Classification of liquid crystal phases with convolutional neural networks

James Harbon

10137627

Department of Physics and Astronomy

University of Manchester

MPhys Project Report

Project performed in collaboration with Joshua Heaton and supervised by Ingo Dierking

May 2021

Abstract

Convolutional neural networks and deep learning methods were used to classify images of liquid crystal textures as specific phases. The dataset we used was built only from images produced by polarised microscopy, which was in contrast to the use of simulated textures in previous literature. The images included the textures of the cholesteric phase and four unique smectic phases. Five subsets of this dataset and three network architectures, which varied in their capacity and types of layers, were created to explore five tasks. The first three were binary classification tasks and the final two were multinomial. The mean test accuracy values varied from 85-99%, and the standard deviation of those values ranged from $\pm$1-6%. Furthermore, confusion matrices were used to identify specific weaknesses or strengths in the classification of a given phase, or class, for each task.

The results we obtained provided evidence that convolutional neural network variants could be a viable approach for the classification of liquid crystal phases. However, we found that the highest test accuracy values for the multinomial tasks, which were both below 90%, had potentially saturated. Additionally, the networks trained for the binary task with the smallest dataset were observed to produce lower test accuracy values and a higher variance in comparison to those trained for the other binary tasks. Consequently, we believe that future work should focus on expanding the size and class balance of the overall dataset to improve the performance of networks. This could be achieved by the creation of a large-scale and open-source database of labelled liquid crystal texture images, which could be contributed to by members of the liquid crystal research community.

Contents

1 Introduction

Artificial intelligence (AI) is a field concerned with understanding and building intelligent agents. Due to the present scope of the field, current approaches are focused on building agents which interact with specific or narrowly defined environments. Examples include: machine translation between various languages, autonomous vehicles, recommendation systems and natural language processing. The area of computer vision is of particular interest to us, and can provide methods for agents to process, analyse and understand data from digital images. The engineering applications have found success in the automation of tasks typically suited for the human visual system, such as the identification of tumours in medical images and faults in manufacturing processes. Furthermore, computer vision methods can be used in tandem with fields such as reinforcement learning to allow agents to reason with visual data and plan ahead [1, p. 1-59].

Computer vision emerged as a discipline in the 1960s, and was viewed by AI researchers as a stepping stone for artificial agents to achieve their goals. Research in the 1970s laid the foundations for modern low-level algorithms such as edge detection and the detection of spatial periodicity in patches of multiple pixels with texture analysis. Subsequently, a higher level operation named image segmentation, which could identify groups of similar pixels as specific objects, was developed. These low-level approaches found success in problems with a narrow scope, however they often struggled with broader environments. This was mainly because the patterns observed in images were interpreted via predetermined rules, which generally can't cover all of the possible object scenarios in images due to changes in orientation, brightness and obscured parts of an object [1, p. 928-965].

Machine learning (ML) is a sub-field of AI and encompasses a broad range of algorithms and methods, which can be used to learn from data and hence improve the performance of an agent for a given task. The use of large training datasets and also models based on variants of artificial neural networks loosely defines the practice of deep learning. Data which has a grid format, such as the values of pixel intensity in images, is generally handled by convolutional neural networks (CNNs). In 1989, Yann LeCun *et al.* first applied a CNN to the recognition of handwritten ZIP codes on mail. Although the algorithm worked, the network required three days to train on the data. Recent advancements in GPU speed and memory, parallel computation and optimisation algorithms have endowed CNNs, and all other variants of neural networks, with drastically lower training times on large datasets [2, 3]. The feasibility of CNNs in commercial applications was given merit in 2012 when Krizhevsky *et al.* won the large-scale ImageNet competition by a significant margin. The competition is focused on the classification of objects which could belong to any one of a thousand classes or more, where the training dataset is typically composed of millions of images. ImageNet can thus provide a useful benchmark for computer vision models [4]. Deep learning methods have also been utilised in the sciences. For example, a meta-analysis of 79 articles was centred on the diagnosis and prognosis of cancer via images and computational models. The analysis concluded that deep learning approaches can improve the accuracy of predictions by 15-25% relative to conventional approaches [5].

Liquid crystals can exist in various phases, where the order of a given phase can range from the isotropy of liquid molecules all the way to the periodic structures of crystals. The phase exhibited by a thermotropic liquid crystal varies with the temperature of the compound, and the phases are typically identified by human observation and inference. The aim of this project is to build models which can learn to correctly classify the phase in a liquid crystal texture image. The literature for this topic is limited and networks have only been trained and tested on simulated textures, which won't necessarily produce successful generalisation to real world data [6–9]. In contrast to this approach, we built a dataset using only experimental data so as to test our models with greater rigour.

2 Background Theory

In this section an overview of liquid crystal and machine learning theory is provided. Supplementary information can be found in the report from the previous semester [10].

2.1 Liquid Crystals

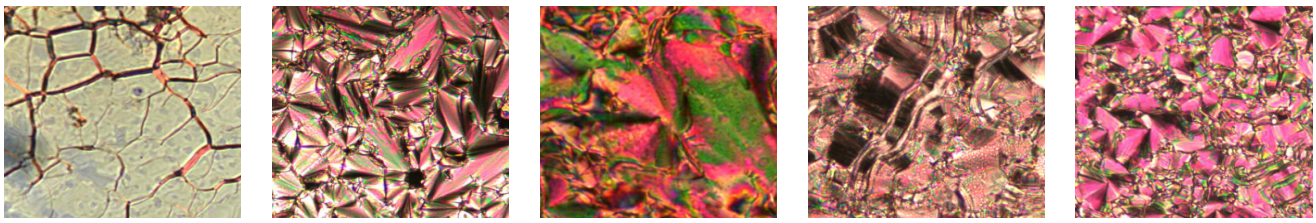
Liquid crystals (LCs) are anistropic compounds which contain partially ordered molecules. The thermodynamic behaviour of such compounds generally lies between the solid state crystalline and isotropic liquid phases. LCs undergo phase transitions in response to changes in their environment. Most LC compounds fall into the categorisation of either thermotropic or lyotropic. The former contains LCs which will undergo phase transitions in response to changes in temperature, and can be further distinguished by the calamitic, or rod-like, shape of its constituent molecules. The latter differs in that the phases form only in the presence of a suitable solvent and the type of phase exhibited depends not only on temperature, but also on the concentration of the LC in the solvent. For this project, we focused only on thermotropic LCs [11, p. 1-7].

The given phase exhibited by an LC can be characterised by the type of order it possesses. Due to the rod-like shape of the molecules, we can define a long molecular axis and a local director vector $\hat{\boldsymbol{n}}$, which is the average of the directions of the long molecular axes in some volume element. A smaller deviation of $\hat{\boldsymbol{n}}$ throughout the volume occupied by the molecules implies a greater degree of orientational order. In addition to this, we can also define the positional order. The centres of mass of the molecules will generally arrange into layers as the positional order increases, and the effects of further increases in order such as a hexagonal structure in the plane of the layers can be observed. Both the orientational and positional order can each be defined as either short or long range, which depends on how much the ordered structure deviates from an arbitrary position in the compound. Furthermore, both types of order generally increase as the temperature of the compound is reduced [11, p. 1-16].

A sequence of LC phases can be generated via either heating or cooling a given compound. At a high enough temperature the isotropic phase, which is defined by a random distribution of the centres of mass and directors of the molecules, usually dominates. Upon cooling, the nematic phase can be observed. This phase only exhibits a long range orientational order. For compounds with the property of chirality, which is defined by a lack of mirror symmetry, a counterpart named the cholesteric phase exists. The only distinction is that this phase forms a helical structure, where the director $\hat{\boldsymbol{n}}$ rotates around the helix axis. After further cooling, groups of smectic phases can be observed. The first of these is the fluid smectic group, where molecules are arranged into layers. There is no positional order within the layers, and the phases in this group exhibit an orientational order similar in magnitude to that in the nematic phase. The two phases within this group are named smectic A and C, and their only distinction is a rotation of the director $\hat{\boldsymbol{n}}$ with respect to the normal axis of the layers. Next, the hexatic group can be observed. This group differs from the fluid group in its greater positional order, which is exhibited in the form of a hexagonal structure within each layer. Three phases exist in this group: smectic B, where the director is parallel to the layer normal axis; smectic I, where the director points towards the apex of the hexagon; and smectic F, where the director points towards the side of the hexagon. Finally, a sequence of soft crystal phases, which have even greater positional and orientational order than the aforementioned phases, can be observed. The phases described above will not necessarily all be observed for a given compound, however the sequence generated will follow the same order [11, p. 7-16].

