CITS4404 - Artificial Intelligence and Adaptive Systems
Semester 2, 2020
Team 17 – Wheat Detection

Name	Student Number
Tom Chan	23023191
Shengxin Zhuang	21463227
Limin Zhou	22634156
Xiaochen Bi	22666881

Introduction

Wheat grain plays a vital role in human's food chain. Many foods are made of this common grain, such as the bread and cereal we have in the morning. Leveraging modern technology to grow and manage wheat fields would be beneficial to the humankind. Assessing the health and maturity of the crops quickly and accurately is crucial for the farmer to make the right management decisions. For this purpose, many research has been focusing on using artificial intelligence to assist in the matter.

The objective of the project is to build a wheat detector to find wheat heads in a given image accurately. The dataset used in the project is from the Global Wheat Detection competition on Kaggle [4]. Our wheat detector is based on the pre-trained Faster RCNN model with a ResNet-50-FPN backbone from PyTorch library for speed and accuracy [10]. The team have experimented noise augmentation and many different hyperparameter values while training the model.

Faster RCNN

Region-based Convolutional Neural Network (RCNN) was proposed by Ross Girshick et al. [1]. RCNN uses selective search for regions of interest in the image; they are also called regional proposals. Finally, these regions are then fed into a CNN for classification.

A faster version of RCNN – Fast RCNN was introduced later, which overcome some of the drawbacks of RCNN [2]. Instead of feeding region proposals to a CNN. A convolutional feature map was generated by feeding input image directly to the CNN with the region of interest pooling to identify the regional proposals. Next, the proposed regions are passed to fully connected layers. In the final layer, a SoftMax classifier is used for object classification and a bounding box regressor is used for object localisation.

Unfortunately, Fast RCNN still has one major drawback, which is the time-consuming selective search. To address the issue, Faster RCNN was introduced by Shaoqing Ren et al. [3], which replaced the selective search with a region proposal network (RPN). Region proposal network is responsible for predict region proposal and another CNN to classify based on the region proposal. As a result, Faster RCNN can achieve near-real-time object detection.

ResNet

Degradation is a common problem while training a traditional deep network. It is a phenomenon that the network is saturated and in a decline of accuracy when the depth of the network increases [11]. Residual Network (ResNet) eliminates the degradation problem by using skip connection, which refers to any layer in the model that decreases the performance will be skipped [11]. It enables the ability to train a deeper and more accurate network.

Methodology

The pre-trained ResNet-50 model used in the project has been trained on Common Objects in Context (COCO) train2017 dataset [9] and Stochastic gradient descent (SGD) optimizer is used. The script used in the project is based on an example on Kaggle [5, 6] and a tutorial from PyTorch website [7, 8] with some modifications to enable noise augmentation and validation. In the project, we have experimented with noise augmentation, different learning rate and weight decay values to find the best result. All models are submitted to the Global Wheat Detection competition on Kaggle and the performance of the model is determined by the public and private score.

Learning rate is one of the critical hyperparameters. If the learning rate is too low, it will take a long time to find the optimal solution; if the learning rate is too high, it will lead to divergent behaviours. With an appropriate learning rate, the optimal can be found in a reasonable timeframe.

