MODULE 2
Computer Vision
DSI Team 4

Marion Boynton, Obed Dzikunu, Catherine Gitau and Martin Page

Image Classification Challenge

Contents

Executive Summary

Background:

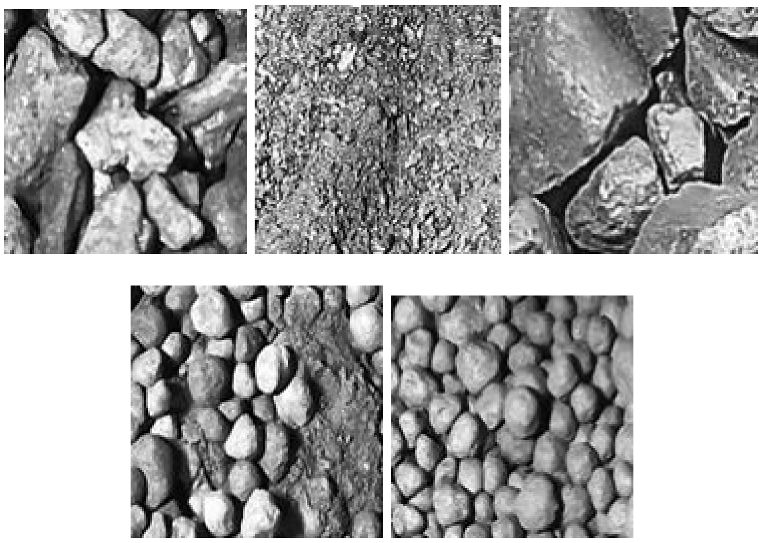
Computer vision deals with how computers gain high-level understanding from digital images. Deep learning is applied in computer vision and is a machine learning method based on artificial neural networks with representation learning. These methods can be used for image classification, which involves assigning a label to an entire image. This is useful for real-world anomaly detection, where a system must create an alert based on the presence of a certain type of image input.

Definition of problem:

A large mining company in South Africa has a system that feeds materials into a furnace. Material handling errors do occur, which leads to inefficiencies in the furnace and drives up production costs. The industry is very cost sensitive. The company requires a digital system that can automatically monitor the input bins for the furnace. An image sampling system is already implemented, and raw image data is available.

Objectives:

1. Train a machine vision classifier that identifies five (5) types of material classes – chip, lump, fines, mixed and pellets – from greyscale images, which should categorise an image input into its respective class



Project Workflow:

The fastai API, a high-level implementation of PyTorch, was used to build the image classification model. After the data were explored, a selection of raw images (about 4500) were manually labelled, and we tried to achieve a balance of the five classes. These images were then used to train deep learning models with the resnet architecture. An outlier detector was built using isolation forests that filters out unlabelled images that are outside of the normal range of a typical good image before image classification. Unlabelled data were used with the trained model (ie a small amount of labelled data) in a semi-supervised learning approach so that more data could be used to improve the model. Finally, test time augmentation, which creates multiple augmented copies of each image, was used when making predictions on the test set.

Step 1: Preparing the Images

Manual Labelling

As a start, a selection of raw images were manually assigned to one of the five classes.

DataBlock API

The fastai DataBlock API was used to create data objects that store information about the data including the type of data, how to get the data, how to split the data, labelling the items, processing methods such as transforms, augmentation and normalisation and whether the data should be batched. PyTorch Datasets and DataLoaders can then be built from these objects and fed to the model for training and testing.

Step 2: Modelling the Data

Learner Class

The fastai Learner class allows one to easily implement a model by defining a Learner object with the data and model architecture and other optional parameters including the loss function and optimisation function. The cnn_learner function can be used to define a Learner object for a convolutional neural network. By default, cnn_learner uses a pretrained model. The model is trained with its layers frozen except for the fully connected layers at the top. cnn_learner also normalises the data by default. A normalisation transform is applied to the data if it has not yet been normalised using the statistics of the pretrained model. The cnn_learner splits layer groups and freezes layers for fine tuning. This approach of using a pretrained model with weights is known as transfer learning.