Researchers studying LC compounds often need to determine the phase transition temperatures and the specific phases exhibited. This is commonly carried out via an experimental setup named polarised microscopy. In this arrangement, unpolarised light is directed through two polarisers oriented

at 90° to each other. A sample of an LC compound is placed between the polarisers, and the resulting image will be dark unless the sample possesses optical properties which can rotate the light's plane of polarisation. For phases other than the isotropic state, there is an alignment of the director, or optic axis in this context, to varying degrees and hence these phases are birefringent. The difference in optical axis directions and the degree of positional order between different phases gives rise to distinct textures which can be observed. However, the distinctions between certain phases can be subtle [11, p. 33-39].



(a) Cholesteric (b) Smectic A (c) Smectic C (d) Smectic I (e) Smectic F

Figure 1: Images of the textures for five unique phases in our dataset. The texture for a given phase often varies slightly for different LC compounds.

2.2 Machine Learning and Neural Networks

The broad range of problems in ML can be specified by defining a task environment. This clarifies how information in the environment is communicated to an agent, the actions an agent can take to interact with the environment and how the performance of the agent is measured. Supervised learning is one of the most commonly used paradigms. In this framework an agent is exposed to a dataset composed of examples $(\boldsymbol{X}, \boldsymbol{y})$, where $\boldsymbol{X}$ is an input vector the agent accepts from the environment and $\boldsymbol{y}$ is the true output vector. A successful agent will be able to create a model which can accept some unseen input and correctly predict the output. The agent can learn to improve the accuracy of this model by training on data. Furthermore, the dataset needs to be partitioned into training and test sets, such that the agent can train on the former set and the latter set can provide a source of novel data which can test how successfully the model can generalise to unseen data. In contrast to this framework, unsupervised learning is concerned only with a dataset of vectors $\boldsymbol{X}$, where the task is to find groups or clusters of similar vectors based on the information in each. This project is focused exclusively on the supervised learning approach [1, p. 693-697].

For a set composed of N input-output pairs, $\{(\boldsymbol{X_1}, \boldsymbol{y_1}), ...(\boldsymbol{X_N}, \boldsymbol{y_N})\}$, we assume that we can represent the relationship with a function $\boldsymbol{y} = f(\boldsymbol{X})$. The output $\boldsymbol{y}$ can belong to a discrete set of values, which defines a classification task, or the output can take continuous values in a regression task. The fundamental problem in supervised machine learning is the search for a function $\hat{\boldsymbol{y}} = h(\boldsymbol{X}; \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is a set of parameters, which can approximate the true function f. The process of finding the optimal function is effectively a search through some function space, which contains families of parametric functions, to find a function which can minimise or maximise a given performance metric. Assumptions about this space will define the families of functions it contains and also loosely quantify the bias and variance of these functions. The bias of the functions increases as more assumptions are built-in about the underlying relationship between input and output. Bias error occurs when the assumptions cause a model to miss the relevant relations between the inputs and outputs in the training set, which is defined as underfitting. Greater variance of the functions is related to more relaxed assumptions about the underlying relationship and also the number of parameters the functions have. High variance can cause significant sensitivity to small changes in the training set and cause an agent to model random

noise. This effect is known as overfitting and can cause poor generalisation of a model to novel data. The bias-variance tradeoff is the search for a function which can simultaneously minimise the aforementioned errors. Generally, it is impracticable to find a specific family of functions which will minimise both types of error for any supervised learning task. In practice the error due to overfitting usually dominates, and hence regularisation methods must be used to select models which produce the smallest test set error [1, p. 693-717].

Neural networks define a broad family of algorithms and are built from layers of nodes, or units, connected by weighted links. A given link propagates the activation σ_i from unit i to a unit j in the next layer, and the associated weight ω_{ij} determines the strength and sign of the connection. Additionally, each unit j has a parameter ω_{0j} called the bias. We are interested in feed-forward networks, where the information can only be propagated in one direction, from units in one layer to those in the next. For these networks the activation of a unit j, which belongs to the set of all units U, is

$$\sigma_j = g(\sum_{i \in P} \omega_{ij}\sigma_i) = g(\Omega_j) \tag{1}$$

where $P \in U$ is the set of units in the layer preceding that which contains unit j and g is called an activation function, which is usually a logistic or hard-threshold function. Due to the non-linearity of g, the network of units can collectively represent a non-linear, parametric function $h(\boldsymbol{X}; \boldsymbol{\omega})$, where $\boldsymbol{X}$ is a vector of input units and $\boldsymbol{\omega}$ is the set of all weights in the network. Data in the input layer is transformed by a hidden layer, which is an intermediate layer between input and output, and then propagated to either the next hidden layer or the output layer. For regression tasks, the output layer often contains only one unit which accepts a continuous value. Classification tasks require at least one unit in the output layer for binary classification, and for multinomial output tasks the number of units is equal to the number of discrete values the output can accept. Provided that the output values have been normalised by an appropriate activation function, the value for each unit represents the confidence, or predicted probability, that the output for a given input vector belongs to the corresponding class for that unit. [1, p. 727-737].

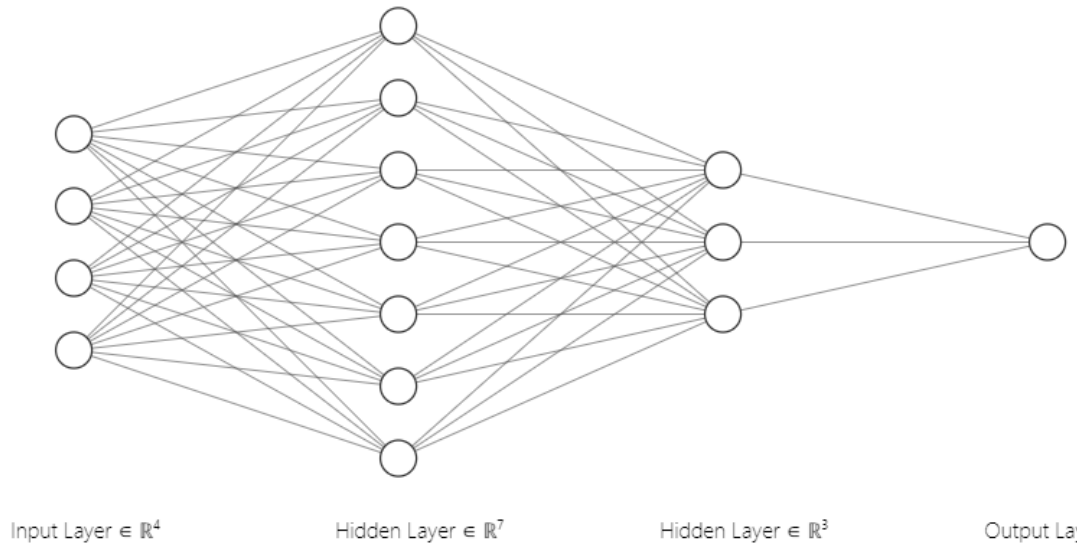


Figure 2: A diagram of a fully-connected neural network with four input units, two hidden layers and one output unit. The lines between units represent the corresponding weighted connections, and information flows from left to right in this diagram. Adapted from [12].

The assumption that there exists a true, deterministic mapping $f(\boldsymbol{X})$ between input and output relies on the absence or negligence of noise in data. Real world data can often be noisy and hence the more fundamental perspective is that we are trying to approximate the conditional probability distribution of the dataset, $P(\boldsymbol{y}|\boldsymbol{X})$, with a parametric model $P(\boldsymbol{y}|\boldsymbol{X};\boldsymbol{\theta})$. However for data with relatively small noise, the existence of $f(\boldsymbol{X})$ can be assumed since it maps an input to the expectation of a narrow distribution. Variation of the parameters $\boldsymbol{\theta}$ can alter the similarity between the true and approximate distributions, and this similarity can be quantified with the Kullback-Leibler (KL) divergence. Maximum likelihood estimation can then be used to find parameters which minimise this divergence. This method leads to the derivation of a number of different loss functions, $L(\boldsymbol{X}, \boldsymbol{y}; \boldsymbol{\theta})$, where the minimisation or maximisation of such a function corresponds to minimisation of the KL divergence. The loss is used during training as a guide for how the parameters should be updated and it can also quantify the confidence of predictions on training and test data. Regression and classification tasks respectively have their own loss functions due to differences in quantifying discrete and continuous output errors [12, p. 129-133].

