final report

Project code: B.AHE.0320

Prepared by: Karen Clark and Spencer Nesbitt
Solentive Systems Pty Ltd

Date published: 7 November 2018

PUBLISHED BY
Meat and Livestock Australia Limited
Locked Bag 1961
NORTH SYDNEY NSW 2059

On-farm faecal worm egg count proof of principle

Meat & Livestock Australia acknowledges the matching funds provided by the Australian Government to support the research and development detailed in this publication.

Executive summary

The current process for obtaining faecal worm egg counts can be a time-consuming exercise; both in terms of the manual counting of eggs, and also the time it takes to send a faecal sample to a laboratory and receive the results. The on-farm faecal worm egg count proof of principle project aims to use machine learning to create an artificial intelligence (AI) model to replace the manual process of identifying and counting parasite eggs in samples. In so doing, it will be possible for the process to be completed on site, without involving a lab. As the ultimate users for the application reside in areas with unreliable mobile phone and internet connections, the project sought to confirm the feasibility of deploying the AI model to a mobile device with only occasional internet connectivity, where the phone's camera would be used to capture images for evaluation.

The project proved that an AI model could be trained to recognise and count strongylid eggs from a set of computer-generated images. The model was then tested on an Android and iOS device; where a generated image on the device's image library could be run against the model to count the number of eggs in the image. It was also proven that the model could be applied to a picture taken of a generated image using the phone's camera. Finally, based on a comparison of detection and classification models, it was determined that a detection model was more precise at identifying different egg types and counts.

In order to train the AI model, tens of thousands of images were required. Individual egg types and plant debris were isolated from a sample image and fed into image-generating software that varied the placement and number of eggs on an image. The images were provided to an open-source machine learning platform, called TensorFlow; where transfer learning was used to 're-train' a convolutional (image recognition) model. This model was then tested against generated images to determine how successful it was at predicting the presence of eggs.

A basic mobile app was created to use the AI model. The model was then deployed to an Android device using the TensorFlow Lite system. A user was able to select a preloaded image and run it against the AI model to obtain an egg count.

The project continued by extending the application to run on an iOS device; where the model was run against pictures taken using the phone's camera. Computer generated images were printed out for this purpose.

Finally, a detection AI model was created to compare the precision and accuracy of a detection model to a classification model. By running the same set of images against both models, it could be determined which model was more effective at identifying and counting different egg types.

In conclusion, AI and machine learning are viable technologies for counting worm eggs in faecal samples. Using specialised systems, AI models can be deployed and run on mobile devices. However, further testing of the models against real images is required and consideration should be given to simplifying the problem by applying physical optimisations. For example, contriving a situation in which eggs cannot overlap and only appear in relatively small numbers.

Regardless of the presence of such a model, there remains a challenge around generating the images themselves. In order to get an image, producers will need to obtain and prepare samples for evaluation. Images must also be captured at the right distance for an accurate assessment.

Table of contents

1 Background

This project is a proof of principle that aims to see if it is possible to use mobile devices to perform on-farm faecal worm egg counts. Faecal worm egg counts require samples to be collected and stored in a specific way before they are sent to a lab. The most common technique is the modified McMaster method, where samples are mixed with a flotation solution, that is piped onto a slide that allows the eggs to be viewed under a microscope. Using a specific magnification, the technician then determines the presence and density of eggs by manually counting them. The end-to-end process can take up to 10 days to get a result.

Meat and Livestock Australia sought to test the feasibility of using machine learning to perform the egg count. Producers could then collect samples, create slides and take images using a mobile phone; which could be run against the AI model to identify and count the eggs. Since internet connectivity is often unreliable in rural and regional areas, the application would also need to work solely on the mobile device.

While different types of machine learning exist, the general problem of categorising and recognising images is relatively well known. A particularly successful approach to this problem is using convolutional neural networks to identify patterns (edges, etc.) in images and then combining these into complex structures. Pre-trained models exist that perform the initial steps of the recognition process but allow for the final stage to be re-trained to meet a specific requirement. This process is referred to as transfer training and was selected as the approach for creating the AI model; where existing mobile-friendly, image recognition model was re-trained to identify and count parasite eggs.

The training of a model can be done in different ways. For this project, supervised learning was used by providing the machine with a set of categorised images where the outcome is known (i.e. the number of strongylid eggs in each image was known).

Within image recognition techniques, there exist various approaches for building an AI model. The two main approaches for building an AI model suitable for this application are:

1. **Categorisation model:** Categorisation of images as one thing or another. In this case the system would learn to differentiate between, say, an image with 3 eggs and an image with 20 eggs. The system does not actually count the eggs but rather learns what an image with a certain number of eggs 'looks' like.
2. **Detection model:** Here the system attempts to specifically recognise a strongylid egg and count the number of occurrences it finds in an image.

The categorisation approach has the advantage of being less computationally intensive and being easier to implement than a detection model. Is was suspected that these characteristics would be desirable for implementations targeted at mobile phones. The disadvantage of this approach, is that it is frequently less accurate than a detection and counting technique; especially when confronted with images containing egg counts that it has not previously seen.

