Multi-phase classification of liquid crystal textures using convolutional neural networks

Joshua Heaton
10133722

Department of Physics and Astronomy, The University of Manchester

MPhys project report

Project performed in collaboration with James Harbon
Supervisor: Dr Ingo Dierking

April 25, 2021

Abstract

Contents

1 Introduction

Machine learning (ML) is the term assigned to a wide range of computer algorithms that use data to automatically improve their performance on a specific task. These tasks can take various forms, including decision making, pattern recognition, and prediction [1]. A sub-field of ML known as deep learning (DL) generally consists of applying large-scale multi-layer neural networks, a type of algorithm inspired by the structure of the brain, to tasks involving highly complex abstractions of data. Such intensive algorithms typically require vast quantities of data and powerful computational resources to be trained effectively, with the advantage that they do not require any manual feature extraction [2]. With the recent explosion in availability of such data and sophisticated computing technology, DL has seen a surge in interest and application among several fields [3]. Computer vision is one such field that has been impacted greatly. Convolutional neural networks (CNNs), a type of neural network suited particularly well to grid-based data, have proven extremely successful in the tasks of image classification, segmentation, and object detection [4].

There are many thousands of individual documented liquid crystal (LC) compounds, with each displaying a certain sequence of identifiable phases between that of a liquid and solid [5]. Commonly, polarised microscopy is used to capture images of the textures produced by LC phases for identification by eye [5]. Literature on machine learning for LC phase classification is sparse, with most studies focusing on the extraction of physical properties of LCs using simulated texture data [6, 7, 8, 9], or other means [10, 11, 12, 13]. Of most relevance is the work by Sigaki et al., in which they utilise CNNs to classify simulated isotropic and nematic phases to high accuracy [6].

In this project, we prepare a novel dataset of LC texture images captured by polarised microscopy (PM), spanning multiple phases of all orders. Subsequently, we apply CNN classifier models to various phase groupings, probing the limits of attainable model accuracy. The work expands on and consolidates that of the first semester report [14], in which we demonstrated the viability of CNNs in some simple LC phase classification tasks. This report will provide a brief overview of LC phases, supervised ML, and CNNs, with further detail found in the first report [14]. Details of the models used will then be provided, followed by a presentation of the results when they are applied to each of the prepared datasets. Summary conclusions and the limitations of the study will then be discussed.

2 Background principles

2.1 Liquid crystals

Liquid crystal phases are characterised by the positional and orientational order of the molecular arrangement. In general, the phase of lyotropic LCs depends on the concentration of the sample in a solvent whilst thermotropic LCs, studied in this project, become more ordered with decreasing temperature. The order and overall structure of the LC phase determines its optical properties, in particular its birefringence. This enables images of the textures of a liquid crystal to be obtained by polarised microscopy, in which the sample is placed between two perpendicularly aligned polarisers. Polarised light incident on the set-up is altered in accordance with the current phase of the LC sample, producing characteristic features in the

resulting image.

At sufficiently high temperatures, thermotropic LCs take the form of a fully anisotropic liquid with no structural order and hence birefringence, resulting in completely dark PM textures. Upon cooling, they will display at least one higher ordered phase before reaching the fully crystalline stage. Of lowest order, just orientational, is the nematic (N) phase, in which the molecules are aligned along a particular axis, called the director, and are still free to move around as in a liquid. Compounds with chiral molecules may instead display the cholesteric (N*) phase, which is the same as the nematic phase except with helical variation of the director. Layered positional order is introduced in the smectic phase, which is divided into three distinct phase groupings. The orientation of the director further categorises these groupings. The fluid smectic (FSm) phases have no positional order within molecular layers. The orientation of the director with respect to the layer planes determines whether the phase is smectic A (SmA), in which it is perpendicular, or C (SmC) otherwise. The smectic B (SmB), I (SmI), and F (SmF) phases are placed into the hexatic smectic (HSm) group, with short-range hexagonal structures generating positional order within the layers. The soft crystal phases differ in that the layers show long-range positional order.

This project uses CNNs to classify PM textures from thermotropic chiral LC compounds, including the phases N*, SmA, SmC, SmI, and SmF.

