PLANT SEEDLING CLASSIFICATIONUSING IBM WATSON

CHAPTER 1

INTRODUCTION

Agriculture is vital for human survival and remains a major driver of several economies around the world; more so in underdeveloped and developing economies. With increasing demand for food and cash crops, due to a growing global population and the challenges posed by climate change, there is a pressing need to increase farm outputs while incurring minimal costs. One majorreason for reduction in crop yield is weed invasion on farmlands. Weeds generally have no useful value in terms of food, nutrition or medicine yet they have accelerated growth and parasitically compete with actual crops for nutrients and space. Inefficient processes such as hand weeding has led to significant losses and increasing costs due to manual labor. Plants continue to serve as a source of food and oxygen for all life on earth. In continents like Africa, where agriculture is predominant, proper automation of the farming process would help optimize crop yield and ensure continuous productivity and sustainability. In recent times, the state of agriculture and the amount of work people need to put in to check if plants/food is growing correctly is phenomenal, because itis 2019 and workers still need to organize and recognize the difference between different plants and weeds. People who are working in the agriculture field still have to have the ability to sort and recognize different plants and weeds, which does take a lot of time and a lot of effort in the long term.

This is where Artificial Intelligence can actually help benefit those workers, as the time and energy to identify plant seedlings will be much shortened. The ability to do so effectively can mean better crop yields and in the long term will result in better care for the environment. As by identifying the difference among the plants and weeds in a timely manner where it is highly accurate can positively impact agriculture.

To better understand different methods of deep learning and validate previous work, self-built CNN, VGG, ResNet, DenseNet and MobileNet are tested. Results shows that all methods can give great results on a 12 species plant seedling classification with minor modification, and MobileNet achieves a great balance in accuracy and time consumption when deployed.

CHAPTER 2

RELATED WORK

Significant works have been done for plant image classification tasks, including classification between maize plants and weeds with 44580 segmented images at early growth with obtained training accuracy of 97.23%. For plant disease classification in wild conditions, some researchers focus on automatic disease detection system like Johannes. The system achieved excellent results for early recognition of three wheat diseases. An analysis was carried out using seven handheld devices of 3500 images across Spain and Germany.

For plant classification, a research using CNN to identify plant by extracting vein morphological patterns, can get an accuracy of 95%. For plant seedling classification, many researchers are using dataset provided by Aarhus University, Department of Engineering Signal Processing Group. Benefited from Kaggle which host the dataset for competition, plenty of jobs have achieved great results.

As we have mentioned earlier, Ashcar got the best results using VGG with an accuracy of 99.48%. Deep learning’s success in different kinds of plant classification testifies its versatility and expandability, not only the more recent methods like VGG can achieve good results, pure CNN can also performs good in some kind of task with finely tuned parameters.

CHAPTER 3

DATASET

• Data Exploration

The dataset of this project can be downloaded from Kaggle Competition which contains images of about 960 unique plants belonging to 12 species at several growth stages. And also the database have been recorded Aarhuas University Flakkebjerg Research station in a collaboration between Southern Denmark and Aarhus University. This dataset contains 5,539 images of crop and weed seedlings. The images are grouped into 12 classes as shown in the above pictures. Each class contains rgb images that show plants at different growth stages. The images are in various sizes and are in png format.So in our project data preprocess, we will change the size of picture and normalize to fit the model and convert the pixels to matrice.