The detection approach is based on a two-step process, firstly identification of a particular object, in this case a strongylid egg and secondly a scan of the image to count the occurrences of the object. The project evaluated both approaches to ascertain the effectiveness of each.

2 Project objectives

The primary aim of this project is to demonstrate that a mobile application can be used to quantify internal parasite eggs in faecal samples. This includes three components:

1. Creating a trained AI model to recognise and count worm eggs
2. Deploying and evaluating the running of an AI model on an Android mobile device
3. Evaluating cross platform capabilities to include iOS devices

2.1 Creating a trained AI model to recognise and count worm eggs

The project aims to 'train' an Artificial Intelligence (AI) model to recognise the eggs of strongylid worms in an image containing eggs of other worm types, grass particles and other debris. Google have made available, as an open source project, their TensorFlow system to help developers build such models.

2.2 Deploying and evaluating the running of an AI model on an Android mobile device

An important part of the project is validating whether the AI model can be deployed to a mobile device. Google also has a TensorFlow Lite system that allow models to run on mobile devices.

2.3 Evaluating cross platform capabilities to include iOS devices

Ideally, the resulting AI model will be capable of working on both Android and iOS phones. The project also validated whether the TensorFlow Lite system can support iOS phones, and determined if any additional setup of programming was required. The same level of accuracy is required for both Android and iOS phones.

3 Methodology

To complete the objectives, the project was divided into two blocks of work.

For block 1, the following methodology was used:

1. Generating image data for machine learning.
2. Creating an AI classification model for pseudo egg counting.
3. Testing the model.
4. Creating a basic app for an Android device to pass images to the model and display the results.
5. Deploying the model to an Android device and processing pre-loaded images.

For block 2, the following methodology was used:

1. Deploying the mobile app and AI model to an iOS device.
2. Using a phone app to take photos of pre-printed images.
3. Processing those images through the model and displaying the results.
4. Creating a detection and counting based model.
5. Evaluating the difference between classification and detection models on test images.

In both the classification and detection approaches, two sets of images were generated. One set to use in training the models and another set in testing the models. Having two separate, independent sets of images allows for a more reliable assessment for how the model is likely to perform against images it has not previously 'seen'.

The methodology for each of these blocks is addressed in the below sections.

3.1 Block 1

3.1.1 Generating data for machine learning

In order to generate a model to identify strongylid eggs, it was necessary to create tens of thousands of images with various egg counts. An image generating program was developed to do this. See Appendix 1 for details of this process but, in summary, a sample image was taken from a description of the McMaster process and decomposed into a set of components from which new images could be created. Figure 3.1 below shows these base components.

Background Strongylid Coccidia Plant debris

Figure 3.1 Image components.

Figure 3.2 below shows a sample of the images generated using the above components.

Figure 3.2 Image components

3.1.2 Creating an AI classification model for egg identification

A 'transfer learning' approach was used to take an existing mobile-optimised, image recognition model, and modify it to look for parasite eggs. The generated image data was processed through the TensorFlow based system to train the AI model. The resulting model was then tested against a set of independent images to ascertain a score of its accuracy. The training process involved resizing the images to 224 * 224 pixels to align with the chosen model inputs (mobilenet_v1_mobilenet_0.50_224).

3.1.3 Testing the model

In order to test the model, a new set of generated images was applied against the model to determine if it could accurately count the number of eggs. The testing resulted in a range of counts for each image and the model's confidence score of the count's accuracy (e.g. for each count there is a probability that reflects the model's confidence in that count) and the top 5 results for any given image was used for display purposes. An assumption was made that, in the field, slides would be arranged to contain an egg count between 0 to 25 eggs.

In order to evaluate the effectiveness of the model, several measures were used:

- **True Positives (TP):** the number of times the model correctly predicts the number of eggs in the image.
- **False Positives (FP):** the number of times the model predicts a higher number of eggs than were actually present.
- **False Negatives (FN):** the number of times the model predicts a lower number of eggs than were actually present.
- **True Negatives (FN):** the number of times the model correctly predicts when no eggs are in the image
- **Precision:** the percentage where the model found eggs that were actually eggs.

  Precision = (TP / (TP + FP)
- **Recall:** the percentage where the model found all the eggs in an image.

  Recall = TP / (TP + FN)
- **Specificity:** the percentage of images that were correctly identified as having no eggs.

  Specificity = TN / (TN + FP)
- **F1 Score:** a weighted average of precision and recall

  F1 Score = 2 x ((Precision x Recall) / (Precision + Recall))
- **Accuracy:** the percentage of predictions that were correct.

  Accuracy = (TP + TN) / Total Tests

A confusion matrix was also created to demonstrate the distribution of errors. This matrix shows the number of times the model counted eggs correctly and incorrectly (see Figure 3.6).

