A Crystal/Clear Pipeline for Applied Image Processing

Christopher J. Watkins[1](✉), Nicholas Rosa[2], Thomas Carroll[3], David Ratcliffe[4], Marko Ristic[2], Christopher Russell[5], Rongxin Li[6], Vincent Fazio[7], and Janet Newman[2]

[1] Scientific Computing, Commonwealth Scientific and Industrial Research Organisation, Clayton, VIC 3181, Australia
chris.watkins@csiro.au

[2] Manufacturing, Commonwealth Scientific and Industrial Research Organisation, Clayton, VIC 3181, Australia
{nick.rosa,marko.ristic,janet.newman}@csiro.au

[3] The University of Melbourne, Parkville, VIC 3052, Australia
t.carroll4@student.unimelb.edu.au

[4] Data61, Commonwealth Scientific and Industrial Research Organisation, Acton, ACT 2601, Australia
david.ratcliffe@csiro.au

[5] Scientific Computing, Commonwealth Scientific and Industrial Research Organisation, Alexandria, NSW 1435, Australia
christopher.russell@csiro.au

[6] Data61, Commonwealth Scientific and Industrial Research Organisation, Marsfield, NSW 2122, Australia
ron.li@csiro.au

[7] Minerals, Commonwealth Scientific and Industrial Research Organisation, Clayton, VIC 3181, Australia
vincent.fazio@csiro.au

Abstract. Many long-standing image processing problems in applied science domains are finding solutions through the application of deep learning approaches to image processing. Here we present one such application; the case of classifying images of protein crystallisation droplets. The Collaborative Crystallisation Centre in Melbourne, Australia is a medium throughput service facility that produces between five and twenty thousand images per day. This submission outlines a reliable and robust machine learning pipeline that autonomously classifies these images using CSIRO's high-performance computing facilities. Our pipeline achieves improved accuracies over existing implementations and delivers these results in real time. We discuss the specific tools and techniques used to construct the pipeline, as well as the methodologies for testing and validating externally developed classification models.

Keywords: Machine learning · Deep learning · Crystallisation image · Image processing · Automation · Classification

1 Introduction

Knowing the shape of an object reveals much about its function: a single glimpse of a Ferrari and a bus allows one to predict quite accurately which vehicle would go faster. Similarly, given a high-resolution picture of a biological molecule (e.g. a protein molecule) a biologist can tell much about how it works. X-ray crystallography is the only technique that can generate very high-resolution pictures of molecules - pinpointing positions of individual atoms within a large complex molecule [10]. This technique, X-ray crystallography, is the basis for modern drug discovery, synthetic biology and indeed any field - academic or commercial - where understanding biology down to the atomic level is important. To generate an X-ray picture, a *crystal* of the biological molecule is bathed in a powerful beam of X-rays [7]. Production of the crystal samples used in X-ray crystallography is the *limiting* step in 'seeing' biology, and thus understanding it. Figure 1 illustrates the crystallography pipeline in structural biology.

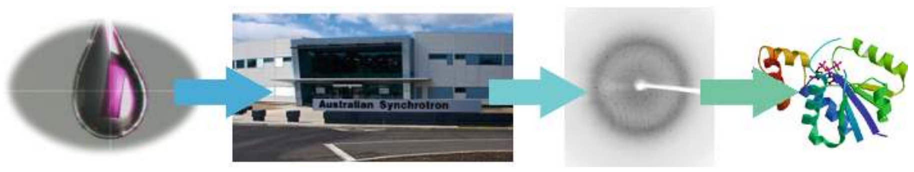


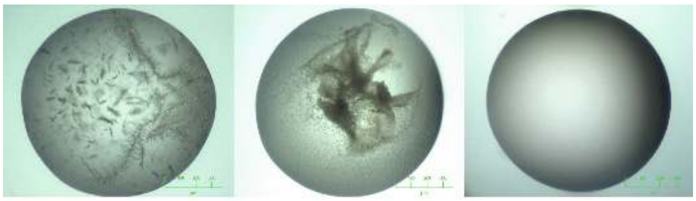
Fig. 1. Structural biology pipeline. Obtaining a crystallised protein is just the beginning. Once a crystal has been grown it is irradiated with X-rays (often at a synchrotron light source). The diffraction images produced at the synchrotron are then used to calculate the atomic structure of the protein.

Crystals of proteins suitable for X-ray analysis are *notoriously difficult* to produce [17]. For each protein, hundreds or sometimes thousands of experiments are set up, where the protein sample is mixed with different cocktails of chemicals in an unsophisticated trial-and-error approach to identify conditions under which the protein will crystallise [30]. Crystal growth is a time-dependent (and intrinsically stochastic) process, so that the trials have to be examined repeatedly over a time-frame of many weeks, with the knowledge that trials that could support crystal growth may not show any crystals, due to the inherent randomness of crystal nucleation. Crystal growth of proteins, or indeed simple molecules like table salt, require that the solution becomes supersaturated and that nucleation occurs. Given supersaturation of a protein solution, the most likely outcome is the formation of a disordered precipitate. Sometimes the crystallisation trial will result in phase separation, and sometimes supersaturation is simply not achieved, and the droplet remains clear. For each of the basic classes of outcome: crystals, phase separation and precipitation there is a huge variation in the type and extent of the outcome. Even in the clear class, where the drop is unchanged, extraneous matter - dust, for example, can give a drop that has features, which are not part of the intended experiment. Further, there are no clear boundaries between the classes - for example, a clear droplet is very hard to distinguish from one with a light precipitate. Often, many outcomes can be seen in the same experiment, see Fig. 2.

The last two decades have seen the application of automation to protein crystallisation experiments - enabling more experiments to be set up, and allowing for the automatic imaging of the experiments. The relative ease of setting up experiments (and the subsequent explosion in the number of experiments created) has made human interpretation of the results unsustainable; even a human annotation of all images containing crystals is becoming increasingly rare, let alone annotation of all the non-crystal outcomes. The high-throughput Collaborative Crystallisation Centre (C3) in Melbourne, Australia has been in operation for over a decade, and has 100 or so active users at any given time. Since its inception in 2006 the C3 has built up a collection of almost 50 million visible light images of the > 3.9 million crystallisation experiments set up; 5–20 thousand new images are collected daily. Less than 5% of the C3 images have been annotated by hand.