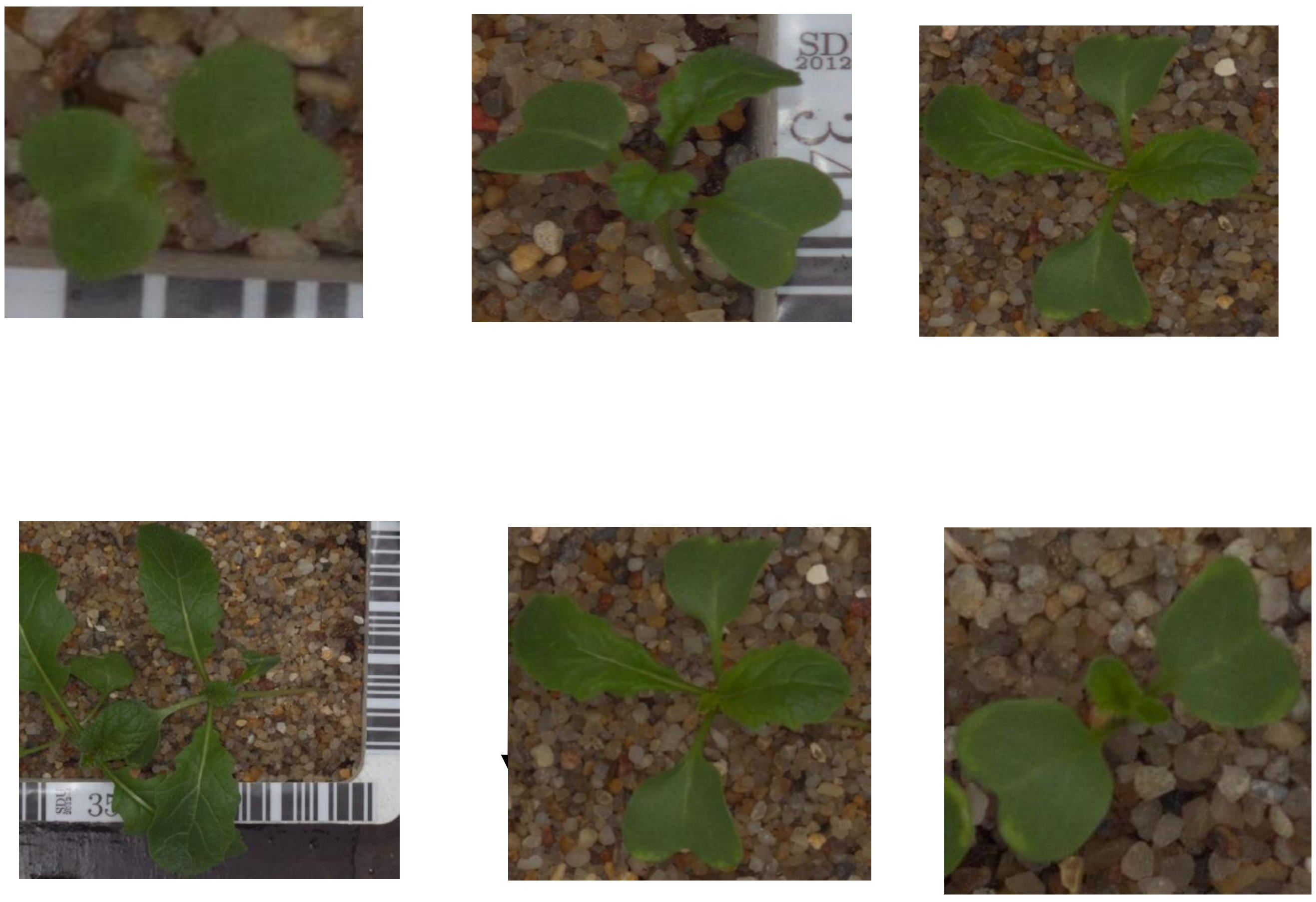


Fig. 1: data samples

• **Data Preprocessing**

1. Resizing images

Images in the data set are not always the same, so we have to resize the size of images to 224*224 to feed it to the neural network.

2. Create mask for the images

Create masks means that we return the matrix with shape height and width of the original images, and in this matrix there are only 0 and 1 values. The 1 values define the interesting part of the original image. We create this mask using HSV of the image. The HSV color-space is suitable for color detection because with the Hue we can define the color and the saturation and value will define “different kinds” of the color. (For example it will detect the red, darker red, lighter red too). We cannot do this with the original RGB color space.

➢ RGB

• The RGB is an additive color space.
• Creating any color by mixing the three primitive additives Red, Green and Blue.
• A pixel with 3 channels contains information of (red, green, blue) light with values between 0 and 255.
• Given all values equal to 0 maps to black, all values 255 maps to white.

➢ HSV

• Colors of each hue follow the radius on the circle. It’s values have an angular dimension starting with the red primary at 0°, passing the green primary at 120° and the blue primary at 240° and finally
ending up with red again at 360°.
• The vertical axis describes the gray scale ranging from black (0) to white (1).

3. Morphological operations

one of most common morphological operation is closing : closing used to close the small halls in the images. This figure below illustrate the image before and after applying closing:

4. Sharpening

Sharpening an image increases the contrast between bright and dark regions to bring out features. It just looks like make the texture of the image more visible. Image before and after sharpening:

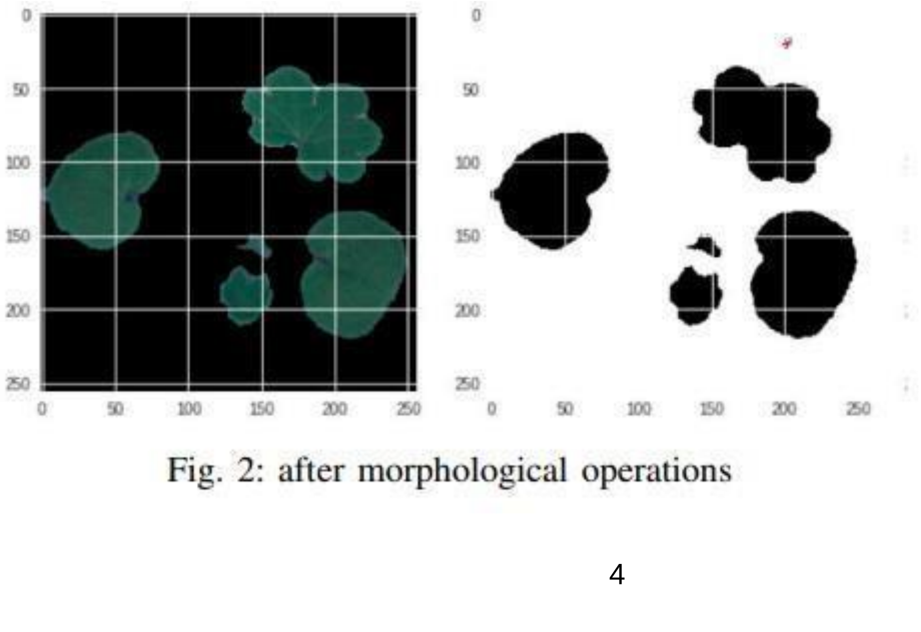


Fig. 2: after morphological operations

5. Label encoding

Using **LabelBinarizer** Binarize labels in a one-vs-all fashion Input: the label of image and the **output** is vector represent the class in binary form

6. Split data into training and testing set

Splitting 70% for training and 30% for testing. Meanwhile, splitting testing data half for testing and half for validation. We use cross-validation to prevent the overfitting and check if it will happen.

