UNIVERISTY OF PIRAEUS - DEPARTMENT OF INFORMATICS

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ – ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

MSc «Informatics»

ΠΜΣ «Πληροφορική»

MSc Thesis
Μεταπτυχιακή Διατριβή

Thesis Title: Τίτλος Διατριβής:	**Deep Learning Methodologies for Plant Recognition** Μεθοδολογίες Βαθιάς Μάθησης για την Αναγνώριση Φυτών
Student's name-surname: Ονοματεπώνυμο φοιτητή:	**Lazaros Filippakos** Λάζαρος Φιλιππάκος
Father's name: Πατρώνυμο:	**Pelops** Πέλωψ
Student's ID No: Αριθμός Μητρώου:	**ΜΠΠΛ/19063**
Supervisor: Επιβλέπων:	**Dionisios Sotiropoulos, Assistant Professor** Διονύσιος Σωτηρόπουλος, Επίκουρος Καθηγητής

January 2021/ Ιανουάριος 2021

3-Member Examination Committee

Τριμελής Εξεταστική Επιτροπή

Dionisios Sotiropoulos **Assistant Professor** Διονύσιος Σωτηρόπουλος Επίκουρος Καθηγητής	**Georgios Tsihrintzis** **Professor** Γεώργιος Τσιχριντζής Καθηγητής	**Evangelos Sakkopoulos** **Assistant Professor** Ευάγγελος Σακκόπουλος Επίκουρος Καθηγητής

Credits

I would like to thank the University of Peiraeus for giving me the opportunity to take part in their Masters program of Informatics and the President of this department, Mrs Virvou, for accepting me. I would like to also thank all the lecturers, who taught me a lot about our discipline and what to look forward to but especially Dr Sotiropoulos Dionisios, who helped me with this paper and guided me through it. Last but not least, I would like to thank my fellow students who made a great team and went through a lot, especially these difficult times.

Abstract

Overpopulation and the modernization of the world has given the agriculture section a big hit and in a big need of an upgrade. It all started in the 90s with the first tractor but now AI and ML techniques are being used for the productivity and sustainability of the sector. Nowadays, it’s being used for seed classification purposes to help understanding the crops and seeds even for the person who will plant them in his own backyard.
This research will be similar to the study presented here [12] but it will use a different database containing multiple types of seeds which will have gone through a research and data augmentation. In this research a CNN model will be used for the classification.
These are the key points of this research:

- Conduct a seed classification by using a model to multiple variations of seeds instead of one kind of seed which was used in previous studies.
- Use CNN model which will be finetuned to optimize time and accuracy of the model.
- Show analytically the methods used.
- Make the model ready for use for other classification methods and or multiple devices.

Key Words: CNN, Machine learning, Deep Learning, Image Seed Classification, Plant Recognition

Περιεχόμενα

Index of Images

Index of Tables

Introduction

The primary sector of production has been and continues to be the backbone of the production process and at the same time the necessary precondition for the existence of the secondary and tertiary sectors. The need to provide goods directly from nature without any processing, treatment or transformation has never ceased to exist and therefore the primary sector is the main source of income for a large part of the world's population.

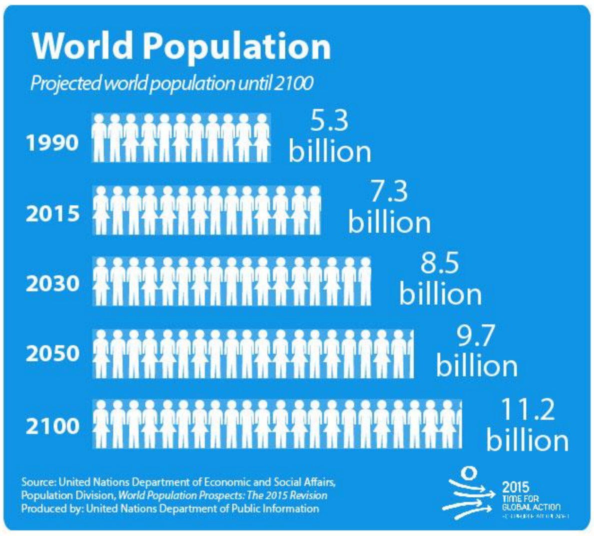


Image 1 The population growth the last 30 years and UN's prediction for the future in 2015

One of the main pillars of the primary sector is agriculture. It is considered to have played a key role in the growth of the population and its cultural evolution through the food surpluses it created. Variations in climate, culture and technologies used have been some of the main factors in the evolution of agriculture over the centuries. Whereas, for example, in the Neolithic period when it first appeared it was a way of life, it has now clearly taken on a commercialized character. The type of production and its quantities are based exclusively on market needs, while the simultaneous introduction of new technological methods and scientific knowledge into the production process have exponentially increased the quantities of each harvest.
However, the rapid growth of the world's population seems to be a barrier even for the already increasing global production. At the same time, a new problem seems to arise, that of the sustainability of agricultural production. Population growth has the direct consequence of reducing the available arable land, with the result that world production is beginning to decline year by year. More than a century has passed since the introduction of the first agricultural tractor (tractor) and the new technological methods adopted in the past seem to be inadequate. Thus, in recent years, innovative industrial technologies based on artificial intelligence and machine learning have started to be introduced in agriculture in order to improve productivity. At the same time, by controlling the environmental impact of any agricultural activity, food security and sustainability for the near and distant future can be achieved.
In addition to industrial agriculture, many people choose to grow their own grain or vegetables in their gardens. However, the separation of seeds and the

selection of the most ideal of them based on climatic conditions and not only requires knowledge that not everyone who decides to take up farming is able to have. This gap is filled by a fully automated system capable of identifying and classifying different types of seeds.

With the introduction of AI with its Machine Learning and Deep Learning capabilities more and more research has been constructed to find the best possible solution for a seed classification. Different types of seeds were used (e.g., weed seeds[2],cottonseeds[3], rice seeds [4][5], oat seeds[6], sunflower seeds[7], tomato seeds[8] and corn[9,10]) with different types of ML and DL techniques (DT[3][8], MLP[3][10], Naïve Bayes[3][8], ANN[8], KNN[4][8], LB[10], RF[10], BN[10], SVM[4], CNN[4]), getting good results and being used in areas of the global agricultural market.

Lastly, with studies comparing DL techniques and proving CNN is the best seed classifier when its image based [4], more and more studies are researching for the optimal setting and result using CNN [11][12].

This Thesis will follow the previous research and will try and optimize and improve some of the results. Goals of success will be as follow:

- Better accuracy and less loss in our results
- Use different type of seeds for a more complex seed classification
- Ease of use
- Friendly for implementation in a mobile environment
- Extra goal: Capable for use in other areas of the primary sector

Related Work

AI has helped the agriculture sector in many ways solving various problems while being costless, high performance driven and flexible. Image processing was introduced a decade ago [12] for seed and crop classification. Different ML and DL methods were used to extract features for object identification and classification.

Studies used cotton seeds for their machine learning models. It was and still is the most used crop in the field. They created their own dataset of pictures of cottons through their different stages of life and tried to use ML techniques for seed classification They used a decision tree classifier and MLP which had an accuracy of 98.7% and Naïve Bayes classifier which had an accuracy rate of 94.22%. They also took into account the time taken of their training models and the MLP took 1000x more than the other 2 models to fully be trained. Their conclusion was that the decision tree classifier would be the best ML model that would increase the cotton productivity and help with its seed classification because of its minimal time of building, minimal errors and high accuracy rate. [3]

Other studies used tomato seeds while extracting features because of their vigour and germination. They tried to extract features and parameters from the same picture and not by the position of the seeds or their area. Algorithms that were used were artificial Neural Networks (ANN) which had an precision rate of 95.44% and were also best accurate of 0.9722 with a time of 13 seconds of training, Naïve Bayes (NBC) of 87.89% and 0.9535, k-nearest neighbors (KNN) of 91,66% and 0.9307, decision tree (DT) of 93.66% and 0.9569 and support vector machines of 93.09% and 0.9498. Their proposed algorithm was ANN with an incorporation of MLP for a complete automated system for the classification of the germination of the tomatoes. [8]

One study studied corns for its seeds classification. Corn is used as much as cotton does and has multiple variations. They used different types of corn and tried to extract a hybrid set of features from it. In order to achieve their seed classification, they used ML techniques such as Multilayer perceptron (MLP), LogitBoost (LB), Random Forest (RF) and Bayes Net (BN). They also used a method called 10-Fold Cross Validation and their results were 98.83%, 97.78%, 97.22% and 96.67% for using MLP, LB, RF and BN respectively. For each of their experiments the results were similar with MLP topping the tables. Their results are solid because of their achievement of extracting hybrid features which would help each classifier equally. Their only comment was that their dataset should have included sunlight factors which disrupted some of their results. [10]

For Deep Learning methods, which are more complex and can extract the smallest details, studies have shown Convolutional Neural Networks outperforming kNN and SVM models. A study used rice to create its own seed classification. Their first discovery was that the more examples they used the better their classifiers performed. That’s because the more examples there are, the more different type of features combinations and feature values you can extract from those sets. Between the classifiers the KNN performed the worst when the number of samples increased, while SVM and CNN models, even

though they started to be similar, on higher sample numbers CNN outperformed the SVM model. CNN had an accuracy rate of 88.6% and 87.0% on their training and test set respectively. Their comment was that in the future more variety of rice crops should be used for their classification. [4]
Recently, a study created its own database of pictures and used CNN as their core Deep Learning technique for the classification. Its dataset concluded of 12 different seeds and not by a single type like before. They were very successful with an accuracy rate of 99.6% through the first 40 iterations. Their biggest drawback was the time it took the model to complete the training(67h).[12]
We will try to improve that model while making it easier to be used widely, ready for mobile use and less time consuming.

CNN Networks

Artificial neural networks are made up of neurons which are positioned in different layers. Each neural network contains an input layer, one or more hidden layers, and an output layer. The nodes are all connected to each other. Each connection has an associated weight and threshold. Convolutional Neural Networks are a subclass of these neural networks. They are a particular version of multilayer perceptrons, which are neural networks with multiple layers of neurons. Convolutional Neural Networks are used specifically for images and they allow the programmer to encode certain properties and assign importance to various characteristics for the images. These properties make the learning process more efficient as it helps in an easier differentiation between images as well as reduce the number of parameters of the network.

The architecture of a convolutional neural network is adapted compared to regular neural networks. Since the input of the neural network consists of images, the information inserted is in three dimensions since it contains the width of the image, the height of the image as well as information concerning the colors of the image. Thus, each layer of the network has neurons arranged in 3 dimensions (width, height, depth). Thus "Every layer transforms the" three-dimensional input volume to a" three-dimensional "output volume of neuron activations." There are three main types of layers in the architecture of such a neural network.

The Convolutional Layer…

The convolutional layer is the first layer of extraction which takes input features of the image by taking data from squares of pixels. Its taking the image matrix which gives the width, the height and the dimension (In our case the RGB numbers). Then it takes a filter matrix (also known as a **kernel**), which needs to be smaller than the actual image, which will go through the whole image. By rolling through the whole image, and multiplying the matrix's together, there will be a new matrix created, also called a **Feature Map**. There are 2 options to configure our kernel to move on our image:

Strides

Strides are the number of pixels the kernel will move through the image. If the stride is one the kernel moves one pixel at a time (Image 1). If its 2 pixel the kernel will jump 2 pixels and will create smaller output volumes.

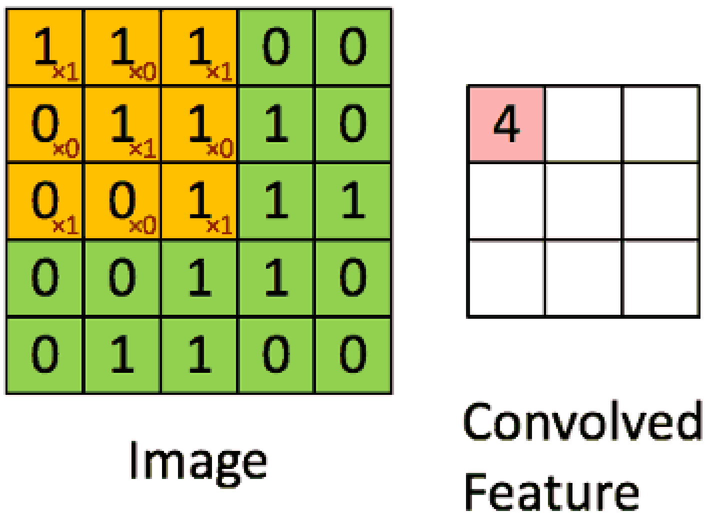


Image 2 CNN with a stride of 1

Padding

When the kernel doesn’t exactly fit on the image then we have the options:

- Add zeros to the picture until the kernel fits (**Zero Padding**).
- Or drop the part of the image where the kernel doesn’t fit (**Valid Padding**).

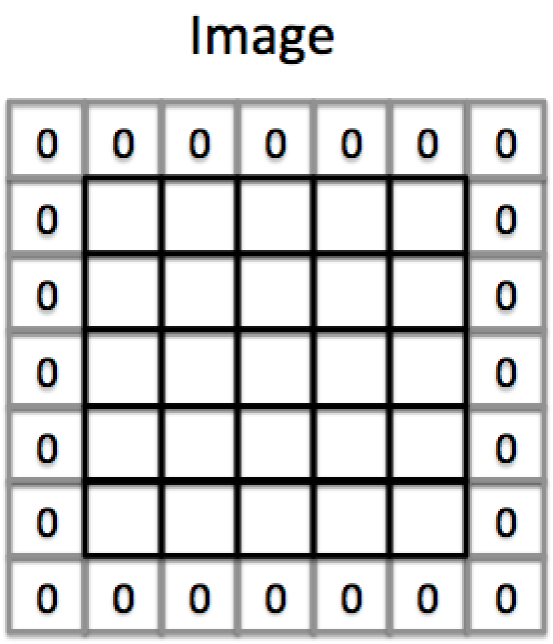


Image 3 How Padding Works (Zero Padding)

Pooling Layer

The Pooling Layer is periodically inserted in between successive convolutional layers. Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network, and hence to also control overfitting. There are 2 types of pooling:

- Max Pooling: While the filter is moving across it selects only the pixel with the biggest value to the output array. This is the most common used method

- Average Pooling: While the filter is being moved across, calculation are being done to find the average value inside the field, which will be sent to the output array.

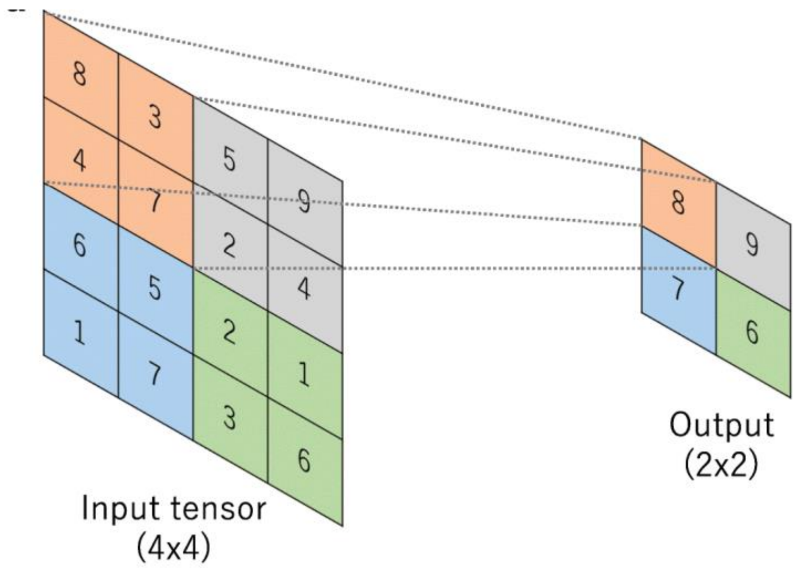


Image 4 Visual Application of Max Pooling

Fully Connected Layer

Lastly, the network also has a Fully-Connected Layer where neurons have full connections to all activations in the previous layer, which is the same as the case of regular neural networks. Its work is to extract the features from previous layers and classify them. Also, Convolutional and Pooling layers use the ReLu function, FC layers use the softmax function to classify the inputs.

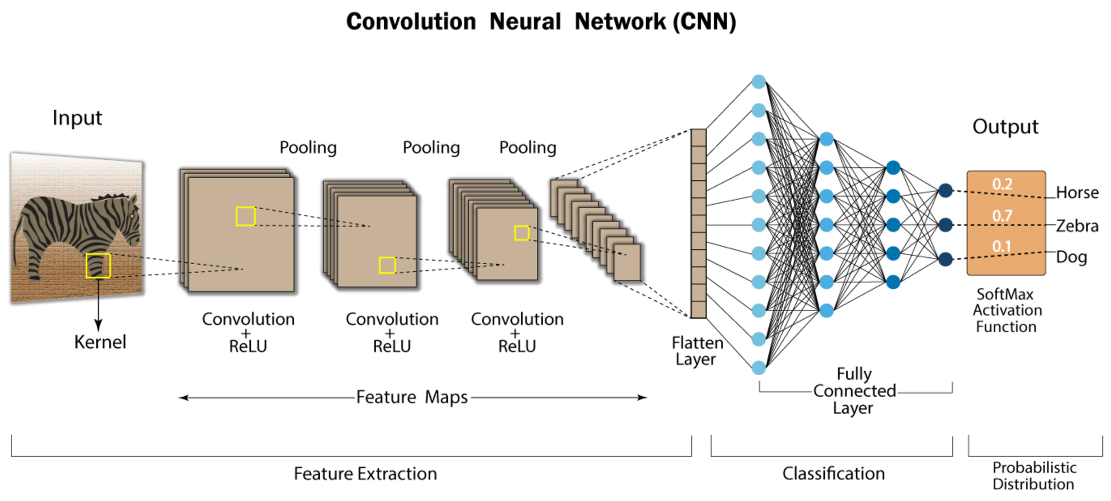


Image 5 Full Visualization of a Convolutional Neural Network

Formulas

Convolutional Layer

The convolutional layer accepts an input of size:

$$W_1 \times H_1 \times D1 \ (2.6.1)$$

It will need four parameters:

- Number of Filters **K**
- Spatial extent **F**
- Stride **S**
- Zero Padding **P**.

And it will produce an output of:

$$W_2 \times H_2 \times D_2 (2.6.2)$$

Where:

- $W_2 = (W_1 - F + 2P)/S + 1 \ (2.6.3)$
- $H_2 = (H_1 - F + 2P)/S + 1 \ (2.6.4)$
- $D_2 = K \ (2.6.5)$

Parameter Sharing

In order to control the number of parameters being used through the training of the CNN there is a scheme that's called parameter sharing. That's happening because the number of parameters would increase dramatically and make our NN way slower and insufficient to calculate our weights. For example, if we had an output of 55x55x96 that are connected to a 11x11x3 input that would give us:

- 55x55x96=290,400 neurons
- 11x11x3=363 weights +1 bias
- 290400*364=105,705,600 parameters only in the first layer of CNN

Clearly that number is way too high. By using a method called depth slice, we could constraint the neurons to have the weights and bias. For our example above, if we used 96 slices of size 55x55 we would have 96 unique set of weights. That would give us 96*11*11*3 = 34.848 Unique weights + 96 biases for a total of 34.944 parameters. (approx. 3000x smaller amount).

Why use a CNN instead of other Deep Learning Algorithms?

The Convolution network is much superior to other Feed Forward Nets because its less time consuming. The pre-processing is much less than before because of linear Algebra, specifically matrix multiplication, which is used to recognize and analyze patterns in an image. Its biggest drawback is that it may use a lot of computational power, primarily from the GPU but also from the CPU.

K-Fold Cross Validation

In this project we will be using a process called k-fold cross validation, which is used for better evaluation of the machine learning model used. In k-fold cross validation, the procedure followed involves shuffling the data randomly and splitting the dataset into groups. The number of groups used is defined by the parameter k. Then the following process is repeated k times. On every repetition a different group out of all the groups of data is kept as a test set. The neural network is then trained using the rest of the groups and then it is evaluated using the test set. Thus, each sample is used as a test set one time and as a part of the training set k-1 times.

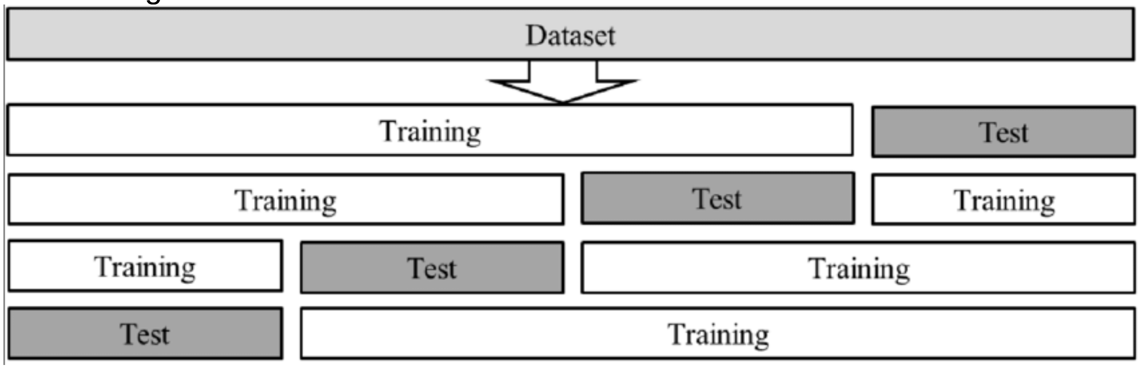


Image 6 Visualization of a 4-Fold Cross Validation

Dataset description

Our Dataset was taken from **this dataset**https://vision.eng.au.dk/plant-seedlings-dataset/ which had 12 folders of seeds containing different images of the same seed. In particular we used V2 for our experiment because of the accuracy of the results.

Table 1 Index of the Dataset

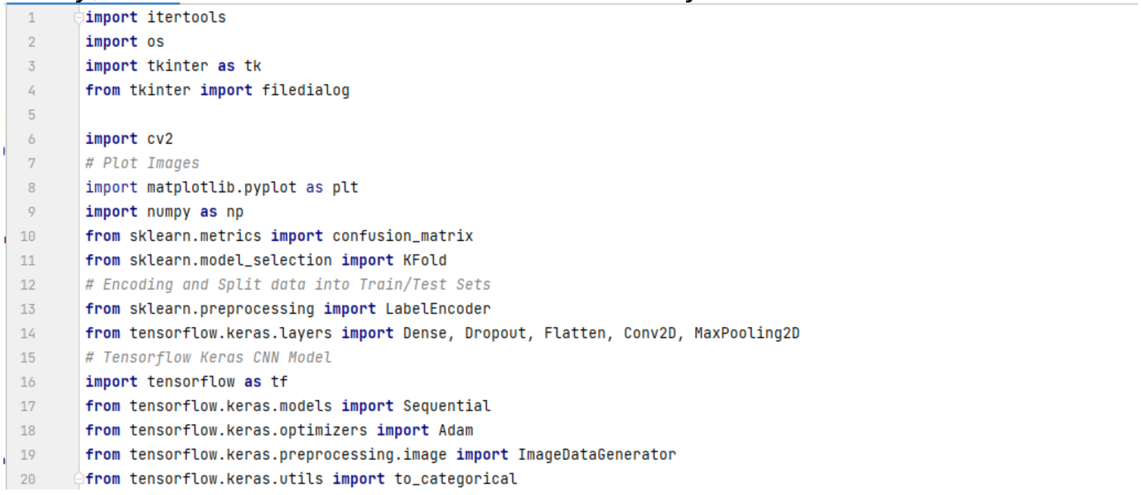
Seed Folder	Number of Images
Black Grass	309
Charlock	452
Cleavers	335
Common Chickweed	713
Common wheat	253
Fat Hen	538
Loose Silky-bent	762
Maize	257
Scentless Mayweed	607
Shepherd’s Purse	274
Small-flowered Cranesbill	576
Sugar beet	463
Total	5,539

Experiment

The experiment was done on a high end Desktop computer which included a AMD Ryzen 5600x, 32GB of Ram and an Nvidia RTX3070 GPU (latest at the time of the experiment) running on Windows 10 Professional. The Programming Language that was used is Python 3.8 and its Keras 2.4.3 library which was the best option for our experiment. We tried using PyTorch but we kept running into issues. Furthermore, the program was run in Jetbrains's App of PyCharm which the University of Pireaus kindly provided us a free license.

Code

Underneath is the code that was used to run our model. First will be shown all the libraries that got implemented. Some of them are natively inside Python's library, some needed to be downloaded manually.

