RECRUITMENT TASK REPORT

Flower species detection

Tomasz Hawro
Wrocław, Poland

09.05.2023

Contents

1 Decisions and steps

1. Create github repository

2. Investigate dataset

   - Check class imbalance to decide if there is a need to implement methods for imbalanced data - 80 images per class, so dataset is balanced
   - Check image shapes to decide how to implement transforms and the Dataset class - images have different shapes, so I decided to implement Dataset which loads images from the files in each call and applies Resize and Crop transforms
   - I couldnt find the class names of the flowers, so I hardcoded the labels using the "Class Examples" from the dataset website.

3. Choose tech stack - the tech stack that I used consists of *PyTorch*, *PyTorch Lightning*, *plotly*, *WandB*, *torchmetrics* and *gradio*:

   - *PyTorch* - neural networks architectures and datasets classes
   - *PyTorch Lightning* - model training and evaluation
   - *plotly* - visualizations
   - *WandB* - metrics, visualizations and model logging
   - *torchmetrics* - metrics calculation
   - *gradio* - application used to show how model works in real world

4. Implement model - since the dataset is quite small (1360 images in total), I decided to implement the simple deep convolutional neural network, that is 4 convolutional layers ended with global pool and linear layer. After each conv there are Batch Normalization, ReLU and MaxPool layers.

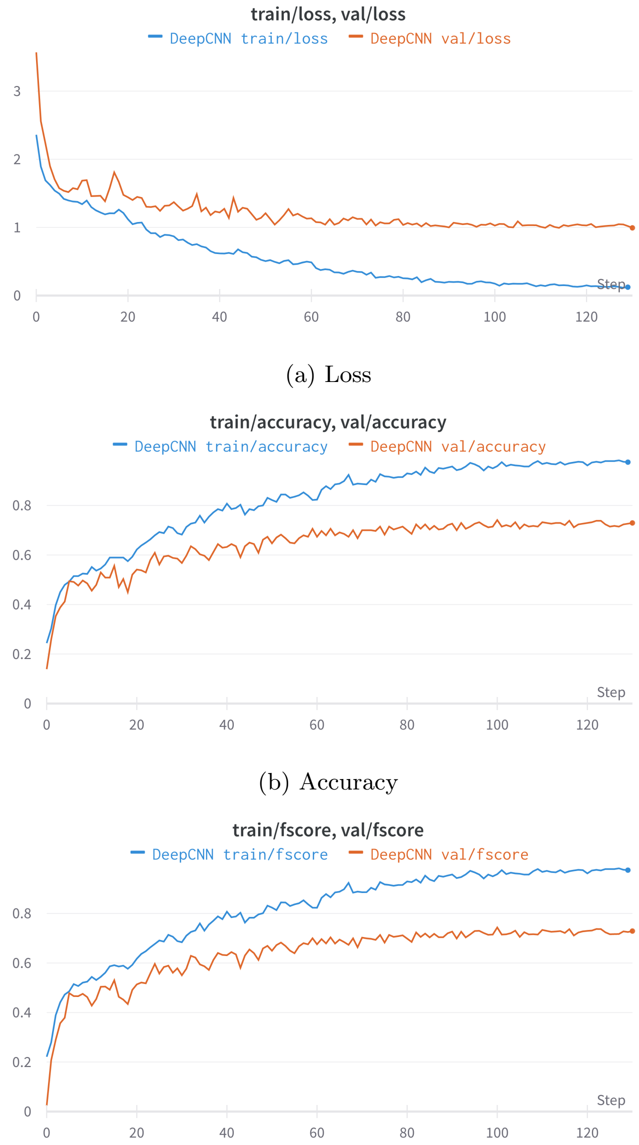
5. Implement training pipeline - the training pipeline is implemented with PyTorch Lightning, wandb and torchmetrics. For each run, the metrics, plots and model (saved with torchscript) are logged to wandb project

6. Add some visualizations - I chose to implement the confusion matrix and example predictions (with probabilities for each class) plots.