The idea behind transfer learning is that the lower layers of a model usually process lower-level features and thus can be used for other similar tasks. There are many image classification models available for use in transfer learning, which can be selected depending on the type of images and what low-level features they have. Resnet is a model often used in transfer learning for image classification that has weights from being pretrained on the imagenet dataset.

To train the layers we used the fit_one_cycle method which uses Leslie Smith's one-cycle policy. This involves one cycle of 2 steps where the learning rate is increased from a lower value to a higher value in the first step and then decreased back down to the lower values in step two. A few iterations of this is implemented, after which the learning rate is dropped significantly. The idea behind this is that during the middle of the cycle when the learning rate is higher, the leaning rate acts as a regularisation method to prevent overfitting. The model is able to avoid steep areas of loss and find better flat minima.

To choose an optimal learning rate we can use a fastai learning rate finder function, lr_find. This method works by training the model with a low learning rate and then the learning rate is increased at each mini-batch until the loss starts to explode at a high learning rate. The loss is recorded at each iteration and plotted over the range of learning rate values. Observing this plot, one can select the learning rate to be at a value where the loss plot is decreasing with its steepest gradient before it reaches the minimum. The learning rate should be chosen an order of magnitude before the value at the minimum loss to avoid an explosion of the loss while still achieving fast training. This chosen learning rate is used as the maximum learning rate value when passed to the fit_one_cycle function.

Step 3: Model Evaluation

The fit_one_cycle method returns the training set loss, validation set loss and accuracy score for each epoch. The performance of the model was evaluated by analysing these values and how they changed overtime to ensure there was no overfitting to the training set.

To evaluate the performance of the model, a ClassificationInterpretation object was instantiated to plot a confusion matrix. The matrix compares the actual target values with those predicted by the model. It was observed that a number of images in the "fines" class were classified as "mixed" and this we identified was due to the underrepresentation of both classes. More images in these classes were identified from the unlabelled bunch and manually classified. The model was retrained.

Step 4: Model Improvements

Applying Anomaly Detector

As one of the pre-processing tasks, we decided to build an anomaly/outlier detector to check for unlabelled images which are out of distribution, in this case, images which do not look like rocks (for example, very blurry images or black/blank images). Anomalies are basically events that deviate from the standard, rarely occur and do not follow the rest of the pattern. We use the anomaly detector to filter out images before model labelling.

Anomaly detection algorithms

There are two well-known anomaly detection algorithms:

- **Outlier detection**

  Here, the input dataset contains examples of both standard events and anomaly events. These algorithms tend to fit regions of the training data where the standard events are most concentrated. These methods can help clean and pre-process datasets before applying additional machine learning techniques.

- **Novelty detection**

  These algorithms have only the standard event data points (ie no anomaly events) during training time. During training, these algorithms are provided with labelled examples of standard events (supervised learning). At testing/prediction time, novelty detection algorithms must detect when an input datapoint is an outlier.

For our problem, we decided to use the **novelty detection algorithm** since we had access to standard data points which we had manually labelled/classified. This algorithm uses **isolation forests,** a type of ensemble algorithm that consists of multiple decision trees used to partition the input dataset into distinct groups of inliers (standard data points). Isolation forests accept an input dataset and then build a manifold surrounding the datasets/groups formed. At test time, the isolation forest can then determine if the input points fall inside the manifold (standard events/inliers) or outside the high-density area (anomaly events)
Isolation forest's basic principle is that outliers are few and far from the rest of the observations.

Self-supervised Learning

Self-supervised Learning is an approach in which a supervised learning model is trained on an unlabelled dataset where the inputs and outputs of the model are functions of the data. The model thus learns important features of the data and can be fine-tuned for the actual task in mind. The self-supervised learning task is called the ‘pretext task’ and the following tasks for fine tuning are called ‘downstream tasks’.

For example, for an image dataset, a commonly used pretext task is an ‘autoencoder’, which is a model that takes the image in a very reduced form and converts it into an image as close as possible to the original image.

We considered trying to implement a self-supervised learning method to pretrain our model but because of time constraints we instead decided to focus on other semi-supervised learning techniques as explained below.

