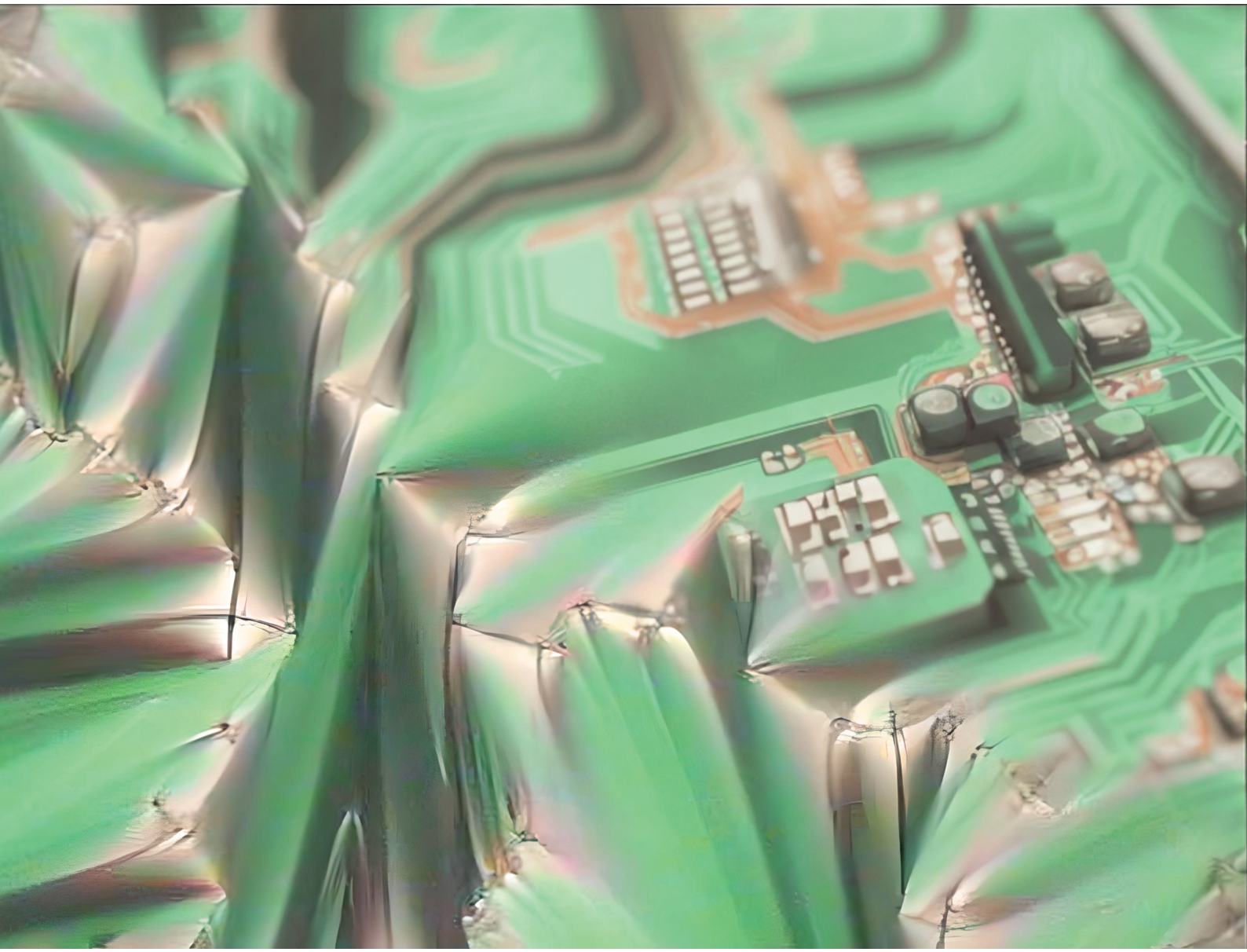
Volume 19
Number 39
21 October 2023
Pages 7471–7674

Soft Matter

rsc.li/soft-matter-journal



ISSN 1744-6848

ROYAL SOCIETY OF CHEMISTRY

PAPER
Rebecca Betts and Ingo Dierking
Machine learning classification of polar sub-phases in liquid crystal MHPOBC

Soft Matter

PAPER

View Article Online
View Journal | View Issue

Cite this: *Soft Matter*, 2023, **19**, 7502

Machine learning classification of polar sub-phases in liquid crystal MHPOBC

Rebecca Betts and Ingo Dierking *

Received 9th July 2023,
Accepted 21st August 2023

DOI: 10.1039/d3sm00902e

rsc.li/soft-matter-journal

Experimental polarising microscopy texture images of the fluid smectic phases and sub-phases of the classic liquid crystal MHPOBC were classified as paraelectric (SmA*), ferroelectric (SmC*), ferrielectric ($SmC_{1/3}$*), and antiferroelectric (SmC_A*) using convolutional neural networks, CNNs. Two neural network architectures were tested, a sequential convolutional neural network with varying numbers of layers and a simplified inception model with varying number of inception blocks. Both models are successful in binary classifications between different phases as well as classification between all four phases. Optimised architectures for the multi-phase classification achieved accuracies of (84 ± 2)% and (93 ± 1)% for sequential convolutional and inception networks, respectively. The results of this study contribute to the understanding of how CNNs may be used in classifying liquid crystal phases. Especially the inception model is of sufficient accuracy to allow automated characterization of liquid crystal phase sequences and thus opens a path towards an additional method to determine the phases of novel liquid crystals for applications in electro-optics, photonics or sensors. The outlined procedure of supervised machine learning can be applied to practically all liquid crystal phases and materials, provided the infrastructure of training data and computational power is provided.

1 Introduction

Artificial Intelligence (AI) and Machine learning (ML) algorithms have existed since the 1950s[1] but have only recently seen significant improvements allowing them to be fully utilised. This is particularly due to increased computing power, but also developments in the algorithms themselves such as the introduction of non-linearity and the back-propagation algorithm. There has also been an increase in the availability of the large datasets required for ML.[2] Convolutional neural networks are a type of ML algorithm which use a convolution operation to extract features from input images or other grid-based data [ref. 3 and references therein]. They have shown particular success in medical imaging applications[4–6] especially in cancer research[7,8] for example in mitosis detection in breast cancer histology images.[9] The use of machine learning can currently be followed through all fields of sciences, for example biology through the use in biosensors,[10] in chemistry for drug discovery[11] and in the computer-aided synthesis planning,[12] in material science[13] and physics[14] for example in nano-photonics,[15] but also astronomy[16–18] and particle physics.[19]

Besides this admittedly inconclusive list of examples, machine learning is of course also employed in the study of condensed matter phases and phase transitions,[20] which links to the investigations of liquid crystals and soft matter.[21–23] The success of machine learning techniques in the characterization of liquid crystals has so far been demonstrated in various aspects of LC research. The prediction of phase transitions has been one of the major issues,[24–27] with work concentrating on the isotropic to nematic transition of thermotropic liquid crystals being studied with simulated Schlieren-textures but also with experimental images. Other investigations were concerned with theoretical predictions of the molecular ordering of binary mixtures of molecules with different length,[28] the self-assembled nanostructures of lyotropic liquid crystals[29] and the local structure of liquid crystalline polymers.[30] Another issue, which is still in its infancy is the prediction of physical properties by machine learning, as demonstrated recently by a comparison of the experimental and predicted dielectric properties of a nematic liquid crystal.[31]

Much of the machine learning work on liquid crystals so far is related to object detection[32] and specifically directed towards topological defects in nematics, *i.e.* the above mentioned Schlieren-textures. For such investigations simulated[33] as well as experimental[34] textures have been employed. A particularly interesting example was that of an active nematic which was studied in relation to hydrodynamics.[35] Proposing a similar background, an investigation has been reported where machine learning is employed to detect islands and bubbles in smectic films,[36] providing an accuracy which exceeds conventional tracking software.