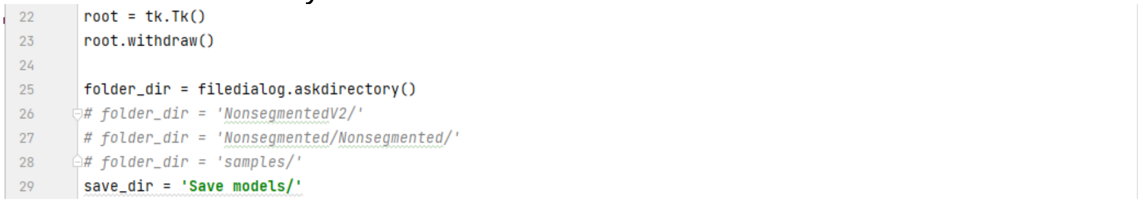


```python
import itertools
import os
import tkinter as tk
from tkinter import filedialog

import cv2
# Plot Images
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import KFold
# Encoding and Split data into Train/Test Sets
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D
# Tensorflow Keras CNN Model
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.utils import to_categorical
```

Image 7

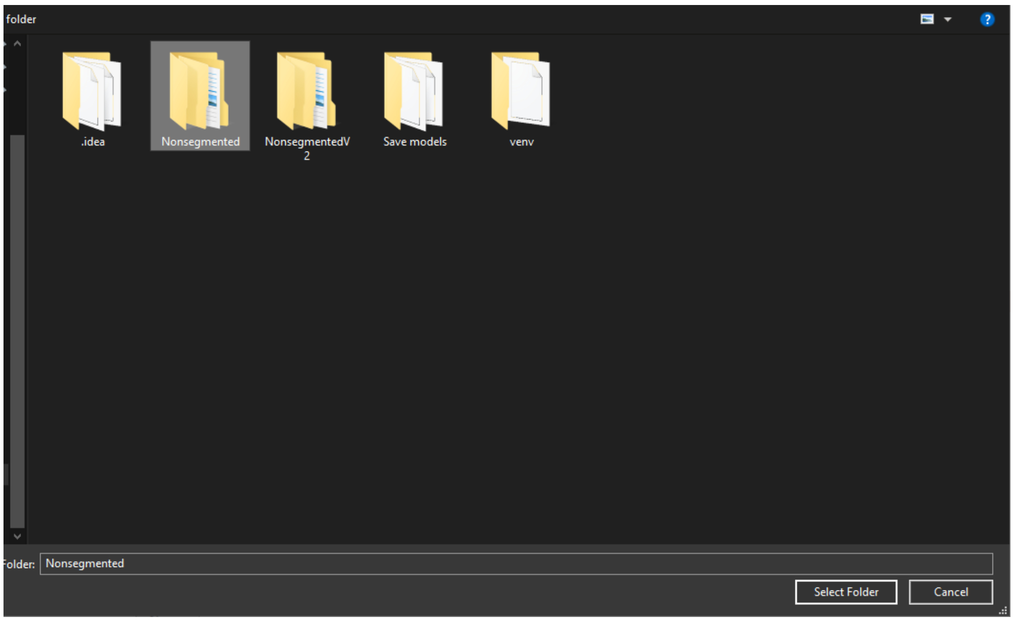
After that, we are starting by asking the user where to get our dataset from with tkinter and the filedialog expression. During the experiment it was already saved on the same folder as the program but using the before-mentioned method is a lot more User-Friendly.



```python
root = tk.Tk()
root.withdraw()

folder_dir = filedialog.askdirectory()
# folder_dir = 'NonsegmentedV2/'
# folder_dir = 'Nonsegmented/Nonsegmented/'
# folder_dir = 'samples/'
save_dir = 'Save models/'
```

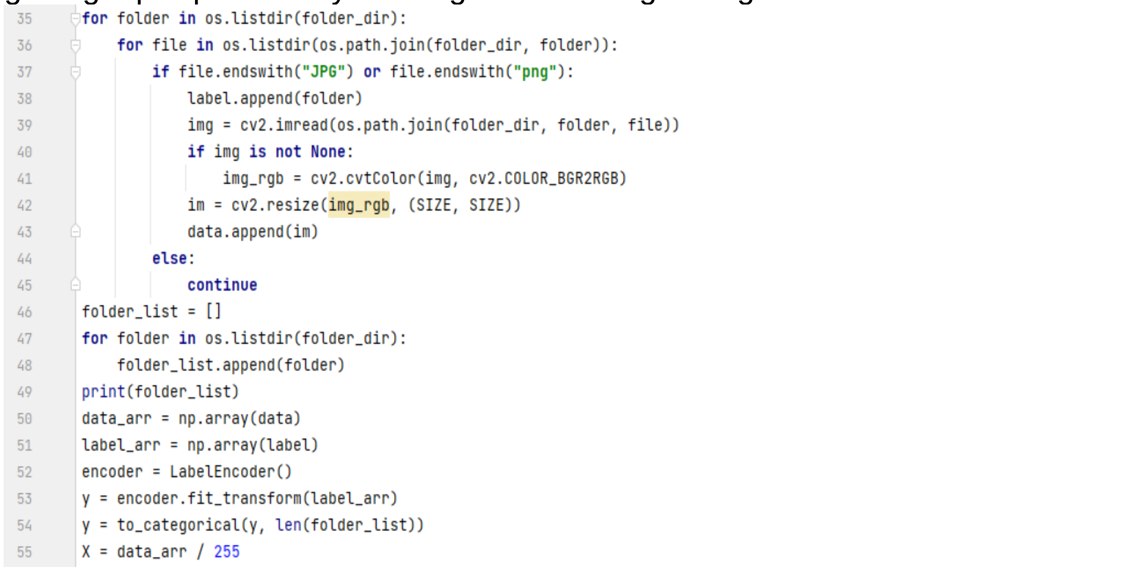
Image 8



Folder: Nonsegmented

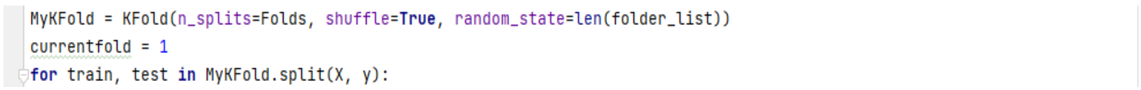
Image 9

After we have implemented our dataset the program will start searching inside the folder and the name of its subfolders. Doing that its saving the names of the classes (and plants) which will be used to be trained. While its searching through the folders and saving its picture by adding it to the array, every one of them is getting a pre-process by resizing it and having the right colors.



```python
for folder in os.listdir(folder_dir):
    for file in os.listdir(os.path.join(folder_dir, folder)):
        if file.endswith("JPG") or file.endswith("png"):
            label.append(folder)
            img = cv2.imread(os.path.join(folder_dir, folder, file))
            if img is not None:
                img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
            im = cv2.resize(img_rgb, (SIZE, SIZE))
            data.append(im)
        else:
            continue
folder_list = []
for folder in os.listdir(folder_dir):
    folder_list.append(folder)
print(folder_list)
data_arr = np.array(data)
label_arr = np.array(label)
encoder = LabelEncoder()
y = encoder.fit_transform(label_arr)
y = to_categorical(y, len(folder_list))
X = data_arr / 255
```

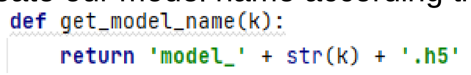
Image 10



```python
MyKFold = KFold(n_splits=Folds, shuffle=True, random_state=len(folder_list))
currentfold = 1
for train, test in MyKFold.split(X, y):
```

Image 11

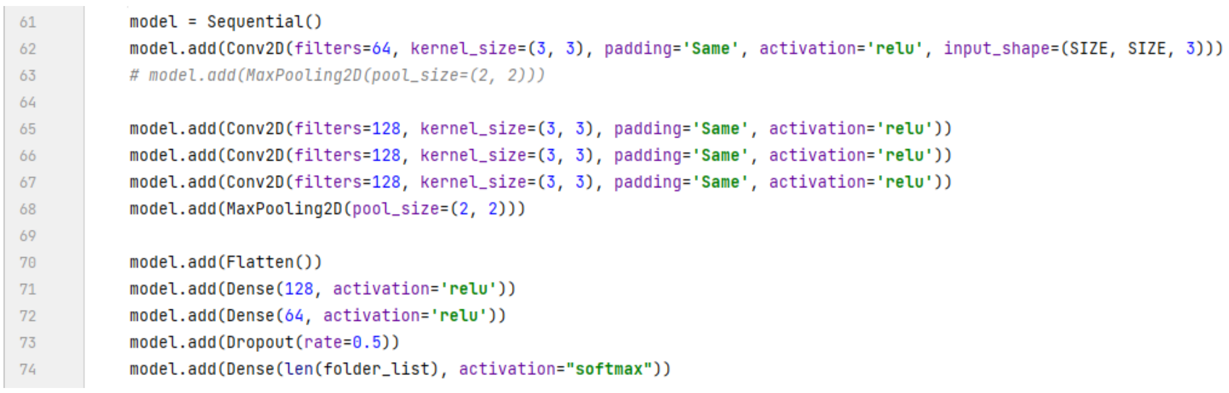
In order to save our models for further use we will just call a definition which will create our model name according the Fold we are in.



```python
def get_model_name(k):
    return 'model_' + str(k) + '.h5'
```

Image 12

We will continue with creating our model by creating our fully connected layers, our kernels and pool layers. We used the default kernel size of 3,3, multiple layers to make our model as accurate as possible and a pool size 2,2. In order to get our 1-D array of numbers down to the amount of Classes the Seeds that are used, we used the Flatten function and the activation functions of ReLu and softmax.

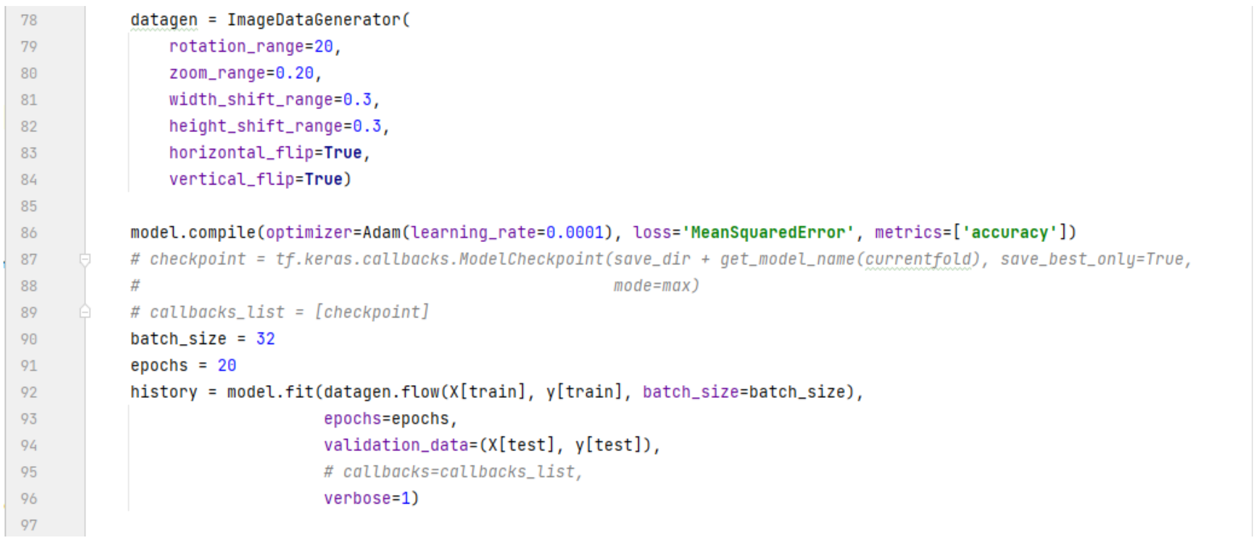


```python
    model = Sequential()
    model.add(Conv2D(filters=64, kernel_size=(3, 3), padding='Same', activation='relu', input_shape=(SIZE, SIZE, 3)))
    # model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Conv2D(filters=128, kernel_size=(3, 3), padding='Same', activation='relu'))
    model.add(Conv2D(filters=128, kernel_size=(3, 3), padding='Same', activation='relu'))
    model.add(Conv2D(filters=128, kernel_size=(3, 3), padding='Same', activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dense(64, activation='relu'))
    model.add(Dropout(rate=0.5))
    model.add(Dense(len(folder_list), activation="softmax"))
```

Image 13

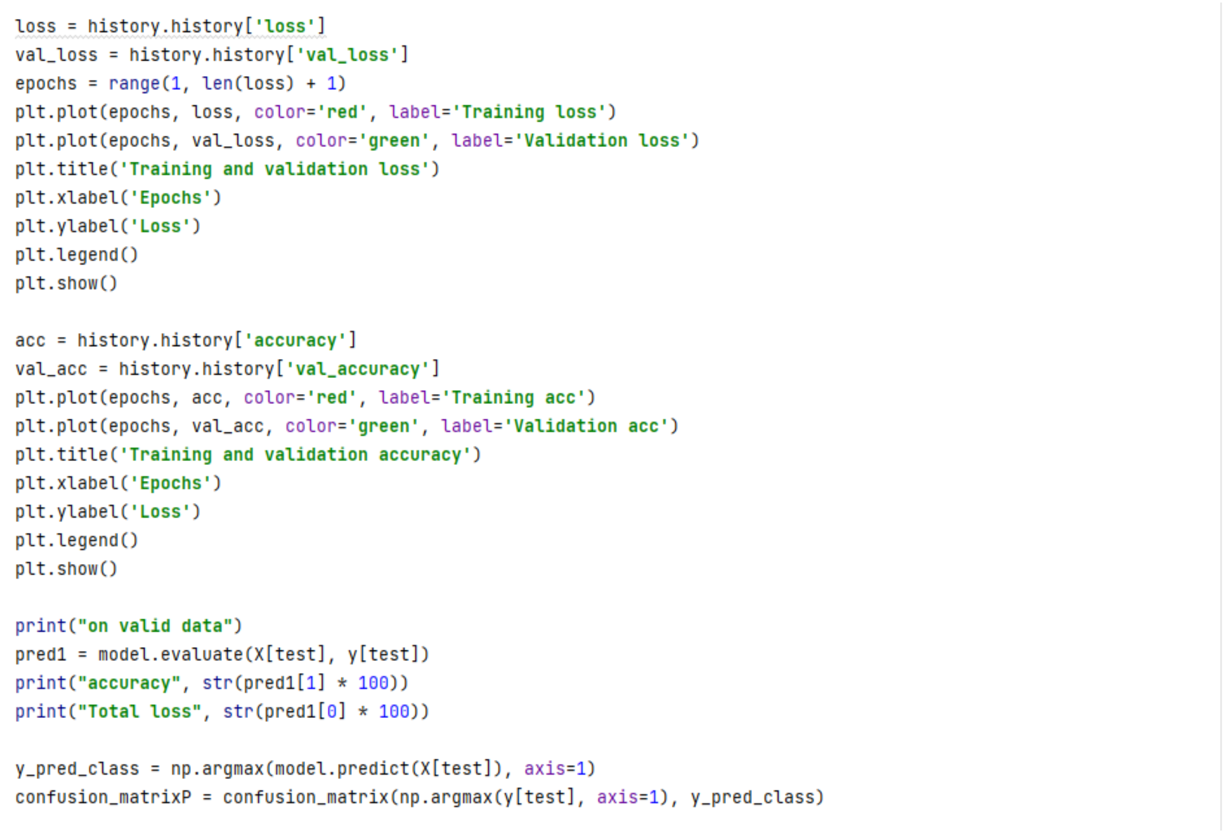


```python
    datagen = ImageDataGenerator(
        rotation_range=20,
        zoom_range=0.20,
        width_shift_range=0.3,
        height_shift_range=0.3,
        horizontal_flip=True,
        vertical_flip=True)

    model.compile(optimizer=Adam(learning_rate=0.0001), loss='MeanSquaredError', metrics=['accuracy'])
    # checkpoint = tf.keras.callbacks.ModelCheckpoint(save_dir + get_model_name(currentfold), save_best_only=True,
    #                                                 mode=max)
    # callbacks_list = [checkpoint]
    batch_size = 32
    epochs = 20
    history = model.fit(datagen.flow(X[train], y[train], batch_size=batch_size),
                        epochs=epochs,
                        validation_data=(X[test], y[test]),
                        # callbacks=callbacks_list,
                        verbose=1)
```

Image 14

After trying numerous ImageDataGenerators found online we used the most accurate one which also had the most basic features. Because we are using that image generator and succeeding an image augmentation, we are avoiding having an overfitment. Other features of the code will be talked in the Results page with their results.



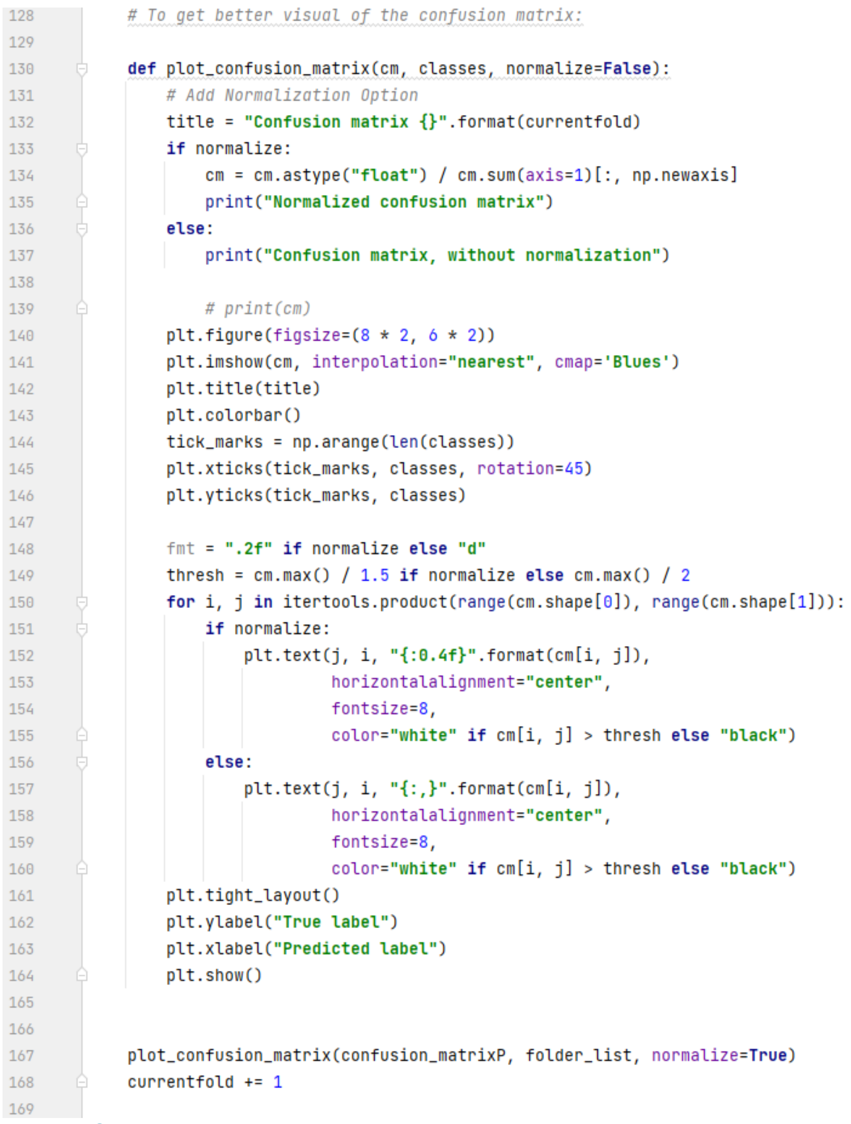
```
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(loss) + 1)
plt.plot(epochs, loss, color='red', label='Training loss')
plt.plot(epochs, val_loss, color='green', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
plt.plot(epochs, acc, color='red', label='Training acc')
plt.plot(epochs, val_acc, color='green', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

print("on valid data")
pred1 = model.evaluate(X[test], y[test])
print("accuracy", str(pred1[1] * 100))
print("Total loss", str(pred1[0] * 100))

y_pred_class = np.argmax(model.predict(X[test]), axis=1)
confusion_matrixP = confusion_matrix(np.argmax(y[test], axis=1), y_pred_class)
```

Image 15



```python
# To get better visual of the confusion matrix:

def plot_confusion_matrix(cm, classes, normalize=False):
    # Add Normalization Option
    title = "Confusion matrix {}".format(currentfold)
    if normalize:
        cm = cm.astype("float") / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print("Confusion matrix, without normalization")

        # print(cm)
    plt.figure(figsize=(8 * 2, 6 * 2))
    plt.imshow(cm, interpolation="nearest", cmap='Blues')
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    fmt = ".2f" if normalize else "d"
    thresh = cm.max() / 1.5 if normalize else cm.max() / 2
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        if normalize:
            plt.text(j, i, "{:0.4f}".format(cm[i, j]),
                     horizontalalignment="center",
                     fontsize=8,
                     color="white" if cm[i, j] > thresh else "black")
        else:
            plt.text(j, i, "{:,}".format(cm[i, j]),
                     horizontalalignment="center",
                     fontsize=8,
                     color="white" if cm[i, j] > thresh else "black")
    plt.tight_layout()
    plt.ylabel("True label")
    plt.xlabel("Predicted label")
    plt.show()


plot_confusion_matrix(confusion_matrixP, folder_list, normalize=True)
currentfold += 1
```

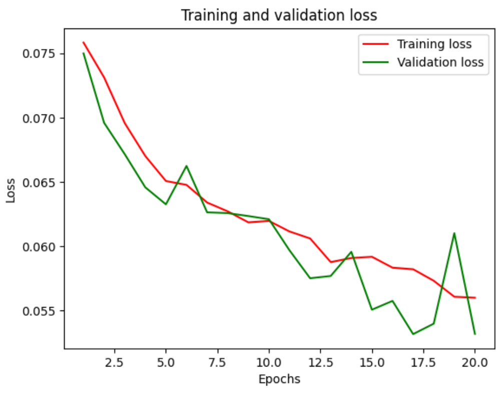
Image 16

Experimental Results

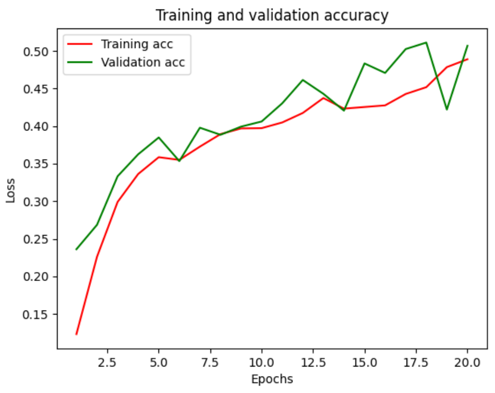
Our experiments goal was to try and hit our biggest score and hit our highest accuracy of our training model on the testing set. In order to achieve that we had to test a few variables, especially the most important ones. First and foremost we focused on the number of Folds that we could use to split our data and have our best result. And we also wanted to test the amount of epochs that we need to achieve our goals. The goal is to reach an accuracy of 85-90% with the minimum time needed to achieve that goal. We started with our Kfold Cross Validation because that's the most consuming element of our code as it starts a new training which each fold. Those were done with 20 epochs each.

Underneath you will find our results of our experiments. We focused on showing our accuracy, our loss and a confusion matrix heat map. While scrolling through them, watch carefully the lines of our accuracy and loss through each epoch that the neural network is getting trained at. Other hint of how good our neural network is performing will be the confusion matrix heat map which shows which seeds are having the correct output. The darker blue the diagonal line gets, the better.