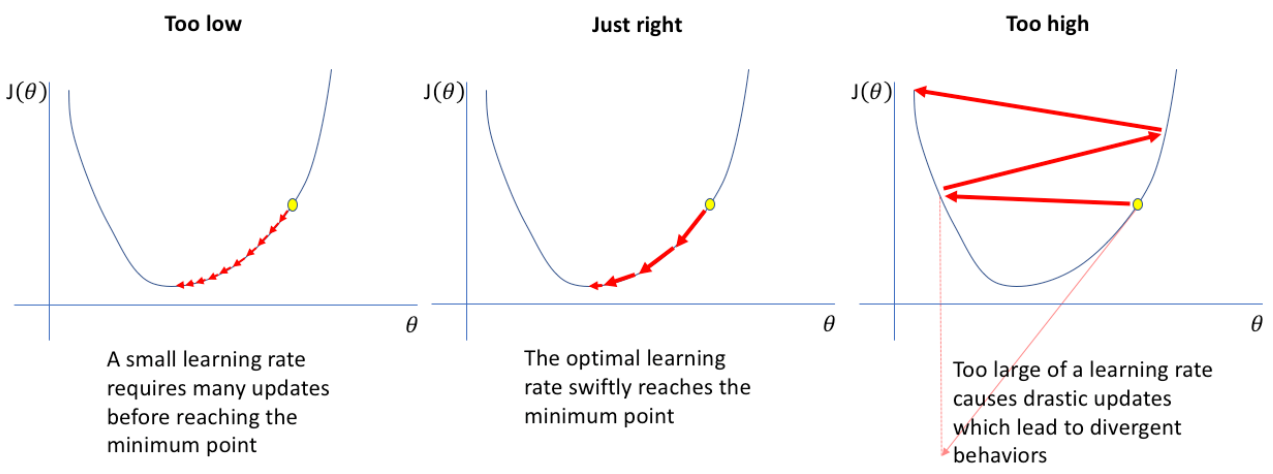
Too low
Just right
Too high
J(θ)
θ
A small learning rate requires many updates before reaching the minimum point
The optimal learning rate swiftly reaches the minimum point
Too large of a learning rate causes drastic updates which lead to divergent behaviors

Figure 1 – The impact of learning rate [12]

Weight decay is another useful hyperparameter which can be used for avoiding underfitting and overfitting. Underfitting indicates the model is oversimplified and unable to generalise the dataset; meanwhile, overfitting indicates the model is too complex so it can only work well on the train data. Weight decay constrains the complexity of the model to avoid those problems.

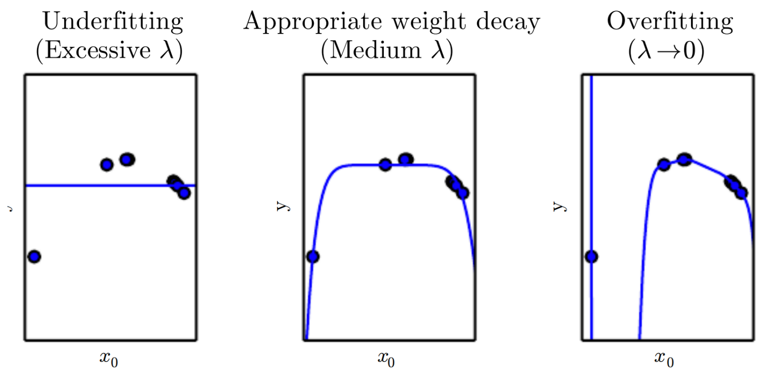

Figure 2 – The impact of weight decay [13]

Noise augmentation is a technique for avoiding overfitting. In reality, the images captured by cameras always contain noise. Noise augmentation would allow the trained model to work with noisy input. In this project, it is done by adding white noise to the train set. During the training, both the original and noisy train sets are fed to the model.