Department of Physics and Astronomy, University of Manchester, Oxford Road, Manchester M139PL, UK. E-mail: ingo.dierking@manchester.ac.uk

Considering the applicational side of machine learning being used in relation to liquid crystals, one needs to mention a variety of sensors,[37] which are all based on stimulated texture transitions of nematic LCs. The general idea was introduced by the Abbott group[38] and later applied to biochemical sensors,[39] for example in the detection of endotoxins from different bacterial species,[40] but also during the recent pandemic for the detection of SARS-CoV-2.[41] With a rather similar experiment the general sensor ideas can also be employed to detect gases[42] and gas mixtures.[43]

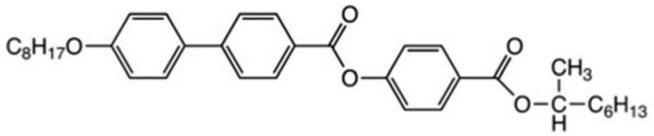
In recent times we have tested methodologies to apply machine learning algorithms not only for the simple transition from the isotropic to nematic phase, or texture transitions from homeotropic to planar, but rather for the identification of whole liquid crystal phase sequences.[44] These were based on convolutional neural networks as well as inception models with varying numbers of layers, inception blocks, and regularization methods. They were applied to a range of different phases including isotropic, nematic, cholesteric, smectic A, smectic C, and the hexatic phases smectic I and smectic F.[45] While binary classifiers resulted in prediction accuracies of larger than 97%, even the prediction of the whole phase sequence, had an accuracy of about 90%, which is certainly satisfactory for a first prediction attempt with a limited number of images and computing facilities. Also phase transitions and transition temperatures could in most cases be very well predicted[46] with accuracies well above 90% and relative temperature accuracies limited by the experimental resolution of the employed hot stage ($\pm$0.1 K).

In this study we extend similar phase identification methodologies to chiral liquid crystals and their polar phases. In particular, we have investigated the classic material 4-(1-methylheptyloxycarbonyl)phenyl 4′-octyloxybiphenyl-4-carboxylate, (MHPOBC). This exhibits on cooling the paraelectric smectic A* phase, smectic C_α*, ferroelectric smectic C*, ferrielectric smectic $C_{1/3}$* and antiferroelectric smectic C_A*. It will be demonstrated that in addition to different classes of phases (fluid, hexatic, soft crystal) machine learning is also able to distinguish between more subtle phases, when appropriately trained.

2 Experimental

2.1 Liquid crystal material and phases

The discovery of ferroelectricity in liquid crystals goes back to the symmetry arguments of Meyer *et al.* and was first demonstrated in a pioneering publication[47] in 1975. In essence it was argued that every chiral, tilted smectic phase can in principle exhibit a spontaneous polarization PS and would therefore be pyroelectric. If this spontaneous polarization can be switched between two stable states by an applied electric field, the material is termed to be ferroelectric. This would for example be the case for the fluid smectic C* (SmC*) phase and the hexatic smectic I* (SmI*) and smectic F* (SmF*) phases. The asterix (*) here denotes molecular chirality. Yet, in order to compensate the spontaneous polarization over short distances, the ferroelectric phases adopt a helical superstructure, due to the coupling between polarization and tilt angle, which in optical polarization microscopy (POM) is often visible as an equidistant line pattern. More correctly, the SmC* phase should therefore be called helielectric, because the polarization is compensated. Nevertheless, when subjecting the phase to boundary conditions such that the cell gap is much smaller than the helical pitch, the helix is unwound and a domain texture with two stable states is observed. This is called the surface stabilized ferroelectric liquid crystal (SSFLC) geometry, invented by Clark and Lagerwall,[48] which is of outmost importance for applications of ferroelectric liquid crystals (FLC). In the investigations presented here, we will rely on bulk samples, which in POM show a fan-like texture (SmA*) or broken fan-like texture (SmC*) with focal conics.[49] Several years later, antiferroelectricity was demonstrated for the compound MHPOBC,[50–52] which is a chiral, calamitic mesogen with the following structural formula:



Later, other subphases were discovered, such as the SmC_α* phase, located between SmA* and SmC*,[52] which exhibits an (incommensurate) helical superstructure with very small pitch, well below the resolution limit of POM,[53] which explains why this phase can hardly be distinguished from SmA* by microscopy alone. And indeed, also machine learning algorithms are therefore not capable to distinguish both phases, because there are no structurally distinguishing features observed by microscopy. This is the reason why we will only shortly mention the SmC_α* phase below in the results section and otherwise omit it from further investigation.

At last, the so-called ferrielectric phase[54,55] $SmC_{1/3}$* is found between the ferroelectric SmC* and the antiferroelectric SmC_A* phase. (In fact, this phase should also be called helielectric, because it does not exhibit a polarization at zero applied electric field.) There is one striking structural feature that distinguishes the ferrielectric phase from both its ferroelectric as well as its antiferroelectric counterparts. In the ferrielectric phase the focal conic defects split up[56] as shown in Fig. 1.

At lower temperatures when the ferrielectric to antiferroelectric phase transition is crossed, the defects will merge again. It physical reason for this phenomena does not seem to be unveiled, but is observed for different compounds as well and can thus be recognized by machine learning as a distinguishing structural feature. The phase sequence of the fluid phases as determined by polarizing microscopy on cooling is thus: Iso 149 SmA* 122 SmC_α* 121 SmC* 119 $SmC_{1/3}$* 118 SmC_A* and is shown in Fig. 2.

Experimental image acquisition was carried out with a Leica DMLP polarizing microscope in combination with a Linkam TP94 temperature controller and LTSE350 hot stage, equipped with a digital camera (IDS uEye at resolution of 2048 $\times$ 1088 pixels). Already visually from Fig. 2 it becomes clear that the classification

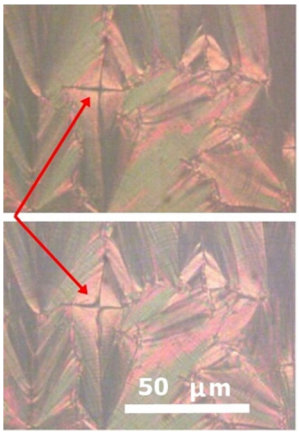


Fig. 1 At the transition from ferroelectric SmC* to ferrielectric $Sm_{1/3}$* focal conic defects split up, a structural feature which can be recognized by machine learning. At the transition from ferrielectric $Sm_{1/3}$* to antiferroelectric SmC_A* the defects merge again.

of the different phases of MHPOBC will not be a trivial task by machine learning algorithms, because the structural features distinguishing between the different phases are minimal, especially as changes in birefringence, *i.e.* slight color changes in the texture photographs cannot be chosen as a criteria.

2.2 Machine learning for classification

This project aims to develop a machine learning algorithm to classify LC phases using polarising microscopy images of their textures. The algorithm is some function mapping the pixel values of the input image, a matrix $\boldsymbol{X}$, to a vector of the probabilities of the image belonging to each possible images from the phase, which can be taken as the predicted phase label, $\hat{\boldsymbol{y}}$,

$$\hat{\boldsymbol{y}} = f(\boldsymbol{X},\boldsymbol{\theta}) \quad (1)$$

where $\boldsymbol{\theta}$ are parameters of the function which will be learned during training. The training consists of each image in a set of N images, $\{\boldsymbol{X_n}\}$, being substituted into the function in eqn (1), and comparing the output, $\hat{\boldsymbol{y}}_n$, to the true label (LC phase), $\boldsymbol{y_n}$. The difference between the predicted and true labels is evaluated with a chosen loss function (discussed further in Section 2.2.1), $L(\hat{\boldsymbol{y}}_n, \hat{\boldsymbol{y}}_n)$, which is used to update the parameters $\boldsymbol{\theta}$ such that the loss function is minimised, hence the predicted labels are closer to the true labels. Through training with many examples (large N), the algorithm should be able to generalise and predict the phases of unseen images.[57] Generally, the entire training dataset is passed through the function multiple times, with each pass being called an epoch. This training process will be discussed in more detail for the case of neural networks below.