Closed-form solutions for optimal parameters do not exist in most cases. We often assume that the loss function has a global minimum and the solution is then reduced to a convex optimisation problem. Gradient descent is one of the most commonly used optimisation algorithms in ML, and involves the computation of the partial derivatives $\nabla_{\boldsymbol{\theta}} L(\boldsymbol{X}, \boldsymbol{y}; \boldsymbol{\theta})$. We can randomly initialise the parameter values by drawing them from a normal distribution, and then the parameters are updated as

$$\theta_i \longleftarrow \theta_i + \Delta\theta_i = \theta_i - \eta \frac{\partial L}{\partial \theta_i}, \tag{2}$$

where the learning rate, η, determines how much we vary the parameters. This iterative method aims to follow the "down-slope" path in the space of L, and will generally cause oscillation around the global minimum in the long term. For larger datasets, the computational time required to execute gradient descent and update parameters signficantly increases. In these cases variations of the method are used, such as stochastic gradient descent (SGD), where the loss function is calculated with a randomly selected $(\boldsymbol{X}, \boldsymbol{y})$ pair for each update, and mini-batch descent (MBD), where a relatively small batch of randomly selected pairs are used for each update. Despite the lower computational time of SGD, it can be prone to production of unstable models and significant variance in accuracy. MBD is hence a suitable tradeoff between the speed of SGD and precision of standard gradient descent [12, p. 80-95]. Neural networks use MBD in tandem with another algorithm called backpropagation. The activation values of units in neural networks depend on the output from units in previous layers, which means that the full analytical expression for computation of partial derivatives contains a lot of repetition. Backpropagation utilises the chain rule to compute partial derivatives layer by layer, starting from the output layer and finishing at the input. This reduces repetition in gradient computations, and hence improves the efficiency of the process and reduces training time. For a loss function L and matrix of weights ω_{ij}, where an element is the weight for the connection between a unit i and a unit j in the next layer, we can use the chain rule and recursive expressions to arrive at

$$\frac{\partial L}{\partial \omega_{ij}} = \sigma_i \delta_j \tag{3}$$

and

$$\delta_j = \frac{\partial L}{\partial \sigma_j}\frac{\partial \sigma_j}{\partial \Omega_j} = \begin{cases} \frac{\partial L}{\partial \sigma_j}\frac{d\sigma_j}{d\Omega_j} & \text{if unit } j \in \text{output layer} \\ (\sum_{n\in N} \omega_{jn}\delta_n)\frac{d\sigma_j}{d\Omega_j} & \text{if unit } j \notin \text{output layer} \end{cases} \tag{4}$$

where N is the set of units in the layer after that which contains unit i. The update for the weights,

$$\omega_{ij} \longleftarrow \omega_{ij} + \Delta\omega_{ij} = \omega_{ij} - \eta\sigma_i\delta_j, \tag{5}$$

is repeated for each batch of data over the dataset during what is called an epoch. Neural networks are often trained over multiple epochs to improve the performance of the model. This is because the networks utilise non-linear functions and a multitude of parameters, which produces multiple local minima in the space of the loss function. The training drives the loss function as low as possible over a predetermined number of epochs [12, p. 175-219].

Finally, a subset of parameters named the hyperparameters define a network and how it trains. Examples include the number of layers, the number of units in each layer, the learning rate η in equation 5, the number of training epochs and the batch size for MBD. Most of the hyperparameters must be fixed before the training begins, and their variation can change the performance of the overall model. Methods such as grid search and specialised hyperparameter tuning algorithms aim to find an optimal configuration [12, p. 200-219].

2.3 Convolutional Neural Networks

Data formats such as images can be represented by a 2-D grid of values, such as in a matrix. Grayscale images contain just one channel and hence can be represented as a 2-D tensor. RGB images contain three channels and must be represented by a 3-D tensor. These tensors can in theory be flattened into a vector input of units and connected to a dense layer, which is the type of layer we described in section 2.2. However, this would create an impracticably high number of parameters between the input layer and first hidden (dense) layer, and thus significantly increase the computational cost. Furthermore, a standard artificial neural network will not necessarily generalise to novel data with success and could be prone to learning noise in images. One type of layer called the convolutional layer, which employs the convolution operation on a tensor and what we call a kernel, has been used to extract meaningful features from images with great success. Convolutional neural networks (CNNs) are defined as containing one or more convolutional layers, with the possibility for a number of other layers to precede or succeed those layers. Below, we give a brief explanation of the mechanism and also a type of operation called a pooling layer, which typically follows the convolutional layer in CNNs [12, p. 326-330].

The key part of a convolutional layer is the kernel, which is convolved with some 2-D tensor input to produce what is generally referred to as a feature map. Multiple kernels can be convolved with a given tensor input, which means that the output tensor generally has multiple channels. The design of the kernel provides the layer with some interesting properties. One of these is named sparse interactions where, rather than representing every pixel input value by an input unit and each interacting with every unit in the following hidden layer, the elements of the feature map are connected to patches of four, nine or more pixels. Each element in a feature map is connected to a patch of pixels with dimensions less than or equal to that of the kernel. The trainable parameters here are the elements of the kernel, which are trained via the convolution of the kernel with the matrix over all of the patches. This leads us to the property of parameter sharing, since a given kernel will have one set of parameters that are being trained. Hence, every element of the feature map has a connection, to its respective patch, that is weighted by the same set of parameters. This further reduces the computational cost of a convolutional layer in comparison to a dense layer. We now consider a general 3-D tensor input $\mathbf{I}$, where element I_{ijp} specifies the value of an input unit in channel i at row j and column p. This input can be convolved with a 4-D kernel tensor $\mathbf{K}$ where an element K_{ijpq} specifies the weight of a connection between a unit in channel i of the output and a unit in channel j of the input, with an offset of p rows and q columns between the input and output units. The overall operation can be expressed as

$$F_{ijp} = \sum_{q,m,n} I_{q,j+m-1,p+n-1} K_{i,q,m,n} \tag{6}$$

where F_{ijp} is an element of the 3-D feature map tensor $\mathbf{F}$ and the summation over q, m and n is over all the values for which the indexing operations on $\mathbf{I}$ are valid. The operation described here is carried out under the assumption that the kernel can move from one patch of pixels to another in a step no greater than one row and one column. In some cases we would like to learn features from a smaller number of patches to reduce the overall computational cost, and we can accomplish this in part by changing the stride, which is the maximum row or column index step size. We can adjust the overall operation with

$$F_{ijp} = C(\mathbf{K}, \mathbf{I}, s) = \sum_{q,m,n} I_{q,(j-1)\times s+m,(p-1)\times s+n} K_{i,q,m,n} \tag{7}$$

where C is a function which represents the downsampled convolution of $\mathbf{K}$ and $\mathbf{I}$ and s is the stride size. Additionally, the height and width of the kernels can be altered to further reduce the computational cost [12, p. 330-343].

The elements in the feature map are often transformed by an extra operation called the pooling layer. This type of layer replaces each output of the feature maps with a summary statistic of neighbouring values. The max pooling operation, where an output in a feature map is replaced by the maximum value in a rectangular neighbourhood of values is one example [13]. Other examples include the average value in that neighbourhood and also the weighted average as a function of distance from the central value. No matter which statistic we use, pooling helps to make the feature maps approximately invariant to small translations of the input, meaning that most of the values in the feature maps will not change after the translation. This can be of great importance for example when we care more about the existence of a feature in an image rather than its precise location [12, p. 335-340].