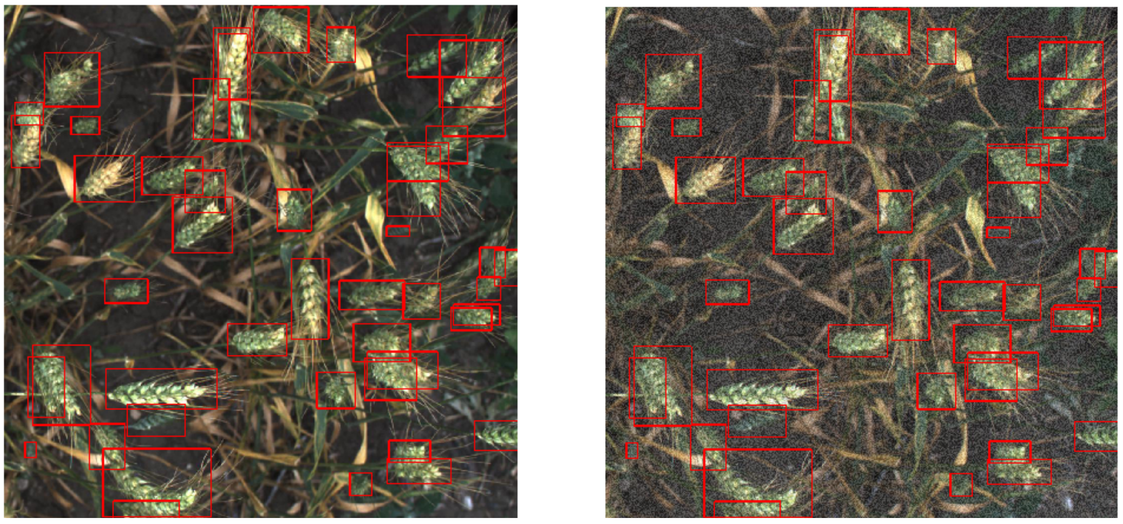

Figure 3 – Noise augmentation (left: original image, right: modified image)

While training a model, it is important to monitor the performance of the model over time. Hence, a validation step is added to the model. The validation set is a subset of the train set which will not be fed to the model to avoid bias. For each epoch of the training, the training data is fed to the model, and then we use the model to predict the label on the validation set and compare the predicted label with the actual label. It means that there are a loss and an accuracy value for each epoch. If both the loss and precision are decreasing, the model is overfitting; if the values are fluctuating and never converged, the model is underfitting.

The output of the model consists of a confidence level, and a bounding box for each wheat head detected. The confidence level has a range between 0 and 1, where 0 represents 0% confidence level, and 1 represents 100%. If the confidence level is greater than 0.5, the region is classified as a wheat head, otherwise, background.

Result

Our first experiment is to train the model with different learning rates to find the optimal learning rate for the wheat dataset. All other hyperparameters are set to the same value, thus the scores are comparable. Weight decay was set to 0.0004. As shown in figure 4, the model is not performing well at the beginning, then the model starts to stabilise from 0.00248 to 0.00349, and lastly, the performance is going down again. The best model is highlighted in red.

Learning rate	Private	Public	Precision
0.000248	0.5313	0.6078	0.484
0.00045	0.4668	0.5431	0.445
0.00248	0.563	0.6429	0.525
0.002985	0.565	0.6449	0.526
0.00349	0.5608	0.6442	0.53
0.0045	0.5518	0.6127	0.531
0.00551	0.5539	0.6162	0.528

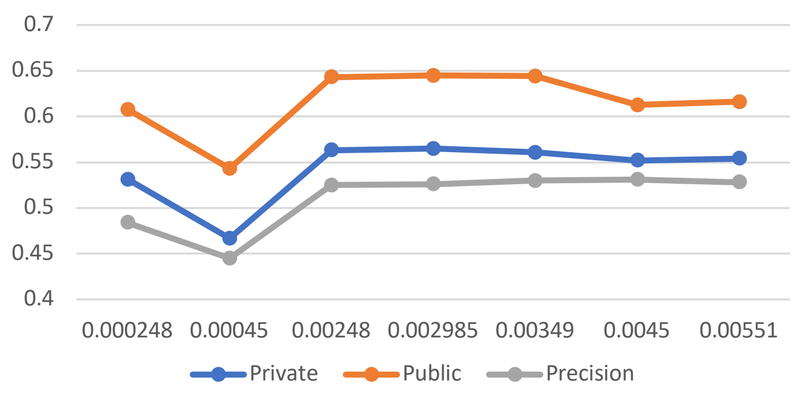

Figure 4 – Model performance with different learning rates without noise augmentation

The team conducted the same experiment with noise augmented dataset. The result does not show any significant improvement over the original dataset. However, the model with noise augmentation seems to perform better with increasing learning rate. In comparison, the model without noise augmentation started to overfit, which leads to a decrease in performance.