2.2.1 Neural networks. Neural networks are a subset of machine learning algorithms, originally designed to mimic biological neural pathways. Their structure consists of a layer of input nodes (the pixel values $\boldsymbol{X_n}$), followed by a number of hidden layers, each with a chosen number of nodes, followed by the output layer. The number of nodes in the output layer will be equal to the number of possible phases, with each node giving the probability of the input image belonging to its corresponding phase, representing the vector $\hat{\boldsymbol{y}}_n$ above. All nodes in adjacent layers are connected, demonstrated in Fig. 3. Each node has an activation $a_i^{[l]}$, analogous to a neuron firing or not firing in a biological neural network, and a bias $b_i^{[l]}$. Here the superscript $[l]$ refers to the lth layer of the network, and subscript i gives the ith node. Each connection between nodes has a weight $w_{ik}^{[l]}$, referring to the connection between the ith node of the lth layer and the kth node of the previous layer. The activation of each node is determined by the biases and weights of all previous layers (as well as the input), according to

$$a_i^{[l]} = f^{[l]}\left(\sum_k w_{ik}^{[l]} a_k^{[l-1]} + b_i^{[l]}\right). \quad (2)$$

ref. 58. The function $f^{[l]}$ is known as the activation function and can take many different forms and differ between layers. In general, it should be non-linear, as a linear function would result in the network behaving as just one layer (because the sum of all layers would still be a linear function), hence the multi-layer neural network structure would be obsolete. A commonly used activation function is the rectified linear unit (ReLU) which has the form,

$$f^{[l]}(z_i) = \max(0, z_i) \quad (3)$$

where z_i is the node output (the term in brackets in eqn (2)). It is computationally inexpensive, allowing fast training, whilst still satisfying the condition of being non-linear. It also mimics biological neurons, with the lowest possible activation being zero. For classification networks, the output layer activation function is generally the softmax function, given by

$$f_i^{[l]}(\boldsymbol{z}) = \frac{e^{z_i}}{\sum_j e^{z_j}}, \quad (4)$$

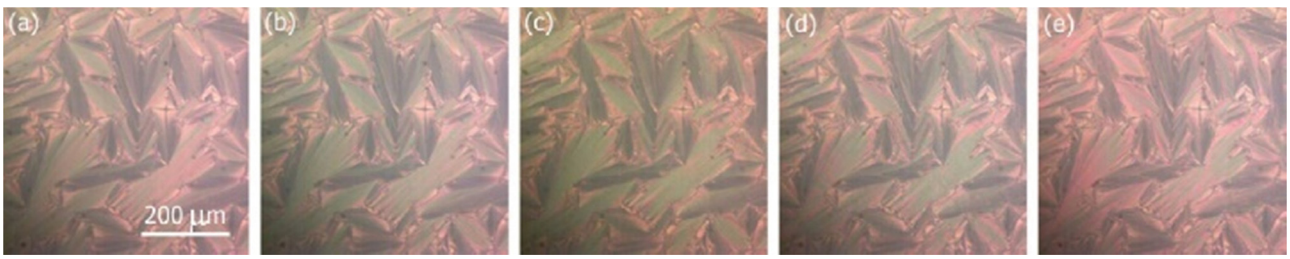


Fig. 2 POM texture dataset of (from left to right) showing the (a) paraelectric SmA* and the (b) SmC_α* phase, (c) ferroelectric SmC*, (d) ferrielectric $SmC_{1/3}$* and the (e) antiferroelectric SmC_A* phase.

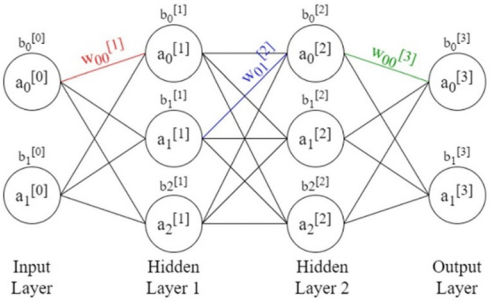


Fig. 3 Schematic of neural network with two hidden layers and weights, biases, and activations labelled.

such that for node output z_i, the activation is defined by all activations in the layer, $\boldsymbol{z}$. This allows each node activation in the final layer to correspond to the predicted probability of the input belonging to each phase, with all activations summing to one.[58]

As discussed above, the training examples are passed through the network and the parameters of the function in eqn (1), the weights and biases, are updated. Often, however, the training data is split into equally sized batches, and the parameters are updated after each batch is passed through the network, speeding up training. The loss function is taken as the average of the loss functions of each training example in the batch. Batches are used primarily to reduce the amount of memory required to load the data. Because the loss function is dependent on all parameters in the network, its derivative with respect to each parameter can be calculated, starting with the final layer, then applying the derivative chain rule for each preceding layer, working backwards in a process known as backpropagation. This gives the gradient of the loss function in the dimensional space of all parameters. By choosing a learning rate, α, each weight and bias can be updated as

$$w_{ik}^{[l]} \rightarrow w_{ik}^{[l]} - \alpha\frac{\partial L}{\partial w_{ik}^{[l]}},\ b_i^{[l]} \rightarrow b_i^{[l]} - \alpha\frac{\partial L}{\partial b_i^{[l]}}, \tag{5}$$

hence minimising the loss function through gradient descent. The method of updating the parameters is known as the optimization algorithm, and this particular method is called stochastic gradient descent. A common improvement, and the algorithm used in this project, is Adaptive Moment Estimation (Adam) which minimises the loss function in fewer steps in two ways: encouraging faster gradient descent in directions it has previously decreased (momentum) and allowing the learning rate to vary between parameters.[58]

The loss function gives a measure of the difference between the true and predicted labels and can be defined in many ways. For classification tasks, categorical cross-entropy, L_{CE}, is generally used. This is defined for a training example with true label, $\boldsymbol{y_n}$, and predicted label, $\hat{\boldsymbol{y}}_{\boldsymbol{n}}$, as

$$L_{CE} = -y_n \cdot \log \hat{y_n}. \tag{6}$$

As discussed above, the true and predicted labels are vectors with each element representing the probability of the image belonging to each class. Hence, for an image belonging to the first class, the true label, $\boldsymbol{y_n}$, will have the form [1, 0, 0, 0] T (assuming four classes). This is known as one-hot encoding. Therefore, the vectors $\boldsymbol{y_n}$ and $\hat{\boldsymbol{y}}_{\boldsymbol{n}}$ can be considered as two discrete probability distributions, and categorical cross-entropy gives a measure of the difference between these two distributions.[59]

2.2.2 Convolutional neural networks. Convolutional neural networks (CNNs) are a specific type of neural network used when the inputs are images or other grid-based data. Rather than the trainable parameters from eqn (1) being the weights and biases of connected nodes, the trainable parameters are elements of a tensor (kernel) which is passed over the grid of the previous layer. For the first convolutional layer, this means passing the kernel over the grid of pixel values representing the input image. At each step the convolution operation, demonstrated in Fig. 4, is then applied; the aligned input and kernel values are multiplied and then summed over the kernel. This results in a feature map, showing areas of similarity to the kernel. Usually, at each convolutional layer, multiple kernels are passed over the input and the resulting feature maps stacked, meaning there are multiple channels input to the next layer. Therefore, the kernel for the next layer must have a depth equal to the number of channels, and the number of channels increases at each subsequent layer. The distance moved by the kernel at each step is known as the stride. In a CNN, each convolutional layer is followed by a pooling layer. This again involves a kernel passing over the input, but in this case, the kernel has no trainable parameters and instead outputs either the maximum (max pooling) or the arithmetic mean (average pooling) of the kernel area. This reduces the width and height of the input while keeping the depth (equivalent to the number of channels) the same. CNNs generally consist of a number of convolutional and max pooling layers, followed by a number of fully connected (dense) layers.