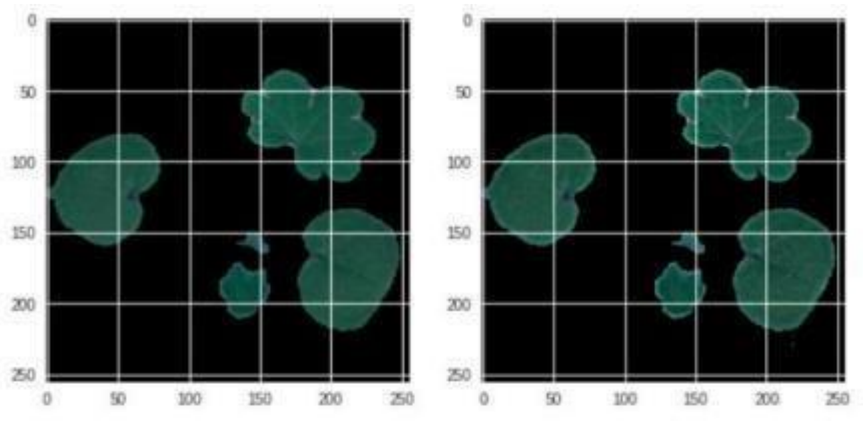


Fig. 3: after Sharpening

7. Data augmentation

The data augmentation will be applied when the distribution of our data is not quite uniform or the number of samples is not very large to feed the model. Through crop,spinning,color or something else to generate more images based on the original data,the sample image rotation, size, width, height, horizontal flip, vertical flip are randomly transformed and then the size of the training data could be larger than before.

CHAPTER4
METHODS

In this project, we use several model (classifiers): self-built CNN model, ResNet model, MobileNet model. The CNN model is a state-of-the-art algorithm of most image processing tasks, which is classification for different kinds of plant seeding in this task. CNN’s are concurrently employed in agriculture mainly, for recognition and classification tasks, and have been proven to provide superior results. It needs a large amount of training data compared to other approaches; and this dataset we have been provided are already sufficient enough to fit this criteria.

4.1 <u>Convolutional Neural Network</u>

The Convolutional Neural Network (CNN) is a deep learning algorithm and consists of an input layer, hidden layers, and an output layer.

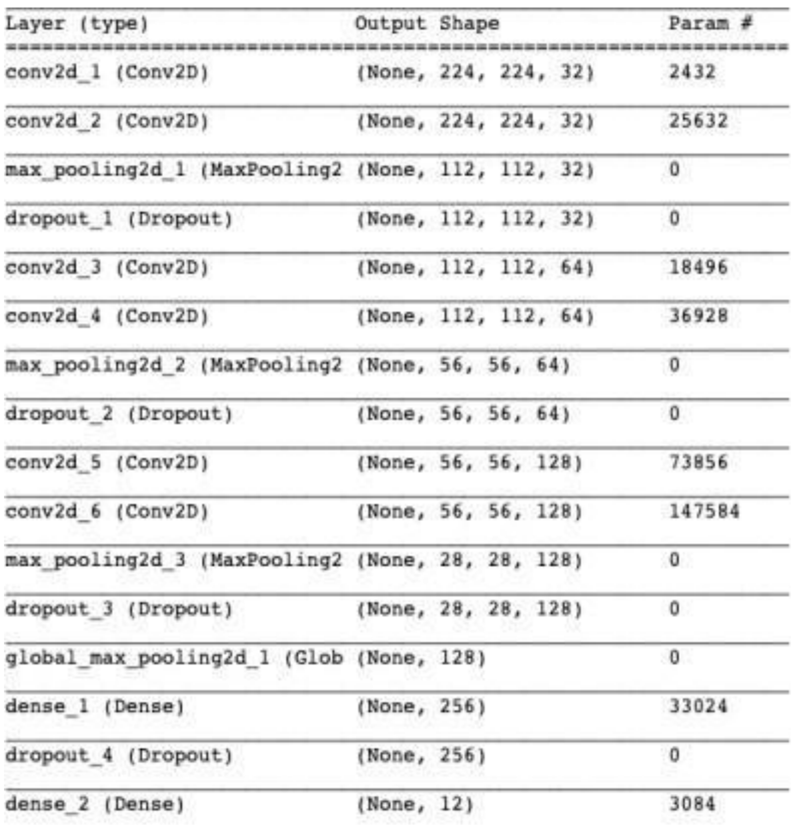
The original seedling images are all equally resized to 224x224 pixels (The size of images can be resized empirically so as to get more satisfactory performance and fit to the input layer).And the hidden layers consist of a lot of layers which will be illustrated in the following table: Following are the layers which are used to construct the CNN model:

- **CONV layer**

The CONV layer will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume. In this project, we choose 32, 64,128 filters for the Conv layers.

- **POOL layer**

Pool layer performs down sampling operation along the spatial dimension(width, height), outputting a reduced volume than the Previous layer. These are used to reduce computational cost and to some extent also reduce over fitting.



```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 224, 224, 32)      2432
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 224, 224, 32)      25632
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 112, 112, 32)      0
_________________________________________________________________
dropout_1 (Dropout)          (None, 112, 112, 32)      0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 112, 112, 64)      18496
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 112, 112, 64)      36928
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 56, 56, 64)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 56, 56, 64)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 56, 56, 128)       73856
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 56, 56, 128)       147584
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 28, 28, 128)       0
_________________________________________________________________
dropout_3 (Dropout)          (None, 28, 28, 128)       0
_________________________________________________________________
global_max_pooling2d_1 (Glob (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               33024
_________________________________________________________________
dropout_4 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 12)                3084
```

Fig. 4: summary of layers in CNN.

• Dense layer

Dense layer is also called fully-connected layer, each neuron will be connected to all the neurons of the previous layer. Each of the nodes of dense layer outputs a score corresponding to a class score.

• Dropout layer

Dropout layer is used as a method of regularization of combat over-fitting of the training set. It 'drops' randomly neurons(setting their weights to zero), resulting in a simpler version of the CNN for each iteration and hence giving the model a hard time to over fit.

4.2 Transfer Learning

Transfer learning is a machine learning technique where a model trained on one task is repurposed on a second related task. In transfer learning, we first train a base network on a base dataset and task, and then we repurpose the learned features, or transfer them, to a second target network to be trained on a target dataset and task. This process will tend to work if the features are general, meaning suitable to both base and target tasks, instead of specific to the base task. There are two common approaches as below.

- Develop Model Approach
- Pre-trained Model Approach

In our project, we choose the second approach and use four pre-trained models: ResNet, DenseNet, VGG, MobileNet. Below we will briefly introduce these four models.

