**Onyx Project Progress Report**

**Prototype before project began**

While working on the project proposal we built a simple prototype. It was built using the eCraft2Learn machine learning library that had previously been developed by Onyx project members. It relied upon a small number of images for six different conditions. It used the pre-trained machine learning model MobileNet to determine which images are the nearest neighbours to a test image. The labels of those nearest neighbours then determined how it labeled the image and its confidence score. The performance of this prototype gave us hope that a more serious implementation could perform well. The prototype can be found at <https://ecraft2learn.github.io/ai/snap/snap-no-logging.html?project=finger%20nails>

**The Onyx deep learning model**

Despite the fact that our prototype relied upon the often-effective transfer learning technique called *K Nearest Neighbours*, we needed to use a different transfer learning technique that would scale when using the larger number of images necessary to improve accuracy. We needed to replace the less than one hundred images with several thousand. We built the Onyx app to use a professionally pre-trained machine learning vision model called MobileNet and a custom model. MobileNet was used to embed each image into a 1280 dimensional space. We designed and implemented a custom model using TensorFlow.js .We trained it on thousands of embedding vectors generated by MobileNet. Given the embedding vector of a new image (either from a camera, the Internet, or other sources) our model reports the probabilities for each potential label.

We initially obtained images from books, collaborators, and the web. We created this visualization of how these images clustered into groups (see <https://projector.tensorflow.org/?config=https://ecraft2learn.github.io/ai/onyx/projector.json>). The quality of the labelling was hindered by the small number of images. We tried and failed to obtain images of nails from Dermnet (<http://www.dermnet.com>).

We finally obtained tens of thousands of images from a project at the Department of Dermatology, Dongtan Sacred Heart Hospital, Hallym University College of Medicine, Dongtan, Korea [<https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0191493>]. These were collected to diagnose onychomycosis. Like the original prototype, we had originally hoped to support several different labels corresponding to conditions such as melanoma, trauma, splinter haemorrhage, clubbing, pitting, and fungal infection. We, however, could only use the Korean images to classify images as normal, fungal infection, or a condition that warrants a second opinion.

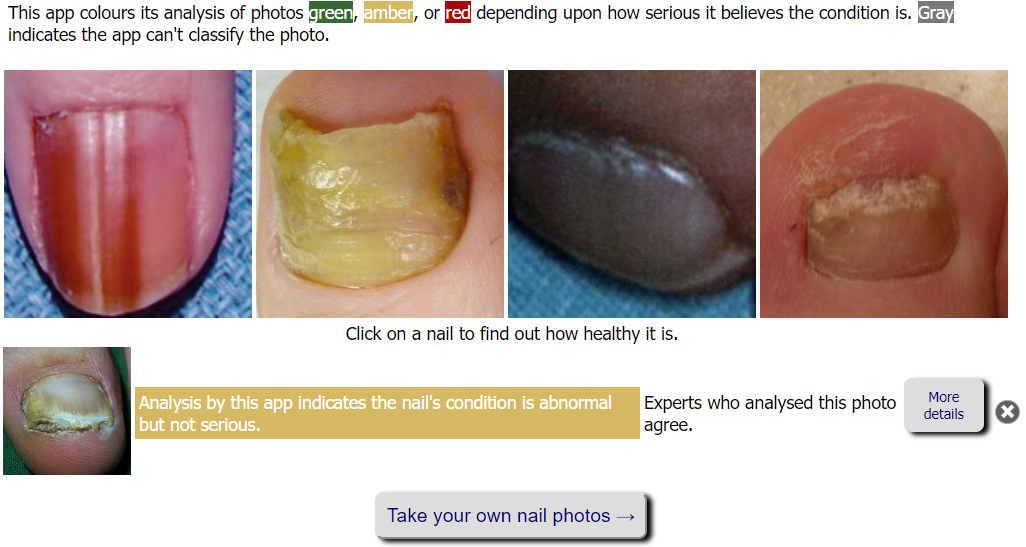
While the Korean project made nearly 50,000 images available we ultimately were only able to use about 5,000. We needed to train the system with roughly the same number of images in each category and there were too few that were an indication of a serious condition. Additionally we eliminated about 15% due to duplications.

Because the app relies only upon static web pages and JavaScript libraries, we were able to host it on the extremely reliable github.io service for no cost. Running the Onyx app in the browser would be too slow to be acceptable were it not for Tensorflow.js. Tensorflow.js is a JavaScript library Google released in 2018 that can accelerate machine learning training and prediction by up to 100 times by using the graphics processing unit (GPU) of the user’s device.