3 Architectures

Three different CNN architectures were created at the start of the project, and variations were created for two of the architectures. The capacity of a network loosely refers to the number of parameters in the architecture. Most of the parameters can be adjusted during training and hence the capacity of a network can affect the success in generalisation to novel data. The three architectures we created had significant differences between their numbers of parameters: the sequential network, where the variants had from 30,000 to 1,880,000 trainable parameters; Inception, where the variants had from 30,000 to 540,000 trainable parameters; and ResNet50, which had 25,580,000 trainable parameters. The latter architecture was used as a static, high capacity architecture. This was because the training time for this network, relative to the other two architectures, was significantly larger and hence made training multiple variants infeasible with our time constraints.

3.1 Sequential Networks

The first of the architectures we created was a sequential network. This was a sequence of convolutional, pooling, batch normalisation and dense layers with an absence of branching layers or skip connections. The main building block we used to construct an entire sequential network consisted of three layers. The first was a Conv2D layer with a variable number of kernels, kernel size of 3×3, stride size of 1, rectified linear unit (ReLU) activation function and "same" padding. Furthermore, the first Conv2D layer in the network was configured to accept input tensors of shape (256, 256, 1), as these were the dimensions

we intended to use for our training images. The second layer, which followed the Conv2D, was a batch normalisation layer. This type of layer can constrain the outputs in the feature maps to a specific range, which has been shown to improve regularisation of the network. Following the batch normalisation, a layer named MaxPooling2D was used. Pooling layers, as discussed in section 2.3, are commonly used to create spatial invariance of features over small translations of the input. The characteristic features in an LC texture will not necessarily be in the same location in every image, which thus justified the use of a pooling layer. Multiple building blocks were added together in a sequence to form a deeper network, where the number of kernels in every Conv2D layer except the first was double the amount in the previous Conv2D layer. Due to this dependence on the number of kernels in the first layer, we were able to vary the capacity of the entire network by changing the initial number of kernels. The naming convention we used for a sequence of the blocks was "n Layers", where n refers to the number of blocks used.

After the transformation of the input tensor by all of the convolutional and pooling layers, a GlobalAveragePooling layer was used to create further pooling and also connect each feature map to a single value, which created a 1-D tensor of values. This tensor was then transformed by a dense layer with 256 units, and then a second dense layer with 128 units. Both layers used the ReLU activation function and each was followed by a dropout layer with a rate of 0.5. The dropout randomly selects only a subset of the unit outputs in one layer to use in the next layer, which can reduce overfitting and hence create further regularisation of the network. The output layer of the network was a dense layer with a number of units equal to the number of classes in a given classification task. The SoftMax activation function was used for the output layer because it normalised the output vector of values to unity, which hence represented each unit value as a predicted probability, or confidence, for its respective class.

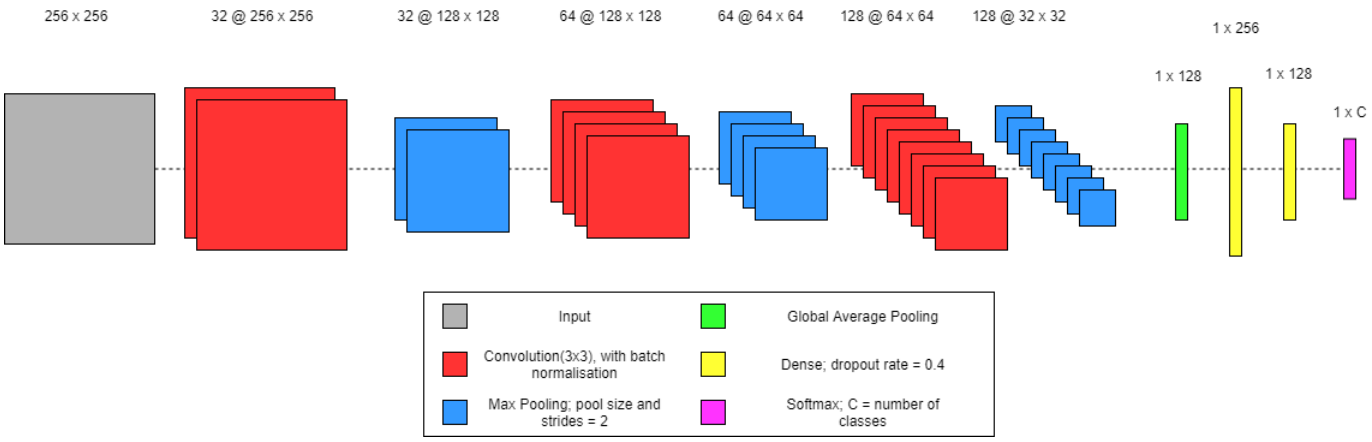


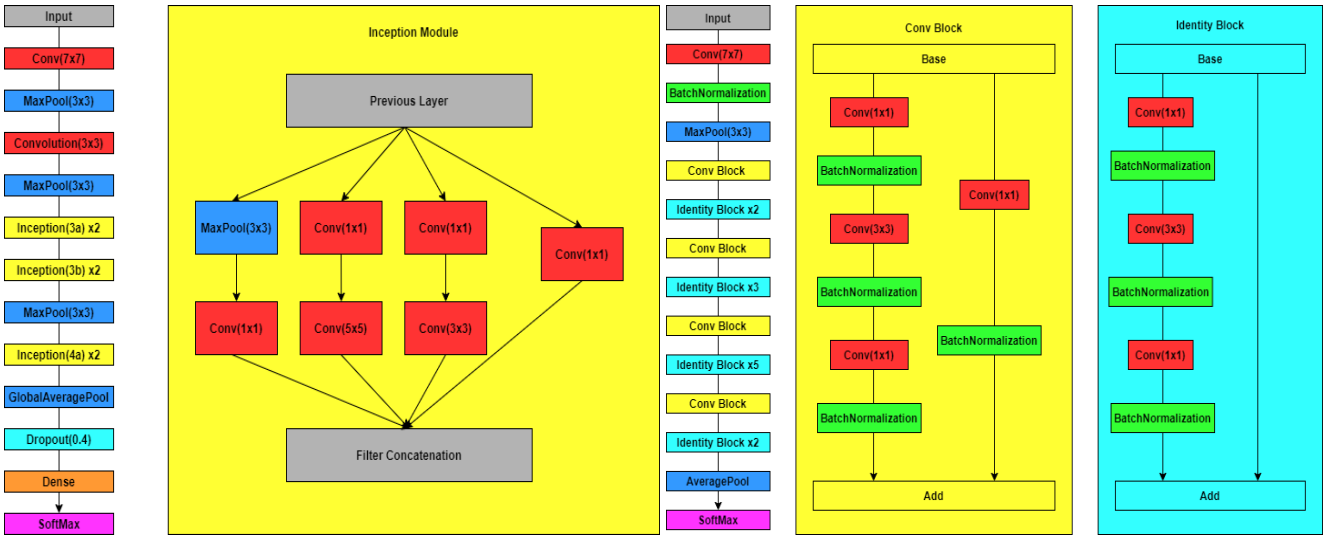
Figure 3: Illustration of a sequential architecture, where three building blocks were used to create a variant named "3 Layers". Figure taken from previous work [10].

3.2 Inception and ResNet50

The second architecture we created is a variant of the 22 layer deep GoogleLeNet, codenamed "Inception", which won the 2014 ImageNet Large Scale Visual Recognition Challenge (ILSVRC). The ILSVRC is a benchmark image classification competition which in 2014 consisted of hundreds of object classes and millions of images [14]. The key element of the architecture is the "Inception module", which has a specific configuration of convolutional layers and forms branches for information to flow through. Different module types can be defined based on the number of kernels, also called filters, in each convolutional layer. The module ends with the concatenation of the kernels from the branches. The use of these

modules has found great success in computer vision tasks due to the increased depth and width they provide for a CNN, at a reduced computational cost relative to a sequential structure of layers. We decided to reduce the capacity of the Inception V3 architecture variant by using no more than the first four module types. Figure 4(a) shows a variant with only the first three module types and an illustration of the structure of a given module. Full details of the V3 architecture can be found in "Going Deeper with Convolutions" [15].