• VGG

VGG mainly contains 16 or 19 layers, including fully connected layers, convolutional layers, max -pool layers. Its kernel size is 3 and max-pool size is 2.

- **ResNet**

ResNet is composed of a series of residual blocks, which can be divided into two parts, direct mapping part and residual part. Its general expression is :

$$xl + 1 = xl + F(xl)$$

- **DenseNet**

DenseNet is composed of a series of dense blocks. For each layer, the feature maps of all previous layers are directly used as input for this layer. ResNet adds features directly through the ”Summation” operation, impeding the information flow in the network to a certain extent. DenseNet combines feature maps through concatenate operations, and each layer is related to other layers, which maximizes the flow of information. Its general expression is :

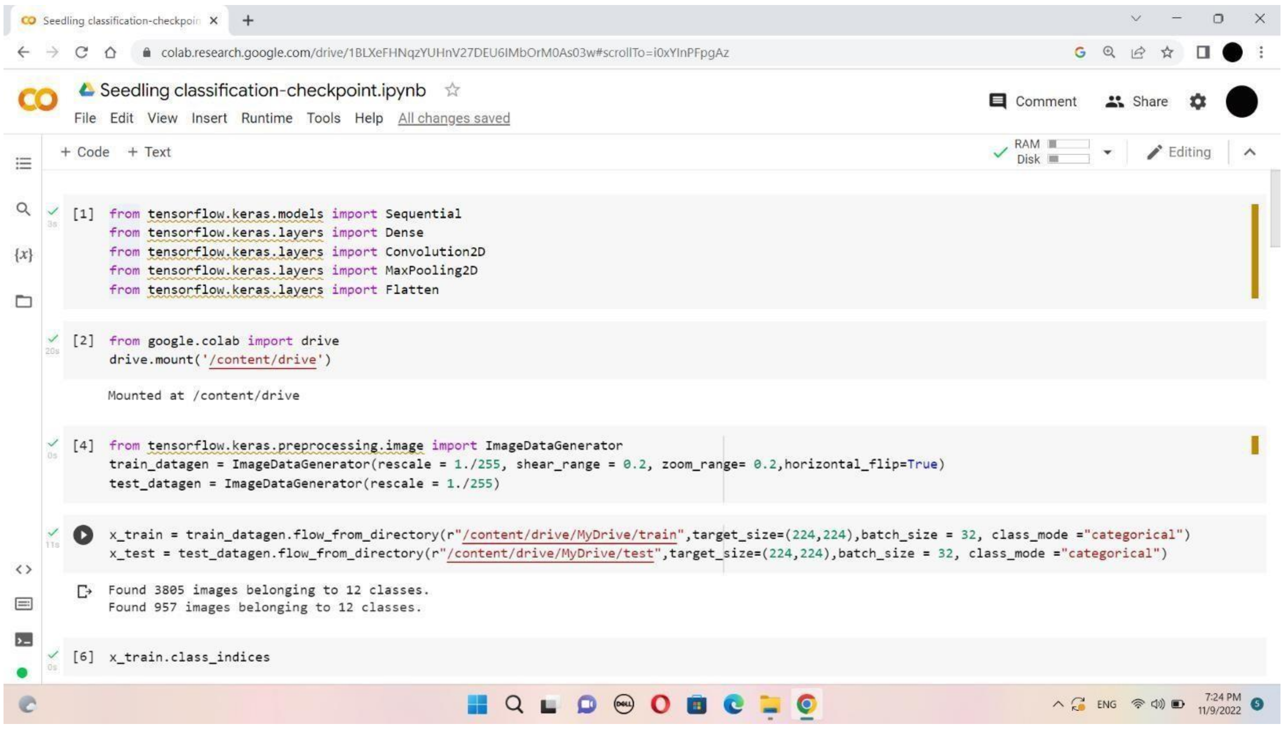
$$xl +1 = F(x0, x1...xl)$$

- **MobileNet**

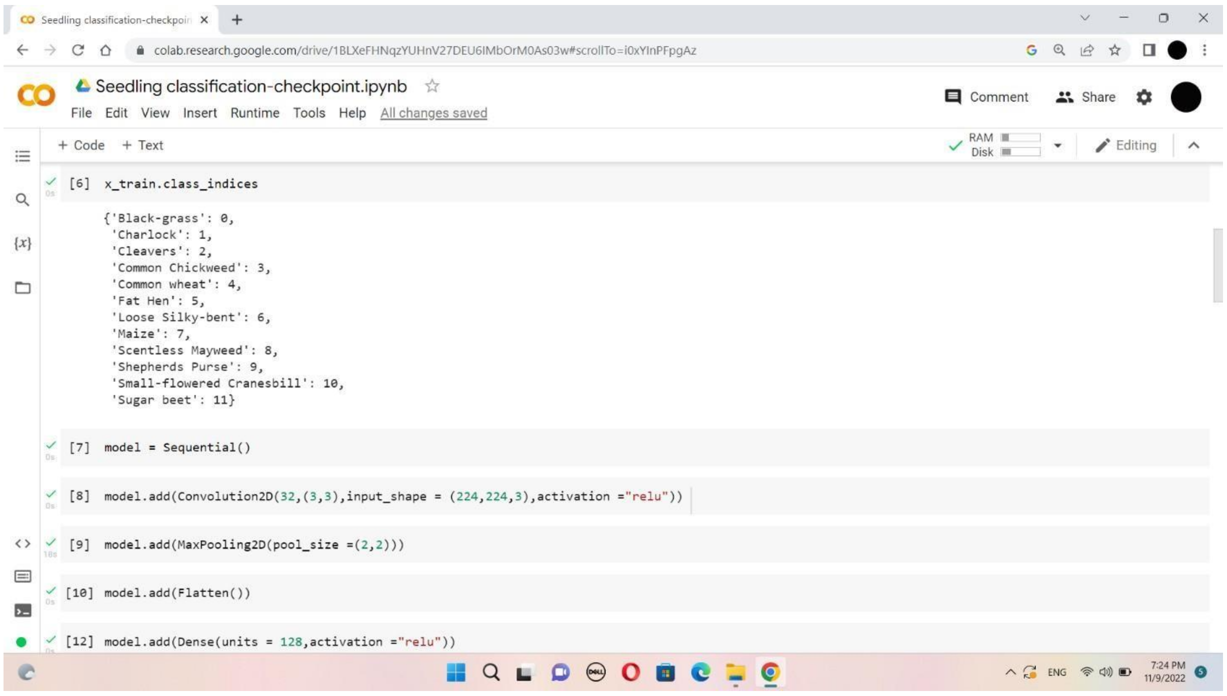
The MobileNet model is based on depth wise separable convolutions which factorize a standard convolution into a depth wise convolution and a pointwise convolution.