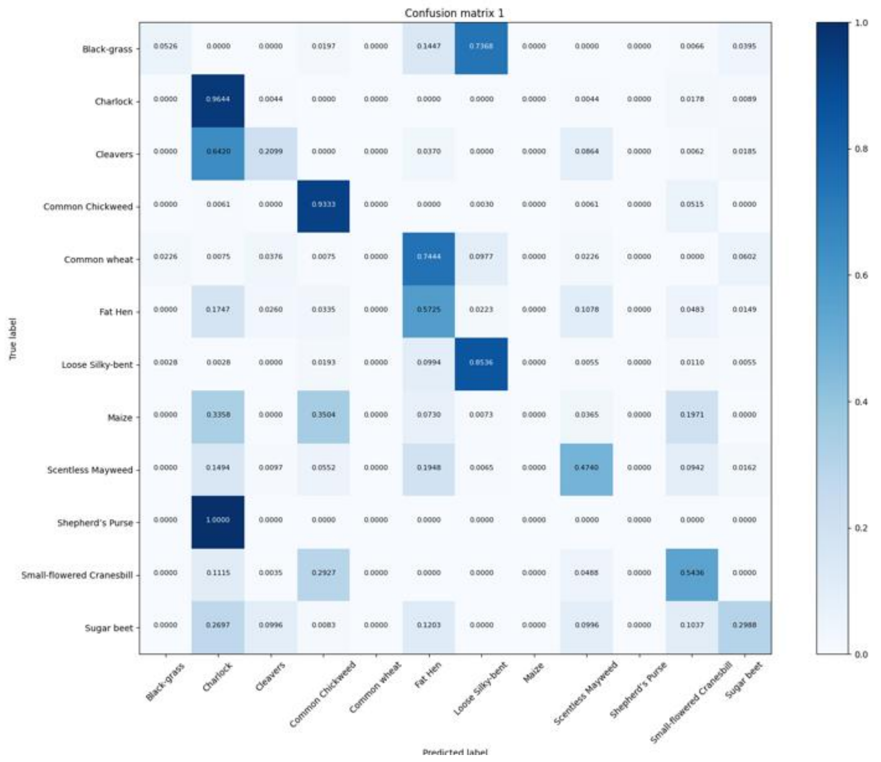
2Fold



2FoldTVL(Fold1)



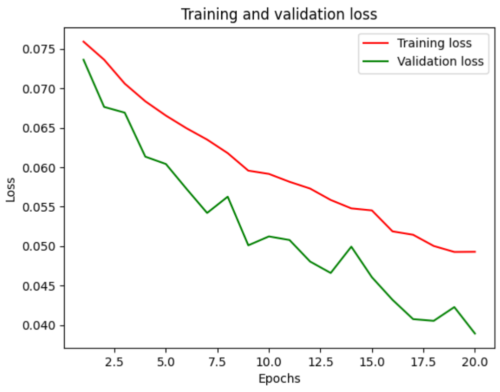
2FoldTVA(Fold1)



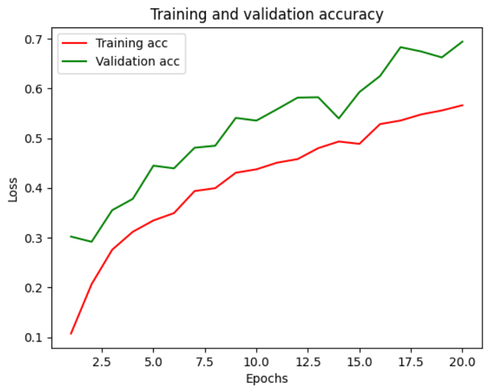
	Black-grass	Charlock	Cleavers	Common Chickweed	Common wheat	Fat Hen	Loose Silky-bent	Maize	Scentless Mayweed	Shepherd's Purse	Small-flowered Cranesbill	Sugar beet
Black-grass	0.0526	0.0000	0.0000	0.0197	0.0000	0.1447	0.7368	0.0000	0.0000	0.0000	0.0066	0.0395
Charlock	0.0000	0.9644	0.0044	0.0000	0.0000	0.0000	0.0000	0.0000	0.0044	0.0000	0.0178	0.0089
Cleavers	0.0000	0.6420	0.2099	0.0000	0.0000	0.0370	0.0000	0.0000	0.0864	0.0000	0.0062	0.0185
Common Chickweed	0.0000	0.0061	0.0000	0.9333	0.0000	0.0000	0.0030	0.0000	0.0061	0.0000	0.0515	0.0000
Common wheat	0.0226	0.0075	0.0376	0.0075	0.0000	0.7444	0.0977	0.0000	0.0226	0.0000	0.0000	0.0602
Fat Hen	0.0000	0.1747	0.0260	0.0335	0.0000	0.5725	0.0223	0.0000	0.1078	0.0000	0.0483	0.0149
Loose Silky-bent	0.0028	0.0028	0.0000	0.0193	0.0000	0.0994	0.8536	0.0000	0.0055	0.0000	0.0110	0.0055
Maize	0.0000	0.3358	0.0000	0.3504	0.0000	0.0730	0.0073	0.0000	0.0365	0.0000	0.1971	0.0000
Scentless Mayweed	0.0000	0.1494	0.0097	0.0552	0.0000	0.1948	0.0065	0.0000	0.4740	0.0000	0.0942	0.0162
Shepherd's Purse	0.0000	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Small-flowered Cranesbill	0.0000	0.1115	0.0035	0.2927	0.0000	0.0000	0.0000	0.0000	0.0488	0.0000	0.5436	0.0000
Sugar beet	0.0000	0.2697	0.0996	0.0083	0.0000	0.1203	0.0000	0.0000	0.0996	0.0000	0.1037	0.2988

2FoldCM(Fold1)

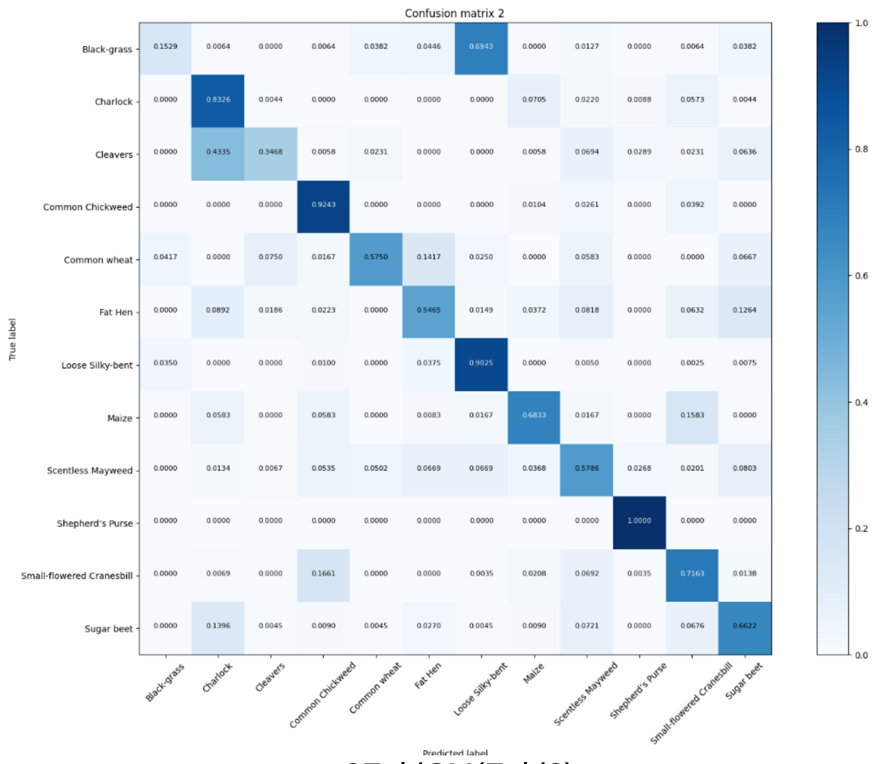
Image 17 - 2Fold Fold1



2FoldTVL(Fold2)



2FoldTVA(Fold2)

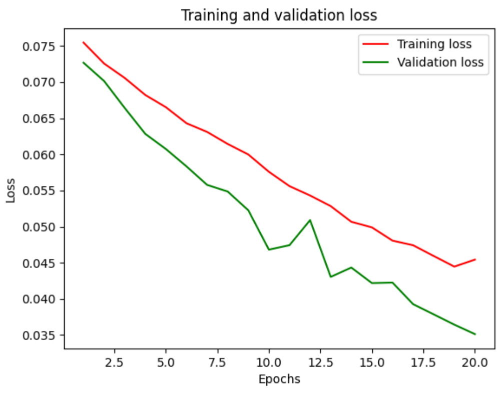


	Black-grass	Charlock	Cleavers	Common Chickweed	Common wheat	Fat Hen	Loose Silky-bent	Maize	Scentless Mayweed	Shepherd's Purse	Small-flowered Cranesbill	Sugar beet
Black-grass	0.1529	0.0064	0.0000	0.0064	0.0382	0.0446	0.6943	0.0000	0.0127	0.0000	0.0064	0.0382
Charlock	0.0000	0.8326	0.0044	0.0000	0.0000	0.0000	0.0000	0.0705	0.0220	0.0088	0.0573	0.0044
Cleavers	0.0000	0.4335	0.3468	0.0058	0.0231	0.0000	0.0000	0.0058	0.0694	0.0289	0.0231	0.0636
Common Chickweed	0.0000	0.0000	0.0000	0.9243	0.0000	0.0000	0.0000	0.0104	0.0261	0.0000	0.0392	0.0000
Common wheat	0.0417	0.0000	0.0750	0.0167	0.5750	0.1417	0.0250	0.0000	0.0583	0.0000	0.0000	0.0667
Fat Hen	0.0000	0.0892	0.0186	0.0223	0.0000	0.5465	0.0149	0.0372	0.0818	0.0000	0.0632	0.1264
Loose Silky-bent	0.0350	0.0000	0.0000	0.0100	0.0000	0.0375	0.9025	0.0000	0.0050	0.0000	0.0025	0.0075
Maize	0.0000	0.0583	0.0000	0.0583	0.0000	0.0083	0.0167	0.6833	0.0167	0.0000	0.1583	0.0000
Scentless Mayweed	0.0000	0.0134	0.0067	0.0535	0.0502	0.0669	0.0669	0.0368	0.5786	0.0268	0.0201	0.0803
Shepherd's Purse	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000
Small-flowered Cranesbill	0.0000	0.0069	0.0000	0.1661	0.0000	0.0000	0.0035	0.0208	0.0692	0.0035	0.7163	0.0138
Sugar beet	0.0000	0.1396	0.0045	0.0090	0.0045	0.0270	0.0045	0.0090	0.0721	0.0000	0.0676	0.6622

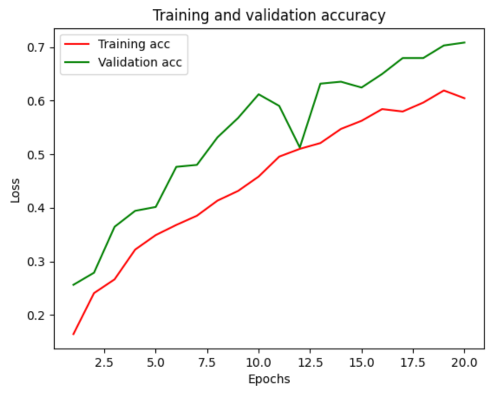
2FoldCM(Fold2)

Image 18 - 2Fold Fold2

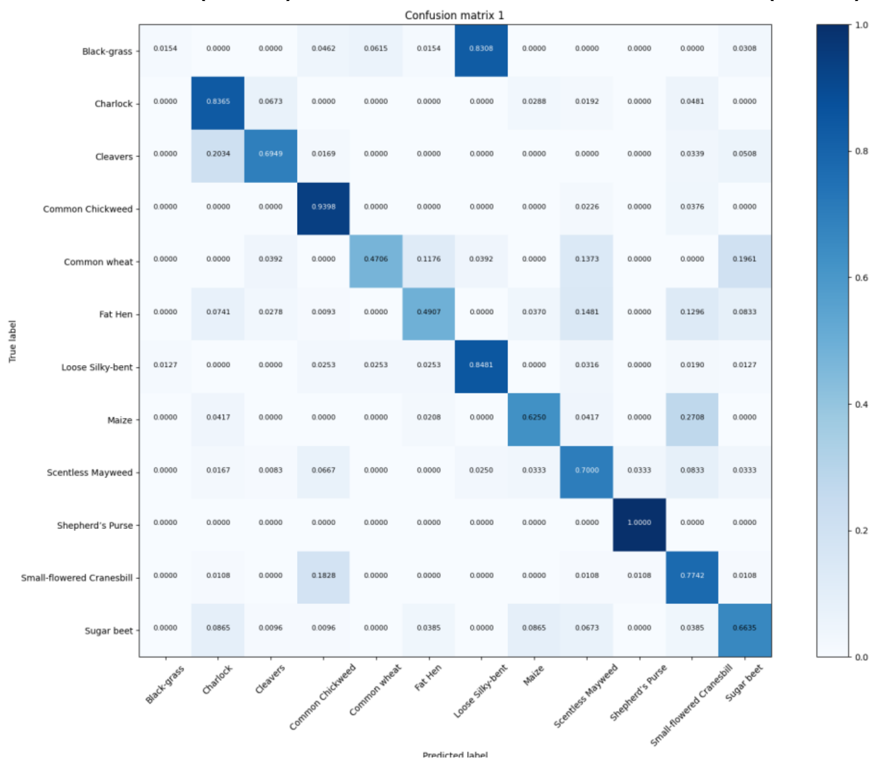
5Fold



5FoldTVL(Fold1)

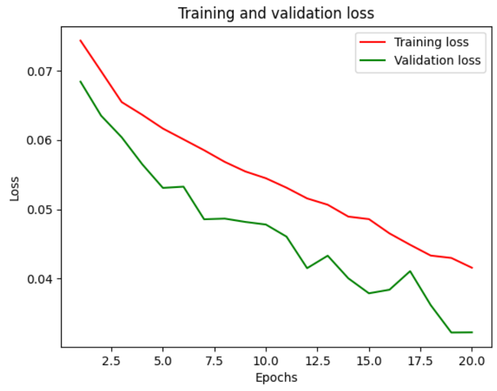


5FoldTVA(Fold1)

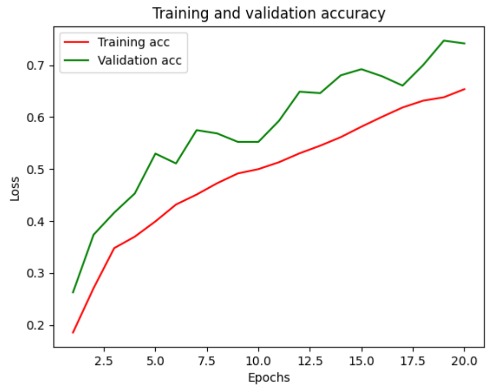


5FoldCM(Fold1)

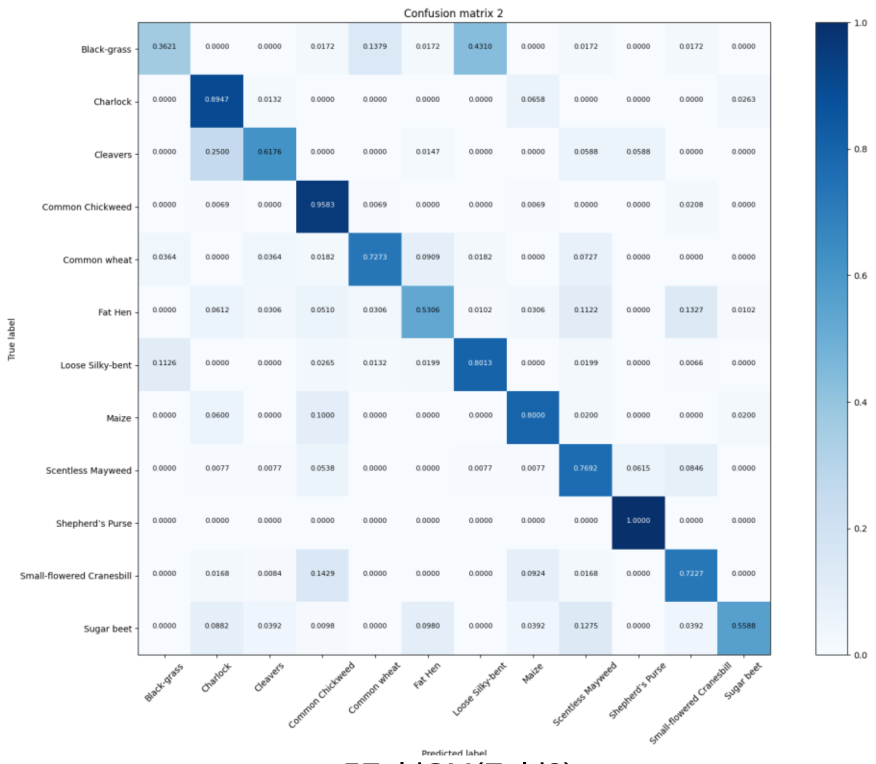
Image 19 - 5Folds Fold1



5FoldTVL(Fold2)

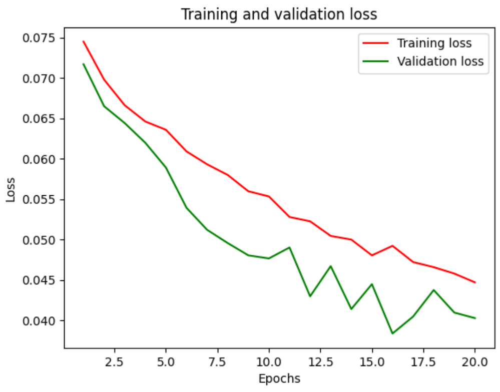


5FoldTVA(Fold2)

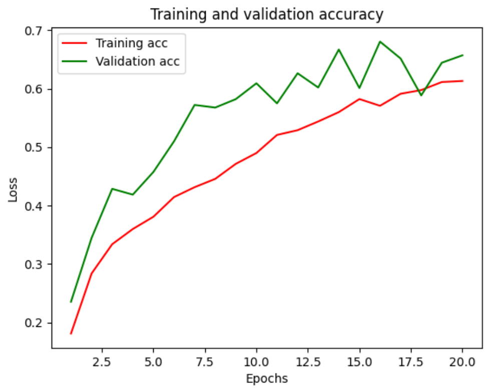


5FoldCM(Fold2)

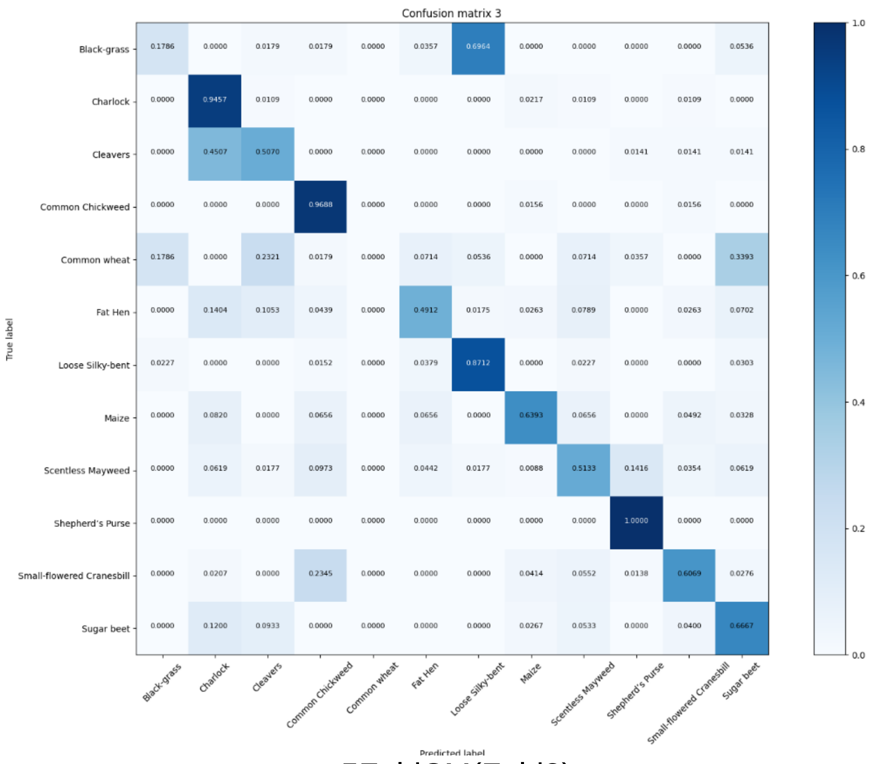
Image 20 - 5Folds Fold2



5FoldTVL(Fold3)



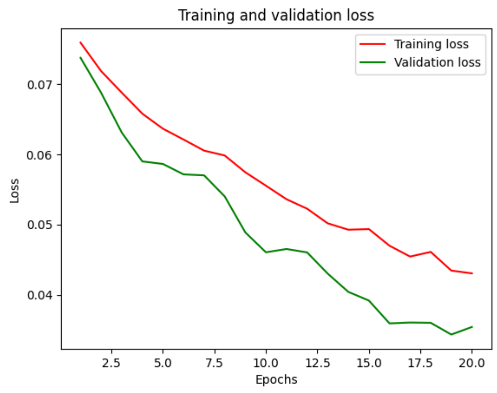
5FoldTVA(Fold3)



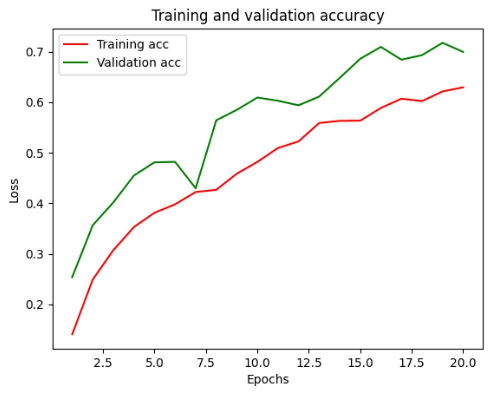
	Black-grass	Charlock	Cleavers	Common Chickweed	Common wheat	Fat Hen	Loose Silky-bent	Maize	Scentless Mayweed	Shepherd's Purse	Small-flowered Cranesbill	Sugar beet
Black-grass	0.1786	0.0000	0.0179	0.0179	0.0000	0.0357	0.6964	0.0000	0.0000	0.0000	0.0000	0.0536
Charlock	0.0000	0.9457	0.0109	0.0000	0.0000	0.0000	0.0000	0.0217	0.0109	0.0000	0.0109	0.0000
Cleavers	0.0000	0.4507	0.5070	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0141	0.0141	0.0141
Common Chickweed	0.0000	0.0000	0.0000	0.9688	0.0000	0.0000	0.0000	0.0156	0.0000	0.0000	0.0156	0.0000
Common wheat	0.1786	0.0000	0.2321	0.0179	0.0000	0.0714	0.0536	0.0000	0.0714	0.0357	0.0000	0.3393
Fat Hen	0.0000	0.1404	0.1053	0.0439	0.0000	0.4912	0.0175	0.0263	0.0789	0.0000	0.0263	0.0702
Loose Silky-bent	0.0227	0.0000	0.0000	0.0152	0.0000	0.0379	0.8712	0.0000	0.0227	0.0000	0.0000	0.0303
Maize	0.0000	0.0820	0.0000	0.0656	0.0000	0.0656	0.0000	0.6393	0.0656	0.0000	0.0492	0.0328
Scentless Mayweed	0.0000	0.0619	0.0177	0.0973	0.0000	0.0442	0.0177	0.0088	0.5133	0.1416	0.0354	0.0619
Shepherd's Purse	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000
Small-flowered Cranesbill	0.0000	0.0207	0.0000	0.2345	0.0000	0.0000	0.0000	0.0414	0.0552	0.0138	0.6069	0.0276
Sugar beet	0.0000	0.1200	0.0933	0.0000	0.0000	0.0000	0.0000	0.0267	0.0533	0.0000	0.0400	0.6667

5FoldCM(Fold3)

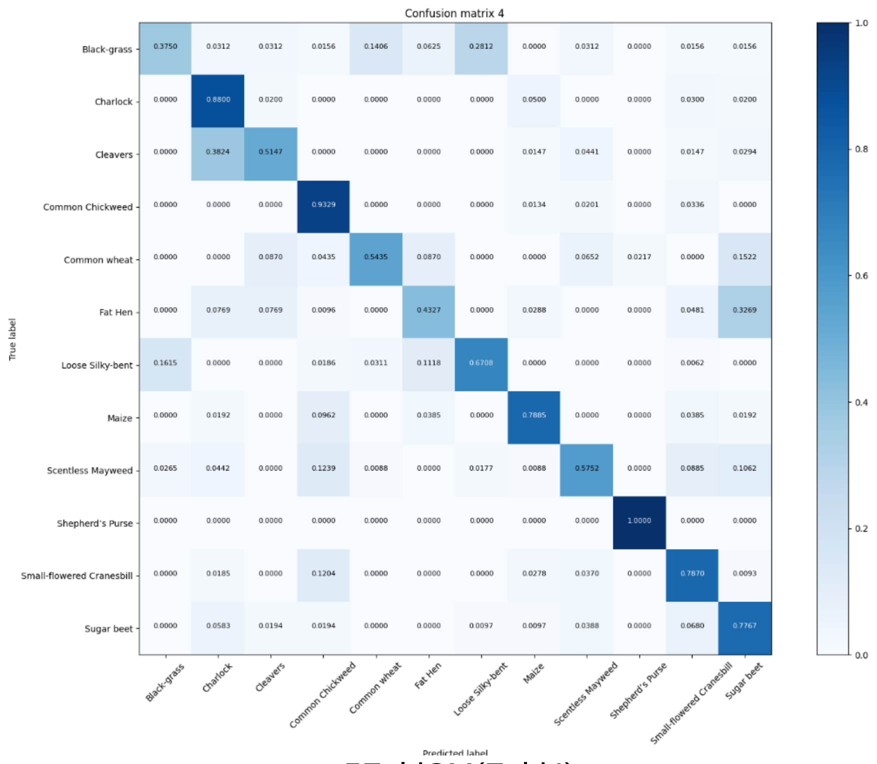
Image 21 - 5Folds Fold3



5FoldTVL(Fold4)

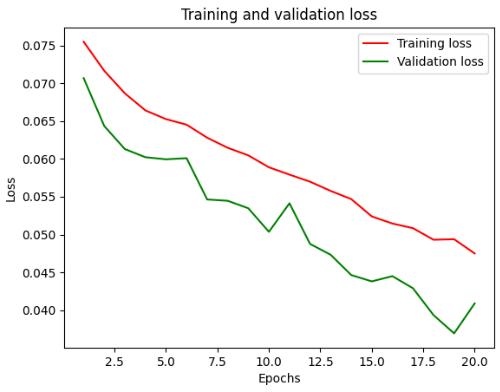


5FoldTVA(Fold4)