Statistics gathered by Structural Genomics initiatives and other studies [32] have suggested that only a small percentage of initial crystallisation experiments produce diffraction quality crystals, and most useful crystals are grown by optimising near misses identified in the initial screening. Although it is widely recognised that there must be useful information that can be garnered from the trials that did not produce crystals, there are no widely available, broadly applicable methods for extracting this information. The paucity of headway in gleaning information from the experiments which fail to yield crystals can be attributed to three factors - incomplete/noisy data about how the trials were set up, incomplete/noisy data about the output of crystallisation trials, and the lack of a clear way of correlating these first two factors, although the value of this type of analysis has been long recognised [13]. The first issue - the problem of defining the crystallisation experiment relies on the development and adoption of standard vocabularies for describing crystallisation experiments, along with punctilious record keeping by the experimentalist [33,35]. The second issue; that of assigning an outcome to each experiment is more tractable, as there are already hundreds of millions of images which capture the output of the crystallisation experiment, due to the widespread adoption of imaging technology in crystallisation labs. What is missing is the reliable and widespread translation of the qualitative image data into quantitative data that could be used in downstream analyses. This process of assigning a value to an image (or to a visual inspection) of a crystallisation experiment is most often called *scoring*, and is the primary focus of this work. The third matter - the lack of tools for correlating the input to the output of an experiment is understandable given the limited amount of data available describing both the input and the output of most crystallisation trials. That is the failure to adequately solve issues one and two.

Analyses of images of crystallisation trials (a solution to the second issue) must fulfil two goals: most importantly, it has to aid in the identification of crystals that might be useful in diffraction experiments, or that might be used as the starting point for optimisation. The longer goal is to have a consistent set of annotations for data mining experiments which would improve the success rate of the current crystallisation process. Crystal formation happens rarely; although no hard numbers are available it is estimated that significantly less than 10% of outcomes are crystalline, which implies low tolerance for false negatives in crystal recognition. To complicate things further, there is no universally recognised set of classes into which images could be sorted, for either machine or human classification. Current scoring systems are generally one-dimensional: "crystal", "precipitate" and "clear". Images which have both crystals and precipitate would be generally annotated as "crystal", as that is the noteworthy outcome. Thus the current human annotations don't necessarily give a good description of what the drop image contains, but are more an indication of the most interesting component of the drop. The blurred boundaries of any crystal image classification system is highlighted by previous work which has shown less than 85% agreement for classification amongst human experts [48].


(a) Crystal (b) Precipitate (c) Clear

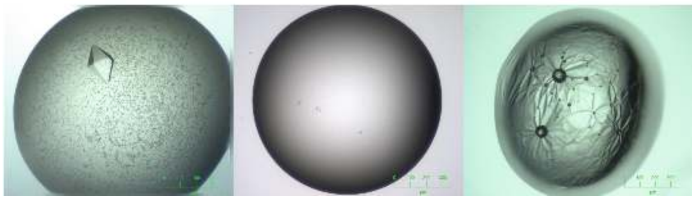

(d) Crystal + Precipitate (e) Plate features (f) skin

Fig. 2. Some example droplet images. Images a-c are clearly single class images (and labelled accordingly), however image d is an illustration of two classes (clear and precipitate) present in a single image. Images e-f are examples of difficult images. e shows a clear droplet where moulding features of the plastic substrate appear like crystals, f shows a droplet with a wrinkled protein skin covering the droplet.

This work presents an automated solution that applies this one-dimensional labelling scheme at scale in a fully distributed High Performance Computing (HPC) environment. We will give an overview of the data challenges, model development and parallelisation strategies used to ensure continuous and robust labelling of new droplet images in near real time.

2 Training and Testing Datasets

With its wealth of crystallisation data and access to world class machine learning engineers, C3 has pioneered the development of an automated classification pipeline, C4 (C3 Classifier), in an attempt to remedy the second problem outlined in Sect. 1, that of incomplete/noisy data about the output of crystallisation trials. To aid in the construction of the pipeline, C3 developed two high quality datasets to use for testing and training purposes, [36]. The images in these datasets were collected using a Rigaku Minstrel crystal imaging system which captured 5 megapixel images with a pixel width representing approximately 5 μm. The current imaging system is a Formulatrix RI1000 (www.formulatrix.com) which produces 5 megapixel images with a pixel width representing between 2 and 10 μm.

The first dataset, “Well Scored”, is a collection of fourteen thousand images scored by a single expert into four classes as listed in Table 1. The second dataset, “One Year”, is a collection of seventeen thousand images collected during the one year period between October 2014 and October 2015. As will be discussed in Sect. 4 the classification model currently in production scores images into the four classes used in the “Well Scored” dataset. Table 1 outlines the mapping from the original “One Year” class labels to the simpler four class system, as well as the mapping for an additional scoring system discussed in Sect. 4.

As will be discussed in Sect. 4 the classification model currently in production scores images into the four classes used in the “Well Scored” dataset. Table 1 outlines the mapping from the original “One Year” class labels to the simpler four class system, as well as the mapping for an additional scoring system discussed in Sect. 4.

In the experiments described in this work the “Well Scored” dataset was used as a completely separate held out test set. That is, we never used the “Well Scored” data for training. We trained all models on the “One Year” dataset and tested on the “Well Scored”. This was primarily due to the empirical perception that the “One Year” dataset was more diverse than the “Well Scored” data, and thus provided a richer training set for the models. As we will discuss in the work that follows we found it particularly important to have a second, completely unseen, held out dataset. The generalisation of the model was of utmost importance and simply testing on a held out fraction of data was not enough to guarantee performance on similar data captured in a different crystallisation facility.

