Reece, A. Rinkevicius, E. Rodrigues, J. Rorie, D. Rousseau, A. Sauers, S. Schramm, A. Schwartzman, H. Severini, P. Seyfert, F. Siroky, K. Skazytkin, M. Sokoloff, G. Stewart, B. Stienen, I. Stockdale, G. Strong, W. Sun, S. Thais, K. Tomko, E. Upfal, E. Usai, A. Ustyuzhanin, M. Vala, J. Vasel, S. Vallecorsa, M. Verzetti, X. Vilasís-Cardona, J.-R. Vlimant, I. Vukotic, S.-J. Wang, G. Watts, M. Williams, W. Wu, S. Wunsch, K. Yang and O. Zapata, *arXiv:1807.02876 [hep-ex, physics:physics, stat]*, 2018.

8 D.-L. Deng, X. Li and S. Das Sarma, *Physical Review B*, 2017, **96**, 195145.

9 J. Carrasquilla and R. G. Melko, *Nature Physics*, 2017, **13**, 431–434.

10 M. J. S. Beach, A. Golubeva and R. G. Melko, *Physical Review B*, 2018, **97**, 045207.

11 C. Wang and H. Zhai, *Physical Review B*, 2017, **96**, 144432.

12 M. Walters, Q. Wei and J. Z. Y. Chen, *Physical Review E*, 2019, **99**, 062701.

13 A. L. Tarca, V. J. Carey, X.-w. Chen, R. Romero and S. Drăghici, *PLoS Computational Biology*, 2007, **3**, e116.

14 O. Y. Al-Jarrah, P. D. Yoo, S. Muhaidat, G. K. Karagiannidis and K. Taha, *Big Data Research*, 2015, **2**, 87–93.

15 P. Kner, B. B. Chhun, E. R. Griffis, L. Winoto and M. G. L. Gustafsson, *Nature Methods*, 2009, **6**, 339–342.

16 B. M. H. Lange, T. Sherwin, I. M. Hagan and K. Gull, *Trends in Cell Biology*, 1995, **5**, 328–332.

17 J. C. Crocker and D. G. Grier, *Journal of Colloid and Interface Science*, 1996, **179**, 298–310.

18 H. Kellay, *Physics of Fluids*, 2017, **29**, 111113.

19 J. Baumgartl and C. Bechinger, *Europhysics Letters (EPL)*, 2005, **71**, 487–493.

20 D. Conte, P. Foggia, C. Sansone and M. Vento, *International Journal of Pattern Recognition and Artificial Intelligence*, 2004, **18**, 265–298.

21 D. Erhan, C. Szegedy, A. Toshev and D. Anguelov, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 2147–2154.

22 M. B. Blaschko and C. H. Lampert, Computer Vision – ECCV 2008, 2008, pp. 2–15.

23 B. Yurke, A. N. Pargellis, T. Kovacs and D. A. Huse, *Physical Review E*, 1993, **47**, 1525–1530.

24 D. Svenšek and S. Žumer, *Physical Review E*, 2002, **66**, 021712.

25 D. Sven/u sek and S. Žumer, *Physical Review Letters*, 2003, **90**, 155501.

26 L. Radzihovsky, *Physical Review Letters*, 2015, **115**, 247801.

27 H. Pleiner, *Physical Review A*, 1988, **37**, 3986–3992.

28 A. N. Pargellis, P. Finn, J. W. Goodby, P. Panizza, B. Yurke and P. E. Cladis, *Physical Review A*, 1992, **46**, 7765–7776.

29 A. N. Pargellis, S. Green and B. Yurke, *Physical Review E*, 1994, **49**, 4250–4257.

30 P. Oswald, P. Pieranski and P. Pieranski, *Nematic and Cholesteric Liquid Crystals : Concepts and Physical Properties Illustrated by Experiments*, CRC Press, 2005.

31 R. Stannarius and K. Harth, *Physical Review Letters*, 2016, **117**, 157801.

32 N. Chattham, E. Korblova, R. Shao, D. M. Walba, J. E. Maclennan and N. A. Clark, *Physical Review Letters*, 2010, **104**, 067801.

33 K. Harth, *PhD thesis*, Otto-von-Guericke-UniversitÃď't Magdeburg, Magdeburg, Germany, 2016.

34 R. Loft and T. A. DeGrand, *Physical Review B*, 1987, **35**, 8528–8541.

35 A. Jelić and L. F. Cugliandolo, *Journal of Statistical Mechanics: Theory and Experiment*, 2011, **2011**, P02032.

36 J. Tobochnik and G. V. Chester, *Physical Review B*, 1979, **20**, 3761–3769.

37 H. T. Trinh, *Darkflow*, GitHub, 2018.

38 J. Redmon and A. Farhadi, *arXiv:1612.08242 [cs]*, 2016.

39 J. Redmon, S. Divvala, R. Girshick and A. Farhadi, *arXiv:1506.02640 [cs]*, 2015.

40 R. Girshick, J. Donahue, T. Darrell and J. Malik, *arXiv:1311.2524 [cs]*, 2013.

41 R. Girshick, *arXiv:1504.08083 [cs]*, 2015.

42 S. Ren, K. He, R. Girshick and J. Sun, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017, **39**, 1137–1149.

43 S. Lawrence, C. L. Giles and A. C. Tsoi, AAAI/IAAI, 1997, pp. 540–545.

44 J. Lever, M. Krzywinski and N. Altman, *Nature Methods*, 2016, **13**, 703–704.

45 S. Aksoy and R. M. Haralick, *Pattern Recognition Letters*, 2001, **22**, 563–582.

46 I. J. Goodfellow, J. Shlens and C. Szegedy, *arXiv:1412.6572 [cs, stat]*, 2014.

47 C. M. Bishop, *Neural Computation*, 1995, **7**, 108–116.

48 M. Kaur, D. Kumar, E. Walia and M. Sandhu, *International Journal of Computer & Communication Technology*, 2014, **3**, 5.

49 M. Koppel and J. Schler, *Computational Intelligence*, 2006, **22**, 100–109.

50 M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn and A. Zisserman, *International Journal of Computer Vision*, 2010, **88**, 303–338.

51 N. Chinchor, Proceedings of the 4th Conference on Message Understanding, Stroudsburg, PA, USA, 1992, pp. 22–29.

52 L. Giomi, M. J. Bowick, P. Mishra, R. Sknepnek and M. Cristina Marchetti, 2014, **372**, 20130365.

53 S. J. DeCamp, G. S. Redner, A. Baskaran, M. F. Hagan and Z. Dogic, 2015, **14**, 1110–1115.

54 E. Meijering, O. Dzyubachyk, I. Smal and W. A. van Cappellen, 2009, **20**, 894–902.