Seedling Image Classification

Transfer Learning using the Inception v3 Convolutional Neural Network

by XXXXX

School of Computer Science, The University of Adelaide

Report submitted for **COMP SCI 7209 Big Data Analysis and Project** at the School of Computer Science, University of Adelaide towards the Master of Data Science

Abstract

A maximum testing accuracy of 98.11% was achieved using transfer learning with the Inspection v3 Convolutional Neural Network. A training process using Cutout as the primary image augmentation technique is shown to consistently produce models that achieve 97.70% testing accuracy. Models trained using Cutout are demonstrated to have testing accuracies 0.50 and 1.89 percentage points higher in both pre-trained and newly initialised networks respectively. Additionally transfer learning was observed to increase the testing accuracy of the Inception network by 1.77 percentage points over a model trained from scratch. Horizontal flipping was seen to benefit when used to train pre-trained networks (0.60%), but diminished the performance of models (0.10%) trained only on the seedlings image set.

Introduction

Increasing crop yields and agricultural efficiency is a globally relevant goal. With an increasing population and poverty gap the necessity of a scalable, cheap and sustainable agricultural industry is paramount. Toward this end, automated crop stewardship is proposed as a means of decreasing the cost of agricultural labour and increasing crop yields. This could be achieved by increasing the amount of care crops receive during their initial stages of growth through the removal of invasive species and the monitoring of development, enabling the real-time addition of nutrients, pesticides or medicine.

Enabling classification of seedlings into species via image recognition is a necessary step towards an automated agricultural industry. Importantly the ability to distinguish between crops and weeds enable the automated removal of invasive species from beds that would otherwise impinge on the growth rate and yield of the crops. Finer grain recognition of crops into their respective species would enable the creation of customised care regimes specific to the seedlings needs without the need for separation of species into separate labelled areas.

Image Database

Originally put together by Aarhus University overall database have been adapted and made available via Kaggle. (Kaggle, 2018) The adapted image set contains 4750 labelled examples and 750 un-labelled testing images, for which the labels have been retained for evaluation on submission. Images contain one of twelve different plant species separated into two main sub-classes of seedling; broad-leaf and grass. Seedlings are photographed at different stages of their development adding to the fine grained nature of the classification problem. Many aspects of image capture were heavily controlled so that clean and consistent images were obtained. To achieve this a special apparatus was designed to shield from external lighting, fix the cameras distance from the seedlings and reduce vibration during capture. This leads to a very clean images overall, some examples of images included in the database are shown in Figure 1. (T. Giselsson, 2017). However the captures were preformed in bulk such that an entire bed of seedlings is captured in one frame and is then sliced into smaller images containing each seedling. This leads to some inconsistency in the resolution, even when re-scaling all images to 299x299 pixels for analysis, leaving some images quite blurry. Additionally some images contain edging from the bed which has ruler marks along then that are not present in others. Hopefully this feature is spread overall different classes uniformly so it is not regarded by the network as a significant feature in classification process. Figure 2 shows the frequency of which each class occurs in the training set of images. Showing about a 1:3 disparity between the least and most represented classes.

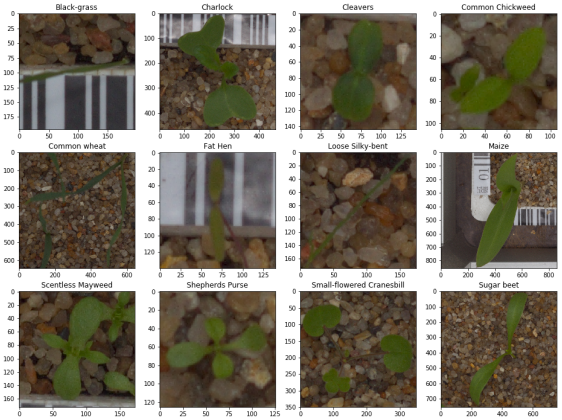

Figure 1: Example of an image from each class in the image data base

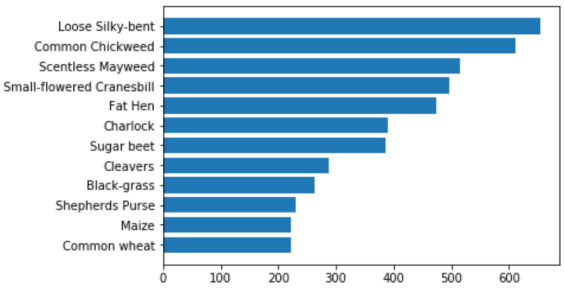

Figure 2: Representation of each class in the set of training images

Implementation

All testing was performed in Jupyter Notebook using a Python kernel. Pytorch was used as the primary deep learning library, responsible for the imported Inception network, image batch loading and transformations, loss functions and their optimisation, updating the network and utilisation of Cuda acceleration.