3.1.4 Creating a mobile app and deploying the model to an Android device

Having created the model, it was deployed to an Android phone where it could be run against an image on the device. The execution of the model on the device was made possible by the use of TensorFlow Lite. The application on the device was used to apply the model to a selected image and

output a confidence level indicating how likely the image was to contain either 0, 1, 2, etc. up to 25 eggs.

The key features of the app included the ability to:

- choose an image from the device's library,
- run the model against the image to identify the number of strongylid eggs and
- show the most likely egg counts for the image and the related confidence of each count.

Figure. 3.3 shows a screen shot of the app.

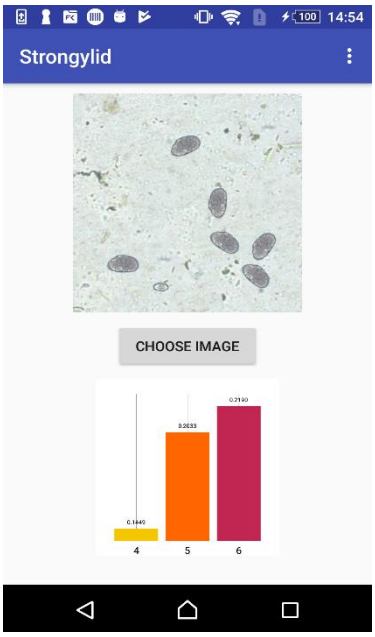

Figure. 3.3 Egg count app on an Android device

The image shown in Figure 3.3 shows that the system has assigned significant probabilities to the image containing 4, 5 or 6 eggs. It may well have assigned non- zero probabilities to all the other values between 0 and 25 but these probabilities were very low. Here it can be seen that the application has assigned a probability of 21% to the image containing 6 eggs, a probability of 20% to the image containing 5 eggs and a probability of 14% to the image containing 4 eggs. It may be inferred from this result that it is not necessarily the absolute percentage probability that is important but the relative probabilities. On average, all values between 0 and 25 other than 4, 5 and 6 will have been assigned a probability of at most 2% (45% divided by 23 other possibilities).

3.2 Block 2

3.2.1 Deploy the app and AI model to an iOS device

In order to test cross-platform capabilities of the TensorFlow Lite system, a native iOS app was developed and, along with the same base model, was installed on an iOS device. Here, in addition to pre-loading images to the device, a function was added to allow an image from the phone's camera

to be passed to the model. To facilitate like for like testing, a selection of images used on the Android device were printed so that they could be photographed by the phone.

The key features of the app included the ability to:

- choose an image from the device's library,
- process an image directly from the phone's camera,
- run the model against the image to identify the number of strongylid eggs,
- show the most likely egg counts for the image and the related confidence of each count.

Figure. 3.4 shows a screen shot of the app.

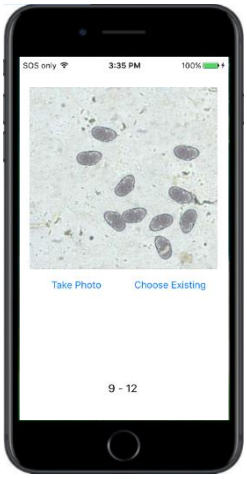

Figure 3.4 Egg count app on an iOS device

A small change from the Android version of the application was showing a range of possible values for the count without the specific probabilities. This was to reflect the importance of the relative values of the probabilities rather than the absolute values. In Figure 3.4 above, it can be seen that the system 'thinks' the image contains between 9 and 12 eggs, with the actual value being 11.

3.2.2 Using a phone app to take strongylid images

The mobile app was updated to allow the model to run against images taken on the phone. Photos were taken of generated images using the phone's camera, and run against the model to evaluate the egg counts.

3.2.3 Evaluating the difference between classification and detection models on more complex images

Having tested a classification model with some success, a natural progression was to build and evaluate a detection and counting model. In this scenario, a model is trained to recognise an individual egg and then detect the number of occurrences within an image. With the ability to recognise an individual object, it was possible to train the model to recognise not just strongylid eggs but also Nematodirus eggs, this allowed the evaluation of how well the system could differentiate between two similar egg types.

Similarly, to the categorisation scenario, it was necessary to generate many thousands of images to train the model. A small but significant difference between the two scenarios, however, is the need to supplement each of the generated images with a XML file describing the location of each of the eggs. To allow this, the image generation software was updated to generate the required XML file for each image.

Figure 3.5 shows an example image with the corresponding XML shown in figure 3.6.