Finally Fig. 3 illustrates the distribution of images across the four classes described above. We can note a fairly similar distribution between the two different datasets, but it is also quite apparent the major over representation of the “crystal” and “precipitate” classes. This is due, in part, to the ambiguity of the definition for both the “clear” and “other” classes.

Table 1. Simplified four class mapping

Well scored	One year	DeepCrystal
Crystal	Crystals high	Alternate spectrum positive
	Crystals mid	Macro
	Crystals low	Micro
	Crystal	
	Crystalline	
	Salt crystals	
Precipitate	Bad precipitate	Precipitate amorphous
	Precipitate	Precipitate non-amorphous
	Good precipitate	
Clear	Clear	Clear
Other	Phase separation	Alternate spectrum negative
	Spherulites	Spherulites
	Clear with stuff	Dry
	Indeterminate	Skin
	Null experiment	Contaminants

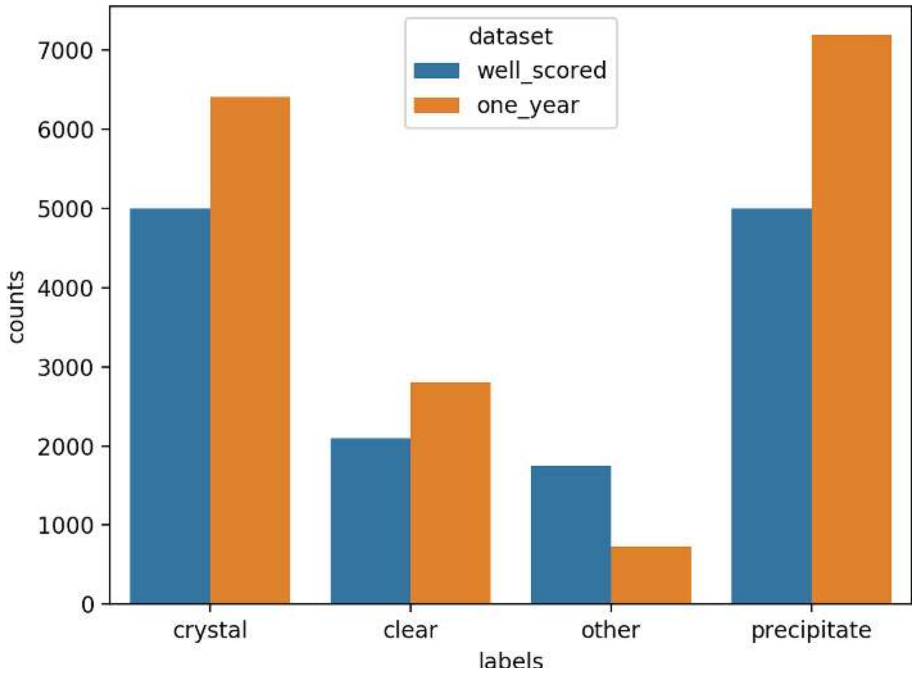


Fig. 3. Distribution of data points across the training (one_year) and validation (well_scored) datasets.

3 Early Attempts

Armed with an abundance of data and a targeted objective (generating accurate annotations for said data) the development of an automated classification pipeline began. This problem is important enough to have driven the development of several machine learning tools already, although none of these tools have been widely adopted, or even used outside of the laboratory in which they were initially developed. We began our work in the area by implementing three existing, externally developed, image classification tools in our own laboratory. See [46] for sample code. This was non-trivial, as none of the three applications we implemented (Besra [11], ALICE [48] and TexRank [34]) had been developed for anything but local use.

Besra was developed as a binary Support Vector Machine classifier [15], using the bag-of-visual-words [16] method to extract a feature vector. The visual vocabulary is computed from the training set by first extracting features (Speeded Up Robust Features [3]) and clustering them into a default 150 clusters using the bag of words k-means [3] clustering function in OpenCV [9]. This training is done on local images. By using Besra alone we obtained a $56.93 \pm 4.63\%$ accuracy when training a binary crystal/not crystal classifier and $63.77 \pm 5.0\%$ accuracy for the binary clear/not clear classifier, these results are summarised in Table 2.

ALICE is a pretrained classifier trained using $1024 \times 1024 \times 8$ bit grayscale bitmap images, corresponding to a pixel width of about 3 μm. It was built using Self-Organising Maps (SOMs) [25] and Learning Vector Quantisation (LVQ) [26] together with Bayesian probabilities [4]. Running ALICE on our test dataset gave an accuracy of $55.68 \pm 1.37\%$ accuracy when trained as a binary crystal/not crystal classifier and $75.90 \pm 2.49\%$ accuracy for the binary clear/not clear classifier, these results are summarised in Table 2. Although this was the poorest performing classifier its results were impressive given the fact that there was no training on local images.

Texrank is another pretrained algorithm, however, this tool was not developed as a classifier. Instead the algorithm was designed to rank a set of images according to their probability of containing a crystal. The ranking is performed by first extracting features by using a pretrained dictionary of textons [23], essentially a numerical descriptor of the textural features in a given image. This feature vector is then passed to a random forest classifier [43], the posterior probability obtained from the classifier is then used to rank droplets. The dataset used to train Texrank contained images with a resolution corresponding to a pixel width of about 4.5 μm. In our binary classification implementations we simply calculated a threshold on the "One Year" independently for both the crystal and clear classifiers. Running Texrank on our test dataset gave an accuracy of $75.03 \pm 0.96\%$ when trained as a binary crystal/not crystal classifier and $74.09 \pm 2.37\%$ accuracy for the binary clear/not clear classifier, these results are summarised in Table 2.

Finally, we amalgamated all three of these methods into a single binary classifier, *Combiner*, using a simple linear combination. We trained a single layer neural network on the "One Year" dataset, labelling the dataset for both crystal and clear classifications. Unsurprisingly the combined approach outperformed the individual techniques, although it never quite reached the level of human accuracy. Running Combiner on our test dataset gave an accuracy of 76.31 ± 2.78% when training a binary crystal/not crystal classifier and 85.12 ± 3.42% accuracy for the binary clear/not clear classifier, these results are summarised in Table 2.