Learning rate	Private	Public	Precision
0.002985	0.5574	0.6276	0.527
0.00349	0.5549	0.6035	0.513
0.00551	0.5619	0.648	0.537

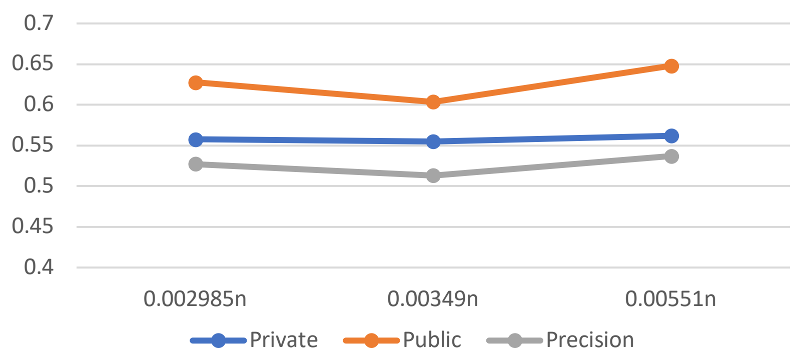

Figure 5 – Model performance with different learning rates with noise augmentation

Learning rate	Private	Public	Private (noise)	Public (noise)
0.002985	0.565	0.6449	0.5574	0.6276
0.00349	0.5608	0.6442	0.5549	0.6035
0.00551	0.5539	0.6162	0.5619	0.648

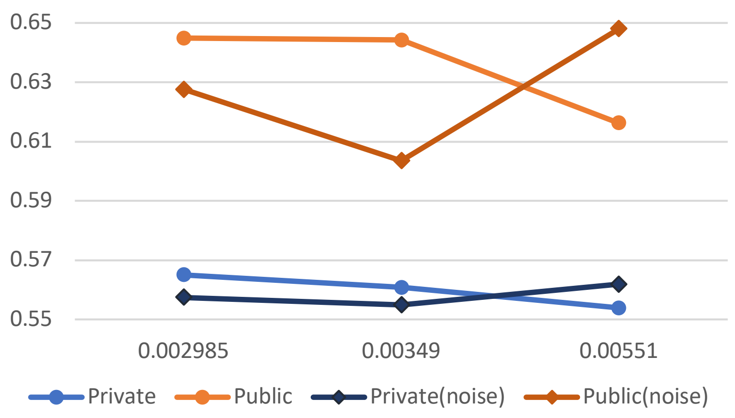

Figure 6 – Model performance comparison between with and without noise augmentation

The team has also tried different weight decay values on the best model, which is the model with learning rate 0.002985 and without noise augmentation. With a weight decay value of 0.1, underfitting occurs. However, other values from 0.01 to 0.00001 shows a similar result.

Weight decay	Private	Public	Precision
0.1	0.2724	0.3204	0.285
0.01	0.5428	0.6199	0.513
0.001	0.5629	0.6434	0.53
0.0004	0.565	0.6449	0.526
0.00001	0.5564	0.6252	0.52

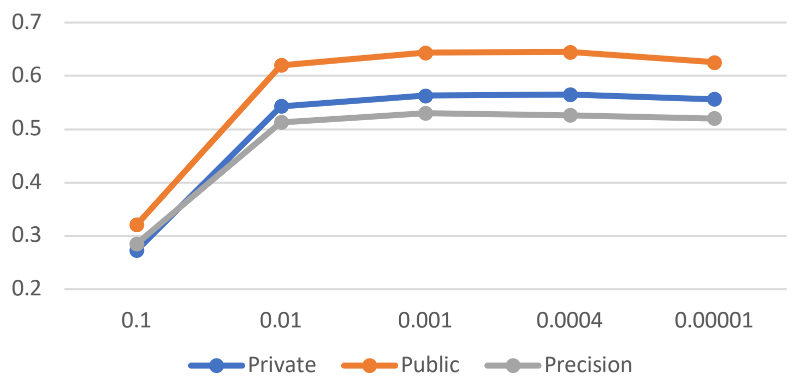

Figure 7 – Model performance with different weight decay values with learning rate 0.002985 and without noise augmentation