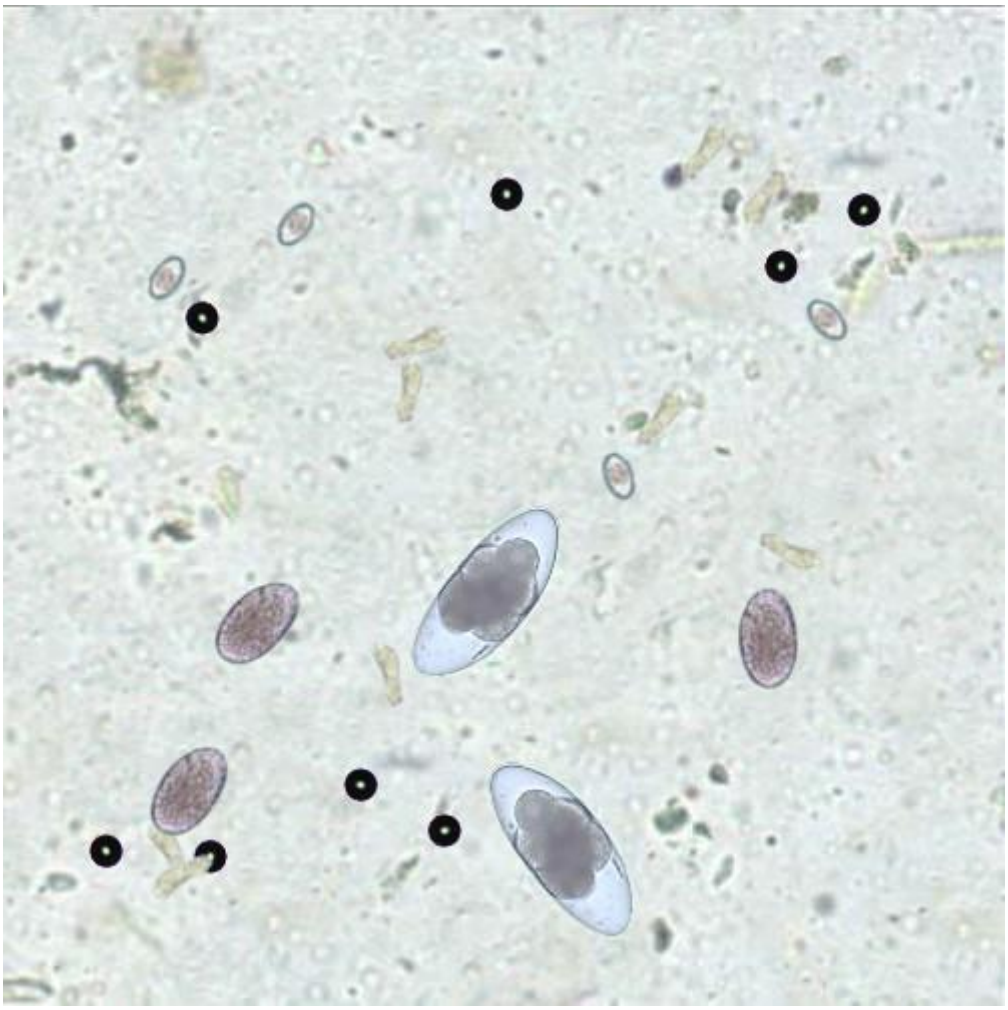

Figure 3.5 Sample image with two strongylid eggs used for training the detection model.

```
<annotation>
  <size>
    <width>500</width>
    <height>500</height>
    <depth>3</depth>
  </size>
  <segmented>0</segmented>
  <object>
    <name>Nematodirus</name>
    <pose>Unspecified</pose>
    <bndbox>
      <xmin>183</xmin>
      <ymin>243</ymin>
      <xmax>300</xmax>
      <ymax>346</ymax>
    </bndbox>
  </object>
  <object>
    <name>Nematodirus</name>
    <pose>Unspecified</pose>
    <bndbox>
      <xmin>235</xmin>
      <ymin>362</ymin>
      <xmax>324</xmax>
      <ymax>480</ymax>
    </bndbox>
  </object>
</annotation>
```

Figure 3.6 XML describing the egg locations for the image in Figure 3.5.

4 Results

4.1 Block 1

4.1.1 Effectiveness the AI classification model to identify and count strongylid eggs

For each image tested, the model output a confidence level for each of the possible classifications, 1 through 25. Figure. 4.1 shows and example with all 'significant' probabilities plotted (i.e. those above about 5%).