2.2 Supervised machine learning

Machine learning algorithms can in general be categorised as supervised, unsupervised, or reinforcement learning. The ML implementations of this project are purely supervised learning algorithms. In this case, the ML model is defined as a function, parametrised by learned values $\boldsymbol{\theta}$, that maps an input data sample $\boldsymbol{x}$ containing various features to an output predicted label $\hat{\boldsymbol{y}}$,

$$\hat{\boldsymbol{y}} = f(\boldsymbol{x}; \boldsymbol{\theta}). \tag{1}$$

$\hat{\boldsymbol{y}}$ can take various forms depending on the specific task, for example regression, in which the model attempts to predict a continuous value given the input data, or classification, in which it predicts a category that the input data sample belongs to. A supervised model attempts to learn appropriate $\boldsymbol{\theta}$ values for the mapping using a set of training data, consisting of pairs, i, of input examples and their corresponding true labels, $\{\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}\}$. The model takes sample $\boldsymbol{x}^{(i)}$ and produces an output label prediction $\hat{\boldsymbol{y}}^{(i)}$ according to Equation 1. A chosen cost function $J(\boldsymbol{y}, \hat{\boldsymbol{y}}; \boldsymbol{\theta})$ is then evaluated, which generally provides a measure of the divergence of the model outputs from the true labels. The parameters are then updated with a particular optimisation algorithm in order to minimise the cost. Successful training as such results in a model that is effective in producing accurate predictions for new unseen data samples. Fixed parameters that define the precise form of f, as well as training configurations, are known as hyperparameters.

When deciding on the form of a supervised model, of great importance is its capacity, which can be thought of as the size of the model. A low capacity model with few trainable parameters may not be able to extract or infer enough features from the training data to form good predictions, known as underfitting. On the other hand, too high a capacity will cause the model to learn meaningless or overly-fine features from the training data and it may not perform well when inferring on new unseen examples. This is referred to as overfitting. The

model's hyperparameters must, therefore, be properly tuned to ensure it does not over or under fit. In addition to limiting model capacity, regularisation techniques can be applied to help prevent overfitting and improve generalisation.

Measurement of a supervised model's performance is critical in determining how well it will generalise to new unseen data, by evaluating a metric on a dataset of samples. As an example, for classification tasks a common metric is the percentage of samples that the model assigned to the correct class. Often, the training dataset is split into three subsets: training, validation, and test. The training set contains the data samples used to update the trainable parameters of the model. The validation set is used to tune the hyperparameters of the model. Evaluation of a trained model on the training and validation sets can reveal if the model has underfitted or overfitted. In the former case a low performance on both sets will be observed, whereas for the latter a high performance on the training set and low on the validation set will occur. Hyperparameters can then be adjusted accordingly before retraining the model. After a satisfactory model configuration and validation performance are reached, it is evaluated on the as-of-yet unseen test set to give an indication of the model's generalisation error.

2.3 Convolutional neural networks

2.3.1 Layers

Neural networks are a type of ML algorithm that pass input data through a series of layers, each containing a number of units. Every unit of a layer is connected in a particular way to the units of the previous layer. Units can take various forms including ones with or without trainable parameters, and specific layers are engineered so as to identify, manipulate, and propagate data features from the input to the output of the network. Perhaps the most simple type, in a dense layer the input to each unit is the outputs of all units of the previous layer. The inputs are multiplied by the unit's learned weight parameters and summed together with a learned bias parameter to calculate the unit's output. An activation function can then be applied to the output of each unit of the layer to introduce non-linearity to the network. Such non-linearities are essential in allowing a neural network to act as a universal function approximator, allowing it to perform highly complex inference on input data. A dense layer can hence be represented in matrix form as

$$\boldsymbol{O} = A(\boldsymbol{W}\boldsymbol{I} + \boldsymbol{B}), \tag{2}$$

where $\boldsymbol{O}$ is the vector containing the outputs for the layer, $\boldsymbol{W}$ is the matrix of layer weights, $\boldsymbol{I}$ is the vector of layer inputs, $\boldsymbol{B}$ is the vector of bias parameters, and A is the activation function applied element-wise.