One advantage of convolutional neural networks is that they make use of the same trainable parameters over the whole area of the image, reducing computational cost in comparison to the standard neural networks discussed previously.[60]

2.2.3 Regularisation. Overfitting is a common problem in neural networks, particularly when the training dataset size is limited. It occurs when the network learns to exactly classify the training images and cannot generalise to new unseen data. To

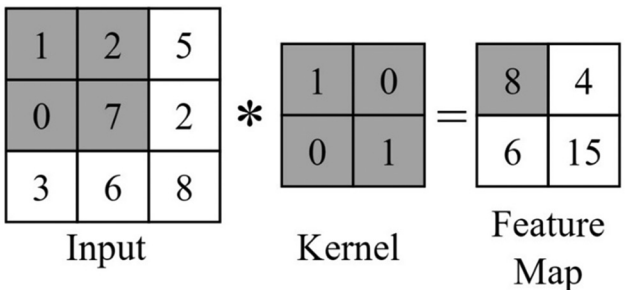


Fig. 4 Demonstration of the convolution operation (written as *) with stride 1 and a 2 × 2 kernel on a 3 × 3 input. For example, the element of the feature map with value 8 is calculated as (1 × 1) + (2 × 0) + (0 × 0) + (7 × 1) (the highlighted grey elements).

diagnose overfitting, the dataset sample is split into training, validation, and testing datasets. Only the training dataset is used to train the network, and after each epoch the network is tested on the validation set. If the network is overfitting, the validation set accuracy will be lower than the testing set accuracy, and the loss function higher. The testing dataset is used after all training epochs to find the accuracy of the network. Overfitting occurs when the network is too big, *i.e.* has too many parameters in comparison to the number of training examples. It is not always possible to increase the training dataset size, and reducing the size of the network can lead to underfitting. In this case the network does not learn any features and both the training and validation accuracies are low. Other methods used in this project to reduce overfitting are detailed below.

Data augmentation involves artificially increasing the dataset size by creating modified copies of the training set images. Based on previous studies,[28,45] each image in the training dataset was reflected horizontally and vertically, tripling the dataset size.

Dropout regularisation involves randomly setting the output of nodes to zero with a given probability (the dropout rate). It is applied to dense layers only and prevents the network relying on any particular feature by forcing new pathways to be created. As a result, the network takes more epochs to train.[61]

When a network overfits, particular weights will grow to be very large. L2 regularisation discourages this by adding a term to the loss function (eqn (7)),

$$+\lambda \sum_{l} \sum_{i} \sum_{j} \left(w_{ij}^{[l]} \right)^2. \tag{7}$$

The sum of all the squared weights is multiplied by an L2 regularisation parameter, λ, penalizing large weights. L1 regularisation (in which the absolute values of the weights are summed) has a similar effect but encourages zero weights leading to many features being ignored by the network.[62] In this particular case of texture classification, where many features are required to identify the texture, L1 regularisation can arbitrarily select a particular feature, resulting in an unstable solution. Therefore, L2 regularisation was chosen.

Batch normalisation involves rescaling the outputs to a layer using two trainable parameters, γ and β, such that that output, z, is normalised as

$$z \to \gamma \left(\frac{z - \mu}{\sigma} \right) + \beta, \tag{8}$$

where μ and σ are the mean and standard deviation of z, respectively, calculated over the current training batch. This has a regularization effect by adding noise to the network, whilst also increasing the speed at which it trains and stabilizing the model.[63,64]

3 Methodology

3.1 Configuring models

Two different architectures of CNNs where compared according to their performance in the task of classifying LC phases, the sequential and inception models detailed below. Both models used the Adam optimisation algorithm,[65] categorical cross-entropy as the loss function and ReLU activation in all layers except for the output (which uses softmax activation). All convolutional layers had a stride of 1×1 and were followed by batch normalisation. Dropout was applied to all dense layers, and L2 regularisation to both dense and convolutional layers. All models were trained with a batch size of 64. The binary classifiers used a learning rate of 5×10^{-5} and the multiphase classifiers used a learning rate of 1×10^{-4}.

The hyperparameters, parameters that control the learning process such as learning rate and number of convolutional layers, were optimised for each model, based on the criteria of achieving the greatest validation accuracy. Each model was monitored during training by plotting both the training and validation datasets' accuracy and loss at each epoch, providing the learning curve. Successful training could be recognised by similar training and validation curves, reaching a plateau at the maximum accuracy and minimum loss. An illustrative example for the accuracy and loss learning curves is depicted in Fig. 5. All possible combinations of hyperparameters could not be tested, so optimisation was achieved by beginning with a small model (for example one convolutional and one dense layer) and then increasing the model size incrementally until overfitting was seen. Other hyperparameters such as batch size, learning rate and dropout rate were also varied until suitable learning curves were seen. All models showed some spikes in the validation loss and corresponding dips in the validation accuracy throughout training, possibly due to the limited dataset size meaning there was some dissimilarity between the validation and training datasets. These spikes were reduced by decreasing the learning rate, however could not be minimised entirely as this also significantly increased the training time. To prevent an epoch with very low validation accuracy being used to evaluate the test dataset, the epoch with highest validation accuracy was saved and tested. This was repeated three times, *i.e.* through three training cycles with the same hyperparamters, and the mean and standard deviation of the test dataset accuracy was calculated. There is some further uncertainty on the test accuracy due to the finite size of the test dataset, however this was found to be negligible. Therefore, the uncertainty on each accuracy is given just by the standard deviation of the three test accuracies. Both

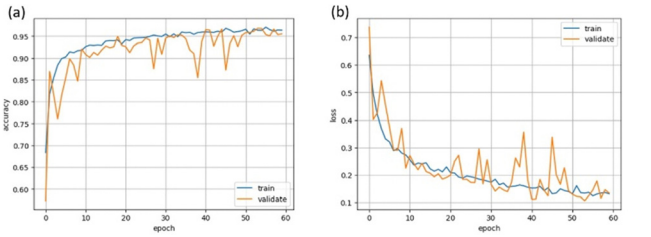


Fig. 5 Exemplary learning curves for (a) accuracy and (b) loss for the training (blue) and the validation (orange) set. The fact that both curves reach a plateau converging towards similar values of accuracy and loss indicates that the network is learning well without any under- or over-fitting.

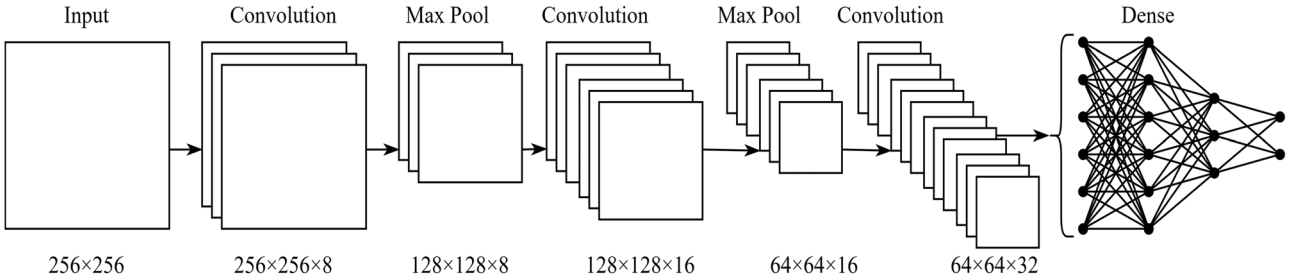


Fig. 6 Schematic of example sequential CNN, showing the dimensions of each layer output. Note that the final convolutional layer is passed to the dense layers *via* global average pooling.

models were trained using GPUs provided by Google Colaboratory[66] and implemented with Keras[67] and TensorFlow libraries.[68]

3.1.1 Sequential model. In the sequential model, the input is passed through a series of alternating convolutional layers of kernel size 3 × 3 and max pooling layers of kernel size 2 × 2. This results in a decrease in the width and height of each layer and an increase in the depth, equivalent to the number of channels or output feature maps from a convolutional layer. Global average pooling, where the kernel size is equal to the width and height of the layer, is then applied to the final convolutional layer, reducing the dimensions and leading to a dense layer with the number of nodes equal to the depth of the previous layer. This is followed by a number of dense layers, with decreasing number of nodes. Finally, the output layer has an equal number of nodes to the number of possible labels. This structure is demonstrated in Fig. 6.