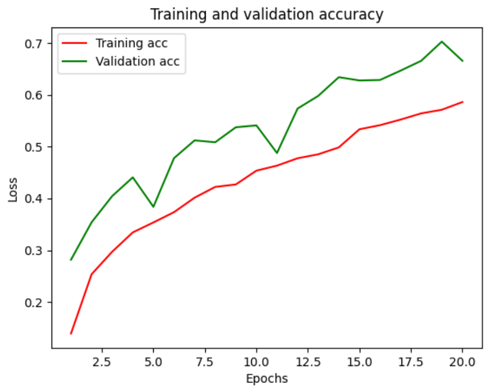


5FoldCM(Fold4)

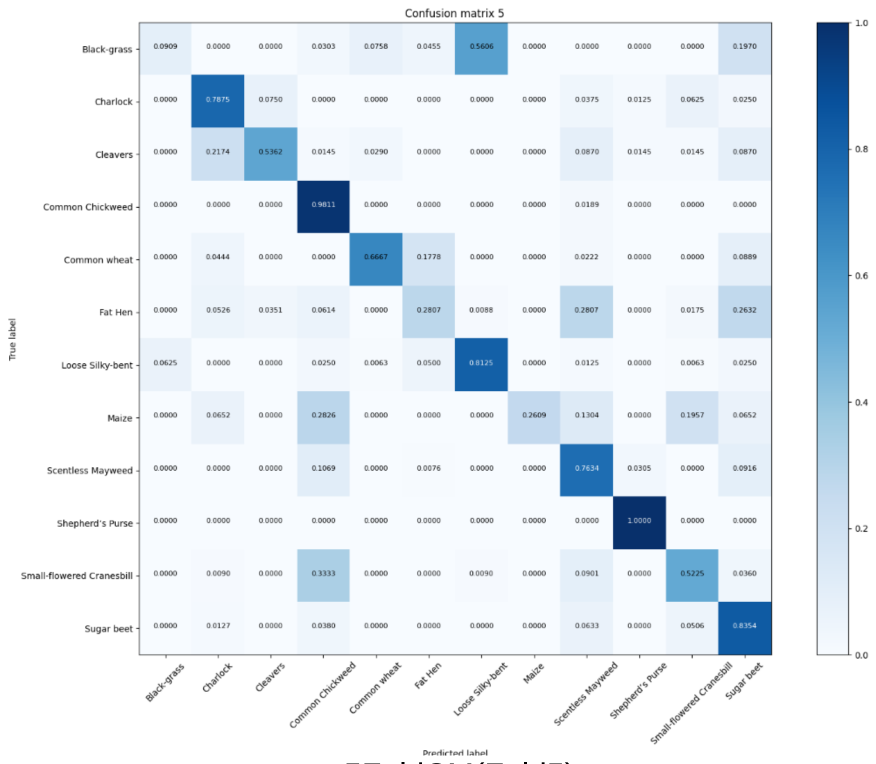
Image 22 - 5Folds Fold4



5FoldTVL(Fold5)



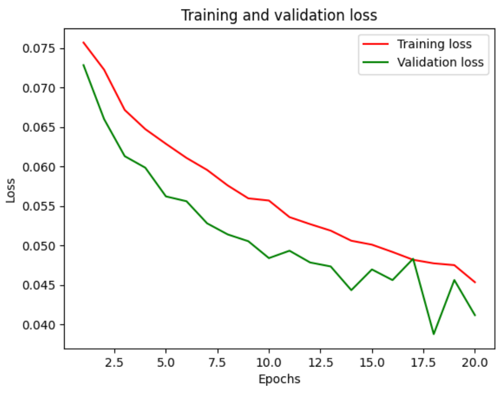
5FoldTVA(Fold5)



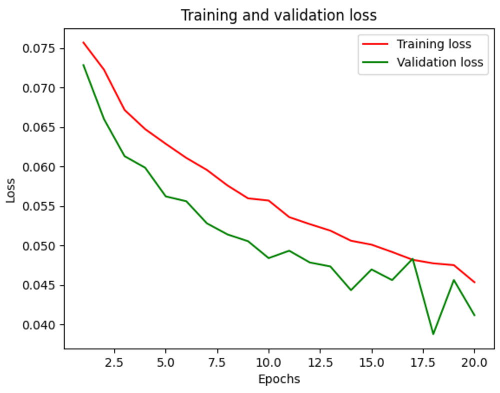
5FoldCM(Fold5)

Image 23 - 5Folds Fold5

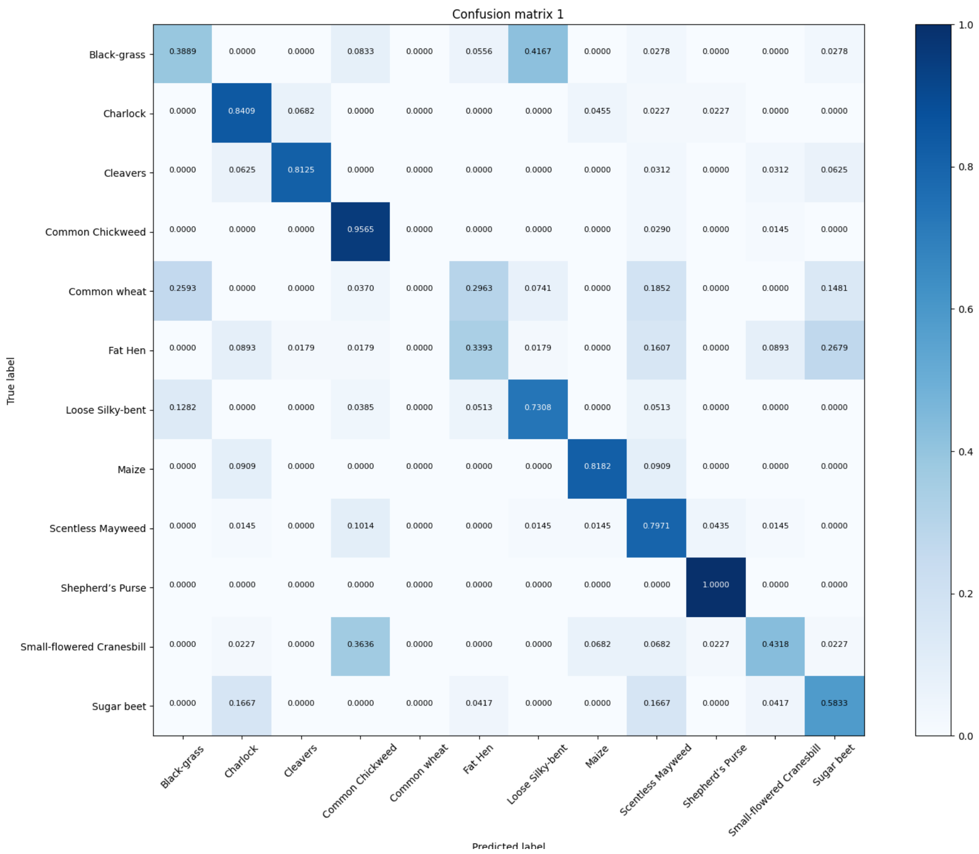
10Fold



10FoldTVL(Fold1)

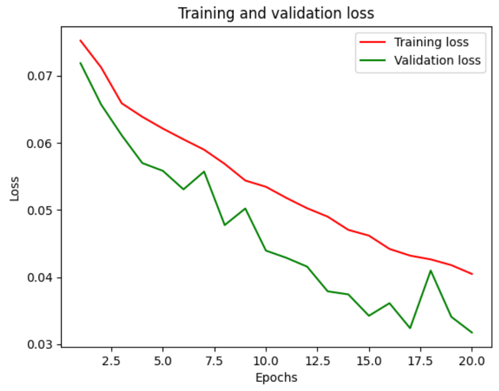


10FoldTVA(Fold1)

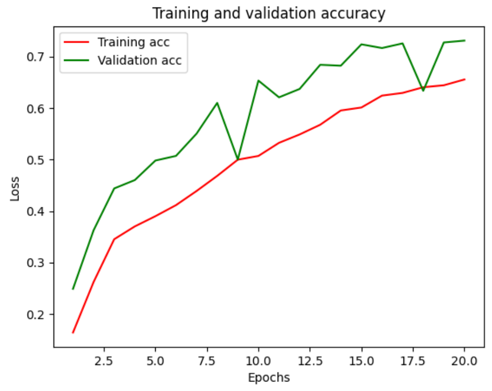


10FoldCM(Fold1)

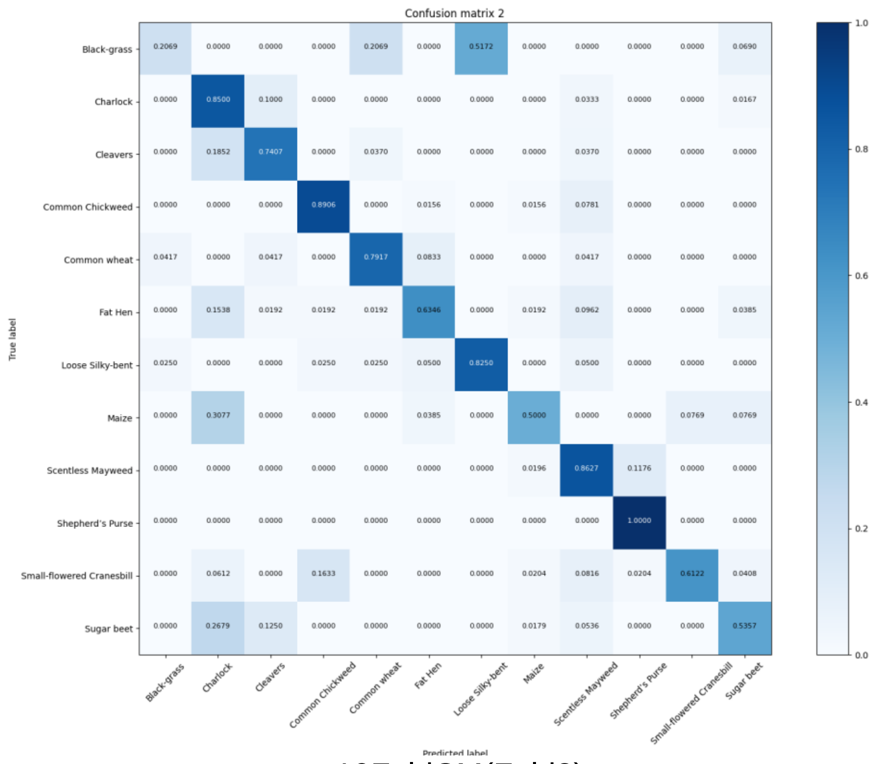
Image 24 - 10Folds Fold1



10FoldTVL(Fold2)

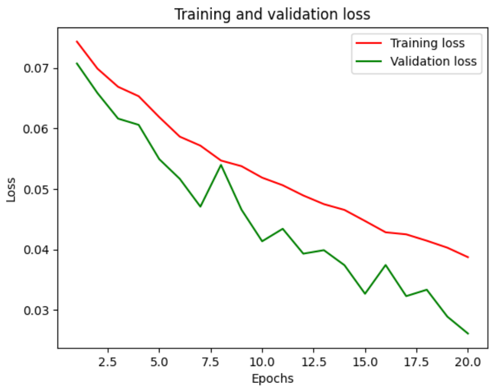


10FoldTVA(Fold2)

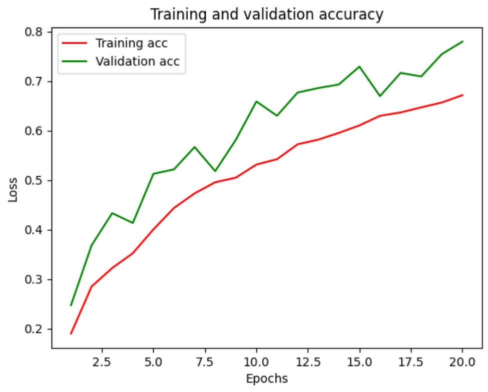


10FoldCM(Fold2)

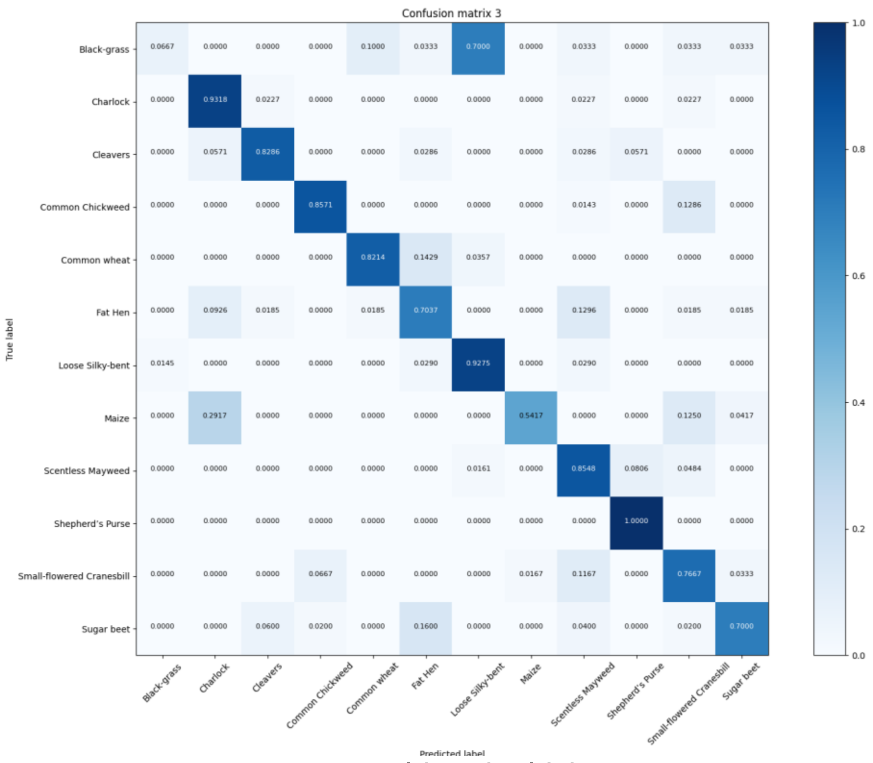
Image 25 - 10Folds Fold2



10FoldTVL(Fold3)



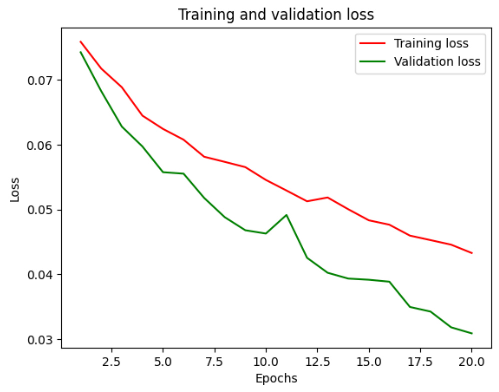
10FoldTVA(Fold3)



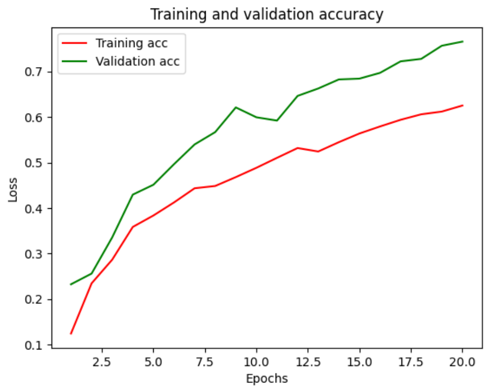
	Black-grass	Charlock	Cleavers	Common Chickweed	Common wheat	Fat Hen	Loose Silky-bent	Maize	Scentless Mayweed	Shepherd's Purse	Small-flowered Cranesbill	Sugar beet
Black-grass	0.0667	0.0000	0.0000	0.0000	0.1000	0.0333	0.7000	0.0000	0.0333	0.0000	0.0333	0.0333
Charlock	0.0000	0.9318	0.0227	0.0000	0.0000	0.0000	0.0000	0.0000	0.0227	0.0000	0.0227	0.0000
Cleavers	0.0000	0.0571	0.8286	0.0000	0.0000	0.0286	0.0000	0.0000	0.0286	0.0571	0.0000	0.0000
Common Chickweed	0.0000	0.0000	0.0000	0.8571	0.0000	0.0000	0.0000	0.0000	0.0143	0.0000	0.1286	0.0000
Common wheat	0.0000	0.0000	0.0000	0.0000	0.8214	0.1429	0.0357	0.0000	0.0000	0.0000	0.0000	0.0000
Fat Hen	0.0000	0.0926	0.0185	0.0000	0.0185	0.7037	0.0000	0.0000	0.1296	0.0000	0.0185	0.0185
Loose Silky-bent	0.0145	0.0000	0.0000	0.0000	0.0000	0.0290	0.9275	0.0000	0.0290	0.0000	0.0000	0.0000
Maize	0.0000	0.2917	0.0000	0.0000	0.0000	0.0000	0.0000	0.5417	0.0000	0.0000	0.1250	0.0417
Scentless Mayweed	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0161	0.0000	0.8548	0.0806	0.0484	0.0000
Shepherd's Purse	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000
Small-flowered Cranesbill	0.0000	0.0000	0.0000	0.0667	0.0000	0.0000	0.0000	0.0167	0.1167	0.0000	0.7667	0.0333
Sugar beet	0.0000	0.0000	0.0600	0.0200	0.0000	0.1600	0.0000	0.0000	0.0400	0.0000	0.0200	0.7000

10FoldCM(Fold3)

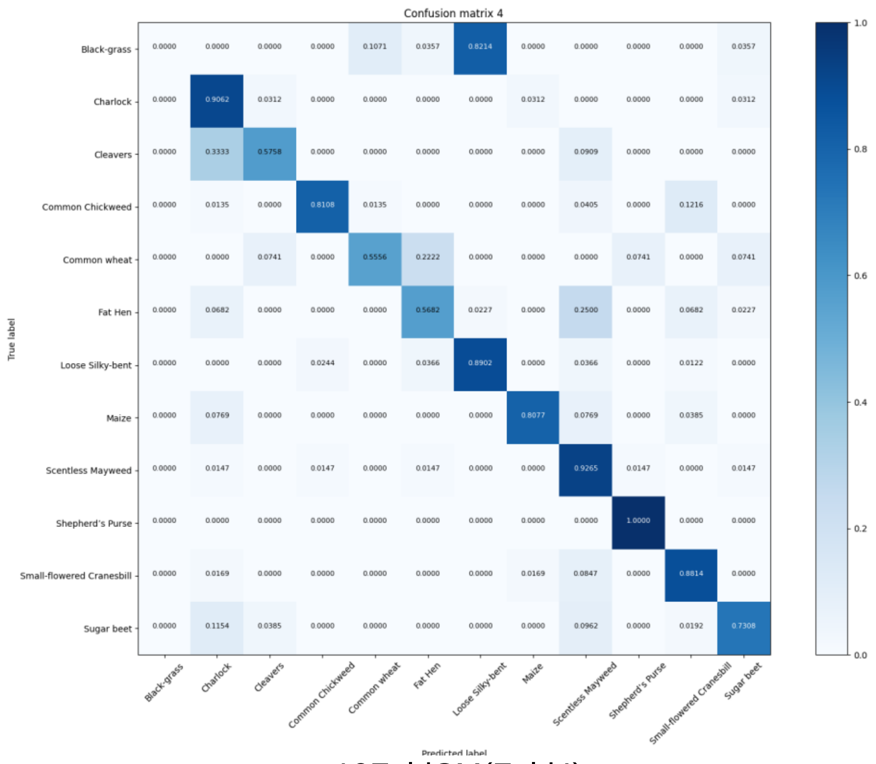
Image 26 - 10Folds Fold3



10FoldTVL(Fold4)



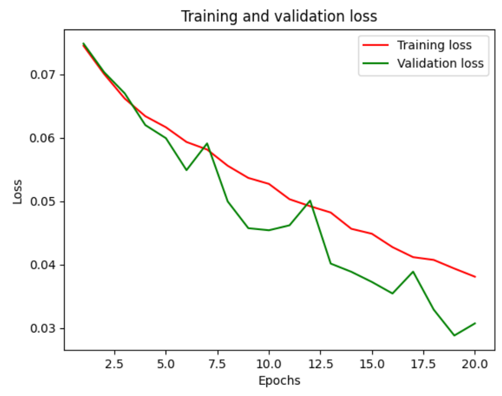
10FoldTVA(Fold4)



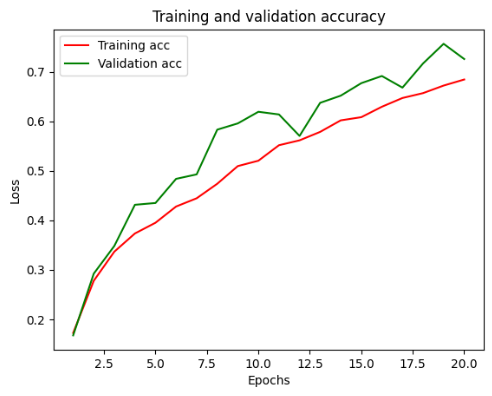
	Black-grass	Charlock	Cleavers	Common Chickweed	Common wheat	Fat Hen	Loose Silky-bent	Maize	Scentless Mayweed	Shepherd's Purse	Small-flowered Cranesbill	Sugar beet
Black-grass	0.0000	0.0000	0.0000	0.0000	0.1071	0.0357	0.8214	0.0000	0.0000	0.0000	0.0000	0.0357
Charlock	0.0000	0.9062	0.0312	0.0000	0.0000	0.0000	0.0000	0.0312	0.0000	0.0000	0.0000	0.0312
Cleavers	0.0000	0.3333	0.5758	0.0000	0.0000	0.0000	0.0000	0.0000	0.0909	0.0000	0.0000	0.0000
Common Chickweed	0.0000	0.0135	0.0000	0.8108	0.0135	0.0000	0.0000	0.0000	0.0405	0.0000	0.1216	0.0000
Common wheat	0.0000	0.0000	0.0741	0.0000	0.5556	0.2222	0.0000	0.0000	0.0000	0.0741	0.0000	0.0741
Fat Hen	0.0000	0.0682	0.0000	0.0000	0.0000	0.5682	0.0227	0.0000	0.2500	0.0000	0.0682	0.0227
Loose Silky-bent	0.0000	0.0000	0.0000	0.0244	0.0000	0.0366	0.8902	0.0000	0.0366	0.0000	0.0122	0.0000
Maize	0.0000	0.0769	0.0000	0.0000	0.0000	0.0000	0.0000	0.8077	0.0769	0.0000	0.0385	0.0000
Scentless Mayweed	0.0000	0.0147	0.0000	0.0147	0.0000	0.0147	0.0000	0.0000	0.9265	0.0147	0.0000	0.0147
Shepherd's Purse	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000
Small-flowered Cranesbill	0.0000	0.0169	0.0000	0.0000	0.0000	0.0000	0.0000	0.0169	0.0847	0.0000	0.8814	0.0000
Sugar beet	0.0000	0.1154	0.0385	0.0000	0.0000	0.0000	0.0000	0.0000	0.0962	0.0000	0.0192	0.7308

10FoldCM(Fold4)

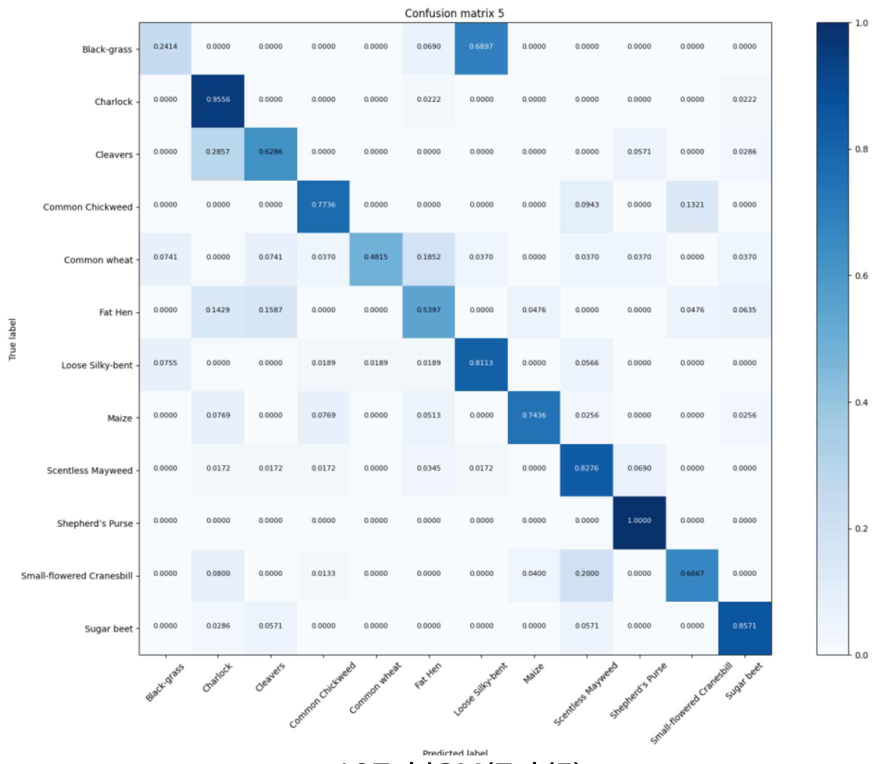
Image 27 - 10Folds Fold4



10FoldTVL(Fold5)

