Lab book

Project: Can liquid crystal phases be identified via machine learning?

Project supervisor: Dr. Ingo Dierking

Other group: Josh Heaton and James Harbon

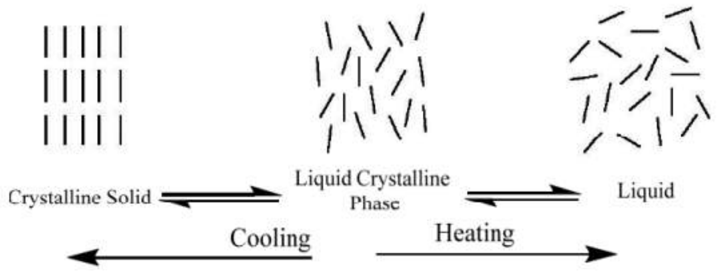
Context

Liquid crystals are a phase of matter between solid and liquid. Using polarizing optical microscopy, liquid crystal phases can be identified from the texture they produce. 'Texture' is due to anisotropy of liquid crystal. Here is a diagram from showing how POM works.

Experts can identify the LC phase, but can be difficult for highly ordered phases and for novices, i.e. phd or undergrad students. Project was designed to see if machine learning could identify liquid crystal phases from these textures. Machine learning is used for automated predictions from data. Specifically, computer vision is machine learning used for visual predictions, which will be useful for this project.

Liquid crystals brief overview:

- Useful diagram from H:\Work\RAMS44~1\4427-2_An.pmd (ipme.ru)



- Partially ordered
- Orientational, some positional order
- Diagram from H:\Work\RAMS44~1\4427-2_An.pmd (ipme.ru) showing catagorization of LC. There are many LCs, thermotropic change phase depending on temperature, lyotropic change due to concentration. Calamitic etc refers to molecular shape, calamitic is rod-like. Nematic, smectic and chiral nematic (sometimes called cholesteric) are most common groups, but there are more (see Textures of liquid crystals by Ingo Dierking) e.g smectic split further into SmA, SmC etc.

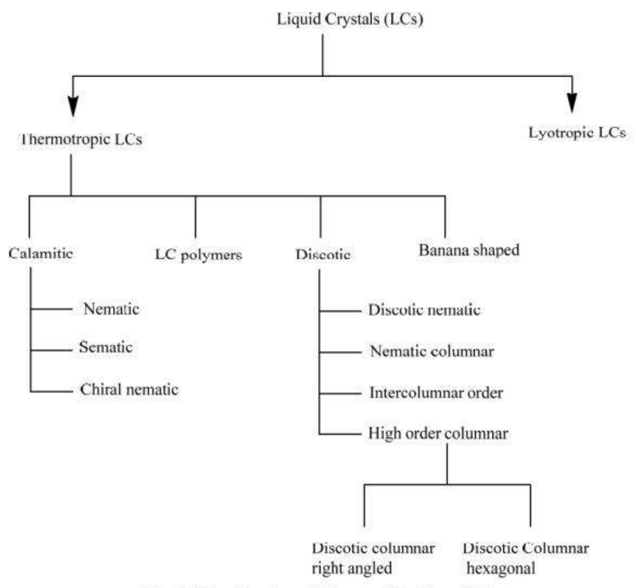


Fig. 3. Classification of LCs, modified from [13].

- Diagrams from Textures of liquid crystals by Ingo Dierking

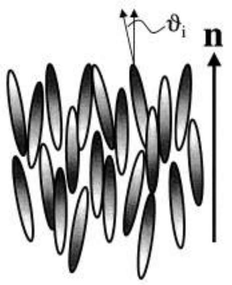


Fig. 1.5. Model structure of the nematic (N) phase. The spatial and temporal average of the long molecular axis is called the director **n**. The angle ϑ_i denotes the deviation of the long molecular axis of an individual mesogen *i* from the director.

Nematic, orientation order only

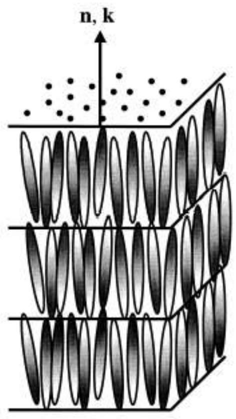


Fig. 1.7. Model structure of the smectic A (SmA) phase. The director **n** is oriented parallel to the smectic layer normal **k**, while within a smectic layer the molecules' centers of mass are isotropically distributed.

Smectic have some positional order, layered. More ordered, hence occur at lower temperature than nematic.

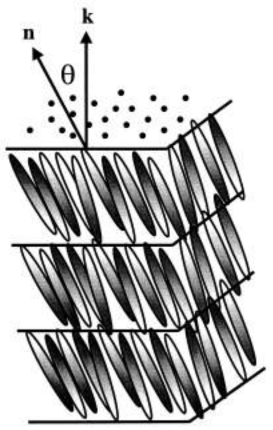


Fig. 1.10. Model structure of the smectic C (SmC) phase. The director **n** is tilted with respect to the smectic layer normal **k** by an angle θ, the director tilt angle, which is a temperature dependent quantity. The molecules' centers of mass within a smectic layer are isotropically distributed.

Smectic C has an additional 'tilt'.

Other smectic phases, with more order

14 | *1.3 General Structure of Liquid Crystal Phases*

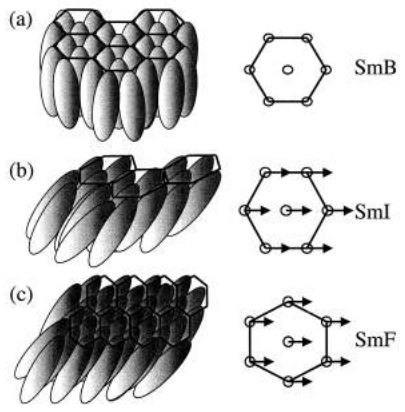


Fig. 1.14. Model structure of (a) the orthogonal SmB phase, (b) the SmI phase being tilted towards the apex of the hexagon, and (c) the SmF phase being tilted towards the side of the hexagon. (The soft crystal B, J, and K phases have a similar structure, only with positional order being long range.)

Some resources on liquid crystals:

- Textures of liquid crystals – Ingo Dierking
- http://www.personal.kent.edu/~bisenyuk/liquidcrystals/ - Shorter description of LC phases

Project aims

Research and application of machine learning techniques in order to classify a liquid crystal's phase from its texture. Possible extension to applying machine learning techniques to determining liquid crystal phase transitions from videos of textures undergoing a phase transition.

Project objectives

- Review of relevant literature, including: any use of machine learning in liquid crystal/soft matter research (may be limited), machine learning techniques for image classification (Convolutional neural networks (CNNs), clustering algorithms, etc.).
- Collect a dataset of LC videos and images (to be shared among the two groups) of liquid crystal textures in different phases.
- Implement machine learning algorithms in Keras and/or TensorFlow to perform image phase classification of the LC textures.

06/10/20 First meeting (video call)

Attendees: Dr. Dierking, Josh Heaton, James Harbon, Jason Dominguez

Discussed:

- Weekly meeting via Zoom at 2pm on Tuesdays.
- What approach, e.g. supervised, unsupervised, etc. (I would prefer to approach project using supervised as I have more experience with this).
- Where we will get the data (possibilities):
    - Youtube LC videos via screen grabbing software.
    - PhD student can get videos from lab of specific LC textures corresponding to a certain phase (so labelled data).
    - Dr Dierking will also see if he has some films.
    - Could simulate textures (I would prefer to use experimental textures)
    - Images off internet (there wont be many) which can be mixed into validation and test sets.
- Number of images may not be large (possible use of transfer learning?)
- Will need to consider image size and computational power (can possibly use free or uni access to more power, e.g. google cloud).
- Images won't need to be too big according to Dr Dierking as textures aren't too finely detailed.

06/10/20 Initial review of literature and initial ideas for project

Aim: Review literature relevant to project and generate some ideas for how to get started and where the project can go (i.e. how to differ from the other group)

Machine learning in liquid crystal/soft matter research:

- Advanced neural network clustering techniques for liquid crystal texture classification – 2013 thesis - Zoltan Karaszi: Compared multilayer perceptron (MLP) neural network and random tree algorithms, used image analysis software to define a feature vector corresponding to the LC textures.
- Machine learning algorithms for liquid crystal-based sensors – 2018 – Yankai Cao et al.
- Learning physical properties of liquid crystals with deep convolutional neural networks – 2019 – Higor Y. D. Sigaki et al.: Used deep CNN on simulated and experimental textures, determined isotropic or nematic and predicted properties such as pitch length from textures.
- Machine learning topological defects of liquid crystals in two dimensions – 2019 thesis – Michael Walters: Used many supervised and unsupervised algorithms on simulated textures to determine defects, also looked at recurrent NNs for isotropic-nematic phase transition.

- Machine learning-aided analysis for complex local structure of liquid crystal polymers – 2019 – Hideo Doi et al.
- Estimating physical properties from liquid crystal textures via machine learning and complexity-entropy methods – 2019 – H. Y. D. Sigaki et al.: Determined physical properties of experimental and simulated LC textures using k-nearest neighbors algorithm.
- End-to-end machine learning for experimental physics: using simulated data to train a neural network for object detection in video microscopy – 2020 – Eric N. Minor et al.: Object detection of defects in simulated LC texture videos.
- Predicting molecular ordering in binary liquid crystal using machine learning – 2019 – Takuya Inokuchi et al.: Predicting phase transition temperature from LC textures.

Liquid crystal physics:

- Textures of liquid crystals – Ingo Dierking
- http://www.personal.kent.edu/~bisenyuk/liquidcrystals/ - Shorter description of LC phases

Machine learning:

- Deep learning specialization – coursera – course I am taking on many aspects of implementing machine learning techniques, taught by Andrew Ng.
- Supervised and semi-supervised self-organizing maps for regression and classification focusing on hyperspectral data – 2019 – Felix M. Riese et al.

Initial ideas for project:

- Apply supervised learning with a deep CNN to labelled textures of LC phases.
- Training data from videos from lab (via PhD student and/or Dr Dierking) and validation and test data from mix of lab movie images and other movies and pictures obtained from internet (images to be obtained from movies using frame grabber software).
- Considering a well known CNN architecture (e.g. LeNet-5) due to simplicity and wealth of use in image classification: simple to implement in Keras, not too computationally expensive (not too deep).
- Note: all coding and file paths mention will be relative to this file path: C:\Users\Jason\Documents\University\Year_4\MPhys_Project(s)

08/10/20 CNNs explained:

- Machine learning aims to generate correct predictions from data.
- In this project we want to predict the phase of a LC from its texture image.
- A type of machine learning/deep learning which is targeted towards image detection is convolutional neural networks CNN.
- Diagram of a traditional neural network

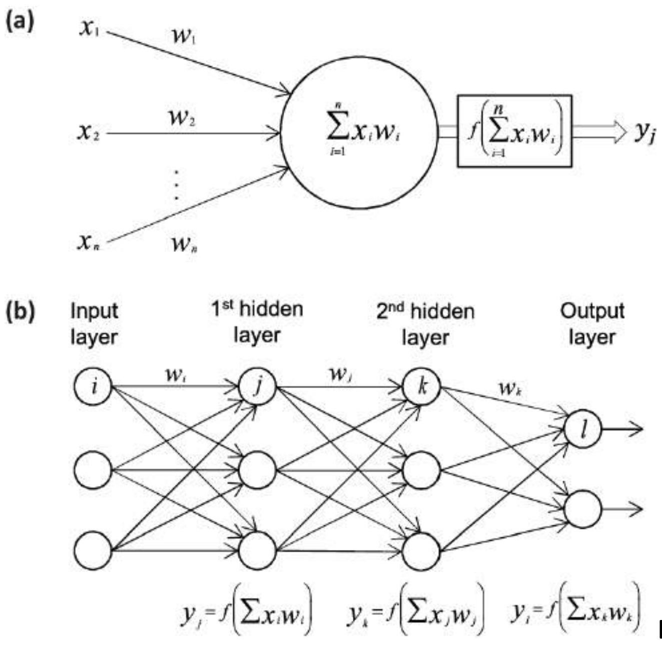


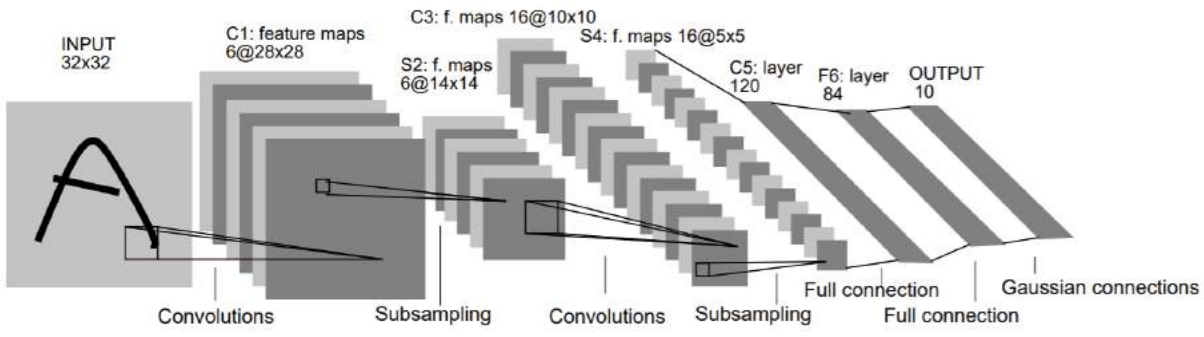
Diagram from https://www.researchgate.net/figure/a-The-building-block-of-deep-neural-networks-artificial-neuron-or-node-Each-input-x_fig1_312205163

A prediction y is generated from many interconnected 'neurons' which individually compute weighted sums of their inputs. The weights used in these sums are 'learned'. Learning requires generating an outcome and comparing it to a predicted outcome/label. Hence this requires labelled data. In this project data can be labelled based on what phase we know the texture is (i.e. 0 for isotropic, 1 for nematic etc.). This is called supervised learning. Weights are updated via backpropagation, e.g. using gradient descent to optimize weights with respect to a loss function. Loss function, e.g. log loss

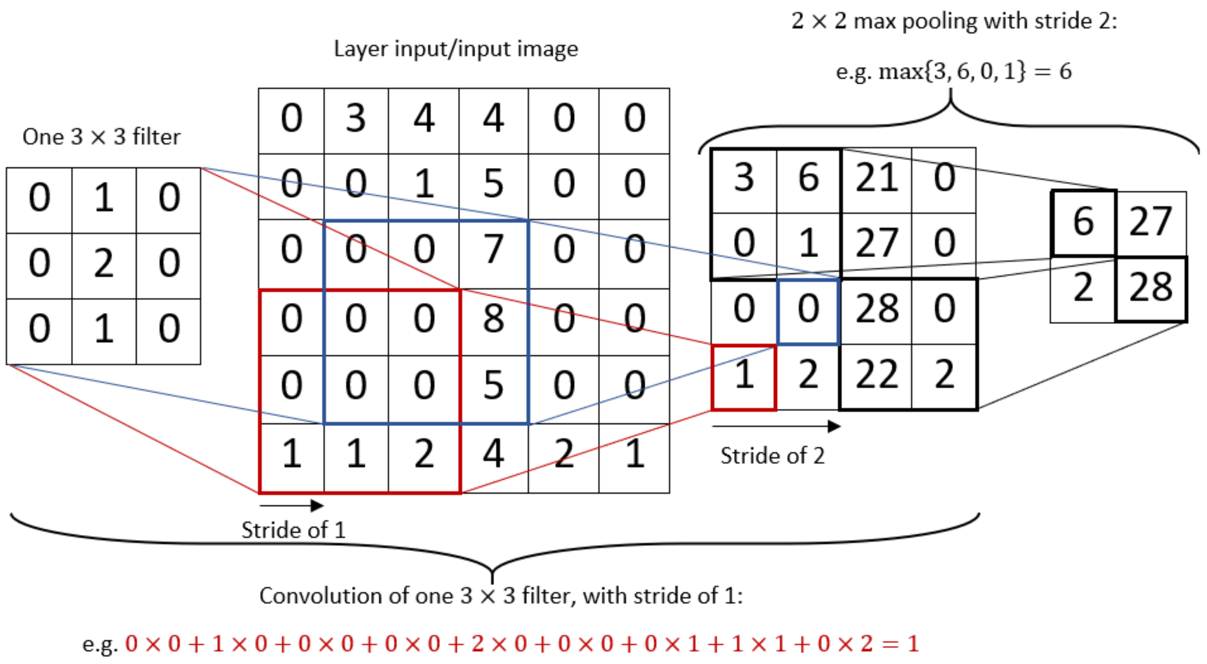
$$\sum_i y_i \log \hat{y_i} + (1 - \ y_i) \log 1 - \hat{y_i}$$

Where y_i is known value for image I and $\hat{y_i}$ is the prediction. The aim is to minimize this loss. Good resources for this topic are https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a and the coursera deep learning course mentioned before.

CNN build upon this neural network by being designed for images. They have convolution and pooling layers which act as feature extraction from the image. LeNet5 is a famous CNN architecture



Here is a diagram I made for explain the process of the convolution and pooling layers


Layer input/input image
One 3 × 3 filter
2 × 2 max pooling with stride 2:
e.g. max{3, 6, 0, 1} = 6
Stride of 1
Stride of 2
Convolution of one 3 × 3 filter, with stride of 1:
e.g. 0 × 0 + 1 × 0 + 0 × 0 + 0 × 0 + 2 × 0 + 0 × 0 + 0 × 1 + 1 × 1 + 0 × 2 = 1

There are many filters associated with each layer (you determine how many) and the values in these filter are what is 'learned for these layers'.

Keras is a coding framework in python used to implement CNN. https://keras.io/

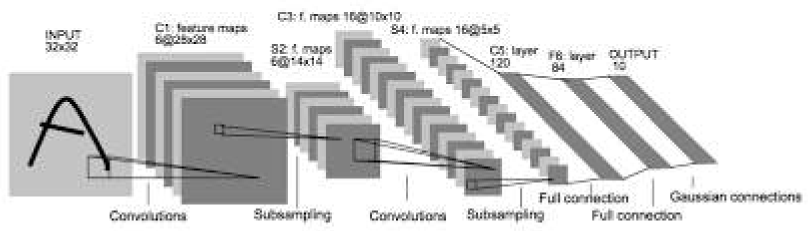
Aims/objectives for the week 1

- Implement simple neural network architecture to do supervised learning on simulated images of isotropic (black) and LC (bright) to return correct phase (LC or not)
  - Binary classification problem
  - (This was suggested as a good place to start by Dr Dierking in the first meeting)
- Look for possible sources of data on Youtube and for free frame grabber software to get images.
- Read and find more literature.

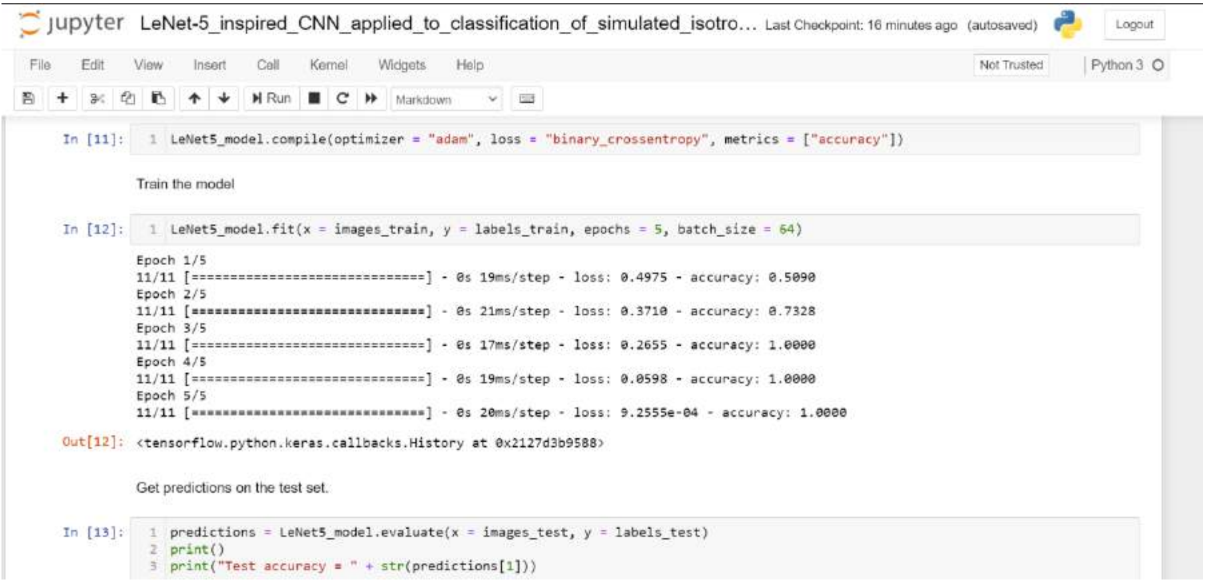
09/10/20 Binary classification of simulated isotropic and LC textures with LeNet-5 inspired CNN

Aim: Implement first CNN to the (simple) problem of isotropic or not from simulated textures

- Applied a LeNet-5 inspired CNN architecture on simulated textures for isotropic and non-isotropic (LC) phases.
- LeNet-5 architecture



- https://medium.com/@mgazar/lenet-5-in-9-lines-of-code-using-keras-ac99294c8086
- I used max pooling instead of average pooling because max pooling in more common in modern CNNs and I used relu activations through (expect on output) for the same reasoning.
- File found at: \Liquid_crystals-machine_learning\LiquidCrystalMachineLearning\Coding\LeNet-5_inspired_CNN_applied_to_classification_of_simulated_isotropic_and_liquid_crystal_(LC)_textures
- Implemented using keras as architecture was simple and this allowed for quick implementation.
- Achieved 100% training and validation accuracy – CNN was probably overkill for such a simple binary classification. (maybe apply a simple logistic regression unit to this problem and/or introduce noise into dark images)
- Screenshot of results


jupyter LeNet-5_inspired_CNN_applied_to_classification_of_simulated_isotro... Last Checkpoint: 16 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Markdown

```
In [11]:  LeNet5_model.compile(optimizer = "adam", loss = "binary_crossentropy", metrics = ["accuracy"])
```

Train the model

```
In [12]:  LeNet5_model.fit(x = images_train, y = labels_train, epochs = 5, batch_size = 64)

Epoch 1/5
11/11 [==============================] - 0s 19ms/step - loss: 0.4975 - accuracy: 0.5090
Epoch 2/5
11/11 [==============================] - 0s 21ms/step - loss: 0.3710 - accuracy: 0.7328
Epoch 3/5
11/11 [==============================] - 0s 17ms/step - loss: 0.2655 - accuracy: 1.0000
Epoch 4/5
11/11 [==============================] - 0s 19ms/step - loss: 0.0598 - accuracy: 1.0000
Epoch 5/5
11/11 [==============================] - 0s 20ms/step - loss: 9.2555e-04 - accuracy: 1.0000

Out[12]: <tensorflow.python.keras.callbacks.History at 0x2127d3b9588>
```

Get predictions on the test set.

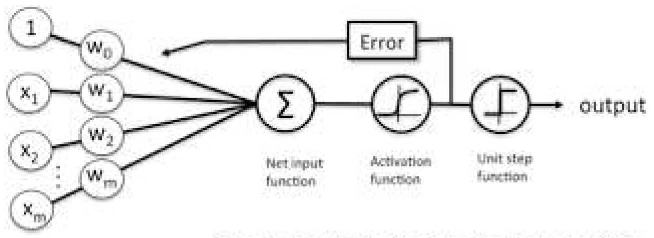
```
In [13]:  predictions = LeNet5_model.evaluate(x = images_test, y = labels_test)
          print()
          print("Test accuracy = " + str(predictions[1]))
```

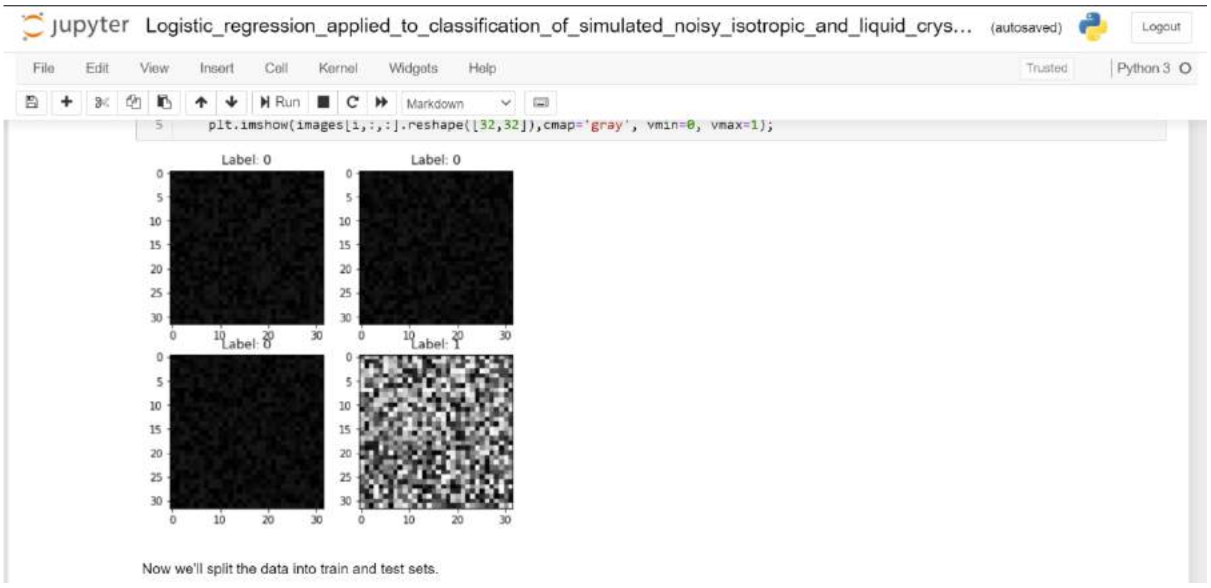
Summary: Implementing LeNet-5 inspired network for this simple classification seemed overkill, to advance introduce noise into dark isotropic images and maybe only a simple single logistic regression unit is necessary.

09/10/20 Simple logistic regression for binary classification of simulated isotropic and LC phases

Aim: As above, now want to attempt the same classification but only using a single logistic regression unit as this may be all that's necessary. And then to use noisy isotropic images

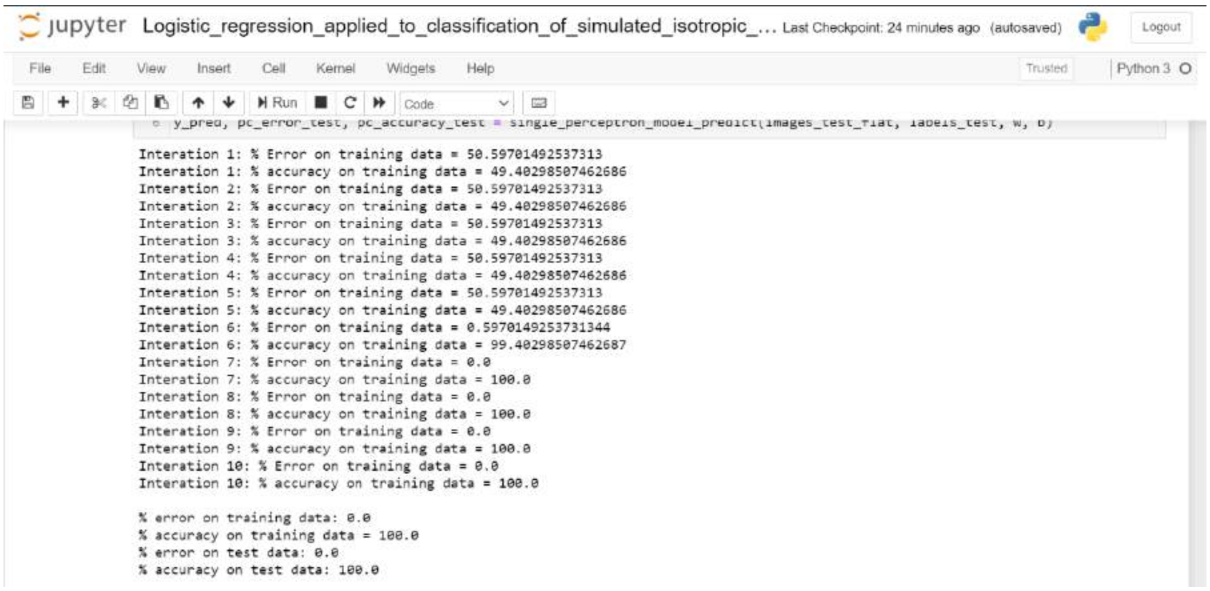
- As expected, even a simple 1 unit binary classification logistic regression was able to perfectly solve this task as 100% accuracy was achieved on training and validation.
- Diagram of single unit binary classification

- 

Schematic of a logistic regression classifier.
https://sebastianraschka.com/faq/docs/logisticregr-neuralnet.html
- File available at: \Liquid_crystals-machine_learning\LiquidCrystalMachineLearning\Coding\Logistic_regression_applied_to_classification_of_simulated_isotropic_and_liquid_crystal_(LC)_textures
- File available at: \Liquid_crystals-machine_learning\LiquidCrystalMachineLearning\Coding\Logistic_regression_applied_to_classification_of_simulated_noisy_isotropic_and_liquid_crystal_(LC)_textures
- Example of simulated images (with noise)
- 
jupyter Logistic_regression_applied_to_classification_of_simulated_noisy_isotropic_and_liquid_crys... (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Markdown
Label: 0
Label: 0
Label: 0
Label: 1

```
plt.imshow(images[1,:,:].reshape([32,32]),cmap='gray', vmin=0, vmax=1);
```

Now we'll split the data into train and test sets.
- Results

- 

jupyter
Logistic_regression_applied_to_classification_of_simulated_isotropic_... Last Checkpoint: 24 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

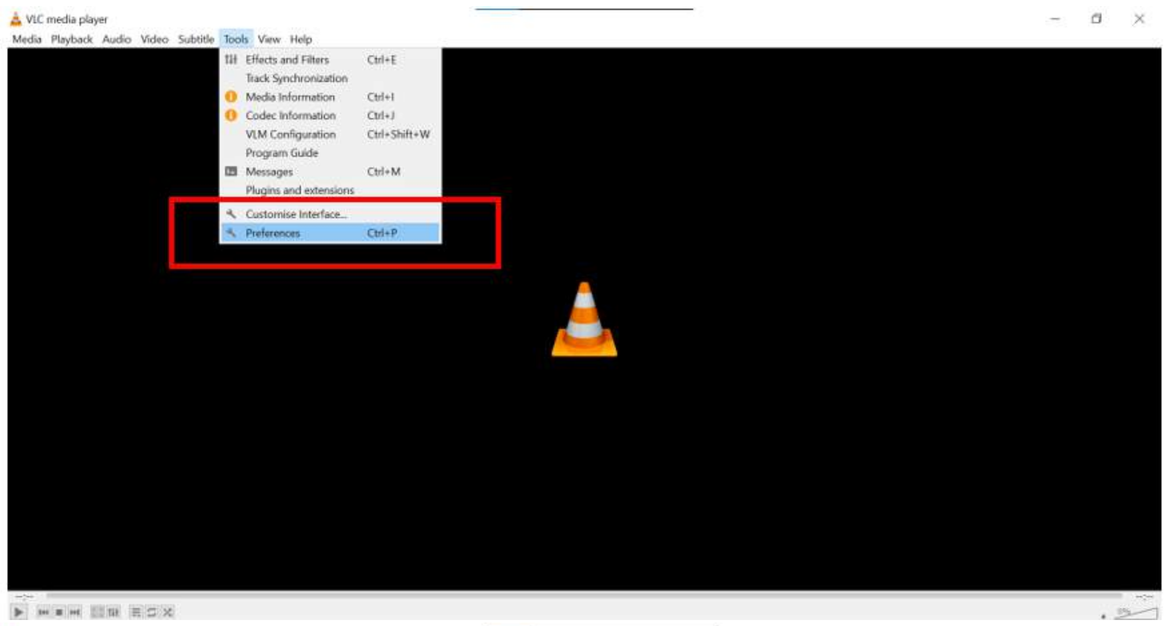
```
y_pred, pc_error_test, pc_accuracy_test = single_perceptron_model_predict(images_test_flat, labels_test, w, b)

Interation 1: % Error on training data = 50.59701492537313
Interation 1: % accuracy on training data = 49.40298507462686
Interation 2: % Error on training data = 50.59701492537313
Interation 2: % accuracy on training data = 49.40298507462686
Interation 3: % Error on training data = 50.59701492537313
Interation 3: % accuracy on training data = 49.40298507462686
Interation 4: % Error on training data = 50.59701492537313
Interation 4: % accuracy on training data = 49.40298507462686
Interation 5: % Error on training data = 50.59701492537313
Interation 5: % accuracy on training data = 49.40298507462686
Interation 6: % Error on training data = 0.5970149253731344
Interation 6: % accuracy on training data = 99.40298507462687
Interation 7: % Error on training data = 0.0
Interation 7: % accuracy on training data = 100.0
Interation 8: % Error on training data = 0.0
Interation 8: % accuracy on training data = 100.0
Interation 9: % Error on training data = 0.0
Interation 9: % accuracy on training data = 100.0
Interation 10: % Error on training data = 0.0
Interation 10: % accuracy on training data = 100.0

% error on training data: 0.0
% accuracy on training data = 100.0
% error on test data: 0.0
% accuracy on test data: 100.0
```

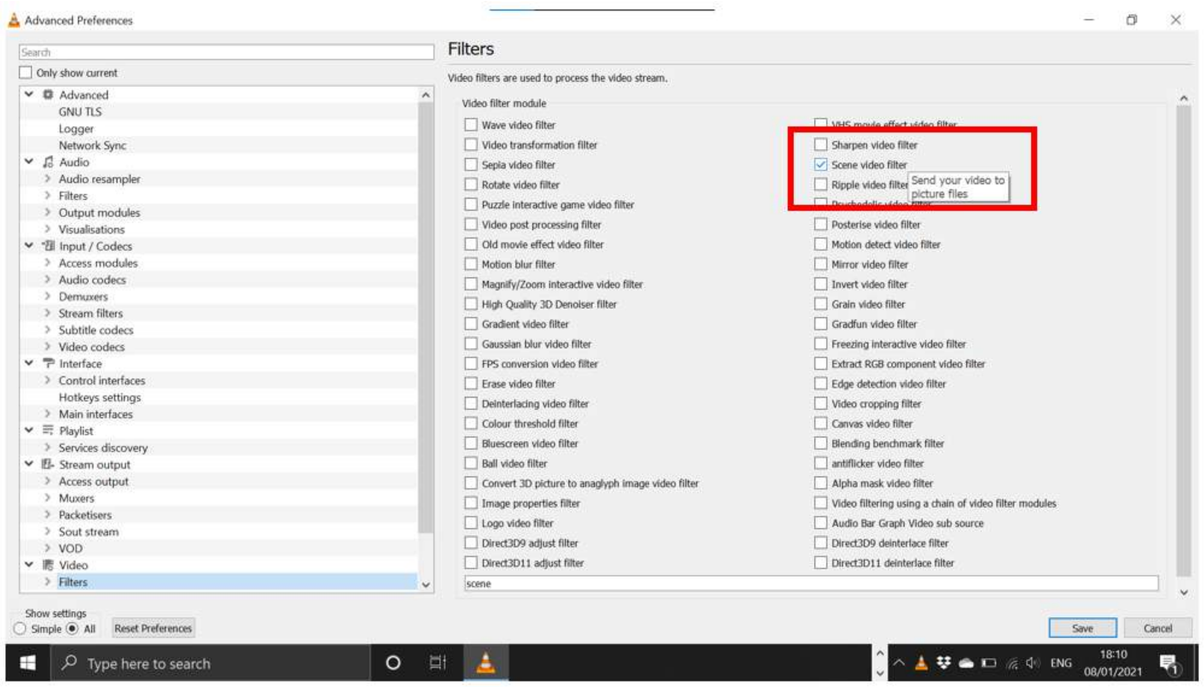
Summary: Even with the introduction of noise perfect results were achieved. Now we can move onto the more difficult task of using real LC textures – may want to return to CNN for more difficult task

10/10/20 Source of online LC texture images and frame grabber software

Aim: Find some online LC texture image/videos that could be used and find an open source frame grabbing software for getting images from videos

- Youtube channel "Vance Williams" has a couple useful looking videos with texture of some LC phases (e.g. cholesteric, focal conic Smectic A, etc.) including some phase transitions (e.g. isotropic-cholesteric, smectic A – nematic, etc.)
- Found a free, open source video software capable of frame grabbing – VLC media player
- Here are screenshots of using the software, scene filter needs to be turned on and then how often to grab frames and where they are saved can be set. Once done, you can load in a video and the grabbed frames will be sent to the designated file





Filters

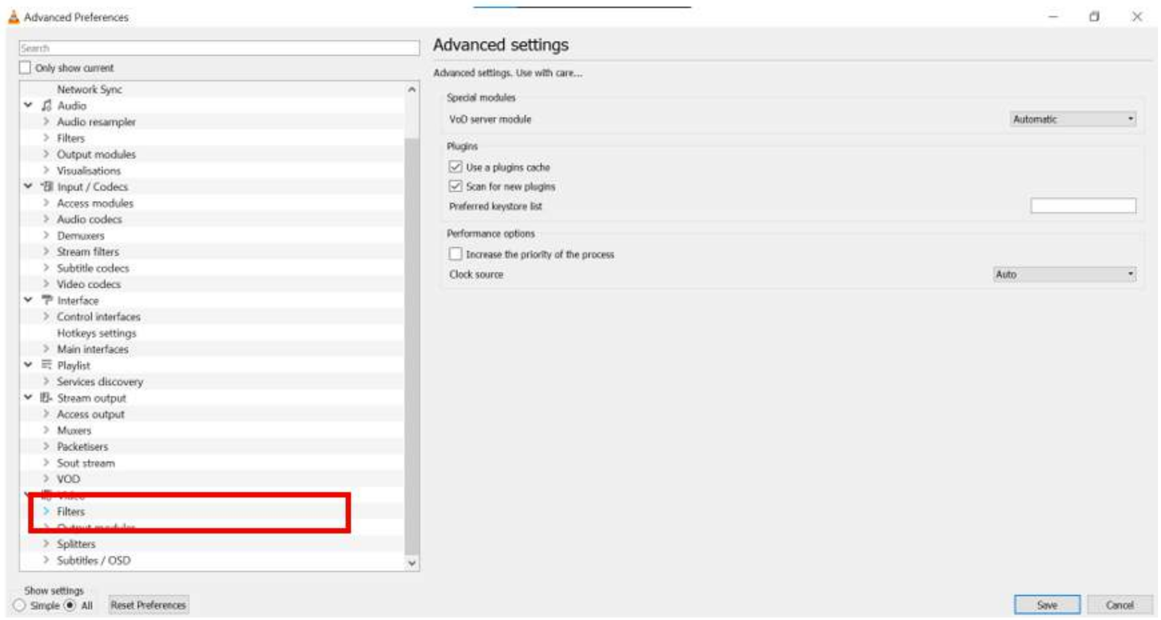
Video filters are used to process the video stream.

Video filter module

- [ ] Wave video filter
- [ ] Video transformation filter
- [ ] Sepia video filter
- [ ] Rotate video filter
- [ ] Puzzle interactive game video filter
- [ ] Video post processing filter
- [ ] Old movie effect video filter
- [ ] Motion blur filter
- [ ] Magnify/Zoom interactive video filter
- [ ] High Quality 3D Denoiser filter
- [ ] Gradient video filter
- [ ] Gaussian blur video filter
- [ ] FPS conversion video filter
- [ ] Erase video filter
- [ ] Deinterlacing video filter
- [ ] Colour threshold filter
- [ ] Bluescreen video filter
- [ ] Ball video filter
- [ ] Convert 3D picture to anaglyph image video filter
- [ ] Image properties filter
- [ ] Logo video filter
- [ ] Direct3D9 adjust filter
- [ ] Direct3D11 adjust filter
- [ ] VHS movie effect video filter
- [ ] Sharpen video filter
- [✓] Scene video filter
- [ ] Ripple video filter
- [ ] Psychedelic video filter
- [ ] Posterise video filter
- [ ] Motion detect video filter
- [ ] Mirror video filter
- [ ] Invert video filter
- [ ] Grain video filter
- [ ] Gradfun video filter
- [ ] Freezing interactive video filter
- [ ] Extract RGB component video filter
- [ ] Edge detection video filter
- [ ] Video cropping filter
- [ ] Canvas video filter
- [ ] Blending benchmark filter
- [ ] antiflicker video filter
- [ ] Alpha mask video filter
- [ ] Video filtering using a chain of video filter modules
- [ ] Audio Bar Graph Video sub source
- [ ] Direct3D9 deinterlace filter
- [ ] Direct3D11 deinterlace filter

Send your video to picture files

scene


Advanced Preferences

Advanced settings

Advanced settings. Use with care...