3.1.2 Inception model. The Inception-v3 network is a CNN architecture intended to extract features of various scales through the use of different kernel sizes, as well as decrease the computational cost in comparison to a sequential CNN by running the convolutions in parallel. This is achieved through a series of inception modules, shown in Fig. 7. Note the 1 × 1 convolutions; these do not learn spatial features in the data but are instead used to decrease the number of channels as well as learning features that occur across the depth of the input between channels. The inception modules are followed by global average pooling and a number of dense layers, as in the sequential model. The inception-v3 network has $2.39 \times 10^{+7}$ parameters,[69] so will likely overfit if applied to the limited dataset size used in this project. Therefore, a simplified version is used, consisting of a small number of inception modules (two or three), then the dense layer structure outlined above. When describing specific model architectures in Section 4, the "number of channels in an inception module" will refer to the value *N* in Fig. 7.

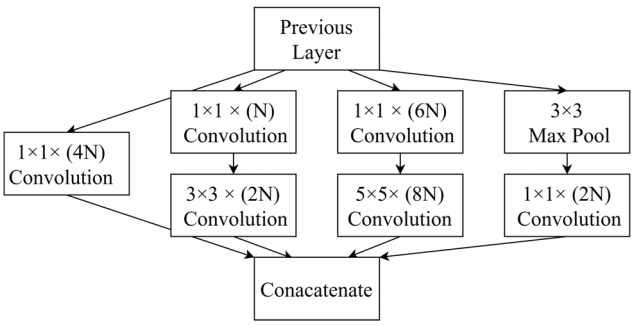


Fig. 7 Schematic of the inception module showing kernel sizes of each convolution and max pooling operation. The numbers in brackets show the number of channels in each convolution.

3.2 Data preparation

Samples of MHPOBC were prepared between glass slides without treatment for boundary conditions, such that they had a thickness of approximately 6 μm. Videos across phase transitions were recorded and images extracted *via* the VLC media player,[70] with their true phase labels being assigned based on the POM characterisation. Each 2048 × 1088 pixel image was split into six 540 × 540 pixel squares. These were each converted to greyscale, scaled to a resolution of 256 × 256 and reflected horizontally and vertically to augment the dataset as discussed in Section 2.2.3.

To prevent too much similarity between the training, validation and testing sets, the videos (rather than the images) were split between the sets at an approximate ratio of 70 : 15 : 15, respectively. If similar images taken from the same video would have been split between the three sets, the accuracy of the network on unseen data would be overestimated. By comparing the final accuracies of each dataset, the effect of bias in the input data could be monitored. Significant bias would lead to the network being unable to perform on unseen data, resulting in a low test accuracy. This is particularly important as the data augmentation applied propagates any bias in the original data. The number of images belonging to each phase is shown in Fig. 8.

At this point it is also reasonable to make a general comment about the determination of liquid crystal phases *via* neural networks. Given the fact that this is implemented by supervised machine learning, phase identification does of course depend on the ability of experts to correctly identify the phases of the training data. The identification of previously unseen data is therefore not resistant against expert errors in the selection of the training data. Yet, this is equivalent for any polarising microscopic study. For a large-scale implementation of machine learning to characterise liquid crystalline phase sequences of novel compounds, it will thus be necessary to collect large quantities of data images of different materials, which will be independently

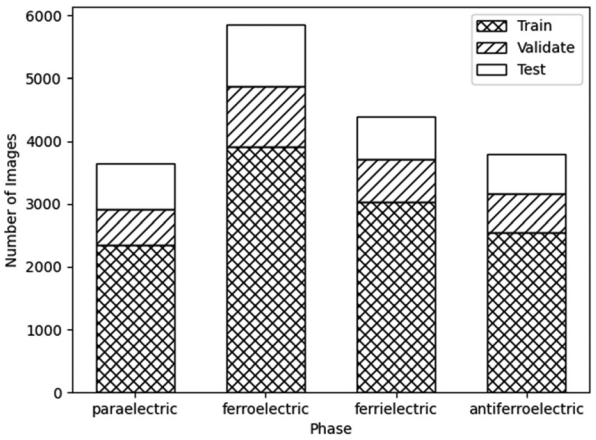


Fig. 8 The total number of images after augmentation of each LC phase, divided into training, validation and testing sets. Note that there are more images in the ferroelectric set, possibly resulting in a bias towards this phase when training the networks.

identified by several different experts. In this work we merely demonstrate the feasibility of such an idea.

4 Results

Before discussing the actual results we would like to make two preliminary comments. The first one is concerning the isotropic to SmA* transition. Like any other transitions involving the formation of the liquid crystalline state, this is, with the exception of well oriented homeotropic alignment which has been discussed in ref. 46, a simple classification task between black and bright. All models discussed in literature so far have had a classification accuracy very close to 100% for this case, so we will not discuss this in any further detail.

The second issue is that of the SmC_α* phase. The transition from SmA* to this phase did not exhibit any changes of structural features which could be observable in polarized microscopy. We suggest that this is due to the short pitch of the helical superstructure, which is far below the resolution limit of optical microscopy. As such, SmA* and SmC_α* are optically equivalent, which implies that in the absence of any differences in structural appearance, machine learning algorithms can in principle not detect the latter phase from textures alone. After several unsuccessful attempts with different sample preparations we were thus forced to ignore the presence of the SmC_α* phase, despite the fact that it has been demonstrated in literature, for example through polarization-analysed resonant X-ray scattering.[71,72] If one would want to distinguish between SmA* and SmC_α*, a completely different experimental methodology would need to be used, which allows a better resolution and therefore differentiation between the two phases. This would represent a very different study in itself and is thus outside the scope of this investigation.

4.1 Binary classifiers

4.1.1 Paraelectric SmA* to ferroelectric SmC* transition. The transition from the paraelectric to the ferroelectric phase can be observed as a change in the smoothness of the texture changing from the fan-shaped texture of the SmA* phase to the broken-fan texture of SmC*, due to the development of the tilt angle in the latter. The ferroelectric phase generally showed lines across the fans appearing as the temperature was decreased, which are absent in SmA* and caused by the formation of the helical superstructure in SmC*. Yet, these lines were not always visible, due to the onset of surface stabilization, thus their presence could not necessarily be used to distinguish the phase. The hyperparameters of the optimised models and their resulting accuracies are shown in Table 1, and confusion matrices of the results are shown in Fig. 9(a) and (b). The sequential model and inception models achieved test accuracies of (90 ± 1)% and (92 ± 2)% respectively, showing no significant difference between them. They also both required a similar number of parameters. Both models further showed a slight bias towards the ferroelectric phase, with (18 ± 5)% and (11 ± 3)% of paraelectric images being incorrectly classified as ferroelectric by each model, respectively. This is likely due to the ferroelectric SmC* dataset being larger than the paraelectric SmA*one. Slightly better accuracies might have been obtained when balancing the size of both sets.