CHAPTER 5

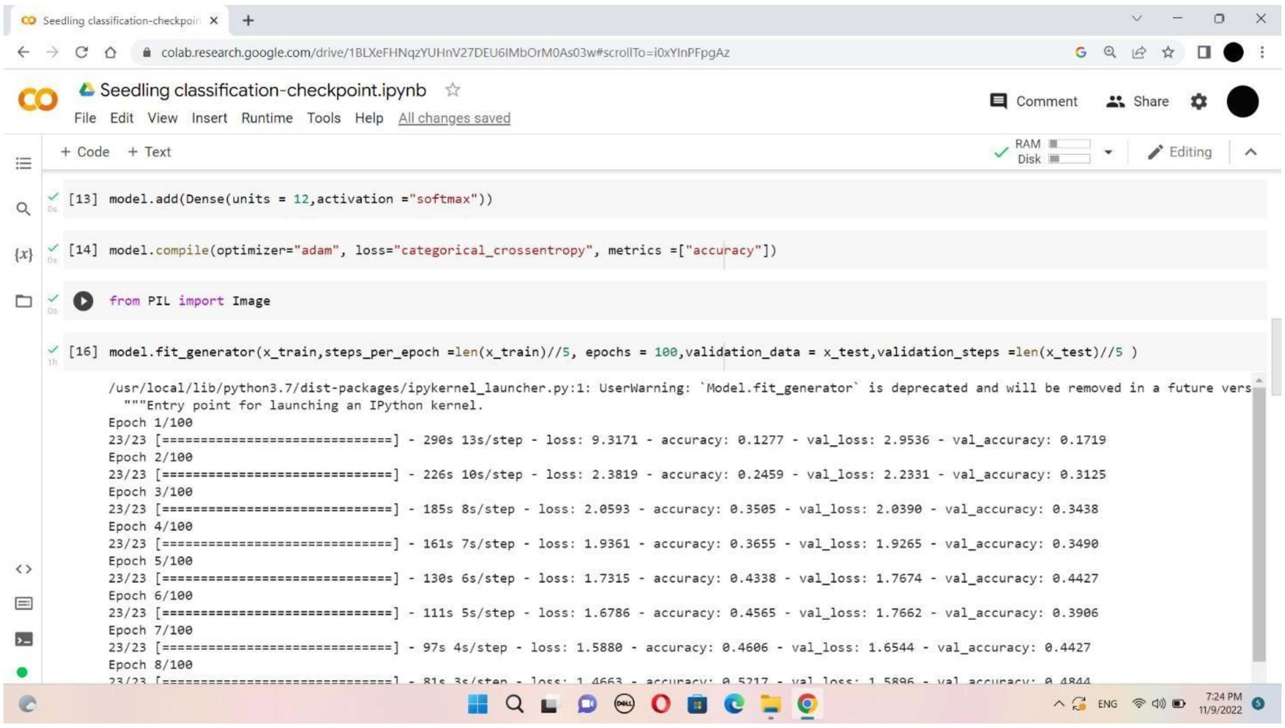
RESULT


Seedling classification-checkpoint.ipynb
colab.research.google.com/drive/1BLXeFHNqzYUHnV27DEU6IMbOrM0As03w#scrollTo=i0xYInPFpgAz
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Editing

```
[1] from tensorflow.keras.models import Sequential
    from tensorflow.keras.layers import Dense
    from tensorflow.keras.layers import Convolution2D
    from tensorflow.keras.layers import MaxPooling2D
    from tensorflow.keras.layers import Flatten

[2] from google.colab import drive
    drive.mount('/content/drive')

    Mounted at /content/drive

[4] from tensorflow.keras.preprocessing.image import ImageDataGenerator
    train_datagen = ImageDataGenerator(rescale = 1./255, shear_range = 0.2, zoom_range= 0.2,horizontal_flip=True)
    test_datagen = ImageDataGenerator(rescale = 1./255)

    x_train = train_datagen.flow_from_directory(r"/content/drive/MyDrive/train",target_size=(224,224),batch_size = 32, class_mode ="categorical")
    x_test = test_datagen.flow_from_directory(r"/content/drive/MyDrive/test",target_size=(224,224),batch_size = 32, class_mode ="categorical")

    Found 3805 images belonging to 12 classes.
    Found 957 images belonging to 12 classes.

[6] x_train.class_indices
```