Using these custom pipelines for online image pre-processing were built which are applied to each mini-batch of images as they are loaded for use. A separate pipeline is used for each stage the images are being loaded in. Four pipes were constructed for training, validation, testing on labelled images and testing on unknown images respectively. The training pipeline is the only one that image augmentations were applied to images besides re-sizing them to a consistent resolution. These pipelines were also used to dynamically create random training and validation sets from the available labelled images at runtime. To implement this each of the training and validation sets are given the same pool of possible images but a different randomly initialised sampler is used on each set that restricts the pool to a section dictated by the sampler. The validation set was chosen to be 15% of the total available images.

Simple training and prediction functions were implemented allowing a network to be trained on or predict labelled for images loaded with the previously outlined pipelines. The training function maintains a list of accuracies and losses at each epoch over the course of training which is returned. Two models are saved as a result of training, one that is the network after its final epoch and another that scored the highest validation accuracy over the course of training. Another function to generate confusion matrices was built using the predictions made by the network on labelled samples. Some utility functions where also defined in order to help with network testing and evaluation. Displaying sample images to visualise how data is loading and that image augmentation/preprocessing is working as intended, plotting the confusion matrices as a simple heat-map and graphing the training of networks. Additionally a chunk which is able to directly map the predictions of a network to the species names and respective image file then export the resulting data frame as a comma separated value file for submission to Kaggle for evaluation was implemented.

One of the image augmentations tested was an imported package called Cutout. The function will randomly set a region of a Pytorch tensor to zero, effectively blacking out a specified size and number of holes in an image before they are given to the model for training, Figure 3. By doing to this the network will be exposed to the same image with different sections redacted from it, forcing classification of the image on different features. It is thought that by doing this the network can begin to weight smaller more fine features over larger structural ones, improving the regularisation of the classifier. However it is important for the practice to work for the cut out region to sometimes not obscure the image at all and it is why the decision to allow the crop to lie outside the region of the image was made by the developers. (DeVries and Taylor, 2017)

Figure 3: Examples of images after processing with Cutout

Deep Learning

For this task the Inception v3 network imported from PyTorch was used in all testing. This convolutional neural network consists of smaller inception modules linked in series and parallel to form the overall classifier. Each module itself has a series of factorised convolution layers in parallel that allow the weighting of the different kernel sizes, Figure 4. The kernel size is usually a parameter that needs to be tuned to the classification problem that represents roughly how large the features that are extracted by the particular layer are. In having these different kernels in parallel the classifier can maintain its ability identify features that are very local in nature as well as boarder more global features. This leads to robustness when the object to classify is present in images with a different location or relative size. The network is of particular interest because it is able to preform as well as comparatively denser networks, with less parameters and thus less computation. (C. Szegedy, 2015) Although ResNet is considered state of the art for most plant based classification problems the Inception v3 architecture was used as it is under represented in literature on this class of problem and its reported robustness to lower resolution input. (Waldhcen, 2018) (C. Szegedy, 2015)

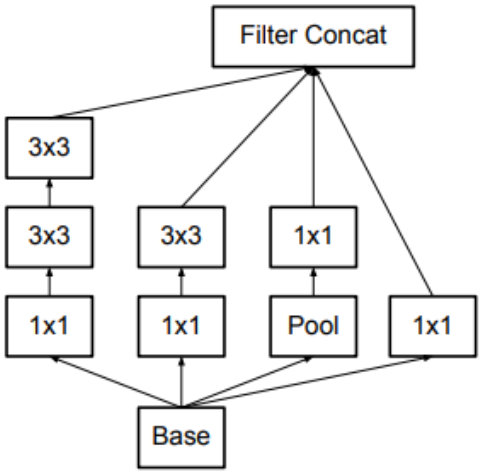

Figure 4: Example of an inception module with factorised convolution layers. (C. Szegedy, 2015)

Loss was evaluated using the Cross Entropy Error function as this includes dependence on the probability of a given label, helping to alleviate the observed differences in sample class frequency in the image data base, Figure 2. To optimise this loss function Stochastic Gradient Decent was used as a computationally cheap optimiser that also has embedded resilience to local minima.