Special modules

VoD server module: Automatic

Plugins

[✓] Use a plugins cache

[✓] Scan for new plugins

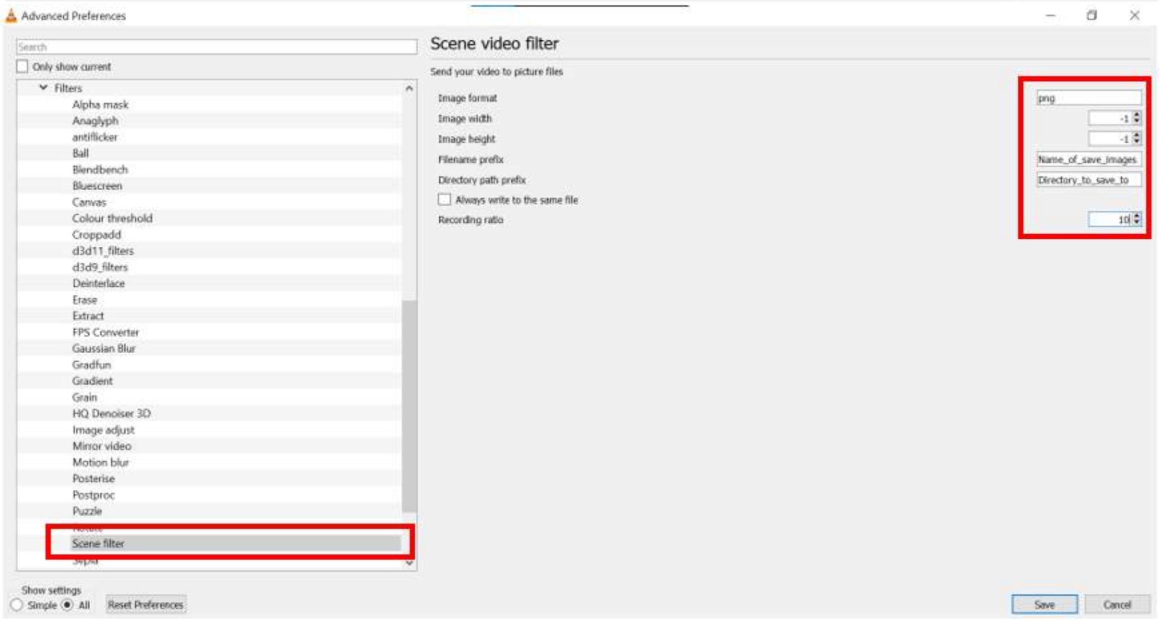
Preferred keystore list:

Performance options

[ ] Increase the priority of the process

Clock source: Auto

Show settings: Simple [✓ All]


Advanced Preferences

Scene video filter

Send your video to picture files

Image format: png

Image width: -1

Image height: -1

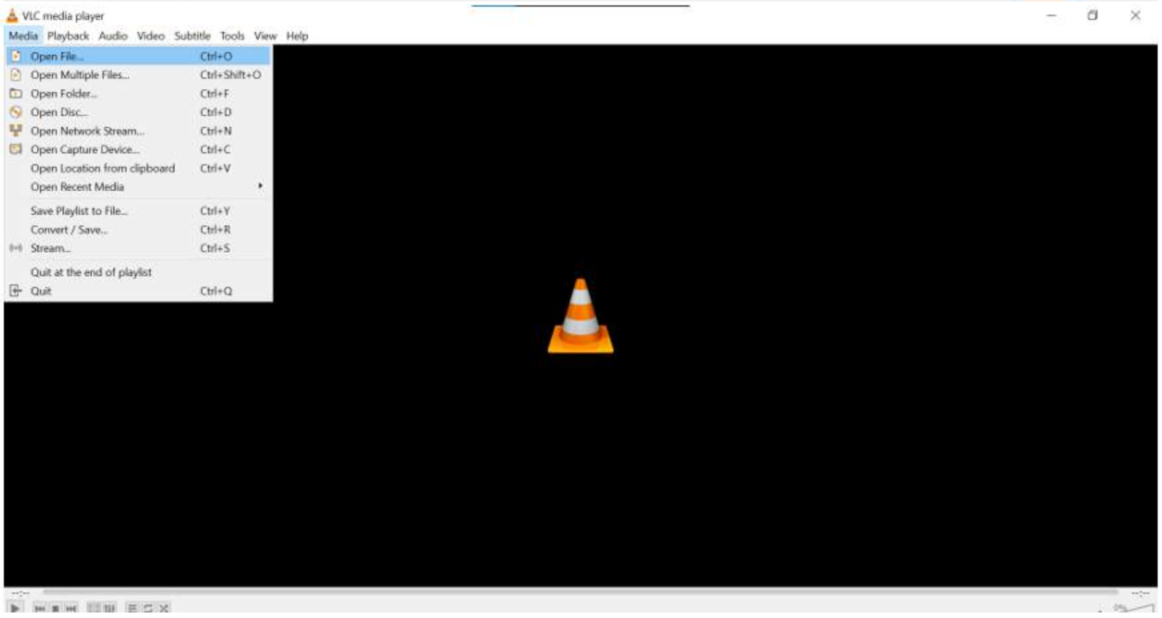
Filename prefix: Name_of_save_images

Directory path prefix: Directory_to_save_to

[ ] Always write to the same file

Recording ratio: 10

Show settings: Simple [✓ All]



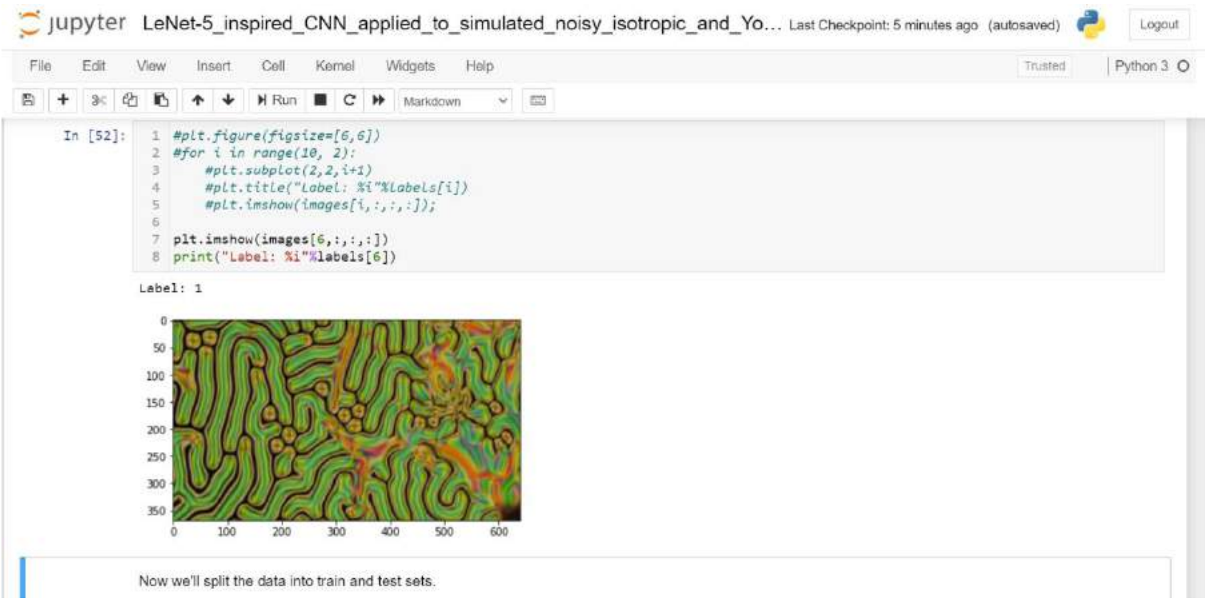
- Downloaded Youtube videos using online – y2mate.gure (doesn't look the most secure, so would use without antivirus software. I'll send to the other team so they don't have to use this software to get the same videos)
- Note on using Youtube videos: downloading videos from other channels is against Youtube's terms and conditions so this may not be okay for long term use (would have to get permission) but will be okay for private testing now.

Summary: Found open source frame grabber and downloaded use youtube videos for LC texture images, but need to get permission for use of these images in the project.

10/10/20 Using CNN on combination of Youtube LC images and simulated isotropic images

Aim: Advance on pervious work by incorporating actual images of LC textures (obtained online) into classification of LC or not

- Example of images used

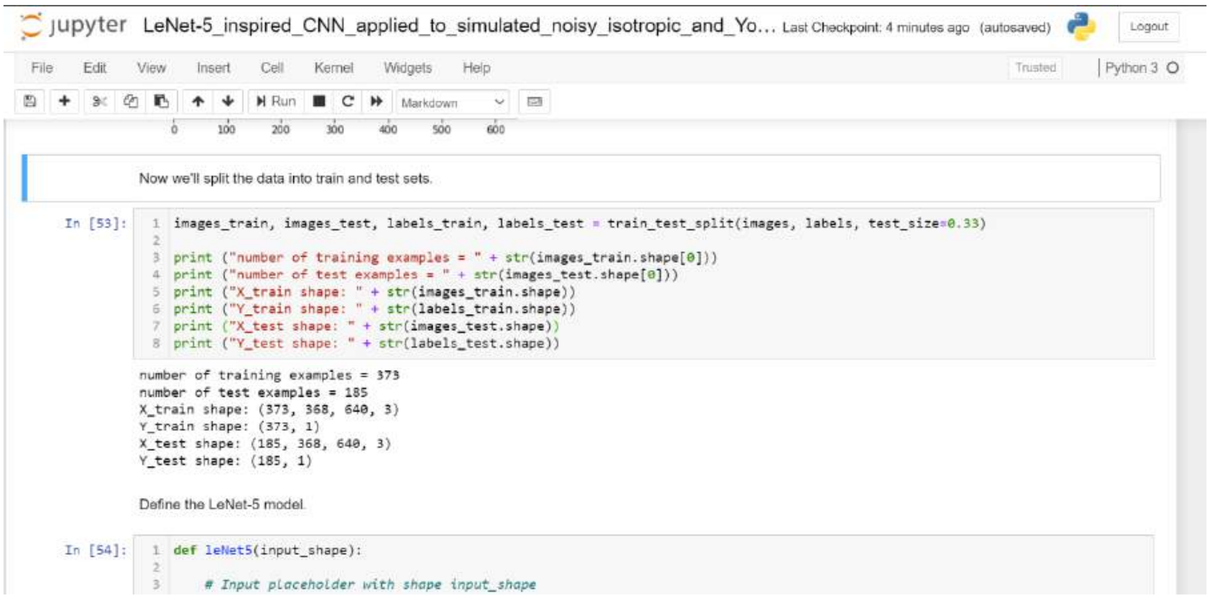

jupyter
LeNet-5_inspired_CNN_applied_to_simulated_noisy_isotropic_and_Yo… Last Checkpoint: 5 minutes ago (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3
Run
Markdown
In [52]:

```
#plt.figure(figsize=[6,6])
#for i in range(10, 2):
    #plt.subplot(2,2,i+1)
    #plt.title("Label: %i"%labels[i])
    #plt.imshow(images[i,:,:,:]);

plt.imshow(images[6,:,:,:])
print("Label: %i"%labels[6])
```

```
Label: 1
```

Now we'll split the data into train and test sets.

-
- Label 1 was for LC, 0 for isotropic
- Used the LeNet-5 inspired architecture in keras used before.
- Size of data array, showing number of images used for training and validation


jupyter LeNet-5_inspired_CNN_applied_to_simulated_noisy_isotropic_and_Yo... Last Checkpoint: 4 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Markdown
0 100 200 300 400 500 600

Now we'll split the data into train and test sets.

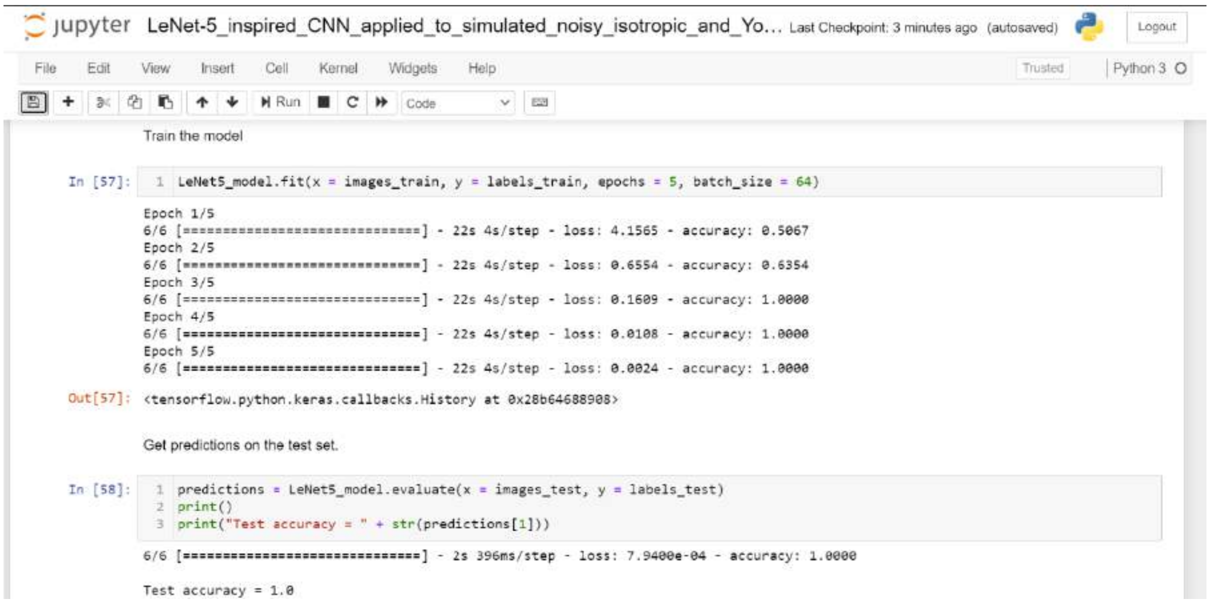
In [53]:
```
images_train, images_test, labels_train, labels_test = train_test_split(images, labels, test_size=0.33)

print ("number of training examples = " + str(images_train.shape[0]))
print ("number of test examples = " + str(images_test.shape[0]))
print ("X_train shape: " + str(images_train.shape))
print ("Y_train shape: " + str(labels_train.shape))
print ("X_test shape: " + str(images_test.shape))
print ("Y_test shape: " + str(labels_test.shape))
```

```
number of training examples = 373
number of test examples = 185
X_train shape: (373, 368, 640, 3)
Y_train shape: (373, 1)
X_test shape: (185, 368, 640, 3)
Y_test shape: (185, 1)
```

Define the LeNet-5 model.

In [54]:
```
def leNet5(input_shape):

    # Input placeholder with shape input_shape
```

-
- Results


jupyter LeNet-5_inspired_CNN_applied_to_simulated_noisy_isotropic_and_Yo... Last Checkpoint: 3 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

Train the model

In [57]:
```
LeNet5_model.fit(x = images_train, y = labels_train, epochs = 5, batch_size = 64)
```

```
Epoch 1/5
6/6 [==============================] - 22s 4s/step - loss: 4.1565 - accuracy: 0.5067
Epoch 2/5
6/6 [==============================] - 22s 4s/step - loss: 0.6554 - accuracy: 0.6354
Epoch 3/5
6/6 [==============================] - 22s 4s/step - loss: 0.1609 - accuracy: 1.0000
Epoch 4/5
6/6 [==============================] - 22s 4s/step - loss: 0.0108 - accuracy: 1.0000
Epoch 5/5
6/6 [==============================] - 22s 4s/step - loss: 0.0024 - accuracy: 1.0000
```

Out[57]:
```
<tensorflow.python.keras.callbacks.History at 0x28b64688908>
```

Get predictions on the test set.

In [58]:
```
predictions = LeNet5_model.evaluate(x = images_test, y = labels_test)
print()
print("Test accuracy = " + str(predictions[1]))
```

```
6/6 [==============================] - 2s 396ms/step - loss: 7.9400e-04 - accuracy: 1.0000

Test accuracy = 1.0
```

-

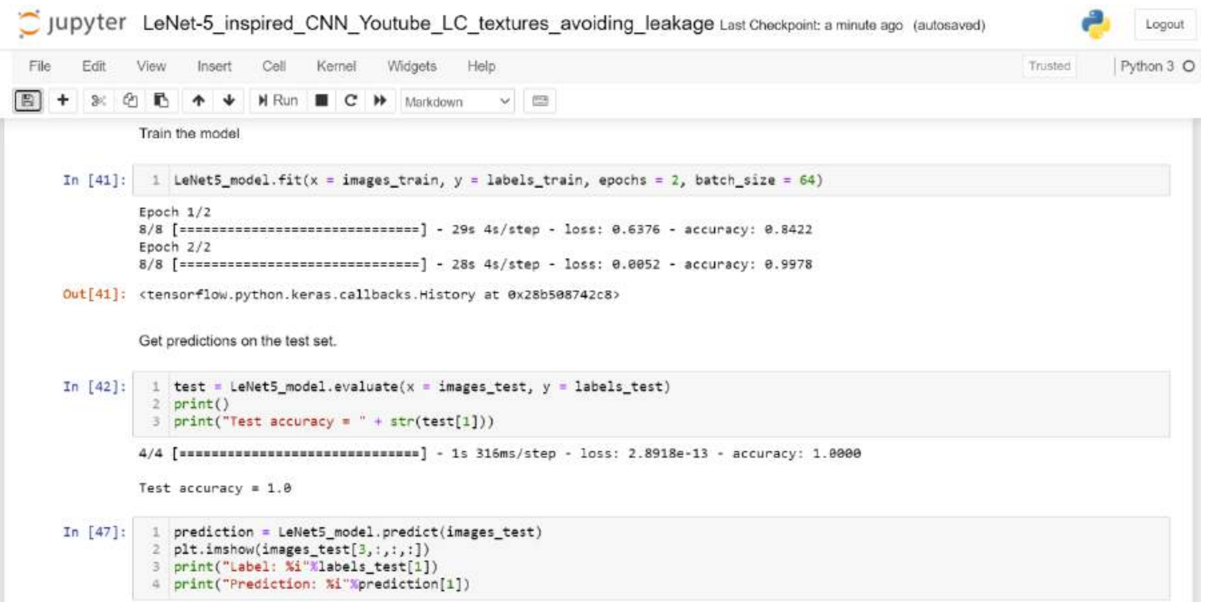
Summary: Perfect training and validation accuracy was achieved when incorporating LC texture from the internet into the dataset and using a simple CNN. However, images from 4 videos were randomly split and shuffles into training and validation sets regardless of what video they came from. Although the images were all different in training and validation set, this may be a form of data leakage – to avoid this, make sure images from the same video are not split amongst training and validation sets.

13/10/20 Solving possible data leakage issue

Aim: Try to resolve the issue of data leakage in the previous experiment by making sure images from the same video are not split among training and validation sets

- Data leakage is when you get a high training and validation accuracy, but it is misleading as examples in the validation set were so close to, or the same as, examples the model has trained on. Essentially it has gotten a peak at the answers and will not generalise well to truly unseen data – which is the overall aim.

- Chose only one of the 4 videos images to be in the validation set, the others were shuffled in the training set.
- File found at: \Liquid_crystals-machine_learning\LiquidCrystalMachineLearning\Coding\LeNet-5_inspired_CNN_Youtube_LC_textures_avoiding_leakage
- Results


jupyter
LeNet-5_inspired_CNN_Youtube_LC_textures_avoiding_leakage Last Checkpoint: a minute ago (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3
Run
Markdown

Train the model

```
In [41]: LeNet5_model.fit(x = images_train, y = labels_train, epochs = 2, batch_size = 64)

Epoch 1/2
8/8 [==============================] - 29s 4s/step - loss: 0.6376 - accuracy: 0.8422
Epoch 2/2
8/8 [==============================] - 28s 4s/step - loss: 0.0052 - accuracy: 0.9978

Out[41]: <tensorflow.python.keras.callbacks.History at 0x28b508742c8>
```

Get predictions on the test set.

```
In [42]: test = LeNet5_model.evaluate(x = images_test, y = labels_test)
         print()
         print("Test accuracy = " + str(test[1]))

4/4 [==============================] - 1s 316ms/step - loss: 2.8918e-13 - accuracy: 1.0000

Test accuracy = 1.0

In [47]: prediction = LeNet5_model.predict(images_test)
         plt.imshow(images_test[3,:,:,:])
         print("Label: %i"%labels_test[1])
         print("Prediction: %i"%prediction[1])
```

- This appears to be working well, but as the datasets are so small, we can’t expect this to generalise well.

Summary: When accounting for possible data leakage we still see good performance, however more data will be need to ensure good generalisation of the model

Week 1 summary:

- Saw positive results (100% accuracies) from using CNNs on simulated and experimental texture images in a LC or not classification.
- Collected images from Youtube videos
- Came across a problem of possible data leakage and attempted to solve it by ensuring images from same video were not split among training and validation sets – this may be useful as model gets better and we get more data and try harder classification problems.
- The process of collecting and reading literature around the topic is an ongoing one, reading around problems as they arise.

13/10/20 Second meeting (video call)

Attendees: Dr. Dierking, Josh Heaton, James Harbon, Jason Dominguez

Discussed:

- Meeting time changed to 4pm-5pm Tuesdays, regular from now on.

- Dr Dierking said he would get his videos to us.
- I asked about how black an isotropic texture actually is (as I had seen in papers that the texture of an isotropic phase wasn't 100% black). Dr Dierking said, except from the ideal case, we're dealing with mixtures where two phases coexist for a time, around the critical temperature – not a sudden switch from all LC to all isotropic.
- Joshua Heaton mentioned that he got permission from Vance Williams to use his videos etc. for the project.
- We need to collect more data, Joshua Heaton mentioned getting images/videos off of Vance Williams Instagram as well as youtube.
- Need to think of ways in which the two projects can differ.

Aims/objectives for week 2

- Progress onto more difficult classification problem (e.g. nematic vs non-nematic, or multi-class phase classification, nematic or cholesteric or smectic A,…)
- Get more Youtube videos for more images to use.
- Share any images/videos with other group
- Continue looking for and reading literature around the topic
- Think about ways in which my project can differ from the other group (maybe looking into the transition between isotropic and LC, maybe comparing different machine learning techniques – transfer learning, unsupervised (bag of visual words, self-organizing map))

14/10/20 GitHub

Aim: Set up a github repository for coding etc. in this project and share with the other group

- I set up a github repository with code done so far that I will share with the other group as we are likely to encounter similar problems.
- Found at: https://github.com/jasonjdominguez99/LiquidCrystalMachineLearning

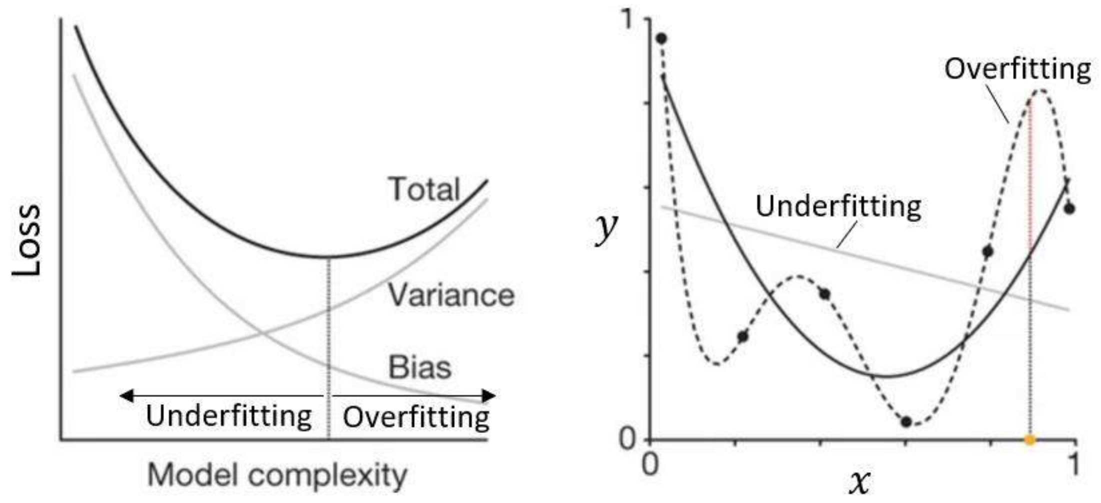
Summary: GitHub repo was set up for keeping updated code for the project and sharing with the other group to solve similar problems faced

15/10/20 First attempt at distinguishing between LC phases

Aim: Progress on the work done in week 1 by applying a CNN architecture to the binary classification problem of cholesteric phase or not

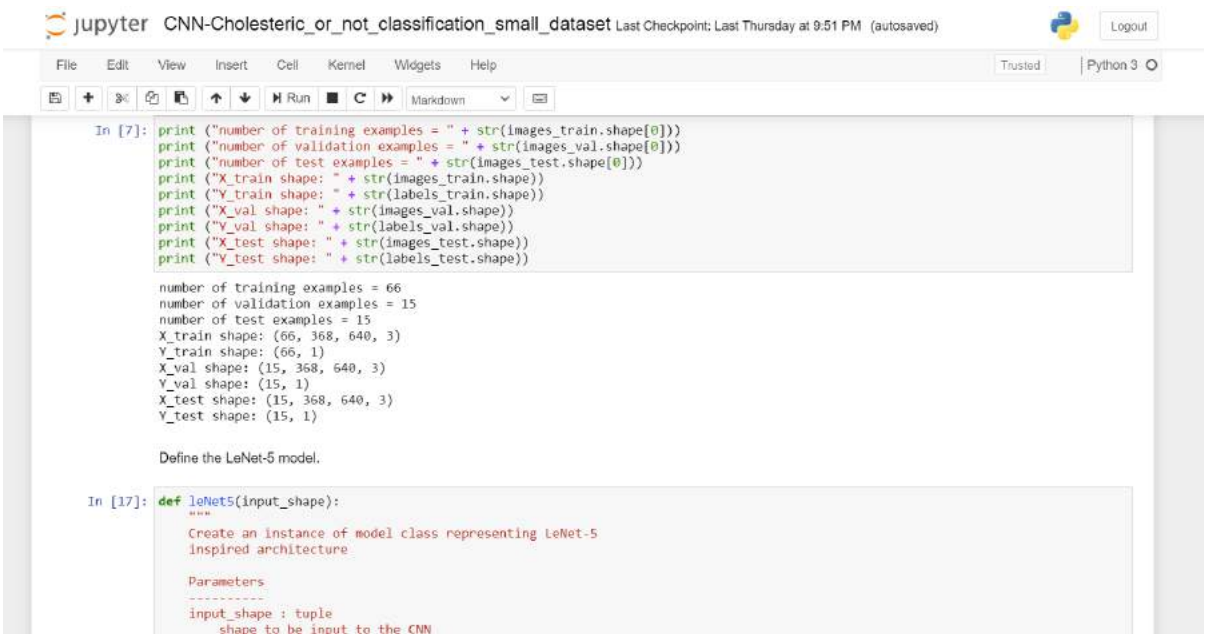
- Firstly I got more videos and images from Vance Williams youtube channel, which I shared with the other group.
- Chose to do cholesteric or not classification as I had many more cholesteric phase texture images than the others.
- Used LeNet-5 inspired architecture as before
- Roughly 60% validation accuracy on unseen data for 100% accuracy on training data

- Suggests overfitting the data as training accuracy >> validation accuracy. Learning patterns that aren't generalizable
- Overfitting is represented by this graph from J. Lever, M. Krzywinski, and N. Altman, "Model selection and overfitting," Nature Methods, vol. 13, no. 9, pp. 703–4, 2016. where I have indicated whether the model is under or overfitting. Overfitting models the noise in data, not the true pattern of the data, underfitting fails to model the complexity of the task.



- Essentially no better than random guessing, possibly an issue due to the imbalanced dataset used. Many cholesteric examples and very few examples of anything else.

- Another problem faced is that loading in all the images takes far too long that I wasn't able to use all the images I have available. Possibly looking into storing images into a more efficient database like HDF5 file and use the h5py library?
- Files available at:
  \Liquid_crystals-machine_learning\LiquidCrystalMachineLearning\Coding\CNN-Cholesteric_or_not_classification
  \Liquid_crystals-machine_learning\LiquidCrystalMachineLearning\Coding\CNN-Cholesteric_or_not_classification_small_dataset
- Training and validation results on small amount of data (66 training examples and 15 validation)

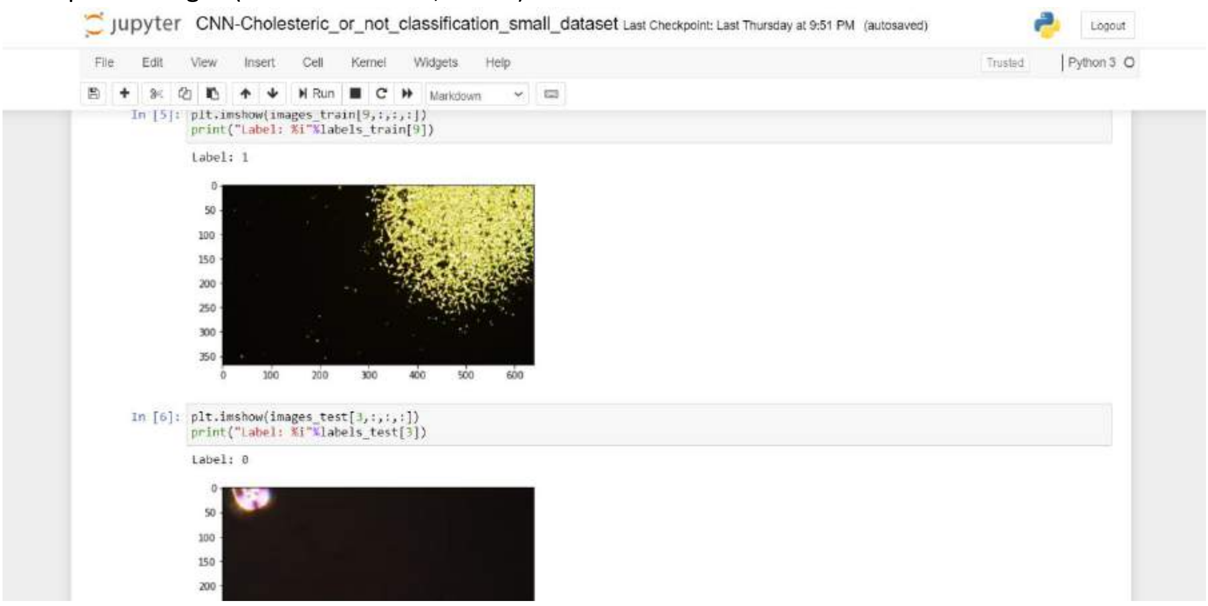

jupyter CNN-Cholesteric_or_not_classification_small_dataset Last Checkpoint: Last Thursday at 9:51 PM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Markdown

```
In [7]: print ("number of training examples = " + str(images_train.shape[0]))
        print ("number of validation examples = " + str(images_val.shape[0]))
        print ("number of test examples = " + str(images_test.shape[0]))
        print ("X_train shape: " + str(images_train.shape))
        print ("Y_train shape: " + str(labels_train.shape))
        print ("X_val shape: " + str(images_val.shape))
        print ("Y_val shape: " + str(labels_val.shape))
        print ("X_test shape: " + str(images_test.shape))
        print ("Y_test shape: " + str(labels_test.shape))

        number of training examples = 66
        number of validation examples = 15
        number of test examples = 15
        X_train shape: (66, 368, 640, 3)
        Y_train shape: (66, 1)
        X_val shape: (15, 368, 640, 3)
        Y_val shape: (15, 1)
        X_test shape: (15, 368, 640, 3)
        Y_test shape: (15, 1)
```

Define the LeNet-5 model.

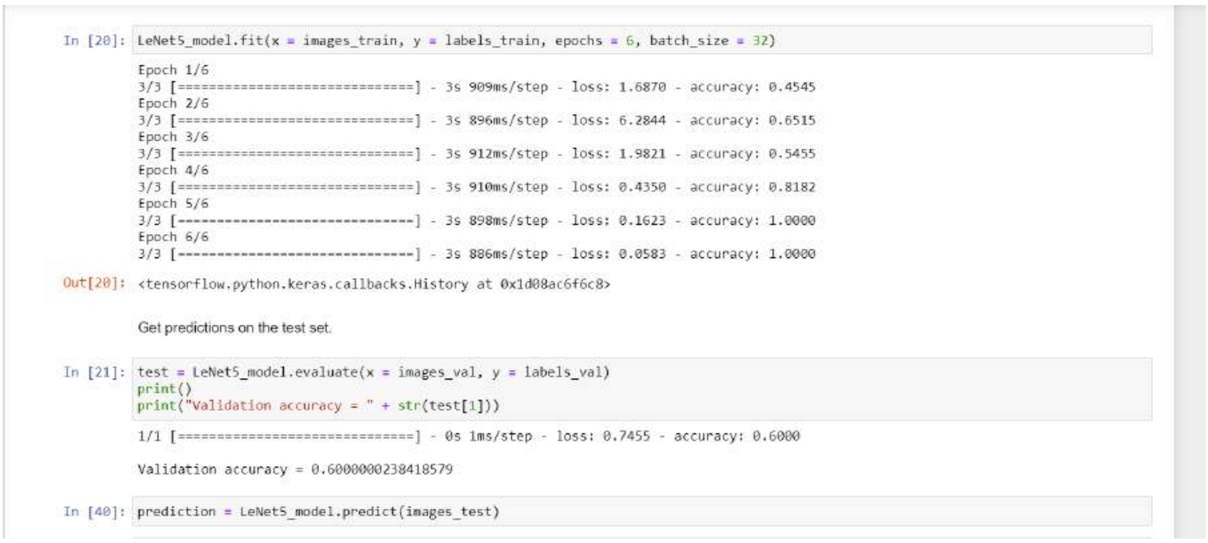
```
In [17]: def leNet5(input_shape):
             """
             Create an instance of model class representing LeNet-5
             inspired architecture

             Parameters
             ----------
             input_shape : tuple
                 shape to be input to the CNN
```

-
- Example of images (1 for cholesteric, 0 else)


jupyter CNN-Cholesteric_or_not_classification_small_dataset Last Checkpoint: Last Thursday at 9:51 PM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Markdown
0
50
100
150
200
250
300
350
0
100
200
300
400
500
600
0
50
100
150
200

```
In [5]: plt.imshow(images_train[9,:,:,:])
        print("Label: %i"%labels_train[9])

        Label: 1

In [6]: plt.imshow(images_test[3,:,:,:])
        print("Label: %i"%labels_test[3])

        Label: 0
```

-
- Results for small amount of data



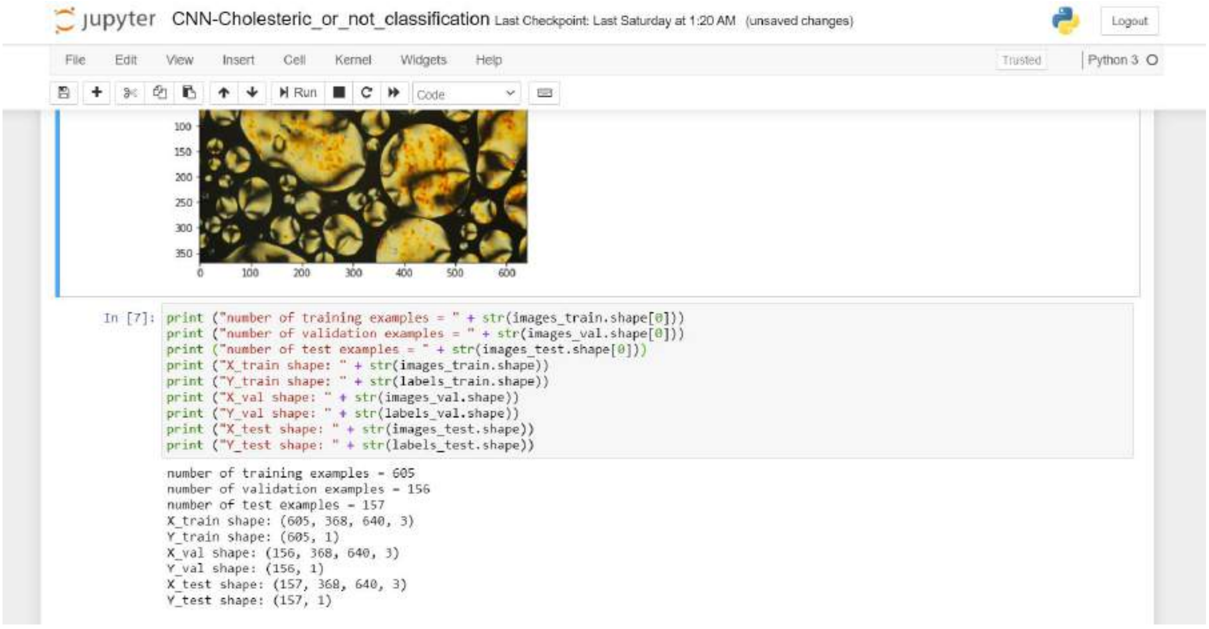
```
In [20]: LeNet5_model.fit(x = images_train, y = labels_train, epochs = 6, batch_size = 32)

         Epoch 1/6
         3/3 [==============================] - 3s 909ms/step - loss: 1.6870 - accuracy: 0.4545
         Epoch 2/6
         3/3 [==============================] - 3s 896ms/step - loss: 6.2844 - accuracy: 0.6515
         Epoch 3/6
         3/3 [==============================] - 3s 912ms/step - loss: 1.9821 - accuracy: 0.5455
         Epoch 4/6
         3/3 [==============================] - 3s 910ms/step - loss: 0.4350 - accuracy: 0.8182
         Epoch 5/6
         3/3 [==============================] - 3s 898ms/step - loss: 0.1623 - accuracy: 1.0000
         Epoch 6/6
         3/3 [==============================] - 3s 886ms/step - loss: 0.0583 - accuracy: 1.0000

Out[20]: <tensorflow.python.keras.callbacks.History at 0x1d08ac6f6c8>
```

Get predictions on the test set.

```
In [21]: test = LeNet5_model.evaluate(x = images_val, y = labels_val)
         print()
         print("Validation accuracy = " + str(test[1]))

         1/1 [==============================] - 0s 1ms/step - loss: 0.7455 - accuracy: 0.6000

         Validation accuracy = 0.6000000238418579

In [40]: prediction = LeNet5_model.predict(images_test)
```

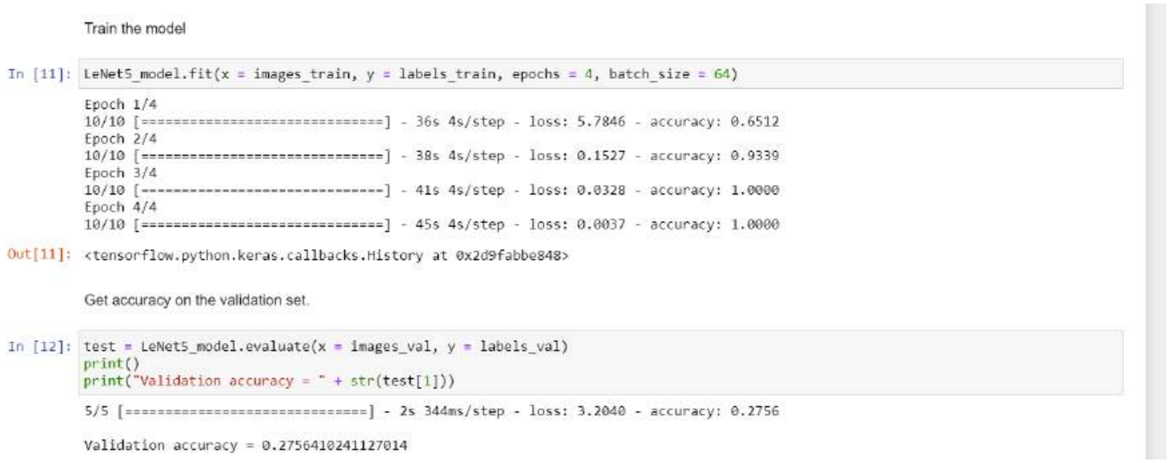
-

- Training and validation results on more amount of data (605 training examples and 157 validation)


jupyter CNN-Cholesteric_or_not_classification Last Checkpoint: Last Saturday at 1:20 AM (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
100
150
200
250
300
350
0
100
200
300
400
500
600

```
In [7]: print ("number of training examples = " + str(images_train.shape[0]))
        print ("number of validation examples = " + str(images_val.shape[0]))
        print ("number of test examples = " + str(images_test.shape[0]))
        print ("X_train shape: " + str(images_train.shape))
        print ("Y_train shape: " + str(labels_train.shape))
        print ("X_val shape: " + str(images_val.shape))
        print ("Y_val shape: " + str(labels_val.shape))
        print ("X_test shape: " + str(images_test.shape))
        print ("Y_test shape: " + str(labels_test.shape))

        number of training examples = 605
        number of validation examples = 156
        number of test examples = 157
        X_train shape: (605, 368, 640, 3)
        Y_train shape: (605, 1)
        X_val shape: (156, 368, 640, 3)
        Y_val shape: (156, 1)
        X_test shape: (157, 368, 640, 3)
        Y_test shape: (157, 1)
```

-
- Results



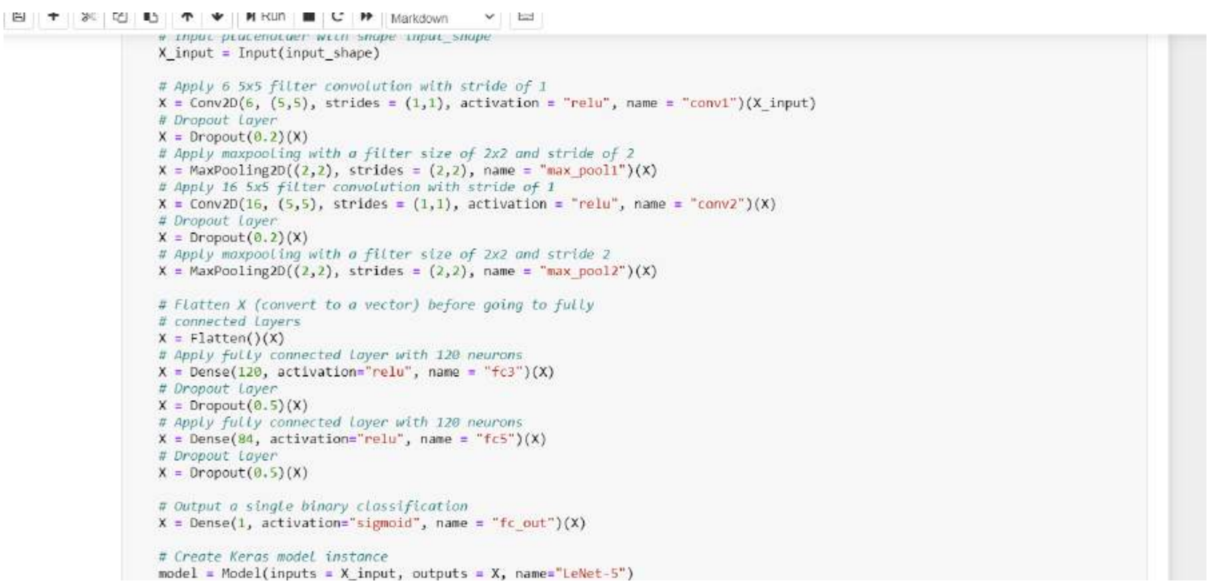
Train the model

```
In [11]: LeNet5_model.fit(x = images_train, y = labels_train, epochs = 4, batch_size = 64)

         Epoch 1/4
         10/10 [==============================] - 36s 4s/step - loss: 5.7846 - accuracy: 0.6512
         Epoch 2/4
         10/10 [==============================] - 38s 4s/step - loss: 0.1527 - accuracy: 0.9339
         Epoch 3/4
         10/10 [==============================] - 41s 4s/step - loss: 0.0328 - accuracy: 1.0000
         Epoch 4/4
         10/10 [==============================] - 45s 4s/step - loss: 0.0037 - accuracy: 1.0000
Out[11]: <tensorflow.python.keras.callbacks.History at 0x2d9fabbe848>
```

Get accuracy on the validation set

```
In [12]: test = LeNet5_model.evaluate(x = images_val, y = labels_val)
         print()
         print("Validation accuracy = " + str(test[1]))

         5/5 [==============================] - 2s 344ms/step - loss: 3.2040 - accuracy: 0.2756

         Validation accuracy = 0.2756410241127014
```

-
- Attempted to solve overfitting by using regularization techniques – in particular I used dropout layers, which randomly ignore a certain amount of input features to the next layer. I also trained for less time. But these did not improve results.



```
    # Input placeholder with shape input_shape
    X_input = Input(input_shape)

    # Apply 6 5x5 filter convolution with stride of 1
    X = Conv2D(6, (5,5), strides = (1,1), activation = "relu", name = "conv1")(X_input)
    # Dropout layer
    X = Dropout(0.2)(X)
    # Apply maxpooling with a filter size of 2x2 and stride of 2
    X = MaxPooling2D((2,2), strides = (2,2), name = "max_pool1")(X)
    # Apply 16 5x5 filter convolution with stride of 1
    X = Conv2D(16, (5,5), strides = (1,1), activation = "relu", name = "conv2")(X)
    # Dropout layer
    X = Dropout(0.2)(X)
    # Apply maxpooling with a filter size of 2x2 and stride 2
    X = MaxPooling2D((2,2), strides = (2,2), name = "max_pool2")(X)

    # Flatten X (convert to a vector) before going to fully
    # connected layers
    X = Flatten()(X)
    # Apply fully connected layer with 120 neurons
    X = Dense(120, activation="relu", name = "fc3")(X)
    # Dropout layer
    X = Dropout(0.5)(X)
    # Apply fully connected layer with 120 neurons
    X = Dense(84, activation="relu", name = "fc5")(X)
    # Dropout layer
    X = Dropout(0.5)(X)

    # Output a single binary classification
    X = Dense(1, activation="sigmoid", name = "fc_out")(X)

    # Create Keras model instance
    model = Model(inputs = X_input, outputs = X, name="LeNet-5")
```

-
- Results
-

Summary: An attempt was made to distinguish between cholesteric textures and other LC phases with not much success. An imbalanced dataset is the major problem I think, as I tried techniques to reduce overfitting which didn't improve results. More data is needed.