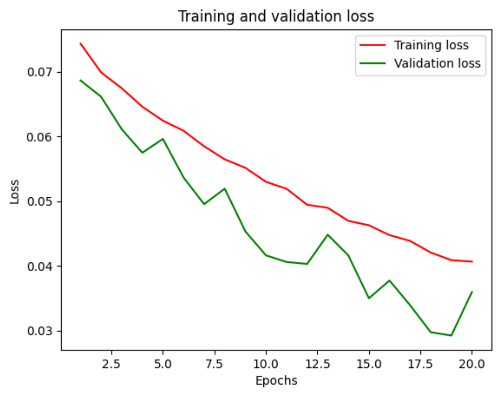


10FoldTVA(Fold5)

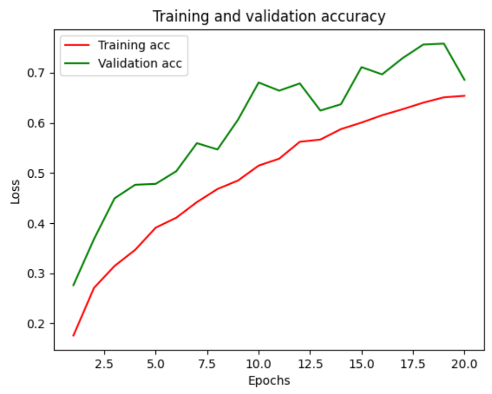


10FoldCM(Fold5)

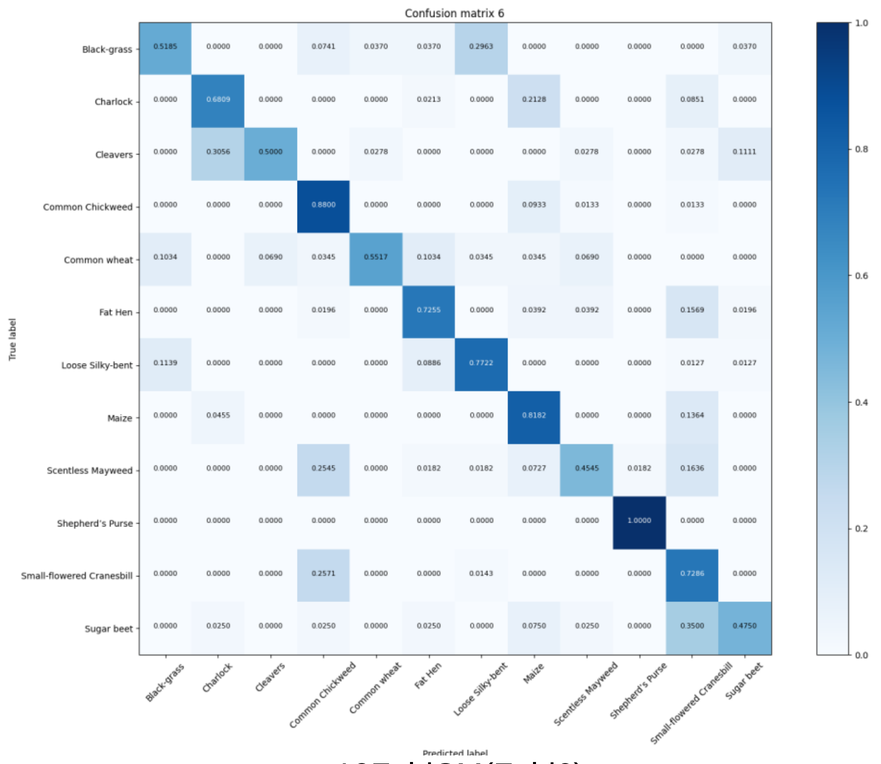
Image 28 - 10Folds Fold5



10FoldTVL(Fold6)

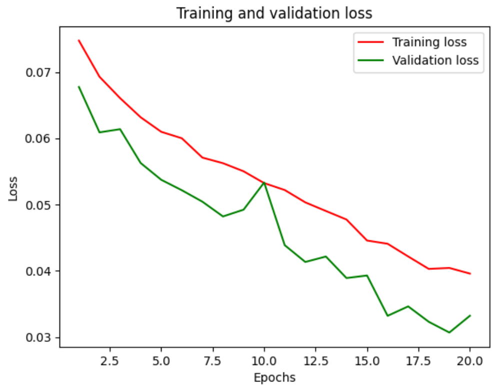


10FoldTVA(Fold6)

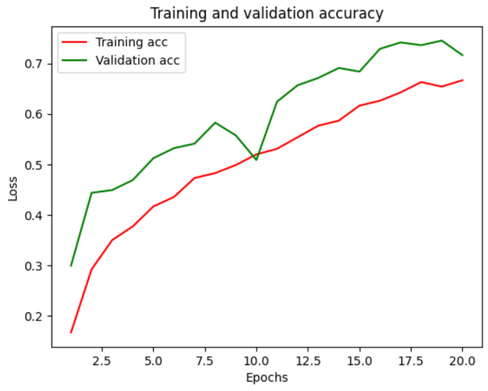


10FoldCM(Fold6)

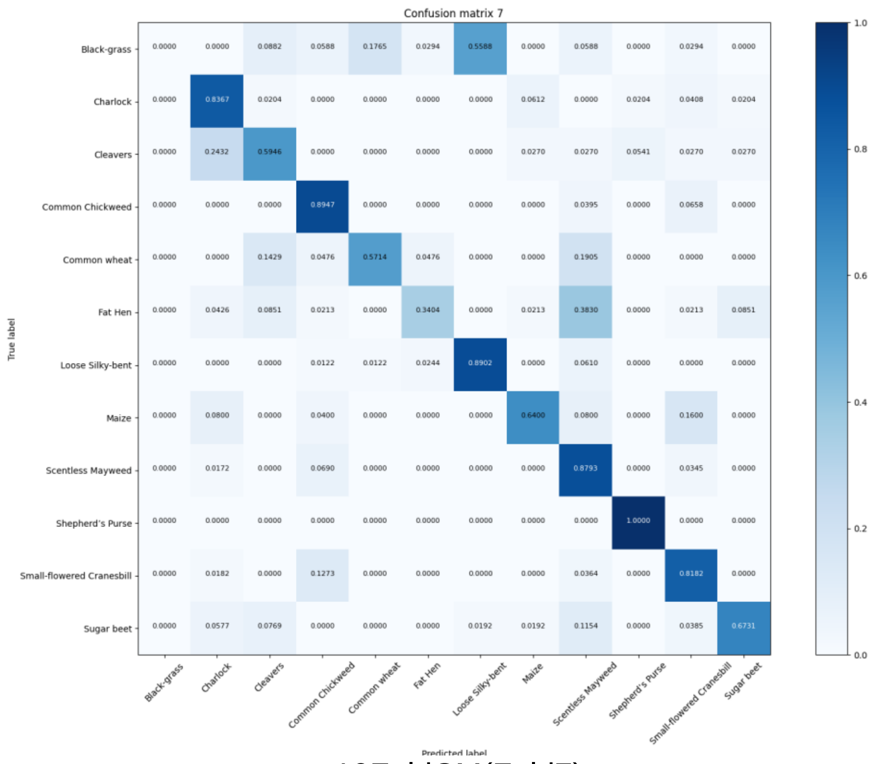
Image 29 - 10Folds Fold6



10FoldTVL(Fold7)



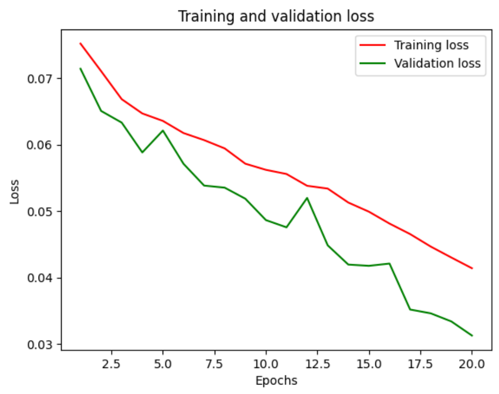
10FoldTVA(Fold7)



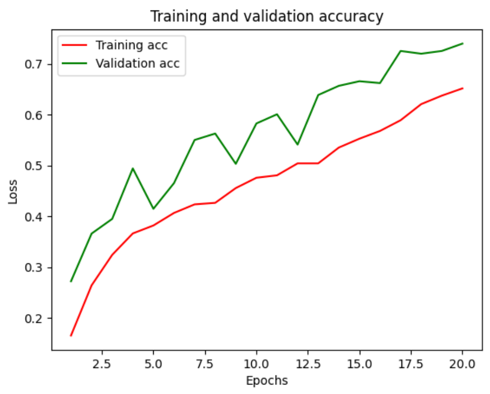
	Black-grass	Charlock	Cleavers	Common Chickweed	Common wheat	Fat Hen	Loose Silky-bent	Maize	Scentless Mayweed	Shepherd's Purse	Small-flowered Cranesbill	Sugar beet
Black-grass	0.0000	0.0000	0.0882	0.0588	0.1765	0.0294	0.5588	0.0000	0.0588	0.0000	0.0294	0.0000
Charlock	0.0000	0.8367	0.0204	0.0000	0.0000	0.0000	0.0000	0.0612	0.0000	0.0204	0.0408	0.0204
Cleavers	0.0000	0.2432	0.5946	0.0000	0.0000	0.0000	0.0000	0.0270	0.0270	0.0541	0.0270	0.0270
Common Chickweed	0.0000	0.0000	0.0000	0.8947	0.0000	0.0000	0.0000	0.0000	0.0395	0.0000	0.0658	0.0000
Common wheat	0.0000	0.0000	0.1429	0.0476	0.5714	0.0476	0.0000	0.0000	0.1905	0.0000	0.0000	0.0000
Fat Hen	0.0000	0.0426	0.0851	0.0213	0.0000	0.3404	0.0000	0.0213	0.3830	0.0000	0.0213	0.0851
Loose Silky-bent	0.0000	0.0000	0.0000	0.0122	0.0122	0.0244	0.8902	0.0000	0.0610	0.0000	0.0000	0.0000
Maize	0.0000	0.0800	0.0000	0.0400	0.0000	0.0000	0.0000	0.6400	0.0800	0.0000	0.1600	0.0000
Scentless Mayweed	0.0000	0.0172	0.0000	0.0690	0.0000	0.0000	0.0000	0.0000	0.8793	0.0000	0.0345	0.0000
Shepherd's Purse	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000
Small-flowered Cranesbill	0.0000	0.0182	0.0000	0.1273	0.0000	0.0000	0.0000	0.0000	0.0364	0.0000	0.8182	0.0000
Sugar beet	0.0000	0.0577	0.0769	0.0000	0.0000	0.0000	0.0192	0.0192	0.1154	0.0000	0.0385	0.6731

10FoldCM(Fold7)

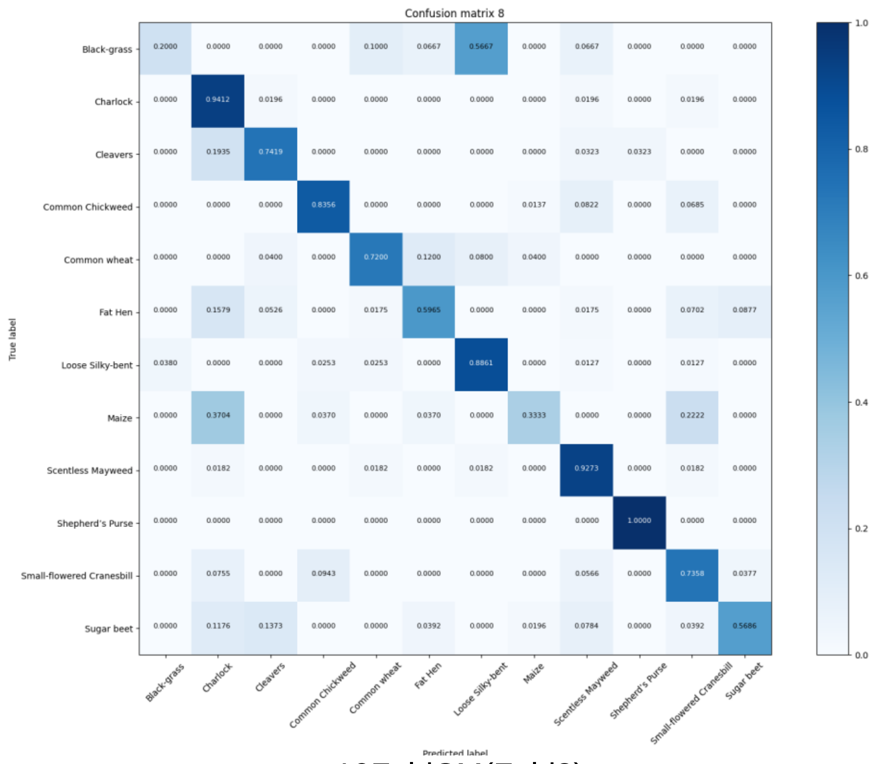
Image 30 - 10Folds Fold7



10FoldTVL(Fold8)

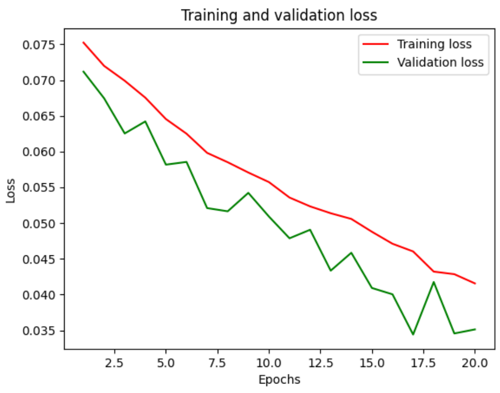


10FoldTVA(Fold8)

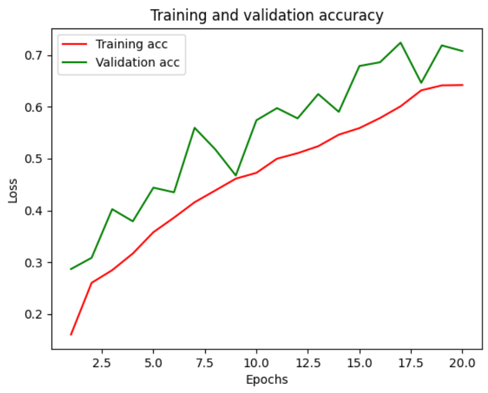


10FoldCM(Fold8)

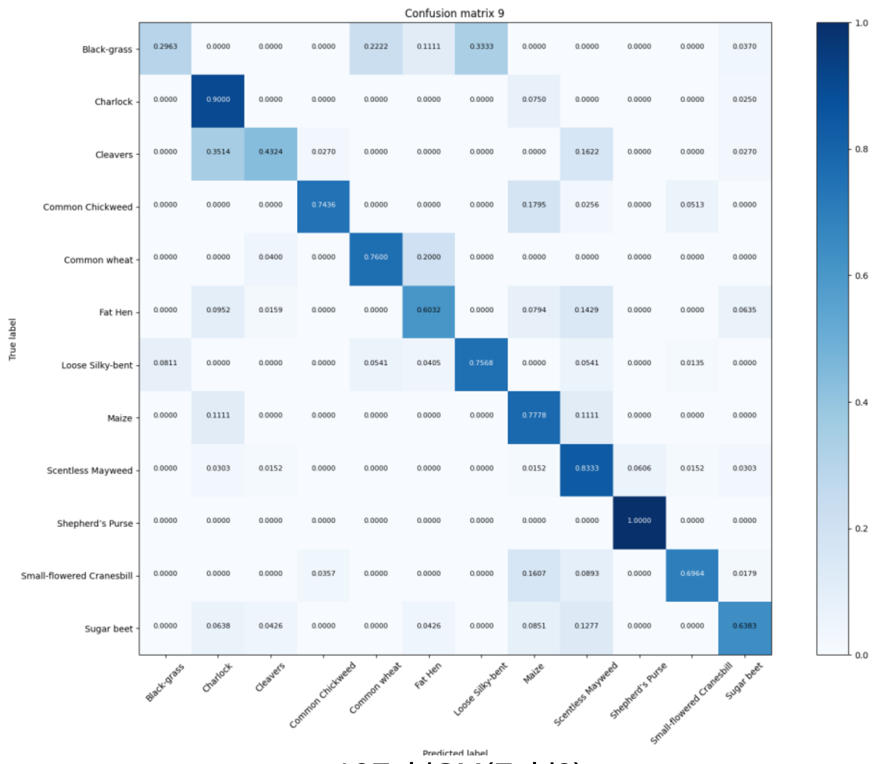
Image 31 - 10Folds Fold8



10FoldTVL(Fold9)

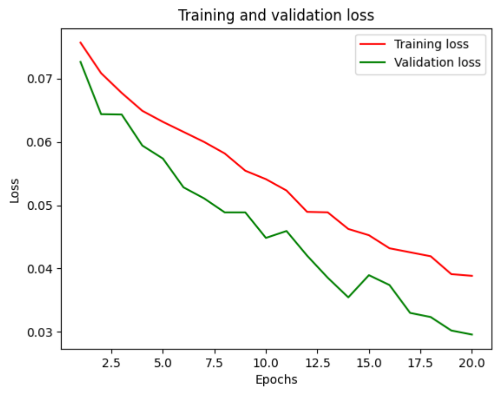


10FoldTVA(Fold9)

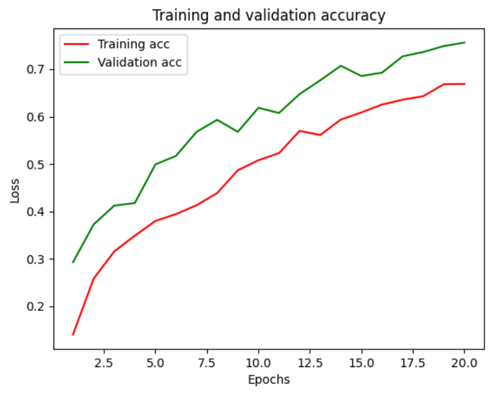


10FoldCM(Fold9)

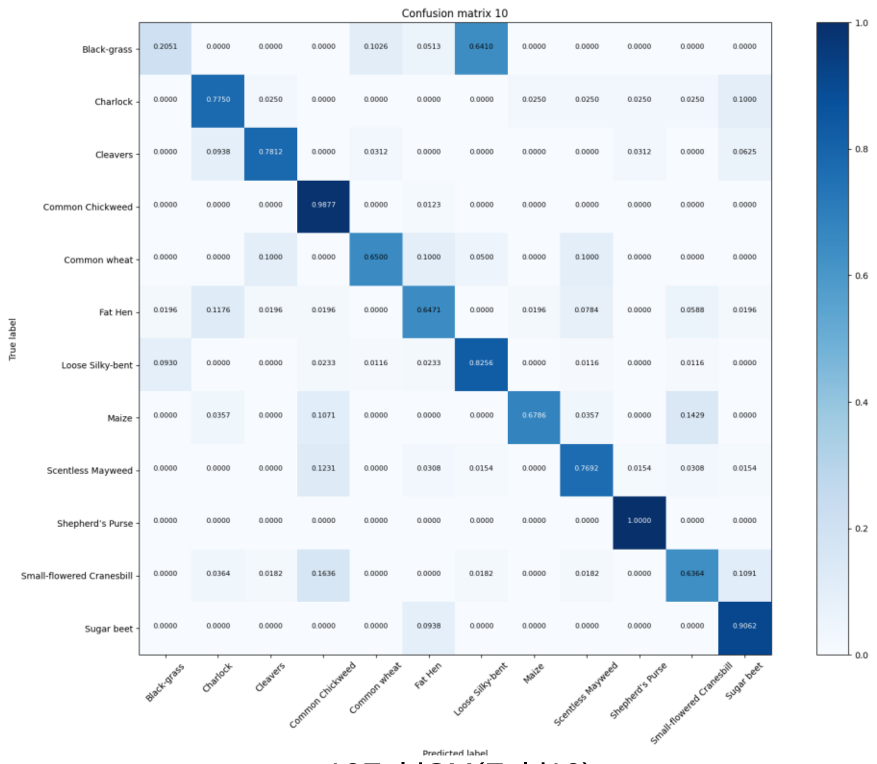
Image 32 - 10Folds Fold9



10FoldTVL(Fold10)



10FoldTVA(Fold10)

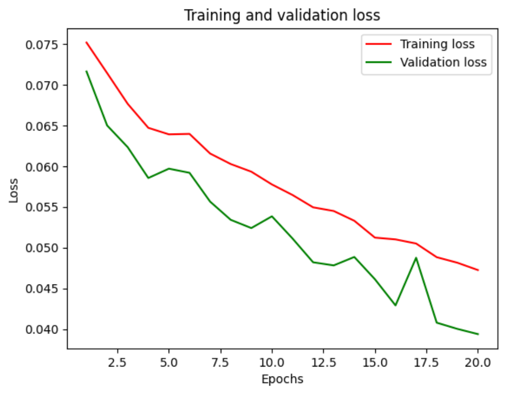


10FoldCM(Fold10)

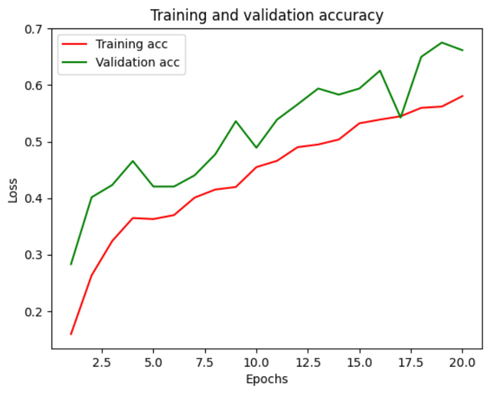
Image 33 - 10Folds Fold10

20 Epoch

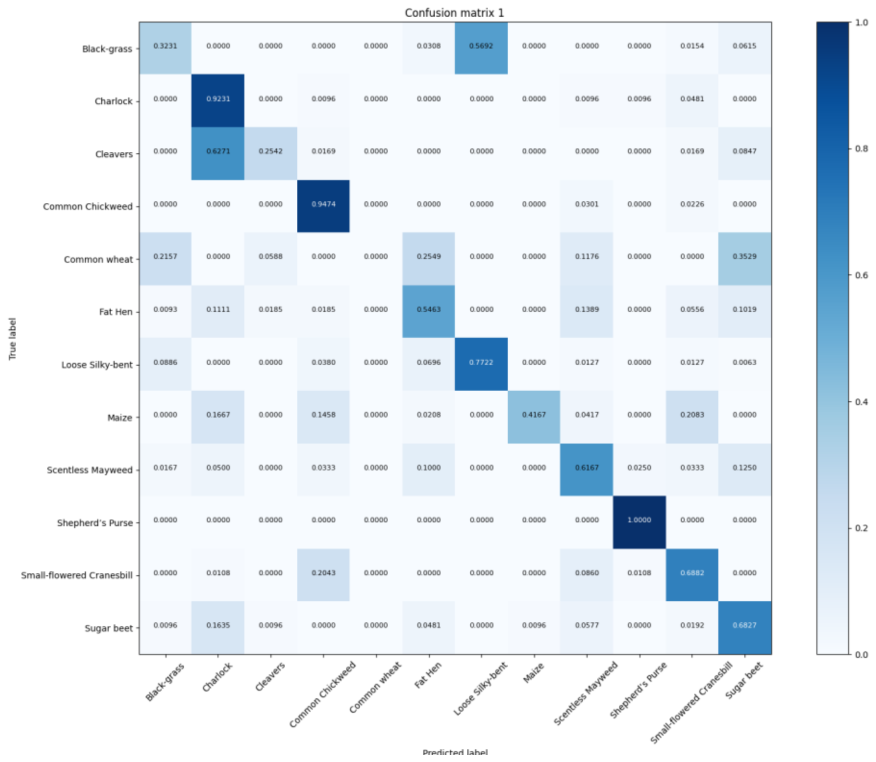
20 Epoch



20EpochTVL(Fold1)

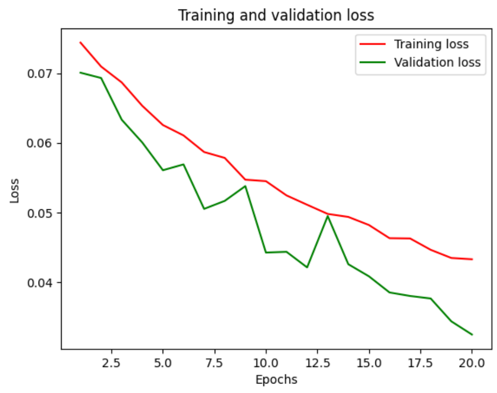


20EpochTVA(Fold1)