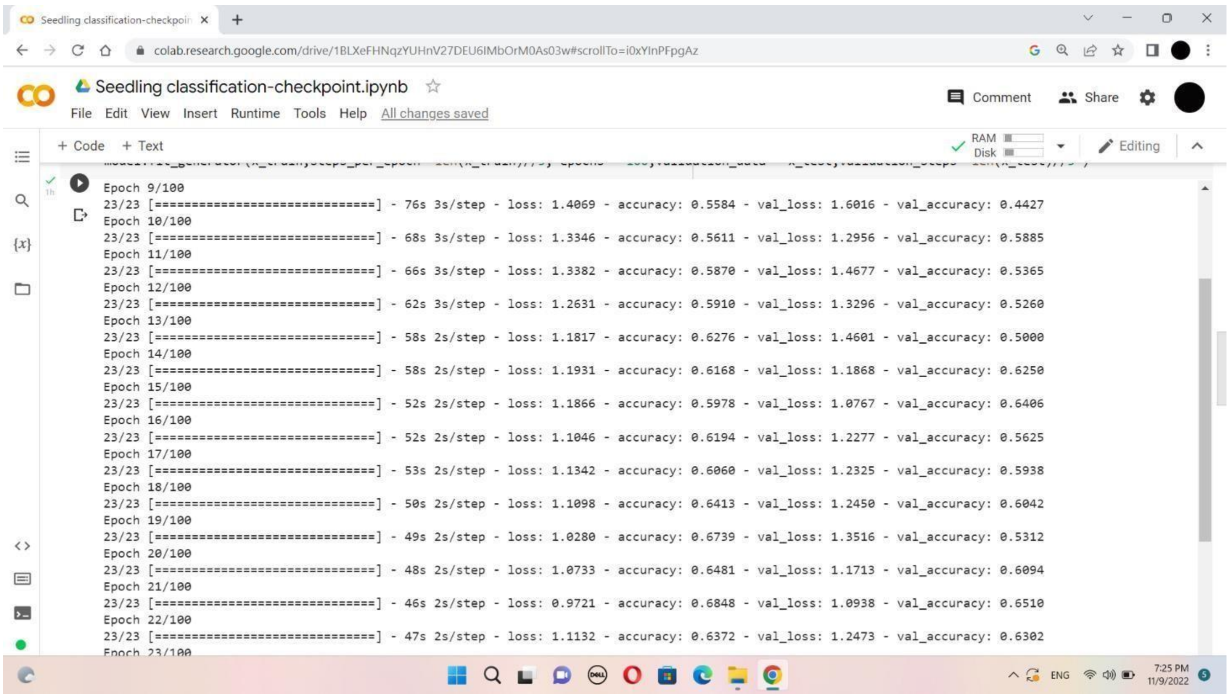

Seedling classification-checkpoint.ipynb
colab.research.google.com/drive/1BLXeFHNqzYUHnV27DEU6IMbOrM0As03w#scrollTo=i0xYInPFpgAz
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Editing

```
[6] x_train.class_indices

    {'Black-grass': 0,
     'Charlock': 1,
     'Cleavers': 2,
     'Common Chickweed': 3,
     'Common wheat': 4,
     'Fat Hen': 5,
     'Loose Silky-bent': 6,
     'Maize': 7,
     'Scentless Mayweed': 8,
     'Shepherds Purse': 9,
     'Small-flowered Cranesbill': 10,
     'Sugar beet': 11}

[7] model = Sequential()

[8] model.add(Convolution2D(32,(3,3),input_shape = (224,224,3),activation ="relu"))

[9] model.add(MaxPooling2D(pool_size =(2,2)))

[10] model.add(Flatten())

[12] model.add(Dense(units = 128,activation ="relu"))
```