15/10/20 More images off of Instagram

Aim: Get more images to create a larger dataset and one which is more balanced – equal amounts of examples for each class

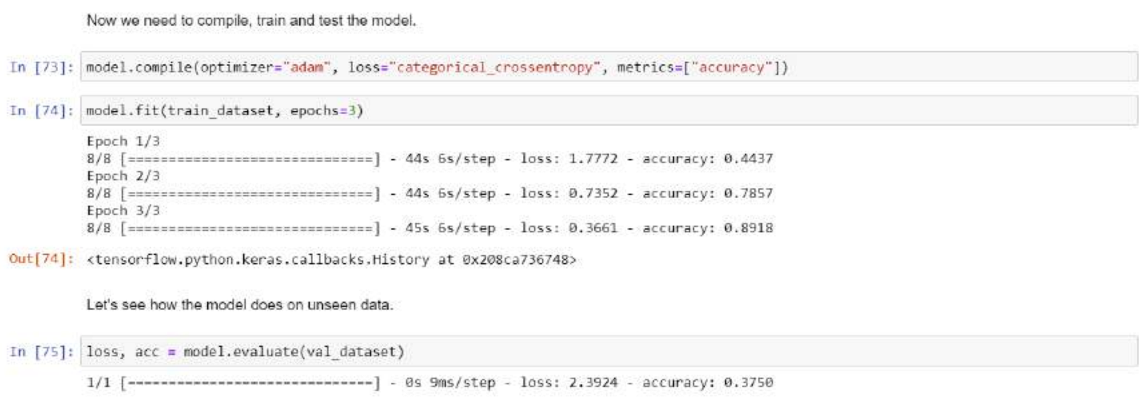
- Went through Vance Williams Instagram page which contain many videos and images of LC textures. Downloaded once that I thought could be useful for the project.

Summary: Got more data from Vance Williams Instagram – should help with training better models and creating a more balanced dataset

16/10/20 Multi-class classification of LC phases

Aim: Using new larger dataset apply a CNN to the problem of classifying many LC phases from their textures

- Josh Heaton advised that using flow_from_directory would help with the speed rather than importing all images in at once.
- Manually split data into validation and training sets so to avoid possible data leakage at ratio of 85% train, 15% validation. Also reshaped images to be 368x640 (same as images from youtube videos) as Instagram images were much larger.
- File available at: \Liquid_crystals-machine_learning\LiquidCrystalMachineLearning\Coding\ Multi-class_classification_of_LC_phases
- Results



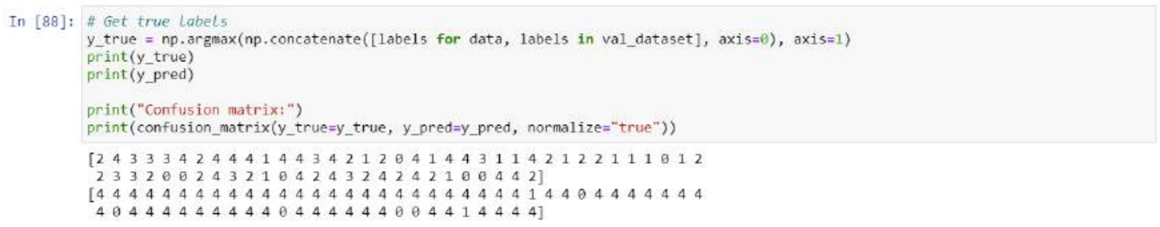
Now we need to compile, train and test the model.

```
In [73]: model.compile(optimizer="adam", loss="categorical_crossentropy", metrics=["accuracy"])

In [74]: model.fit(train_dataset, epochs=3)

         Epoch 1/3
         8/8 [==============================] - 44s 6s/step - loss: 1.7772 - accuracy: 0.4437
         Epoch 2/3
         8/8 [==============================] - 44s 6s/step - loss: 0.7352 - accuracy: 0.7857
         Epoch 3/3
         8/8 [==============================] - 45s 6s/step - loss: 0.3661 - accuracy: 0.8918

Out[74]: <tensorflow.python.keras.callbacks.History at 0x208ca736748>
```

Let's see how the model does on unseen data.

```
In [75]: loss, acc = model.evaluate(val_dataset)

         1/1 [==============================] - 0s 9ms/step - loss: 2.3924 - accuracy: 0.3750
```

-
- Performance is not necessarily better than random guessing, got predictions to see how it is guessing



```
In [88]: # Get true labels
         y_true = np.argmax(np.concatenate([labels for data, labels in val_dataset], axis=0), axis=1)
         print(y_true)
         print(y_pred)

         print("Confusion matrix:")
         print(confusion_matrix(y_true=y_true, y_pred=y_pred, normalize="true"))

         [2 4 3 3 3 4 2 4 4 4 1 4 4 3 4 2 1 2 0 4 1 4 4 3 1 1 4 2 1 2 2 1 1 1 0 1 2
          2 3 3 2 0 0 2 4 3 2 1 0 4 2 4 3 2 4 2 4 2 1 0 0 4 4 2]
         [4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 1 4 4 0 4 4 4 4 4 4 4
          4 0 4 4 4 4 4 4 4 4 4 0 4 4 4 4 4 4 0 0 4 4 1 4 4 4 4]
```

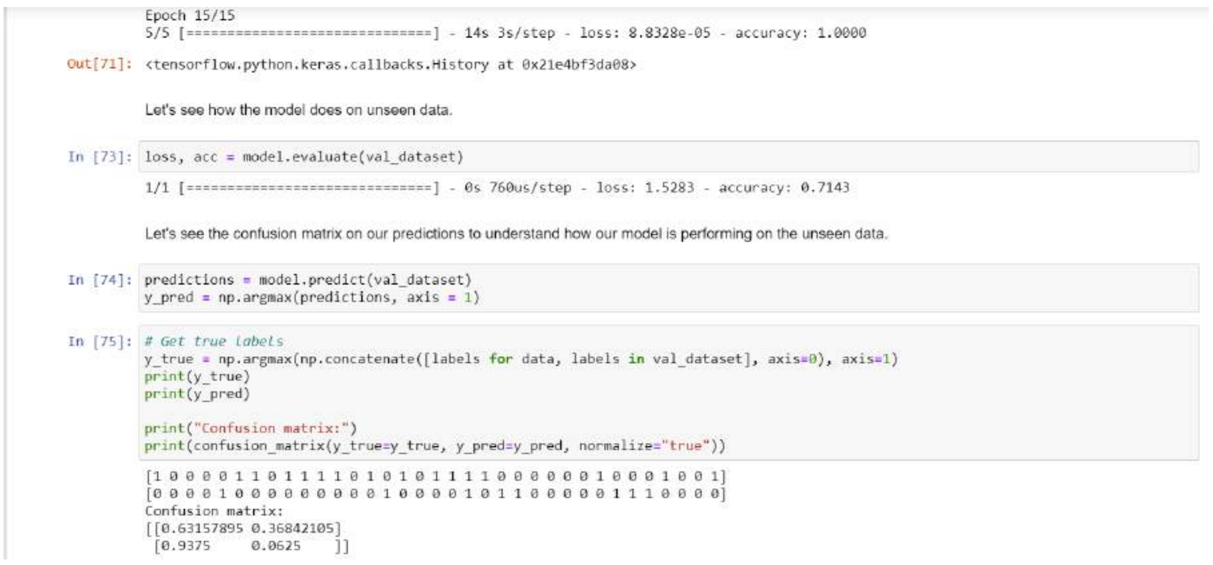
-
- As can be seen above, most guesses are 4 (representing cholesteric) hence I believe this problem is mainly due to imbalanced dataset (many more cholesteric images) as before

Summary: Multi class classification of LC phases from texture attempted with not much success (guessing almost all cholesteric) most likely due to an imbalanced dataset. To progress maybe try binary nematic or cholesteric classification as the are more nematic and cholesteric images than any others

17/10/20 Cholesteric or nematic classification

Aim: See if the problem of cholesteric or nematic has more success as the dataset is more balanced when only using images of these phases

- File available at: \Liquid_crystals-machine_learning\LiquidCrystalMachineLearning\Coding\Cholesteric_or_Nematic_classification_of_LC_phases
- Used only cholesteric and nematic images as these are the ones I have most images of.
- Results



```
Epoch 15/15
5/5 [==============================] - 14s 3s/step - loss: 8.8328e-05 - accuracy: 1.0000
```

```
Out[71]: <tensorflow.python.keras.callbacks.History at 0x21e4bf3da08>
```

Let's see how the model does on unseen data.

```
In [73]: loss, acc = model.evaluate(val_dataset)
```

```
1/1 [==============================] - 0s 760us/step - loss: 1.5283 - accuracy: 0.7143
```

Let's see the confusion matrix on our predictions to understand how our model is performing on the unseen data.

```
In [74]: predictions = model.predict(val_dataset)
         y_pred = np.argmax(predictions, axis = 1)
```

```
In [75]: # Get true labels
         y_true = np.argmax(np.concatenate([labels for data, labels in val_dataset], axis=0), axis=1)
         print(y_true)
         print(y_pred)

         print("Confusion matrix:")
         print(confusion_matrix(y_true=y_true, y_pred=y_pred, normalize="true"))
```

```
[1 0 0 0 0 1 1 0 1 1 1 1 0 1 0 1 0 1 1 1 1 0 0 0 0 0 0 1 0 0 0 1 0 0 1]
[0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 1 1 0 0 0 0 0 1 1 1 0 0 0 0]
Confusion matrix:
[[0.63157895 0.36842105]
 [0.9375     0.0625    ]]
```

-
- Confusion matrix above shows that it gets 63% of unseen cholesteric images correct, but 94% of unseen nematic images it guesses are cholesteric. We are still suffering from imbalanced data.

Summary: Results from nematic or cholesteric classification show that the model learns to almost always guess cholesteric suggest we still have an imbalanced dataset. To improve more images of other phase should be collected.

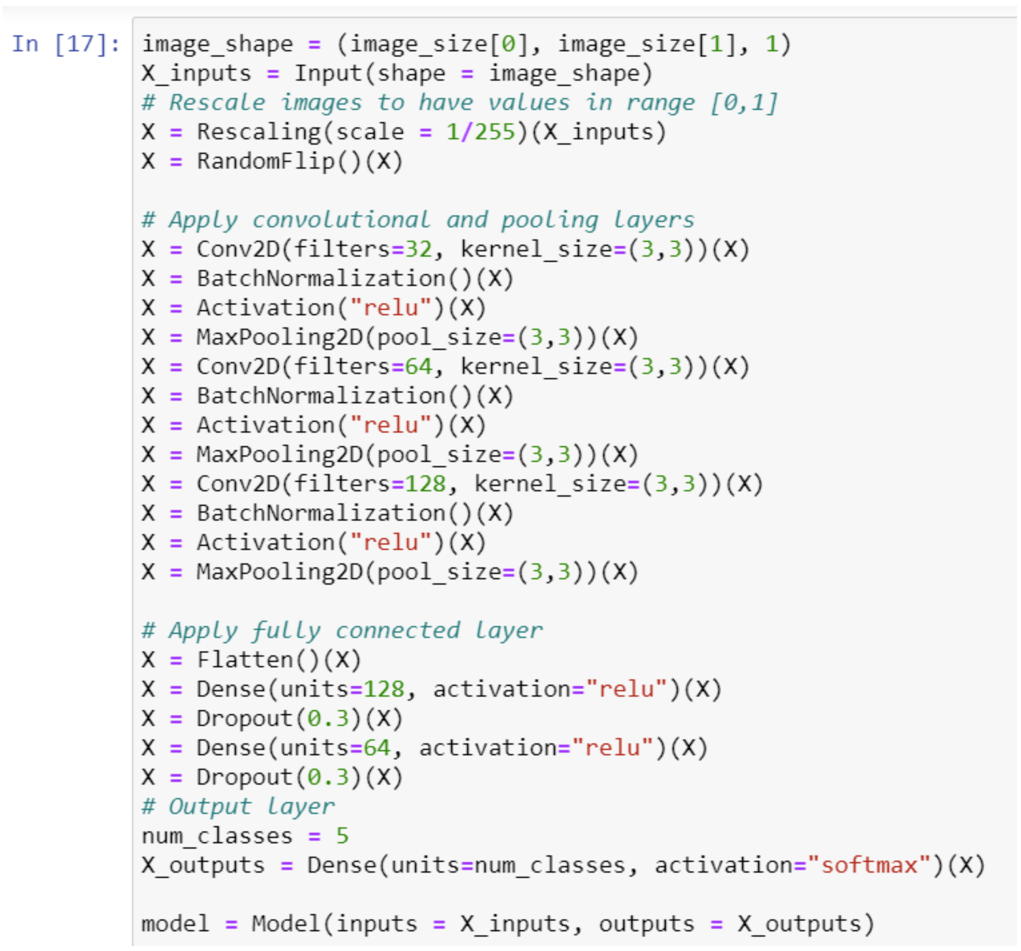
18/10/20 More data brought together and shared by Josh Heaton

- Josh Heaton shared a drive link to files containing images for nematic, cholesteric, twist-grain-boundary, smectic and columnar. Around 2000 images – possibly still a bit imbalanced but much larger than before.
- He also shared his group's github page so we are now able to help out on code.
- His group has been resizing images to 200x200 and so I will do the same for consistency. The size is small enough to allow quicker computation but not so small that valuable information is lost.
- They also suggested working in grayscale rather than in rgb colour. This makes sense as texture rather than colour is the important feature we're trying to detect.

19/10/20 Using new dataset for multi-class classification

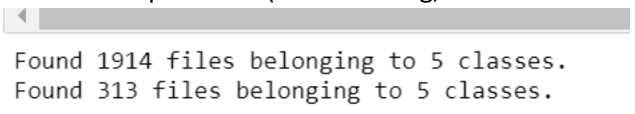
Aim: Use new larger dataset to revisit multi-class classification

- File found at: \Liquid_crystals-machine_learning\LiquidCrystalMachineLearning\Coding\Phase_classification_grayscale_images_from_drive
- Model used



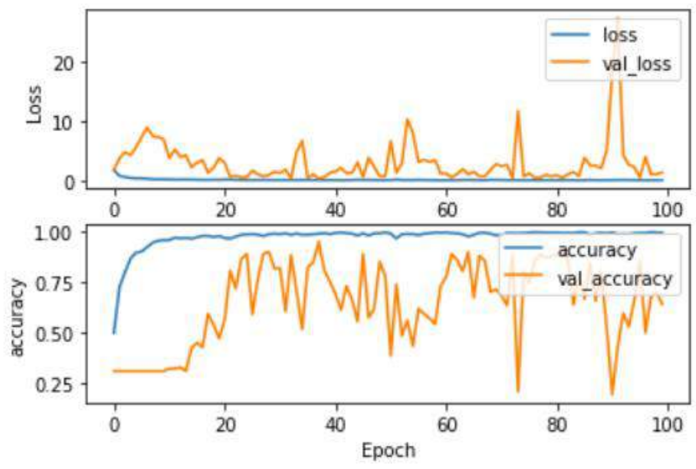
```
In [17]: image_shape = (image_size[0], image_size[1], 1)
         X_inputs = Input(shape = image_shape)
         # Rescale images to have values in range [0,1]
         X = Rescaling(scale = 1/255)(X_inputs)
         X = RandomFlip()(X)

         # Apply convolutional and pooling layers
         X = Conv2D(filters=32, kernel_size=(3,3))(X)
         X = BatchNormalization()(X)
         X = Activation("relu")(X)
         X = MaxPooling2D(pool_size=(3,3))(X)
         X = Conv2D(filters=64, kernel_size=(3,3))(X)
         X = BatchNormalization()(X)
         X = Activation("relu")(X)
         X = MaxPooling2D(pool_size=(3,3))(X)
         X = Conv2D(filters=128, kernel_size=(3,3))(X)
         X = BatchNormalization()(X)
         X = Activation("relu")(X)
         X = MaxPooling2D(pool_size=(3,3))(X)

         # Apply fully connected layer
         X = Flatten()(X)
         X = Dense(units=128, activation="relu")(X)
         X = Dropout(0.3)(X)
         X = Dense(units=64, activation="relu")(X)
         X = Dropout(0.3)(X)
         # Output layer
         num_classes = 5
         X_outputs = Dense(units=num_classes, activation="softmax")(X)

         model = Model(inputs = X_inputs, outputs = X_outputs)
```

-
- Number of examples used (1914 training, 313 validation



```
Found 1914 files belonging to 5 classes.
Found 313 files belonging to 5 classes.
```

-
- Results (graph plotted using Josh Heaton code)



-
- Peak validation accuracy achieved was 90.1% on epoch 65 (did not have model saving on so will have to redo and save at epoch of best validation accuracy)
- Other group have achieved validation accuracies of the order 99%. However I suggested to them that there may be data leakage due to them randomly splitting all images among training and validation sets, whereas here I selected training and validation sets to not have images from the same videos.

Summary: A CNN architecture was used to achieve 90.1% on unseen validation LC textures when classifying among nematic, cholesteric, twist-grain-boundary, smectic and columnar phases. Will need to repeat this result with model saving on in order to reuse from best trained model (around epoch 65).

Week 2 summary:

- Progress was made onto more complex classification problems including cholesteric vs nematic and multi-class using cholesteric, columnar, nematic, smectic, twist grain boundary – using the new dataset put together by Josh Heaton.
- Images, videos and code was share among the two groups via GitHub and google drive.
- Possibility of further looking unsupervised machine learning techniques to compare to the performance of cnn in phase classification problem.
- Achieved a highest accuracy of 90.1% on unseen texture images – but will have to repeat in order to get a saved model. Will also need to test this model to see whether it is truly as good as it seems.

21/10/20 Third meeting (video call)

Attendees: Dr. Dierking, Josh Heaton, James Harbon, Jason Dominguez

Discussed:

- The other group mentioned accuracies at 99-100% on validation data. However I pointed out that there may be data leakage due to their images being randomly split from same videos into training and validation data. I said I will try see if this is the case by training my same model using the kind of random split they used.
- Ask Dr Dierking to possibly find more videos/images and to ask PhD student to get videos from the lab.
- Dr Dierking said that maybe columnar and twist grain boundary are too complex to do at this stage and that maybe distinguishing between what type of smectic would be a good progression.

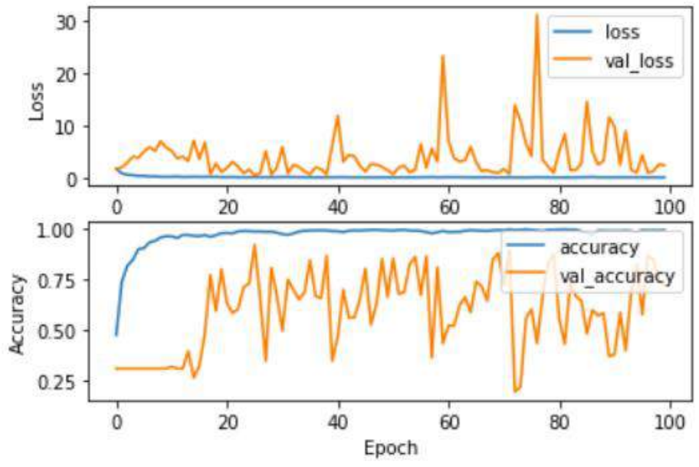
Aims/objectives for week 3

- Retrain my last model but saving at it's best point. Then use this to see how well it is actually performing.
- See if data leakage was the reason for other group achieving 99-100% accuracy on validation data.
- Looking into training a model to differentiate between just Smectics
- Look into other techniques (transfer learning, unsupervised) to compare to this technique

21/10/20 Retraining model on multiclassification of phases and testing on new images to see how well it generalizes

Aim: Get the best performance out of my current model (based on validation accuracy) saved and use this on new images obtained from internet to see how well it generalizes.

- Screenshotted images from Dr Dierking's book Textures of Liquid Crystals to be used as test data.
- Files found at:
  \Liquid_crystals-machine_learning\LiquidCrystalMachineLearning\Coding\
  Phase_classification_grayscale_images_from_drive-saving_model
  \Liquid_crystals-machine_learning\LiquidCrystalMachineLearning\Coding\
  Phase_classification_grayscale_images_from_drive-predictions_on_test_from_saved_model
- Results of retraining the model – to be saved at epoch of best validation accuracy



-
- Peak validation accuracy is at epoch 26 with 92.33% accuracy

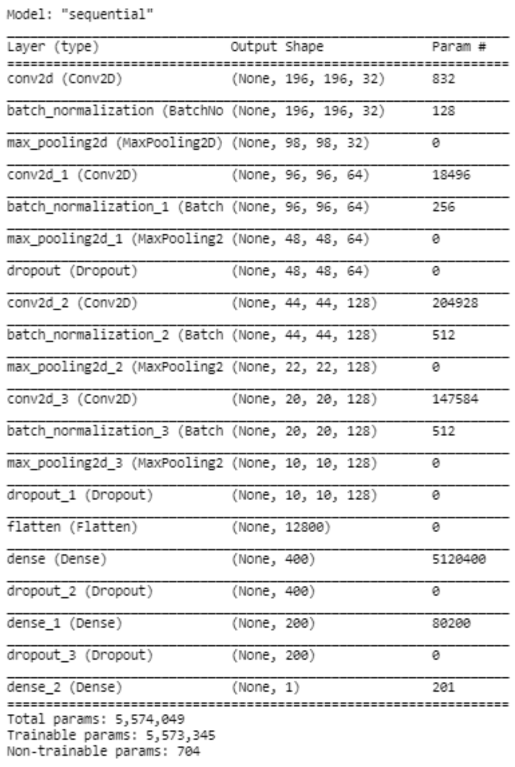
Summary: When getting predictions on test images taken from Textures of Liquid crystals by Ingo Dierking we got mixed results. This suggests possible overfitting to the validation images (as performance was high on both validation images and training images) and that our data maybe doesn't represent the general distribution of texture images. For example, as our entire dataset is on the small side ~2000 and many images are from the same movie, it's possible we are missing out many different looking textures for a particular phase. Also noticed many more nematic and cholesteric predictions than other – this is due to an imbalanced dataset. We have significantly less images of Columnar, Twist grain boundary and Smectic phases than Nematic and Cholesteric.

23/10/20 Focusing on nematic vs cholesteric classification using a balanced dataset

Aim: Construct a balanced dataset consisting of only nematic and cholesteric phase textures to work on nematic vs cholesteric classification

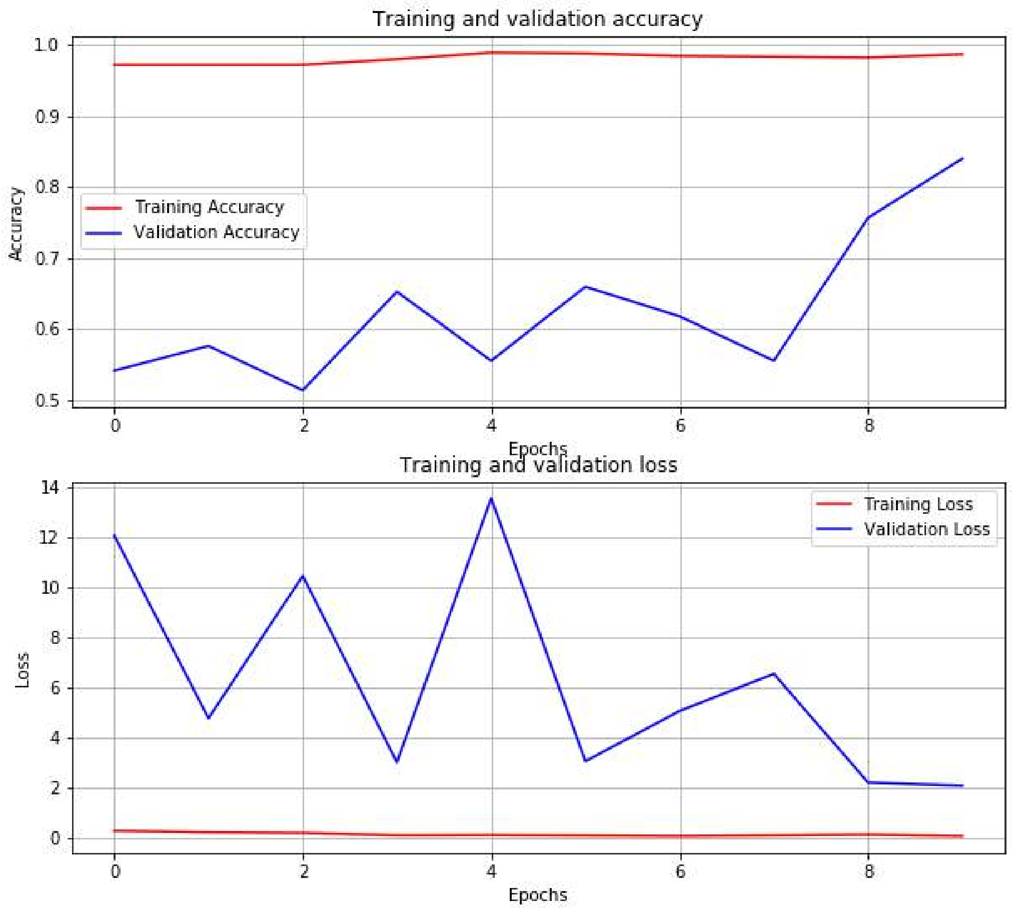
- Manually split data into train, validation and test sets at ratio of ~7 : 1.5 : 1.5 (Train: 403 cholesteric, 467 nematic; validation: 71 cholesteric, 73 nematic; test: 65 cholesteric, 72 nematic).
- As before, images from videos which are in training dataset are not used in validation or test data, to avoid data leakage giving high validation and/or test accuracy.

- Also included the images used as test images in previous experiment (from texture of liquid crystals by Ingo Dierking) as these seemed more difficult for the model last time and so if they're split among validation and test sets then maybe they'll lead to our accuracy results giving a better indication of whether the model will generalize well.
- Files found at:
  \Liquid_crystals-machine_learning\LiquidCrystalMachineLearning\Cholesteric-Nematic_classification\Cholesteric-Nematic-CNN_model_saved2.ipynb
  \Liquid_crystals-machine_learning\LiquidCrystalMachineLearning\Cholesteric-Nematic_classification\Cholesteric-Nematic-CNN_model_saved3.ipynb
- Trained model saved at:
  C:\Users\Jason\Documents\University\Year_4\MPhys_Project(s)\Cholesteric-Nematic-CNN_model-saved2
  C:\Users\Jason\Documents\University\Year_4\MPhys_Project(s)\Cholesteric-Nematic-CNN_model-saved3
- Image dataset used found at:
  \Liquid_crystals-machine_learning\Nematic-Cholesteric-balanced-24-10-20
- Model architecture used

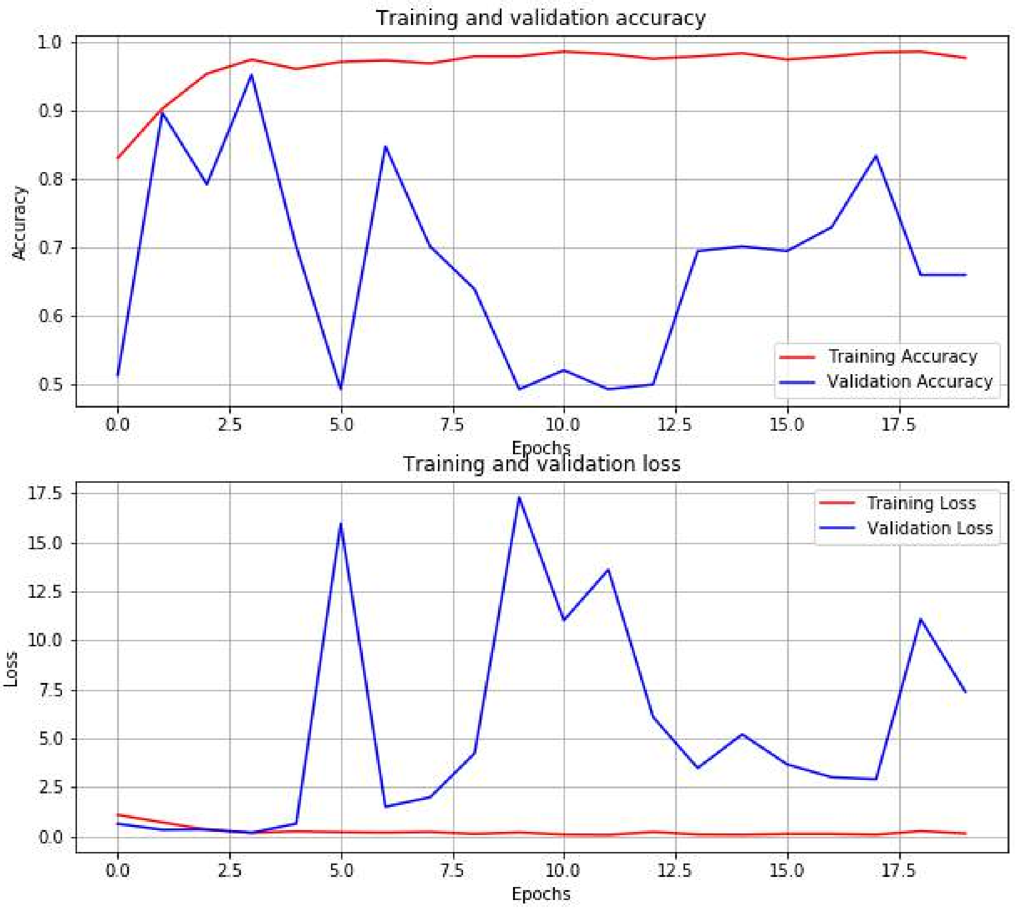


```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 196, 196, 32)      832
_________________________________________________________________
batch_normalization (BatchNo (None, 196, 196, 32)      128
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 98, 98, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 96, 96, 64)        18496
_________________________________________________________________
batch_normalization_1 (Batch (None, 96, 96, 64)        256
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 48, 48, 64)        0
_________________________________________________________________
dropout (Dropout)            (None, 48, 48, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 44, 44, 128)       204928
_________________________________________________________________
batch_normalization_2 (Batch (None, 44, 44, 128)       512
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 22, 22, 128)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 20, 20, 128)       147584
_________________________________________________________________
batch_normalization_3 (Batch (None, 20, 20, 128)       512
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 10, 10, 128)       0
_________________________________________________________________
dropout_1 (Dropout)          (None, 10, 10, 128)       0
_________________________________________________________________
flatten (Flatten)            (None, 12800)             0
_________________________________________________________________
dense (Dense)                (None, 400)               5120400
_________________________________________________________________
dropout_2 (Dropout)          (None, 400)               0
_________________________________________________________________
dense_1 (Dense)              (None, 200)               80200
_________________________________________________________________
dropout_3 (Dropout)          (None, 200)               0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 201
=================================================================
Total params: 5,574,049
Trainable params: 5,573,345
Non-trainable params: 704
_________________________________________________________________
```

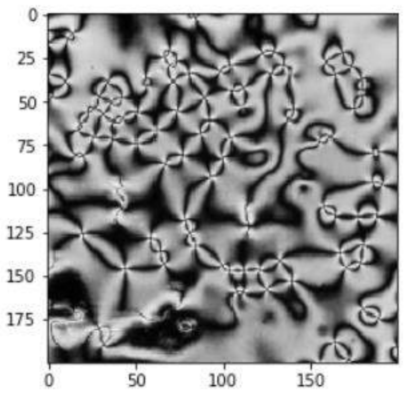
- Image augmentation was applied during training, rotation range 45deg, width and height shift range of 0.2, zoom range of 0.2 and horizontal and vertical flipping. This was to help reduce overfitting due to the small dataset and help the model generalize. Dropout layers were another measure used to try avoid overfitting.
- Results from training for 10 epochs, model was saved at 10th epoch with 84.03% validation accuracy and 98.3% training accuracy



-
- Accuracy on test data was 84.7%
- Also trained another model (same architecture) for more epochs
- Results



-
- Results from training for 20 epochs, model was saved at 4th epoch with 95.1% validation accuracy and 97.4% training accuracy
- Accuracy on test data was 95.6%
- Example of prediction on image outside of train, val and test data



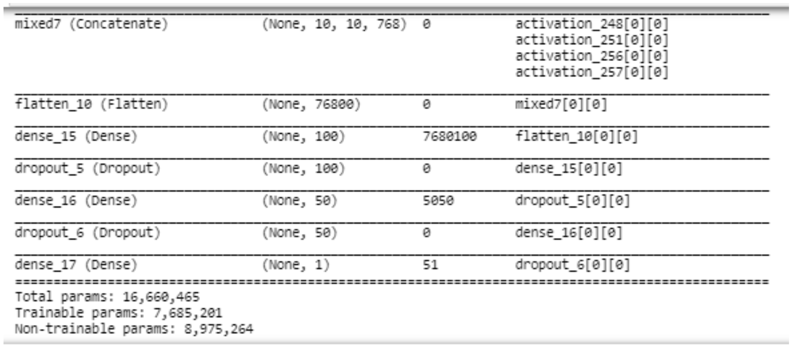
```
1/1 [==============================] - 0s 0s/step
Predicted phase:
nematic
Confidence level:
1.0
```

-

Summary: A CNN model was trained and saved with a training accuracy of 97.4%, validation accuracy of 95.1% and test accuracy of 95.6%. This seems like a very good result. Of course with a dataset this small it is always difficult to rely on these accuracy readings as an indication of whether this model will generalize well to all liquid crystal texture for cholesteric and nematic phases. However, precautions were taken to ensure the validation and test accuracy we as minimally misleading as possible – avoiding training data leakage by ensuring images from each video is not split among training and val, test sets; also larger val and test sets were used compared to before, to get more of a spread of different texture images that are unseen during training.

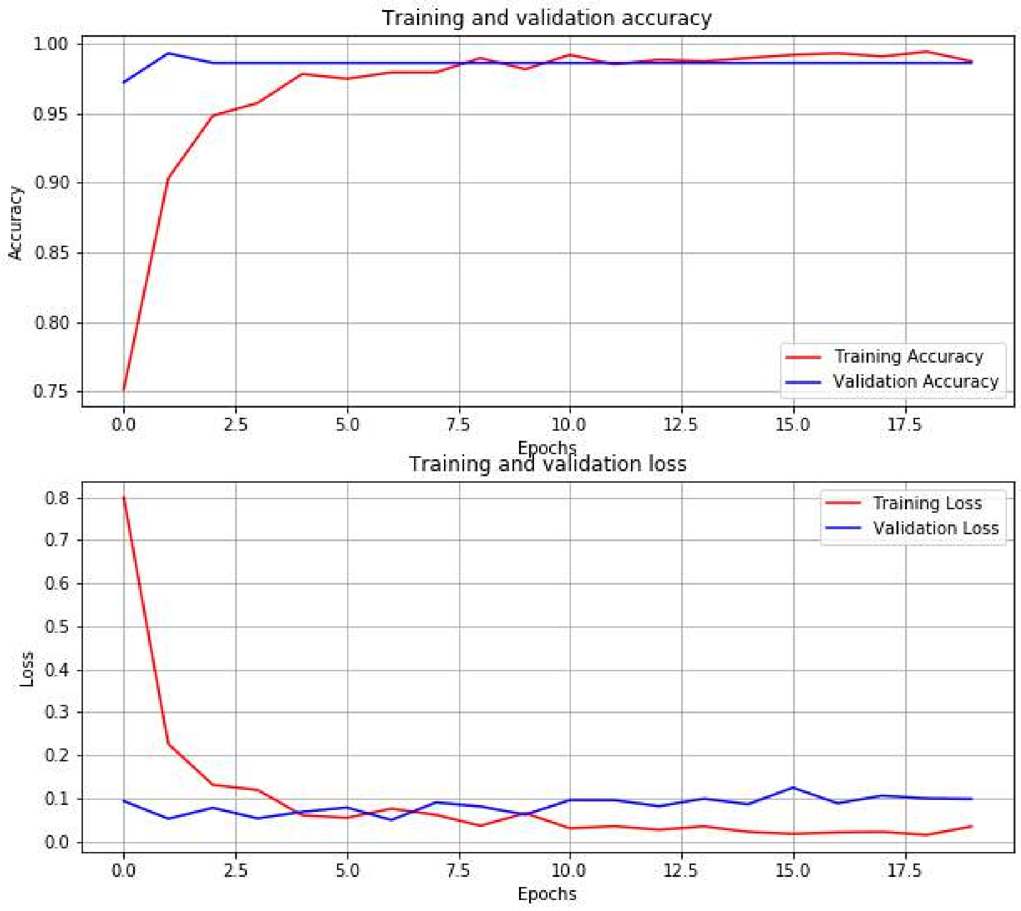
24/10/20 Transfer learning for nematic vs cholesteric classification

Aim: As we have a relatively small dataset I want to try using transfer learning to see if we can improve results on the above experiment

- Data preparation was the same as the above experiment
- For transfer learning I used the InceptionV3 network using pre-trained weights from training on the ImageNet dataset (keras has this built in).
- 

```
mixed7 (Concatenate)            (None, 10, 10, 768)  0           activation_248[0][0]
                                                                 activation_251[0][0]
                                                                 activation_256[0][0]
                                                                 activation_257[0][0]
__________________________________________________________________________________________________
flatten_10 (Flatten)            (None, 76800)        0           mixed7[0][0]
__________________________________________________________________________________________________
dense_15 (Dense)                (None, 100)          7680100     flatten_10[0][0]
__________________________________________________________________________________________________
dropout_5 (Dropout)             (None, 100)          0           dense_15[0][0]
__________________________________________________________________________________________________
dense_16 (Dense)                (None, 50)           5050        dropout_5[0][0]
__________________________________________________________________________________________________
dropout_6 (Dropout)             (None, 50)           0           dense_16[0][0]
__________________________________________________________________________________________________
dense_17 (Dense)                (None, 1)            51          dropout_6[0][0]
==================================================================================================
Total params: 16,660,465
Trainable params: 7,685,201
Non-trainable params: 8,975,264
```

- I cut off the Inception V3 net at layer “mixed7” and added my own trainable fully-connected layers as seen above.
- File found at: \Liquid_crystals-machine_learning\LiquidCrystalMachineLearning\Cholesteric-Nematic_classification\Cholesteric-Nematic-transfer_learning.ipynb
- Best model saved in: \Cholesteric-Nematic-transfer_learning-saved
- Results



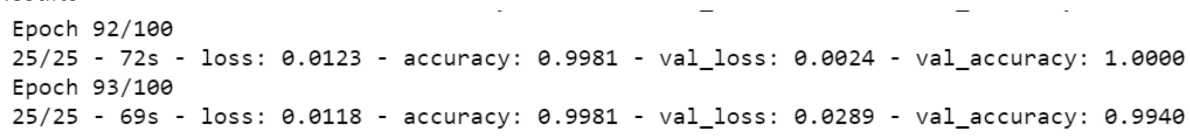
- Curves are a lot smoother and rest a steady high accuracy on validation unlike previous results which fluctuated wildly.
- Achieved highest validation accuracy of 99.3% with 0.0521 at epoch 2 (training accuracy is lower than validation due to image augmentation used during training) – training accuracy and loss was 90.3% and 0.2257 respectively.
- On test set (using best model from epoch 2) achieved an accuracy of 98.5% and a loss of 0.0739.

Summary: Transfer learning, using some of inceptionV3 network with pre-trained weights, was used on top of a shallow trainable fully connect network in order to achieve higher and smoother validation accuracy results than the previous classification experiment without it. Highest validation accuracy and loss was 99.3% and 0.0521 at epoch 2 and on test images 98.5% accuracy and 0.0739 loss was achieved.

25/10/20 Testing data leakage by using other group's approach of randomly selecting training and validation images from same group of images

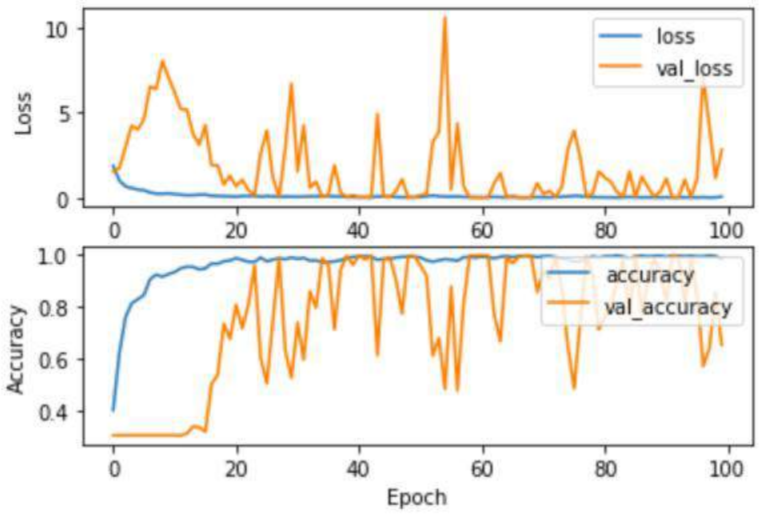
Aim: The other group were able to achieve very high validation accuracy 99-100% for multi-class classification problem I was doing on 19/10/20 and 20/10/20. Their approach for selecting training and validation dataset was to randomly take images from the same set – I said this may have caused data leakage and therefore unreliable high validation accuracy as images from the same video were spread among training and validation image sets. So this experiment aimed to use the same set up as in 20/10/20 but using their approach to see if I also achieved very high accuracies.

- Set up was the same as for 20/10/20 except images were taken randomly from a single file for training and test with 30% validation split.
- Results



```
Epoch 92/100
25/25 - 72s - loss: 0.0123 - accuracy: 0.9981 - val_loss: 0.0024 - val_accuracy: 1.0000
Epoch 93/100
25/25 - 69s - loss: 0.0118 - accuracy: 0.9981 - val_loss: 0.0289 - val_accuracy: 0.9940
```

- Results of 100% were achieved on epoch 92



- File found at: \Liquid_crystals-machine_learning\LiquidCrystalMachineLearning\Coding\Phase_classification_grayscale_images_from_drive-testing_data_leakage.ipynb

Summary: Randomly selecting images for training and test sets was done to see if it was a source of data leakage. As we can see results on validation were able to achieve higher values than when taking images that were split manually (to avoid images from the same video being in both training and validation sets). On one occasion we even had 100% validation accuracy, this shows an indication that there is data leakage present, giving misleadingly high validation accuracy.

26/10/20 Dr Dierking sent more videos

- Dr Dierking sent over more LC texture videos, saved at: \Liquid_crystals-machine_learning\Images_from_Ingo_26-10-20

- Will be able to obtain more images from frame grabbing these videos, but I will need clarification on what phases these videos correspond to.
- Also, we now maybe have a sufficient number of videos with which we can attempt so phase transition recognition from videos (is there a phase change or not) using RNNs, LSTMs, etc for time sequence deep learning.
- Now how some more Smectic images which can be used in a smectic classifier (SmA or SmC or SmF)

Week 3 summary:

- Tested data leakage source in the approach of taking images randomly for training and validation datasets – achieved higher (100% at one epoch) validation accuracy suggesting data leakage was present.
- Got more videos from Dr Dierking which can be use for video recognition and more image for better image classification.
- Applied transfer learning and non-transfer learned CNNs to nematic or cholesteric classification using a balanced dataset.
- Achieved a highest accuracy of 90.1% on unseen texture images – but will have to repeat in order to get a saved model. Will also need to test this model to see whether it is truly as good as it seems.
- Not yet looking into unsupervised approaches.
- Not yet worked on a smectic classifier, due to limited data, but thanks to new videos from Dr Dierking this could now be possible.