Tuning

Experimental tuning of the learning rate was performed by observing the networks convergence after 15 epochs. Difficulties in evaluating this were due to the use of a small batch size of 25. This was initially thought to be the maximum capabilities of the hardware being utilised and would lead to the network undergoing approximately 200 weight updates in a single epoch. This often lead to training curves that start at very high accuracies. The experimentation with learning were done under this regime, in general it was found that a learning rate of 0.02 showed strong convergence within the first 10 epochs. It is interesting to note that the overall performance of the network is un-affected when a slower learning rate is set for the pre-trained network versus the classification layer. In fact the largest accuracy was observed when both the network and the classification layer learnt at the same rate of 0.02, Table 1. In this table it can also be seen that the network does not tend to benefit from training once it has achieved convergence but also does not appear to over-fit and degrade either. This is likely an effect of the exponential decay scheduler used on the learning rate which was scaled to have a half life equal to a third of the total number of epochs when this number was over 100. Tests under 100 were done using a constant learning rate. This initial phase was used to inform the training regime for the main experimental work performed around the model. One of the models produced during the testing of the implementation of an image augmentation technique is currently the best model that has been found when using testing accuracy as its method of evaluation and was included in Table 1. This model reveals the potential of the network but without being able to reproduce this result it has ended up as an erroneous example.

Table 1: Testing accuracies scored on Kaggel for various hyper-parameters and image augmentation techniques recorded during initial network training

Hyper-parameters				Image Augmentation		Accuracy (%)
Batch Size	α-Network	α-Final Layer	Epochs	H Flip	Cutout	Testing
25	E-2	E-2	150	N	N	97.10
25	E-2	E-2	300	N	N	96.73
25	E-2	E-2	300	Y	2x75	97.23
25	E-2	E-2	15	Y	1x150	98.11
25	E-4	E-2	150	Y	1x150	97.61
25	E-4	E-2	150	Y	1x150	97.73
24	E-4	E-2	150	Y	1x150	96.35

By implementing memory de-allocation after each function is called the maximum batch size was able to be increased to around 100. A batch size of 94 was chosen as it provided some buffer for other programs running on the computer. This enabled better monitoring of the initial convergence of the network, evidenced by the noise introduced by stochastic gradient decent being able to be seen in the training curves, Figure 5.

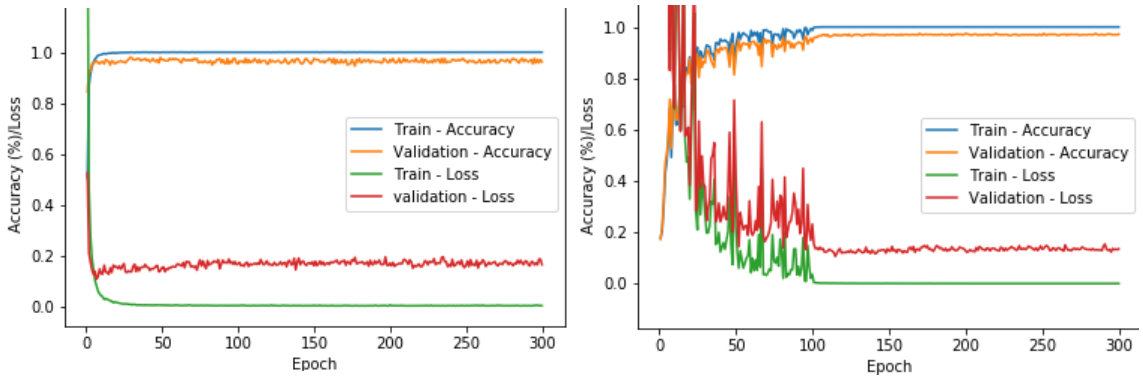

Figure 5: Training curves for two models, the left plot is a model using a batch size of 25e and the right a batch size of 94

Table 2: Testing accuracies scored on Kaggel for hyper-parameters and image augmentation techniques

Hyper-parameters				Image Augmentation		Accuracy (%)
Pre-Trained	Batch Size	α	Epochs	H Flip	Cutout	Testing
Y	94	E-2	100	Y	1x200	97.73
Y	94	E-2	100	N	1x200	97.23
Y	94	E-2	100	N	N	96.73
N	94	E-2	300	Y	1x200	96.73
N	94	E-2	300	N	1x200	96.85
N	94	E-2	300	Y	N	94.33
N	94	E-2	300	N	N	94.96

Table 2 shows the experimental results obtained for training a pre or untrained network with image augmentation techniques turned on or off. These tests where preformed for 100 epochs when using a pre-trained model and 300 epochs on an untrained model. All trails had a consistent batch size of 94 and global learning rate of 0.02. The two augmentations considered where a random horizontal flip of the image and Cutout.