During the training, validation is used for estimating the performance of the model at every epoch. We use the precision score as the comparison baseline. Figure 4, 5 and 7 show that the precision score is close to the private and public score so it should be reliable for performance evaluation and estimation. The loss and precision graph of the worst model (figure 8), the lines are diverging, which suggests the model is underfitting; in the best model (figure 9), the lines are converging, no underfitting or overfitting occurs.

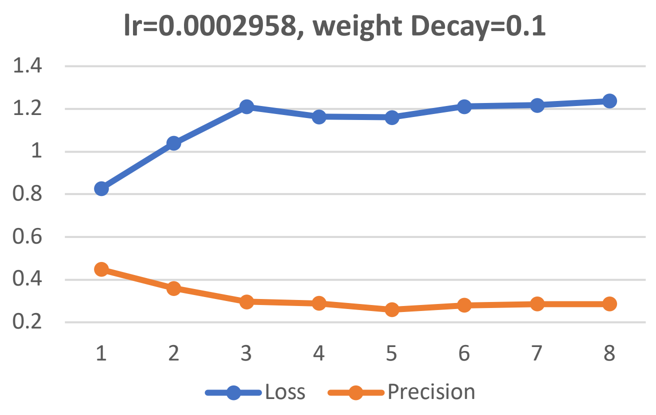

Figure 8 – Loss and precision graph of the worst model

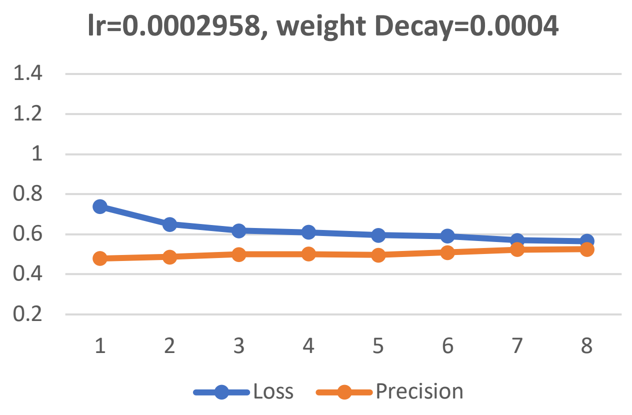

Figure 9 – Loss and precision graph of the best model

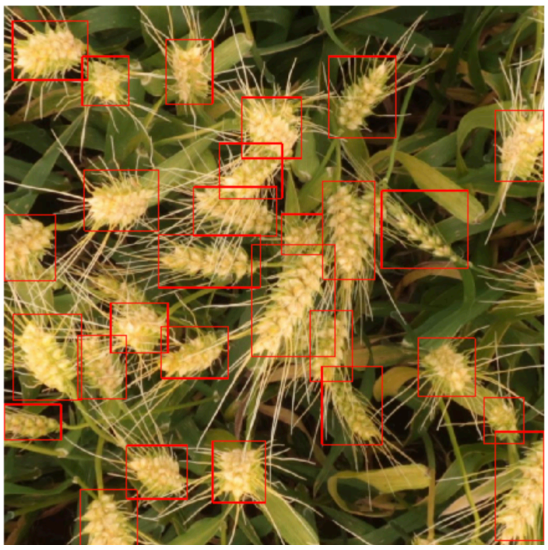
Figure 10 – The predicted label of one of the test images using our best model

Conclusion

With the help of PyTorch library and the high-quality wheat dataset, the team was able to build a wheat detector with good performance. Learning rate value from 0.0025 to 0.0035 works well in training, and a reasonable weight decay value (from 0.01 to 0.00001) does not affect the training much. Validation is useful during the training because we can estimate the performance of the model before submitting to the competition. Also, we can stop the training early if unexpected behaviours occurred. Other than that, our result shows that the model with noise augmentation is less likely to overfit. Finally, our best model can achieve a score of 0.565 on the private dataset and 0.6449 on the public dataset.