28/10/20 Fourth meeting (video call)

Attendees: Dr. Dierking, Josh Heaton, James Harbon, Jason Dominguez

Discussed:

- Dr Dierking mentioned that representation of how well these models work is key, people can't just take our word for it. We discussed using confusion matrix plot, train/val accuracy against training epoch plot and possibly more plots where we change some parameter (say amount of training data, etc.) and see how accuracy is affected.
- Also said it would be good to see how accuracy it is on each phase for example, this can be done with confusion plot.
- Also mentioned how it would be good to get a record of how long training and prediction took ask this will be useful to know for people trying to reproduce results. How long is necessary to get something that works well for example.
- Discussed that we will need to spend most of this week frame grabbing from new videos and, as they are large size, possibly splitting them up into many 200x200 or so images
- Dr Dierking will send over the paper necessary for labels the phases we see in the videos

- Dr Dierking also mentioned how the PhD student may be able to get more videos on Friday 30/10/20.

Aims/objectives for week 4

- Need to use frame grabber on videos obtained from Dr Dierking on 26/10/20
- Need to split images from videos into 200x200 images (200x200 has seemed to be a good choice so far in terms of computation time and accuracy, but will test this at some point)
- With these images need to label them according to what phase they are – details about what phases we expect to see at what temperatures is in this paper:
  "Mesomorphic properties of a homologous series of chiral liquid crystals containing the alpha-chloroester group" by J.Schacht et al.
- Provided this gets done (could take a while) first I want to use some of these images as completely unseen images to test my current nematic vs cholesteric models against and get a confusion matrix plot
- Further more I will try to develop a model for another multiclass classification model, but this will depend on the amounts of data for each phase that we have available. I will sort all possible images we have into files and see what approach is best.

28/10/20 Grabbing frames

- I grabbed frames for videos sent by Dr Dierking – the M series
- I sorted them initially into phases, using this paper
  Mesomorphic properties of a homologous series of chiral liquid crystals containing the α-chloroester group by J. Schacht et al.
- Had some trouble noticing some phases so shared my sorted images to the other group (Josh Heaton and James Harbon) for final sorting and splitting of larger images into smaller images.
- I mentioned that it is best to keep images corresponding to one molecule together as we don't want to split them among training and validation sets, to avoid data leakage.

30/11/20 New videos

- Received another set of videos from Dr Dierking – D liquid crystals.
- Grabbed frames and sorted into phases as above, using this paper
  Properties of higher-ordered ferroelectric liquid crystal phases of a homologous series by I. Dierking at al.
- Again shared these with the other group to sort anything I couldn't and to split. They mentioned they would split into 3 images width wise and 2 images depth wise, so as to not create images that are too zoomed in and blurry. They also said they would reshape and crop into 256 as this may be a better size for networks as it's a power of 2. I mentioned that I needed colour images for my transfer learning network.

03/11/20 Applying CNN trained for image classification for video classification frame-by-frame

Aim: Based on a technique learned from https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/ apply a CNN trained for image phase classification to frame-by-frame video classification

- Trained an inception v3 based transfer learning CNN network on a quick dataset of a couple hundred images for phases isotropic, nematic, cholesteric and smectic.
- Was only doing this quickly to test the idea and the accuracies on each phase were mixed but the applied this to a video frame by frame. Due to poorly trained network, the label of the video wasn't correct for each frame (i.e. mistaking nematic for isotropic) however it did show a change of classification upon a phase change, despite being wrong. Video saving capability wasn't quite working so don't have this video saved, but I will work on that code. Currently just display the video.
- File available at \Liquid_crystals-machine_learning\LiquidCrystalMachineLearning\Coding\Video_recognition_proof_of_concept

Summary: Applied a quickly trained transfer learned CNN to video phase classification frame by frame using OpenCV for python. Was unable to save video and the network wasn't classifying correctly – due to small dataset to train on. This as just to test the idea. I will advance on this by trying to get the video saving working and also by training a cnn on the new large dataset to get a better multi class phase classifier.

Week 4 summary

- Most of the week was spent grabbing frames and sorting images into categories.
- Did try an initial test of applying a cnn to video frame-by-frame classification – was only a test of whether the idea could work, so will advance on it by using a network trained on a larger amount of images to get better classification. Also need to work on how to save the videos.

03/11/20 Fifth meeting (video call)

Attendees: Dr. Dierking, Josh Heaton, James Harbon, Jason Dominguez

Discussed:

- Not too much was discussed, we mention how we spent the week sorting through images, other group mentioned how they were looking into simulating textures and using GANs.
- I mentioned my brief experiment this morning (see above) with video classification.

- Dr Dierking again mentioned the importance of getting visual representation of the model, how different parameters affect performance etc. so as to justify what choices we made to people reading about it.
- Dr Dierking also mentioned that it's best to start with the bigger picture questions and then move to more specific ones, i.e. LC or not, then isotropic, nematic or cholesteric or smectic, then maybe distinguishing between less and more ordered smectics. The final ultimate test for a network would be being able to distinguish a twist grain boundary texture as this is very hard for people.

Aims/Objectives for week 5:

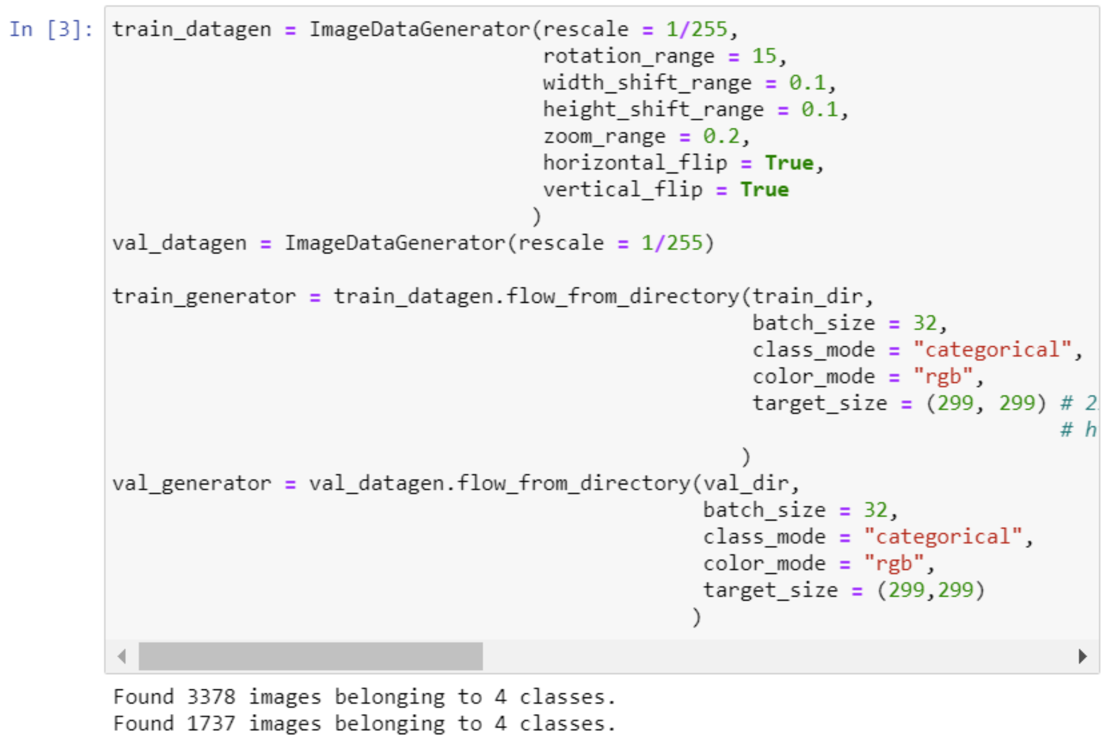
- Work on the video classification attempted at the end of last week by using new data – larger dataset – to get better classification among phases and try to get video saving to work. Start with Isotropic, Nematic, Smectic, Cholesteric. If struggling to get good results try just liquid crystal or isotropic.
- Maybe get started on other techniques for classification to compare to cnn, such as k nearest neighbours (knn), support vector machine (SVM) and multi-layer-perceptron (MLP) to see what performs best
- Work on plotting graphs etc to show hyperparameter tuning of network to justify choices made in getting the best network

05/11/20 Training a CNN for four class phase classification

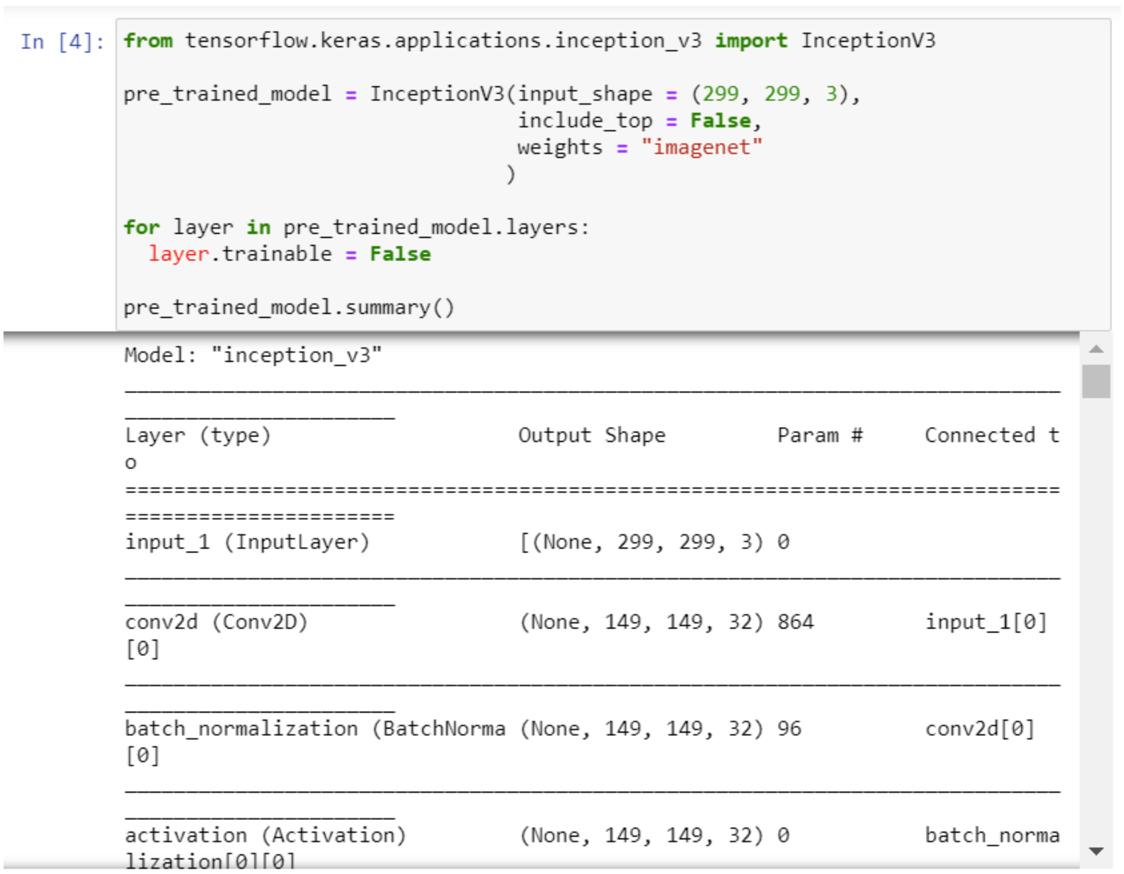
- Before working with the new full dataset, I wanted to work with smaller dataset of images that I could assure had no data leakage. I made sure of this by selecting which videos I wanted in my test set of videos (these were chosen to be two nematic videos from Dr Dierking (to test isotropic-nematic transitions), videos for M6 molecule (which showed various isotropic, smectic and cholesteric transitions), and videos for the 8CB molecule (which shows various nematic, smectic, isotropic transitions).
- Images from these chosen videos were not included in training or validation image datasets.
- Also, images from a single molecule regardless of video were not split among training and test.
- In fact, apart from isotropic images (which weren't considered to be too problematic with regards to splitting among training and test as they are all just very dark images) images obtained off of the internet from Vance Williams were used for validation and images obtained from Dr Dierking videos were used for training
- Images were deleted from some phases to balance the datasets (deleting was done by doing every other frame for images from a video that produced a large amount of similar images)
- Final splits were:

	Train	Val	Total
Isotropic	666 (20%)	458	
Nematic	562 (17%)	214	
Cholesteric	1151 (34%)	531	
Smectic	999 (30%)	534	
Total	3378 (66%)	1737 (34%)	5115

- To make up for lower numbers of nematic images in training, took some nematic images from Vance Williams images, but still no data leakage occurred.
- These datasets are stored at \Liquid_crystals-machine_learning\LiquidCrystalMachineLearning\(I,N,Chol,Sm)Images And \Liquid_crystals-machine_learning\LiquidCrystalMachineLearning\(I,N,Chol,Sm)Test_videos
- Trained two models: transfer learning model using inception v3 net, and a self made cnn
- For transfer learning this was the augmentation, model, optimizer and results (images were reshaped to (299,299,3) rgb as this is what inception v3 was designed for

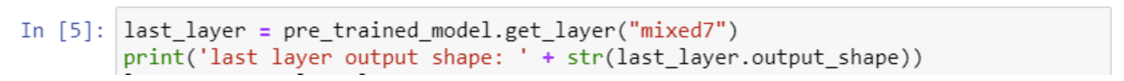


```
In [3]: train_datagen = ImageDataGenerator(rescale = 1/255,
                                           rotation_range = 15,
                                           width_shift_range = 0.1,
                                           height_shift_range = 0.1,
                                           zoom_range = 0.2,
                                           horizontal_flip = True,
                                           vertical_flip = True
                                          )
        val_datagen = ImageDataGenerator(rescale = 1/255)

        train_generator = train_datagen.flow_from_directory(train_dir,
                                                            batch_size = 32,
                                                            class_mode = "categorical",
                                                            color_mode = "rgb",
                                                            target_size = (299, 299) # 2
                                                                                     # h
                                                           )
        val_generator = val_datagen.flow_from_directory(val_dir,
                                                        batch_size = 32,
                                                        class_mode = "categorical",
                                                        color_mode = "rgb",
                                                        target_size = (299,299)
                                                       )

        Found 3378 images belonging to 4 classes.
        Found 1737 images belonging to 4 classes.
```



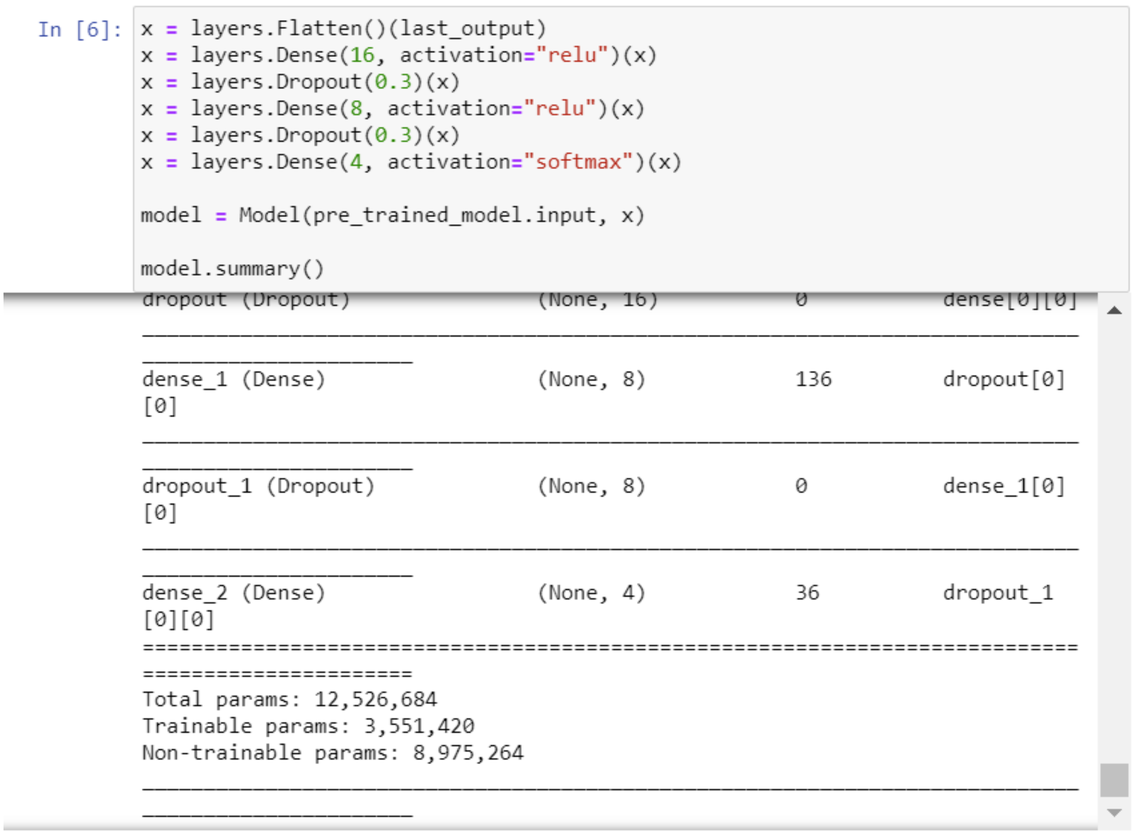
```
In [4]: from tensorflow.keras.applications.inception_v3 import InceptionV3

        pre_trained_model = InceptionV3(input_shape = (299, 299, 3),
                                        include_top = False,
                                        weights = "imagenet"
                                       )

        for layer in pre_trained_model.layers:
          layer.trainable = False

        pre_trained_model.summary()
```

```
Model: "inception_v3"
____________________________________________________________________________
______________________
Layer (type)                    Output Shape         Param #     Connected t
o
============================================================================
======================
input_1 (InputLayer)            [(None, 299, 299, 3) 0
____________________________________________________________________________
______________________
conv2d (Conv2D)                 (None, 149, 149, 32) 864         input_1[0]
[0]
____________________________________________________________________________
______________________
batch_normalization (BatchNorma (None, 149, 149, 32) 96          conv2d[0]
[0]
____________________________________________________________________________
______________________
activation (Activation)         (None, 149, 149, 32) 0           batch_norma
lization[0][0]
```



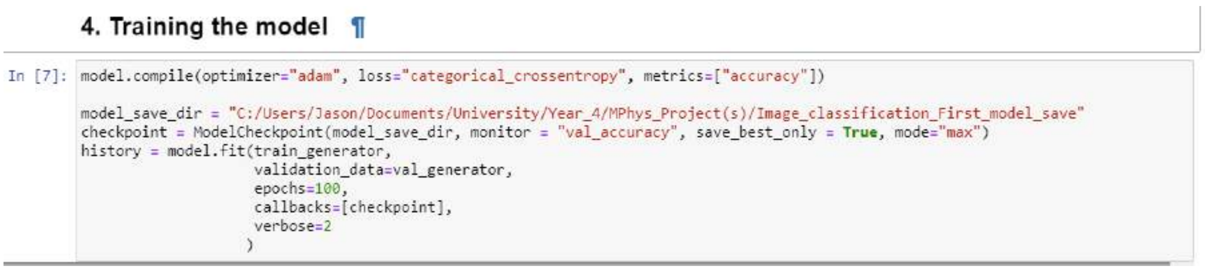
```
In [5]: last_layer = pre_trained_model.get_layer("mixed7")
        print('last layer output shape: ' + str(last_layer.output_shape))
```

Dense layer I added to mixed7 layer of inception v3



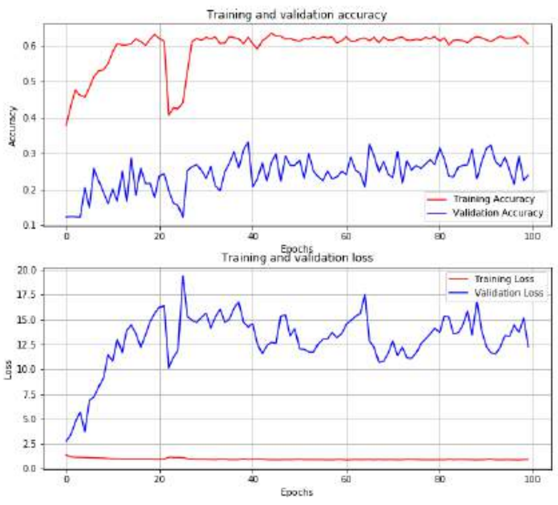
```
In [6]: x = layers.Flatten()(last_output)
        x = layers.Dense(16, activation="relu")(x)
        x = layers.Dropout(0.3)(x)
        x = layers.Dense(8, activation="relu")(x)
        x = layers.Dropout(0.3)(x)
        x = layers.Dense(4, activation="softmax")(x)

        model = Model(pre_trained_model.input, x)

        model.summary()
```

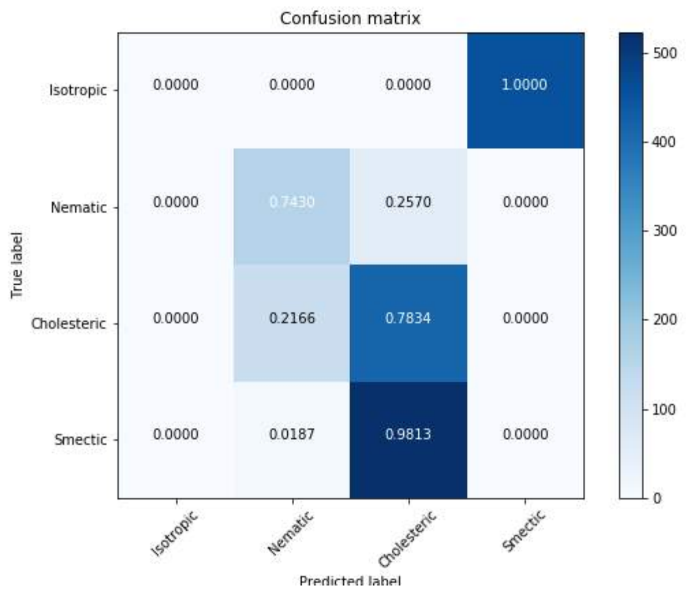
```
dropout (Dropout)               (None, 16)           0           dense[0][0]
____________________________________________________________________________
______________________
dense_1 (Dense)                 (None, 8)            136         dropout[0]
[0]
____________________________________________________________________________
______________________
dropout_1 (Dropout)             (None, 8)            0           dense_1[0]
[0]
____________________________________________________________________________
______________________
dense_2 (Dense)                 (None, 4)            36          dropout_1
[0][0]
============================================================================
======================
Total params: 12,526,684
Trainable params: 3,551,420
Non-trainable params: 8,975,264
____________________________________________________________________________
______________________
```



4. Training the model ¶

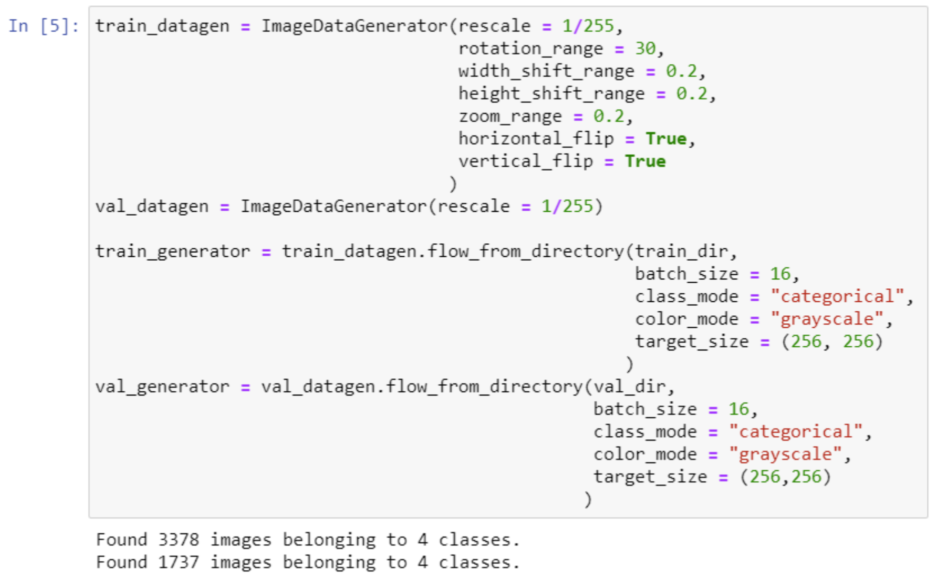
```
In [7]: model.compile(optimizer="adam", loss="categorical_crossentropy", metrics=["accuracy"])

        model_save_dir = "C:/Users/Jason/Documents/University/Year_4/MPhys_Project(s)/Image_classification_First_model_save"
        checkpoint = ModelCheckpoint(model_save_dir, monitor = "val_accuracy", save_best_only = True, mode="max")
        history = model.fit(train_generator,
                            validation_data=val_generator,
                            epochs=100,
                            callbacks=[checkpoint],
                            verbose=2
                           )
```



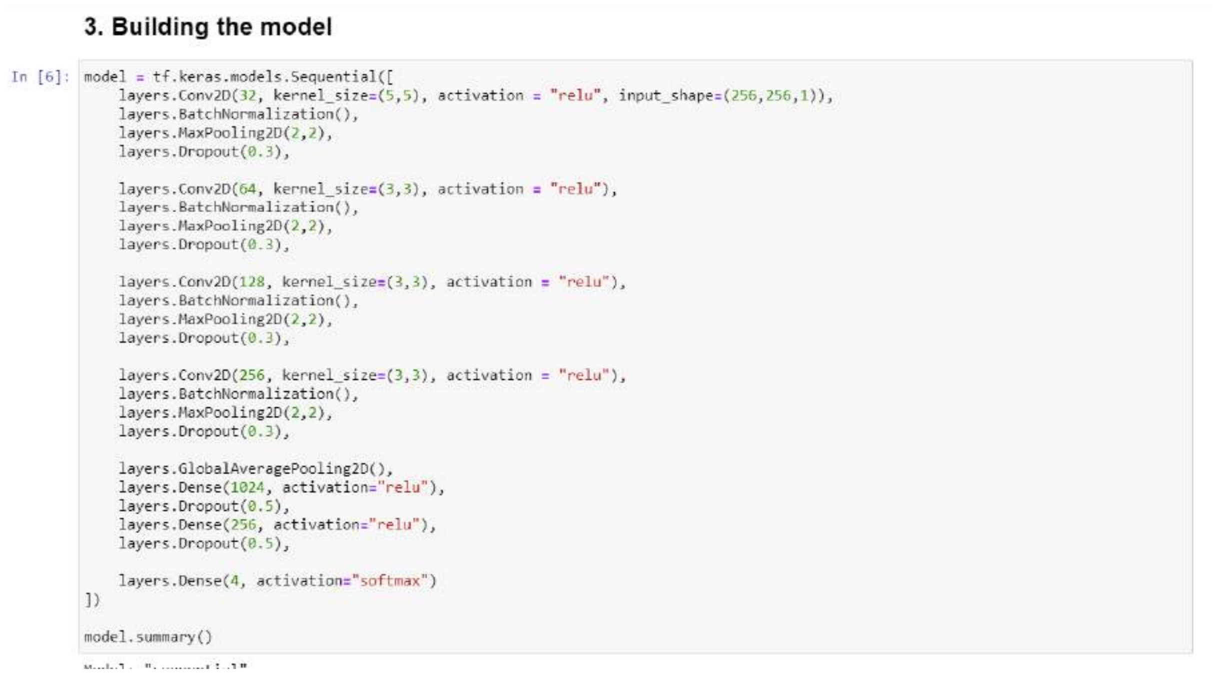


- Average time per epoch was 500s.
- Results show clear overfitting from the fact that validation loss increases. Best validation accuracy was 33% - from confusion matrix it's seen that It failed to label any isotropic or smectic phases correct in unseen data.
- File found at: \Liquid_crystals-machine_learning\LiquidCrystalMachineLearning\Video_classification\Video_recognition_proof_of_concept-transfer_learning.ipynb
- Due to long computation time and an overcomplex model possibly causing overfitting I decided to try a self made cnn model

- Here are the details and results, images were reshaped to 256x256 as powers of 2 made be better, will also try see how images size affect results later



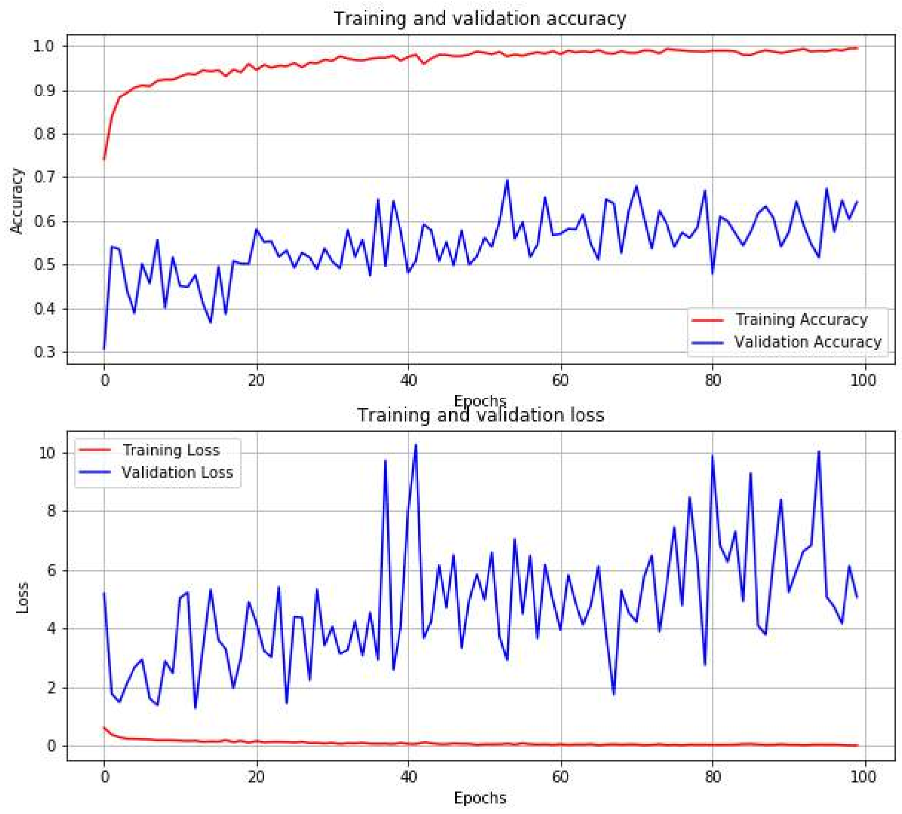
```
In [5]: train_datagen = ImageDataGenerator(rescale = 1/255,
                                           rotation_range = 30,
                                           width_shift_range = 0.2,
                                           height_shift_range = 0.2,
                                           zoom_range = 0.2,
                                           horizontal_flip = True,
                                           vertical_flip = True
                                          )
        val_datagen = ImageDataGenerator(rescale = 1/255)

        train_generator = train_datagen.flow_from_directory(train_dir,
                                                            batch_size = 16,
                                                            class_mode = "categorical",
                                                            color_mode = "grayscale",
                                                            target_size = (256, 256)
                                                           )
        val_generator = val_datagen.flow_from_directory(val_dir,
                                                        batch_size = 16,
                                                        class_mode = "categorical",
                                                        color_mode = "grayscale",
                                                        target_size = (256,256)
                                                       )

        Found 3378 images belonging to 4 classes.
        Found 1737 images belonging to 4 classes.
```

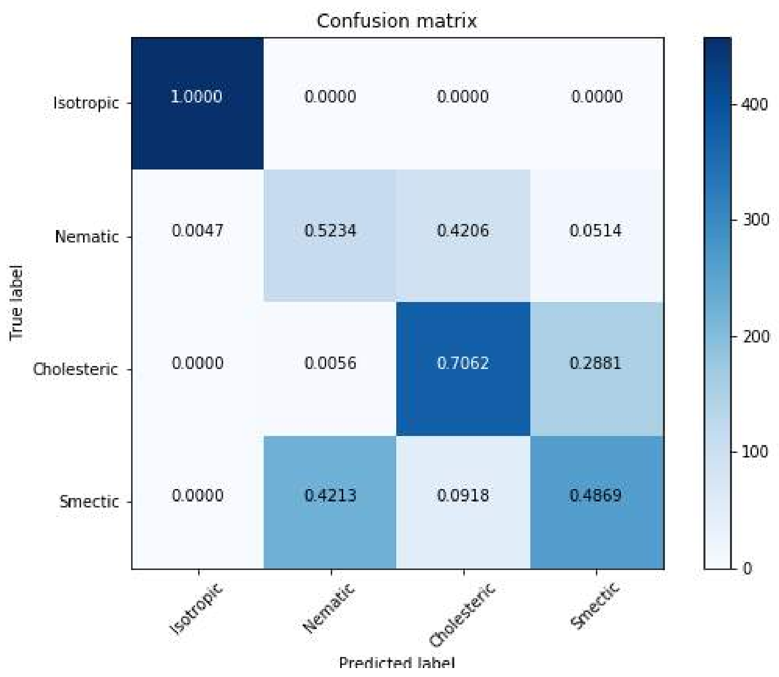


3. Building the model

```
In [6]: model = tf.keras.models.Sequential([
            layers.Conv2D(32, kernel_size=(5,5), activation = "relu", input_shape=(256,256,1)),
            layers.BatchNormalization(),
            layers.MaxPooling2D(2,2),
            layers.Dropout(0.3),

            layers.Conv2D(64, kernel_size=(3,3), activation = "relu"),
            layers.BatchNormalization(),
            layers.MaxPooling2D(2,2),
            layers.Dropout(0.3),

            layers.Conv2D(128, kernel_size=(3,3), activation = "relu"),
            layers.BatchNormalization(),
            layers.MaxPooling2D(2,2),
            layers.Dropout(0.3),

            layers.Conv2D(256, kernel_size=(3,3), activation = "relu"),
            layers.BatchNormalization(),
            layers.MaxPooling2D(2,2),
            layers.Dropout(0.3),

            layers.GlobalAveragePooling2D(),
            layers.Dense(1024, activation="relu"),
            layers.Dropout(0.5),
            layers.Dense(256, activation="relu"),
            layers.Dropout(0.5),

            layers.Dense(4, activation="softmax")
        ])

        model.summary()
```

Global average pooling was used as opposed to flatten in order to minimize the number of parameters in the network





- Best validation accuracy achieved was 69%. We still see very unstable validation loss – possibly fixed by taking out batch normalization- and also overfitting – possibly solved by using more dropout and or regularization. Average time per epoch of 480s
- File found at:

\Liquid_crystals-machine_learning\LiquidCrystalMachineLearning\Video_classification\Video_phase_recognition-2.ipynb

06/11/20 Applying trained model to test videos

Aim: See how trained model performs on test videos and attempt to save these videos with labels.

- Saved labelled test videos can be found at: \Liquid_crystals-machine_learning\LiquidCrystalMachineLearning\Video_classification\Video_phase_recognition2_videos
- Saving was eventually successful after downloading MMFPEG and using cv videowriter, frame by frame
- Annotated saved video with the predicted phase as well as the confidence (taken as the output value of the softmax for that prediction)
- Model Video_phase_recognition-2 performed well with isotropic_LC and LC-isotropic phase transitions. Also smectic-nematic and nematic-smectic transition were classified very well. Transitions involving cholesteric were not done as well, often not detecting a change between smectic and cholesteric, or confusing an inbetween isotropic and nematic to be cholesteric.
- But, consider the model used was only 69% accurate on validation data this is working quite well.

Summary: I solved the issue of videos not saving but installing MMFPEG to work alongside cv to write labelled frames to a video file. The classifier, despite only being 69% accurate on validation images worked quite well, expect for transitions involving cholesteric. Will try to improve the current model using dropout and regularization to see how it improves.

Week 5 summary:

- Successfully applied a trained model to Isotropic, Nematic, Cholesteric, Smectic transition videos, with saving capabilities.
- Model trained showed a good nematic smectic phase transition recognition. Differentiating cholesteric and smectic was not as good.
- Looking at transitions it probably the main direction I will take this project in.
- Rather than focusing on other machine learning techniques, I will focus on the technique Ive used so far in looking at transitions.

10/11/20 Sixth meeting (video call)

Attendees: Dr. Dierking, Josh Heaton, James Harbon, Jason Dominguez

Discussed:

- Showed some of my current videos, Dr Dierking said these seem pretty good, even the ones which aren't perfectly accurate are explainable, i.e. during a transition when confusing nematic for cholesteric, this is understandable as the texture wasn't very characteristic and we know the molecule is not chiral so it must be nematic.
- Dr Dierking suggested that, given we know the frames per second and the rate of temperature increase of the video, we can relate everything back to temperature and use these videos as an estimate for transition temperature.
- Dr Dierking said, after I asked, that videos from the same molecule can be split up between training and test without seeming to be a problem, as long as images from a single video are not.
- Dr Dierking suggested plotting graphs of confidence against temperature as this would show interesting results for the phase transition.

Aims/Objectives for week 6:

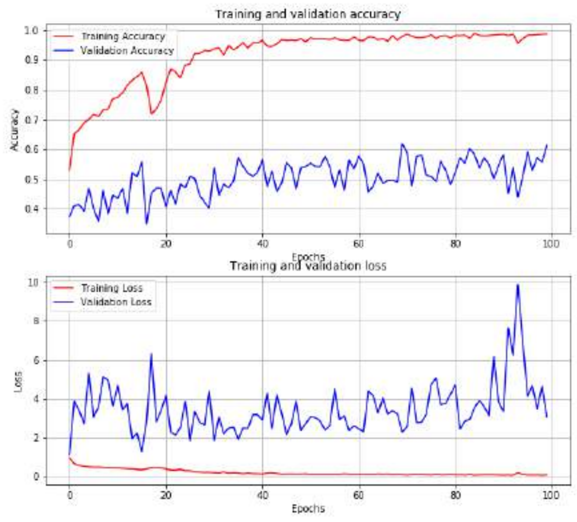
- Plot phase confidence vs temperature graphs for videos already saved
- Use full data set for new classifier, possibly attempt including SmC and SmA classification
- Move model training to google colab as this will enable use of gpu, making training models must faster so I can iterate through ideas quicker.

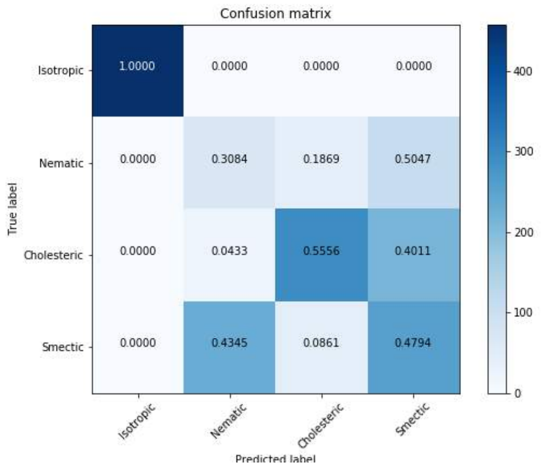
12/11/20 Started using google colab and trained several models for (I,N,N*,Sm) and (I,N,N*,SmC,SmA) classification using full dataset so far

Aim: Move to using google colab for quicker training times. Use full dataset to hopefull get higher validation accuracy models. Apply models to video aiming for more accurate labelling for phase transitions.