Semi-Supervised Learning

Semi-supervised learning is an approach using a small amount of labelled data and a large amount of unlabelled data to train a supervised learning model. There are several semi-supervised learning methods that can be used for computer vision. The main methods are:

1. **Self-training:** this involves training a model on the labelled data and then using this model to predict pseudo-labels for the unlabelled data. The model is then trained on all the data. (e.g. pseudo-label, noisy student)
2. **Consistency Regularisation**: this method aims to ensure input data with added noise, such as image augmentations, return the same predictions. (e.g. p-model, temporal ensembling, mean teacher, unsupervised data augmentation)
3. **Hybrid Methods:** these methods combine ideas from the other two with additional components to improve performance (e.g. MixMatch, FixMatch)

We first built a self-training model by predicting the classes for the unlabelled data with three slightly different resnet-based models and if the average prediction for a class had above 0.9 probability, the image was copied to its predicted class folder. These pseudo-labelled images along with the originally labelled images were then fed to another model for training and the performance was evaluated.

However, we instead decided to implement a hybrid method for semi-supervised learning. Research into the most effective techniques for semi-supervised learning in computer vision indicated that the FixMatch algorithm has great potential to improve model performance. FixMatch is a hybrid method – a combination of consistency regularisation and pseudo-labelling. The FixMatch algorithm first produces pseudo-labels for the unlabelled data by running the model on weakly augmented versions of the unlabelled images. These pseudo-labelled images are kept if they are high-confidence predictions. The model is then fed strongly-augmented versions of the images for prediction. The model is tuned to minimise the cross entropy loss ($H(p, q)$ in the image below) between these predictions and the images’ pseudo-labels.

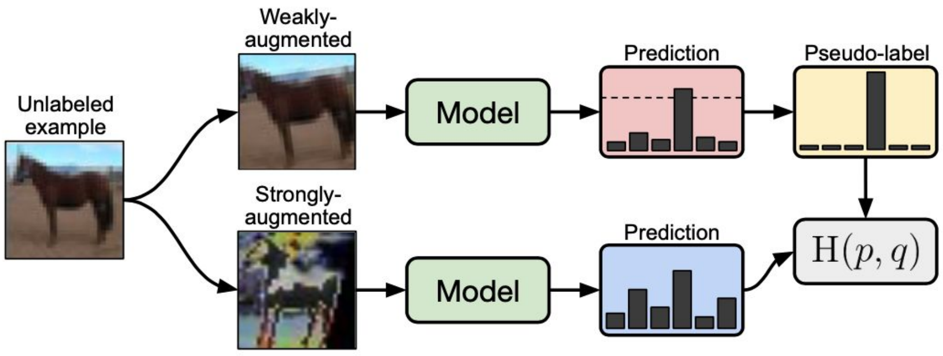


Image source:
https://medium.com/analytics-vidhya/fixmatch-semi-supervised-learning-method-that-can-be-learned-even-if-there-is-only-one-labeled-e7e1b37e8935

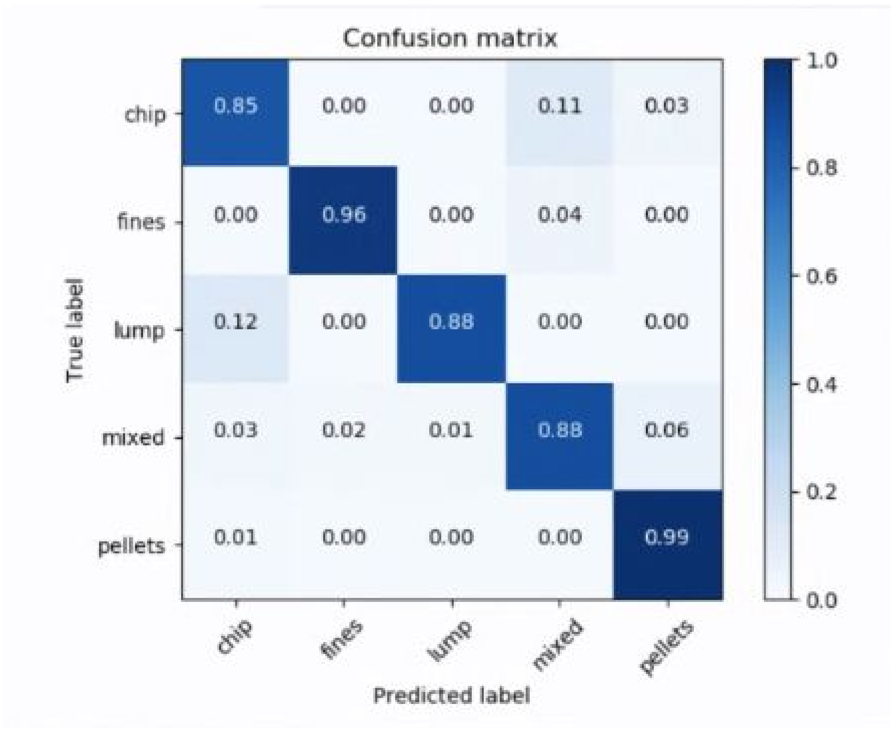
We decided to implement the FixMatch algorithm to improve our model. An open source fastai implementation of the FixMatch algorithm was found so we could use this to apply to the unlabelled data. Our model improved in performance with the FixMatch algorithm.