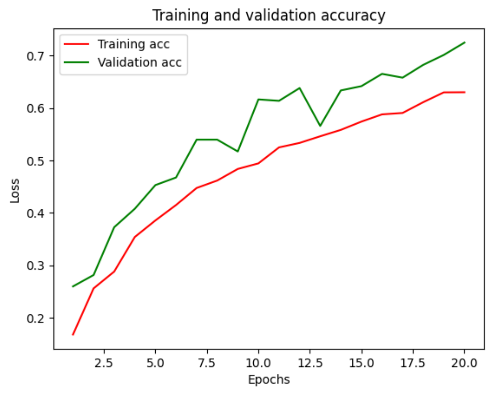


20EpochCM(Fold1)

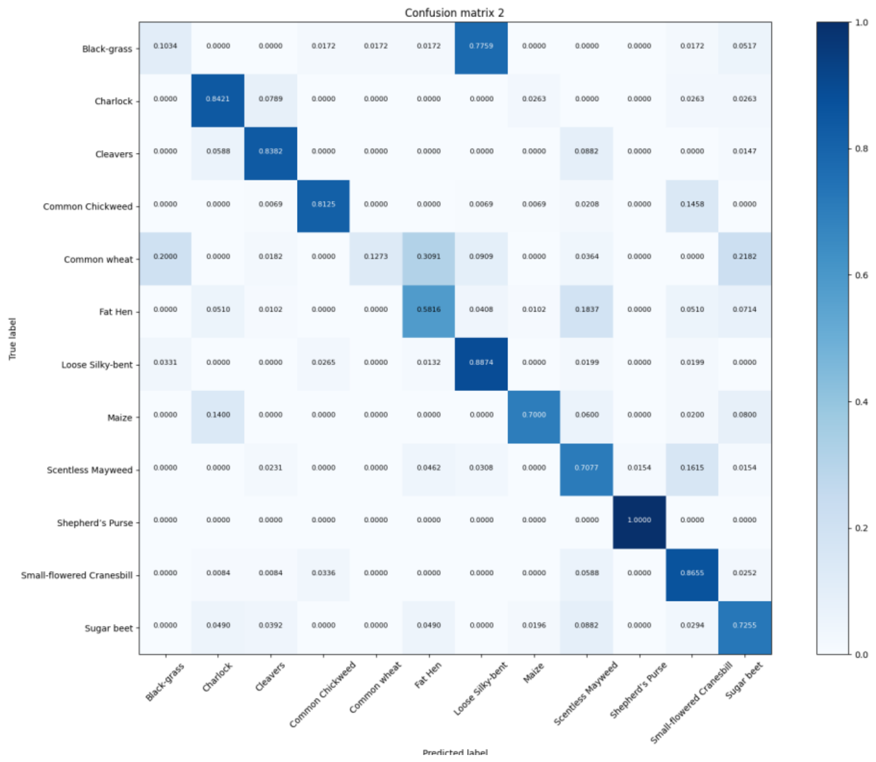
Image 34 - 20 Epochs Fold1



20EpochTVL(Fold2)

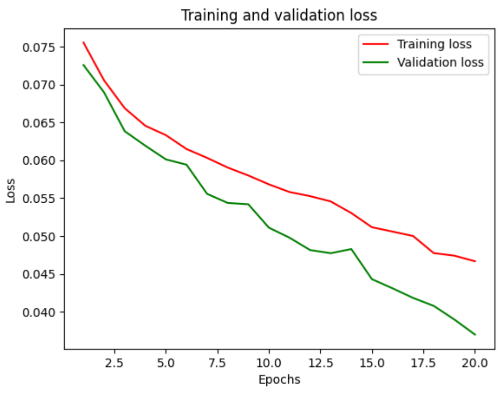


20EpochTVA(Fold2)

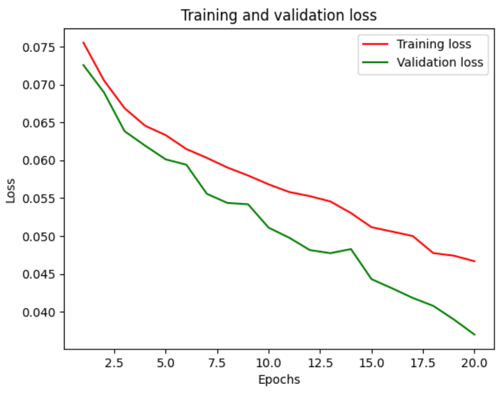


20EpochCM(Fold2)

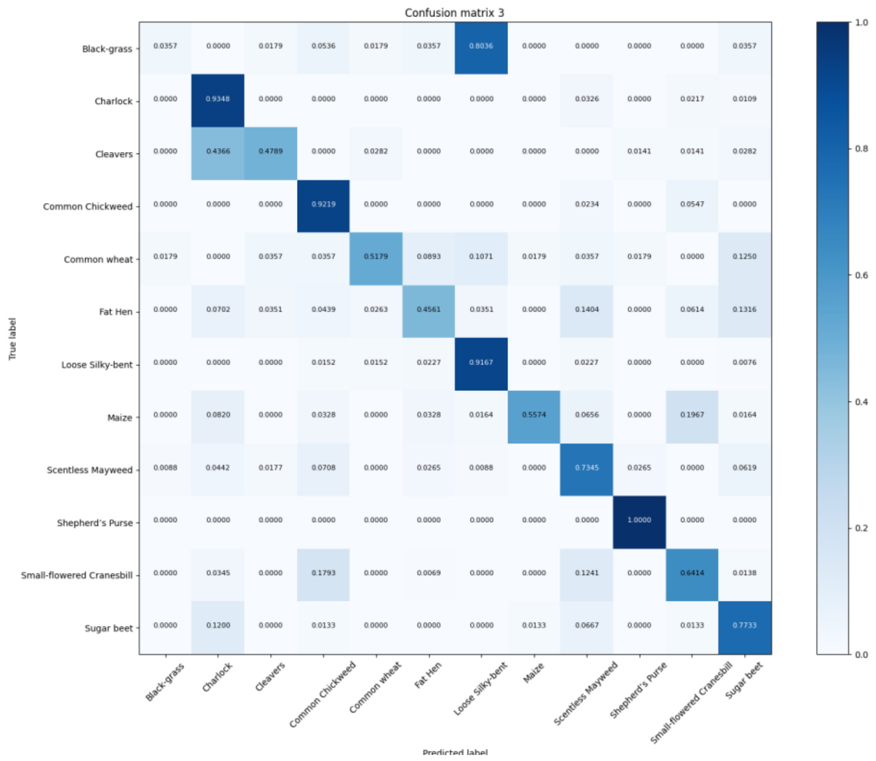
Image 35 - 20 Epochs Fold2



20EpochTVL(Fold3)

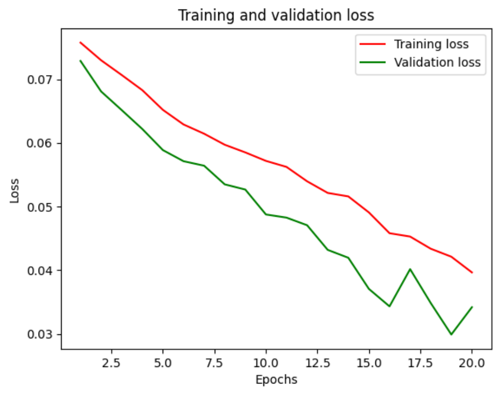


20EpochTVA(Fold3)

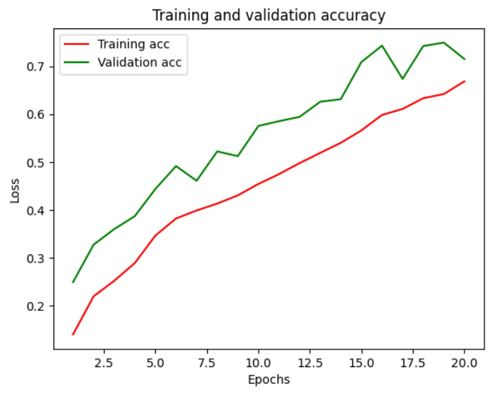


20EpochCM(Fold3)

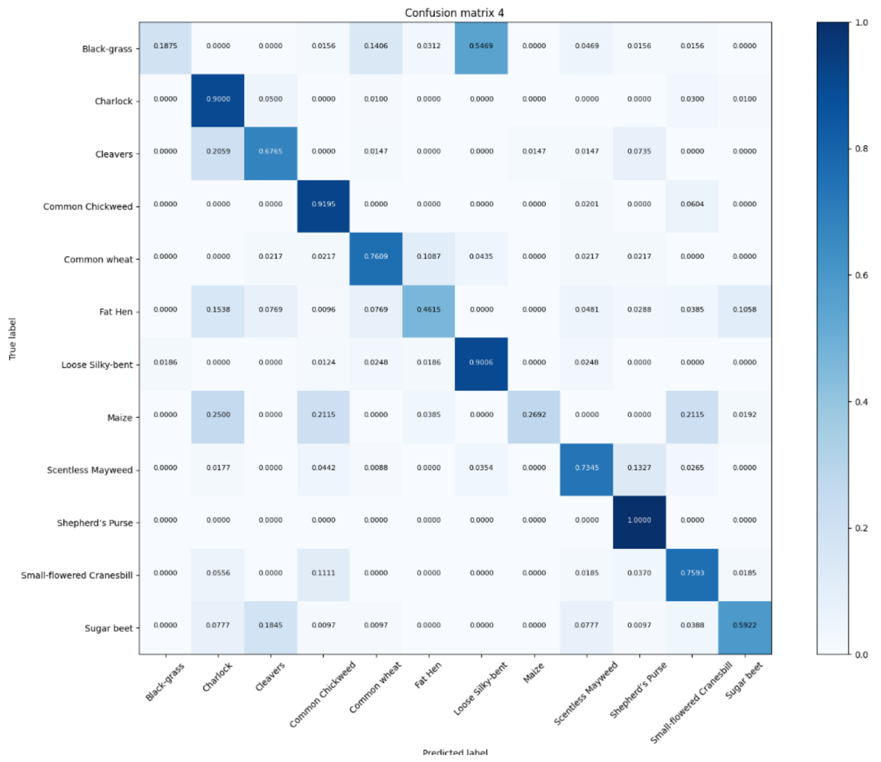
Image 36 - 20 Epochs Fold3



20EpochTVL(Fold4)



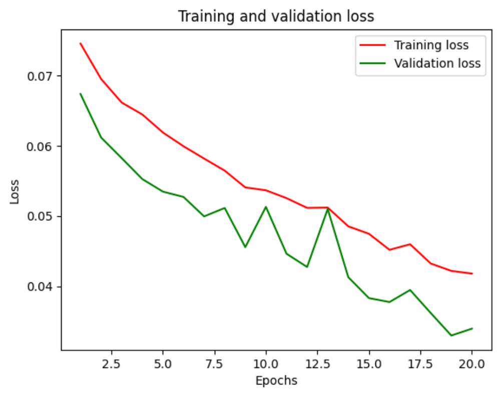
20EpochTVA(Fold4)



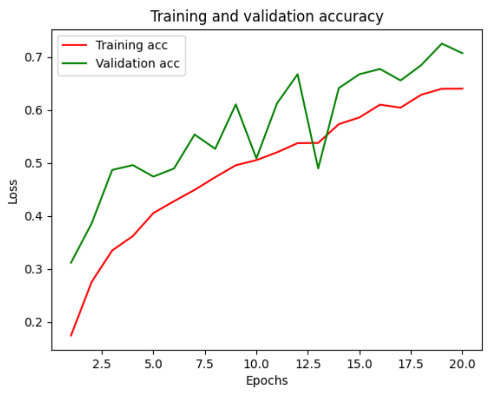
	Black-grass	Charlock	Cleavers	Common Chickweed	Common wheat	Fat Hen	Loose Silky-bent	Maize	Scentless Mayweed	Shepherd's Purse	Small-flowered Cranesbill	Sugar beet
Black-grass	0.1875	0.0000	0.0000	0.0156	0.1406	0.0312	0.5469	0.0000	0.0469	0.0156	0.0156	0.0000
Charlock	0.0000	0.9000	0.0500	0.0000	0.0100	0.0000	0.0000	0.0000	0.0000	0.0000	0.0300	0.0100
Cleavers	0.0000	0.2059	0.6765	0.0000	0.0147	0.0000	0.0000	0.0147	0.0147	0.0735	0.0000	0.0000
Common Chickweed	0.0000	0.0000	0.0000	0.9195	0.0000	0.0000	0.0000	0.0000	0.0201	0.0000	0.0604	0.0000
Common wheat	0.0000	0.0000	0.0217	0.0217	0.7609	0.1087	0.0435	0.0000	0.0217	0.0217	0.0000	0.0000
Fat Hen	0.0000	0.1538	0.0769	0.0096	0.0769	0.4615	0.0000	0.0000	0.0481	0.0288	0.0385	0.1058
Loose Silky-bent	0.0186	0.0000	0.0000	0.0124	0.0248	0.0186	0.9006	0.0000	0.0248	0.0000	0.0000	0.0000
Maize	0.0000	0.2500	0.0000	0.2115	0.0000	0.0385	0.0000	0.2692	0.0000	0.0000	0.2115	0.0192
Scentless Mayweed	0.0000	0.0177	0.0000	0.0442	0.0088	0.0000	0.0354	0.0000	0.7345	0.1327	0.0265	0.0000
Shepherd's Purse	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000
Small-flowered Cranesbill	0.0000	0.0556	0.0000	0.1111	0.0000	0.0000	0.0000	0.0000	0.0185	0.0370	0.7593	0.0185
Sugar beet	0.0000	0.0777	0.1845	0.0097	0.0097	0.0000	0.0000	0.0000	0.0777	0.0097	0.0388	0.5922

20EpochCM(Fold4)

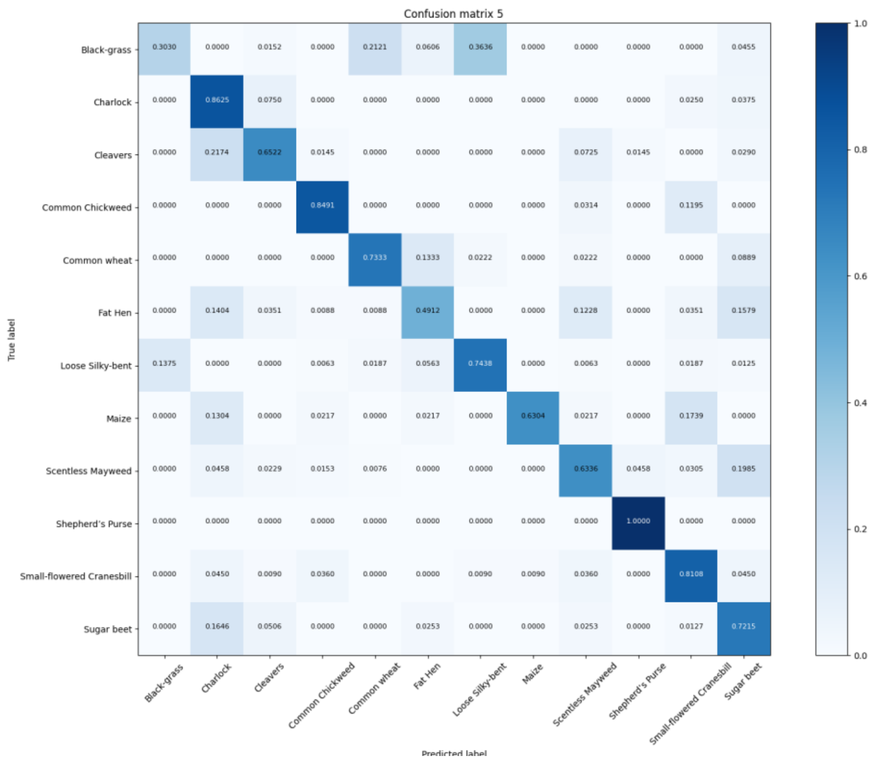
Image 37 - 20 Epochs Fold4



20EpochTVL(Fold5)



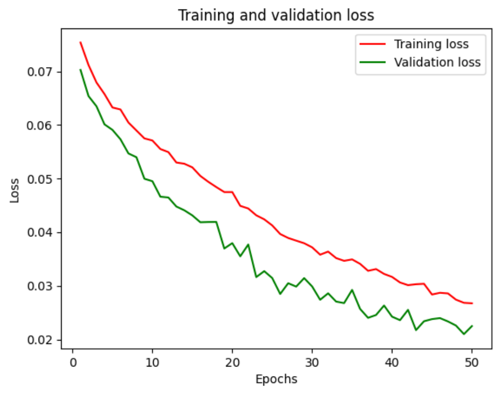
20EpochTVA(Fold5)



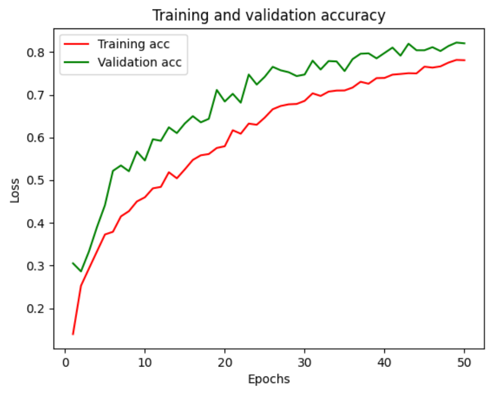
20EpochCM(Fold5)

Image 38 - 20 Epochs Fold5

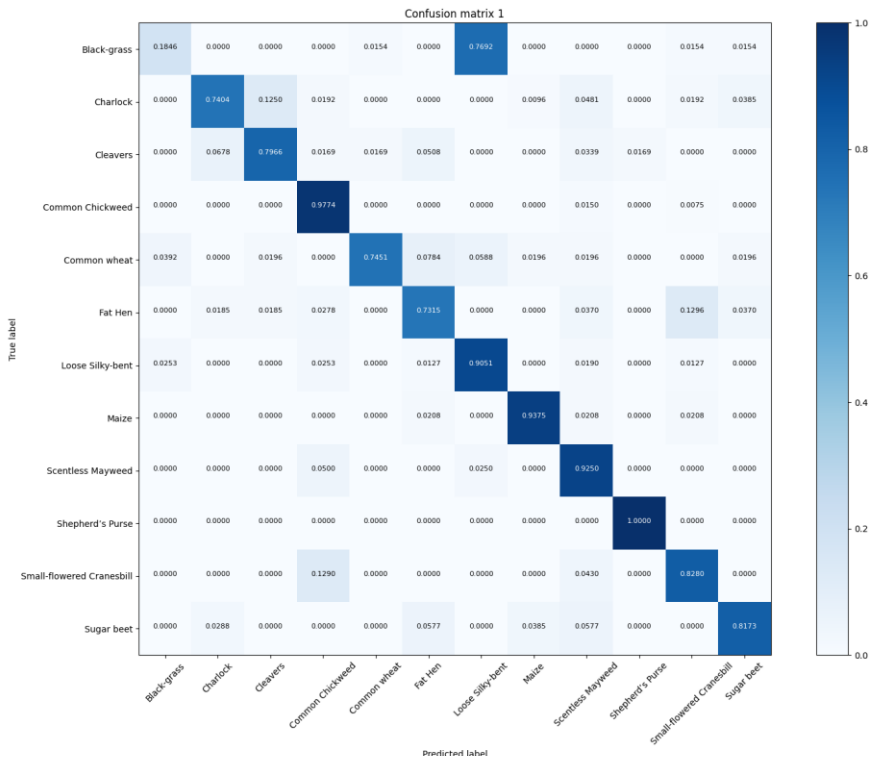
50Epochs



50EpochTVL(Fold1)



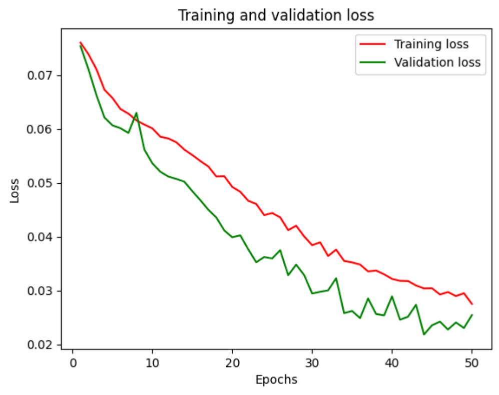
50EpochTVA(Fold1)



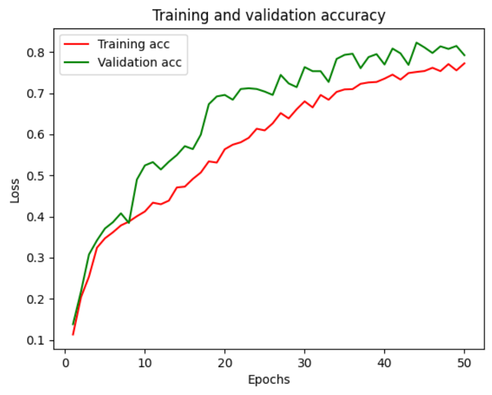
	Black-grass	Charlock	Cleavers	Common Chickweed	Common wheat	Fat Hen	Loose Silky-bent	Maize	Scentless Mayweed	Shepherd's Purse	Small-flowered Cranesbill	Sugar beet
Black-grass	0.1846	0.0000	0.0000	0.0000	0.0154	0.0000	0.7692	0.0000	0.0000	0.0000	0.0154	0.0154
Charlock	0.0000	0.7404	0.1250	0.0192	0.0000	0.0000	0.0000	0.0096	0.0481	0.0000	0.0192	0.0385
Cleavers	0.0000	0.0678	0.7966	0.0169	0.0169	0.0508	0.0000	0.0000	0.0339	0.0169	0.0000	0.0000
Common Chickweed	0.0000	0.0000	0.0000	0.9774	0.0000	0.0000	0.0000	0.0000	0.0150	0.0000	0.0075	0.0000
Common wheat	0.0392	0.0000	0.0196	0.0000	0.7451	0.0784	0.0588	0.0196	0.0196	0.0000	0.0000	0.0196
Fat Hen	0.0000	0.0185	0.0185	0.0278	0.0000	0.7315	0.0000	0.0000	0.0370	0.0000	0.1296	0.0370
Loose Silky-bent	0.0253	0.0000	0.0000	0.0253	0.0000	0.0127	0.9051	0.0000	0.0190	0.0000	0.0127	0.0000
Maize	0.0000	0.0000	0.0000	0.0000	0.0000	0.0208	0.0000	0.9375	0.0208	0.0000	0.0208	0.0000
Scentless Mayweed	0.0000	0.0000	0.0000	0.0500	0.0000	0.0000	0.0250	0.0000	0.9250	0.0000	0.0000	0.0000
Shepherd's Purse	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000
Small-flowered Cranesbill	0.0000	0.0000	0.0000	0.1290	0.0000	0.0000	0.0000	0.0000	0.0430	0.0000	0.8280	0.0000
Sugar beet	0.0000	0.0288	0.0000	0.0000	0.0000	0.0577	0.0000	0.0385	0.0577	0.0000	0.0000	0.8173

50EpochCM(Fold1)

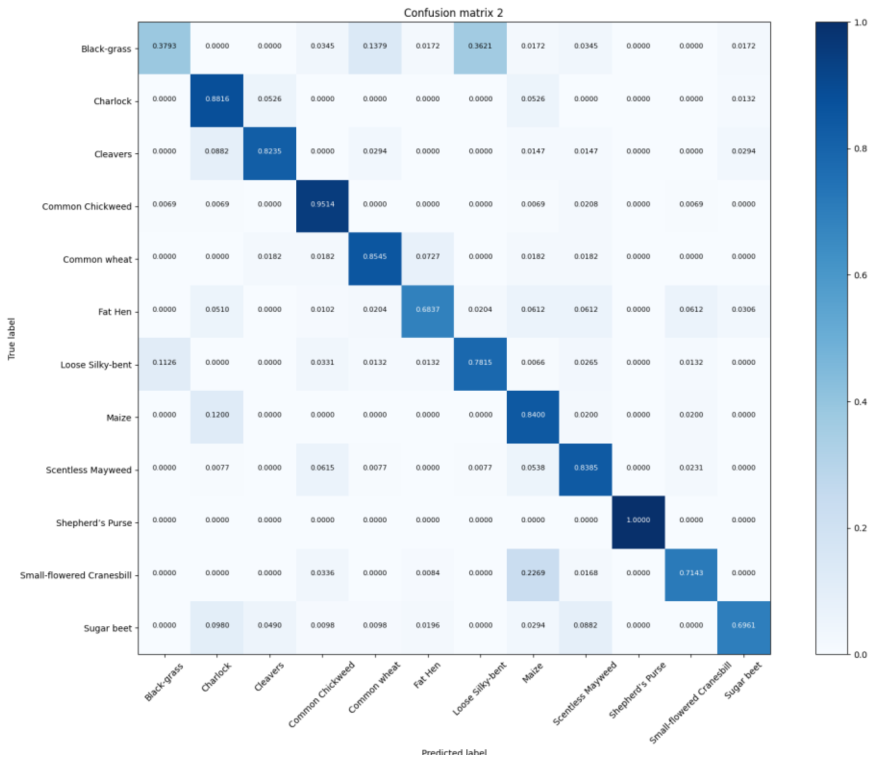
Image 39 - 50 Epochs Fold1



50EpochTVL(Fold2)