Future Tasks

Although our model has an acceptable score, other teams on Kaggle are managed to achieve a much more impressive score. PyTorch library provides various model and optimization algorithms. Thus, we could try different things to get a better model:

1. Use different optimizers such as Adam.
2. Use other hyperparameter values, for example, a larger number of epochs and a more intense noise augmentation.
3. Use a different model besides the Faster RCNN model used in the project. The team has noticed that many of the teams on Kaggle achieved a higher score with the Yolo3 model.
4. Implement early stopping to avoid overfitting.

Reference

[1]. Girshick R, Donahue J, Darrell T, Malik T. Rich feature hierarchies for accurate object detection and semantic segmentation. In: 2014 IEEE conference on computer vision and pattern recognition; 2014. p. 580–7

[2]. Girshick R. Fast R-CNN. In: 2015 IEEE international conference on computer vision (ICCV); 2015. p. 1440–48

[3]. Ren S, He K, Girshick R, Sun J. Faster R-CNN: towards real-time object detection with region proposal networks. IEEE Trans Pattern Anal Mach Intell. 2017;39(6):1137–49.

[4]. "Global Wheat Detection | Kaggle", Kaggle.com, 2020. [Online]. Available: https://www.kaggle.com/c/global-wheat-detection?rvi=1. [Accessed: 23- Oct- 2020].

[5]. "Pytorch Starter - FasterRCNN Train", Kaggle.com, 2020. [Online]. Available: https://www.kaggle.com/pestipeti/pytorch-starter-fasterrcnn-train. [Accessed: 23- Oct- 2020].

[6]. "Pytorch Starter - FasterRCNN Inference", Kaggle.com, 2020. [Online]. Available: https://www.kaggle.com/pestipeti/pytorch-starter-fasterrcnn-inference. [Accessed: 23- Oct- 2020].

[7]. "TorchVision Object Detection Finetuning Tutorial — PyTorch Tutorials 1.6.0 documentation", Pytorch.org, 2020. [Online]. Available: https://pytorch.org/tutorials/intermediate/torchvision_tutorial.html. [Accessed: 23- Oct- 2020].

[8]. "Google Colaboratory", Colab.research.google.com, 2020. [Online]. Available: https://colab.research.google.com/github/pytorch/vision/blob/temp-tutorial/tutorials/torchvision_finetuning_instance_segmentation.ipynb. [Accessed: 23- Oct- 2020].

[9]. "torchvision.models — PyTorch 1.6.0 documentation", Pytorch.org, 2020. [Online]. Available: https://pytorch.org/docs/stable/torchvision/models.html. [Accessed: 23- Oct- 2020].

[10]. "Why use a pre-trained model rather than creating your own?", Medium, 2020. [Online]. Available: https://medium.com/udacity-pytorch-challengers/why-use-a-pre-trained-model-rather-than-creating-your-own-d0e3a17e202f. [Accessed: 23- Oct- 2020].

[11]. "ResNet: A Simple Understanding of the Residual Networks", Medium, 2020. [Online]. Available: https://medium.com/swlh/resnet-a-simple-understanding-of-the-residual-networks-bfd8a1b4a447. [Accessed: 23- Oct- 2020].

[12]. "Setting the learning rate of your neural network.", Jeremy Jordan, 2020. [Online]. Available: https://www.jeremyjordan.me/nn-learning-rate/. [Accessed: 23- Oct- 2020].

[13]. "Papers with Code - Weight Decay Explained", Paperswithcode.com, 2020. [Online]. Available: https://paperswithcode.com/method/weight-decay. [Accessed: 23- Oct- 2020].