Table 2. Classification accuracies for the binary crystal/not crystal and clear/not clear classifications models when applied to C3 data. Errors are given by the standard deviation over a 10-fold cross validation. The baseline value is generated by predicting each sample to be the majority class. Bold values represent the highest accuracy model for each classifier.

Classifier	Crystal	Clear
Baseline	63.87 %	84.87 %
Besra	56.93 ± 4.63%	63.77 ± 5.0%
ALICE	55.68 ± 1.37%	75.90 ± 2.49%
TexRank	75.03 ± 0.96%	74.09 ± 2.37%
Combiner	76.31 ± 2.78%	85.12 ± 3.42%
CNN	75.80 ± 2.54%	75.80 ± 2.10%
DeepCrystal	76.39 ± 1.00%	80.27 ± 1.01%
MARCO	**91.00 ± 5.00%**	**97.90 ± 5.00%**

During the development of these traditional hand crafted feature extraction approaches it was difficult to ignore the huge advances that were being made with deep learning, particularly in the image processing domains [20,27,39,42]. With this in mind we set out to develop a simple test network to evaluate the efficacy of the approach. Our initial experiments were built using tflearn [18] a Tensorflow [2] powered Python framework. The final network was a simple variant on the original AlexNet [27] architecture: four convolution layers [29] with ReLU activations [31] and max pooling followed by two dense layers with dropout for regularisation [40], the details of which are outlined in Table 3. The immense learning capacity of the network allowed it to outperform any of the traditional computer vision approaches and be on a comparable level to the combination of all methods. Although we were still below the level of human ability, this deep learning approach gave an accuracy of 75.80 ± 2.54% when trained as a binary crystal/not crystal classifier and 75.80 ± 2.10% accuracy for the binary clear/not clear classifier using our test dataset, these results are summarised (as CNN) in Table 2.

4 Deep Learning Solution

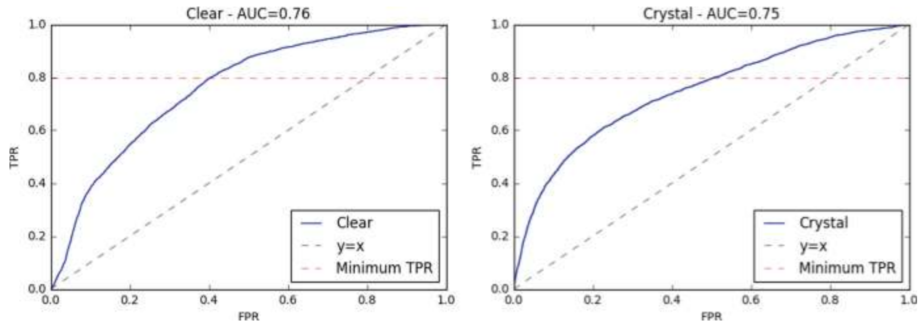
Motivated by the same advances in deep learning that excited us, the VC backed world of Silicon Valley has driven the development of a wealth of applications

Table 3. Preliminary convolutional neural network architecture.

Layer	Description
Input	$128 \times 128 \times 3$ RGB image
Convolution	3×3, 32 outputs, ReLU activation
Max pool	2×2
Convolution	3×3, 64 outputs, ReLU activation
Max pool	2×2
Convolution	3×3, 128 outputs, ReLU activation
Max pool	2×2
Convolution	3×3, 256 outputs, ReLU activation
Max pool	2×2
Dense	512 outputs, tanh activation
Dropout	dropout rate = 0.5
Dense	512 outputs, tanh activation
Dropout	dropout rate = 0.5
Dense	2 outputs

focused machine learning technology. Fortuitously one such application was in the area of machine learning for interpreting crystallisation images. This application was targeted at pharmaceutical and large biotech firms which use structural biology in their lead development pipeline. The recently acquired DeepCrystal (www.deepcrystal.com) had developed a 13-class droplet classification model based on Facebook's high performing convolutional architecture, ResNext [49]. The success of the DeepCrystal model was not so much due to the algorithm itself, which was an implementation of an existing tool, but lay in the diversity of their training data. Through their collaborations with both academic and private institutions (as well as a concentrated web-scraping effort) DeepCrystal was able to build a model with a (self reported) accuracy of 91%. An astounding result. In the spirit of collaboration C3 provided DeepCrystal with a small sample of competently scored images that were representative of those encountered at C3. In exchange DeepCrystal provided access to a closed, black-box implementation of their model, i.e. we were able to pass images in and get classifications out, however we were unable to modify or fine tune the model at all.

The closed nature of the model posed some challenges when trying to validate the claimed accuracy, and complicated comparisons to the other models implemented in C3. The comparison of results from the DeepCrystal model to the others implemented in C3 was stymied by the lack of accepted standards describing outcome classes of crystallisation experiments. We used the mapping shown in Table 1 to map the 13 classes of the DeepCrystal model to a simpler four class output for comparison. Two classes of the DeepCrystal model (the two Alternate Spectrum classes) are inappropriate for the visual light images that we used in our tests. Applying the class mapping we obtained an accuracy of $55.96 \pm 1.84\%$ for the four class scoring task. This poor performance is mostly likely due to the unclear mappings between DeepCrystal classes and C3 classes.



(a) Clear (b) Crystal

Fig. 4. Reciever Operator Characteristic curves for the binary DeepCrystal clear and crystal classifiers.