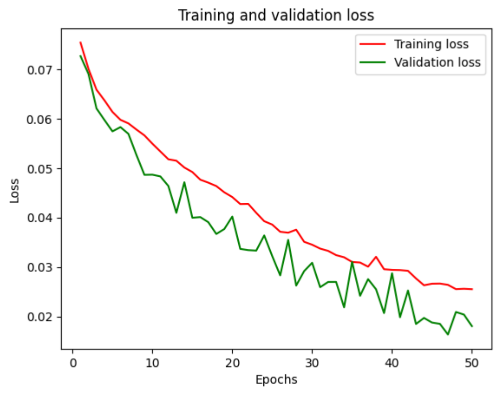
50EpochTVA(Fold2)


Confusion matrix 2
True label
Predicted label

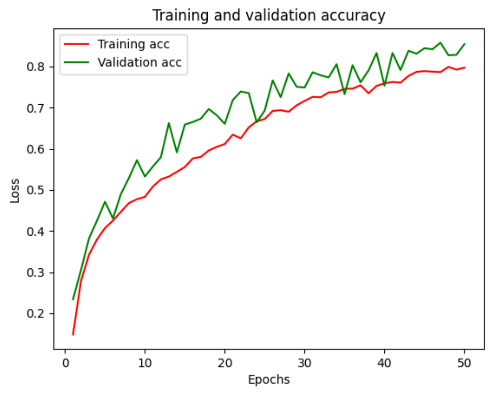
True label \ Predicted label	Black-grass	Charlock	Cleavers	Common Chickweed	Common wheat	Fat Hen	Loose Silky-bent	Maize	Scentless Mayweed	Shepherd's Purse	Small-flowered Cranesbill	Sugar beet
Black-grass	0.3793	0.0000	0.0000	0.0345	0.1379	0.0172	0.3621	0.0172	0.0345	0.0000	0.0000	0.0172
Charlock	0.0000	0.8816	0.0526	0.0000	0.0000	0.0000	0.0000	0.0526	0.0000	0.0000	0.0000	0.0132
Cleavers	0.0000	0.0882	0.8235	0.0000	0.0294	0.0000	0.0000	0.0147	0.0147	0.0000	0.0000	0.0294
Common Chickweed	0.0069	0.0069	0.0000	0.9514	0.0000	0.0000	0.0000	0.0069	0.0208	0.0000	0.0069	0.0000
Common wheat	0.0000	0.0000	0.0182	0.0182	0.8545	0.0727	0.0000	0.0182	0.0182	0.0000	0.0000	0.0000
Fat Hen	0.0000	0.0510	0.0000	0.0102	0.0204	0.6837	0.0204	0.0612	0.0612	0.0000	0.0612	0.0306
Loose Silky-bent	0.1126	0.0000	0.0000	0.0331	0.0132	0.0132	0.7815	0.0066	0.0265	0.0000	0.0132	0.0000
Maize	0.0000	0.1200	0.0000	0.0000	0.0000	0.0000	0.0000	0.8400	0.0200	0.0000	0.0200	0.0000
Scentless Mayweed	0.0000	0.0077	0.0000	0.0615	0.0077	0.0000	0.0077	0.0538	0.8385	0.0000	0.0231	0.0000
Shepherd's Purse	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000
Small-flowered Cranesbill	0.0000	0.0000	0.0000	0.0336	0.0000	0.0084	0.0000	0.2269	0.0168	0.0000	0.7143	0.0000
Sugar beet	0.0000	0.0980	0.0490	0.0098	0.0098	0.0196	0.0000	0.0294	0.0882	0.0000	0.0000	0.6961

50EpochCM(Fold2)

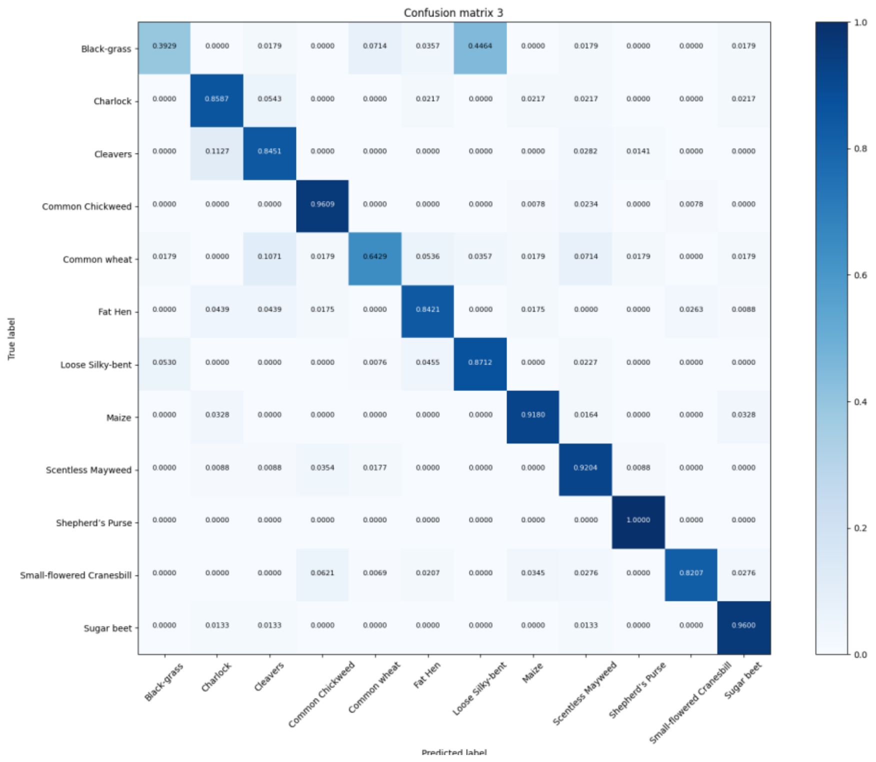
Image 40 - 50 Epochs Fold2



50EpochTVL(Fold3)

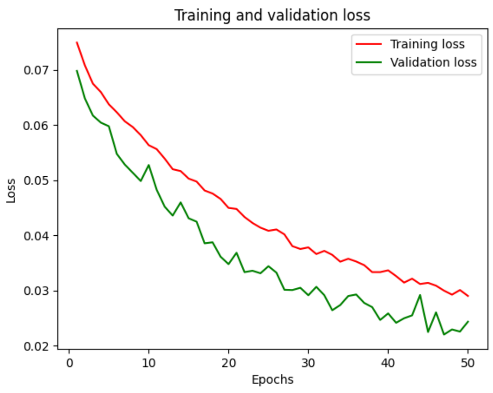


50EpochTVA(Fold3)

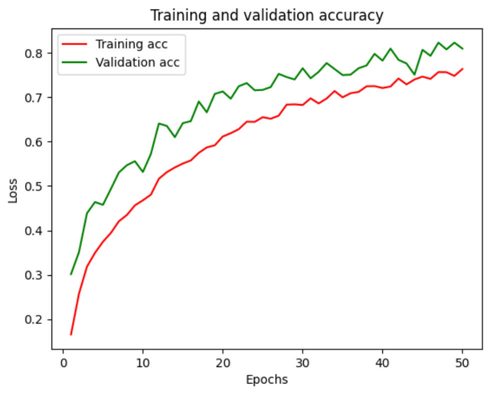


50EpochCM(Fold3)

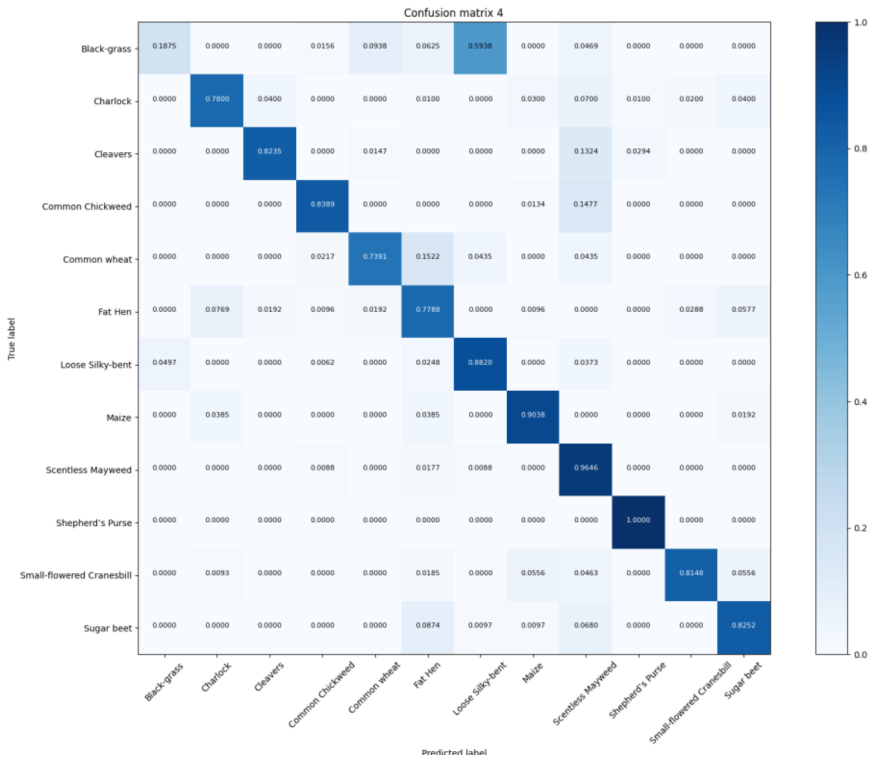
Image 41 - 50 Epochs Fold3



50EpochTVL(Fold4)



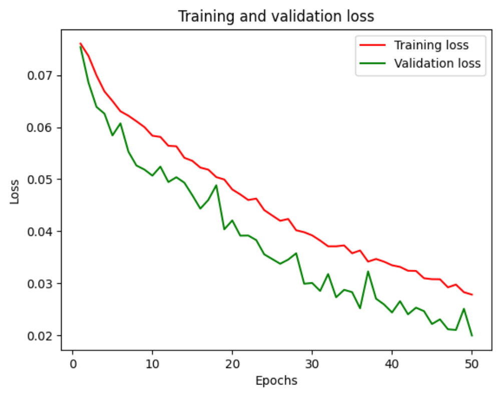
50EpochTVA(Fold4)



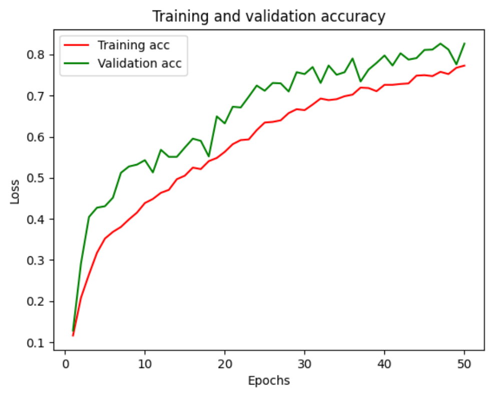
	Black-grass	Charlock	Cleavers	Common Chickweed	Common wheat	Fat Hen	Loose Silky-bent	Maize	Scentless Mayweed	Shepherd's Purse	Small-flowered Cranesbill	Sugar beet
Black-grass	0.1875	0.0000	0.0000	0.0156	0.0938	0.0625	0.5938	0.0000	0.0469	0.0000	0.0000	0.0000
Charlock	0.0000	0.7800	0.0400	0.0000	0.0000	0.0100	0.0000	0.0300	0.0700	0.0100	0.0200	0.0400
Cleavers	0.0000	0.0000	0.8235	0.0000	0.0147	0.0000	0.0000	0.0000	0.1324	0.0294	0.0000	0.0000
Common Chickweed	0.0000	0.0000	0.0000	0.8389	0.0000	0.0000	0.0000	0.0134	0.1477	0.0000	0.0000	0.0000
Common wheat	0.0000	0.0000	0.0000	0.0217	0.7391	0.1522	0.0435	0.0000	0.0435	0.0000	0.0000	0.0000
Fat Hen	0.0000	0.0769	0.0192	0.0096	0.0192	0.7788	0.0000	0.0096	0.0000	0.0000	0.0288	0.0577
Loose Silky-bent	0.0497	0.0000	0.0000	0.0062	0.0000	0.0248	0.8820	0.0000	0.0373	0.0000	0.0000	0.0000
Maize	0.0000	0.0385	0.0000	0.0000	0.0000	0.0385	0.0000	0.9038	0.0000	0.0000	0.0000	0.0192
Scentless Mayweed	0.0000	0.0000	0.0000	0.0088	0.0000	0.0177	0.0088	0.0000	0.9646	0.0000	0.0000	0.0000
Shepherd's Purse	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000
Small-flowered Cranesbill	0.0000	0.0093	0.0000	0.0000	0.0000	0.0185	0.0000	0.0556	0.0463	0.0000	0.8148	0.0556
Sugar beet	0.0000	0.0000	0.0000	0.0000	0.0000	0.0874	0.0097	0.0097	0.0680	0.0000	0.0000	0.8252

50EpochCM(Fold4)

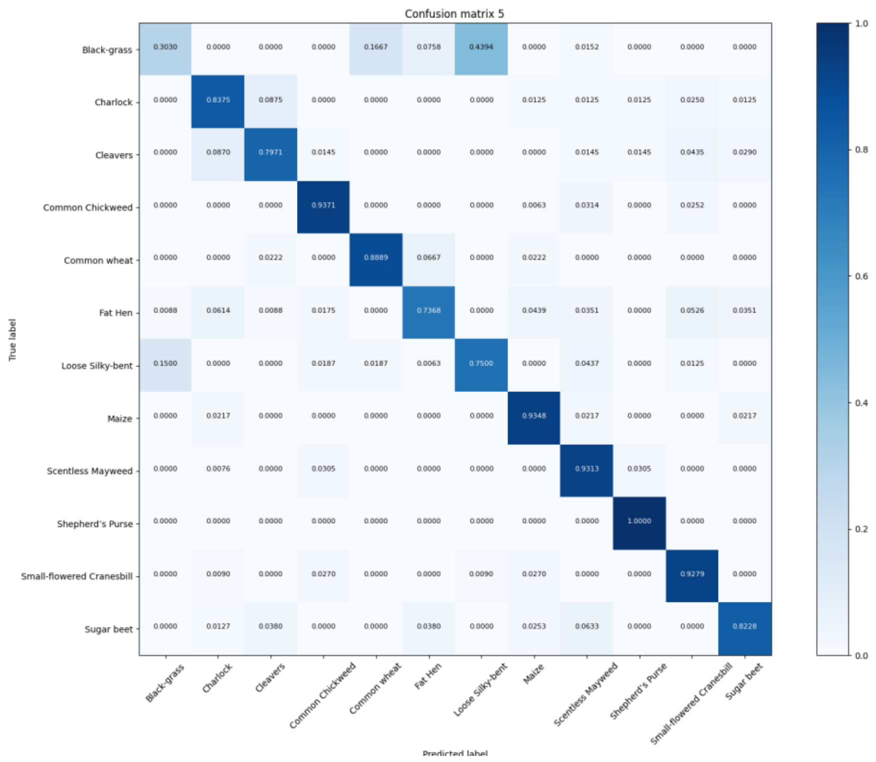
Image 42 - 50 Epochs Fold4



50EpochTVL(Fold5)



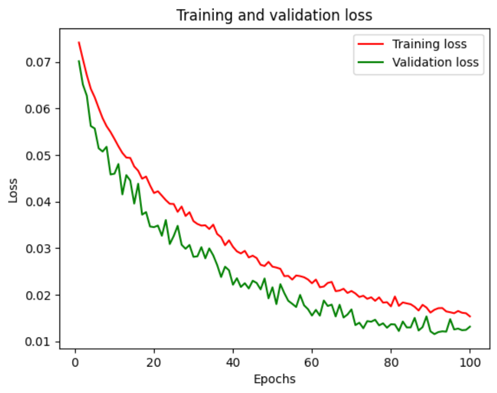
50EpochTVA(Fold5)

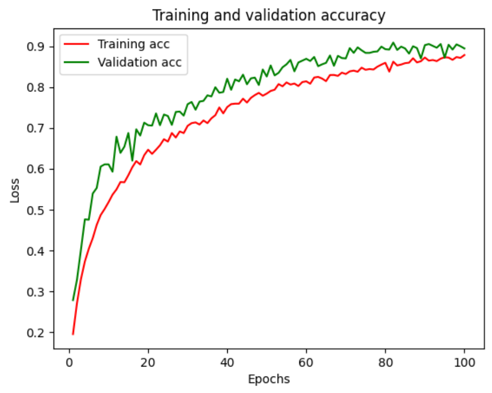


50EpochCM(Fold5)

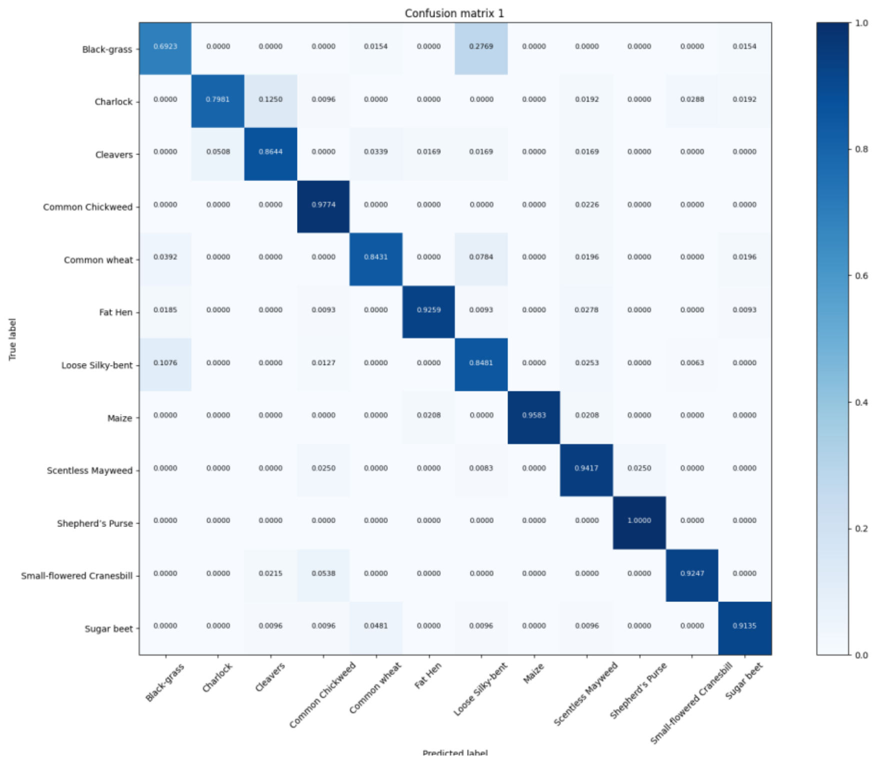
Image 43 - 50 Epochs Fold5

100 Epochs





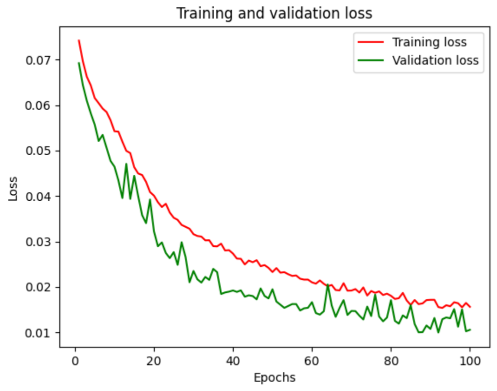
100EpochTVL(Fold1) 100EpochTVA(Fold1)



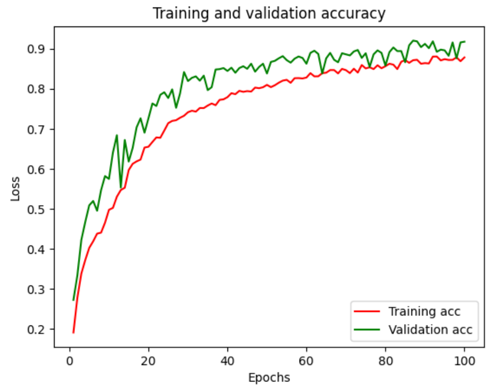
	Black-grass	Charlock	Cleavers	Common Chickweed	Common wheat	Fat Hen	Loose Silky-bent	Maize	Scentless Mayweed	Shepherd's Purse	Small-flowered Cranesbill	Sugar beet
Black-grass	0.6923	0.0000	0.0000	0.0000	0.0154	0.0000	0.2769	0.0000	0.0000	0.0000	0.0000	0.0154
Charlock	0.0000	0.7981	0.1250	0.0096	0.0000	0.0000	0.0000	0.0000	0.0192	0.0000	0.0288	0.0192
Cleavers	0.0000	0.0508	0.8644	0.0000	0.0339	0.0169	0.0169	0.0000	0.0169	0.0000	0.0000	0.0000
Common Chickweed	0.0000	0.0000	0.0000	0.9774	0.0000	0.0000	0.0000	0.0000	0.0226	0.0000	0.0000	0.0000
Common wheat	0.0392	0.0000	0.0000	0.0000	0.8431	0.0000	0.0784	0.0000	0.0196	0.0000	0.0000	0.0196
Fat Hen	0.0185	0.0000	0.0000	0.0093	0.0000	0.9259	0.0093	0.0000	0.0278	0.0000	0.0000	0.0093
Loose Silky-bent	0.1076	0.0000	0.0000	0.0127	0.0000	0.0000	0.8481	0.0000	0.0253	0.0000	0.0063	0.0000
Maize	0.0000	0.0000	0.0000	0.0000	0.0000	0.0208	0.0000	0.9583	0.0208	0.0000	0.0000	0.0000
Scentless Mayweed	0.0000	0.0000	0.0000	0.0250	0.0000	0.0000	0.0083	0.0000	0.9417	0.0250	0.0000	0.0000
Shepherd's Purse	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000
Small-flowered Cranesbill	0.0000	0.0000	0.0215	0.0538	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.9247	0.0000
Sugar beet	0.0000	0.0000	0.0096	0.0096	0.0481	0.0000	0.0096	0.0000	0.0096	0.0000	0.0000	0.9135

100EpochCM(Fold1)

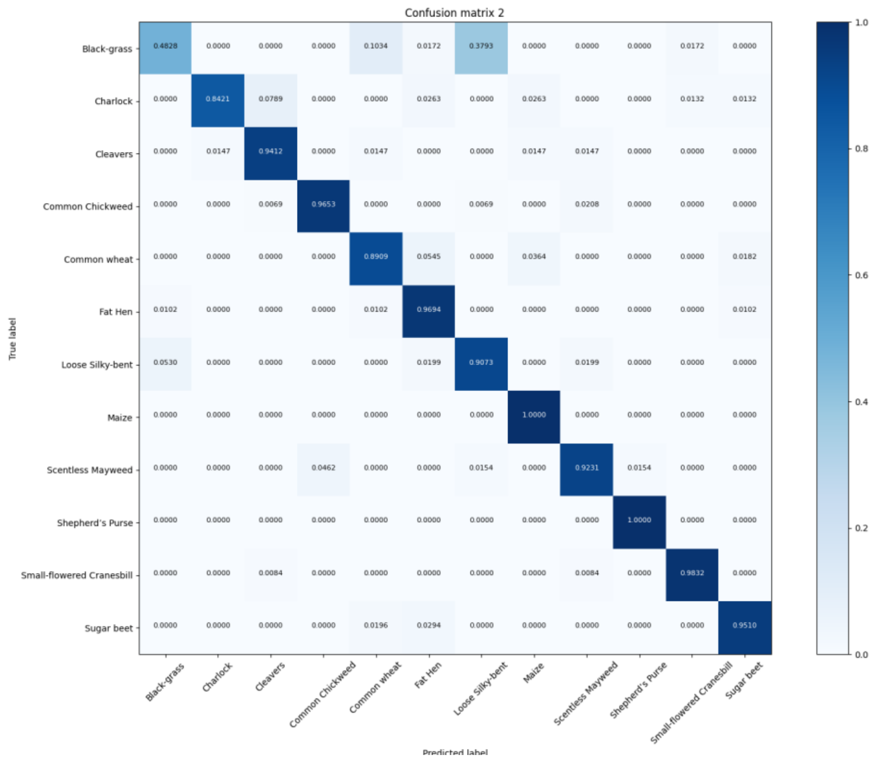
Image 44 - 100 Epochs Fold1



100EpochTVL(Fold2)