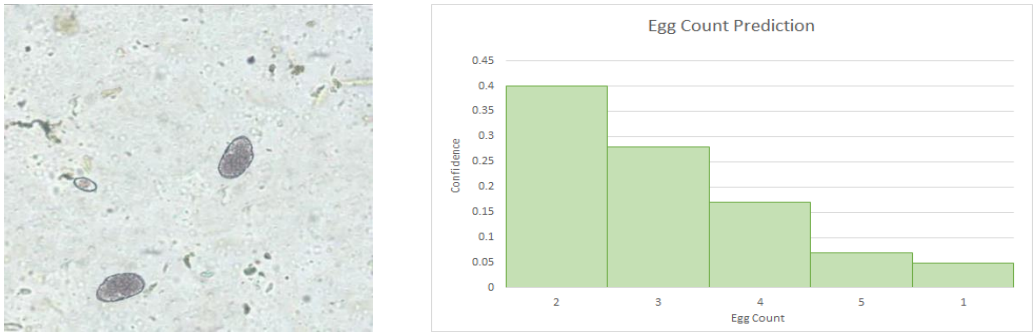

Figure 4.1 Sample test image and results

When evaluating a model's performance, there are a number of numerical measures that can give insight. Perhaps the most obvious one is accuracy, i.e. the number of times the predicted count matches the actual count. This, in general however, is not necessarily the best measure on which to compare models. For example, according to https://bowel-cancer.canceraustralia.gov.au/statistics the estimated percentage of all deaths from cancer, in Australia, in 2018 was 8.5%. It can be imagined that we could process large amounts of data with the aim of creating a model to predict if an individual in Australia will die of cancer. If we imagine that we create a model that has an accuracy of 75%, i.e. it correctly predicts the outcome 75% of the time and that we wish to compare this model to, say one created by another team, should we base the comparison on accuracy alone. If the previously stated statistic is true, then we could produce a more accurate 'model' by simply predicting 'no' in all cases. This model would have an expected accuracy of 91.5% and so perform much better than our first model, yet it would be of no use in identifying anyone likely to die of cancer.

To allow for a more meaningful comparison, a number of other measures exists described below:

- **True Positives (TP):** the number of times the model correctly predicts the number of eggs in the image.
- **False Positives (FP):** the number of times the model predicts a higher number of eggs than were actually present.
- **False Negatives (FN):** the number of times the model predicts a lower number of eggs than were actually present.
- **True Negatives (FN):** the number of times the model correctly predicts when no eggs are in the image
- **Precision:** the percentage where the model found eggs that were actually eggs.

  Precision = (TP / (TP + FP)
- **Recall:** the percentage where the model found all the eggs in an image.

Recall = TP / (TP + FN)

- **Specificity:** the percentage of images that were correctly identified as having no eggs.

  Specificity = TN / (TN + FP)

- **F1 Score:** a weighted average of precision and recall

  F1 Score = 2 x ((Precision x Recall) / (Precision + Recall))

- **Accuracy:** the percentage of predictions that were correct.

  Accuracy = (TP + TN) / Total Tests

For the detection model, these values came out as shown in table 4.1:

Measure	**Value**
Test Cases	3,667
True Positives	1,220
False Positives	1,560
False Negatives	887
Precision	43%
Recall	58%
F1 Score	50%

Table 4.1 Test result summary for the classification model

Note that the system was trained and tested using images the contained between 1 and 25 eggs so the concept of false negatives is not so easy to apply, hence scores for True Negatives, Specificity and Accuracy have not been included in the table above. Instead the F1 score has been taken as the preferred measure for comparison.

The above scores are based on the egg count that the model thought most likely, rather than whether the actual number of eggs fell into the range of values predicted by the model. This means the overall F1 score can be considered to be lower than the score the model would achieve in this less stringent regime. Using the more stringent version of the scoring was done to allow for a more direct comparison to the detection and counting model.

Generally speaking, it was felt that the initial results, as shown above, were promising enough to warrant continuing with the development of the mobile application. Part of this optimism was the realisation that some additional measures could be taken to simplify the problem that the model had to solve. Some examples of these measures are described in section 5.1.3, but, in summary are:

- Prevent eggs from overlapping
- Greyscale the image and increase the contrast
- Add chemical dye to highlight eggs
- Physically segment eggs using a mesh or similar

Figure 4.2 shows examples of the effects of some of these factors on an image.

No overlap Greyscale High contrast

Figure 4.2 Image manipulation effects.

4.1.2 Use of the AI model on an Android device

The mobile app was successfully installed onto an Android device along with a set of pre-generated images to use for demonstration purposes. TensorFlow Lite allowed for the server generated model to be deployed on the mobile device and produce, essentially the same results. There were some small differences due to the fact the operating system and peripheral tools (such as image rescaling) are different. These differences, however were not significant in terms of the model performance.

4.2 Block 2

4.2.1 Use of the AI model on an iOS device

The mobile app was also successfully installed onto an iOS device. Here, rather than pre-load a set of images, additional code was written to allow images to be processed directly from the phone's camera. Since 'real' images of actual samples were not readily available, a set of computer generated images were printed out so that they could then be photographed by the camera.

4.2.2 Evaluating Classification Vs Detection Models

The same measures were used to evaluate the detection model as were used for the classification model. These values, alongside those for the classification model are shown in table 4.2 below:

Measure	Classification Value	Detection Value
Test Cases	3,667	3,582
True Positives	1,220	3,125
False Positives	1,560	248
False Negatives	887	209
Precision	43%	93%
Recall	58%	94%
F1 Score	50%	93%

Table 4.2 Comparison of test results for classification vs detection models

From this it can be seen that the detection model performs significantly better than the classification model. As discussed earlier, however, it is worth noting that the classification model performed well in terms of predicting the correct range of values but less well predicting the precise value. It also performed better when the images contained a lower density of egg. A more significant benefit of

the detection model was its ability to distinguish similar egg typed from one another. E.g. it could distinguish Strongylid eggs from the quite similar looking Nematodirus eggs.