Test-time augmentation

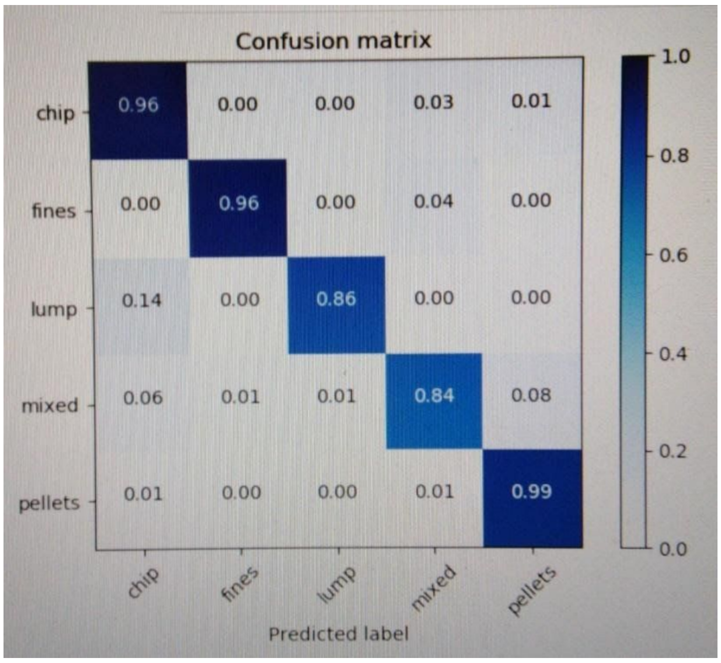
We applied test-time augmentation to the test set when making predictions. Test-time augmentation uses data augmentation techniques when making predictions. Data augmentation artificially expands the dataset by creating copies of an image using manipulation strategies such as zooms, flips, shifts, crop, etc. Test-time augmentation involves creating multiple augmented copies of each image in the test set, having the model make a prediction for each, and then returning an ensemble of those predictions.

Conclusions

First model attempt confusion matrix – resnet50:



Second model attempt confusion matrix – resnet50:

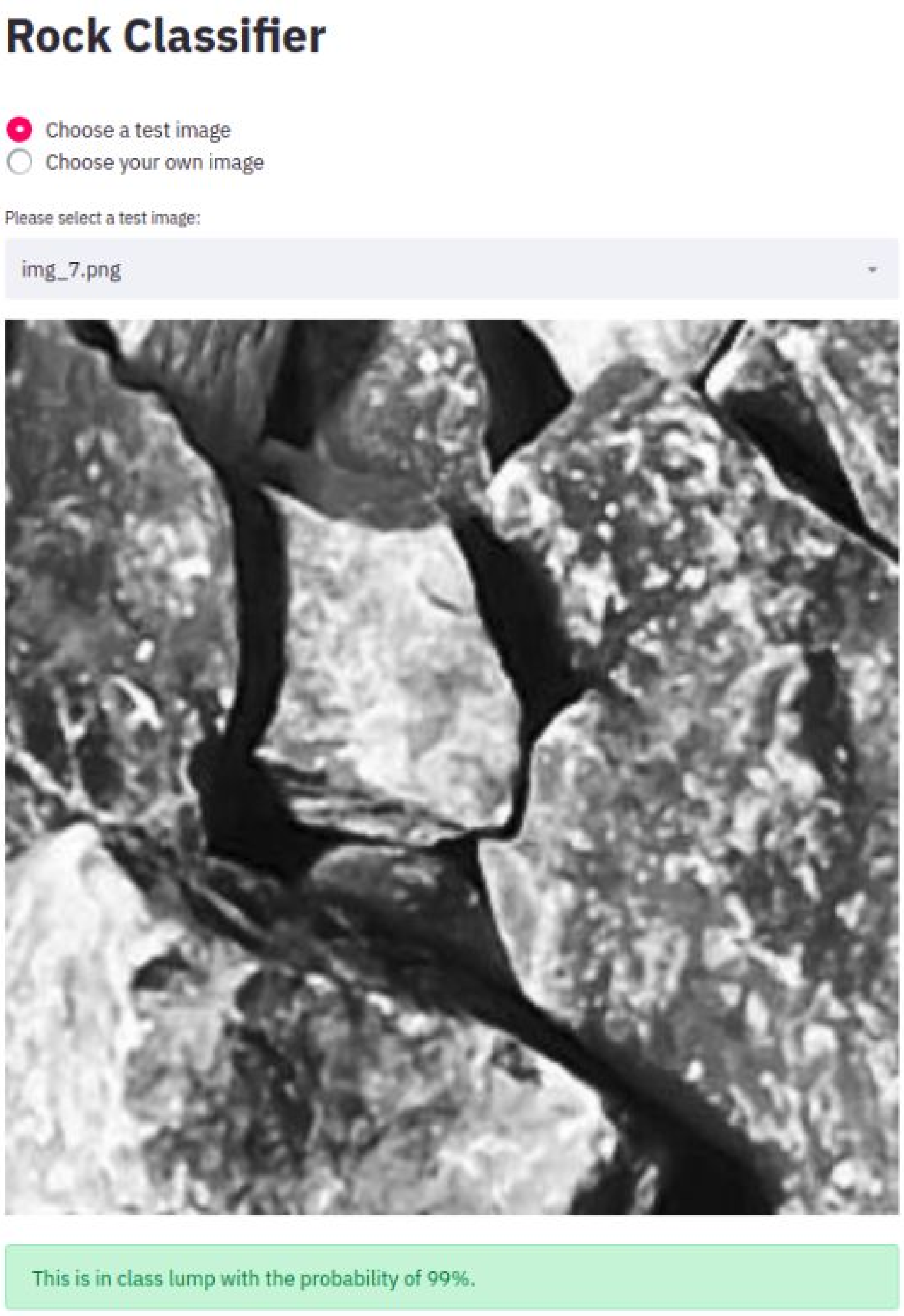


Our model performed well on the test set. The first resnet50 model without any improvements yielding 91.3% categorical accuracy. The second resnet50 model had an accuracy of 92.4%.

We believe that improvements can be achieved by manually auditing the chip and lump classes, as these classes were difficult to classify by hand and we as data labellers may be the source of the ambiguity between these two classes. More examples can also be found of the mixed class, as this class is slightly underrepresented in number and also is very varied class as it can comprise any combination of the other materials.

Extra

We also built a simple app for classifying the rocks, which is available at: https://github.com/CateGitau/Rock-Classifier. Below are some images from the app.