The classification of images which contain complex, high-level features generally requires significantly deep CNNs. In theory it is expected that the addition of more layers will only improve the generalisation of a network to novel data, however it has been observed that the accuracy, in a classification task for example, saturates and can even decrease for sequential networks with many layers. This effect is often attributable to the vanishing gradient problem. The origin of this problem lies in the backpropagation algorithm, where repeated multiplication of the gradients in equation 4 via the chain rule can cause gradient size to gradually diminish layer by layer. If the gradients at the first hidden layers are too small, then learning can effectively slow down or cease since the weights in those layers experience little change. One solution to this problem is the use of residual networks, which are also called "ResNets". These types of networks employ skip connections, which allow information to flow through alternate paths that skip one or more layers. This in turn allows the gradient information to reach the layers close to the input during backpropagation. We implemented a specific residual network architecture named ResNet50, which is a 50-layer residual network that belongs to an ensemble of six different ResNet architectures that won the 2015 ILSVRC competition. The skip connections are implemented as "identity blocks" and they form stages with the "convolutional blocks". Figure 4(b) shows the structure of these blocks and the overall sequence of layers in ResNet50. Specific details of the architecture can be found in "Deep Residual Learning for Image Recognition" [16].



(a) Inception (b) ResNet50

Figure 4: The figure in (a) illustrates the inception module and a variant of the V3 architecture which only uses the first three module types. The full ResNet50 architecture is shown in (b). Both figures are from the previous report [10]

4 Methodology

4.1 Data Collection and Preparation

The dataset that was used to train the networks had to be created from the ground up due to the lack of a labelled liquid crystal phases dataset online. All of the data we collected and used was provided by our supervisor Ingo Dierking. The data consisted of videos, obtained via polarised microscopy and a digital camera, which showed an LC compound transitioning between phases due to either being heated or cooled. Each video was labelled with a temperature range and an ordered sequence of phases observed. VLC Player's scene filter function was used to extract images from the videos.

Training, validation and test sets were created with a 70:15:15 split, where the validation set was used to select an optimal model before the test set was used to measure success of generalisation. The images needed to be sorted based on the LC phase, which was achieved via the sequence of phases in the video label and identification of transitions in the video. For unknown phases we referred to a paper co-authored by our supervisor which contained the temperature ranges of various phases for the LC compounds that were observed [17]. The temperature in the video increased or decreased at a constant rate and hence the temperature ranges in the paper assisted the identification of phases. Additionally, we ensured that images of a given phase from a given video were not shared across any of the three sets we created. We implemented this condition via an alphabetical naming convention, which allowed us to distinguish between images from different videos. This distinction was necessary to reduce data leakage between the training and either of the validation and test sets. If unaccounted for, data leakage can inflate validation and test accuracy values, which can thus create overly optimistic results and invalidate the models [18].

The videos and extracted images had a resolution of 2048×1088, which are unnecessarily large dimensions for training images in most cases. Each image was split into six sub-images with a partition grid built from two rows and three columns. This was done to maximise the amount of data we could use for training from each image, since down-scaling the resolution of each image without splitting it first would effectively omit a significant amount of data. Each sub-image was cropped into a square focused on the centre, and the new square dimensions of resolution 544×544 were scaled down to 256×256. This choice was based on the trade off between the higher amount of training data at higher resolutions and the reduced training time requirements at lower resolutions. Next, we scaled the pixel intensity values to a range of [0,1]. Finally, we converted the images to greyscale, since we believed that the colours in textures did not provide meaningful features and could potentially cause overfitting.

The overall dataset we used for training networks contained cholesteric and smectic A, C, I and F phases. The data we extracted also contained smectic E and soft crystal phases, however we decided to omit these because the amount of images was significantly less than the other phases. Furthermore, we had the option to simulate isotropic phase textures with a "salt and pepper" noise method. In the previous semester we decided to use this method and include the isotropic class in one of our tasks. The classification of this phase is trivial due to its characteristic dark appearance and it was thus observed to inflate the overall validation and test accuracy values for every network. Based on this observation, we decided to omit the isotropic phase. The overall dataset is shown in figure 5 and the distribution of images across the training, validation and test sets can be found in Appendix A.

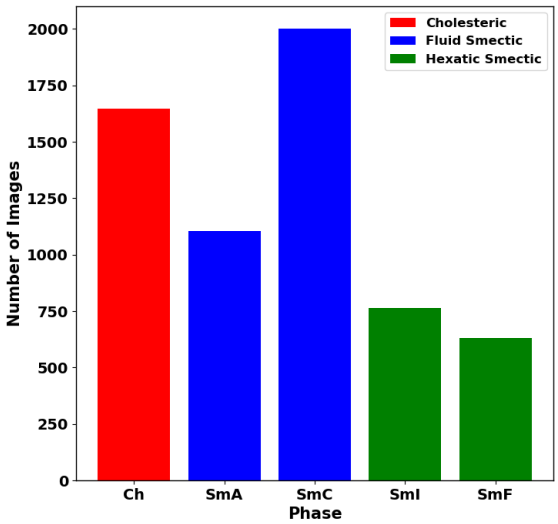


Figure 5: The numbers of images stored in our overall dataset for each phase, where the total was 6,978. The three groups which the phases belong to have been colour-coded for clarity.

4.2 Training

The Tensorflow and Keras [19,20] libraries were used to implement the architectures and create training loops. The networks were trained with either a 12GB NVIDIA Tesla K80 GPU, which was provided by the cloud service Google Colab, or an NVIDIA RTX 2060. Deep networks are trained almost exclusively with GPU hardware due to the ability to perform tensor computations at a significantly faster rate than CPUs, which hence greatly reduces training time [21].

Augmentations are a set of transformations, such as rotations and zooming, which can be randomly applied to a batch of images during training. The altered images expose a network to different orientations of the objects in images and can improve regularisation. Based on work from the previous semester, we found that a set called "flip augmentations", which only included image flips with respect to the horizontal and vertical axes, produced the highest average accuracy for models and we thus decided to use only that set for our work this semester [10].

We decided to tune a subset of the hyperparameters for every dataset to improve model generalisation. The set included: the learning rate, which was used in the "Adam" optimiser and was varied from 10^{-5} to 10^{-3}; the batch size, which determined how many images would be processed together by the GPU and was varied from 16 to 64; and the number of kernels in the first convolutional layer, which was varied from 4 to 128 for the sequential and Inception architectures. The hyperparameters we tuned were exclusively chosen as positive integer powers of 2. Since GPU hardware has a number of physical processors which falls into the same set of numbers, this produces more efficient use of the GPU's memory [21].

The categorical cross entropy function was used to calculate the loss for each batch of images. Optimal weight values, for a given dataset and network variant, were selected based on which produced the lowest validation set loss value. We observed that the optimal model was generally found within 50 epochs and consequently decided to constrain the maximum number of epochs to this number. An early stopping operation was used to monitor validation loss and stop the training if it did not change by a value of at least 1 during 25 epochs, which reduced total training time.

Variance of both the validation and test accuracy values was observed and we thus decided to create ten models for a given hyperparameter configuration. The mean and sample standard deviation were then computed for the aforementioned accuracy values. This was repeated for many different

configurations, and was effectively a manual grid search of different hyperparameters to find the optimal configuration for each dataset. The imbalance of the class sizes, which can be seen in figure 5, meant that a relatively high number of correctly classified images for a given class could inflate the overall accuracy if that class had a significantly higher number of images compared to other classes. We decided to create a confusion matrix for the test accuracy, where the diagonal elements correspond to individual class accuracy values and the other elements contain the proportions of images incorrectly classified as a different class. This type of analysis can create a more complete picture of the performance of a network and can hence identify potential weaknesses in the classification of certain LC phases. The typical confusion matrix analysis was extended into two matrices to account for the ten repeat models. A given element in the first contained the mean of that same element from the ten models, and the second matrix contained the sample standard deviation of each element.

5 Classification Results

The first classification tasks we approached consisted of three binary class datasets. These tasks required less training time than those with a higher number of classes and also provided a gauge for how the networks might perform in the more complex tasks. The two other classification tasks we explored contained three and five classes respectively. Networks need to learn a greater number of features to correctly classify images of a given phase in these cases, and these tasks hence posed a greater challenge than the binary cases.