- Dataset used for (I,N,N*,SmC,SmA) classification "C:\Users\Jason\Documents\University\Year_4\MPhys_Project(s)\Liquid_crystals-machine_learning\All_images_12-11-20-(I,N,Chol,Sm)"
- Dataset used for (I,N,N*,SmC,SmA) classification "C:\Users\Jason\Documents\University\Year_4\MPhys_Project(s)\Liquid_crystals-machine_learning\All_images_10-11-20-(I,N,Chol,SmC,SmA)"
- Test video were stored at "C:\Users\Jason\Documents\University\Year_4\MPhys_Project(s)\Liquid_crystals-machine_learning\All_images_Test_videos_10-11-20"
- Several model were trained and applied to videos, but these were not saved as they didn't improve upon the already obtained labelled videos and I had limited hard drive space (the models have the same basic structure as the last model – Video_phase_recognition-2 – any differences are mentioned:

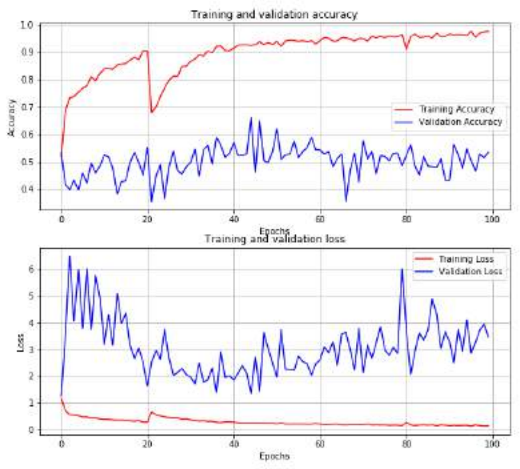
- "C:\Users\Jason\Documents\University\Year_4\MPhys_Project(s)\Liquid_crystals-machine_learning\LiquidCrystalMachineLearning\Video_classification\Video_phase_recognition-3.ipynb"

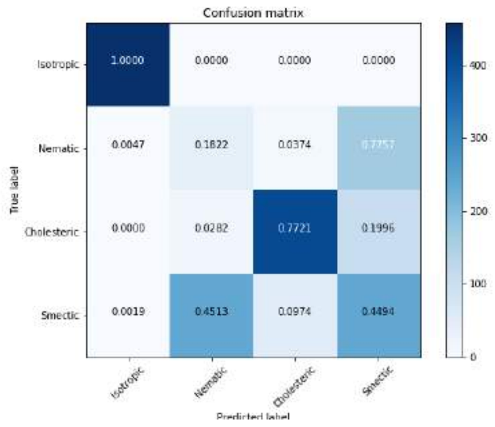


- 

	Isotropic	Nematic	Cholesteric	Smectic
Isotropic	1.0000	0.0000	0.0000	0.0000
Nematic	0.0000	0.3084	0.1869	0.5047
Cholesteric	0.0000	0.0433	0.5556	0.4011
Smectic	0.0000	0.4345	0.0861	0.4794

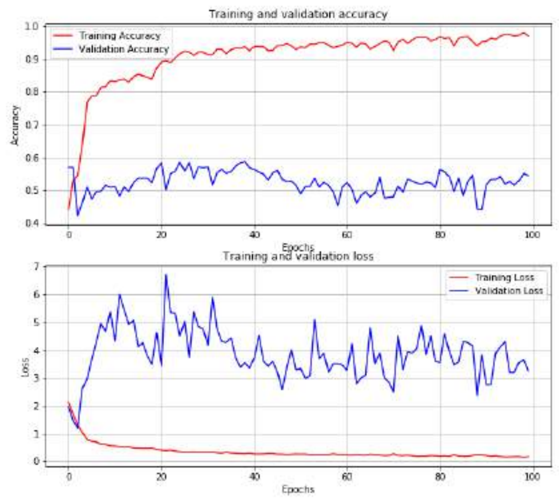
-
- This model had: rotation_range = 45, width_shift_range = 0.3, height_shift_range = 0.3, zoom_range = 0.3 augmentation, batch_size 16, conv dropouts 0.3, global average pooling before dense layers, 1024 dense then 256 dense, 0.5 dropout between dense, adam optimizer, 100 epochs.
- "C:\Users\Jason\Documents\University\Year_4\MPhys_Project(s)\Liquid_crystals-machine_learning\LiquidCrystalMachineLearning\Video_classification\Video_phase_recognition-4.ipynb"



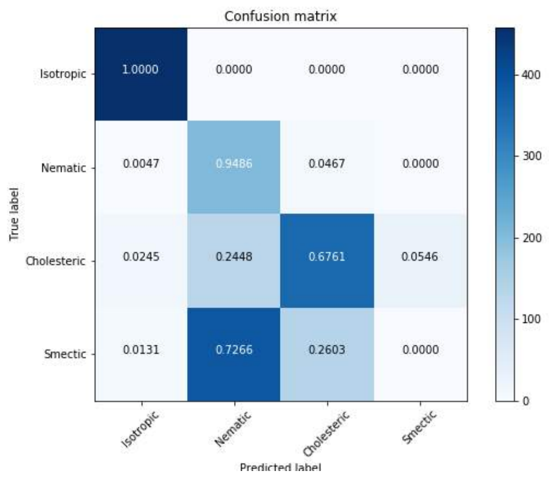
- 

	Isotropic	Nematic	Cholesteric	Smectic
Isotropic	1.0000	0.0000	0.0000	0.0000
Nematic	0.0047	0.1822	0.0374	0.7757
Cholesteric	0.0000	0.0282	0.7721	0.1996
Smectic	0.0019	0.4513	0.0974	0.4494

-
- This model had: rotation_range = 30, width_shift_range = 0.2, height_shift_range = 0.2, zoom_range = 0.2 augmentation, and 0.0005 regularization on every layer, bias and kernel
- "C:\Users\Jason\Documents\University\Year_4\MPhys_Project(s)\Liquid_crystals-machine_learning\LiquidCrystalMachineLearning\Video_classification\Video_phase_recognition-5.ipynb"

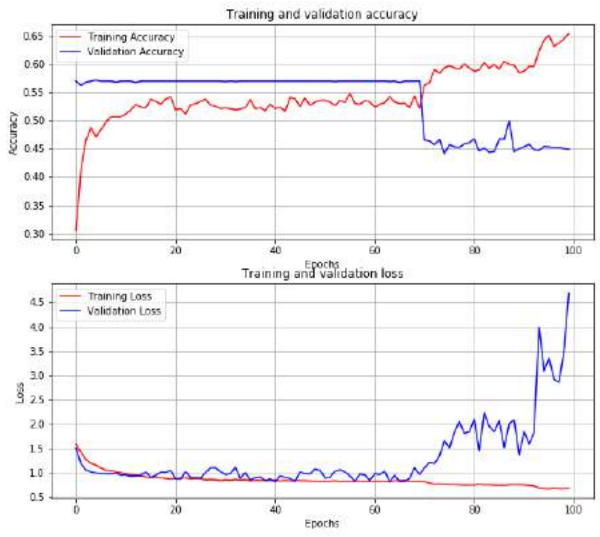


-
-

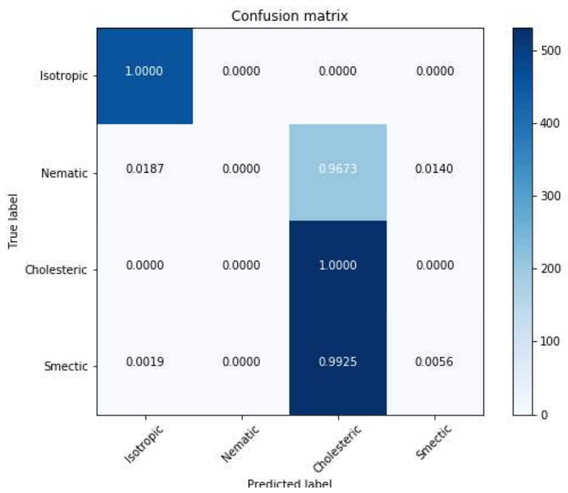


	Isotropic	Nematic	Cholesteric	Smectic
Isotropic	1.0000	0.0000	0.0000	0.0000
Nematic	0.0047	0.9486	0.0467	0.0000
Cholesteric	0.0245	0.2448	0.6761	0.0546
Smectic	0.0131	0.7266	0.2603	0.0000

-
- This model had: 128 batch size, 0.001 regularization, adam optimizer with 0.0005 learning rate
- "C:\Users\Jason\Documents\University\Year_4\MPhys_Project(s)\Liquid_crystals-machine_learning\LiquidCrystalMachineLearning\Video_classification\Video_phase_recognition-6.ipynb"



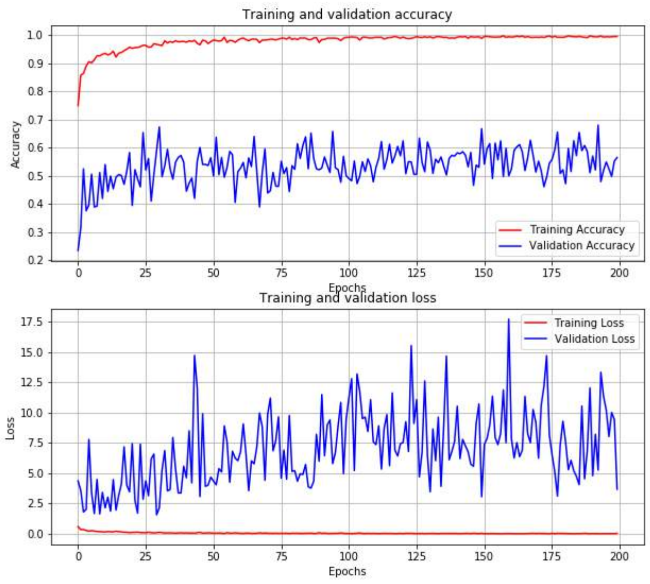
-

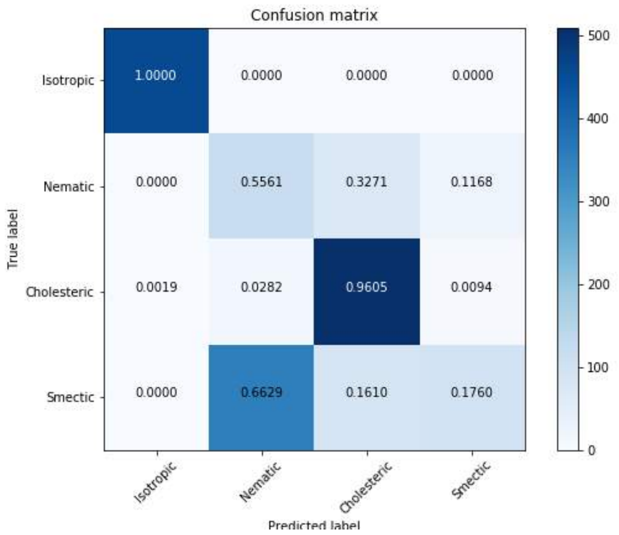


	Isotropic	Nematic	Cholesteric	Smectic
Isotropic	1.0000	0.0000	0.0000	0.0000
Nematic	0.0187	0.0000	0.9673	0.0140
Cholesteric	0.0000	0.0000	1.0000	0.0000
Smectic	0.0019	0.0000	0.9925	0.0056

-

- This model had: Flatten layer before dense, 32 dense followed by 16 dense, still with dropout 0.5 between. Dropout layers for conv changed to 0.2, deafault adam optimizer learning rate used
- "C:\Users\Jason\Documents\University\Year_4\MPhys_Project(s)\Liquid_crystals-machine_learning\LiquidCrystalMachineLearning\Video_classification\Video_phase_recognition-7.ipynb"



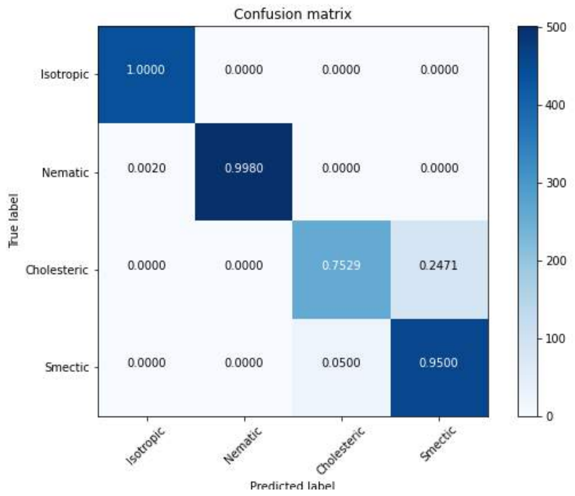
- 

	Isotropic	Nematic	Cholesteric	Smectic
Isotropic	1.0000	0.0000	0.0000	0.0000
Nematic	0.0000	0.5561	0.3271	0.1168
Cholesteric	0.0019	0.0282	0.9605	0.0094
Smectic	0.0000	0.6629	0.1610	0.1760

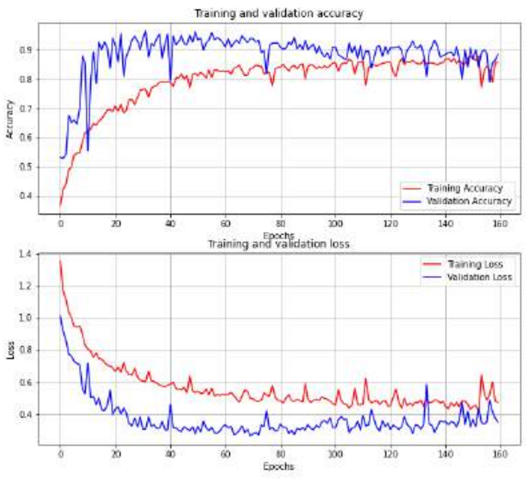
-
- This model had: same as 3 but with 200 epochs

- Model found at: "C:\Users\Jason\Documents\University\Year_4\MPhys_Project(s)\Liquid_crystals-machine_learning\LiquidCrystalMachineLearning\Video_classification\Video_phase_recognition-google-colab(I,N,Chol,Sm)1.ipynb"

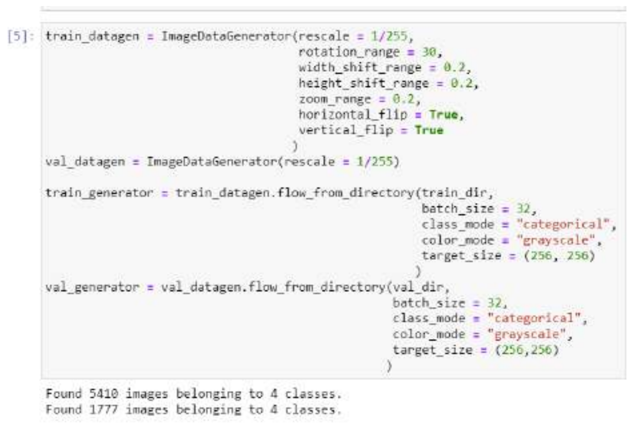
- Used to label videos, found at "C:\Users\Jason\Documents\University\Year_4\MPhys_Project(s)\Liquid_crystals-machine_learning\LiquidCrystalMachineLearning\Video_classification\(I,N,Chol,Sm)_first_videos"

- Labelled videos are not more accurate than the best videos so far, particularly with nematic smectic, which is unusual as this model was using more data and got a lot better validation accuracy.
- 

	Isotropic	Nematic	Cholesteric	Smectic
Isotropic	1.0000	0.0000	0.0000	0.0000
Nematic	0.0020	0.9980	0.0000	0.0000
Cholesteric	0.0000	0.0000	0.7529	0.2471
Smectic	0.0000	0.0000	0.0500	0.9500

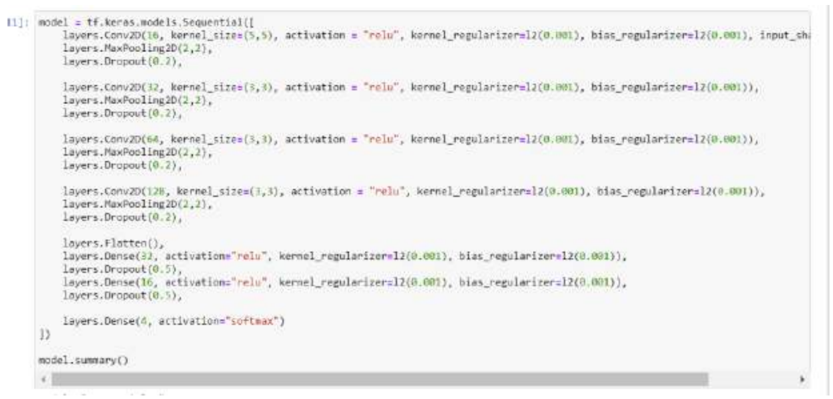
- 

- This model was run on google colab gpu, with average time per epoch of 187s



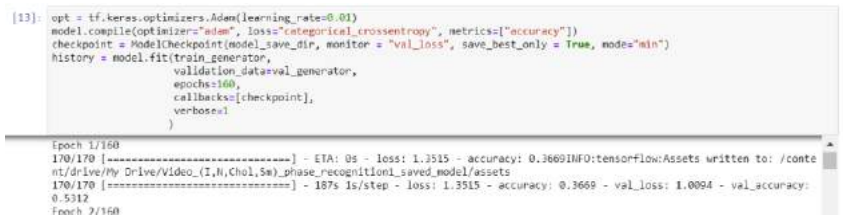
```
[5]: train_datagen = ImageDataGenerator(rescale = 1/255,
                                      rotation_range = 30,
                                      width_shift_range = 0.2,
                                      height_shift_range = 0.2,
                                      zoom_range = 0.2,
                                      horizontal_flip = True,
                                      vertical_flip = True
                                     )
     val_datagen = ImageDataGenerator(rescale = 1/255)

     train_generator = train_datagen.flow_from_directory(train_dir,
                                                         batch_size = 32,
                                                         class_mode = "categorical",
                                                         color_mode = "grayscale",
                                                         target_size = (256, 256)
                                                        )
     val_generator = val_datagen.flow_from_directory(val_dir,
                                                     batch_size = 32,
                                                     class_mode = "categorical",
                                                     color_mode = "grayscale",
                                                     target_size = (256,256)
                                                    )

     Found 5410 images belonging to 4 classes.
     Found 1777 images belonging to 4 classes.
```

-



```
11]: model = tf.keras.models.Sequential([
         layers.Conv2D(16, kernel_size=(5,5), activation = "relu", kernel_regularizer=l2(0.001), bias_regularizer=l2(0.001), input_sh
         layers.MaxPooling2D(2,2),
         layers.Dropout(0.2),

         layers.Conv2D(32, kernel_size=(3,3), activation = "relu", kernel_regularizer=l2(0.001), bias_regularizer=l2(0.001)),
         layers.MaxPooling2D(2,2),
         layers.Dropout(0.2),

         layers.Conv2D(64, kernel_size=(3,3), activation = "relu", kernel_regularizer=l2(0.001), bias_regularizer=l2(0.001)),
         layers.MaxPooling2D(2,2),
         layers.Dropout(0.2),

         layers.Conv2D(128, kernel_size=(3,3), activation = "relu", kernel_regularizer=l2(0.001), bias_regularizer=l2(0.001)),
         layers.MaxPooling2D(2,2),
         layers.Dropout(0.2),

         layers.Flatten(),
         layers.Dense(32, activation="relu", kernel_regularizer=l2(0.001), bias_regularizer=l2(0.001)),
         layers.Dropout(0.5),
         layers.Dense(16, activation="relu", kernel_regularizer=l2(0.001), bias_regularizer=l2(0.001)),
         layers.Dropout(0.5),

         layers.Dense(4, activation="softmax")
     ])

     model.summary()
```

-
-



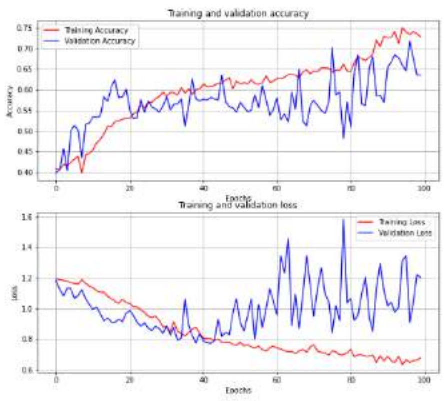
```
[13]: opt = tf.keras.optimizers.Adam(learning_rate=0.01)
      model.compile(optimizer="adam", loss="categorical_crossentropy", metrics=["accuracy"])
      checkpoint = ModelCheckpoint(model_save_dir, monitor = "val_loss", save_best_only = True, mode="min")
      history = model.fit(train_generator,
                          validation_data=val_generator,
                          epochs=160,
                          callbacks=[checkpoint],
                          verbose=1
                         )

      Epoch 1/160
      170/170 [==============================] - ETA: 0s - loss: 1.3515 - accuracy: 0.3669INFO:tensorflow:Assets written to: /conte
      nt/drive/My Drive/Video_(I,N,Chol,Sm)_phase_recognition1_saved_model/assets
      170/170 [==============================] - 187s 1s/step - loss: 1.3515 - accuracy: 0.3669 - val_loss: 1.0094 - val_accuracy:
      0.5312
      Epoch 2/160
```

-

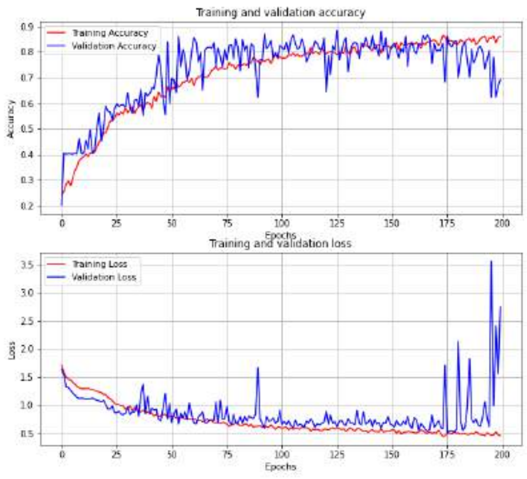
- The following models were used to label videos for transitions, including N, chol, I and smA and smC:
- "C:\Users\Jason\Documents\University\Year_4\MPhys_Project(s)\Liquid_crystals-machine_learning\LiquidCrystalMachineLearning\Video_classification\Video_phase_recognition-google-colab1.ipynb"
- "C:\Users\Jason\Documents\University\Year_4\MPhys_Project(s)\Liquid_crystals-machine_learning\LiquidCrystalMachineLearning\Video_classification\Video_phase_recognition-google-colab2.ipynb"
- "C:\Users\Jason\Documents\University\Year_4\MPhys_Project(s)\Liquid_crystals-machine_learning\LiquidCrystalMachineLearning\Video_classification\Video_phase_recognition-google-colab3.ipynb"
- Corresponding videos found at:

- "C:\Users\Jason\Documents\University\Year_4\MPhys_Project(s)\Liquid_crystals-machine_learning\LiquidCrystalMachineLearning\Video_classification\(I,N,Chol,SmC,SmA)_first_videos"
- "C:\Users\Jason\Documents\University\Year_4\MPhys_Project(s)\Liquid_crystals-machine_learning\LiquidCrystalMachineLearning\Video_classification\(I,N,Chol,SmC,SmA)_second_videos"
- "C:\Users\Jason\Documents\University\Year_4\MPhys_Project(s)\Liquid_crystals-machine_learning\LiquidCrystalMachineLearning\Video_classification\(I,N,Chol,SmC,SmA)_third_videos"

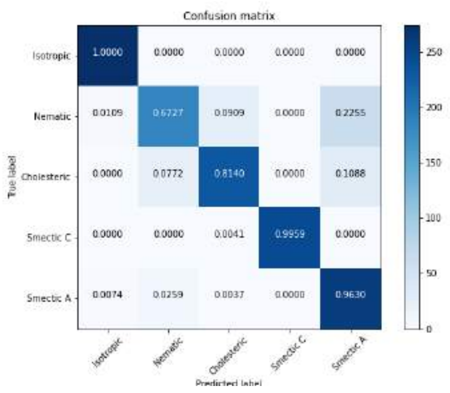
- For the first

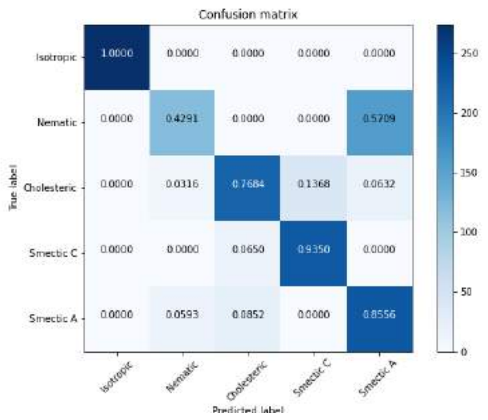


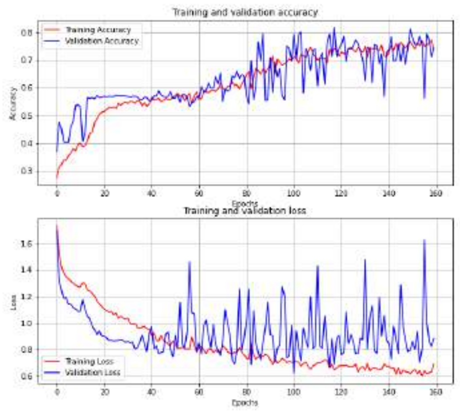
-
- Same model as above but with 16 batch size
- Second model:



-

- 

- Same as above but with 200 epochs
- Third:
- Same as above but with 160 epochs
- 

- 

- Although validation accuracies of 80%+ were achieved with these models, labelled videos did not notice smectic A to C changes, and other labelled transitions were still not as good as the current best, even though these models achieved higher validation accuracy and used more data.

Summary: Many models were trained using google colab gpu for fast training. For both the (I,N,N*,Sm) and (I,N,N*,SmC,SmA) classification problems, high accuracies, higher than 80% and sometimes above 90% were achieved, however models did not seem to improve the current best labelling of test videos. One thought as to why could be that the videos are un cut up and in their original high resolution 2000x1000 or so pixels, whereas the validation set does not match this, being made up of cut up frames, of size 640x320. This is the only current explanation right not for why the weaker looking model (60ish% validation accuracy and less data to train on) is currently the best at labelling videos.

14/11/20 Dr Dierking sent more videos

- Dr Dierking sent 3 more videos of phase transitions including SmA, SmE and isotropic phases
- Videos stored in "C:\Users\Jason\Documents\University\Year_4\MPhys_Project(s)\Liquid_crystals-machine_learning\Videos_from_Ingo_14-11-20"
- Split videos into their frames using VLC media player, capturing every 10 frames as this was judged to be enough frames not to miss and changes in the video frames and not too often so that images looked all the same.
- Images sorted and stored at "C:\Users\Jason\Documents\University\Year_4\MPhys_Project(s)\Liquid_crystals-machine_learning\Images_from_Ingo_14-11-20"

16/11/20 Plotted graphs of confidence against temperature for labelled videos

Aim: As per Dr Dierking's suggestion, I want to plot graphs of confidence against temperature for the labelled phase transition videos

- Videos of phase transitions were labelled with a start temperature, temperature change rate (for some), whether it was heating or cooling and the compound (e.g. 8CB).
- It was found for some videos that maybe they were mislabelled, as the temperature range listed didn't match with the temperature change rate and length of the video. I will ask Dr Dierking about this, but for now it seemed best to take the temperature range as the truth and ignore the temperature range when it conflicts.
- Python file used for plotting confidence temperature graphs, as well as exporting data as a csv and labelling videos is found at: "C:\Users\Jason\Documents\University\Year_4\MPhys_Project(s)\Liquid_crystals-machine_learning\LiquidCrystalMachineLearning\Video_classification\Confidence-temperature_data\Confidence-temperature_graph_plotter.ipynb"
- The temperature assigned to each frame of the video was calculated as

$$T = T_{start} + \frac{T_{end} - T_{start}}{n_{frames}} N,$$

where T_{start} is start temperature of video T_{end} is the end temperature, n_{frames} is the number of frames in the video (obtained using OpenCV library in python) and N is the frame number iterated through from 0 to n_{frames}.
- If a end temperature wasn't defined, but a temperature rate ΔT was then

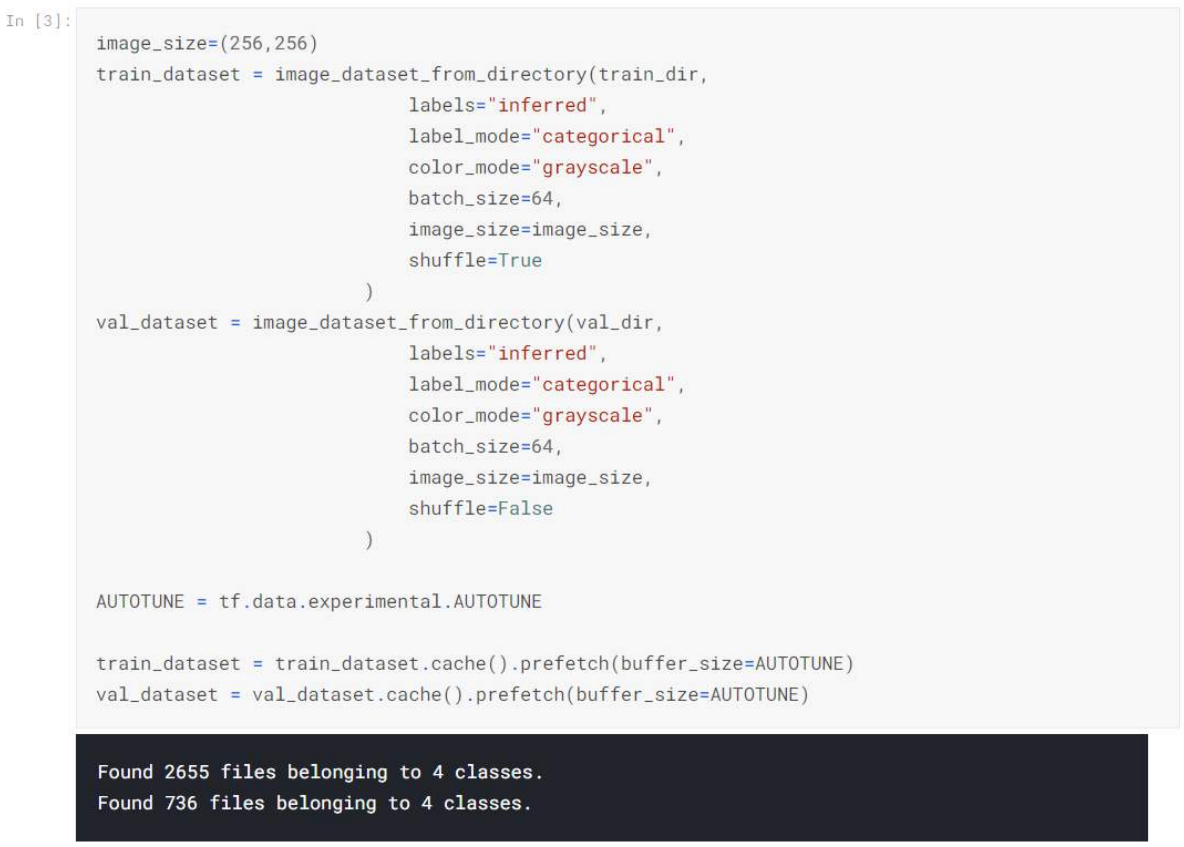
$$T = T_{start} + \frac{\Delta T}{fps} N$$

where fps is the videos frames per second, obtained using OpenCV.

- Labelled videos for which I plotted graphs (with numbers indicating temp range) all labelled videos and data files can be found in this folder, with the names listed below, "C:\Users\Jason\Documents\University\Year_4\MPhys_Project(s)\Liquid_crystals-machine_learning\LiquidCrystalMachineLearning\Video_classification\Confidence-temperature_data":
    1. 8CB
        - I-N_40.8
        - I-N_41
        - N-Sm_34-33(3)
        - N-Sm_34-33(2)
        - N-Sm_34-33
        - Sm-N_32-36
        - Sm-N_33-36
    2. M6
        - 100-86
        - 124-150
        - 154-184
        - 175-115
        - 181-154
- Model used for labelling these videos was trained on dataset that has images from these videos removed, dataset is found at "C:\Users\Jason\Documents\University\Year_4\MPhys_Project(s)\Liquid_crystals-machine_learning\(I,N,Chol,Sm)Images-more_balanced"
- I used Kaggle GPU to train this model, so this dataset was uploaded to Kaggle
- This was the splitting of dataset, with no images from same videos split among training and validation sets

• Phase	• Training set	• Validation set
• Isotropic	• 666	• 186
• Nematic	• 597	• 166
• Cholesteric	• 670	• 191
• Smectic	• 674	• 193

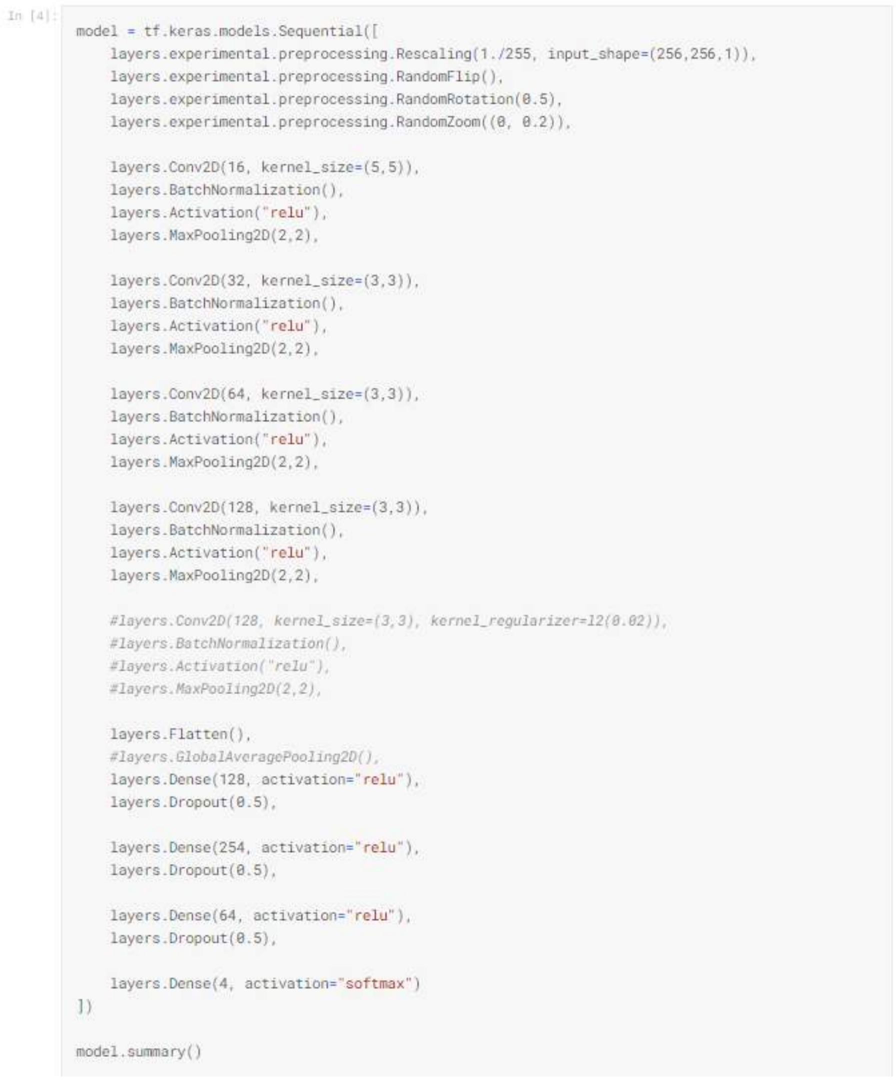
- Images were deleted from cholesteric and smectic to make a more balanced dataset as seen above. Nematic only had 597 and cholesteric had 1151 for example.



```
In [3]:
image_size=(256,256)
train_dataset = image_dataset_from_directory(train_dir,
                             labels="inferred",
                             label_mode="categorical",
                             color_mode="grayscale",
                             batch_size=64,
                             image_size=image_size,
                             shuffle=True
                         )
val_dataset = image_dataset_from_directory(val_dir,
                             labels="inferred",
                             label_mode="categorical",
                             color_mode="grayscale",
                             batch_size=64,
                             image_size=image_size,
                             shuffle=False
                         )

AUTOTUNE = tf.data.experimental.AUTOTUNE

train_dataset = train_dataset.cache().prefetch(buffer_size=AUTOTUNE)
val_dataset = val_dataset.cache().prefetch(buffer_size=AUTOTUNE)
```

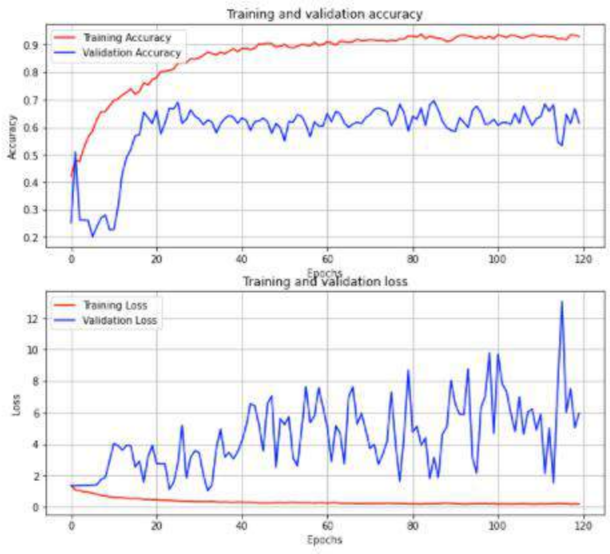
```
Found 2655 files belonging to 4 classes.
Found 736 files belonging to 4 classes.
```

-
- Cache feature was used. This allowed for the images in the dataset to be stored in a cache after first epoch, leading to much faster training.



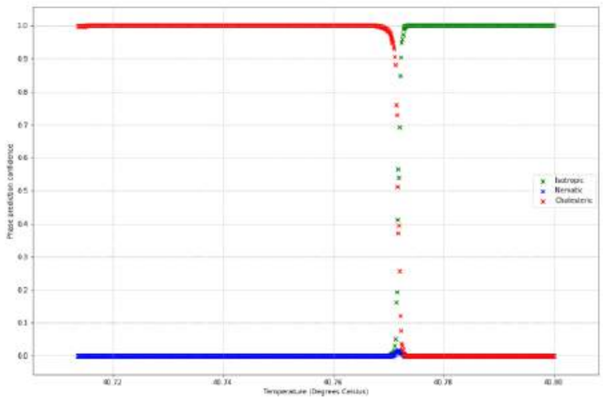
```
In [4]:
model = tf.keras.models.Sequential([
    layers.experimental.preprocessing.Rescaling(1./255, input_shape=(256,256,1)),
    layers.experimental.preprocessing.RandomFlip(),
    layers.experimental.preprocessing.RandomRotation(0.5),
    layers.experimental.preprocessing.RandomZoom((0, 0.2)),

    layers.Conv2D(16, kernel_size=(5,5)),
    layers.BatchNormalization(),
    layers.Activation("relu"),
    layers.MaxPooling2D(2,2),

    layers.Conv2D(32, kernel_size=(3,3)),
    layers.BatchNormalization(),
    layers.Activation("relu"),
    layers.MaxPooling2D(2,2),

    layers.Conv2D(64, kernel_size=(3,3)),
    layers.BatchNormalization(),
    layers.Activation("relu"),
    layers.MaxPooling2D(2,2),

    layers.Conv2D(128, kernel_size=(3,3)),
    layers.BatchNormalization(),
    layers.Activation("relu"),
    layers.MaxPooling2D(2,2),

    #layers.Conv2D(128, kernel_size=(3,3), kernel_regularizer=l2(0.02)),
    #layers.BatchNormalization(),
    #layers.Activation("relu"),
    #layers.MaxPooling2D(2,2),

    layers.Flatten(),
    #layers.GlobalAveragePooling2D(),
    layers.Dense(128, activation="relu"),
    layers.Dropout(0.5),

    layers.Dense(254, activation="relu"),
    layers.Dropout(0.5),

    layers.Dense(64, activation="relu"),
    layers.Dropout(0.5),

    layers.Dense(4, activation="softmax")
])

model.summary()
```

-
- Flip, rotation and zoom image augmentation was used
- Ran for 120 epochs, adam optimizer, learning rate 0.0001, model saved with best accuracy, first epoch took 163s and on average, the rest took between 3s or 7s when saving model

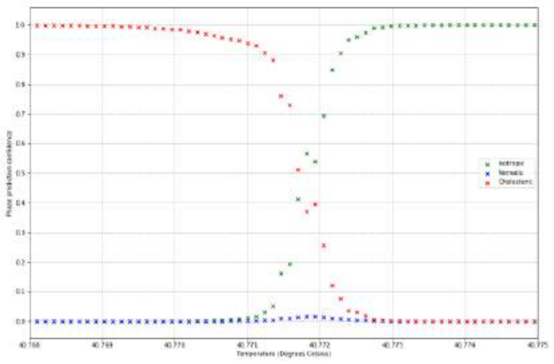


-
- These were the results, with peak validation accuracy of about 70%
- Video results:

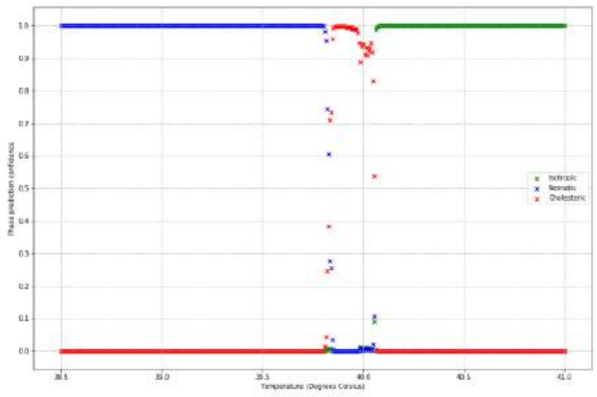
- 8CB I-N40.8



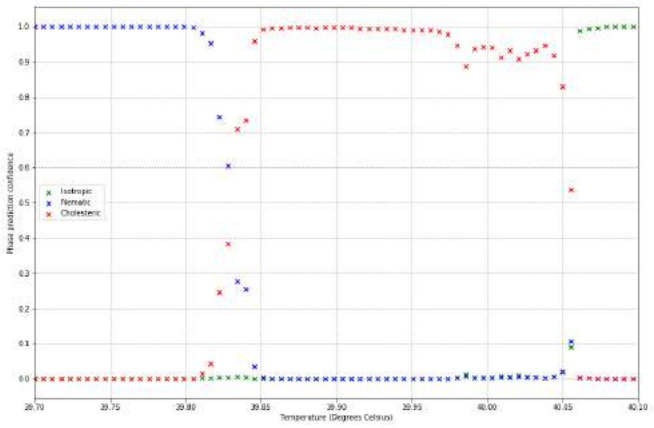
-



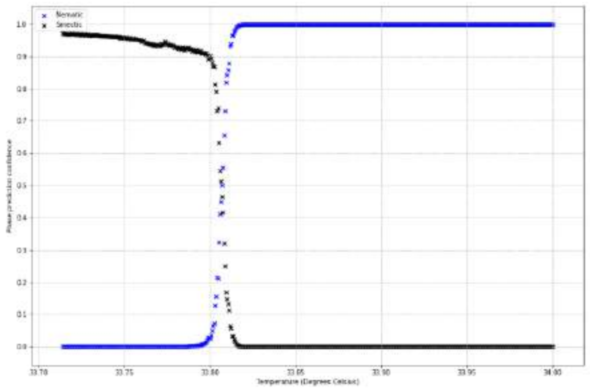
-
- Misclassified I-N transition as I-Cholesteric transition
- 8CB I-N_41



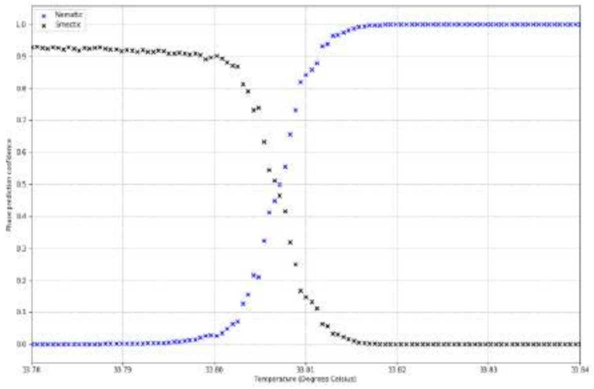
-



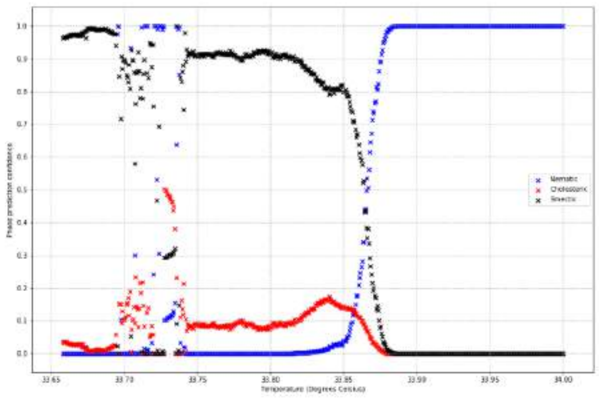
-
- Initially misclassified I-N transition as I-Chol, but then correctly recognized that it was N
- 8CB N-Sm_34-33(3)



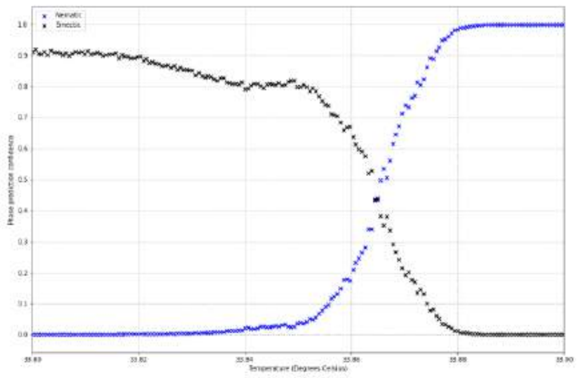
-



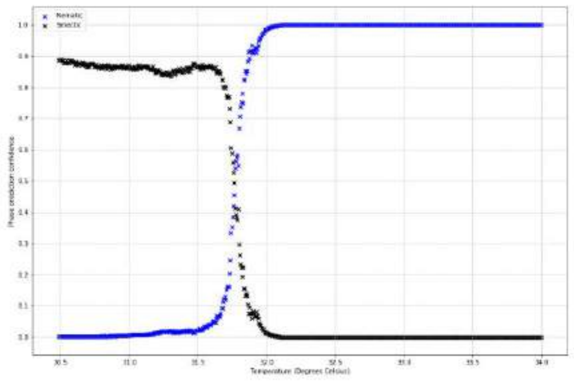
-
- Correctly classified N-Sm transition
- 8CB N-Sm_34-33(2)