If we were able to modify the architecture we could simply have adjusted the output layer to match our classification system, and run a few epochs to fine tune the network, unfortunately that option was not possible. Instead the approach we took was to consider the output value for a single class and threshold that value to create a binary classifier. In the case of the crystal/not-crystal binary classifier we thresholded the “Macro” class, such that if “Macro” probability was above a certain value we would flag the presence of a crystal. Similarly for the clear/not-clear classifier we thresholded the “Clear” class. Using the holdout dataset we were able to optimise the threshold values by inspecting the behaviour of their corresponding ROC curves, shown in Figs. 4(a) and (b). The threshold values were found to be 0.4% and 4.3% for the “Clear” and “Macro” classes respectively. These values result in binary classification accuracies of $80.27 \pm 1.01\%$ for clear and $76.39 \pm 1.00\%$ for crystal. However, this system is still better than any of the other single tools, and performed similarly to Combiner, but had the advantage of being significantly simpler and faster than Combiner (in the sense that it was a single model). The ROC curves show far from ideal behaviour resulting in an appreciable false negative rate (FPR) even for low true positive rates (TPRs). In determining the optimum value for the threshold it was decided that an acceptable minimum TPR was 0.8. Users of the crystallisation centre are typically very intolerant of false negatives. Protein crystals are extremely hard to produce, and as such users *do not* want to miss any samples that may possibly contain a crystal. As a result users are more tolerant of false positives, that is, the inclusion of images with no crystals in the set containing crystals is an acceptable compromise in order to not miss any images that may contain crystals.

Around the time that DeepCrystal was being acquired and support for its improvement was lost the MAchine Recognition of Crystal Outcomes (MARCO) initiative was bearing fruit [12]. This collaborative effort between an international collection of academic institutions and pharmaceutical companies amassed a dataset of almost 500,000 images, which have been made publicly available, [14] and have been classified using the same four class system described in Sect. 2. The MARCO model uses the Inception-v3 architecture [41], has been trained using the open source MARCO dataset and has been open sourced itself [45], making it more flexible than our DeepCrystal implementation. Additionally MARCO reports some excellent results[1], producing accuracies of $97.90 \pm 5.00\%$ for clear and $91.00 \pm 5.00\%$ for crystal. Such results are far greater than any previously implemented approaches thus we now deploy the MARCO model as part of the image classifying pipeline discussed in Sect. 5.

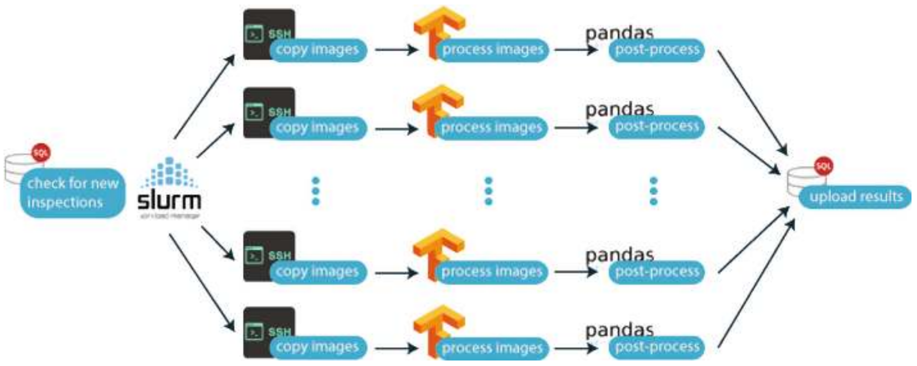


Fig. 5. Illustration of the C4 parallelisation scheme. Inspections of images are processed in parallel which are further classified in parallel on using GPU accelerated Tensorflow models.

5 Enabling Infrastructure

Previous sections have outlined the machine learning component of our solution: the models tested and how they are used. Here we will describe in more detail how the machine learning has been integrated into a fully autonomous end-to-end image classification pipeline, C3 Classifier (C4). This code is available at [47]. The two automated imagers in C3 (Formulatrix RI1000, www.formulatrix.com) produce JPG images of all droplets found in a single experimental plate – this is called an *inspection*; each inspection contains either 24 or 96 (or some multiple) images. Initially the images are stored locally on the imaging machine,

[1] The results reported here are those from [12], and thus have been produced using the MARCO dataset.

but are transferred almost immediately to a larger in-house cloud storage device (Bowen) for long term storage. Simultaneously the metadata for both the inspection and each image collected as part of the inspection is pushed into an Oracle database that is hosted in the same local cloud data centre. There are front-end applications which are available to the C3 users that allow them to inspect and classify images. Both the scores, and other associated information (scorer, score time) are captured in this same database. Scores generated by the machine learning tool are also stored in the database, the "scorer" field in the database is used to mark these as machine generated scores. Thus the database contains a record of all past and present images, noting which images have been classified, either manually by a human or automatically by our autonomous pipeline.

C4 is run from CSIRO's GPU cluster, Bracewell[2], which is composed of 114 Dell PowerEdge C4130 servers each with 4 NVIDIA Tesla P100s and dual Intel Xeon 14-core E5-2690 v4s connected on a FDR10 InfiniBand interconnect (a total of 456 GPUs, 228 Xeons; approximately 2.5 Pflops). This cluster is located in physically located about 600 Km from C3, at the Canberra Data Centre. Overall, the pipeline follows these steps:

- Check database for new, unscored (by C4) images
- Copy unscored images from cloud storage on Bowen to local storage on Bracewell
- Inspect images using the algorithms discussed in Sect. 4
- Save raw output to local cold storage
- Upload scores to the database
- Remove local image copies

C4 is written using Python and is tightly integrated with the Slurm Workload Manager. Some initial experiments were developed using the Luigi [8] workflow package for Python, but integration of Luigi with Slurm didn't fit the traditional High Performance Computing (HPC) model. A possible substitute is SciLuigi [28] which natively supports Slurm. The custom work flow employed by C4 has been modelled on Luigi: several distinct components, with checkpoints at the end of each component. The components of C4 are described in detail in the sections that follow and have been illustrated in Fig. 5.