5.1 Summary of Previous Results

In the previous semester, we trained and tested multiple architectures on four different datasets to classify images of liquid crystal phases. The architectures we used included multiple sequential architectures, a down-scaled Inception V3 and ResNet50. The first dataset, "INChSm", was composed of four classes which included isotropic, nematic, cholesteric and a class which grouped together all of the smectic phase images. The second dataset was built from the fluid, hexatic and soft crystal (FHSC) groups. The third was a binary classifier dataset with only the smectic A and C phases (SmAC). The final dataset contained six classes which were smectic A, C, I, F, X1 and X2 (Sm6). Three repeats were conducted for each architecture and the final accuracy for each was the mean value, where the error on this estimate was computed as half the range. The highest mean test set accuracy and the corresponding architecture for each dataset is shown in table 1. Three out of the four classifiers provided evidence that deep learning could be applied to the classification of liquid crystal phases. However, the accuracy for the dataset with six classes was significantly worse than the other classifiers. We concluded that the size and imbalance of classes in our dataset created a bottleneck for the accuracy of our models and that we would need to collect extra data to improve the performance of networks [10].

Dataset	Number of Classes	Optimal Architecture	Highest Mean Test Accuracy/%
INChSm	4	Sequential 2 Layers	91 ± 4
FHSC	3	Sequential 4 Layers	86 ± 6
SmAC	2	Sequential 4 Layers	97 ± 1
Sm6	6	ResNet50	54 ± 1

Table 1: A tabulation of the highest mean test accuracy and corresponding architecture for each dataset in the previous work [10].

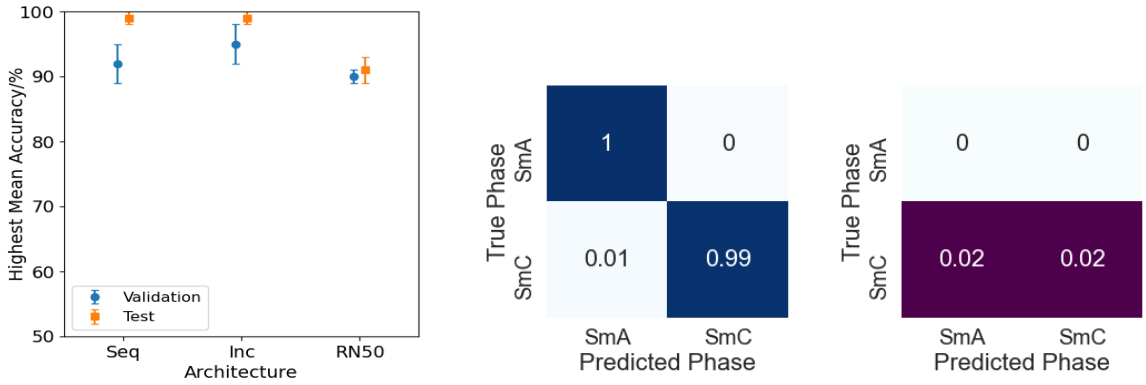
5.2 Smectic A and C

The first binary task contained a dataset with only the smectic A and C phases. The total amount of images for this dataset was 3,106, which had increased from 2,545 images in the previous work [10].

Architecture	Sequential	Inception	ResNet50
Layers	4	N/A	N/A
Modules	N/A	2	N/A
Starting Kernels	8	2	N/A
Batch Size	16	16	16
Learning Rate	10^{-4}	10^{-4}	10^{-5}
Mean Validation Accuracy/%	92 ± 3	95 ± 3	90 ± 1
Highest Mean Test Accuracy/%	99 ± 1	99 ± 1	91 ± 2

Table 2: The optimal configurations, highest mean test accuracy and the corresponding mean validation accuracy for each architecture trained on the smectic A and C dataset.

It can be seen in figure 6(a) that all three networks achieved a mean test accuracy above 90%. Despite the equivalent mean test accuracy that the sequential and Inception networks achieved, we decided that the latter network was optimal based on the higher mean validation accuracy it achieved. The mean test accuracy of (99±1)% was an improvement compared to the accuracy of (97±1)% achieved in the previous work [10]. The slight increase in accuracy could have been due to the larger size of the dataset.



(a) Summary (b) Mean (c) Standard Deviation

Figure 6: An illustration of the results for the smectic A and C classification task.

The mean and standard deviation confusion matrices for the Inception network are shown in figures 6(b) and 6(c). Both class accuracy values were approximately unity and the only source of error was a significantly small incorrect classification rate of smectic C phases as smectic A.

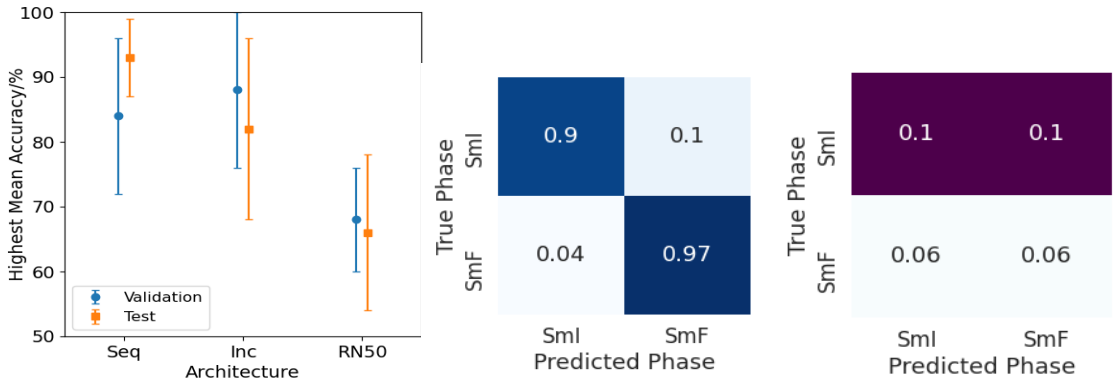
5.3 Smectic I and F

The next binary task required the distinction between smectic I and F phase textures. Our dataset for this task contained a total of 1,392 images, and was thus the smallest out of the three binary class datasets. The number of images in each class was approximately equal, which can be seen in Figure 5.

Architecture	Sequential	Inception	ResNet50
Layers	3	N/A	N/A
Modules	N/A	2	N/A
Starting Kernels	128	8	N/A
Batch Size	16	16	16
Learning Rate	10^{-4}	10^{-4}	10^{-5}
Mean Validation Accuracy/%	84 ± 12	88 ± 12	68 ± 8
Highest Mean Test Accuracy/%	93 ± 6	82 ± 14	66 ± 12

Table 3: The optimal configurations, highest mean test accuracy and the corresponding mean validation accuracy for each architecture trained on the smectic I and F dataset.

Figure 7(a) shows that the sequential network achieved the highest mean test accuracy of (93±6)%. The Inception network achieved (82±14)%, and ResNet50 displayed a poor performance with (66±12)%. We noted that the standard deviation of each of the validation and test accuracy values for each network was significantly greater than that which was seen for the binary tasks in sections 5.2 and 5.4.



(a) Summary (b) Mean (c) Standard Deviation

Figure 7: An illustration of the results for the smectic I and F classification task.

The individual class accuracy values for the optimal sequential network can be seen in the sub-figure 7(b), and the corresponding standard deviation values in 7(c). The mean proportion of correctly classified smectic F images was approximately unity, whereas that for smectic I had a lower value of 0.9 and provided a larger source of error in the overall test accuracy. The larger error and lower accuracy in classification of images from each class relative to the other binary tasks could have been due to the smaller size of the smectic I and F dataset. Furthermore, the features which distinguish the phases could be more subtle than those we extracted for the smectic A and C task.

5.4 Cholesteric and Smectic

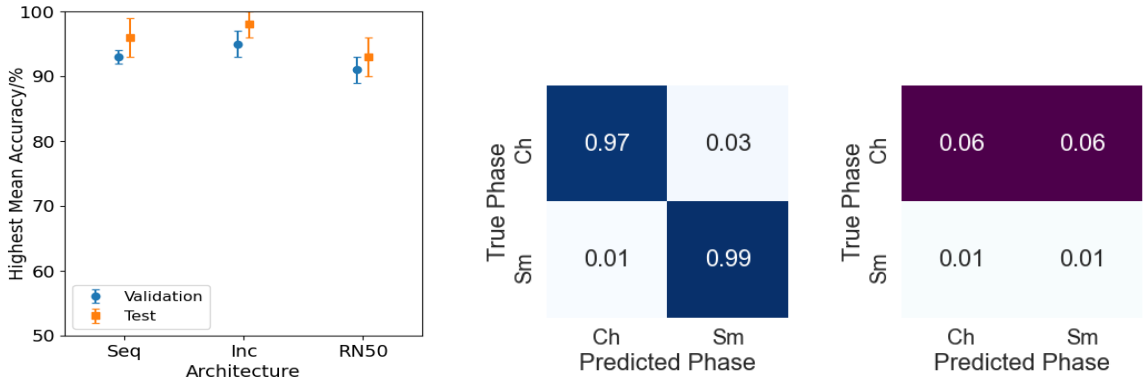
The final binary task we explored used a dataset that contained the cholesteric phase and another class, which was a collection of all of the smectic phase images we had. The total amount of images was 6,144, which was the highest out of all of the binary tasks.