Figures 4.2 and 4.3 show examples of test results using a detection model:

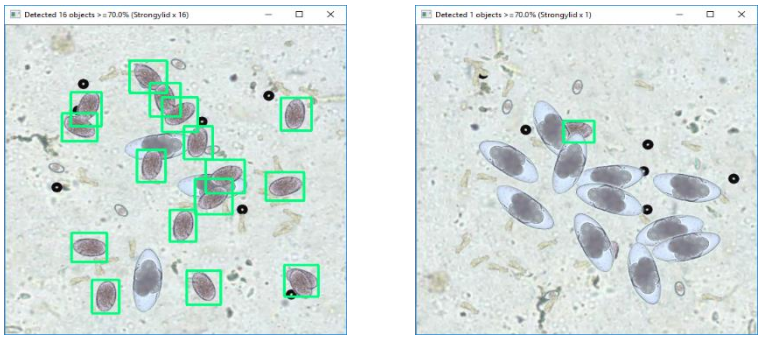

Figure 4.2: Model trained on Strongylid eggs only

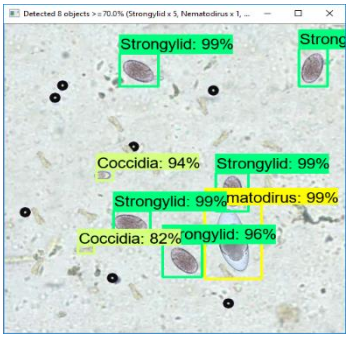

Figure 4.3: Model trained on Strongylid, Coccidia and Nematodirus eggs

5 Discussion

5.1 Use of an AI model to count parasites (Objective 1)

The project proved that is was possible to build an AI model to count worm eggs. It was found that whilst detection models are more successful at correctly predicting specific number of eggs, the simpler categorisation model may well be accurate enough for the purposes of predicting parasite burdens. For example, if the aim is to identify the relative burden across a flock or herd so that animals with natural resistance can be identified then the classification model may be perfectly adequate.

5.1.1 Using real images to create AI models

In both the categorisation and detection cases, the training and testing was performed solely on computer generated images. This allowed for the basic premise to be proved but does not guarantee that the complexity and variation in 'real' images can be handled by such models. Testing these models against 'real' images is the natural next step and some areas of interest in such a study would be to understand:

- Whether animals from different regions, breeds, farms exhibit consistent differences in the colour, opacity and other visual attributes of their faeces resulting in the need for some homogenisation process or the creation of specific models for specific breeds etc.
- Whether there is a need to build a specific device to take and process the images to provide a more consistent image in terms of the focus, lighting, pixel density etc.

Performing such a study is likely to require obtaining tens of thousands of 'real' images to train and test the AI model. An alternative approach may be to obtain a small sample of images from various locations, breeds etc. and then use these as inputs to the image generation tool. This may significantly reduce the time and effort needed to collect images without compromising the applicability or performance of the final model.

As alluded to earlier, there may also be measures that can be taken to simplify the problem, such as applying a software filter to adjust the contrast of objects in an image. One can imagine many other complications that could affect the predictive ability of the model. Two contrived, computer generated examples are shown in 5.1 and 5.2. In Figure 5.1 we see a small number of strongylid eggs swamped by a large number of Coccidia eggs and general debris. In Figure 5.2 we see an image with made up similar looking eggs from two different parasites. Establishing the effects of such complexities on the performance of the models is important prior to attempting to mitigate them so that research and development time is efficiently directed to areas that are likely to have the greatest impact.

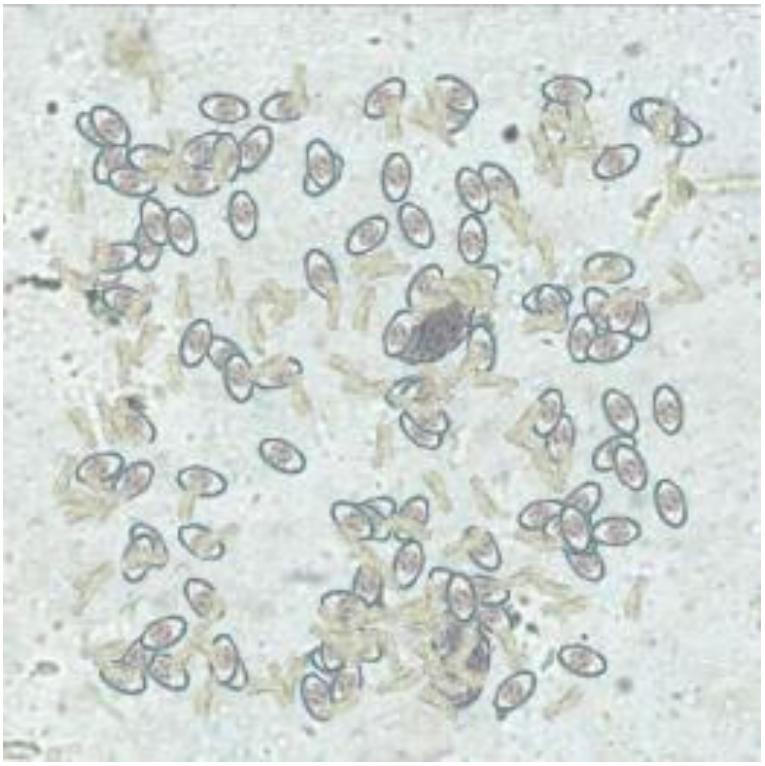

Figure 5.1 A Coccidia and debris swamped image.