Interestingly, these results indicate that horizontal flipping on its own will increase the overall performance of a pre-trained network using Cutout by 0.50 percentage points but decreases the observed testing accuracy when training from scratch by 0.60 percentage points. It was also seen to decrease the testing accuracy by 0.10 percentage points when used in conjunction with Cutout when used on an untrained network. As of yet no concrete reason for this has been found but an initial hypothesis is that the pre-trained network may have some robustness to translation, rotation and mirroring of images from its previous training, as opposed to the freshly initialised network which struggles with these variations and requires more training to form resilience to these effects than is possible to achieve with the limited size of the training set available.

Another outcome of this experimentation shows that the pre-trained network alone adds to the observed testing performance of the model by 1.77 percentage points. This finding supports the concept of transfer learning. Even though the image classification task the network was previously trained to perform has little correlation with the current task, the features it learned to extract from those images can be used on other classification problems to gain advantages over models that where not exposed to the initial training.
These results show that Cutout provides a 0.50 and 1.89 percentage point increase in accuracy when used during the training of a pre-trained model and an untrained model respectively. The respective training curves, Figure 6, show that the improved regularisation Cutout offers increases the time the model takes to over-fit the training data and also decreases the gap between the training and validation accuracies. Initially this appears as a noisy training curve but once convergence is reached the difference is clear, the model trained with Cutout has a smaller generalisation gap and lower validation loss.

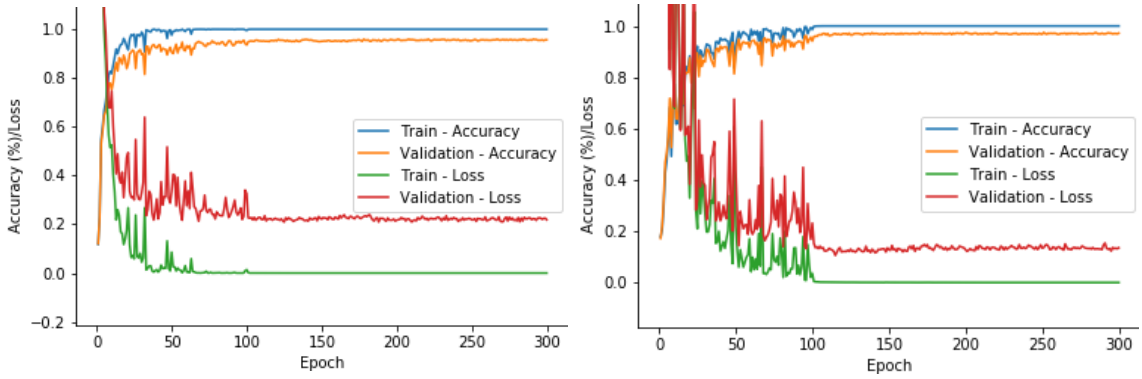

Figure 6: Training curves for two models with no pre-training. The left plot has no image augmentation and the right uses Cutout

Confusion matrices were used to understand the incorrect labels models are assigning to classes. These where then used in an attempt to create weights for the losses of frequently mislabelled classes to reward the model for resolving the confusion. An example of this can be seen in Figure 7, this shows a matrix after fifteen epochs of training being used to inform the next fifteen epochs resulting in the net matrix. From this it can been seen that the re-weighting may have had some impact on the training, increasing the correct classifications overall, but in many cases after sufficient training this confusion becomes very small. Figure 8 shows the confusion matrix of a typical model produced. In this case we see the major source of confusion is from seven incorrect labels of black-grass given to loose silky-bent. These two classes can be seen in Figure 1 and are both very similar species of grass. The reason that the inverse labelling does not occur as frequently is likely a result loose silky-bent's representation in the image set is larger, giving it a more likely prior probability than black-grass.