Figure 8(a) shows that all three networks produced validation and test accuracy values which exceeded 90% and had small standard deviation values. An Inception network variant achieved the

Architecture	Sequential	Inception	ResNet50
Layers	3	N/A	N/A
Modules	N/A	1	N/A
Starting Kernels	64	16	N/A
Batch Size	16	16	16
Learning Rate	5×10^{-5}	10^{-4}	10^{-4}
Mean Validation Accuracy/%	93 ± 1	95 ± 2	91 ± 2
Highest Mean Test Accuracy/%	96 ± 3	98 ± 2	93 ± 3

Table 4: The optimal configurations, highest mean test accuracy and the corresponding mean validation accuracy for each architecture trained on the cholesteric and smectic dataset.

highest mean test accuracy of (98±2)%. These results were expected because the smectic phases exhibit a higher order than the cholesteric phase, and hence give rise to textures which can be easily distinguished from the latter.



(a) Summary (b) Mean (c) Standard Deviation

Figure 8: An illustration of the results for the cholesteric and smectic classification task.

Figures 8(b) and 8(c) respectively show the mean and standard deviation confusion matrices for the optimal Inception network. Both class accuracy values approached unity and the only significant source of variance in these values was the confusion of cholesteric phases with those from the mixed smectic group. One possible reason for this could have been a small bias created towards smectic phases during learning. This would have been due to the significantly large difference between the number of images in each of the two classes, which can be seen in figure 5.

5.5 Cholesteric and Two Smectic Groups

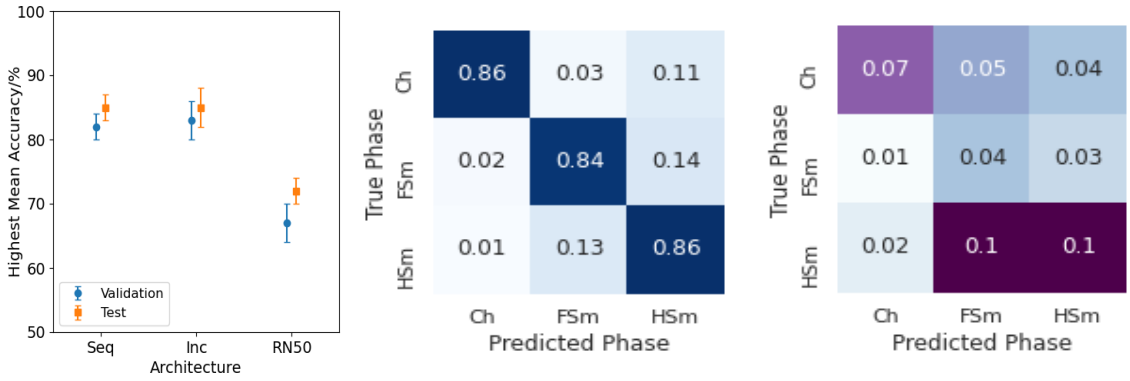
The smectic A and C phases were grouped into a fluid class, and the smectic I and F phases into a hexatic class. Along with the cholesteric phase, this formed a dataset with three classes and 6,978 images.

The sequential network produced a mean test accuracy of (85±2)%, the Inception network an accuracy of (85±3)% and ResNet50 produced a lower accuracy of (72±2)%. Figure 9(a) shows a summary of the results produced by the optimal networks.

The individual mean class accuracy values for the sequential network can be seen in figure 9(b), and the corresponding standard deviation values in figure 9(c). All three diagonal elements were over

Architecture	Sequential	Inception	ResNet50
Layers	3	N/A	N/A
Modules	N/A	1	N/A
Starting Kernels	64	8	N/A
Batch Size	16	16	16
Learning Rate	5×10^{-5}	10^{-4}	10^{-5}
Mean Validation Accuracy/%	82 ± 2	83 ± 3	67 ± 3
Highest Mean Test Accuracy/%	85 ± 2	85 ± 3	72 ± 2

Table 5: The optimal configurations, highest mean test accuracy and the corresponding mean validation accuracy for each architecture trained on the cholesteric and two smectic groups dataset.



(a) Summary (b) Mean (c) Standard Deviation

Figure 9: An illustration of the results for the cholesteric and two smectic groups classification task.

80%, however we noticed that each class was confused with a different class at a rate of 11-14%. In particular, cholesteric textures were confused with the hexatic group, and the fluid and hexatic groups were confused with each other. Based on the successful binary classification results, the size of the aforementioned incorrect classification rates was unexpected. It is possible that the higher number of classes in this task, and thus the higher number of initial features a model needs to extract, contributed to these errors. Consequently, we believe that exposing a network to a greater number of images during training could reduce the size of these errors.

5.6 Cholesteric and Four Smectic Phases

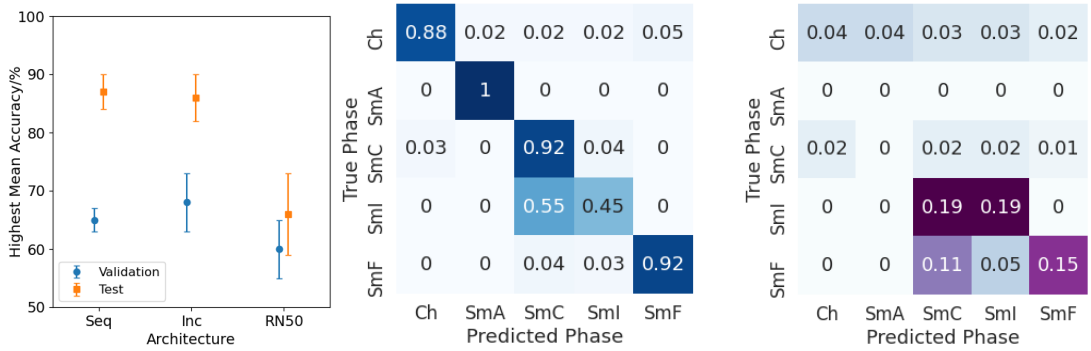
The second multinomial classification task we investigated was focused on a dataset which contained the cholesteric and smectic A, C, I and F phases. This dataset had 6,144 images and five classes, which was the highest out of all our datasets. The reduction in number of images compared to the cholesteric and two smectic groups dataset was due to the omission of certain smectic I images which we believed were mislabelled.

Mean test accuracy values of (87±3)% and (86±4)% were respectively achieved by the optimal sequential and Inception networks. The corresponding mean validation accuracy values were significantly lower, at (65±2)% and (68±5)%, which lead us to believe that there was sensitivity in how we sorted our dataset into training, validation and test sets. It is possible that the particular partition we used

Architecture	Sequential	Inception	ResNet50
Layers	3	N/A	N/A
Modules	N/A	2	N/A
Starting Kernels	128	4	N/A
Batch Size	16	16	16
Learning Rate	10^{-5}	10^{-4}	5×10^{-4}
Mean Validation Accuracy/%	65 ± 2	68 ± 5	60 ± 5
Highest Mean Test Accuracy/%	87 ± 3	86 ± 4	66 ± 7

Table 6: The optimal configurations, highest mean test accuracy and the corresponding mean validation accuracy for each architecture trained on the cholesteric and four smectic phases dataset.

created a greater degree of similarity between the training and test sets than that between the training and validation sets, which hence lead to the observed accuracy values. Before training, we were aware that our choice of partition could potentially create significant differences between validation and test accuracy values. We initially considered a more systematic method based on metrics which could measure the degree of similarity between images. However due to the number of images in our dataset, the time required to compare each image to every other image and measure the similarity would have required an impracticable amount of time. Researchers and practitioners with access to greater computational resources and a dataset similar in size to ours, which could suffer from the same partition sensitivity, could use a systematic method in order to potentially produce more robust results.