Table 1 Architecture details, and validation and test accuracies for each model, optimised for the paraelectric/ferroelectric dataset (the numbers in brackets show the number of channels in each convolution layer, inception module or dense layer)

Paraelectric SmA* to ferroelectric SmC*	Sequential model	Inception model
Convolutional layers/inception modules	(8, 16, 32)	(1, 2)
Dense layers	(16, 8)	(8, 4)
Dropout rate	0.5	0.5
Trainable parameters	6682	7036
Validation accuracy	0.912 ± 0.003	0.934 ± 0.008
Test accuracy	0.90 ± 0.01	0.92 ± 0.02

4.1.2 Ferroelectric SmC* to ferrielectric $SmC_{1/3}$* transition. The ferroelectric SmC* to ferrielectric $SmC_{1/3}$* transition was most visibly observed as a change in the lines across the fans, as well as by a breakup of the focal conic defects as discussed

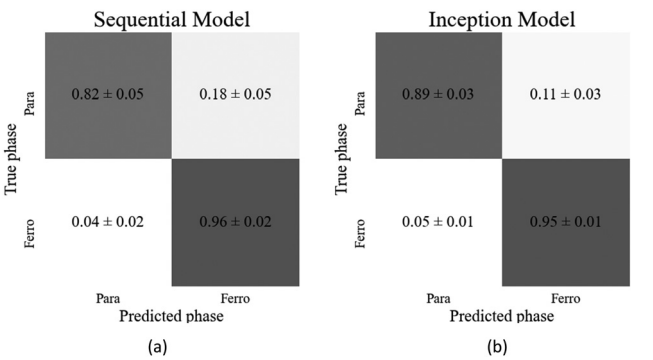


Fig. 9 Confusion matrices of the best performing (a) sequential and (b) inception models on the test datasets of the paraelectric (SmA*) to ferroelectric (SmC*) phase. The error shows the standard deviation over three training runs of the same model.

Table 2 Architecture details, and validation and test accuracies for each model, optimised for the ferroelectric/antiferroelectric dataset

Ferrielectric $SmC_{1/3}$* to antiferroelectric SmC_A*	Sequential model	Inception model
Convolutional layers/inception modules	(8, 16, 32)	(2, 2)
Dense layers	(16, 8)	(16, 8)
Dropout rate	0.6	0.5
Trainable parameters	6682	11 560
Validation accuracy	0.990 ± 0.003	0.996 ± 0.001
Test accuracy	0.986 ± 0.003	0.994 ± 0.002

above. The change in texture appeared less subtle than the paraelectric SmA* to ferroelectric SmC* transition, and both CNNs achieved higher accuracies as expected. The sequential and inception models resulted in test accuracies of (97.1 ± 0.4)% and (97.7 ± 0.3)% respectively, again showing no significant difference between the two models.

However, the inception model did require more parameters than for the previous binary classifier, using approximately 1.7 times the number of parameters to reach the same accuracy as the sequential model. Both models showed a slight bias towards the ferrielectric phase, despite the ferrielectric dataset being smaller. The optimised model architectures and their resulting accuracies are shown in Table 2, and the confusion matrices for each model are found in Fig. 10(a) and (b).

4.1.3 Ferrielectric $SmC_{1/3}$* to antiferroelectric SmC_A* transition. The ferrielectric $SmC_{1/3}$* to antiferroelectric SmC_A* transition displayed a visible change in texture, with the helix lines across the fans disappearing, and the texture of lines along the fans becoming more pronounced. Furthermore, the typical non-equidistant bars across the fans appeared in the SmC_A* phase.

The sequential and inception models correctly classified the test images with accuracies of (98.6 ± 0.3)% and (99.4 ± 0.2)%, respectively, thus achieving very high accuracies for this transition. As the confusion matrices (Fig. 11(a) and (b)) show, there was no particular bias towards either dataset. In this task the inception model achieved a (0.8 ± 0.4)% higher accuracy than the sequential model. However, this small increase in accuracy

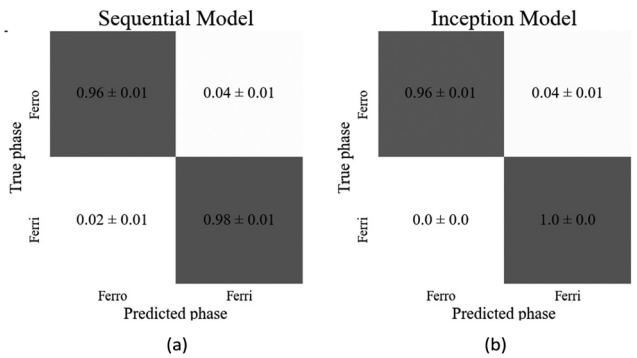


Fig. 10 Confusion matrices of the best performing models for the ferroelectric (SmC*) to ferrielectric ($SmC_{1/3}$*) phases for (a) the sequential and (b) the inception architecture.

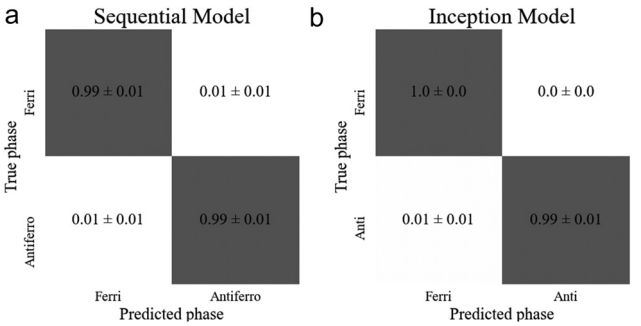


Fig. 11 Confusion matrices of the best performing models for the ferrielectric ($SmC_{1/3}$*) to antiferroelectric (SmC_A*) transition for the (a) sequential and (b) inception model.

Table 3 Architecture details, and validation and test accuracies for each model, optimised for the ferroelectric/antiferroelectric dataset

Ferroelectric SmC* to ferrielectric $SmC_{1/3}$*	Sequential model	Inception model
Convolutional layers/inception modules	(8, 16, 32)	(2, 2)
Dense layers	(16, 8)	(16, 8)
Dropout rate	0.6	0.5
Trainable parameters	6682	11 560
Validation accuracy	0.97 ± 0.01	0.97 ± 0.01
Test accuracy	0.971 ± 0.004	0.977 ± 0.003

required about 1.7 times as many parameters, so is at the expense of significantly increased computational cost. Table 3 displays the architecture details and resultant accuracies of each model.

4.2 Multiphase classifier

For the multiphase classification task, the inception model performed significantly better than the sequential model, resulting in test accuracies of (93 ± 1)% *versus* (84 ± 2)% respectively. The inception model also relied on comparatively fewer trainable parameters making it less computationally expensive. This demonstrates that the inception model is more suitable for this particular task and for more complex classification tasks in general, as highlighted in Fig. 12. The optimised model architectures and accuracies are shown in Table 4.

The confusion matrix for the sequential model (Fig. 13(a)) shows that the ferrielectric and antiferroelectric phases were only correctly classified in (84 ± 3)% and (87 ± 3)% of the cases, respectively. This is a significant decrease in comparison to the binary sequential classifier, which achieved (99 ± 1)% accuracy in classifying both of these phases. The multiphase inception model performed well in classifying these phases (see Fig. 13(b)), suggesting it may be better suited to handling multiphase classification.

5 Conclusions

It is demonstrated that supervised machine learning architectures like sequential convolutional neural networks CNNs and

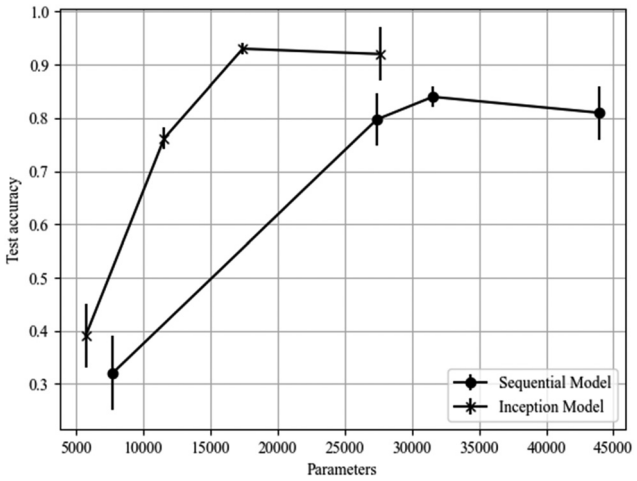


Fig. 12 Test accuracy against number of trainable parameters for each model optimised for the complete (four phase) dataset. As the number of parameters increases each model reaches a maximum accuracy before beginning to overfit. The advantage of the inception model is clearly shown; the model reaches a higher accuracy with fewer parameters. The lines joining data points are merely a guide to the eye.