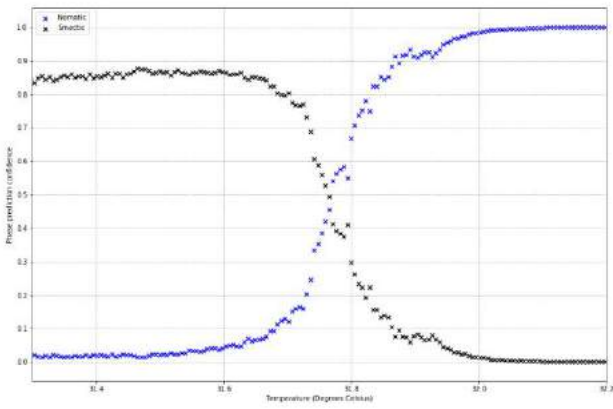
-



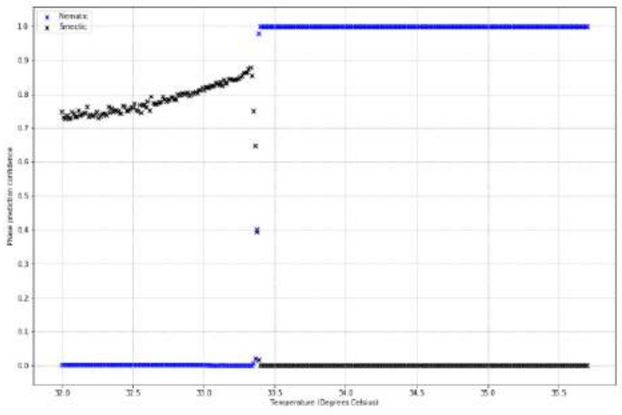
-
- Correctly classified N-Sm transition, confusion at the lower temperatures seen in the above graph is due to the slide moving around in the video, blurring the image.
- 8CB N-Sm_34-33



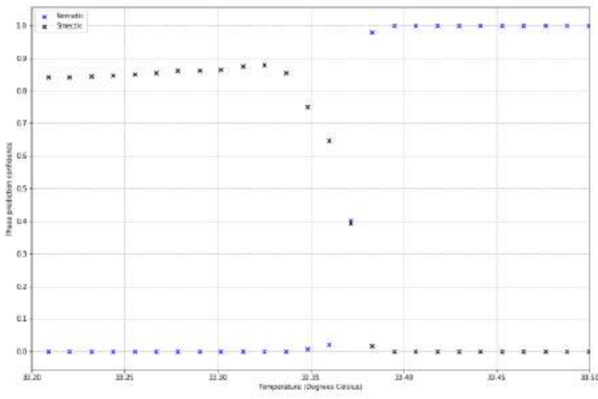
-



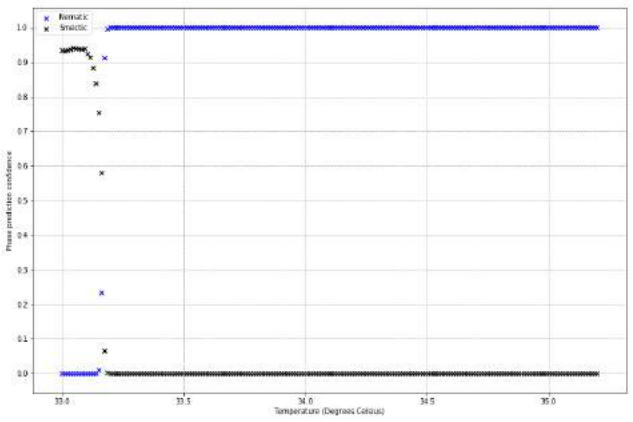
-
- Correctly classified N-Sm transition
- 8CB Sm-N_32-36

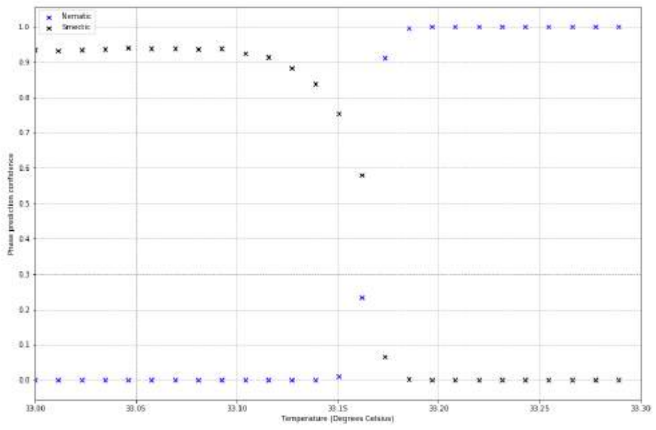


-

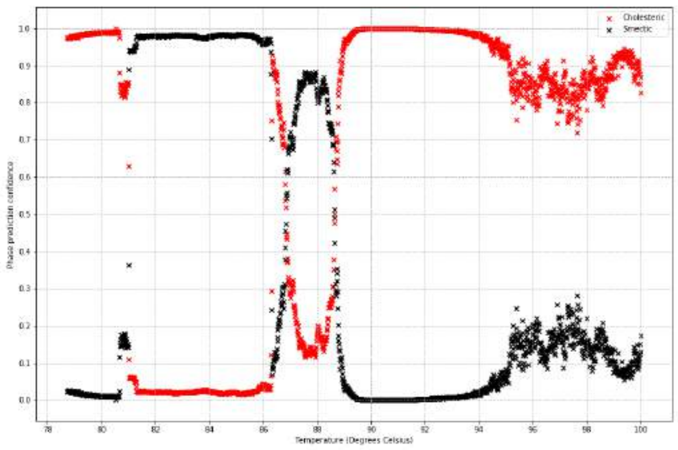


-
- Correctly classified Sm-N transition
- 8CB Sm-N_33-36

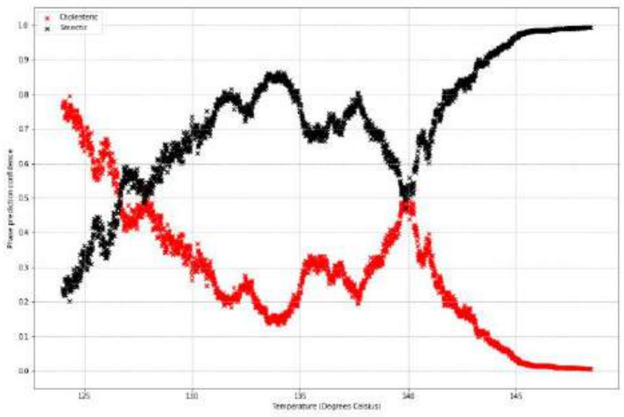


- 

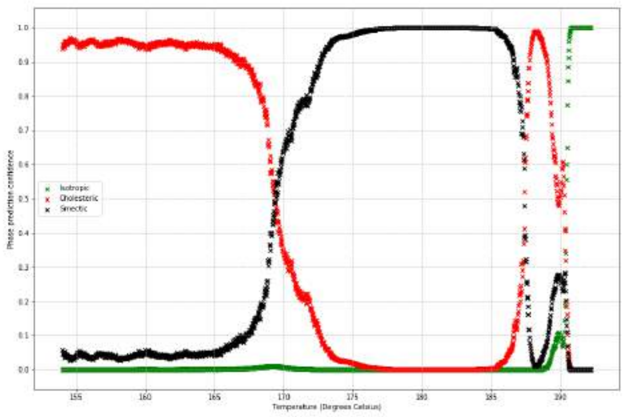
-
- Correctly classified Sm-N transition
- M6 100-86



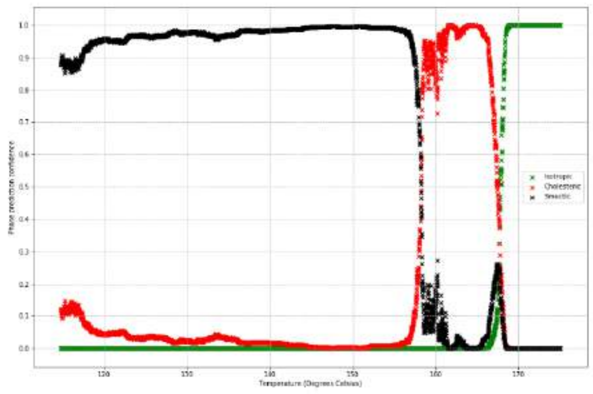
-
- Didn't seem to work well for this transition, it missed where I think the transition occurred however it seemed to classify a quick transition from SmC-A as Sm-Chol at 81
- M6 124-150



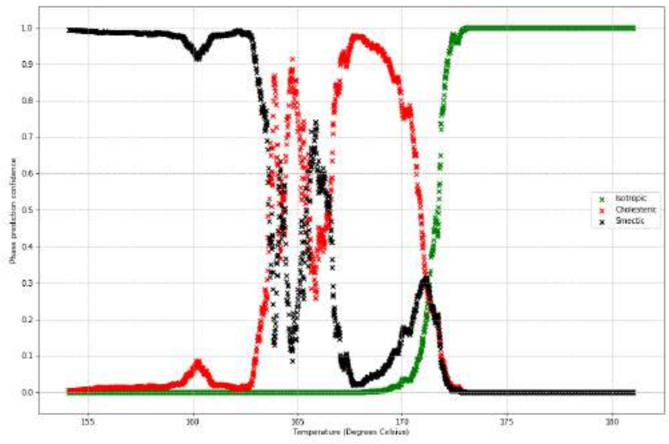
-
- Misclassified Sm-Chol as Chol-Sm
- M6 154-184



-
- Identifies Smectic although I believe there was only cholesteric to isotropic. Correctly identifies cholesteric to isotropic transition
- M6 175-115



-
- Correctly identifies I-Chol transition, but it “identifies” a Chol-Sm transition too early
- M6 181-154



-
- Correctly identifies I-Chol transition, however the rest is confusing to me

Summary: Confidence temperature graphs were plotted for several labelled videos. The model used only achieved roughly 60% validation accuracy but was able to perform very well on 8CB transitions, leading to nice graphs indicating temperature of transition. Performs on M6 involving Sm and Cholesteric was less good. This should be tried for other models, maybe ones that performed better on cholesteric smectic transitions. Need to ask Dr Dierking about the confusion over some videos possibly being mislabelled with temperature change rate. Also should find out how to assign error to the temperatures.

Week 6 summary:

- Moved to google colab for faster training with gpu and using cache
- Trained several models and found that overfitting to validation set can occur as some models performed very well on validation set but did not generalize so well to test videos
- Also plotted confidence against temperature graphs for labelled videos to see what was happening at phase transitions.
- Need to find out how to deal with errors in temperature in these graphs and also try improve the model used for classifying the transitions.

17/11/20 Seventh meeting (video call)

Attendees: Dr. Dierking, Josh Heaton, James Harbon, Jason Dominguez

Discussed:

- I showed some of the graphs I had so far
- Dr Dierking spoke about comparing these to the kind of thing obtained with calorimetry in determining phase transition temperatures.
- Dr Dierking said he would look at videos he had send us so far and email us about what video highlight particular transitions nicely.
- I said ill upload images from the latest videos sent by Dr Dierking which I sorted by phase to the other group and they will split images into smaller images and sent back.

Week 7 Aims:

- Ive ran out of GPU time on colab so training wont be so easy, so the biggest aim for this week is to start working on/planning the semester 1 MPhys report.

21/11/20 Created report plan

Aim: write a report plan to send to Dr Dierking to see if I'm including everything that needs to be included.

- Decided to include section on calorimetry to make comparisons in terms of how well this method identifies transitions.

Summary: Report plan was made in LaTex, now needs to be sent to Dr Dierking for feedback. Also I should look into calorimetry.

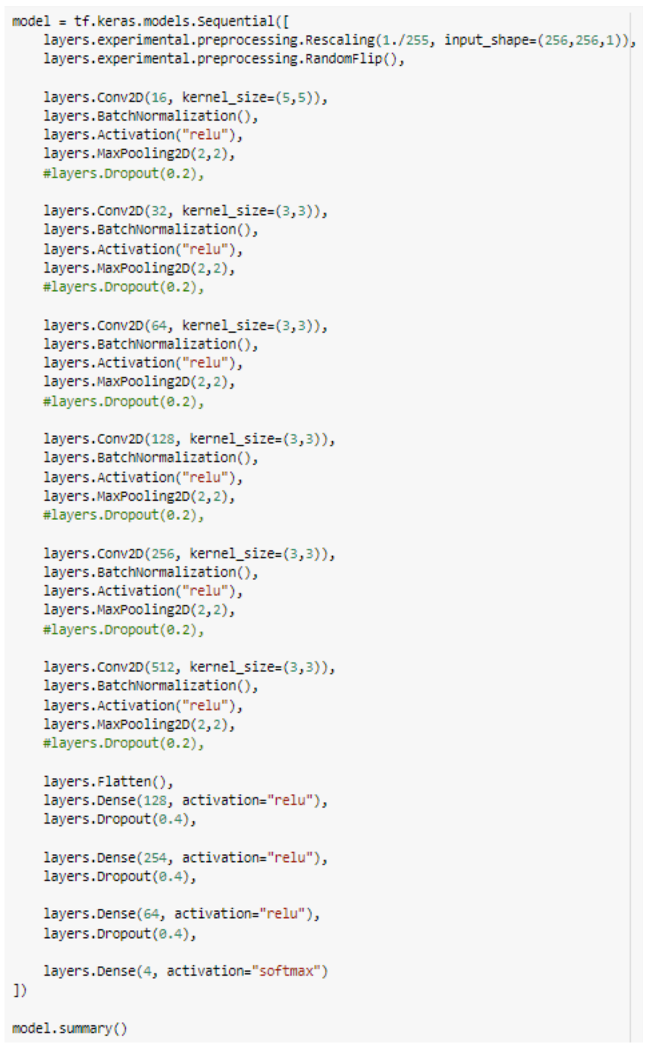
22/11/20 More confidence-temperature graphs

Aim: To create more confidence-temperature graphs for another model and some different phase transition videos

- Model trained for (I,N,N*,Sm) classification
- Dataset used was All_images_17-11-20-(I,N,Chol,Sm) (all images to date of these phases, with some images deleted in order create a more balanced dataset)

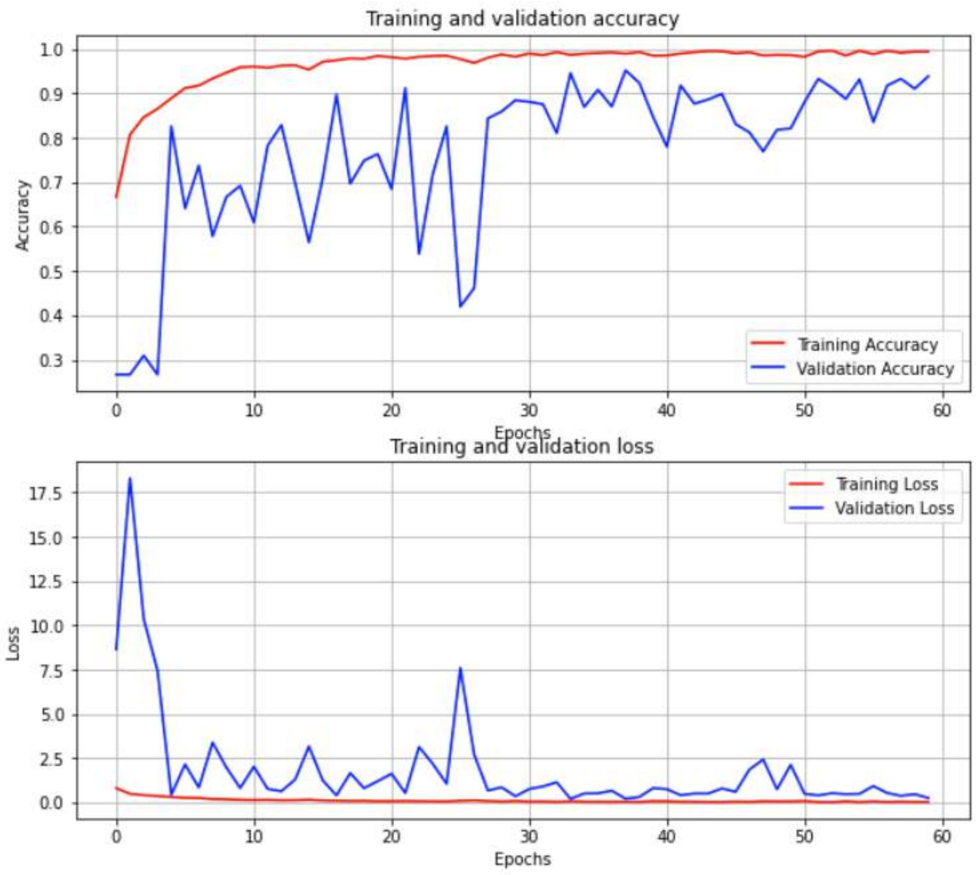
• Phase	• Train	• Validation
• Iso	• 727	• 440
• Nem	• 1587	• 503
• Chol	• 1333	• 354
• Sm	• 1916	• 591

-



```
model = tf.keras.models.Sequential([
    layers.experimental.preprocessing.Rescaling(1./255, input_shape=(256,256,1)),
    layers.experimental.preprocessing.RandomFlip(),

    layers.Conv2D(16, kernel_size=(5,5)),
    layers.BatchNormalization(),
    layers.Activation("relu"),
    layers.MaxPooling2D(2,2),
    #layers.Dropout(0.2),

    layers.Conv2D(32, kernel_size=(3,3)),
    layers.BatchNormalization(),
    layers.Activation("relu"),
    layers.MaxPooling2D(2,2),
    #layers.Dropout(0.2),

    layers.Conv2D(64, kernel_size=(3,3)),
    layers.BatchNormalization(),
    layers.Activation("relu"),
    layers.MaxPooling2D(2,2),
    #layers.Dropout(0.2),

    layers.Conv2D(128, kernel_size=(3,3)),
    layers.BatchNormalization(),
    layers.Activation("relu"),
    layers.MaxPooling2D(2,2),
    #layers.Dropout(0.2),

    layers.Conv2D(256, kernel_size=(3,3)),
    layers.BatchNormalization(),
    layers.Activation("relu"),
    layers.MaxPooling2D(2,2),
    #layers.Dropout(0.2),

    layers.Conv2D(512, kernel_size=(3,3)),
    layers.BatchNormalization(),
    layers.Activation("relu"),
    layers.MaxPooling2D(2,2),
    #layers.Dropout(0.2),

    layers.Flatten(),
    layers.Dense(128, activation="relu"),
    layers.Dropout(0.4),

    layers.Dense(254, activation="relu"),
    layers.Dropout(0.4),

    layers.Dense(64, activation="relu"),
    layers.Dropout(0.4),

    layers.Dense(4, activation="softmax")
])

model.summary()
```

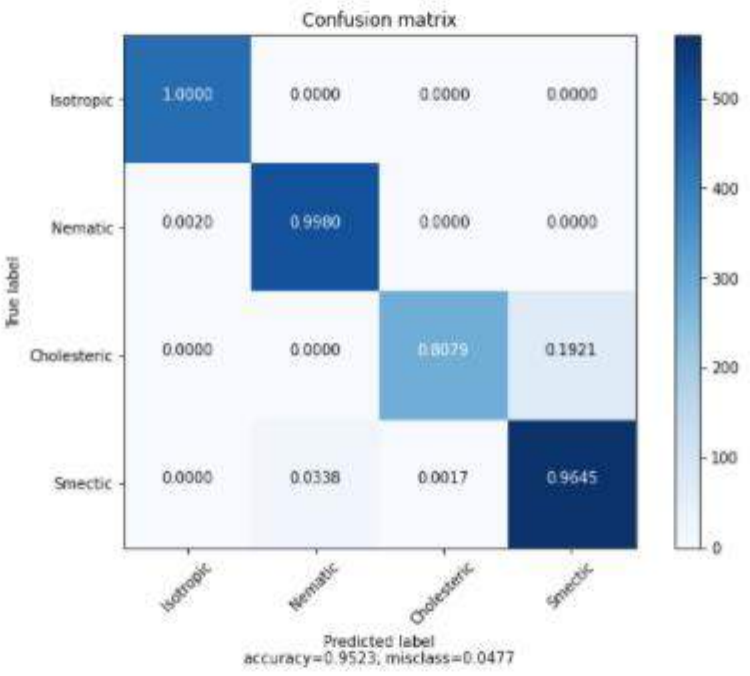
- Only random flip augmentation used.
- Training results, 60 epochs, default adam optimizer, avg 6s per epoch, 95.23% max validation accuracy.



-

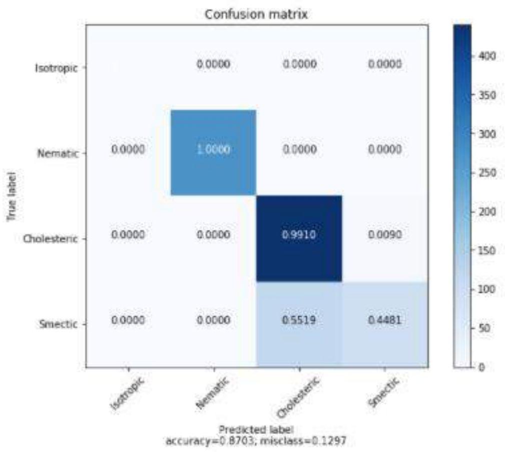
```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       440
           1       0.96      1.00      0.98       503
           2       1.00      0.81      0.89       354
           3       0.89      0.96      0.93       591

    accuracy                           0.95      1888
   macro avg       0.96      0.94      0.95      1888
weighted avg       0.96      0.95      0.95      1888
```

- on validation set
-

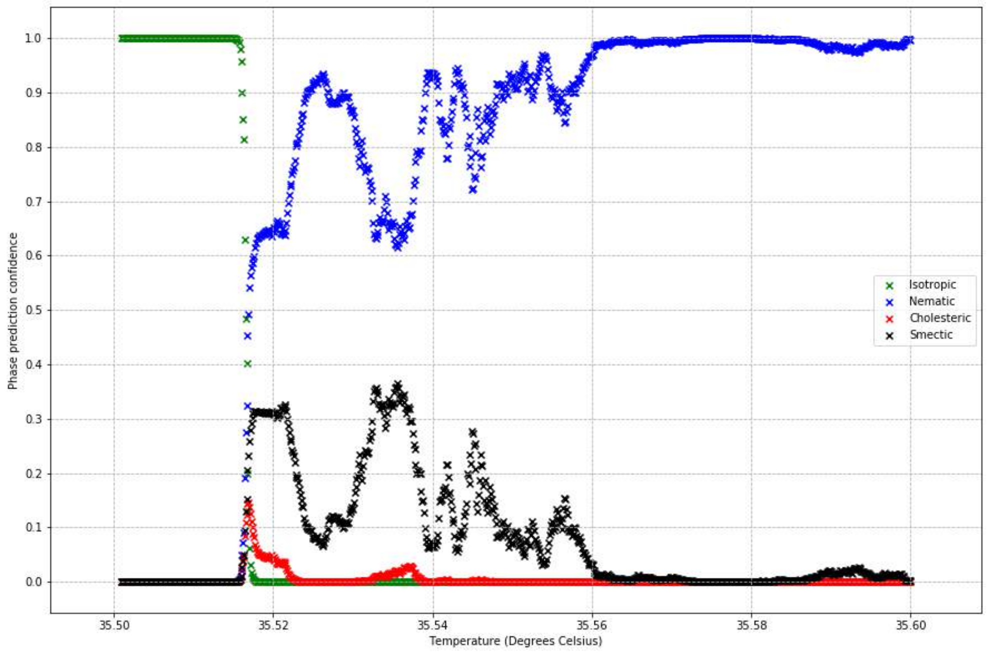


```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         1
           1       1.00      1.00      1.00       276
           2       0.79      0.99      0.88       444
           3       0.96      0.45      0.61       212

    accuracy                           0.87       933
   macro avg       0.94      0.86      0.87       933
weighted avg       0.89      0.87      0.85       933
```

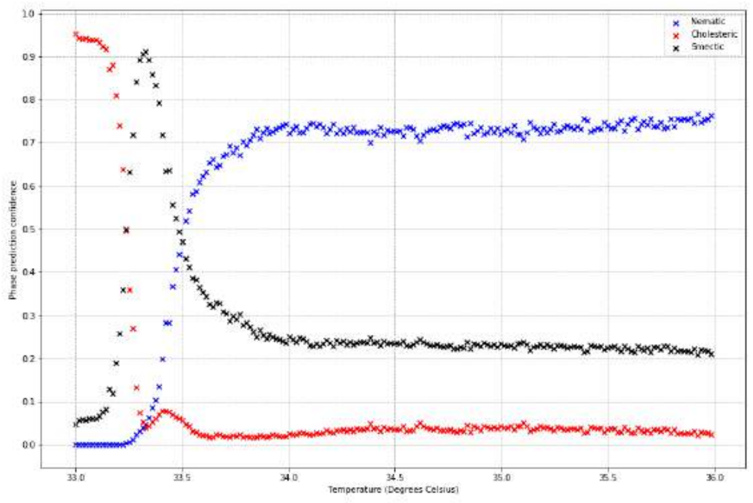
- on some frames taken from test set videos



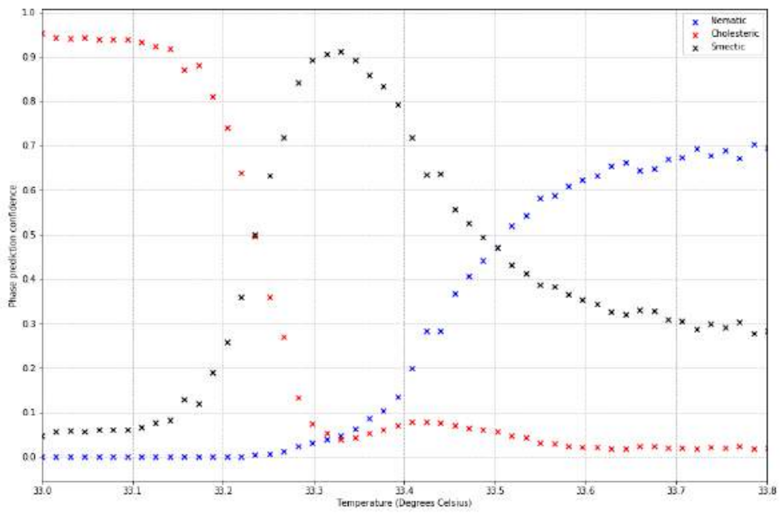
-
- Results of applying model to videos and getting confidence temperature graphs
- 5CB 35.6 cooling
- Correctly identifies Nematic to Isotropic transition



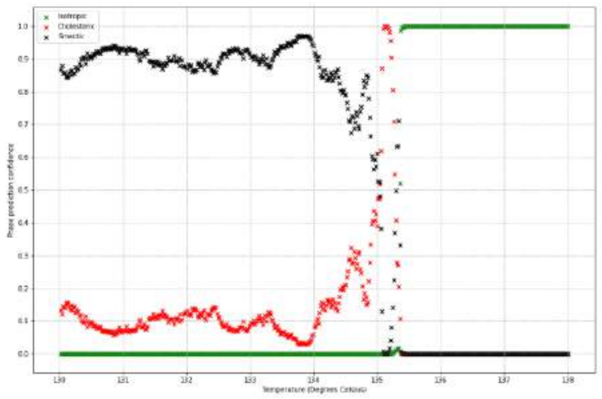
- 8CB 33-36 heating



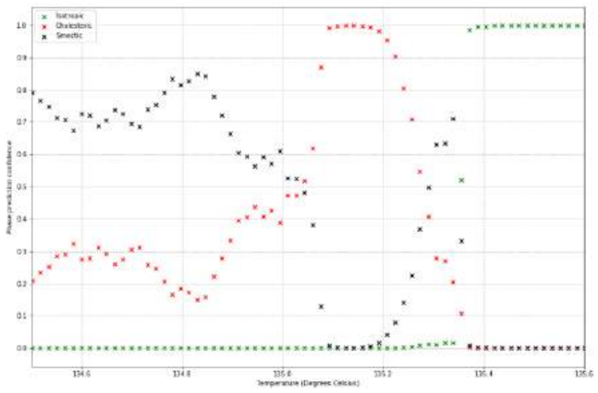
-



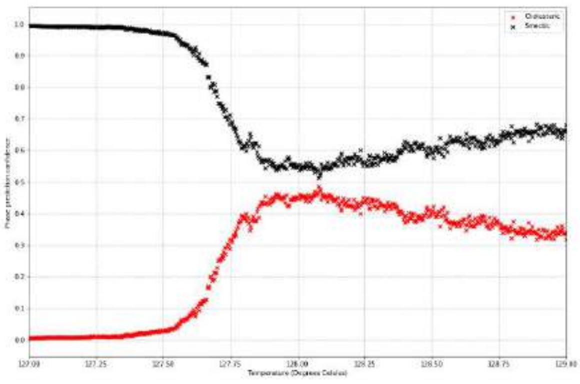
-
- Incorrectly starts at cholesteric when it should be Sm, switches quickly to Sm before correctly going to Nematic. Overall this is worse classified than with the worse model previously
- D6 138-130 cooling



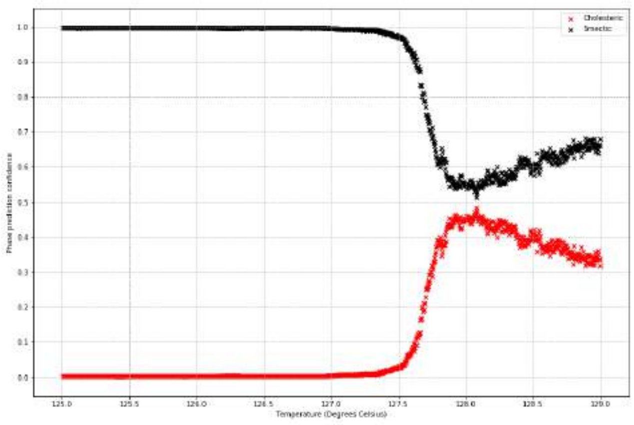
-



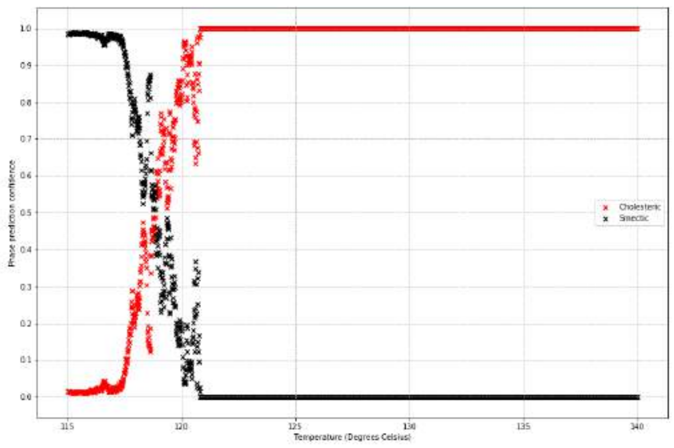
-
- Correctly identifies the transition from isotropic, but I am uncertain as to whether cholesteric or smectic is the correct classification, I expect I should have gone to cholesteric, however it does look smectic like to me
- D8 129-125 cooling



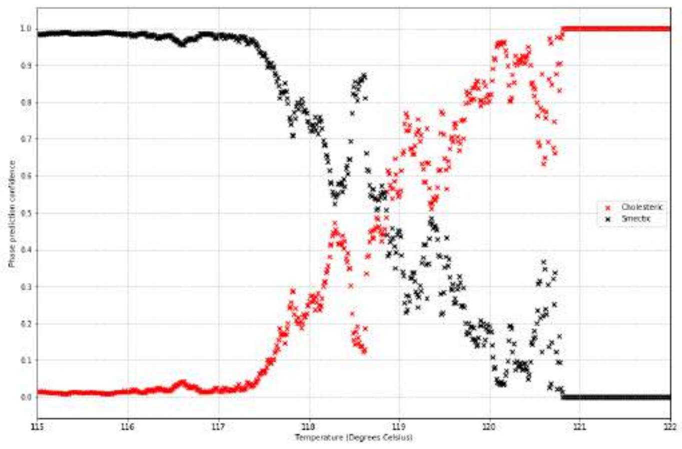
-



-
- Falsely identifies no transition, however while it is cholesteric, the confidence in the incorrect prediction of smectic is lower, then switches to being more certain when phase is actually smectic
- M5 140-115 cooling



-



-
- Correctly identifies cholesteric to smectic transition.

Summary: Overall, graphs show that, despite a very well perform model, videos weren't labelled particularly well, one highlight was M5 which was done well. No sign of the method being better for cooling rather than heating.

Week 7 summary:

- Created a plan for the MPhys report which I will send to Dr Dierking to get feedback on
- Made more confidence – temperature graphs with a better performing model, surprisingly despite a very well performing model on images, videos were not as well labelled.

24/11/20 8th meeting (video call)

Attendees: Dr. Dierking, Josh Heaton, James Harbon, Jason Dominguez

Discussed:

- Confirmed that I want to do this as a year long project
- Dr Dierking discussed possible extensions to work on next semester.
- For example: looking at the harder transitions, some particularly hard ones, such as nematic with axis parallel to POM light so it appears black (I pointed out that this may require a sequence model), looking at other types of liquid crystals such as lyotropic which are harder

Week 8 Aims:

- Send MPhys report plan to Dr Dierking for feedback
- Create datasets focused on SmC – SmA classification and/or Fluid smectic-Hexatic classification.
- May need transfer learning for Fluid smectic to hexatic due to very small dataset size. From past use of transfer learning a model such as VGG16 which isn't too deep might be best as high-level pretrained feature may not be useful in deeper networks.

25/11/20 Feedback from Dr Dierking

- Sent my report plan and got feedback from Dr Dierking, including that I should have an abstract

26/11/20 More videos from Dr Dierking

- Dr Dierking sent videos of some of the different things discussed in the last meeting, including nematic which appears black and isotropic to crystalline transitions.

29/11/20 Created datasets for SmC – SmA and Fluid Smectic to hexatic classification

Aim: Created datasets, split into training and validation, with test video images removed, for the classification for SmC and SmA and classification of Smectic or Hexatic.

- This is the split for the SmC to SmA dataset

• Phase	• Train	• Validation
• SmC	• 1232	• 156
• SmA	• 826	• 210

- There is a slight imbalance which can be sorted by removing more images if it causes problems such as learning to only detect one class.
- Due to small amount of data, only one test video was chose, D8-Green_cooling129-125C_5Cmin
- This is the split for the Fluid Sm to Hexatic dataset

• Phase	• Train	• Validation
• Fluid Sm	• 2413	• 114
• Hexatic	• 516	• 102

- A lot more imbalanced here.
- Due to small amount of data, only two test video were chose, D7-Green_cooling_95-90C_5Cmin and D8-Green_cooling_95-92_5Cmin, chosen as they shown different types of fluid sm to hexatic transition

Summary: Datasets were created and some signs of imbalance. They are quite small, so may benefit from transfer learning.

Week 8 summary:

- Created datasets for SmC and smA classification and fluid smectic to hexatic classification
- Unfortunately ran out of google colab gpu time before training reasonably good model on the data, but have to code to implement VGG16 transfer learning.
- Got feedback on report plan from Dr Dierking.

01/12/20 9th meeting (video call)

Attendees: Dr. Dierking, Josh Heaton, James Harbon, Jason Dominguez

Discussed:

- Other team showed using inception style network for fluid smectic vs hexatic vs soft crystal classification with 60-90% accuracy in some cases
- Dr Dierking said hes working on getting more images for us and James Harbon gave a list of prioritized phases
- Dr Dierking said it would be interesting for me to apply network that classifies nematic videos well on the nematic video which appears isotropic.

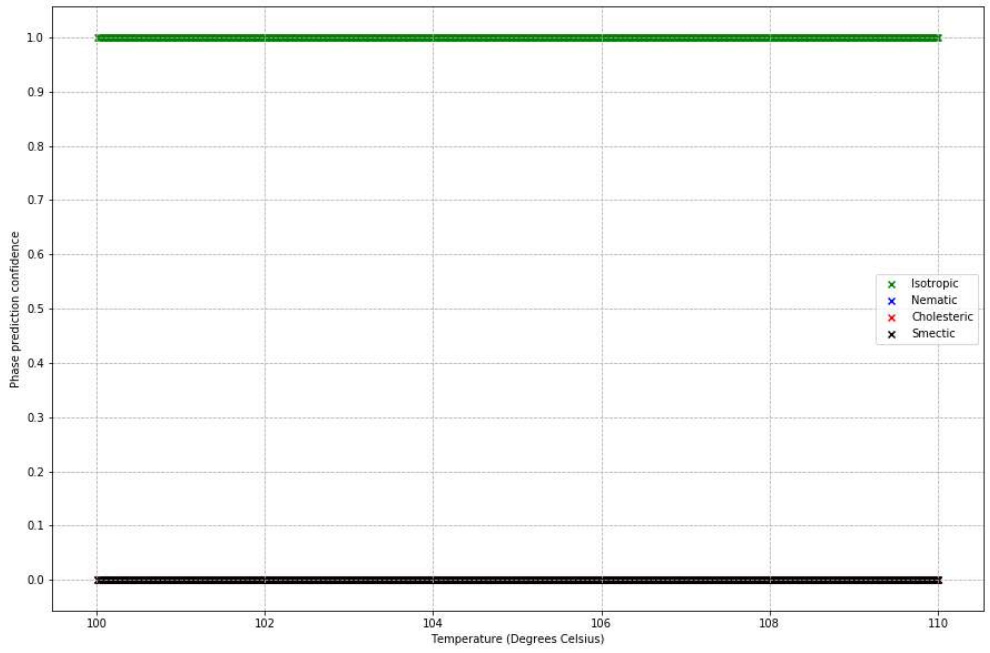
Week 9 Aims:

- Apply trained model on homeotropic nematic videos to see whether they are detected as nematic, or, as they are pseudo-isotropic, as isotropic.
- Apply trained model to crystalline videos, just to see what happens (expect it to predict transition to smectic, as this is the most ordered phase that it is trained for).
- After the success the other team have use an inception network, try an inception network for the I,N,N*,Sm classification.

Applied trained model from 16/11/20 (first CNN model) on homeotropic nematic and crystalline textures

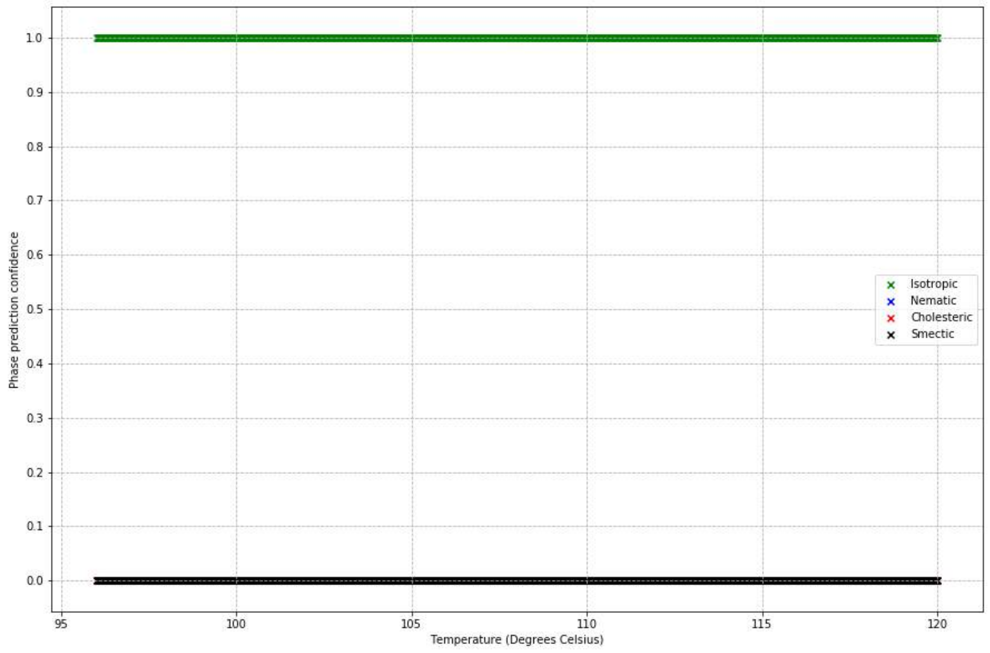
Aim: Apply trained model on homeotropic nematic textures to see if it can detect these. It is expected that they could be predicted as isotropic as they look very isotrpic, almost black. Also apply the model to crystalline videos just to see what it predicts, expected that it will predict smectic as this Is the most ordered phase that it has been trained for.