Figure 5.2 Similar looking Strongylid and Nematodirus eggs.

5.2 Deployment of AI models to a mobile device (Objectives 2 and 3)

The project proved that is was possible to deploy an AI classification model to run on both Android and iOS devices. It was also shown to be possible to process images directly from the phone's camera. Apart from the initial installation of the application, no network connectivity was required and so, it was also shown that the egg counts could be performed in remote areas without a data connection.

6 Conclusions/recommendations

The project was successful at meeting its objectives. AI models were trained to recognise parasite eggs based on generated images. A model was also deployed, and run on both Android and iOS devices. However, the application still requires further refinement before it can be made available for wide-scale use.

Recommendations for next steps include:

- Testing the models against 'real' images
- Investigation into the effects of complications such as overlapping eggs, colours, contrast, density of objects, etc.

- Trailing the application on a wide range of Android and iPhone models to gauge the minimum specifications required and the effect of the differing capabilities of the devices, such as the pixel density of the images taken by the camera.

7 Key messages

While this project showed that an AI model can be trained to count eggs from a generated image on a mobile device, further research is required before the application can be commercialised. There are opportunities to further optimise the models by testing using real images, and addressing sample complications like overlapping eggs and high-density images.

In addition to refining the model, there remains a challenge around generating consistent images in the field. This may require dedicated hardware that incorporates a mobile device in a chassis that provides consistent conditions at varying times of the day and across diverse locations.

AI and machine learning are viable technologies to assist the industry in manual and time-intensive tasks such as identifying and counting parasite eggs. The wide-spread use of mobile phones also facilitates the ease and accessibility of these applications to those in rural and regional areas.

8 Bibliography

8.1 Software libraries

8.1.1 Mathematical and Machine Learning

The machine learning models described in this report were built on the TensorFlow platform. Official documentation and downloads for this platform can be found at https://www.tensorflow.org/. The ability to run the generated models on mobile devices was provided by TensorFlow Lite, information about this can also be found on the TensorFlow site, here https://www.tensorflow.org/lite/.

8.1.2 Image pre-processing and model execution code.

The Python programming language was used to resize images, process them though the model and return the results to the device specific application (Android and iOS). Documentation and downloads for the Python language can be found here https://www.python.org/.

8.1.3 Mobile device native platforms.

The two device platforms used in this project were Google's Android and Apple's iOS. Documentation and downloads for these platforms can be found here https://www.android.com/ and here https://www.apple.com/ios.

8.1.4 Machine Learning principals and Guidance.

The basic techniques and considerations used for building the models used in this project are summarised in this excellent course by Andrew Ng at Stamford, https://www.coursera.org/learn/machine-learning.

9 Appendix

9.1 Appendix 1 - Sample Images and Image Generation

9.1.1 Source Image and components.

Machine learning projects commonly consist of building an algorithm to perform a task and having the parameters of that algorithm 'tuned' by repeatedly applying it to data and comparing the results with a predetermined expectation. This process, known generally as supervised machine learning, is heavily dependent on large data sets to allow sufficient opportunities for the complexities in the problem to be incorporated.
At the inception of this project, no collection of laboratory images was available for 'training' the model. To mitigate this in a cost effective and timely manner, a set of images where created using an example image as the base. In taking this approach, the following assumptions were made:

- Strongylid eggs in faeces are of a particular size and shape and are indistinguishable from each other
- Strongylid eggs look sufficiently different from the eggs of other worms
- Eggs don't overlap the edges of the frame
- The 'background' is of a relatively consistent colour and opacity
- The accuracy of the count needs to be no higher than to the nearest 150/200 eggs/gram (3 or 4 eggs per image)

Figure A1.1 below show the sample image used as a base for all the computer-generated images used in this study. The approach was to take this image and split out component parts from which additional images could be created. The component parts are shown in figures A1.2, A1.3, A1.4 and A1.5. The base image was taken from a PDF document supplied by MLA; https://web.uri.edu/sheepngoat/files/McMaster-Test_Final3.pdf. The image is that shown as Figure 7.