7. Implement gradio app for live presentation

2 Model performance

2.1 Model Training



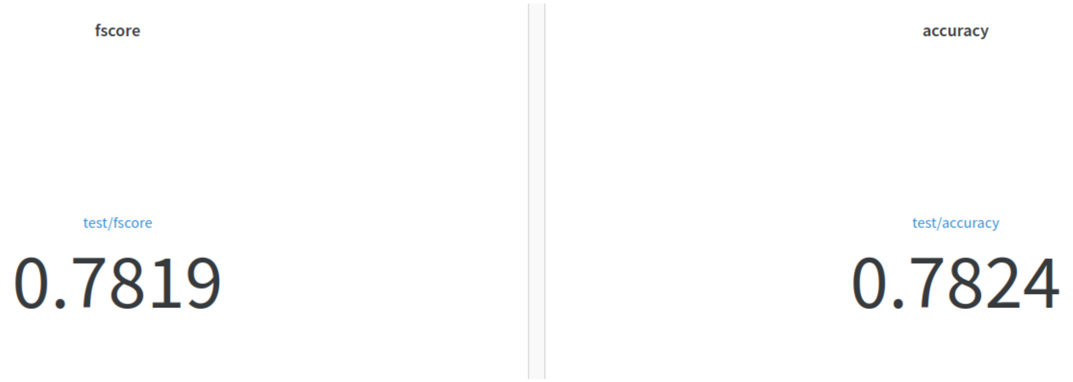
(a) Loss

(b) Accuracy

(c) F1-score

Figure 1: Model performance metrics

2.2 Model evaluation (on the test set)



fscore

test/fscore

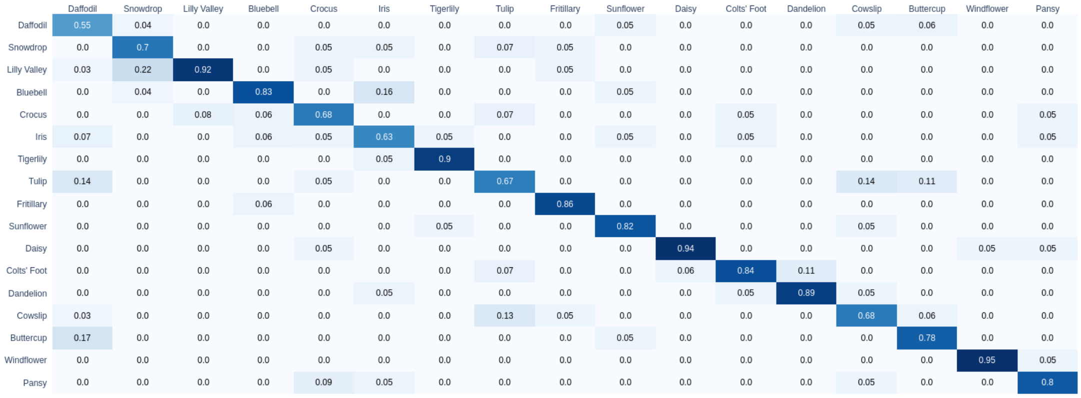
0.7819

accuracy

test/accuracy

0.7824

Figure 2: Test evaluation metrics



	Daffodil	Snowdrop	Lilly Valley	Bluebell	Crocus	Iris	Tigerlily	Tulip	Fritillary	Sunflower	Daisy	Colts' Foot	Dandelion	Cowslip	Buttercup	Windflower	Pansy
Daffodil	0.55	0.04	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.05	0.0	0.0	0.0	0.05	0.06	0.0	0.0
Snowdrop	0.0	0.7	0.0	0.0	0.05	0.05	0.0	0.07	0.05	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Lilly Valley	0.03	0.22	0.92	0.0	0.05	0.0	0.0	0.0	0.05	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Bluebell	0.0	0.04	0.0	0.83	0.0	0.16	0.0	0.0	0.0	0.05	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Crocus	0.0	0.0	0.08	0.06	0.68	0.0	0.0	0.07	0.0	0.0	0.0	0.05	0.0	0.0	0.0	0.0	0.05
Iris	0.07	0.0	0.0	0.06	0.05	0.63	0.05	0.0	0.0	0.05	0.0	0.05	0.0	0.0	0.0	0.0	0.05
Tigerlily	0.0	0.0	0.0	0.0	0.0	0.05	0.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Tulip	0.14	0.0	0.0	0.0	0.05	0.0	0.0	0.67	0.0	0.0	0.0	0.0	0.0	0.14	0.11	0.0	0.0
Fritillary	0.0	0.0	0.0	0.06	0.0	0.0	0.0	0.0	0.86	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Sunflower	0.0	0.0	0.0	0.0	0.0	0.0	0.05	0.0	0.0	0.82	0.0	0.0	0.0	0.05	0.0	0.0	0.0
Daisy	0.0	0.0	0.0	0.0	0.05	0.0	0.0	0.0	0.0	0.0	0.94	0.0	0.0	0.0	0.0	0.05	0.05
Colts' Foot	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.07	0.0	0.0	0.06	0.84	0.11	0.0	0.0	0.0	0.0
Dandelion	0.0	0.0	0.0	0.0	0.0	0.05	0.0	0.0	0.0	0.0	0.0	0.05	0.89	0.05	0.0	0.0	0.0
Cowslip	0.03	0.0	0.0	0.0	0.0	0.0	0.0	0.13	0.05	0.0	0.0	0.0	0.0	0.68	0.06	0.0	0.0
Buttercup	0.17	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.05	0.0	0.0	0.0	0.0	0.78	0.0	0.0
Windflower	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.95	0.05
Pansy	0.0	0.0	0.0	0.0	0.09	0.05	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.05	0.0	0.0	0.8