5.1 Inspection Finder

C4 is set up to be run on an hourly basis, the initial execution is managed by a Cron job. Cron launches a single batch job which runs the Python script for inspecting the database for new inspections (via the Oracle ORM Python package). For each new un(auto)scored inspection a series of dependant batch jobs will be launched. This allows us to process all new inspections in parallel across the cluster. First the inspection classification script is queued. Then the post processing script is queued with the classification job listed as a dependency, so that it will not launch until the classification job has exited. Finally a data

[2] https://www.csiro.au/en/Research/Technology/Scientific-computing/Bracewell.

egress script is launched which has all of the previously submitted batch jobs (classification and post-processing) listed as dependencies. Only a single instance of this script is run and it will collate all the results and update the database with a single post so as to minimise the number of database connections. The post processing script also performs the clean up. All of these dependencies can be programatically determined, with their job IDs, from within Python. There are several Slurm Python wrappers, but we have made use of the PySlurm [37] implementation.

5.2 Inspection Classifier

Each individual classification job copies all the images in an inspection from Bowen cloud storage to node local Bracewell storage via `scp`. Specifically the copy is made using a combination of the Paramiko [19] and `scp` Python packages.

Once the image copy is complete the images are classified in parallel using the MARCO model discussed in Sect. 4. The MARCO model is an Inception-v3 architecture written using Tensorflow. The architecture and weights for the model can be obtained from [45]. As such the model can be called directly from within Python scripts and easily accelerated using the GPU. Images are currently processed in batches of 32. Probability distribution vectors returned from the MARCO model are saved into a temporary file as a means of checkpointing. Once the outputs had been successfully written to file we could remove the original inspection images and safely exit the program which would enable the corresponding post processing script to be executed when sufficient resources were available.

5.3 Post Processing

The post processing script takes the output probability distribution and identifies any interesting drops. Since replacing the DeepCrystal model with MARCO the post processing stage has become much simpler. First it reads in the temp file containing the probabilities for each image. Then it applies an argmax to the probability distribution, returning the most probably class prediction. These results are then saved in a new temp file, awaiting final processing.

5.4 Upload Results and Clean up

Once the all the inspections are processed the final data egress script will collate the results in all of the temp files and upload the interesting drops to the database in a single INSERT call, minimising the number of database connections the need to be made. The temp files are then removed, with the results probability files appended to a larger single file in long term storage.

5.5 Logging

With a fully autonomous pipeline running independently of human oversight, it can be quite tempting to simply assume it is running correctly. Software engineering best practices however, suggest that in these situations it is best to set up a logger, that is code which outputs information about the pipelines state at any given time. Using Python's logging package C4 has been able to build multiple loggers for each component of the pipeline. That is, each component has a regular logger printing to `stdout`, a debug logger printing to file as well as a custom logger that posts to a Slack channel in the project's team space. The Slack logger is the most important as experience has shown us how easy it is to forget to inspect log files. This custom logger will post selected updates to it's own channel, so as not to clutter the regular communication streams. The posts have also been formatted such that errors and progress updates are easily discernible, as well as the point of failure should an error occur. Future implementations of the logger will also send failure alerts via email, should Slack's current raging popularity diminish.

5.6 Cinder and Ashes

In an effort to improve the model on an ongoing basis we have begun collecting images that humans perceive MARCO has scored incorrectly. In a weekly Python script, Ashes, we identify any newly scored images where the human classification disagrees with the MARCO classification. These images are collected and saved for fine tuning in the future. However as discussed in Sect. 1 there is often disagreement among domain specialists as to what score to assign to images. Thus it is not enough to merely take the human score as the ground truth. To remedy this problem C3 has upgraded its image scoring app, Cinder [1] (Crystallographic Tinder), to allow for consensus scoring of badly scored images. That is to say, through the Cinder app users (generally crystallography experts) can score images that have been collected by Ashes. Once we have collected enough scores which are in agreement about a single image we can add that image to the data set for training, satisfied that its ground truth label is sufficiently accurate. This "citizen science" approach to data labelling has only begun recently and sufficient data has not yet been collected to begin the fine tuning process.

6 Deployment and Future Challenges

C4 has been in operation for over a year and its deployment has been well received. This has been most notably observed through a *reduction* in human scoring activity. C4 seamlessly processes up to twenty thousand images per day, casually fitting brief batch jobs into the typical HPC scheduler. A single inspection ($\sim$200 images) can be processed in under two minutes, with the majority of the processing time being consumed by data transport and database communication. While C4 is more than capable of keeping up with the continuously produced images, the ultimate goal of the C4 pipeline is to score the large backlog of over forty million images, enabling an understanding of the crystallisation landscape. While C4 has been designed with this goal in mind at present there are two limiting components: data ingress and data egress.

The server which manages the copying of data from storage to the compute is a low powered cloud machine. As such it cannot handle multiple connections over `ssh`. This has resulted in C4 limiting the number of simultaneous connections to the data server to ten, which ultimately stems the flow of data to the compute servers. We are currently investigating the root cause of the data flow problem, but it will most likely be solved by smarter scheduling around data handling. Similarly on the data egress front, the database does not accept a high number of simultaneous connections. This issue is currently handled by collating batches into a single upload, but this solution will not be acceptable nor stable enough to manage processing the backlog of data. One possible solution (for both ingress and egress) is to have some sort of data buffer that is periodically filled and emptied as required.

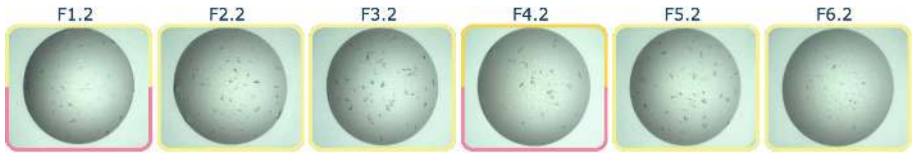


Fig. 6. An example visualisation from the See3 web inspection application available to C3 users. Typically users are shown an interface displaying thumbnails of their processed experiments. All scores are shown as border colours, The border starts at 9 o'clock and the colours are arranged so that user scores are shown first. An image with two scores will have two colours, with the upper colour being the human score, and the lower colour the machine score, an image with a single colour border has only been assigned a single score. (Color figure online)

With the droplet classifications stored in a database it is easy to integrate the findings into the existing C3 web application, See3. Figure 6 illustrates how the C4 classifications have been integrated into the See3 application. The yellow and orange colours on the upper border indicate that the user has assigned these images a crystal class (C3 uses multiple levels within the 4 broad categories in its classification system). The two samples F1.2 and F4.2 that have a pink lower border are images that have been flagged by C4 as containing crystals. The cases that have a single colour are images that have been scored only by the user, that is C4 has missed these images. You can see in this random sampling that C4 has missed quite a few instances of crystallisation.