Table 4 Architecture details, and validation and test accuracies for each model, optimised for the complete dataset

Multiphase	Sequential model	Inception model
Convolutional layers/inception modules	(8, 16, 32, 64)	(2, 2, 2)
Dense layers	(64, 32, 16, 8)	(16, 8)
Dropout rate	0.6	0.4
Trainable parameters	31 564	17 420
Validation accuracy	0.84 ± 0.01	0.925 ± 0.002
Test accuracy	0.84 ± 0.02	0.93 ± 0.01

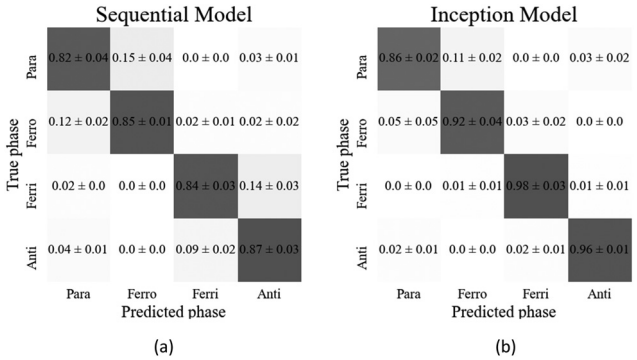


Fig. 13 Confusion matrices of the best performing models on the paraelectric (SmA*), ferroelectric (SmC*), ferrielectric ($SmC_{1/3}$*) and antiferroelectric (SmC_A*) test dataset for the (a) sequential and (b) inception convolution neural network architecture.

inception models can be employed to classify polar liquid crystals with phases such as the paraelectric SmA*, the ferroelectric SmC*, the ferrielectric $SmC_{1/3}$* and the antiferroelectric SmC_A* phase. This can be achieved *via* binary classifiers but also *via* multiphase classifiers with very good accuracies of the order of 90% and above.

In the binary classification tasks, both the sequential and inception models achieved similar accuracies, suggesting no particular advantage of one over the other architecture. In the multiphase classification task, the inception model performed significantly better, achieving a test accuracy of (93 ± 1)% in comparison to (84 ± 2)% achieved with the sequential model. This suggests that a more complex classification problem is better approached with an inception than a sequential model.

The investigation has further provided evidence that the characterisation of liquid crystals *via* texture observation can in the future be automated, provided that phases are studied which exhibit structural features that are different from those of other phases. Phases that appear identical cannot be identified individually from textures alone. Similarly, supervised machine learning as a texture characterization tool will also not be able to predict novel phases. Nevertheless, the presented results open a route to an additional characterization method, especially when sufficient training data is available, together with the needed computational infrastructure.

While the results show that CNNs are capable of classifying LC phases, a disadvantage of this method of classification is that it is difficult to extract information from the CNN on how the phases are distinguished. The CNN is a function approximating the relationship between input images and their labels, however this function involves thousands of parameters so cannot be used to give insight into any simple relationship between the input and the label. Analysis of the feature maps and kernels produced at each convolutional layer may give some information as to what features of an image the CNN uses, but there is a large number of these with complex interactions between them. As such, neural networks are often described as black box models, so the reasons behind the predictions they make cannot be understood. Although CNNs cannot directly be used to explain why a given phase displays a particular texture, by automating the task of classifying textures they can still contribute significantly to LC research. The ultimate vision of this and similar studies could be a sufficiently large database of all different liquid crystalline phases with varying boundary conditions, collected from a large range of different compounds and independently verified by a number of experts. This could then be used as training data, provided that sufficient computational power is available. With supervised machine learning algorithms of the CNN type with varying complexity, one could then automatically identify the large amounts of phase sequences of novel materials produced every year worldwide.

Author contributions

RB carried out the texture studies and the machine learning aspects of this study. ID conceived the investigation, supervised its implementation and contributed to the texture identification and writing of the manuscript.

Conflicts of interest

There are no conflicts to declare.