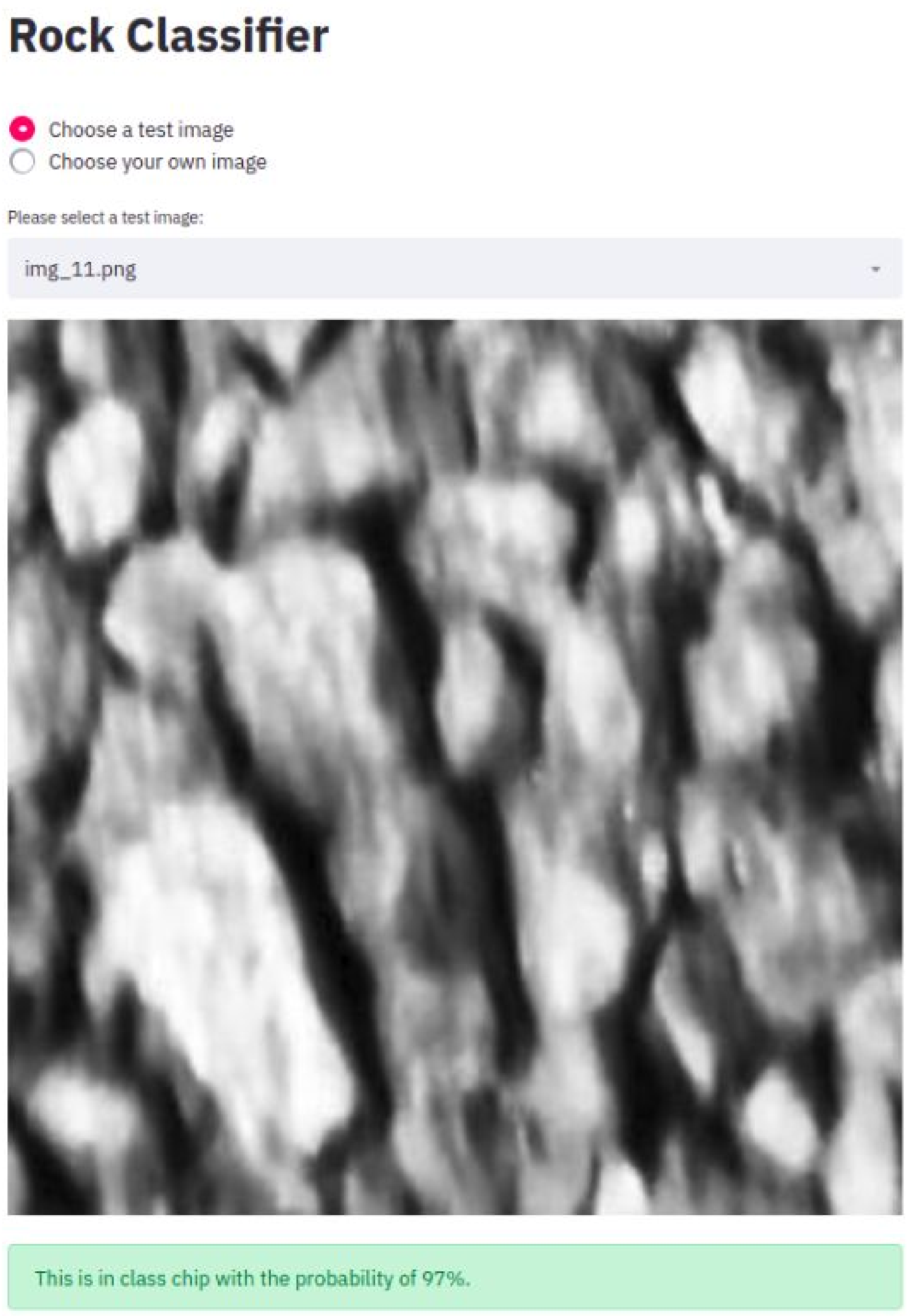
Rock Classifier

● Choose a test image
○ Choose your own image

Please select a test image:

img_7.png

This is in class lump with the probability of 99%.



Rock Classifier

[✓ Choose a test image]
Choose your own image

Please select a test image:

img_11.png

This is in class chip with the probability of 97%.

Team Reflection

Marion

This has been a challenging project but overall great fun and a great learning journey. I have fallen in love with the fastai library and am inspired by their team of people passionate about making deep learning accessible and understandable to the masses. I will certainly be watching more of their videos and reading their blog so I can work with this library more often in the future.

I was also excited to learn about different approaches to dealing with unlabelled data - self-supervised learning and semi-supervised learning. It is incredible to see how many papers in these areas that have achieved improved results on standard datasets have been released just in the past year!