The multiple shades for each class are an attempt at circumventing the class labelling issue discussed in Sect. 4. By refining the crystal (or other) class into more nuanced subclasses we are able to better capture the continuous nature of the crystallisation spectrum. In practice, there is not a discrete phase change from say, precipitate to crystal; there is a continuum between droplets containing only precipitate, droplets containing both precipitate and crystals and droplets containing only crystals; as such it can be difficult to define class boundaries. This partly explains the divergence of classification labels among domain specialists, with each expert identifying different features that are interesting to them. One potential approach to solving this problem may be to investigate the use of unsupervised learning methods. Given that C3 has a wealth of unlabelled diverse data we could use this to train an unsupervised feature extractor, something like a convolutional autoencoder [5]. The features learned by the autoencoder could then be used by a clustering algorithm, perhaps t-Stochastic Neighbourhood Embedding [44], to find local groups of visually similar images. If obviously distinct classes cannot be differentiated perhaps at least some intuition as to how the droplet classes can be arranged together. Additionally, the labelled and unlabelled datasets could be combined in a semi-supervised scheme similar to [6,21,22] or [24].

One of the fundamental lessons we have learned is that the diversity of the data set is of critical importance. We have experienced a number of times that models trained on datasets that do not accurately represent the data to be classified post training fail to generalise in their application. Whether it be over sampled classes, redundant (or repeated) images or general lack of diversity in samples and sources, the dataset quality is key to the model performance. This is one of the most pressing issues in automating the online training process; how to ensure quality in the automatically extracted training set. While there is some work that suggests deep neural networks are robust to noise in the training data [38], when we are trying to fine tune a network it will be quality, well classified images from the boundaries of the class domains that will ensure a robust and reliable model moving forward.

7 Conclusions

C3 has developed and deployed a reliable and robust autonomous classification system for protein crystallisation images. The system has been deployed in a production system delivering almost real time results through the massive parallelisation of image processing. The pipeline has received qualitatively good feedback on its performance, although it is clear that further development is required. With the ability to compare human and machine scores C3 is now seeking to develop an online training technique by mining hard-to-classify images.

References

1. Cinder "crystallographic tinder". https://research.csiro.au/crystal/user-guide/c3-cinder/. Accessed 02 Jan 2019
2. Abadi, M., et al.: TensorFlow: large-scale machine learning on heterogeneous distributed systems March 2016. http://arxiv.org/abs/1603.04467

3. Bay, H., Tuytelaars, T., Van Gool, L.: SURF: speeded up robust features. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3951, pp. 404–417. Springer, Heidelberg (2006). https://doi.org/10.1007/11744023_32
4. Bayes, F.R.S.: An Essay towards Solving a Problem in the Doctrine of Chances. Philos. Trans. R. Soc. Lond. **53**(0), 370–418 (1763). https://doi.org/10.1098/rstl.1763.0053
5. Bengio, Y.: Learning deep architectures for AI. Found. Trends® Mach. Learn. **2**(1), 1–127 (2009). https://doi.org/10.1561/2200000006. www.nowpublishers.com/article/Details/MAL-006
6. Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H.: Greedy layer-wise training of deep networks. In: Schölkopf, B., Platt, J.C., Hoffman, T. (eds.) Advances in Neural Information Processing Systems 19, pp. 153–160. MIT Press (2007). http://papers.nips.cc/paper/3048-greedy-layer-wise-training-of-deep-networks.pdf
7. Rupp, B.: Garland Science - Book: Biomolecular Crystallography + 1. Garland Science, 1st edn. (2009). http://www.garlandscience.com/product/isbn/9780815340812
8. Bernhardsson, E., Freider, E., Rouhani, A.: Luigi (2012). https://github.com/spotify/luigi
9. Bradski, G.: The OpenCV library. Dr. Dobb's J. Soft. Tools (2000)
10. Brändén, C.I., Tooze, J.: Introduction to Protein Structure. Garland Pub, Spokane (1999)
11. Bruno, A.E.: Besra (2015). https://doi.org/10.5281/zenodo.60970, https://www.researchgate.net/publication/309319298_Besra
12. Bruno, A.E., et al.: Classification of crystallization outcomes using deep convolutional neural networks. PLoS One **13**(6) (2018). https://doi.org/10.1371/journal.pone.0198883
13. Carter, C.W., Carter, C.W.: Protein crystallization using incomplete factorial experiments. J. Biol. Chem. **254**(23), 12219–12223 (1979). www.jbc.org/cgi/content/short/254/23/12219
14. Charbonneau, P.: Machine recognition of crystal outcomes (2018)
15. Cortes, C., Vapnik, V.: Support-vector networks. Mach. Learn. **20**(3), 273–297 (1995). https://doi.org/10.1007/BF00994018
16. Csurka, G., Csurka, G., Dance, C.R., Fan, L., Willamowski, J., Bray, C.: Visual categorization with bags of keypoints. In: Workshop on Statistical Learning in Computer Vision, ECCV, pp. 1–22 (2004). http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.72.604
17. Cudney, R., Patel, S., Weisgraber, K., Newhouse, Y., McPherson, A.: Screening and optimization strategies for macromolecular crystal growth. Acta Crystallogr. Sect. D Biol. Crystallogr. **50**(4), 414–423 (1994). https://doi.org/10.1107/S0907444994002660. http://www.ncbi.nlm.nih.gov/pubmed/15299395
18. Damien, A., et al.: TFLearn (2016)
19. Forcier, J.: Paramiko (2017)
20. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition (2015). http://arxiv.org/abs/1512.03385
21. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science **313**(5786), 504–507 (2006). https://doi.org/10.1126/science.1127647. www.science.sciencemag.org/content/313/5786/504
22. Jarrett, K., Kavukcuoglu, K., Ranzato, M., LeCun, Y.: What is the best multi-stage architecture for object recognition? In: 2009 IEEE 12th International Conference on Computer Vision, pp. 2146–2153, September 2009. https://doi.org/10.1109/ICCV.2009.5459469