- Applied the trained model to homeotropic nematic videos, these are the resulting graphs.
- All videos found at: "C:\Users\Jason\Documents\University\Year_4\MPhys_project\Project_work\Results\Homeotropic_Nematic_results"
- Iso-N_homeo_Green_110-100_cooling

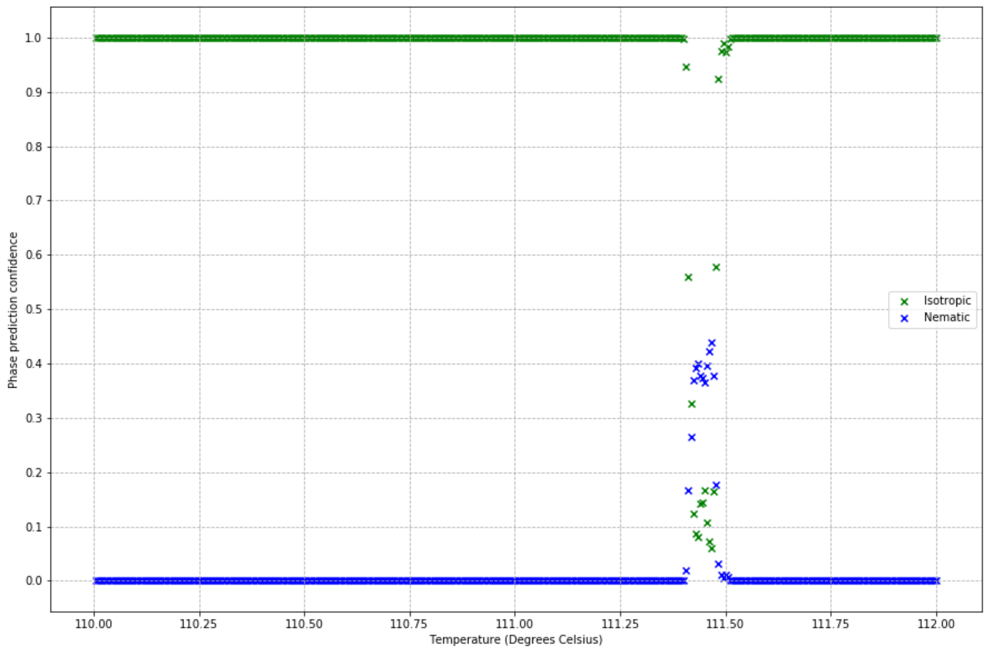


-
-

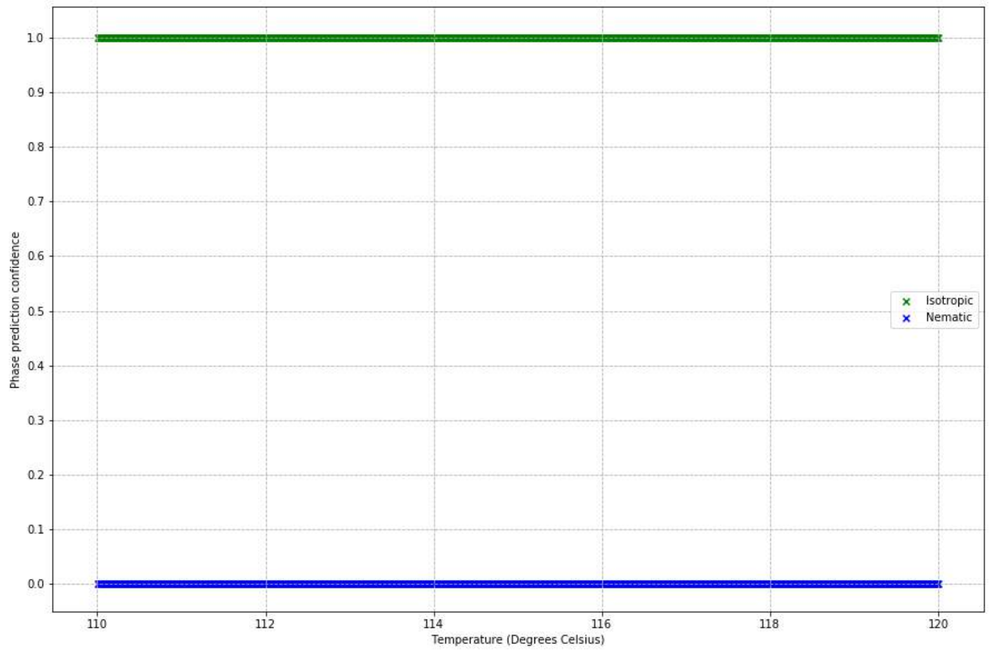
Iso-N_homeo_Green_112-110_cooling – This was briefly detected as nematic, before returning to an isotropic prediction



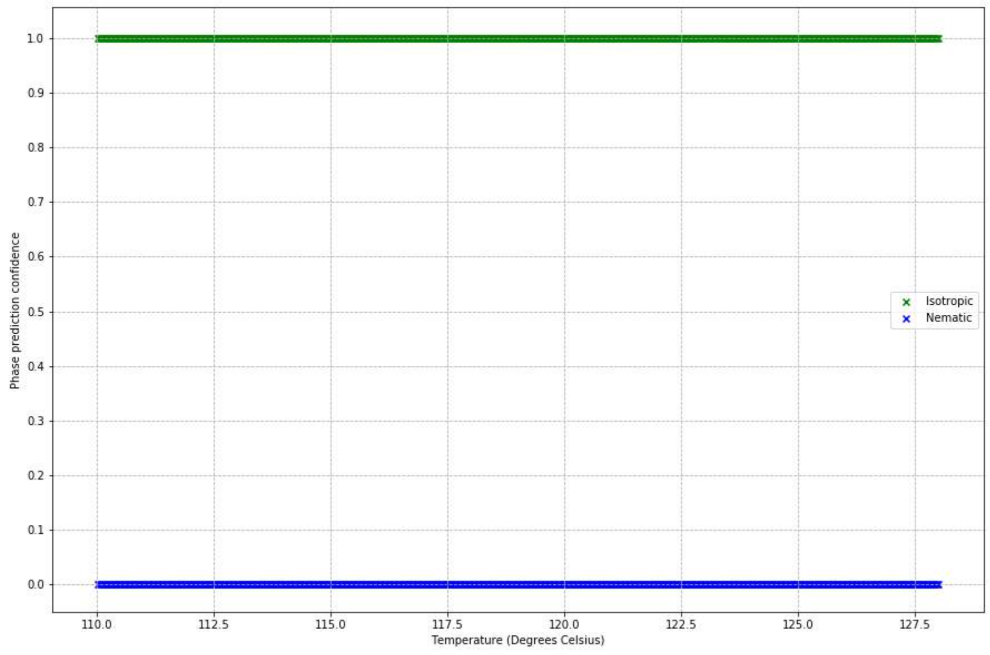
Iso-N_homeo_Green_120-96_cooling



Iso-N_homeo-Green_120-110_cooling

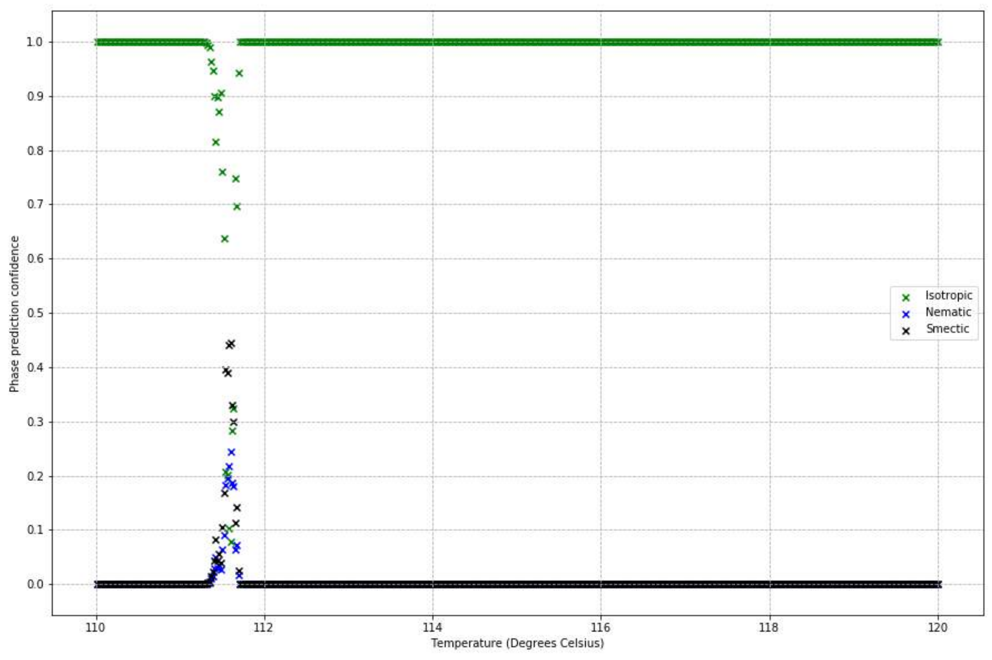


Iso-N_homeo-Green_120-110(2)_cooling



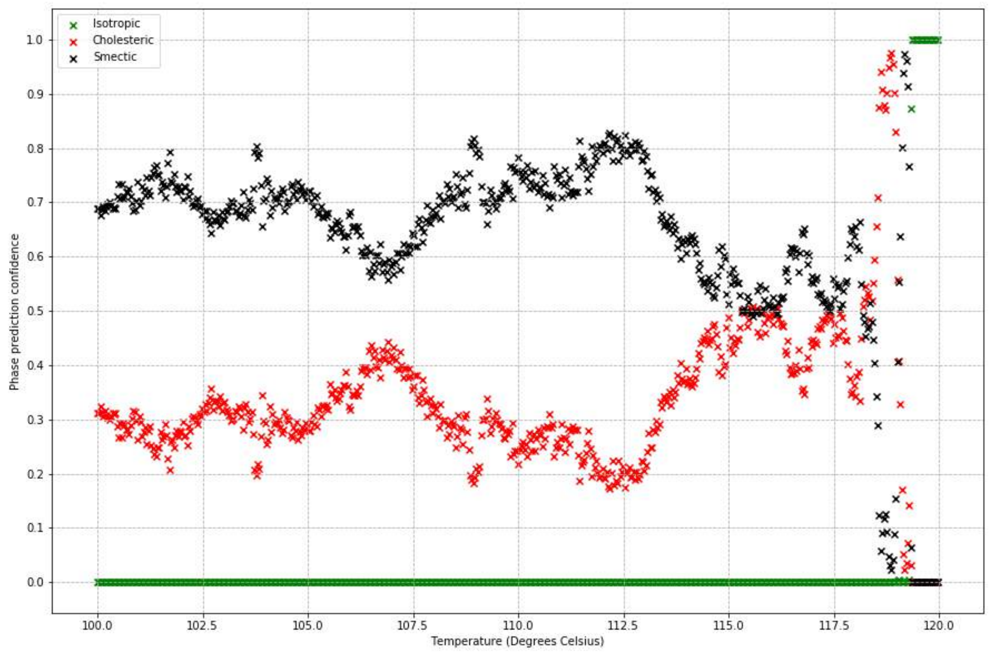
Iso-N_homeo-Green_128-110_cooling

Iso-N_homeo-Green_128-110_cooling

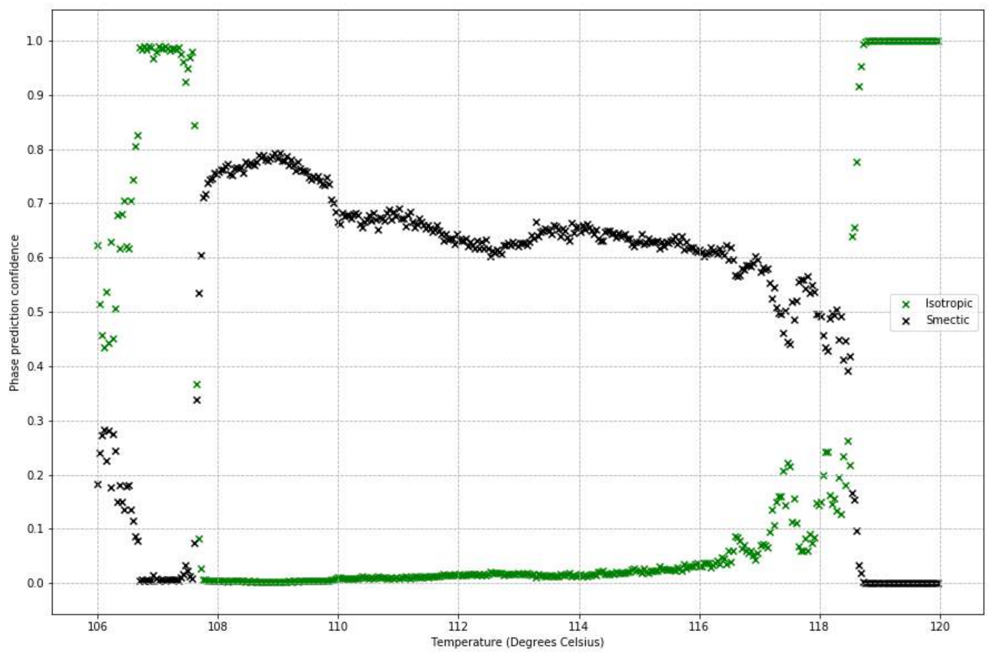


- Overall, the transitions were either missed completely, predicting isotropic the whole time, or a brief change in prediction when the texture increased in brightness, before returning to an isotropic prediction. These results suggest that the model has only learned dark means isotropic, which is understandable as no homeotropic nematic images were included in training or validation. To improve, include some of the images from these videos into the dataset.

Now the crystal transitions:

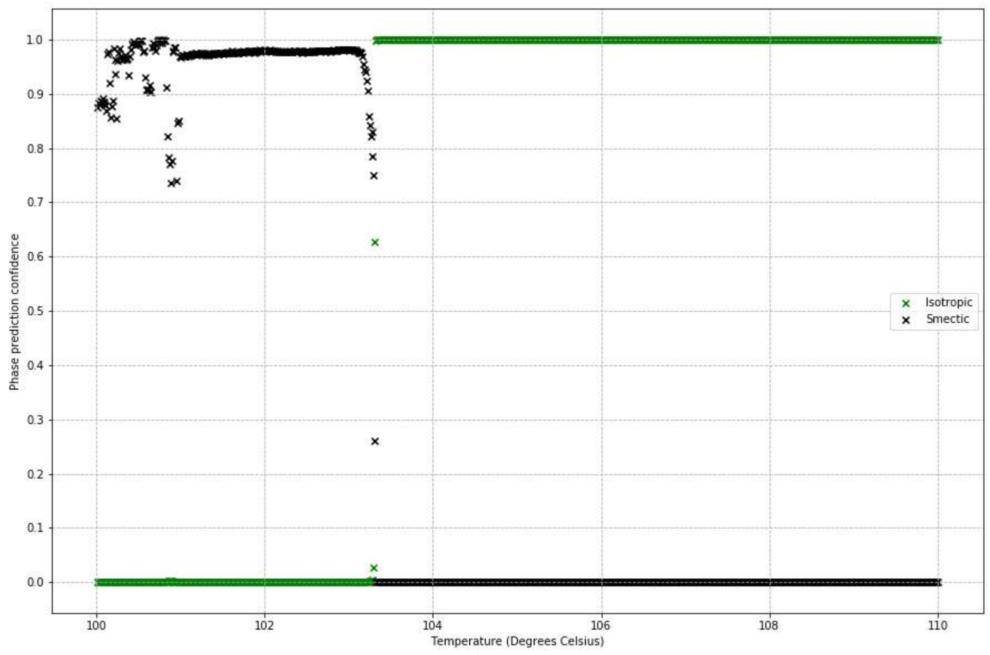


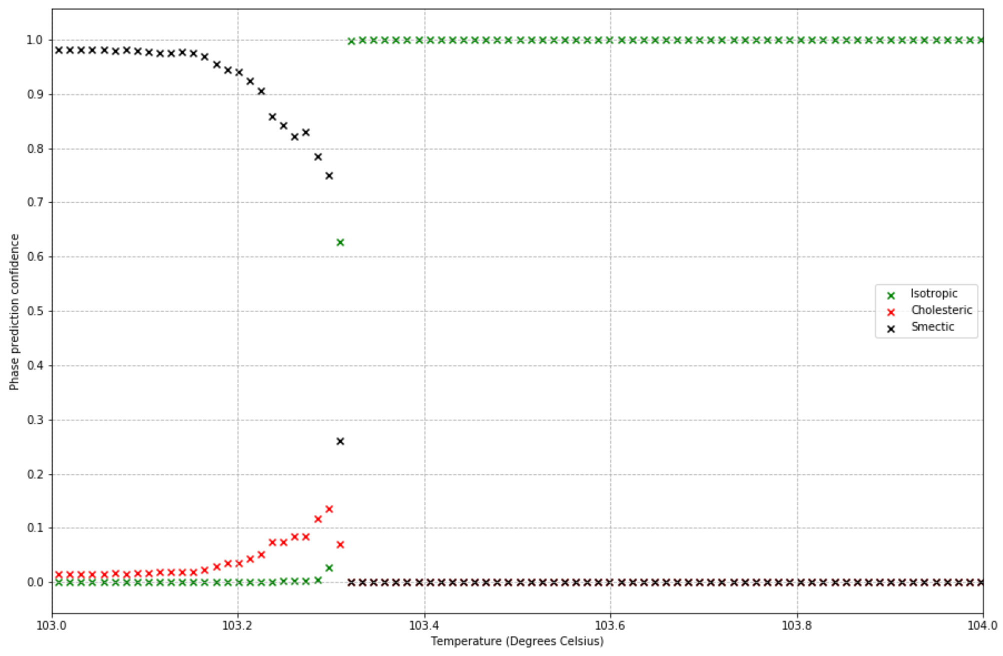
Iso-Crystal_Green_100-120_heating – by eye this got the transition to isotropic correct, with some confusion in phases around the transition. As expected, it predicted smectic when it was crystalline as it has not been trained for crystalline.



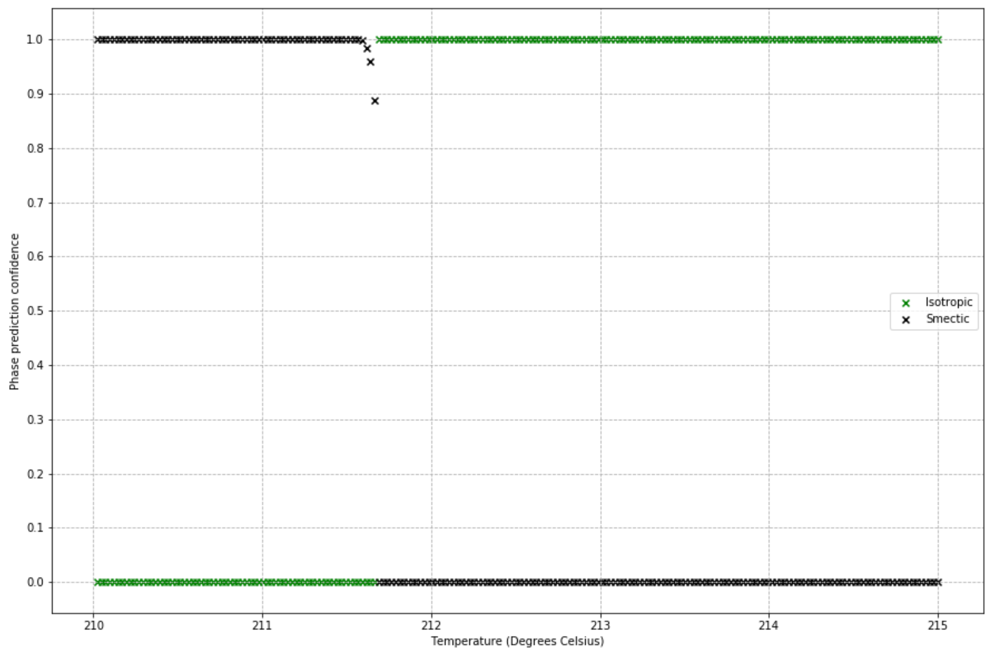
Iso-Crystal_Green_106-120_heating – Incorrectly initially predicts isotropic, possibly as this video was quite low brightness, again showing that the model has learned that dark means isotropic.

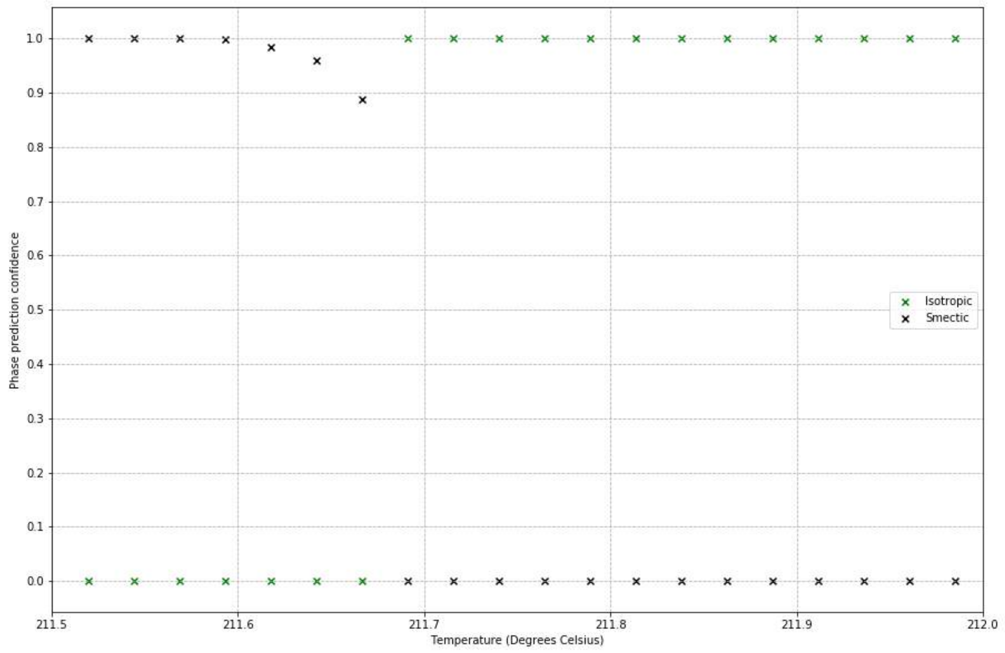
Possible could mitigate against this by including textures of different brightnesses, such as manually editing brightnedd of some images to reduce models dependance on texture brightness..





Iso-Crystal_NMe-Green_110-100_cooling – accurately detects isotropic to crystal transition , but as isotropic to smectic as expected. Bt of uncertainty at the lower T due to motion in the video causing blurriness.





Iso-Crystal_Green_215-210_cooling – As expected, again gets transition correct, as isotropic to smectic though.

Summary: As expected, crystalline textures were identified as smectic, as these were the most ordered phase it was trained for. Also, homeotropic nematic phases were mostly misrecognized as isotropic, but with a brief identification as nematic. To improve, homeotropic nematic images should be included in the dataset. Also could include crystalline image as it would be useful in a final model to be able to see not only what LC phase, but whether it is LC at all.

Week 9 summary:

- Applied previously trained model on homeotropic nematic and crystalline transition videos, results were as expected: crystalline detected as smectic as this was the most ordered phase the model was trained for, Homeotropic commonly mistaken as isotropic. Decided to include these images in future datasets.

08/12/20 10th meeting (video call)

Attendees: Dr. Dierking, Josh Heaton, James Harbon, Jason Dominguez

Discussed:

- Showed the results of crystalline, homeotropic nematic labelled videos.

Week 10 Aims:

- Use all data so far to try an inception model for I,N,N*,Sm classification, to see if it improve on best CNN model so far.
- Train a model for SmA-SmC classification
- Train a model for fluid smectic – hexatic classification

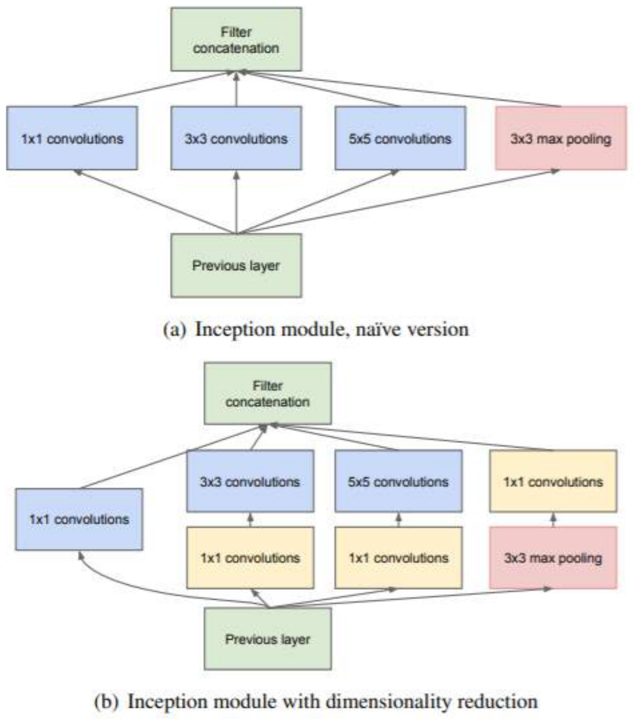
09/12/20 Training different networks for I,N,N*,Sm classification:

Aim: Train different model architectures for I,N,N*,Sm to systematically find a best architecture for this classification problem, using all images so far.

- Made a new dataset of images for this classification problem, using all images obtained so far, deleted images of some phases to make more balanced, for example there were 3299 sm images. Includes homeotropic nematic images.

• Phase	• Train	• Validation
• Iso	• 1446	• 259
• Nem	• 1367	• 392
• Chol	• 1547	• 352
• Sm	• 1499	• 543

- Test videos were chosen to be the same as for 16/11/20 to see if performance can improve, especially for using new inception network.
- Diagram of inception network, the network is described in this paper https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/43022.pdf. Essentially have different types of layers in parallel.



- Figure 2: Inception module

  Figure from that paper, of inception module which is using instead of a single layer in CNN. 1x1, used to reduce number of computations.
- Different model architectures were tried, CNNs and Inception networks, with results repeated three times, these are shown in the table below,

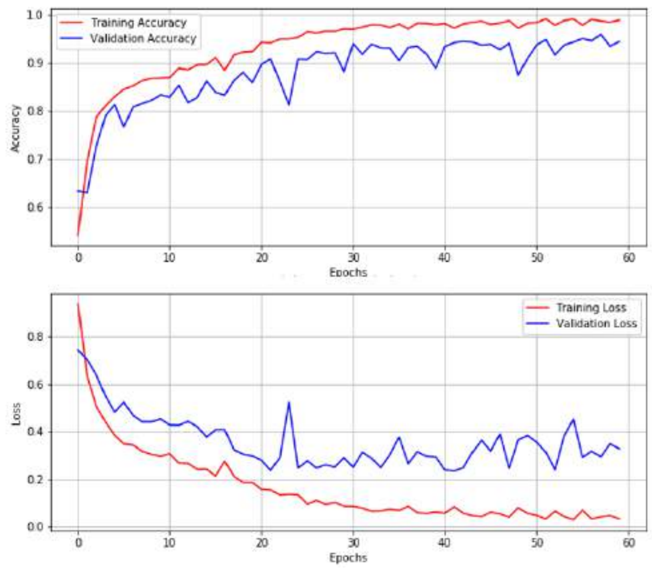
MODEL NAME	LAYERS	EXTRA FEATURES	MAXIMUM VALIDATION ACCURACY			
			1	2	3	Error
CONVNETLC-V1	CP,CP,FC,O	None	0.75355763	0.705558	0.733675	0.048
CONVNETLC-V2	CP,CP,FC,O	Flip, 0.5 Dropout	0.69766785	0.744502	0.672502	0.072
CONVNETLC-V3	CP,CP,CP,FC,O	None	0.8034221	0.813713	0.7307128	0.083
CONVNETLC-V4	CP,CP,CP,FC,O	Flip, 0.3 Dropout	0.828589916	0.75159	0.7778901	0.077
CONVNETLC-V5	CP,CP,CP,CP,FC,FC,O	None	0.824708939	0.887109	0.7807089	0.044
CONVNETLC-V6	CP,CP,CP,CP,FC,FC,O	Flip, 0.5 Dropout	0.828566596	0.857467	0.8945666	0.066
INCEPTIONNETLC-V1	CP,I,P,FC,O	None	0.906678	0.90815	0.8561501	0.052
INCEPTIONNETLC-V2	CP,I,P,FC,O	Flip, 0.5 Dropout	0.879689515	0.816689515	0.849712	0.063
INCEPTIONNETLC-V3	CP,I,I,P,FC,O	None	0.958602846	0.902603	0.9466557	0.056
INCEPTIONNETLC-V4	CP,I,I,P,FC,O	Flip, 0.5 Dropout	0.906602846	0.910022	0.9586028	0.052

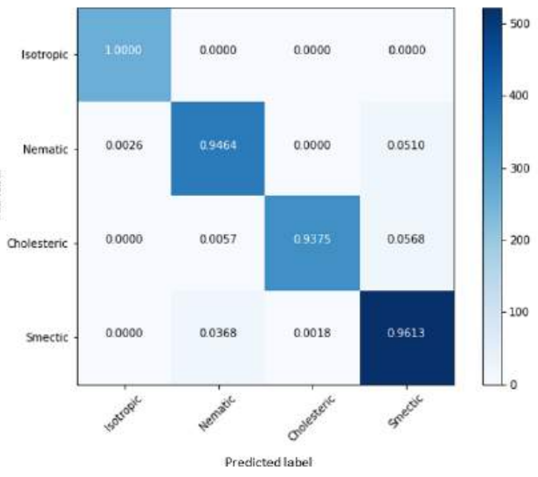
MODEL NAME	MAXIMUM VALIDATION ACCURACY	TRAINING ACCURACY ON EPOCH OF MAXIMUM VALIDATION	MAXIMUM VALIDATION LOSS	EPOCH OF MAXIMUM VALIDATION ACCURACY	TIME PER EPOCH (S)
CONVNETLC-V1	0.75355763	0.999829352	1.237751603	7	3
CONVNETLC-V2	0.744502	0.655572653	0.641016	18	3
CONVNETLC-V3	0.813712835	0.991295457	1.002240896	3	6
CONVNETLC-V4	0.828589916	0.882744491	0.72082907	23	6
CONVNETLC-V5	0.887109	0.977470577	1.33101964	4	6
CONVNETLC-V6	0.894566596	0.871309102	0.431761026	45	6
INCEPTIONNETLC-V1	0.908150077	0.983785629	0.573254883	57	35
INCEPTIONNETLC-V2	0.879689515	0.878989577	0.368283689	46	35
INCEPTIONNETLC-V3	0.958602846	0.989076614	0.219371736	42	90
INCEPTIONNETLC-V4	0.958602846	0.986687124	0.294466287	57	90

- For the best validation accuracy epoch, further results were taken to see how much overfitting was occurring and other useful quantities such as time per epoch were recorded

- Results show that increasing layer of CNN help achieve higher validation accuracy, but most importantly, inception networks show the most accurate performance. These do take a lot longer to train, however these times are still manageable, taking a maximum of a couple of hours to complete. The InceptionLC-V4 model and V3 performed the best on the validation set, with minimal overfitting, shown by the only slightly higher training accuracy and small loss.

- Here is the training graph and confusion matrix for the Inception V4 model





-
- Slight over fitting is starting to occur, seen by the loss starting to flatten out and even increase, however this is only slight.
- InceptionLC-V4 was chosen as the model to be used on test videos due to its high validation accuracy, here are the results on the 8CB and M6 videos.
- All videos and the model code found in "C:\Users\Jason\Documents\University\Year_4\MPhys_project\Project_work\Data\I-N-Chol-Sm_dataset-balanced\Results\InceptionNets\InceptionNetLC-V4(6)\"
- Correct identification of N-Sm transitions at 33 +/- 1 C, uncertainty taken from the absolute temperature uncertainty of the device used for controlling temperature of the liquid

crystals.

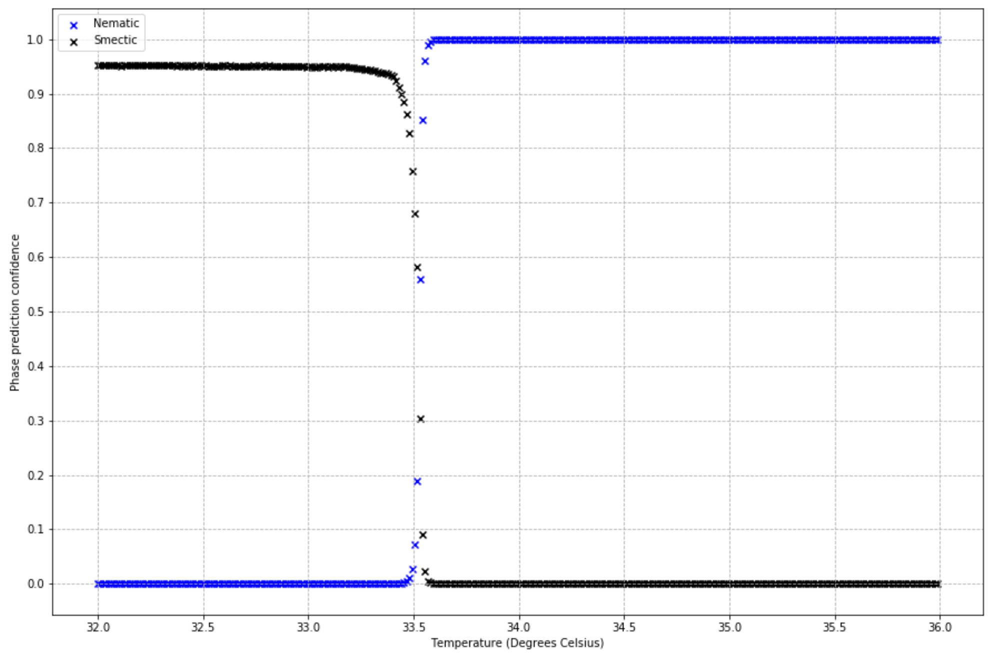
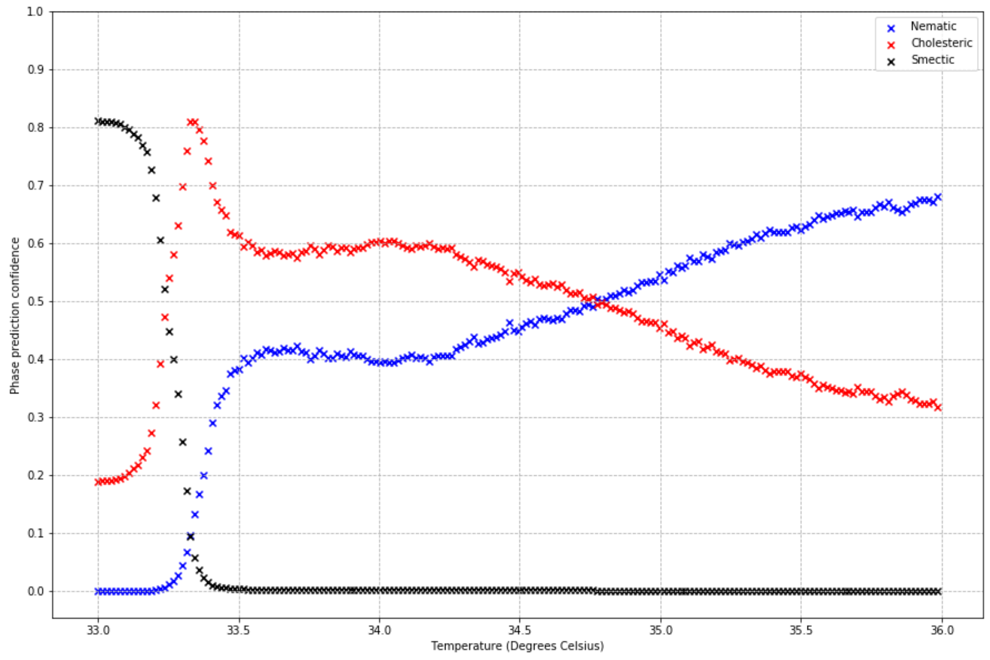


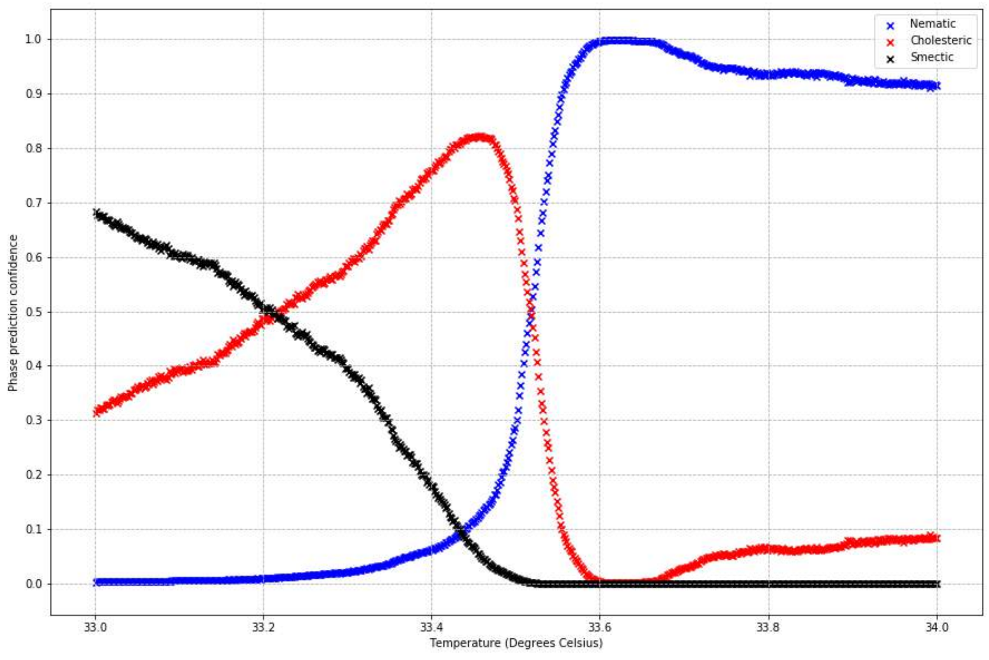
Figure 1 8CB-Yellow_32-36C_heating_10Cmin_Inception



-

Figure 2 8CB-Yellow_33-36C_heating_10Cmin_2_Inception

-



-

Figure 3 8CB-Yellow_34-33C_cooling_5Cmin_Inception

- Some videos wer labelled correctly, but with confusion with cholesteric phase at the transition.

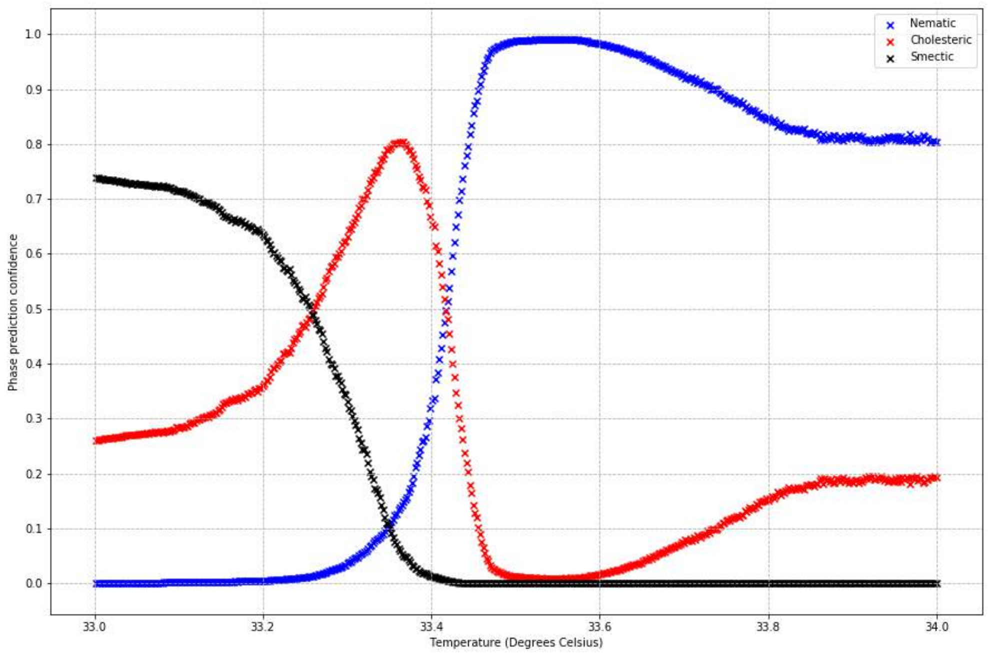
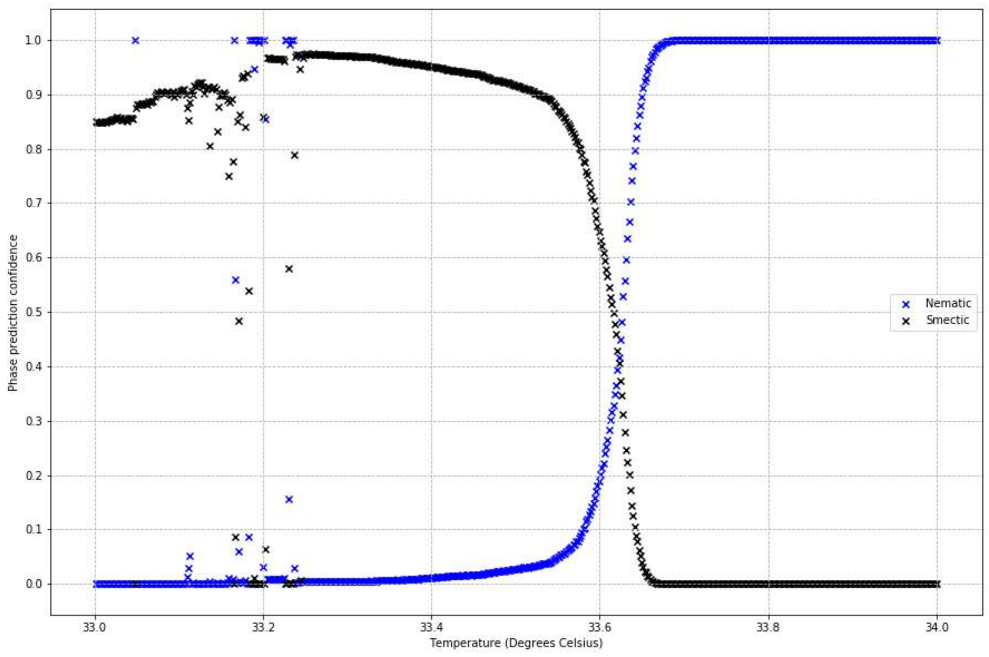
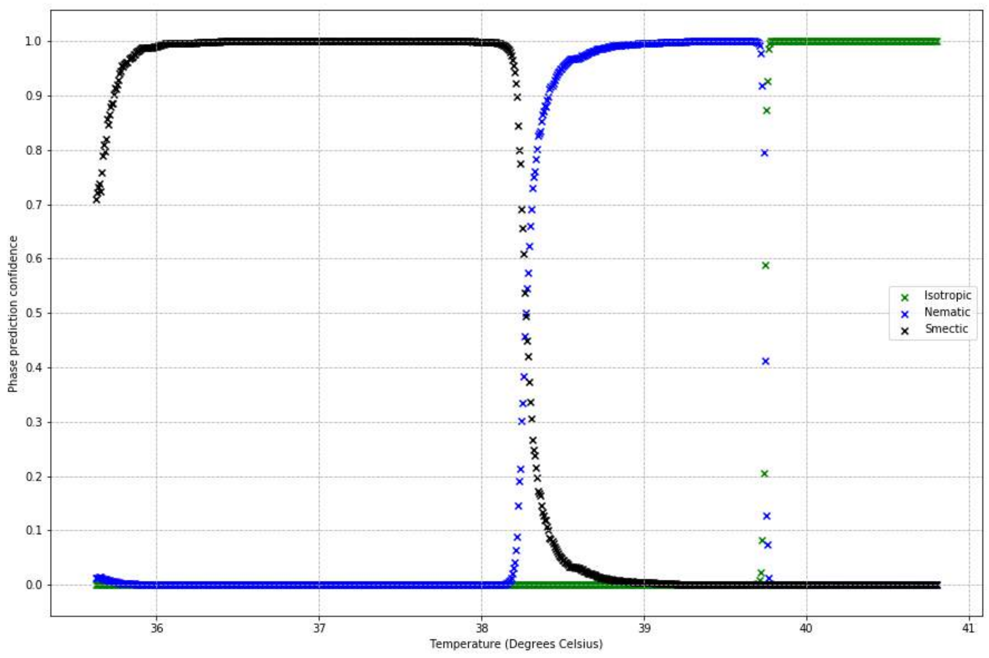


Figure 4 8CB-Yellow_34-33C_cooling_5Cmin_3_Inception



-

Figure 5 8CB-Yellow_34-33C_cooling_5Cmin_2_Inception lower temperature confusion was due to motion in the video, causing blurriness



-

Figure 6 8CB-Yellow_40.8C_cooling_0.1Cmin_Inception

Figure 7Incorrectly detects a transition to smectic which does not occur, but does correctly identify transition from isotropic to nematic at 40 +/- 1 C (error from device)

Here isotopic to smectic phase transition is correctly identified, however with a lot of confusion in the transition from this a transition temp of 32 +/- 4 C where 4 C comes from the approximate spread of the confusion region.