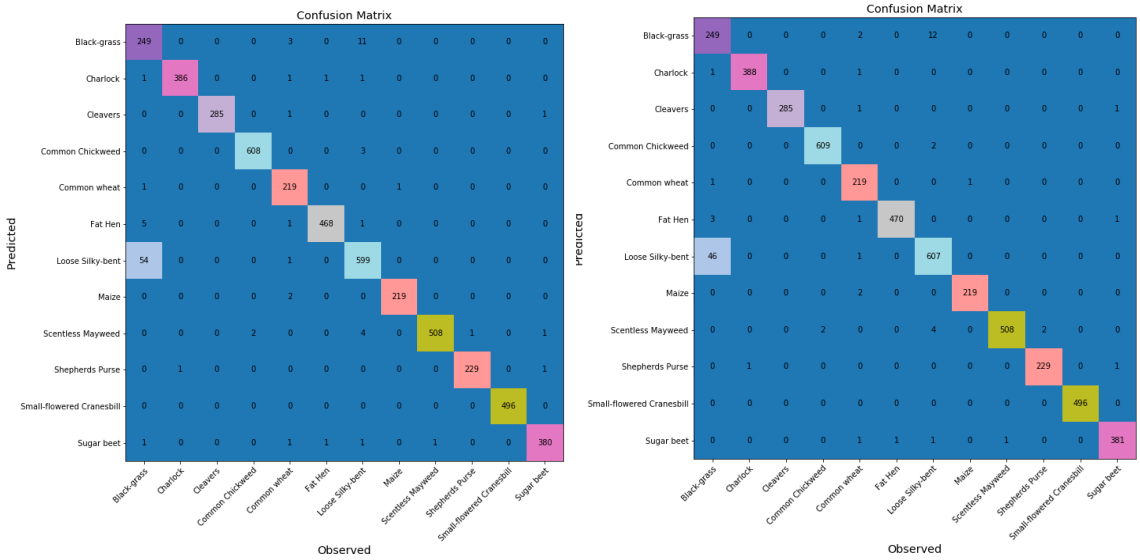

Figure 7: Confusion matrices highlighting a models incorrectly labelled samples. The diagonal represents the number of correctly classified examples. The matrix on the left is after an initial training of fifteen epochs. This matrix was then used to give the most frequently misclassified class a higher weighting in the loss functions evaluation. This adjusted loss was then used to train the network for a further fifteen epochs yielding the matrix on the right

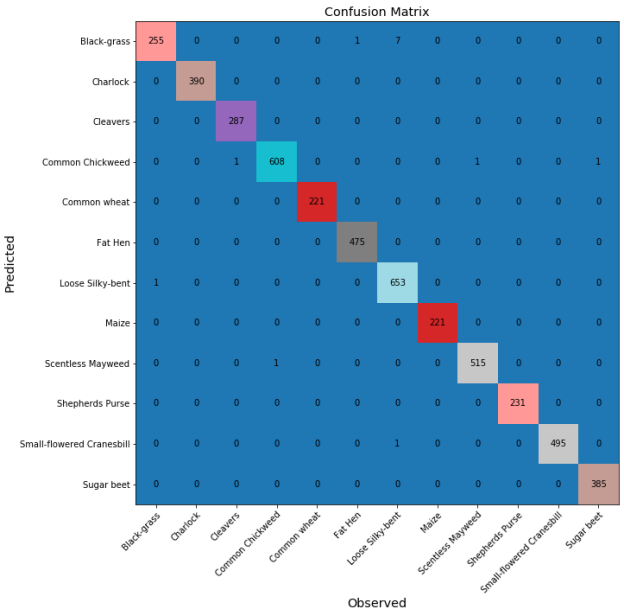

Figure 8: A confusion matrix from a typical model produced by the outlined training regime

Evaluation

A few things were considered when evaluating the models trained to complete the task at hand. Primarily testing accuracy was used as the ultimate metric to determine a models ability to classify images of seedlings but the repeatability of training successful models and the generalisation of those models to real world applications are considered. By testing accuracy alone, the model trained during image augmentation achieved 98.11% accuracy, but was unable to be replicated. Consistently models that achieve around 97.70% testing accuracy can be trained using the hyper-parameters experimentally determined above and basic image augmentation techniques. In most cases the validation accuracy is a good predictor of a models achievement in testing. This implies the testing images are likely sampled from a distribution that closely resembles the training data. Images of this clean and controlled nature may not be available in real world application and it is a natural extension of this investigation to train and test the models built on these controlled images on the wild seedling database also put together by Aarhus University. Improvements to the model and its training that could be made include: testing a wider range of image augmentation techniques, using a more informed loss optimisation processes than stochastic gradient decent and enabling an effective maximum batch size increase by changing how often the network is updated.

References

et al. C. Szegedy. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015. URL `http://arxiv.org/abs/1512.00567`.

T. DeVries and G. Taylor. Improved regularization of convolutional neural networks with cutout. *CoRR*, abs/1708.04552, 2017. URL `http://arxiv.org/abs/1708.04552`.

Kaggle. Plant seedlings classification, 2018. URL `https://www.kaggle.com/c/plant-seedlings-classification`.

et al. T. Giselsson. A public image database for benchmark of plant seedling classification alogirthms. *CoRR*, abs/1711.05458, 2017. URL `http://arxiv.org/abs/1711.05458`.

J. Waldhcen. Automated plant species identificationtrends and future directions, April 2018. URL `https://journals.plos.org/ploscompbiol/article/metrics?id=10.1371/journal.pcbi.1005993`.