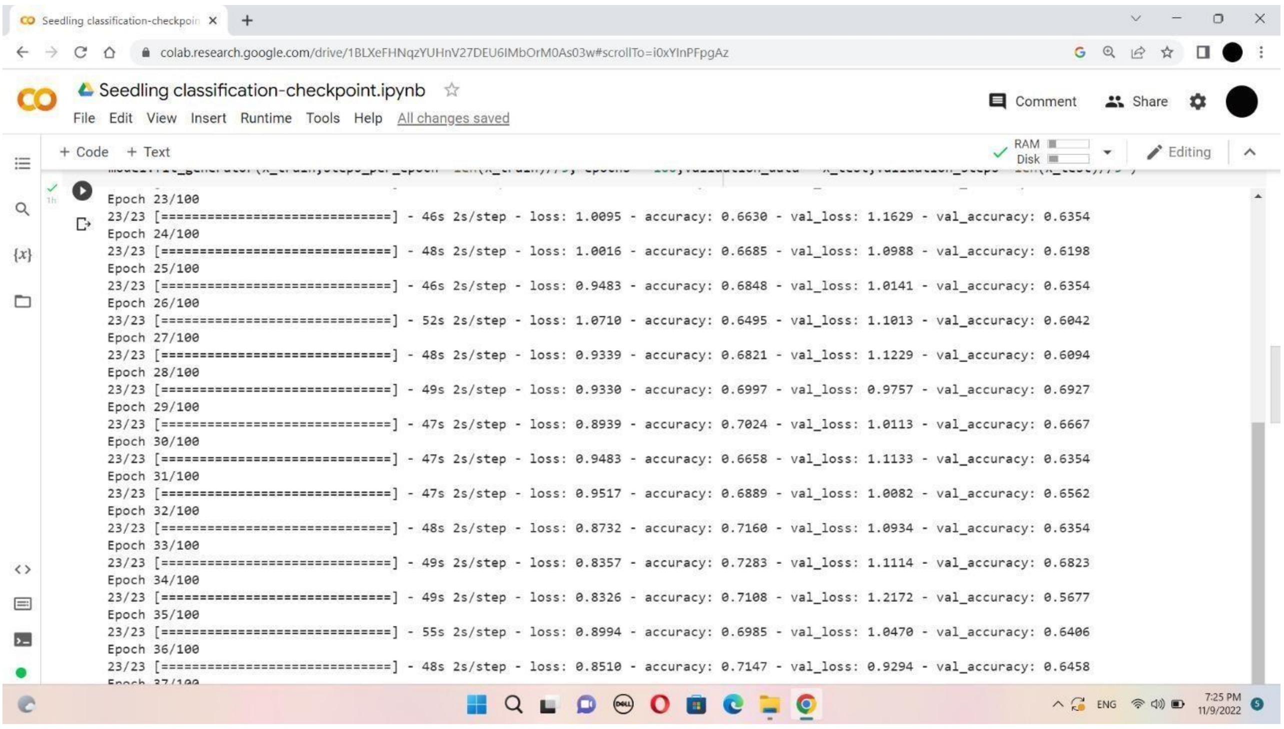
Seedling classification-checkpoint.ipynb
colab.research.google.com/drive/1BLXeFHNqzYUHnV27DEU6IMbOrM0As03w#scrollTo=i0xYInPFpgAz
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Editing
7:24 PM
11/9/2022

```
[13] model.add(Dense(units = 12,activation ="softmax"))

[14] model.compile(optimizer="adam", loss="categorical_crossentropy", metrics =["accuracy"])

     from PIL import Image

[16] model.fit_generator(x_train,steps_per_epoch =len(x_train)//5, epochs = 100,validation_data = x_test,validation_steps =len(x_test)//5 )

     /usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future vers
       """Entry point for launching an IPython kernel.
     Epoch 1/100
     23/23 [==============================] - 290s 13s/step - loss: 9.3171 - accuracy: 0.1277 - val_loss: 2.9536 - val_accuracy: 0.1719
     Epoch 2/100
     23/23 [==============================] - 226s 10s/step - loss: 2.3819 - accuracy: 0.2459 - val_loss: 2.2331 - val_accuracy: 0.3125
     Epoch 3/100
     23/23 [==============================] - 185s 8s/step - loss: 2.0593 - accuracy: 0.3505 - val_loss: 2.0390 - val_accuracy: 0.3438
     Epoch 4/100
     23/23 [==============================] - 161s 7s/step - loss: 1.9361 - accuracy: 0.3655 - val_loss: 1.9265 - val_accuracy: 0.3490
     Epoch 5/100
     23/23 [==============================] - 130s 6s/step - loss: 1.7315 - accuracy: 0.4338 - val_loss: 1.7674 - val_accuracy: 0.4427
     Epoch 6/100
     23/23 [==============================] - 111s 5s/step - loss: 1.6786 - accuracy: 0.4565 - val_loss: 1.7662 - val_accuracy: 0.3906
     Epoch 7/100
     23/23 [==============================] - 97s 4s/step - loss: 1.5880 - accuracy: 0.4606 - val_loss: 1.6544 - val_accuracy: 0.4427
     Epoch 8/100
     23/23 [==============================] - 81s 3s/step - loss: 1.4663 - accuracy: 0.5217 - val_loss: 1.5896 - val_accuracy: 0.4844
```

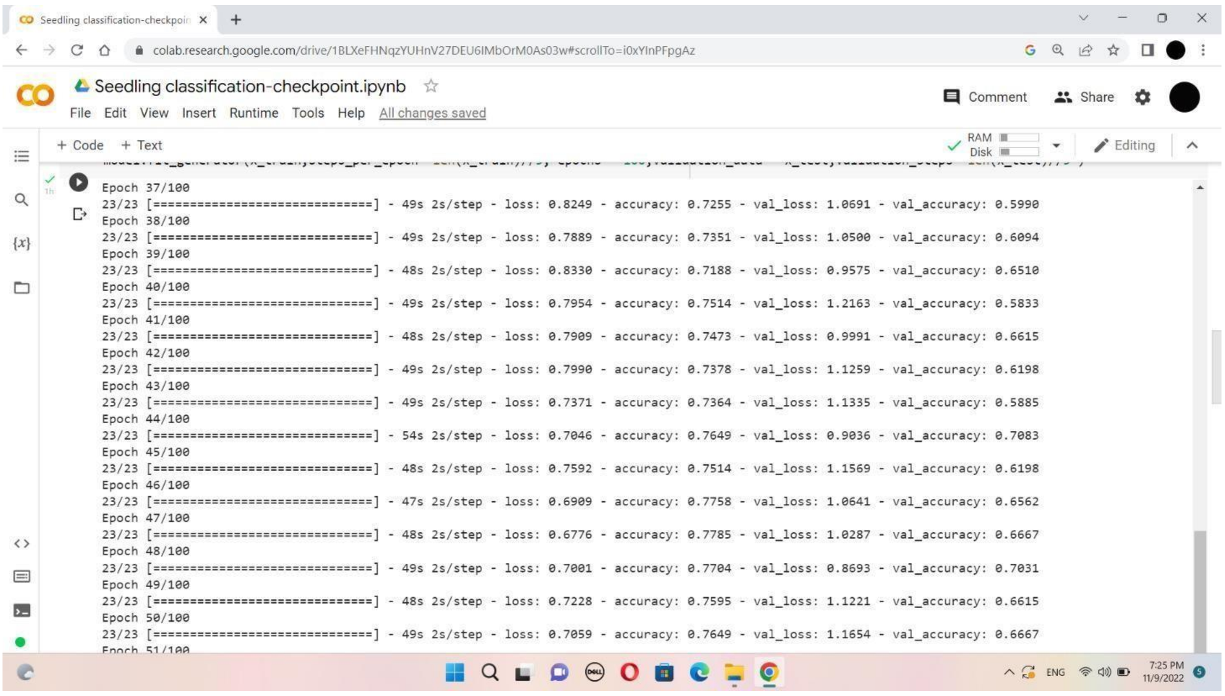


Seedling classification-checkpoint.ipynb
colab.research.google.com/drive/1BLXeFHNqzYUHnV27DEU6IMbOrM0As03w#scrollTo=i0xYInPFpgAz
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Editing
7:25 PM
11/9/2022