References

1 H. D. Block, *Rev. Mod. Phys.*, 1962, **34**, 123–135.
2 Y.-C. Wu and J.-W. Feng, *Wireless Pers. Commun.*, 2018, **102**, 1645–1656.
3 A. Isozaki, J. Harmon, Y. Zhou, S. Li, Y. Nakagawa, M. Hayashi, H. Mikami, C. Lei and K. Goda, *Lab Chip*, 2020, **20**, 3074–3090.
4 A. Maier, C. Syben, T. Lasser and C. Riess, *Z. Med. Phys.*, 2019, **29**, 86–101.
5 J.-G. Lee, S. Jun, Y.-W. Cho, H. Lee, G. B. Kim, J. B. Seo and N. Kim, *Korean J. Radiol.*, 2017, **18**, 570–584.
6 M. Shehab, L. Abualigah, Q. Shambour, M. A. Abu-Hashem, M. K. Y. Shambour, A. I. Alsalibi and A. H. Gandomi, *Comput. Biol. Med.*, 2022, **145**, 105458.
7 S. L. Goldenberg, G. Nir and S. E. Salcudean, *Nat. Rev. Urol.*, 2019, **16**, 391–403.
8 A. Hosny, C. Parmar, J. Quackenbush, L. H. Schwartz and H. J. W. L. Aerts, *Nat. Rev. Cancer*, 2018, **18**, 501–510.
9 D. C. Cires, A. Giusti, L. M. Gambardella and J. Schmidhuber, *Medical Image Computing and Computer-Assisted Intervention – MICCAI*, 2013, Springer, Berlin, pp. 411–418.
10 F. Cui, Y. Yue, Y. Zhang, Z. Zhang and H. S. Zhou, *ACS Sens.*, 2020, **5**, 3346–3364.
11 H. Chen, O. Engkvist, Y. Wang, M. Olivecrona and T. Blaschke, *Drug Discovery Today*, 2018, **23**, 1241–1250.
12 C. W. Coley, W. H. Green and K. F. Jensen, *Acc. Chem. Res.*, 2018, **51**, 1281–1289.
13 L. Zhang and S. Shao, *J. Appl. Phys.*, 2022, **132**, 100701.
14 G. Carleo, I. Cirac, K. Cranmer, L. Laudet, M. Schuld, N. Tishby, L. Vogt-Maranto and L. Zdeborova, *Rev. Mod. Phys.*, 2019, **91**, 045002.
15 S. So, T. Badloe, J. Noh, J. Bravo-Abad and J. Rho, *Nanophotonics*, 2020, **9**, 1041–1057.
16 N. M. Ball and R. J. Brunner, *Int. J. Mod. Phys. D*, 2010, **19**, 1049–1106.
17 S. Sen, S. Agarwal, P. Chakraborty and K. P. Singh, *Exp. Astron.*, 2022, **53**, 1–43.
18 S. K. Meher and G. Panda, *Eur. Phys. J.-Spec. Top.*, 2021, **230**, 2285–2317.
19 A. Radovic, M. Williams, D. Rousseau3, M. Kagan, D. Bonacorsi5, A. Himmel, A. Aurisano, K. Terao and T. Wongjirad, *Nature*, 2018, **560**, 41–48.
20 J. Carrasquilla and R. G. Melko, *Nat. Phys.*, 2017, **13**, 431–434.
21 P. S. Clegg, *Soft Matter*, 2021, **17**, 3991–4005.
22 A. L. Ferguson, *J. Phys.: Condens. Matter*, 2018, **30**, 043002.
23 T. Orlova, A. Piven, D. Darmoroz, T. Aliev, T. Mahmoud, T. Abdel Razik, A. Boitsev, N. Grafeeva and E. Skorb, *Digital Discovery*, 2023, **2**, 298–315.
24 H. Y. D. Sigaki, R. F. de Souza, R. T. de Souza, R. S. Zola and H. V. Ribeiro, *Phys. Rev. E*, 2019, **99**, 013311.
25 H. Y. D. Sigaki, E. K. Lenzi, R. S. Zola, M. Perc and H. V. Ribeiro, *Sci. Rep.*, 2020, **10**, 7664.
26 A. A. B. Pessa, R. S. Zola, M. Perc and H. V. Ribeiro, *Chaos, Solitons Fractals*, 2022, **154**, 111607.
27 C.-H. Chen, K. Tanaka and K. Funatsu, *Mol. Inf.*, 2019, **38**, 1800095.
28 T. Inokuchi, R. Okamoto and N. Arai, *Liq. Cryst.*, 2020, **47**, 438–448.
29 T. C. Le and N. Tran, *ACS Appl. Nano Mater.*, 2019, **2**, 1637–1647.
30 H. Doi, K. Z. Takahashi, K. Tagashira, J.-I. Fukuda and T. Aoyagi, *Sci. Rep.*, 2019, **9**, 16370.
31 P. Y. Taser, G. Onsal and O. Ugurlu, *Bull. Mater. Sci.*, 2023, **46**, 1.
32 A. Dhillon and G. K. Verma, *Prog. Artif. Intell.*, 2020, **9**, 85–112.
33 M. Walters, Q. Wei and J. Z. Y. Chen, *Phys. Rev. E*, 2019, **99**, 062701.
34 E. N. Minor, S. D. Howard, A. A. S. Green, M. A. Glaser, C. S. Park and N. A. Clark, *Soft Matter*, 2020, **16**, 1751–1759.
35 J. Colen, M. Han, R. Zhang, S. A. Redford, L. M. Lemma, L. Morgan, P. V. Ruijgrok, R. Adkins, Z. Bryant, Z. Dogic, M. L. Gardel, J. J. de Pablo and V. Vitelli, *Proc. Natl. Acad. Sci. U. S. A.*, 2021, **118**, e2016708118.
36 E. Hedlund, K. Hedlund, A. Green, R. Chowdhury, C. S. Park, J. E. Maclennan and N. A. Clark, *Phys. Fluids*, 2022, **34**, 103608.
37 K. Nayani, Y. Yang, H. Yu, P. Jani, M. Mavrikakis and N. Abbott, *Liq. Cryst. Today*, 2020, **29**, 24–35.
38 Y. Cao, H. Yu, N. L. Abbott and V. M. Zavala, *ACS Sens.*, 2018, **3**, 2237–2245.
39 X. Zhan, Y. Liu, K.-L. Yang and D. Luo, *Biosensors*, 2022, **12**, 577.
40 S. Jiang, J.-H. Noh, C. Park, A. D. Smith, N. L. Abbott and V. M. Zavala, *Analyst*, 2021, **146**, 1224–1233.
41 Y. Xu, A. M. Rather, S. Song, J.-C. Fang, R. L. Dupont, U. I. Kara, Y. Chang, J. A. Paulson, R. Qin, X. Bao and X. Wang, *Cell Rep. Phys. Sci.*, 2020, **1**, 100276.
42 E. Ramou, S. I. C. J. Palma and A. C. A. Roque, *ACS Appl. Mater. Interfaces*, 2022, **14**, 6261–6273.
43 N. Bao, S. Jiang, A. Smith, J. J. Schauer, M. Mavrikakis, R. C. Van Lehn, V. M. Zavala and N. L. Abbott, *ACS Sens.*, 2022, **7**, 2545–2555.
44 I. Dierking, J. Dominguez, J. Harbon and J. Heaton, *Liq. Cryst.*, 2023, DOI: **10.1080/02678292.2023.2221654**.
45 I. Dierking, J. Dominguez, J. Harbon and J. Heaton, *Liq. Cryst.*, 2022, DOI: **10.1080/02678292.2022.2150790**.
46 I. Dierking, J. Dominguez, J. Harbon and J. Heaton, *Front. Soft Matter*, 2023, **3**, 1114551.
47 R. B. Meyer, L. Liebert, L. Strzelecki and P. Keller, *J. Phys., Lett.*, 1975, **36**, 69–71.
48 N. A. Clark and S. T. Lagerwall, *Appl. Phys. Lett.*, 1980, **36**, 899–901.
49 J. W. Goodby, *Handbook of Visual Display Technology*, Springer, 2016, pp. 1911–1915, DOI: **10.1007/978-3-319-14346-0_82**.
50 A. D. L. Chandani, T. Hagiwara, Y.-I. Suzuki, Y. Ouchi, H. Takezoe and A. Fukuda, *Jpn. J. Appl. Phys.*, 1988, **27**, L729.
51 A. D. L. Chandani, E. Gorecka, Y. Ouchi, H. Takezoe and A. Fukuda, *Jpn. J. Appl. Phys.*, 1989, **28**, L1265.
52 A. Fukuda, Y. Takanishi, T. Isozaki, K. Ishikawa and H. Takezoe, *J. Mater. Chem.*, 1994, **4**, 997–1016.
53 D. Schlauf, Ch Bahr and H. T. Nguyen, *Phys. Rev. E: Stat., Nonlinear, Soft Matter Phys.*, 1999, **60**, 6816–6925.

54 H. T. Nguyen, J. C. Rouillon, P. Cluzeau, G. Sigaud, C. Destrade and N. Isaert, *Liq. Cryst.*, 1994, **17**, 571.
55 H. Takezoe, J. Lee, Y. Ouchi and A. Fukuda, *Mol. Cryst. Liq. Cryst.*, 1991, **202**, 85–90.
56 J. P. F. Lagerwall, D. D. Parghi, D. Kruerke, F. Gouda and P. Jagemalm, *Liq. Cryst.*, 2002, **29**, 163–178.
57 M. P. Deisenroth, *Mathematics for machine learning*, Cambridge University Press, Cambridge, 2020.
58 R. Shanmugamani, *Deep learning for computer vision: expert techniques to train advanced neural networks using TensorFlow and Keras*, Paths International Ltd, Birmingham, 2018.
59 M. Kline and L. Berardi, *Neural. Comput. Appl.*, 2005, **14**, 310–318.
60 K. O'Shea and R. Nash, *arXiv*, 2015, preprint, arXiv:1511.08458v2 [cs.NE], DOI: **10.48550/arXiv.1511.08458**.
61 S. Wager, S. Wang and P. S. Liang, *Advances in Neural Information Processing Systems*, 2013, vol. 26.
62 P. Murugan and S. Durairaj, *arXiv*, 2017, preprint, arXiv: 1712.04711v1 [cs.CV], DOI: **10.48550/arXiv.1712.04711**.
63 S. Ioffe and C. Szegedy, *Proceedings of the 32nd International Conference on Machine Learning*, PMLR, 2015, **37**, pp. 448–456.
64 P. Luo, X. Wang, W. Shao and Z. Peng, *arXiv*, 2018, preprint, arXiv:1809.00846v4 [cs.LG], DOI: **10.48550/arXiv.1809.00846**.
65 D. P. Kingma and J. B. Adam, *arXiv*, 2014, preprint, arXiv:1412.6980v9 [cs.LG], DOI: **10.48550/arXiv.1412.6980**.
66 Google Colaboratory. Available from: **https://colab.research.google.com/**.
67 Keras API. Available from: **https://keras.io/**.
68 TensorFlow APl. Available from: **https://www.tensorflow.org/**.
69 C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, 2015 IEEE \Conference on Computer Vision and Pattern Recognition (CVPR), 2014, **https://openaccess.thecvf.com/content_cvpr_2015/papers/Szegedy_Going_Deeper_With_2015_CVPR_paper.pdf**.
70 VideoLan, Vlc media player, 2006. Available from: **https://www.videolan.org/**.
71 P. Mach, R. Pindak, A.-M. Levelut, P. Barois, H. T. Nguyen, C. C. Huang and L. Furenlid, *Phys. Rev. Lett.*, 1998, **81**, 1015–1018.
72 P. Mach, R. Pindak, A.-M. Levelut, P. Barois, H. T. Nguyen, H. Baltes, M. Hird, K. Toyne, A. Seed, J. W. Goodby, C. C. Huang and L. Furenlid, *Phys. Rev. E: Stat., Nonlinear, Soft Matter Phys.*, 1999, **60**, 6793–6802.