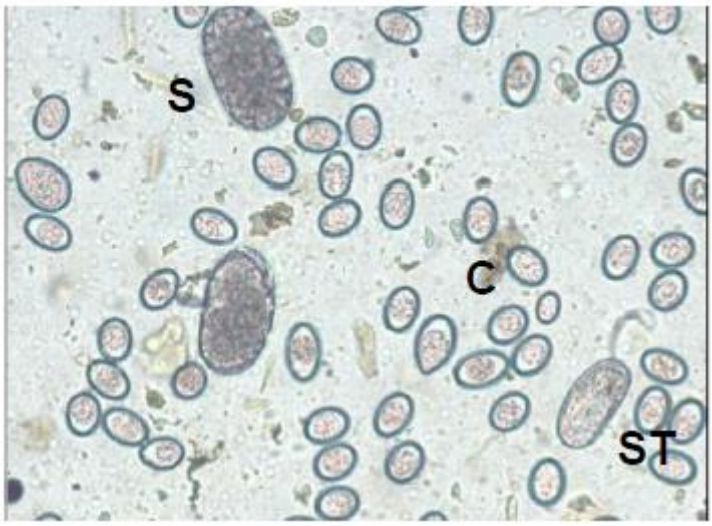

Figure A1.1 Image used as the base for all computer-generated material

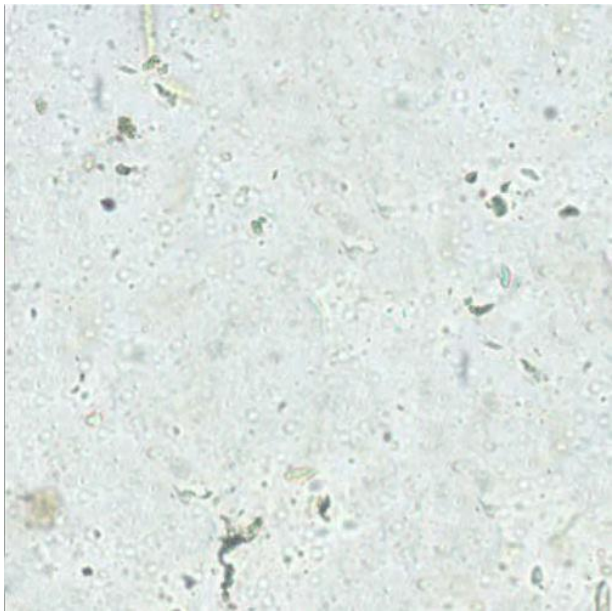

Figure A1.2 The extracted background image on which other items can be placed.

Figure A1.3 Extracted strongylid egg.

Figure A1.4 Extracted Coccidia egg.

Figure A1.5 Extracted grass particle.

Having extracted the components shown above, a computer program was written to allow images to be created with a known number of strongylid eggs and a random assortment of Coccidia eggs and grass particles. In this way, it was hoped to generate a large number of images that would not be too dissimilar to those found in the field. Some examples of these images, with the number of strongylid eggs indicated is shown below in Figure A1.6.

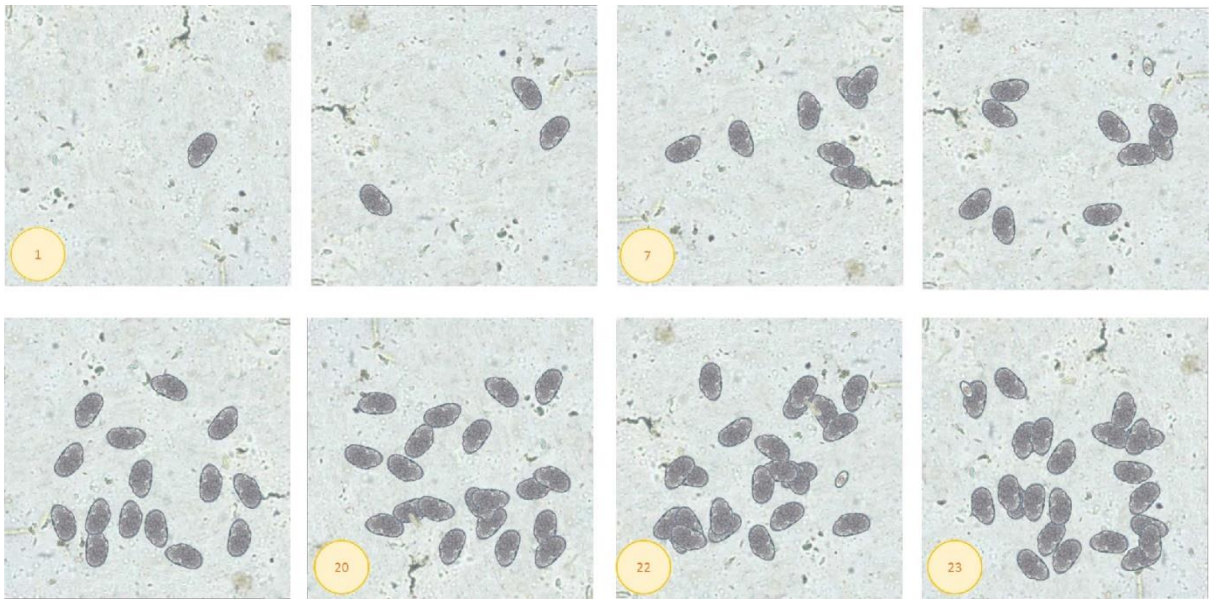

Figure A1.6 Sample generated images.

The computer program introduced additional random elements in to the generated images by applying random rotations to the eggs and grass particles.