Figure 3: Test confusion matrix

The model performed well on the test (unseen) dataset (0.782 fscore), so I think it can work fine as a proof of concept (POC). As can be seen from the plots, there is a small overfitting, which could probably be improved by using external datasets and/or changing the model architecture, but the model was intended to work as a POC, so no further improvements were applied. Looking at the confusion matrix we can see that the major cases where the model predicted the wrong flower class were due to the high similarity between the species, for example dandelion and colt's foot, cowslip and tulip, buttercup and daffodil.

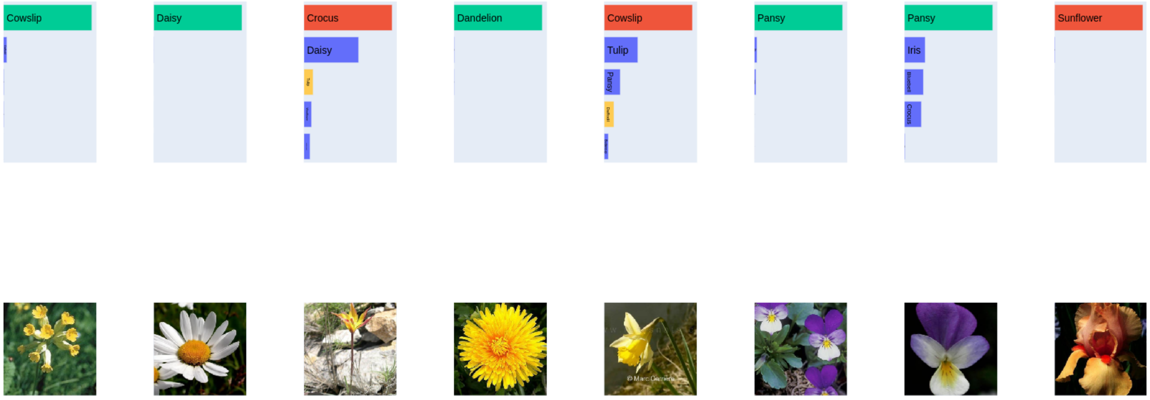


Figure 4: Test example predictions

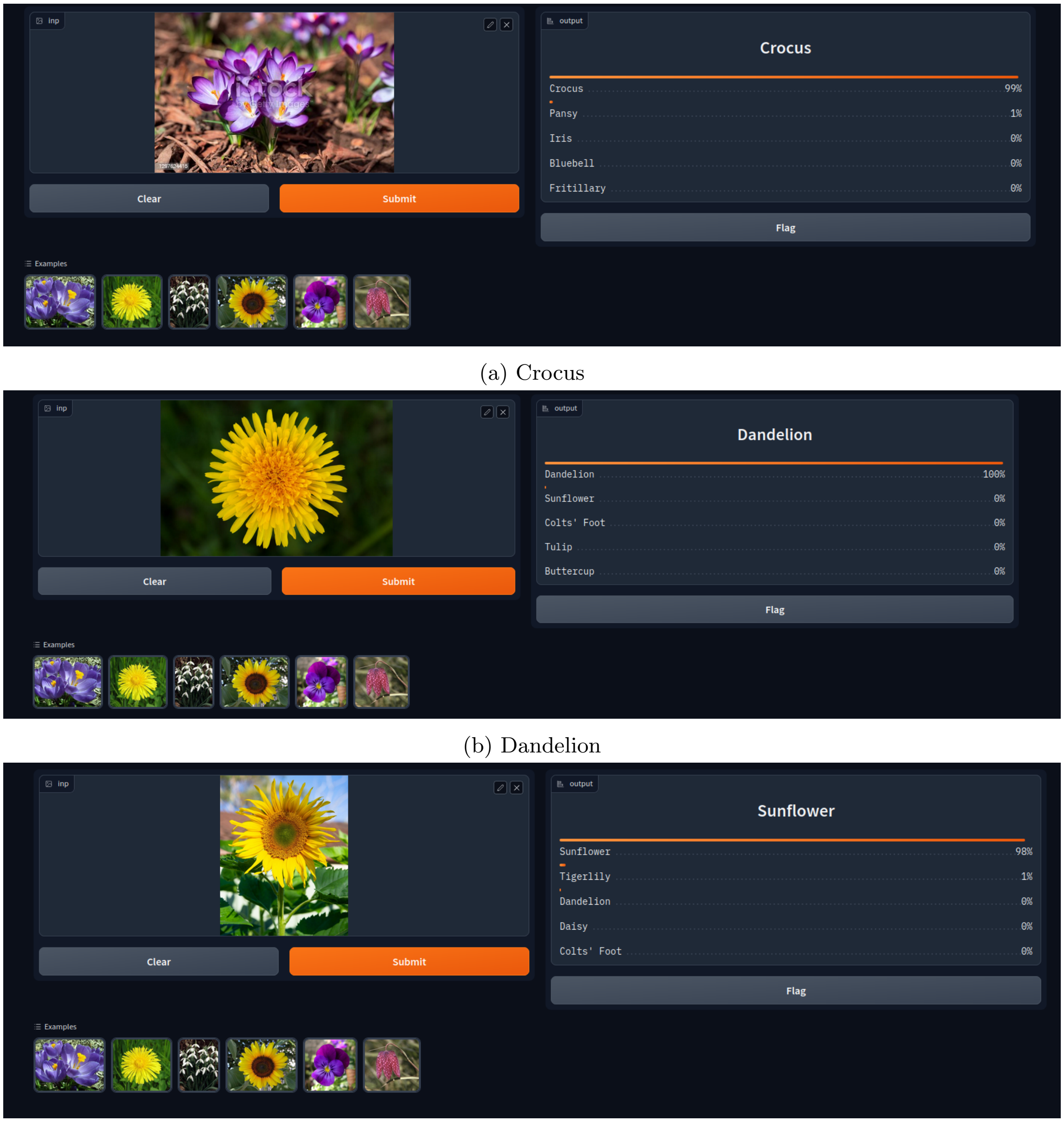
3 Questions

3.1 How would you share your findings with the client?

1. Show the results using the confusion matrix and explain the cases when model predicted wrong labels (similar flower species)

2. Use accuracy as the metric since the data was balanced and accuracy is pretty easy to understand

3. Show how the model works in real world using the gradio application. Gradio application loads trained model and allows to test the model on new, uploaded flower images:



Crocus

```
Crocus ........ 99%
Pansy ......... 1%
Iris .......... 0%
Bluebell ...... 0%
Fritillary .... 0%
```

(a) Crocus

Dandelion

```
Dandelion ..... 100%
Sunflower ..... 0%
Colts' Foot ... 0%
Tulip ......... 0%
Buttercup ..... 0%
```

(b) Dandelion

Sunflower

```
Sunflower ..... 98%
Tigerlily ..... 1%
Dandelion ..... 0%
Daisy ......... 0%
Colts' Foot ... 0%
```

(c) Sunflower

Figure 5: Gradio application

3.2 What would your comments be to a colleague building the app, regarding the model?

- The most important would be to ensure that the same transformations are applied to the images in the application
- Mention about using torchscript for inference

4 Time spent on task

- 2h for data gathering, PyTorch Lightning pipeline implementation and model training
- 4h for visualizations, wandb logging and gradio app
- 1h for report

5 Summary

All code is available in github repository and model training is logged in wandb run.