23. Julesz, B.: Textons, the elements of texture perception and their interactions. Nature **290**(5802), 91–97 (1981). https://doi.org/10.1038/290091a0. www.nature.com/doifinder/10.1038/290091a0
24. Kavukcuoglu, K., Sermanet, P., lan Boureau, Y., Gregor, K., Mathieu, M., Cun, Y.L.: Learning convolutional feature hierarchies for visual recognition. In: Lafferty, J.D., Williams, C.K.I., Shawe-Taylor, J., Zemel, R.S., Culotta, A. (eds.) Advances in Neural Information Processing Systems, vol. 23, pp. 1090–1098. Curran Associates, Inc. (2010). http://papers.nips.cc/paper/4133-learning-convolutional-feature-hierarchies-for-visual-recognition.pdf
25. Kohonen, T.: Self-organized formation of topologically correct feature maps. Biol. Cybern. **43**(1), 59–69 (1982). https://doi.org/10.1007/BF00337288
26. Kohonen, T.: Learning Vector Quantization. Springer, Heidelberg (2001). https://doi.org/10.1007/978-3-642-56927-2_6
27. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, vol. 25, pp. 1097–1105 (2012)
28. Lampa, S., Alvarsson, J., Spjuth, O.: Towards agile large-scale predictive modelling in drug discovery with flow-based programming design principles. J. Cheminformatics **8**(1), 67 (2016). https://doi.org/10.1186/s13321-016-0179-6
29. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998). https://doi.org/10.1109/5.726791
30. Luft, J.R., Newman, J., Snell, E.H.: Crystallization screening the influence of history on current practice. Acta Crystallogr. Sect. F Struct. Biol. Commun **70**(7), 835–53 (2014). https://doi.org/10.1107/S2053230X1401262X. www.ncbi.nlm.nih.gov/pubmed/25005076
31. Nair, V., Hinton, G.E.: Rectified linear units improve restricted Boltzmann machines. In: Proceedings of the 27th International Conference on Machine Learning, vol. 3, pp. 807–814, Haifa, Israel (2010). https://doi.org/10.1.1.165.6419. http://www.cs.toronto.edu/fritz/absps/reluICML.pdf
32. Newman, J., et al.: On the need for an international effort to capture, share and use crystallization screening data. Acta Crystallogr. Sect. F Struct. Biol. Crystallization Commun. **68**(3), 253–258 (2012). https://doi.org/10.1107/S1744309112002618. www.ncbi.nlm.nih.gov/pubmed/22442216
33. Newman, J., Peat, T.S., Savage, G.P.: What's in a name? Moving towards a limited vocabulary for macromolecular crystallisation. Aust. J. Chem. **67**(12), 1813 (2014). https://doi.org/10.1071/CH14199. www.publish.csiro.au/?paper=CH14199
34. Ng, J.T., Dekker, C., Kroemer, M., Osborne, M., von Delft, F.: Using textons to rank crystallization droplets by the likely presence of crystals. Acta crystallogr. Sect. D, Biol. crystallogr. **70**(10), 2702–2718 (2014). https://doi.org/10.1107/S1399004714017581. www.ncbi.nlm.nih.gov/pubmed/25286854
35. Ng, J.T., Dekker, C., Reardon, P., von Delft, F.: Lessons from ten years of crystallization experiments at the SGC. Acta Crystallogr. Sect. D Struct. Biol. **72**(2), 224–35 (2016). https://doi.org/10.1107/S2059798315024687. www.ncbi.nlm.nih.gov/pubmed/26894670
36. Ratcliffe, D., Carroll, T., Watkins, C., Newman, J.: CSIRO data access portal - crystallisation images from C3 (2016). https://data.csiro.au/dap/landingpage?pid=csiro:20158&v=3&d=true
37. Roberts, M., Torres, G.: PySlurm (2017). https://pyslurm.github.io/
38. Rolnick, D., Veit, A., Belongie, S., Shavit, N.: Deep learning is robust to massive label noise, May 2017. http://arxiv.org/abs/1705.10694

39. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition, September 2014. http://arxiv.org/abs/1409.1556
40. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**, 1929–1958 (2014). https://doi.org/10.1214/12-AOS1000
41. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2818–2826, June 2016. https://doi.org/10.1109/CVPR.2016.308
42. Szegedy, C., et al.: Going deeper with convolutions, September 2014. http://arxiv.org/abs/1409.4842
43. Ho, T.K.: The random subspace method for constructing decision forests. IEEE Trans. Pattern Anal. Mach. Intell. **20**(8), 832–844 (1998). https://doi.org/10.1109/34.709601
44. Van Der Maaten, L., Hinton, G.: Visualizing Data using t-SNE. J. Mach. Learn. Res. **9**, 2579–2605 (2008). www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf
45. Vanhoucke, V.: Automating the evaluation of crystallization experiments. https://github.com/tensorflow/models/tree/master/research/marco (2018)
46. Watkins, C.J.: C3 Computer vision algorithms (2017). https://data.csiro.au/dap/landingpage?pid=csiro:29414
47. Watkins, C.J.: C4–C3 Classification pipeline (2018). https://data.csiro.au/dap/landingpage?pid=csiro:29413
48. Watts, D., Cowtan, K., Wilson, J.: IUCr: automated classification of crystallization experiments using wavelets and statistical texture characterization techniques. J. Appl. Crystallogr. **41**(1), 8–17 (2008). https://doi.org/10.1107/S0021889807049308
49. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks, November 2016. http://arxiv.org/abs/1611.05431