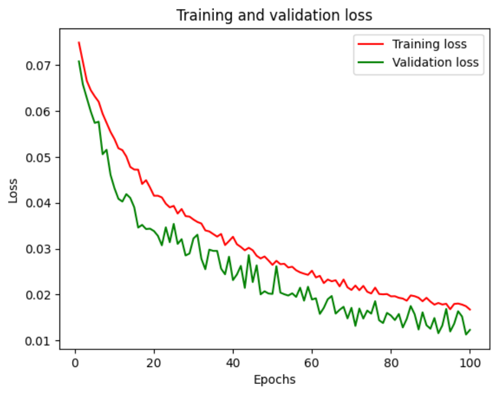
100EpochTVA(Fold2)



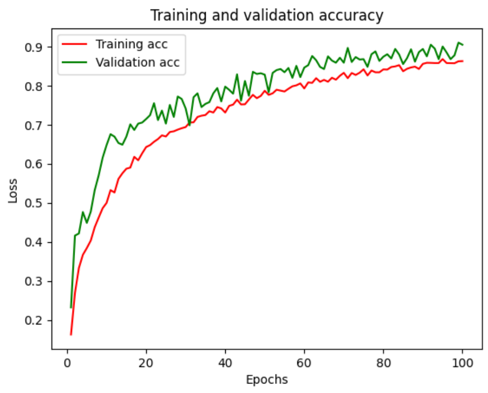
	Black-grass	Charlock	Cleavers	Common Chickweed	Common wheat	Fat Hen	Loose Silky-bent	Maize	Scentless Mayweed	Shepherd's Purse	Small-flowered Cranesbill	Sugar beet
Black-grass	0.4828	0.0000	0.0000	0.0000	0.1034	0.0172	0.3793	0.0000	0.0000	0.0000	0.0172	0.0000
Charlock	0.0000	0.8421	0.0789	0.0000	0.0000	0.0263	0.0000	0.0263	0.0000	0.0000	0.0132	0.0132
Cleavers	0.0000	0.0147	0.9412	0.0000	0.0147	0.0000	0.0000	0.0147	0.0147	0.0000	0.0000	0.0000
Common Chickweed	0.0000	0.0000	0.0069	0.9653	0.0000	0.0000	0.0069	0.0000	0.0208	0.0000	0.0000	0.0000
Common wheat	0.0000	0.0000	0.0000	0.0000	0.8909	0.0545	0.0000	0.0364	0.0000	0.0000	0.0000	0.0182
Fat Hen	0.0102	0.0000	0.0000	0.0000	0.0102	0.9694	0.0000	0.0000	0.0000	0.0000	0.0000	0.0102
Loose Silky-bent	0.0530	0.0000	0.0000	0.0000	0.0000	0.0199	0.9073	0.0000	0.0199	0.0000	0.0000	0.0000
Maize	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000	0.0000	0.0000
Scentless Mayweed	0.0000	0.0000	0.0000	0.0462	0.0000	0.0000	0.0154	0.0000	0.9231	0.0154	0.0000	0.0000
Shepherd's Purse	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000
Small-flowered Cranesbill	0.0000	0.0000	0.0084	0.0000	0.0000	0.0000	0.0000	0.0000	0.0084	0.0000	0.9832	0.0000
Sugar beet	0.0000	0.0000	0.0000	0.0000	0.0196	0.0294	0.0000	0.0000	0.0000	0.0000	0.0000	0.9510

100EpochCM(Fold2)

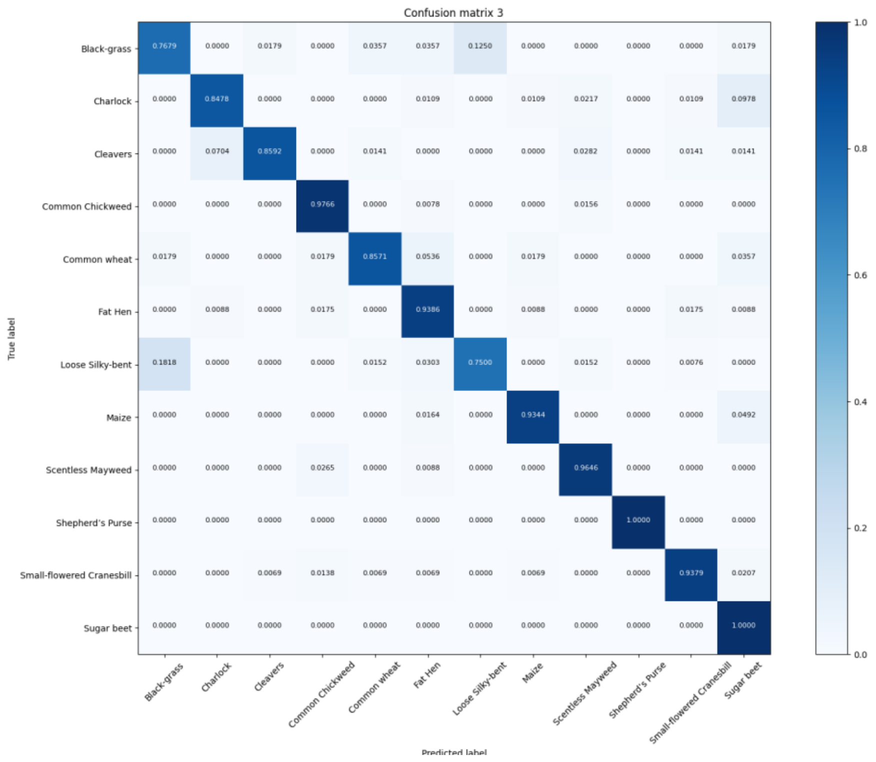
Image 45 - 100 Epochs Fold2



100EpochTVL(Fold3)



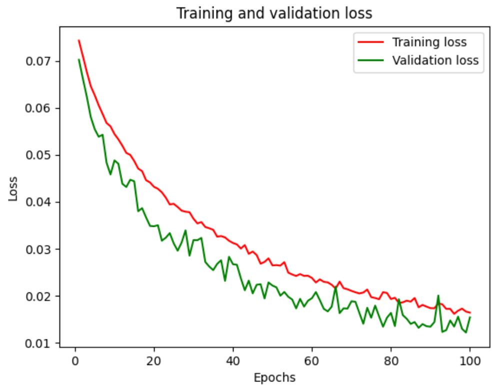
100EpochTVA(Fold3)



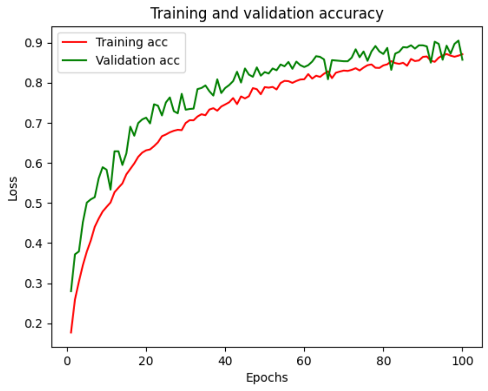
	Black-grass	Charlock	Cleavers	Common Chickweed	Common wheat	Fat Hen	Loose Silky-bent	Maize	Scentless Mayweed	Shepherd's Purse	Small-flowered Cranesbill	Sugar beet
Black-grass	0.7679	0.0000	0.0179	0.0000	0.0357	0.0357	0.1250	0.0000	0.0000	0.0000	0.0000	0.0179
Charlock	0.0000	0.8478	0.0000	0.0000	0.0000	0.0109	0.0000	0.0109	0.0217	0.0000	0.0109	0.0978
Cleavers	0.0000	0.0704	0.8592	0.0000	0.0141	0.0000	0.0000	0.0000	0.0282	0.0000	0.0141	0.0141
Common Chickweed	0.0000	0.0000	0.0000	0.9766	0.0000	0.0078	0.0000	0.0000	0.0156	0.0000	0.0000	0.0000
Common wheat	0.0179	0.0000	0.0000	0.0179	0.8571	0.0536	0.0000	0.0179	0.0000	0.0000	0.0000	0.0357
Fat Hen	0.0000	0.0088	0.0000	0.0175	0.0000	0.9386	0.0000	0.0088	0.0000	0.0000	0.0175	0.0088
Loose Silky-bent	0.1818	0.0000	0.0000	0.0000	0.0152	0.0303	0.7500	0.0000	0.0152	0.0000	0.0076	0.0000
Maize	0.0000	0.0000	0.0000	0.0000	0.0000	0.0164	0.0000	0.9344	0.0000	0.0000	0.0000	0.0492
Scentless Mayweed	0.0000	0.0000	0.0000	0.0265	0.0000	0.0088	0.0000	0.0000	0.9646	0.0000	0.0000	0.0000
Shepherd's Purse	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000
Small-flowered Cranesbill	0.0000	0.0000	0.0069	0.0138	0.0069	0.0069	0.0000	0.0069	0.0000	0.0000	0.9379	0.0207
Sugar beet	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000

100EpochCM(Fold3)

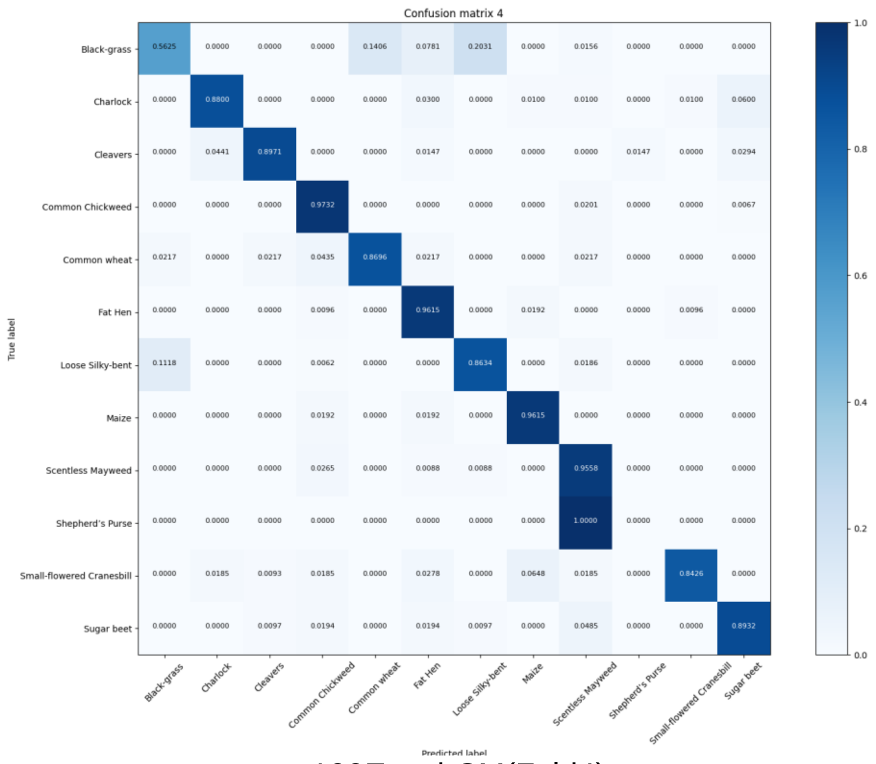
Image 46 - 100 Epochs Fold3



100EpochTVL(Fold4)



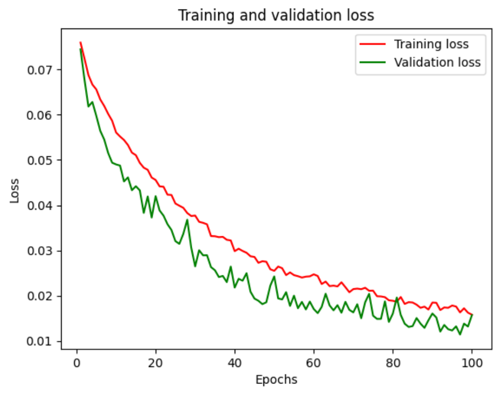
100EpochTVA(Fold4)



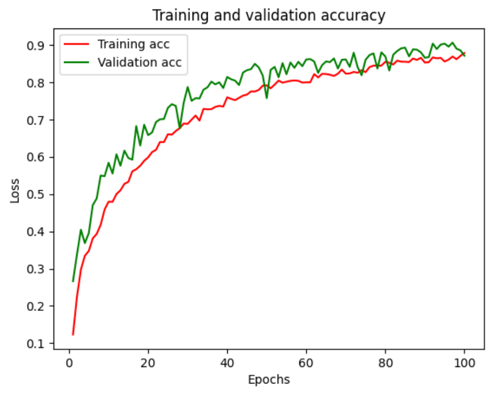
	Black-grass	Charlock	Cleavers	Common Chickweed	Common wheat	Fat Hen	Loose Silky-bent	Maize	Scentless Mayweed	Shepherd's Purse	Small-flowered Cranesbill	Sugar beet
Black-grass	0.5625	0.0000	0.0000	0.0000	0.1406	0.0781	0.2031	0.0000	0.0156	0.0000	0.0000	0.0000
Charlock	0.0000	0.8800	0.0000	0.0000	0.0000	0.0300	0.0000	0.0100	0.0100	0.0000	0.0100	0.0600
Cleavers	0.0000	0.0441	0.8971	0.0000	0.0000	0.0147	0.0000	0.0000	0.0000	0.0147	0.0000	0.0294
Common Chickweed	0.0000	0.0000	0.0000	0.9732	0.0000	0.0000	0.0000	0.0000	0.0201	0.0000	0.0000	0.0067
Common wheat	0.0217	0.0000	0.0217	0.0435	0.8696	0.0217	0.0000	0.0000	0.0217	0.0000	0.0000	0.0000
Fat Hen	0.0000	0.0000	0.0000	0.0096	0.0000	0.9615	0.0000	0.0192	0.0000	0.0000	0.0096	0.0000
Loose Silky-bent	0.1118	0.0000	0.0000	0.0062	0.0000	0.0000	0.8634	0.0000	0.0186	0.0000	0.0000	0.0000
Maize	0.0000	0.0000	0.0000	0.0192	0.0000	0.0192	0.0000	0.9615	0.0000	0.0000	0.0000	0.0000
Scentless Mayweed	0.0000	0.0000	0.0000	0.0265	0.0000	0.0088	0.0088	0.0000	0.9558	0.0000	0.0000	0.0000
Shepherd's Purse	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000	0.0000
Small-flowered Cranesbill	0.0000	0.0185	0.0093	0.0185	0.0000	0.0278	0.0000	0.0648	0.0185	0.0000	0.8426	0.0000
Sugar beet	0.0000	0.0000	0.0097	0.0194	0.0000	0.0194	0.0097	0.0000	0.0485	0.0000	0.0000	0.8932

100EpochCM(Fold4)

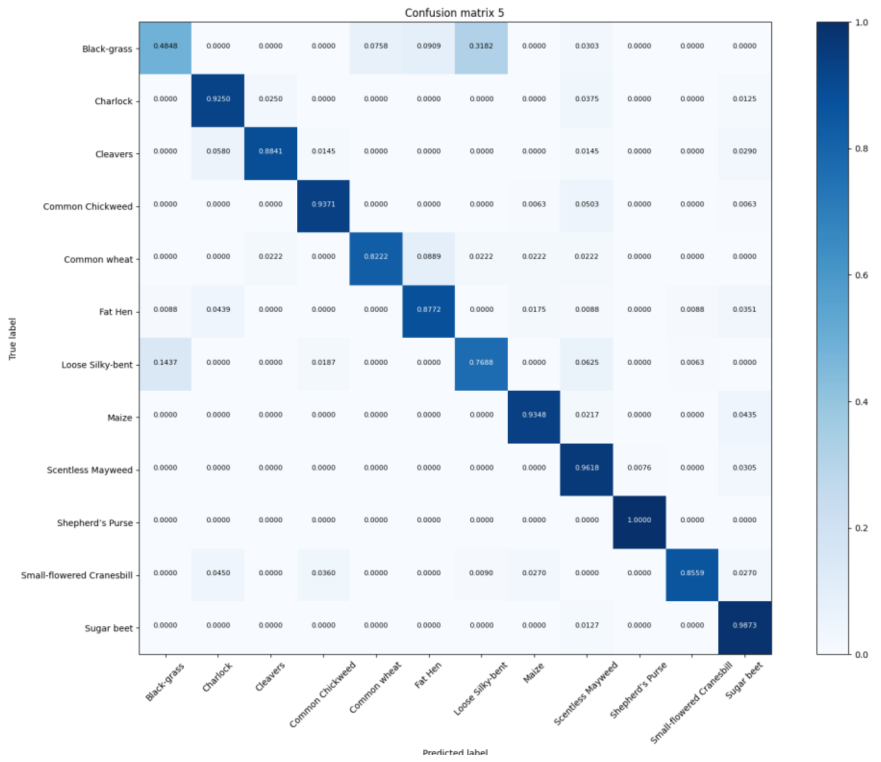
Image 47 - 100 Epochs Fold4



100EpochTVL(Fold5)



100EpochTVA(Fold5)



100EpochCM(Fold5)

Image 48 - 100 Epochs Fold5

Table 2 Results of 2-Fold

2Fold	Accuracy	Total loss
Fold 1	50.69	5.32
Fold 2	69.45	3.89
Average	60.07	4.61

Table 3 Results of 5 Fold

5Fold	Accuracy	Total loss
Fold 1	70.85	3.51
Fold 2	74.19	3.23
Fold 3	65.70	4.03
Fold 4	69.95	3.54
Fold 5	66.58	4.09
Average	69.45	3.68

Table 4 Results of 10 Fold

10 Fold	Accuracy	Total loss
Fold 1	66.43	4.11
Fold 2	73.10	3.18
Fold 3	77.98	2.61
Fold 4	76.53	3.09
Fold 5	72.56	3.07
Fold 6	68.59	3.60
Fold 7	71.66	3.32
Fold 8	74.01	3.13
Fold 9	70.76	3.51
Fold 10	75.59	2.96
Average	72.72	3.26

Table 5 Results of 20 Epochs

20 Epoch	Accuracy	Total loss
Fold 1	66.16	3.94
Fold 2	72.47	3.25
Fold 3	69.95	3.70
Fold 4	71.57	3.42
Fold 5	70.73	3.40
Average	70.18	3.54

Table 6 Results of 50 epochs

50 Epoch	Accuracy	Total loss
Fold 1	82.04	2.25
Fold 2	79.24	2.54
Fold 3	85.47	1.80
Fold 4	80.96	2.44
Fold 5	82.66	2.00
Average	82.07	2.21

Table 7 Results of 100 Epochs

100 Epoch	Accuracy	Total loss
Fold 1	89.44	1.32
Fold 2	91.79	1.05
Fold 3	90.52	1.23
Fold 4	85.74	1.54
Fold 5	87.17	1.57
Average	88.93	1.34

Table 8 Times of Each Configuration

Configuration	Time
2 Fold	7 min 22sec
5 Fold	22min 55sec
10 Fold	50min 20sec
20 Epochs	23min 17sec
50 Epochs	1hr 13min 31sec
100 Epochs	1hr 51min 37sec

Review of The Results

After multiple configurations we finally obtained a solution. First of all, when we tried the 2-Fold configuration we saw a pattern where the 1st Fold wasn't really successful. That's why it gave us such a low average score of 60%(Table 2), which is way below our average score of 69% and 72% of our 5 and 10-Fold respectively(Table 3 & Table 4). Even though 10 Folds gave us better results and more consistent with its runs, it was very time consuming. So that's why we stuck with 5 Folds for our Epoch Configuration. Even though we would lose accuracy and loss, we believed we could gain that back with the number of Epochs used for each Fold.

Our Epoch Configuration gave us the result we expected and with quite high marks. First, we started by using a baseline of 20 Epochs and based on the results up our number. 20 Epochs gave us an average of 70% Accuracy (0.7% better than our Fold Configuration) which is quite a way off our target of 85%(Table 5). After raising the number of Epochs to 50, we had an average score of 82%, which is close but not enough for our target (Table 6). We do have to mention that one of our folds did reach 85% but we couldn't accept a 1/5 of the results given. Then, we tried to double that number to 100 and check the results. 100 Epochs gave an average of 88.93% (Table 7), way above the target, with all Folds consistently having above 85% accuracy and some even getting above 90% (Folds 2 &3).

We also took into consideration the criteria of time running of our tests to check what was best for the future and what was the actual improvement (Table 8). Basically, when upping the number of Folds, looked like its more time consuming than upping the number of Epochs. Between the 5 and 10-Fold there is 2.2 times increase of time but from 50 to 100 Epochs there is a 1.5x increase of time.

Last mentioned, through all of our tests we had an inconsistency from our Black Grass being predicted as an Loose Silky Bent for reasons unknown to us, but its % kept getting lowered when we started reaching higher accuracy percentages.

Conclusion and Future Work

Even though we reached our target of 85-90% it would still be desirable to improve the accuracy of our results by implementing a more layered CNN Network or just by increasing the number of epochs in each fold. It should be mentioned that the code provided is also easily modified to accommodate classification of other photos, not just of plants. But in order to achieve that, the database provided needs to include a large amount of pictures. While testing our model with databases containing a very small number of pictures (3 for each plant), the trained model wasn't providing accurate results. Instead, it was providing rather random results. That's why we didn't include it in our results. Last but not least, code is simple and used in Python which some could later implement in an Mobile app.

References/Bibliography

https://www.kaggle.com/amarjeet007/visualize-cnn-with-keras
https://towardsdatascience.com/increase-the-accuracy-of-your-cnn-by-following-these-5-tips-i-learned-from-the-kaggle-community-27227ad39554
https://en.wikipedia.org/wiki/Convolutional_neural_network
https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53
https://www.ibm.com/cloud/learn/convolutional-neural-networks
https://cs231n.github.io/convolutional-networks/
https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/
https://www.analyticsvidhya.com/blog/2020/02/learn-image-classification-cnn-convolutional-neural-networks-3-datasets/
https://www.tensorflow.org/tutorials/images/cnn
https://www.tensorflow.org/tutorials/images/classification
https://morioh.com/p/43b6419364e7
https://vision.eng.au.dk/plant-seedlings-dataset/
2. Xinshao, W. Weed Seeds Classification Based on PCANet Deep Learning Baseline. In Proceedings of the
2015 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference APSIPA
ASC, Hong Kong, China, 16–19 December 2015; pp. 408–415.
3. Jamuna, K.S.; Karpagavalli, S.; Vijaya, M.S.; Revathi, P.; Gokilavani, S.; Madhiya, E. Classification of Seed
Cotton Yield Based on the Growth Stages of Cotton Crop Using Machine Learning Techniques. In Proceedings
of the ACE 2010–2010 International Conference on Advances in Computer Engineering, Bangalore, India,
20–21 June 2010; pp. 312–315.
4. Qiu, Z.; Chen, J.; Zhao, Y.; Zhu, S.; He, Y.; Zhang, C. Variety identification of single rice seed using
hyperspectral imaging combined with convolutional neural network. Appl. Sci. 2018, 8, 212.
5. Kiratiratanapruk, K.; Temniranrat, P.; Sinthupinyo, W.; Prempree, P.; Chaitavon, K.; Porntheeraphat, S.;
Prasertsak, A. Development of Paddy Rice Seed Classification Process using Machine Learning Techniques
for Automatic Grading Machine. J. Sens. 2020, 2020, 7041310.
6. Wu, N.; Zhang, Y.; Na, R.; Mi, C.; Zhu, S.; He, Y.; Zhang, C. Variety identification of oat seeds using

hyperspectral imaging: Investigating the representation ability of deep convolutional neural network.
RSC Adv. 2019, 9, 12635–12644.]
7. Luan, Z.; Li, C.; Ding, S.; Wei, M.; Yang, Y. Sunflower Seed Sorting Based on Convolutional Neural
Network. In Proceedings of the ICGIP 2019 Eleventh International Conference on Graphics and Image
Processing, Hangzhou, China, 12–14 October 2019; Pan, Z.,Wang, X., Eds.; SPIE: Bellingham,WA, USA, 2020;
Volume 11373, p. 129.
8. Rozman, Cˇ . Denis Stajnko Assessment of germination rate of the tomato seeds using image processing and
machine learning. Eur. J. Hortic. Sci. 2015, 80, 68–75.
9. Taylor, J.; Chiou, C.P.; Bond, L.J. A methodology for sorting haploid and diploid corn seed using terahertz
time domain spectroscopy and machine learning. AIP Conf. Proc. 2019, 2102.
10. Ali, A.; Qadri, S.; Mashwani, W.K.; Brahim Belhaouari, S.; Naeem, S.; Rafique, S.; Jamal, F.; Chesneau, C.;
Anam, S. Machine learning approach for the classification of corn seed using hybrid features. Int. J. Food Prop.
2020, 23, 1097–1111.
11. LeCun, Y.; Bengio, Y. Convolutional networks for images, speech, and time series. Handb. Brain Theory
Neural Netw. 1995, 3361, 1995.
12. Yonis Gulzar 1, Yasir Hamid 2, Arjumand Bano Soomro, Ali A. Alwan and Ludovic Journaux
A Convolution Neural Network-Based Seed
Classification System
7th December 2020