(a) Summary (b) Mean (c) Standard Deviation

Figure 10: An illustration of the results for the cholesteric and four smectic phases classification task.

The sequential network produced individual class accuracy values, shown in figure 10(b), which were all approximately 90% or higher apart from smectic I. This phase had an individual accuracy of 45%, which was unexpected based on the successful distinction between smectic I and F in section 5.3 and also between the fluid and hexatic groups in section 5.5. The greatest source of variance in the results, which can be seen in figure 10(c), appeared to originate from the confusion of the smectic I and F phases with smectic C. Additionally, the corresponding cofusion matrices we observed for the optimal Inception network appeared almost identical to those of the sequential network. Due to the significant architectural differences between these networks, this indicated that the improvement in accuracy had become independent of the specific variant we used for either of these networks. In this case, the size of the dataset itself would have been the bottleneck. ResNet50 displayed a significantly different

performance relative to the other two architectures for this task, and we concluded that the capacity had caused overfitting and was thus too high.

6 Conclusion

The viability of using CNNs to classify LC phase textures was explored. Three CNN architectures were created and specific configurations of networks were trained with images of LC textures, which were obtained via polarised microscopy. The hyperparameters were tuned to find the optimal network for a given dataset. Five image classification tasks were explored, each with a unique dataset. Three of these were binary tasks, where there were two classes, and the other two tasks respectively had three and five classes. Table 7 shows a summary of the results we observed.

Dataset	Number of Classes	Optimal Architecture	Mean Validation Accuracy/%	Highest Mean Test Accuracy/%
SmAC	2	Inception	95 ± 3	99 ± 1
SmIF	2	Sequential 3 Layers	84 ± 12	93 ± 6
ChSm	2	Inception	95 ± 2	98 ± 2
ChSm2	3	Sequential 3 Layers	82 ± 2	85 ± 2
ChSm4	5	Sequential 3 Layers	65 ± 2	87 ± 3

Table 7: Tabulation of the optimal architecture for each dataset and the corresponding mean validation and test accuracy values.

The optimal network for each of the binary classification tasks produced a mean test accuracy in excess of 90%, and the accuracy for the "SmAC" and "ChSm" datasets approached unity. The former showed an improvement compared to our previous result of 97±1%, and both results are not trivial due to the physical similarity between the two phases. The latter met expectations due to a large difference in order between the cholesteric and smectic phases. However, the variance for the accuracy on smectic I and F was significantly higher than that for the other two binary tasks. We noted that the smectic I and F dataset had only 1,392 images, which was small relative to the other two binary datasets. The lack of data could have caused the higher variance in accuracy values and lower mean test accuracy. Furthermore, it is possible that the distinct features in smectic I and F textures are more subtle than those in the other two binary tasks.

The multinomial classification tasks posed a greater challenge than the binary tasks, which was due to the higher number of classes and hence higher number of meaningful features which needed to be extracted. The optimal sequential and Inception networks achieved mean test accuracy values which were in excess of 80% and approximately equal. Additionally, their variance values were almost equal. The confusion matrices we observed for both optimal networks had a similar colour shade pattern, where the intensity of the shade can indicate which elements are smaller or larger. This lead us to believe that the mean test accuracy had started to saturate. Furthermore, the use of a new variant or different type of network appeared to have a diminished effect on this accuracy. We believe that the improvement in performance on the multinomial tasks is now mostly dependent on the size of the datasets used for training.

The sequential and Inception networks were observed to produce the optimal performance for all

of the classification tasks, where the only significant difference occurred for the smectic I and F task. ResNet50 perfomed worse than the former architectures for every task, with a particularly poor performance for the "SmIF", "ChSm2" and "ChSm4" tasks. In section 3, we noted that ResNet50 has a substantially large capacity, with 25,580,000 trainable parameters. It is possible that the capacity of the network combined with the small overall dataset caused overfitting. However, for classification tasks which involve a number of classes greater than that in "ChSm4" and a significantly larger dataset, we believe that high capacity networks could be required to extract and transform a greater number of features in images.

The results we produced have provided evidence that CNNs can successfully classify images of LC phase textures. Our work has consolidated the previous result we observed for the "SmAC" task, and has expanded upon the other tasks we explored in the previous semester. Furthermore, the use of exclusively experimental data, in contrast to the use of simulated textures in previous literature, has provided a more robust test for this application. We believe that the size of the datasets is the main limitation of network performance. For an overall dataset with a significantly larger size, we would expect that all accuracy values would exceed 90% and potentially approach unity. The creation of a large-scale database which contains LC texture images, where researchers and technicians could contribute labelled images or videos, would allow networks to be trained on datasets significantly larger than ours. This approach could also provide networks with enough data to learn the correct classification of additional phases such as smectic B, smectic E and soft crystal, which we had to omit from our dataset due to a relatively small amount of images. Finally, an increase in the size of test sets would provide a more robust evaluation of the ability of networks to generalise to novel LC images.

References

[1] Russell, S.J., Norvig, P., *Artificial Intelligence: A Modern Approach*. Prentice Hall, third ed., 2010.

[2] LeCun, Y., et al., "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, pp. 541–551, 1989.

[3] LeCun, Y., Bengio, Y., Hinton, G., "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.

[4] Krizhevsky, A., Sutskever, I., Hinton, G.E., "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[5] Cruz, J. A., Wishart, D. S., "Applications of machine learning in cancer prediction and prognosis," *Cancer Informatics*, vol. 2, pp. 59–77, 2006.

[6] Sigaki, H.Y.D., Lenzi, E.K., Zola, R.S., Perc, M., Ribeiro, H.V., "Learning physical properties of liquid crystals with deep convolutional neural networks," *Scientific Reports*, vol. 10, p. 7664, 2020.

[7] Minor, E.N., Howard, S.D., Gree, A.A.S., Glaser, M.A., Park, C.S., Clark, N.A., "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, pp. 1751–1759, 2020.

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

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

[10] Harbon, J., "Classification of liquid crystal phases via machine learning," 2021.

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

[12] Goodfellow, I., Bengio, Y., Courville, A., *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[13] Yamaguchi, K., Sakamoto, K., Akabane, T., Fujimoto, Y., "A neural network for speaker-independent isolated word recognition," *First International Conference on Spoken Language Processing*, 1990.

[14] Russakovsky, O., Deng, J., Su, H., Krause, J., et al., "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, pp. 211–252, 2015.

[15] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A. et al., "Going deeper with convolutions," *arXiv preprint arXiv:1409.4842*, vol. 1409, 2014.

[16] He, K., Zhang, X., Ren, S., Sun, J., "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[17] Schacht, J., Dierking, I., Gießelmann, F., Mohr, K., Zaschke, H., Kuczyński W., Zugenmaier, P., "Mesomorphic properties of a homologous series of chiral liquid crystals containing the -chloroester group," *Liquid Crystals*, vol. 19, pp. 151–157, 1995.

[18] Kaufman, S., Rosset, S., Perlich, C., "Leakage in data mining: Formulation, detection, and avoidance," *ACM Transactions on Knowledge Discovery from Data*, vol. 6, 2011.

[19] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C. et al., "TensorFlow: Large-scale machine learning on heterogeneous systems." Software available from tensorflow.org.

[20] F. Chollet *et al.*, "Keras." `https://keras.io`, 2015.

[21] S. Shi, Q. Wang, P. Xu, and X. Chu, "Benchmarking state-of-the-art deep learning software tools," 2017.

A Dataset Distribution

Phase	Total	Train	Validation	Test
Cholesteric	1646	1148	245	253
Smectic A	1106	722	180	204
Smectic C	2000	1388	301	311
Fluid Smectic	3106	2110	481	515
Smectic I	762	534	108	120
Smectic F	630	420	90	120
Hexatic Smectic	2226	1488	372	366
Smectic	5332	3598	853	881

Table 8: The number of images in each of the training, validation and test sets for all classes used. The datasets for each task were subsets of the overall dataset.