A few of the difficult parts of the project were data labelling at times when I could not decide what class an image belongs to…, slow internet connection, a great challenge when working on google drive, and some general frustrations in library dependency errors.

It has been great working with this team and I feel that our different skill sets complement each other well. I also feel that I personally improved since the last project as I had a better idea of the fast pace of the project process and what was expected for the report and scrum. I was more organized with my time from an early stage and made sure to update and check the trello board regularly.

Obed

The project happens to be my first dive into computer vision and discovering the fastai framework makes me delighted. I now find myself to be familiar with the frequently used terminologies and somewhat confident to approach a computer vision task. My teammates have also been helpful in assisting me to grasp the concepts.

I have explored the scrum process more and made use of the trello board. I have also leveraged more on the varying skill set of my team members to have a coordinated workflow.

The challenges that I faced in the initial stages of the project stemmed from the varied resources that are available in relation to the subject matter. I could not find what particular resource will be good enough to provide a well-rounded introductory information. Also, I had a difficult time working with fastai on google colab since there were a number of significant changes in the libraries in the available versions (ie. 1.0 and 2.0).

Catherine

The project was very challenging but I really enjoyed working on it. I have not worked on a semi-supervised problem before and I had to learn a lot in a short period of time. I also liked the project mostly because it was more practical since there are very few cases out there in the industry where you'd find labelled data and I believe I'll utilise the knowledge that I have gained in the past few weeks in the near future.

One of the challenging things about it though was working with Colab as well as Google drive. It was a bit tasking moving images around the drive because of drive/folder ownership and this was a bit frustrating. Also working on different versions of Fastai also brought up some issues but we were able to work around it. It would have been wise to ask team members to use similar project requirements which will make it easier for the team to share and run code easily.

I really enjoyed working with the team and how we were able to share tasks and collaborate easily.

Martin

This was a very challenging task. Not only have I not worked on a computer vision problem before and had to learn very quickly about deep learning but working with and managing a large dataset was also big challenge. Working on Colabs and Google Drive was especially frustrating. When I was building the model, Colabs could at first detect all the image folders that represented the five classes but when the model was run, only three of the five folders could be detected and the modelling step would throw an error. Checking the file paths again, indeed two of the five folders could no longer be detected. Similarly when running a shared notebook, at first it ran fine and could load the exported model from the file path, and when reset and re-run, the same file path with nothing changed could not be found. I still do not know the solutions to these technical difficulties.

It was also unwieldy to use Google Drive for such a large dataset with tens of thousands of files. I think a future solution would be to create a shared folder so that all the same files are available to everyone but then download and synchronised that folder on each person's local computer so that the files can be managed locally and updates pulled/pushed from/to the cloud. This module has taught me how much time can be taken up by trying to manging files and trying to solve technical difficulties. I have learnt that it is worthwhile to dedicate time to setup robust systems and to develop an efficient process that will then ultimately enable a team to focus more time on the actual modelling task.

Something that even from the previous module I do not think has been reliably established is a good may to collaborate on code. It is still a challenge to have a clear idea of what everyone is working on, to understand their work, and to bring the different pieces together at the end. (As a side note, Google Doc and its compatibility with MS Word is also a headache that seems unnecessary to have to endure...). Indeed, we had a good lesson on compatibility of library versions! A possible solution is to dedicate time to code reviews so that everyone is kept up-to-date and can also learn in the process.

But overall this module was a very enriching experience and I feel that I have learnt a lot about deep learning as well as about the surrounding systems that are also needed for such a project to be carried out. I really enjoyed the computer vision aspect (it's just a pity that the actual vision part was the smallest component in relation to all the other challenges around what needed to be solved to actually create and implement the model). It was also nice to learn from my other team members and see how they approach a new task and how they research different ways to achieve the goal.

Sources

https://medium.com/analytics-vidhya/fastai-v2-an-end-to-end-deep-learning-tutorial-for-arabic-character-recognition-afd42aa218c8

https://medium.com/swlh/algorithms-to-detect-anomalies-in-images-56a1793eba56

https://www.pyimagesearch.com/2020/03/02/anomaly-detection-with-keras-tensorflow-and-deep-learning/