Convolutional layers take grid-based input data such as two-dimensional images, and convolve it with a kernel of trainable parameters. The kernel has a width and height smaller than that of the input. The kernel is moved iteratively over the input, with the distance moved each iteration known as the stride of the convolution. At each step the kernel's parameters are multiplied with aligned input values, with the results summed together to produce the final output value. An activation function is often applied to this output value. The complete layer output is formed as a grid containing the ordered output values from the convolution. The size of the output grid depends on the dimensions of the kernel and the stride of the convolution, as these together determine the number of convolution steps in each direction. The input and

output of the layer can have a third dimension, for example with the three colour channels of an RGB image. In this case, the kernel will have a different set of parameters for each channel. The number of channels can be increased from input to output by stacking the results from multiple kernels. A convolutional layer's padding refers to the behaviour of the kernel at the edges of the input. When the kernel is confined completely within the input space it is called valid padding. Same padding is when the kernel extends beyond the input space such that each value is visited by the kernel the same number of times, with kernel values outside the space multiplied by zero. For a stride of one in both directions, same padding will result in an output with the same shape as the input, and valid padding will result in dimensionality reduction.

The main advantage of convolutional layers over dense layers is the greatly reduced computational and memory cost, since they require far fewer parameters. The kernel parameters are shared over the entire input, which can also improve regularisation. Each kernel can be thought of as learning to extract a particular feature from the input to be passed on to the next layer, such as edges of objects in an image.

Pooling layers are often used after convolutional layers to reduce the dimensions of the network. They work in an analogous way to convolutional layers, however, the kernel has no trainable parameters to convolve and instead performs a specific operation. This could be, for example, taking the maximum value from the aligned input values at each step, known as max pooling. For pooling layers the kernel size is instead known as the pool size, and the pooling operation is applied to each input channel individually. Global pooling refers to the case in which the pool size is equal to the input size, which results in a vector output with one value for each of the input channels. Pooling layers are utilised to reduce the overall size and computational cost of the network whilst providing it with some invariance to translations of the input.

2.3.2 Training

3 Summary of previous work

4 CNN Models

4.1 Sequential

4.2 Inception

4.3 ResNet

5 Methodology

5.1 Data preparation

5.2 Model training configurations

6 Classification tasks and results

6.1 Cholesteric and smectic

6.2 Smectic A and C

6.3 Smectic I and F

6.4 Cholesteric, fluid smectic and hexatic smectic

6.5 Cholesteric, smectic A, C, and hexatic smectic

7 Conclusions

References

[1] K. Murphy, *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[3] A. Shrestha and A. Mahmood, "Review of deep learning algorithms and architectures," *IEEE Access*, vol. 7, pp. 53040–53065, 2019.

[4] A. Voulodimos, N. Doulamis, A. Doulamis, E. Protopapadakis, and D. Andina, "Deep learning for computer vision: A brief review," *Intell. Neuroscience*, vol. 2018, 2018.

[5] I. Dierking, *Textures of Liquid Crystals*. WILEY-VCH, 2003.

[6] H. Y. D. Sigaki *et al.*, "Learning physical properties of liquid crystals with deep convolutional neural networks," *Scientific Reports*, vol. 10, p. 7664, 2020.

[7] H. Y. D. Sigaki, R. F. de Souza, R. T. de Souza, R. S. Zola, and H. V. Ribeiro, "Estimating physical properties from liquid crystal textures via machine learning and complexity-entropy methods," *Phys. Rev. E*, vol. 99, p. 013311, 2019.

[8] E. N. Minor *et al.*, "End-to-end machine learning for experimental physics: using simulated data to train a neural network for object detection in video microscopy," *Soft Matter*, vol. 16, p. 1751, 2020.

[9] M. Walters, Q. Wei, and J. Z. Y. Chen, "Machine learning topological defects of confined liquid crystals in two dimensions," *Phys. Rev. E*, vol. 99, p. 062701, 2019.

[10] F. Leon, S. Curteanu, C. Ta, L. Lin, and N. Hurduc, "Machine learning methods used to predict the liquid-crystalline behavior of some copolyethers," *Molecular Crystals and Liquid Crystals*, vol. 469, 2007.

[11] C. Butnariu, C. Lisa, F. Leon, and S. Curteanu, "Prediction of liquid-crystalline property using support vector machine classification," *Journal of Chemometrics*, vol. 27, no. 7-8, pp. 179–188, 2013.

[12] H. Doi, K. Takahashi, K. Tagashira, J. Fukuda, and T. Aoyagi, "Machine learning-aided analysis for complex local structure of liquid crystal polymers," *Scientific Reports*, vol. 9, 2019.

[13] T. Inokuchi, R. Okamoto, and N. Arai, "Predicting molecular ordering in a binary liquid crystal using machine learning," *Liquid Crystals*, vol. 47, no. 3, pp. 438–448, 2020.

[14] J. Heaton, "Classification of liquid crystal textures using machine learning," 2020.

Appendices