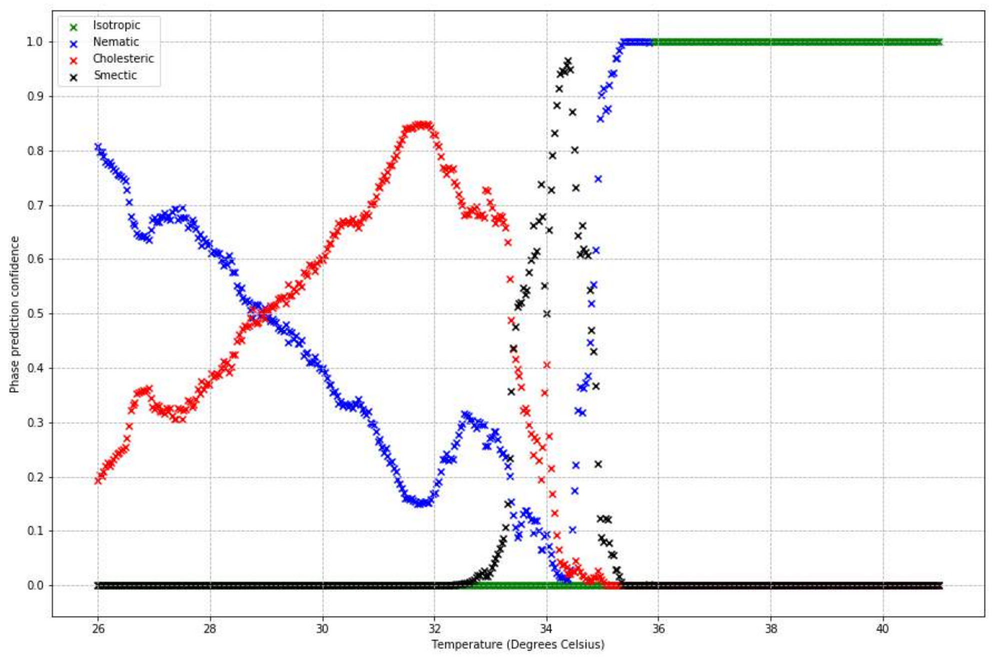


Figure 8 8CB-Yellow_41C_5Cmin_cooling_Inception

M6 Videos:

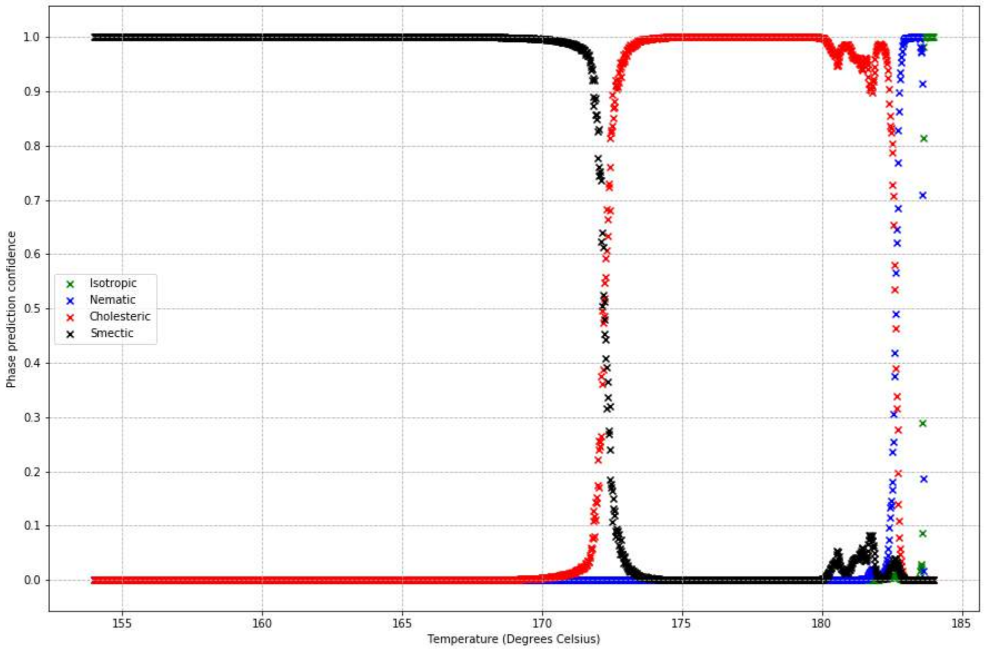


Figure 9 M6-Green_cooling_175-115C_10Cmin_Inception

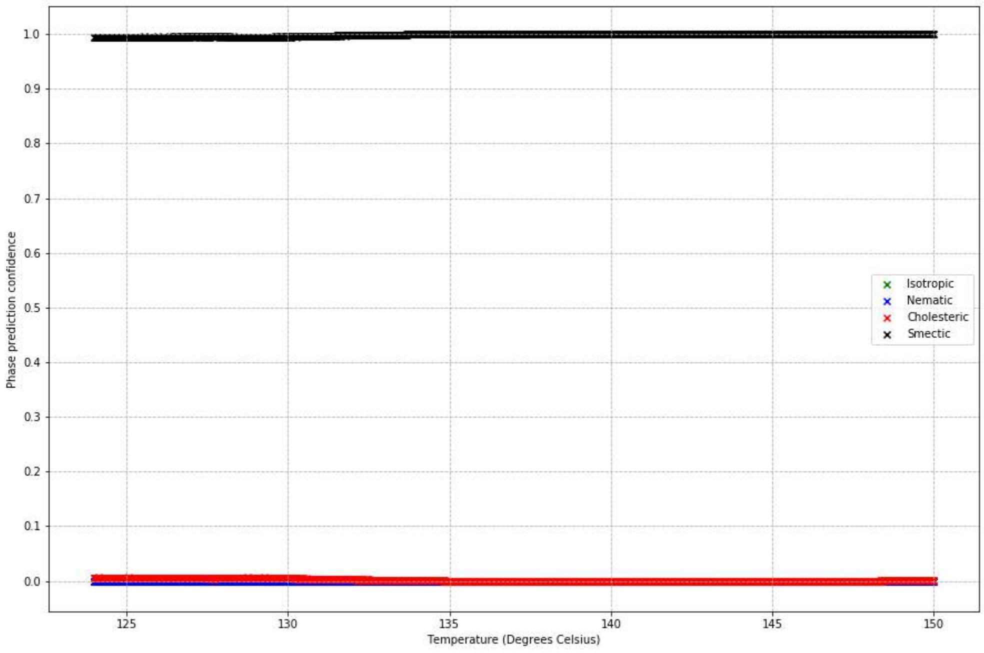


Figure 10 M6-Green_cooling_100-85C_10Cmin_Inception

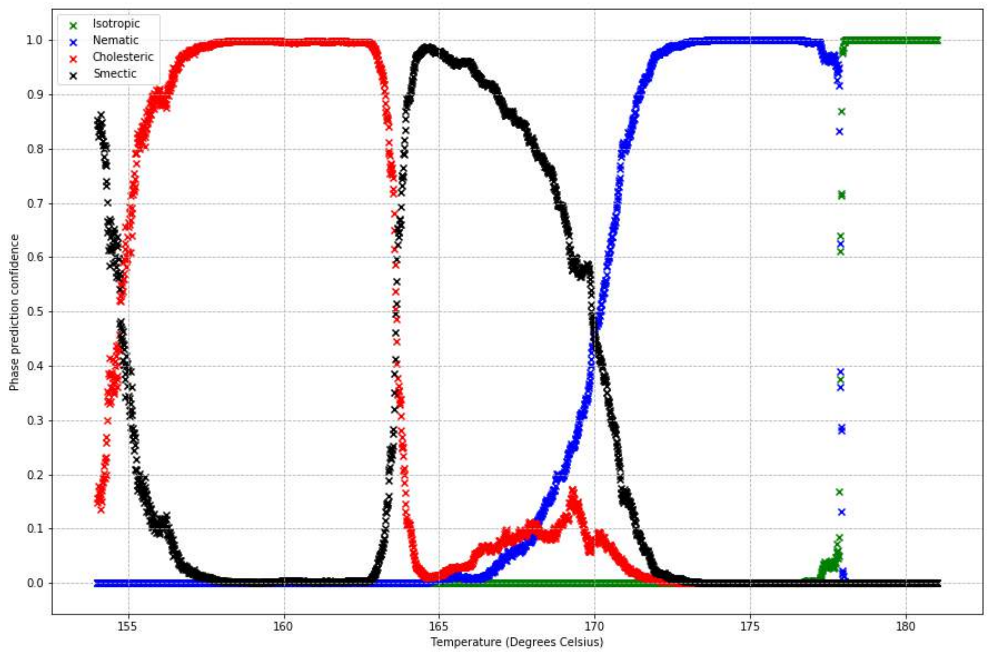


Figure 11 M6-Green_cooling_181-154C_10Cmin_Inception

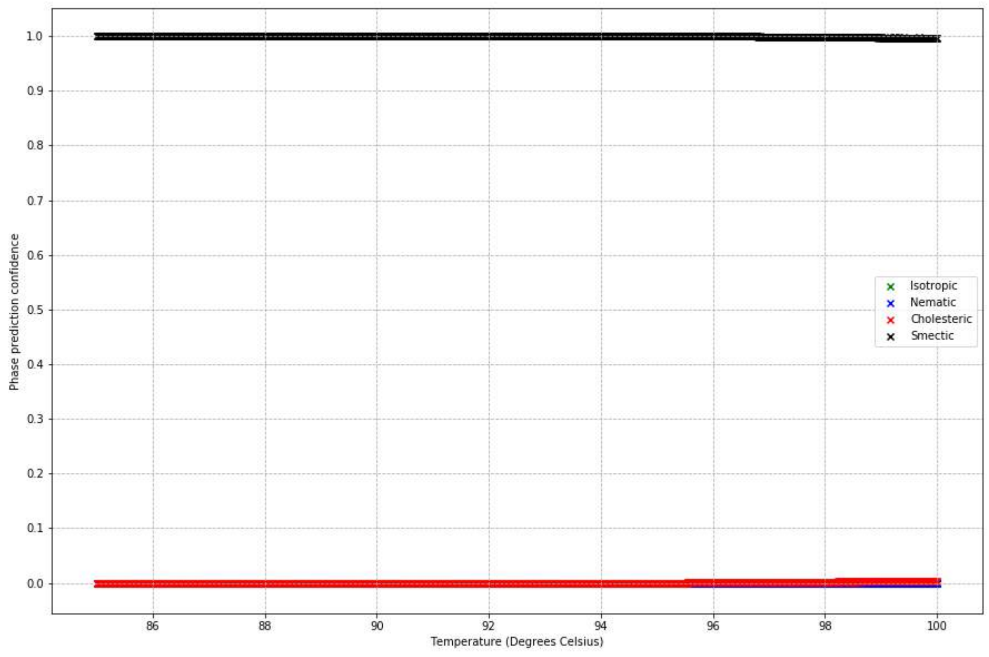
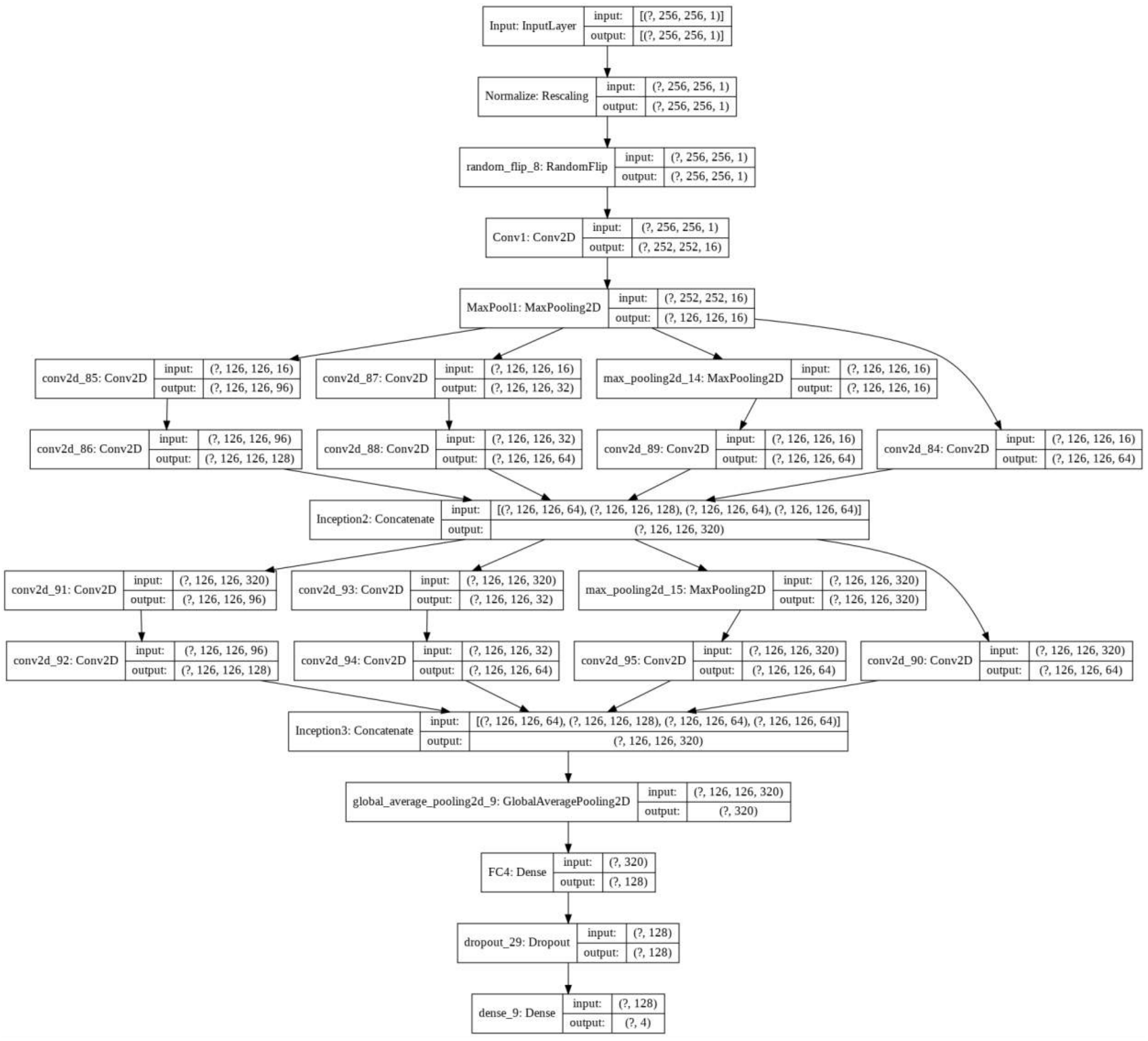


Figure 12 M6-Green_heating_124_150C_10Cmin_Inception

In general, the results from these graphs is that sometimes smectic-cholesteric transiton are missed, or transition are detected where there are none. Transitions between isotropic and cholesteric were correctly identified, with some confusion with nematic, possibly due to the inclusion of homeotropic nematic images in the dataset, meaning the model learned that low brightness but not black textures were nematic.

Here is the inception architecture used


Input: InputLayer | input: [(?, 256, 256, 1)] | output: [(?, 256, 256, 1)]
Normalize: Rescaling | input: (?, 256, 256, 1) | output: (?, 256, 256, 1)
random_flip_8: RandomFlip | input: (?, 256, 256, 1) | output: (?, 256, 256, 1)
Conv1: Conv2D | input: (?, 256, 256, 1) | output: (?, 252, 252, 16)
MaxPool1: MaxPooling2D | input: (?, 252, 252, 16) | output: (?, 126, 126, 16)
conv2d_85: Conv2D | input: (?, 126, 126, 16) | output: (?, 126, 126, 96)
conv2d_87: Conv2D | input: (?, 126, 126, 16) | output: (?, 126, 126, 32)
max_pooling2d_14: MaxPooling2D | input: (?, 126, 126, 16) | output: (?, 126, 126, 16)
conv2d_86: Conv2D | input: (?, 126, 126, 96) | output: (?, 126, 126, 128)
conv2d_88: Conv2D | input: (?, 126, 126, 32) | output: (?, 126, 126, 64)
conv2d_89: Conv2D | input: (?, 126, 126, 16) | output: (?, 126, 126, 64)
conv2d_84: Conv2D | input: (?, 126, 126, 16) | output: (?, 126, 126, 64)
Inception2: Concatenate | input: [(?, 126, 126, 64), (?, 126, 126, 128), (?, 126, 126, 64), (?, 126, 126, 64)] | output: (?, 126, 126, 320)
conv2d_91: Conv2D | input: (?, 126, 126, 320) | output: (?, 126, 126, 96)
conv2d_93: Conv2D | input: (?, 126, 126, 320) | output: (?, 126, 126, 32)
max_pooling2d_15: MaxPooling2D | input: (?, 126, 126, 320) | output: (?, 126, 126, 320)
conv2d_92: Conv2D | input: (?, 126, 126, 96) | output: (?, 126, 126, 128)
conv2d_94: Conv2D | input: (?, 126, 126, 32) | output: (?, 126, 126, 64)
conv2d_95: Conv2D | input: (?, 126, 126, 320) | output: (?, 126, 126, 64)
conv2d_90: Conv2D | input: (?, 126, 126, 320) | output: (?, 126, 126, 64)
Inception3: Concatenate | input: [(?, 126, 126, 64), (?, 126, 126, 128), (?, 126, 126, 64), (?, 126, 126, 64)] | output: (?, 126, 126, 320)
global_average_pooling2d_9: GlobalAveragePooling2D | input: (?, 126, 126, 320) | output: (?, 320)
FC4: Dense | input: (?, 320) | output: (?, 128)
dropout_29: Dropout | input: (?, 128) | output: (?, 128)
dense_9: Dense | input: (?, 128) | output: (?, 4)

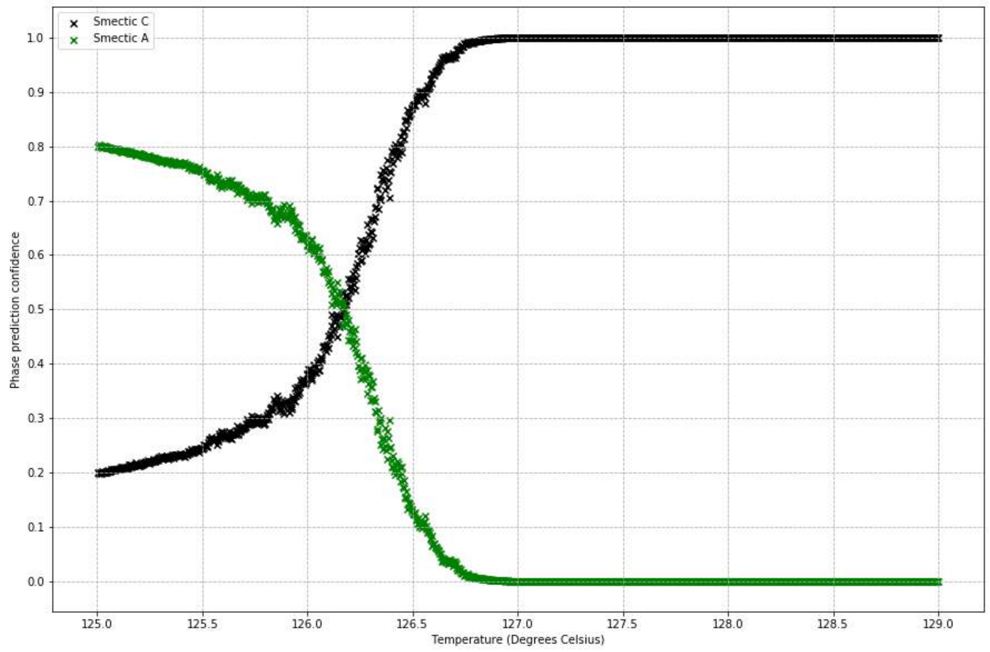
Summary: Overall, performance of this model was similar to the previous best model, the overfitting model from 16/11/20. Although some nematic-smectic transitions was misclassified as cholesteric at the transition, slightly reducing the precision. This is surprising due to the much better validation accuracy achieved by this model. This shows overfitting to the validation set, showing that the method of choosing a random validation set may not be best as these images may not fully represent the test set video textures. Could use an alternative method, called k-cross fold validation, this involves creating several datasets, with different validation sets. The model is then training and validation on these (so in affect you train a model k times, which increases computation time) and if the model performs well on all the validation sets, then it has been validated on more datasets, so is more likely to generalize well.

11/12/20 Applied inception network to SmA-SmC classification (done on google colab using gpu)

Aim: Train an inception model for the classification of SmA-SmC textures then apply trained model to test videos to identify transition. Using dataset from 29/11/20.

- Based on the above results, decided to use the IneptionLC-V4 model architecture, but with the output changed to a binary output, as this is binary classification.
- All results and code found at: "C:\Users\Jason\Documents\University\Year_4\MPhys_project\Project_work\Results\SmC-SmA_results"
-

- D8-Green_cooling_129-125C_5Cmin_labelled



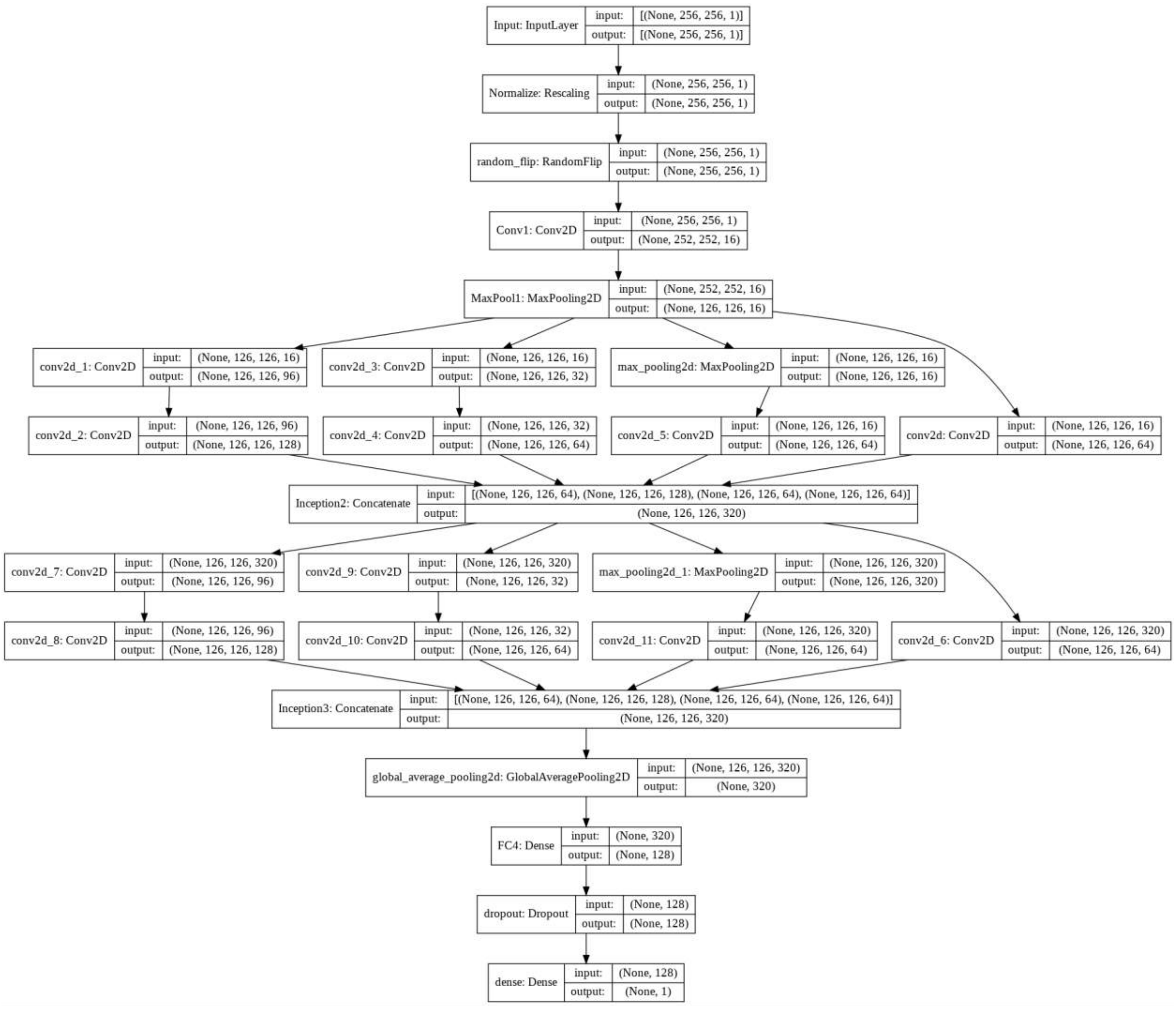
- There was only one test video with this dataset, so results aren't conclusive. Hopefully can get more SMA and SmC images to allow for more test videos to be kept out of training. But overall this predicted the transition slightly delayed and got it incorrect, SmC when it was SmA and vice versa. This could be a mistake, maybe I put the images in folders and labelled them wrong. I will check this.

Summary: Although I initially thought the transition was guessed correctly, Dr Dierking suggested that it was a little inaccurate, but also pointed out that it was getting the transition the wrong way round. I will check that I haven't made a mistake when putting images into folder. For example I might have put SmA into a folder labelled SmC and vice versa.

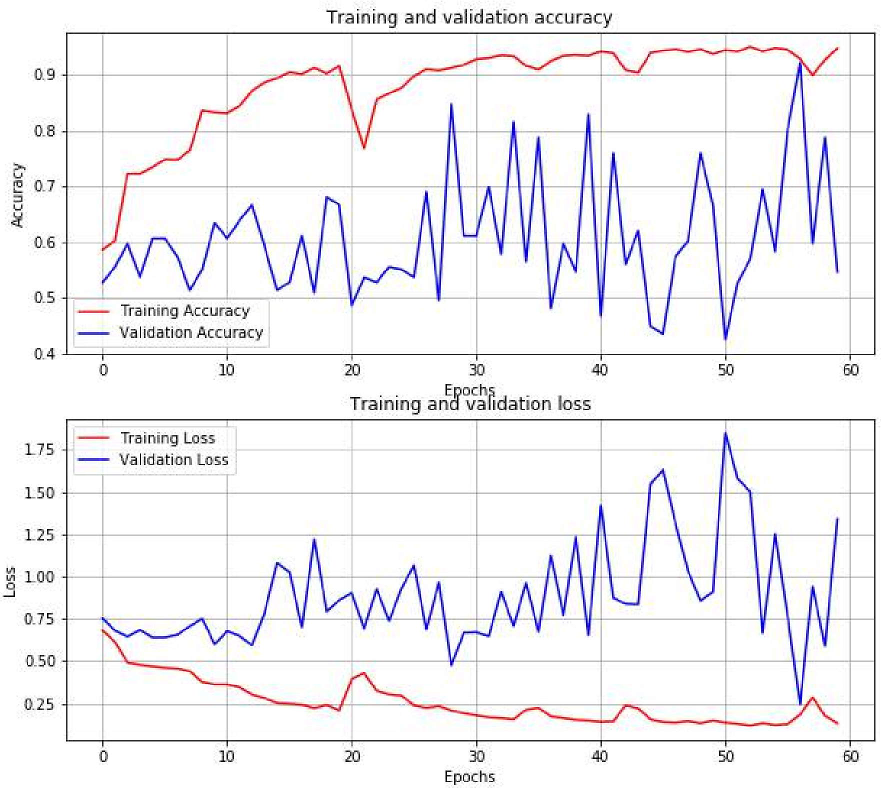
12/12/20 Applied inception network to fluid smectic – hexatic classification (done on google colab using gpu)

Aim: Train an inception model for the classification of fluid smectic – hexatic textures then apply trained model to test videos to identify transition. Using dataset from 29/11/20.

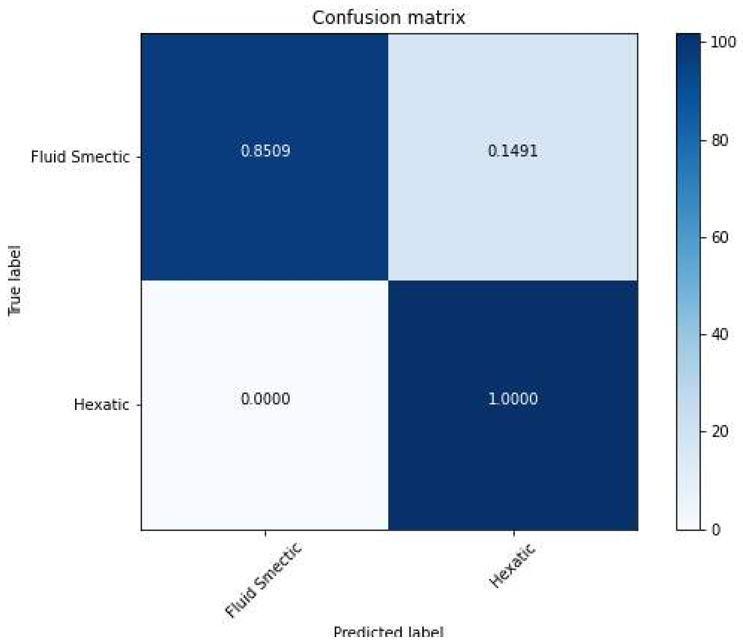
- Based on the above results, decided to use the IneptionLC-V4 model architecture, but with the output changed to a binary output, as this is binary classification.
- All videos, results and code found at "C:\Users\Jason\Documents\University\Year_4\MPhys_project\Project_work\Results\Fluid_Smectic-Hexatic_results"
- Only two videos to test on, again like with SmA and SmC, would be useful to have more videos to both allow for more videos in test set and more training images.
- Model


Input: InputLayer | input: [(None, 256, 256, 1)] | output: [(None, 256, 256, 1)]
Normalize: Rescaling | input: (None, 256, 256, 1) | output: (None, 256, 256, 1)
random_flip: RandomFlip | input: (None, 256, 256, 1) | output: (None, 256, 256, 1)
Conv1: Conv2D | input: (None, 256, 256, 1) | output: (None, 252, 252, 16)
MaxPool1: MaxPooling2D | input: (None, 252, 252, 16) | output: (None, 126, 126, 16)
conv2d_1: Conv2D | input: (None, 126, 126, 16) | output: (None, 126, 126, 96)
conv2d_3: Conv2D | input: (None, 126, 126, 16) | output: (None, 126, 126, 32)
max_pooling2d: MaxPooling2D | input: (None, 126, 126, 16) | output: (None, 126, 126, 16)
conv2d_2: Conv2D | input: (None, 126, 126, 96) | output: (None, 126, 126, 128)
conv2d_4: Conv2D | input: (None, 126, 126, 32) | output: (None, 126, 126, 64)
conv2d_5: Conv2D | input: (None, 126, 126, 16) | output: (None, 126, 126, 64)
conv2d: Conv2D | input: (None, 126, 126, 16) | output: (None, 126, 126, 64)
Inception2: Concatenate | input: [(None, 126, 126, 64), (None, 126, 126, 128), (None, 126, 126, 64), (None, 126, 126, 64)] | output: (None, 126, 126, 320)
conv2d_7: Conv2D | input: (None, 126, 126, 320) | output: (None, 126, 126, 96)
conv2d_9: Conv2D | input: (None, 126, 126, 320) | output: (None, 126, 126, 32)
max_pooling2d_1: MaxPooling2D | input: (None, 126, 126, 320) | output: (None, 126, 126, 320)
conv2d_8: Conv2D | input: (None, 126, 126, 96) | output: (None, 126, 126, 128)
conv2d_10: Conv2D | input: (None, 126, 126, 32) | output: (None, 126, 126, 64)
conv2d_11: Conv2D | input: (None, 126, 126, 320) | output: (None, 126, 126, 64)
conv2d_6: Conv2D | input: (None, 126, 126, 320) | output: (None, 126, 126, 64)
Inception3: Concatenate | input: [(None, 126, 126, 64), (None, 126, 126, 128), (None, 126, 126, 64), (None, 126, 126, 64)] | output: (None, 126, 126, 320)
global_average_pooling2d: GlobalAveragePooling2D | input: (None, 126, 126, 320) | output: (None, 320)
FC4: Dense | input: (None, 320) | output: (None, 128)
dropout: Dropout | input: (None, 128) | output: (None, 128)
dense: Dense | input: (None, 128) | output: (None, 1)

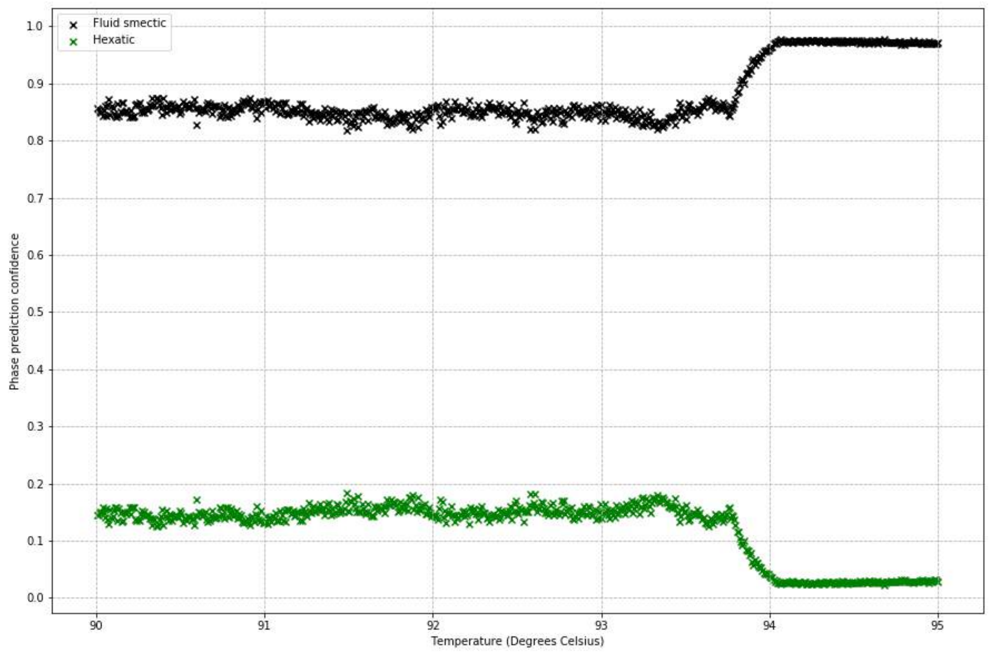
Training graph (achieved 98.2% validation accuracy at epoch 46)



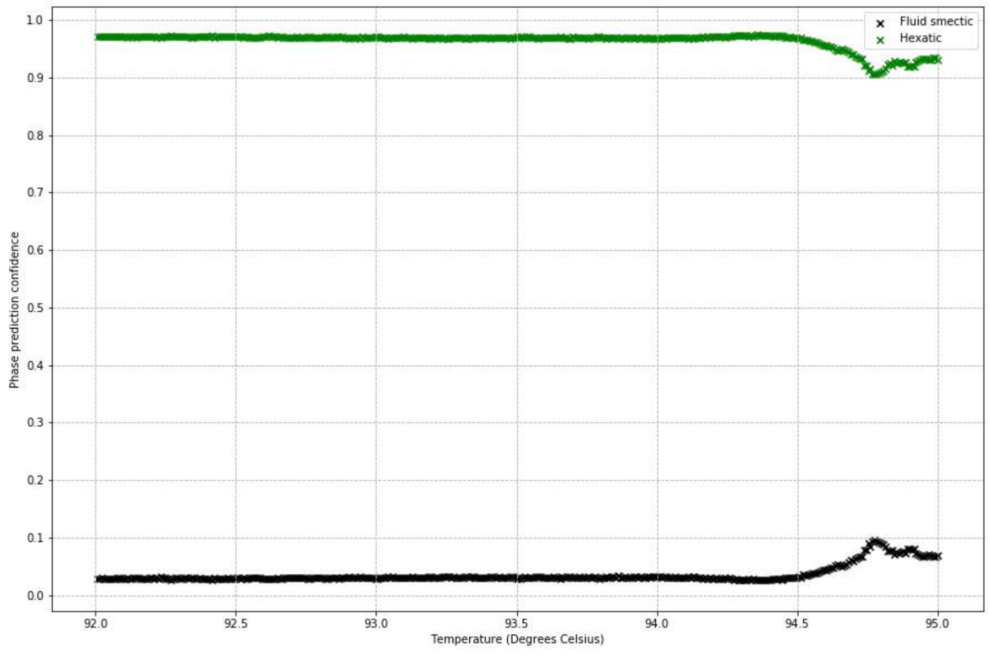
Validation performance (highly accurate, identifying almost all textures correctly)



D7_95-90_cooling



D8_95-92_cooling



- In both videos, no transition was detected. However there was a noticeable change in prediction confidence where the transition occued. This indicates some sign of texture change. But it is very likely that more images will be needed to identify this transition.

Summary: Overall, an inception model like that why performed best for the I,N,N*,Sm task was applied to the task of classifying fluid smectic vs heatic. However, despite a high validation accuracy, the model did not identify the correct phases in the test videos. Some sign of the transition was seen with a 'bump' in the graph, but this problem probably needs more images for training. Also, as high validation accuracy was achieved, maybe a different choice of validation set is needed. Also I could try k-cross fold validation, where a model is trained k times with different choices of training and validation sets, thereby a good model will have achieved a high accuracy on many different validation sets and not just one randomly chosen one. This could lead to increased generalizability.

Week 10 summary:

- Trained a model on all data so far for the I,N,N*,Sm classification to try improve on past trained model. Many model were trained, with increased number of layers in CNN showing increased validation accyracy. But main result was that inception networks performed this classification highly accurately. The most accurate model InecptLC-V4 was applied to test videos.
- InceptLC-V4 model achieved similar results on test videos to past model despite achieving a much higher validation accuracy ~98% compared to ~70%. This suggests that validation set chosen doe not accurate represent the range of textures seen in the test videos.
- Also trained the same inception architecture foe SmA-SmC and fluid smectic to hexatic classification. Neither were accurately labelled on test videos, despite achieving high validation accuracies. Again shows that validation set choice, not sufficient.
- Overall, from this week it seems that an improvement to be made would be to use a different method of validating models. Instead of choosing one small validation set which may or may not represent the wide range of possible textures well, could use k-fold cross validation, where a model is trained several times, using different train -val splits of data. This way, a model is validated on many more images before being used on tst video.

15/12/20 10th meeting (video call)

Attendees: Dr. Dierking, Josh Heaton, James Harbon, Jason Dominguez

Discussed:

- Showed the results of SmA-SmC and Fluid smectcic transition videos labelled this week.
- Dr Dierking mentioned that SmA-SmC was probably mislabelled by me, which I said I would look into.
- Mentioned that we might have a meeting next week if needed.

Week 11 Aims:

- Write MPhys report
- Figure out problem with SMC-SmA classification
- Upload video results from latest inception model and previous CNN model to youtube for someone reading the report to be able to look at.

30/12/20 Found error in SmA, SmC classifier

- The reason for the incorrect performance of the SmA, SmC classifier is that I accidentally trained it on fluid smectic and hexatic images. For bow, as the project is nearing the end for this semester, I will leave this transition as an aim for next semester.

10/01/21 Uploaded videos to Youtube

Uploaded results hown at 16/11/20 for CNN model (inplaylist called first CNN model)

Found at:

https://www.youtube.com/playlist?list=PL9DMoyiEP8lovK_F3HoeruDLEoV3TXISt

and also the results from last week InceptLC-V4 model on tasks I,N,N*,Sm classification and Fluid smectic to hexatic results (in InceptLC-V4 playlist).

Found at: https://www.youtube.com/playlist?list=PL9DMoyiEP8lpA3ty4e8tDkq4hIf8aTda_

Semester 1 project summary

Original aims:

Use machine learning to identify LC phases from their textures

Modified aims:

Use CNN for texture identification on LC transition videos to identify LC transition.

Results obtained:

Overall the transitions of I-N, I-N* and N-Sm were well labelled by the two most accurate CNN models developed in this project (see 09/12/20 & 16/11/20) which are shown in the videos now posted on youtube (see 10/01/21). CNN and Inception CNN architectures were combined using Keras deep learning framework and trained to label textures. These the labelled frames of videos and the resulting prediction confidence against temperature graphs allowed for transitions to be identified.

There was less success with N*-Sm transitions, missing the transitions or identify transition where there were none. This could be down the the small range of textures used so far in this project. Deep learning works best with more data. Also despite models achieving higher validation accuracies, they didn't necessarily perform more accurately on test videos. Finally, when looking at fluid smectic – hexatic transitions, despote high validation accuracy on images, the model trained did not identify transitions in test videos correctly.

Overall the limitations in the project so far seem to be down to two reasons:

- Lack of images
- Validation set not representing test videos well.

To progress with this project, more images could be obtained. This would also allow for model to identify more ordered phases, such as the fluid smectic or other ones which are not currently well represented in the dataset.

Another improvement to be made would involve the validation method. Single hold-out set was used in this project, randomly choosing a small set of images to act as unseen images to see how the model will perform on unseen textures during training. However high validation accuracies didn't always lead to accuractly labelled videos. A method that could be used is k-cross fold validation. Here is a diagram explanation and link to further info.

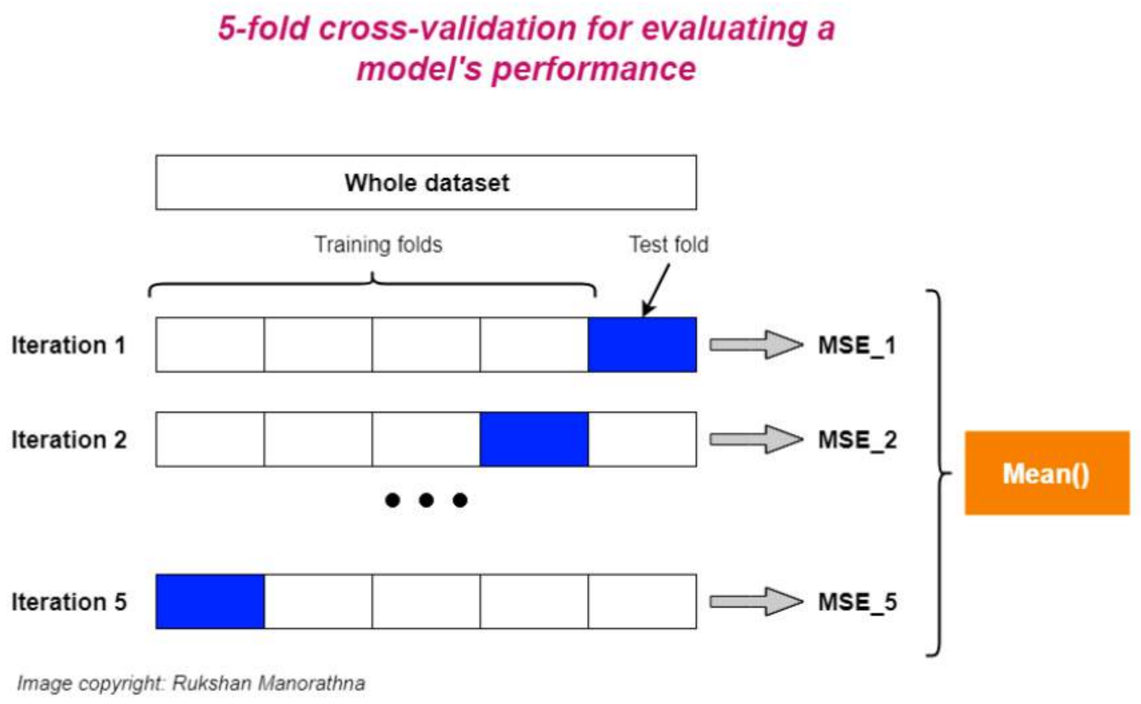

5-fold cross-validation for evaluating a model's performance
Whole dataset
Training folds
Test fold
Iteration 1
Iteration 2
Iteration 5
MSE_1
MSE_2
MSE_5
Mean()
Image copyright: Rukshan Manorathna

Image by Author

[k-fold cross-validation explained in plain English | by Rukshan Pramoditha | Dec, 2020 | Towards Data Science]()

Basically a model is training k separate times, each time time a different choice of validation set is used. You then take the average validation accuracy of all the trained models. This gives a better indication of how the model will generalize, as it had been validated on more images. Then, before testing on images you would train the same model architecture on the entire image set.