**Development of the Onyx web page and app**

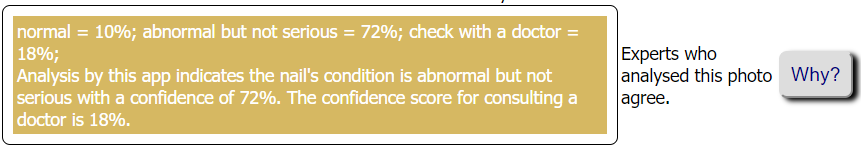
The Onyx app (<https://ecraft2learn.github.io/ai/onyx/>) consists only of static JavaScript, CSS, and HTML files. It can be used to label random images (that were not used in training the model) or to label images from the device’s camera.

In late July and August, we tested an early version of the Onyx app on 25 nail technician students in London and in Oxford. The aim of this testing phase was to get feedback on the functionality and design of the app while simultaneously teaching a tutorial on nail health to the nail technicians. Both, the tutorial and the testing of the app received very positive feedback. Since then, the design and implementation of the interface has undergone several iterations in response to feedback from nail technician students.

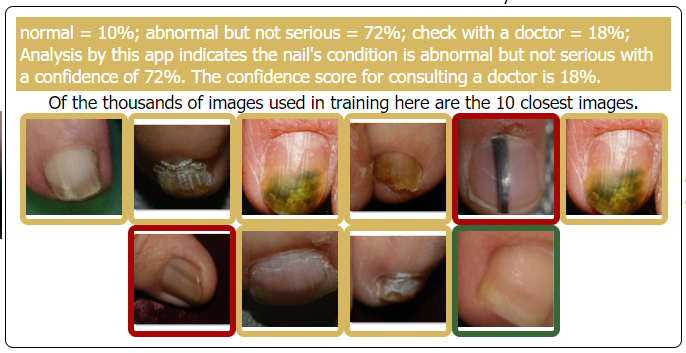


*Figure 1 – The Onyx interface in a desktop browser*

Some of the students who tested the app asked if the app could provide more information about its analysis. They wanted to know the rationale behind an analysis. While deep learning neural networks are well-known to be “black boxes” we partially addressed this demand by displaying the ten closest images and the labels given to those images by experts.



*Figure 2 - Additional information is displayed in response to clicking “More details”*



*Figure 3 - The “rationale” for an analysis (hovering over an image reveals more information)*

**Mobile phone version of the Onyx app**

The Onyx app can be accessed by the browser on a smartphone. Due to the limited screen area and use of touch instead of a pointing device the interface needed to be simplified. Unlike the desktop version, when an image analysis is requested on a smartphone any previous analysis is removed. Also there is a beta version that uses the device’s camera app instead of the more limited camera access provided by the web browser (another request from trialling it with nail technician students). The mobile version does not support drag and drop of images to analyse (though one can use the device’s camera to capture an image displayed on another device). Nor does it support the interface for specifying a cropping region of the image.

The mobile version does support an optional Progressive Web App (PWA) version of the web page. This makes the app behave much like a native app. It has a launching icon. And it can cache all the files it needs locally on the device so after doing so it will function without a network connection. We anticipate there are many contexts where a network connection isn’t available including usage in developing countries.

**Additional work completed**

On 10 December 2019, we were granted an extension until 31 January 2019, which we used to discern between several ways forward. After investigating how to train the model from scratch instead of relying upon MobileNet, we decided instead of “fine tune” the MobileNet model. Even if we used the it was clear that this would be a major effort. MobileNet had been trained on millions of images on large clusters of specialised hardware.

[Fine tuning MobileNet](https://www.tensorflow.org/tutorials/images/transfer_learning) was a better alternative to using the [Oxford Advanced Research Computing facilities](https://www.arc.ox.ac.uk/), which we had initially considered. Fine tuning MobileNet entailed creating a new deep neural network by using the first 83 layers of the professionally trained MobileNet. We replaced the last 5 layers that are not appropriate for our application with 2 to 4 new layers.

The resulting model has several advantages over the previous work that relied upon two separate models: one being unmodified MobileNet and the other being a large model trained on the embeddings produced by MobileNet. Our new model is faster and loads faster than the two models. While disappointingly the accuracy was about the same as before, the literature suggests with more effort it will surpass the two-model alternative.

A full record of the experiments carried out with the Onyx app can be found at <https://docs.google.com/spreadsheets/d/1JTaDiTSOJDbyxqS2fJvHAAebRXBWEWxFSKeZ8kUNKyU/edit#gid=0>