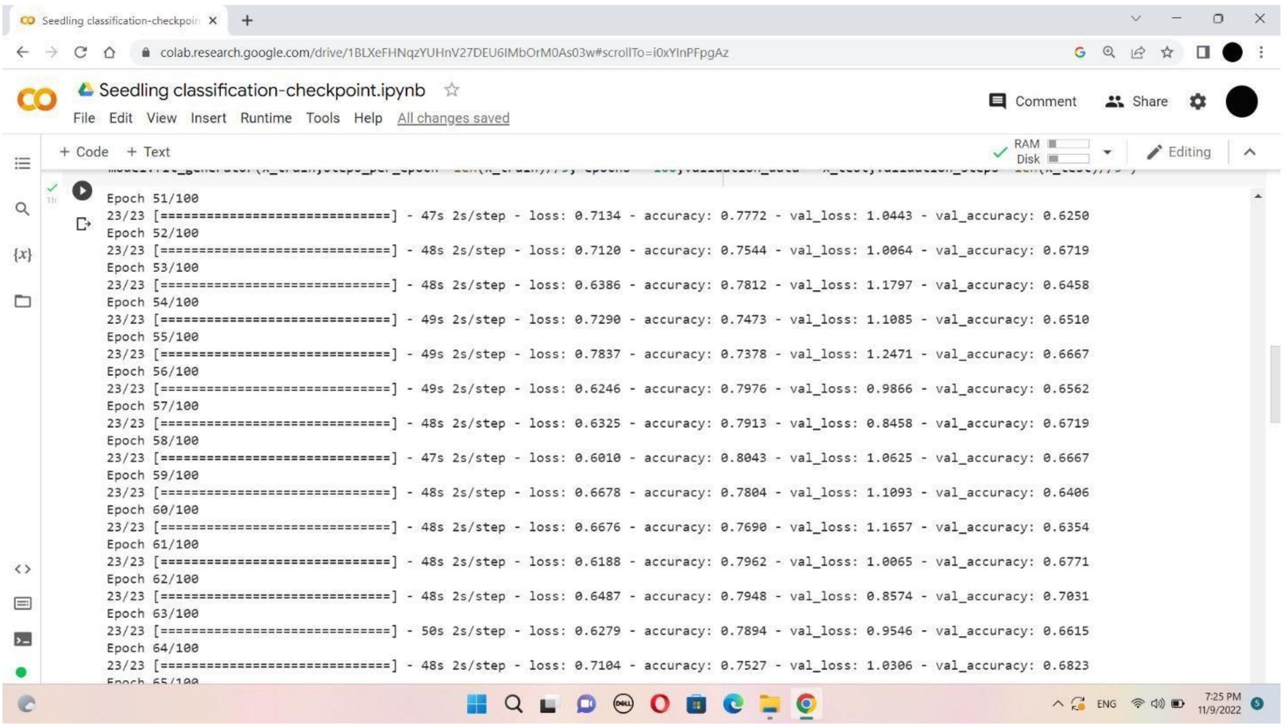
```
Epoch 9/100
23/23 [==============================] - 76s 3s/step - loss: 1.4069 - accuracy: 0.5584 - val_loss: 1.6016 - val_accuracy: 0.4427
Epoch 10/100
23/23 [==============================] - 68s 3s/step - loss: 1.3346 - accuracy: 0.5611 - val_loss: 1.2956 - val_accuracy: 0.5885
Epoch 11/100
23/23 [==============================] - 66s 3s/step - loss: 1.3382 - accuracy: 0.5870 - val_loss: 1.4677 - val_accuracy: 0.5365
Epoch 12/100
23/23 [==============================] - 62s 3s/step - loss: 1.2631 - accuracy: 0.5910 - val_loss: 1.3296 - val_accuracy: 0.5260
Epoch 13/100
23/23 [==============================] - 58s 2s/step - loss: 1.1817 - accuracy: 0.6276 - val_loss: 1.4601 - val_accuracy: 0.5000
Epoch 14/100
23/23 [==============================] - 58s 2s/step - loss: 1.1931 - accuracy: 0.6168 - val_loss: 1.1868 - val_accuracy: 0.6250
Epoch 15/100
23/23 [==============================] - 52s 2s/step - loss: 1.1866 - accuracy: 0.5978 - val_loss: 1.0767 - val_accuracy: 0.6406
Epoch 16/100
23/23 [==============================] - 52s 2s/step - loss: 1.1046 - accuracy: 0.6194 - val_loss: 1.2277 - val_accuracy: 0.5625
Epoch 17/100
23/23 [==============================] - 53s 2s/step - loss: 1.1342 - accuracy: 0.6060 - val_loss: 1.2325 - val_accuracy: 0.5938
Epoch 18/100
23/23 [==============================] - 50s 2s/step - loss: 1.1098 - accuracy: 0.6413 - val_loss: 1.2450 - val_accuracy: 0.6042
Epoch 19/100
23/23 [==============================] - 49s 2s/step - loss: 1.0280 - accuracy: 0.6739 - val_loss: 1.3516 - val_accuracy: 0.5312
Epoch 20/100
23/23 [==============================] - 48s 2s/step - loss: 1.0733 - accuracy: 0.6481 - val_loss: 1.1713 - val_accuracy: 0.6094
Epoch 21/100
23/23 [==============================] - 46s 2s/step - loss: 0.9721 - accuracy: 0.6848 - val_loss: 1.0938 - val_accuracy: 0.6510
Epoch 22/100
23/23 [==============================] - 47s 2s/step - loss: 1.1132 - accuracy: 0.6372 - val_loss: 1.2473 - val_accuracy: 0.6302
Epoch 23/100
```


Seedling classification-checkpoi
colab.research.google.com/drive/1BLXeFHNqzYUHnV27DEU6IMbOrM0As03w#scrollTo=i0xYInPFpgAz
Seedling classification-checkpoint.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Editing
7:25 PM
11/9/2022

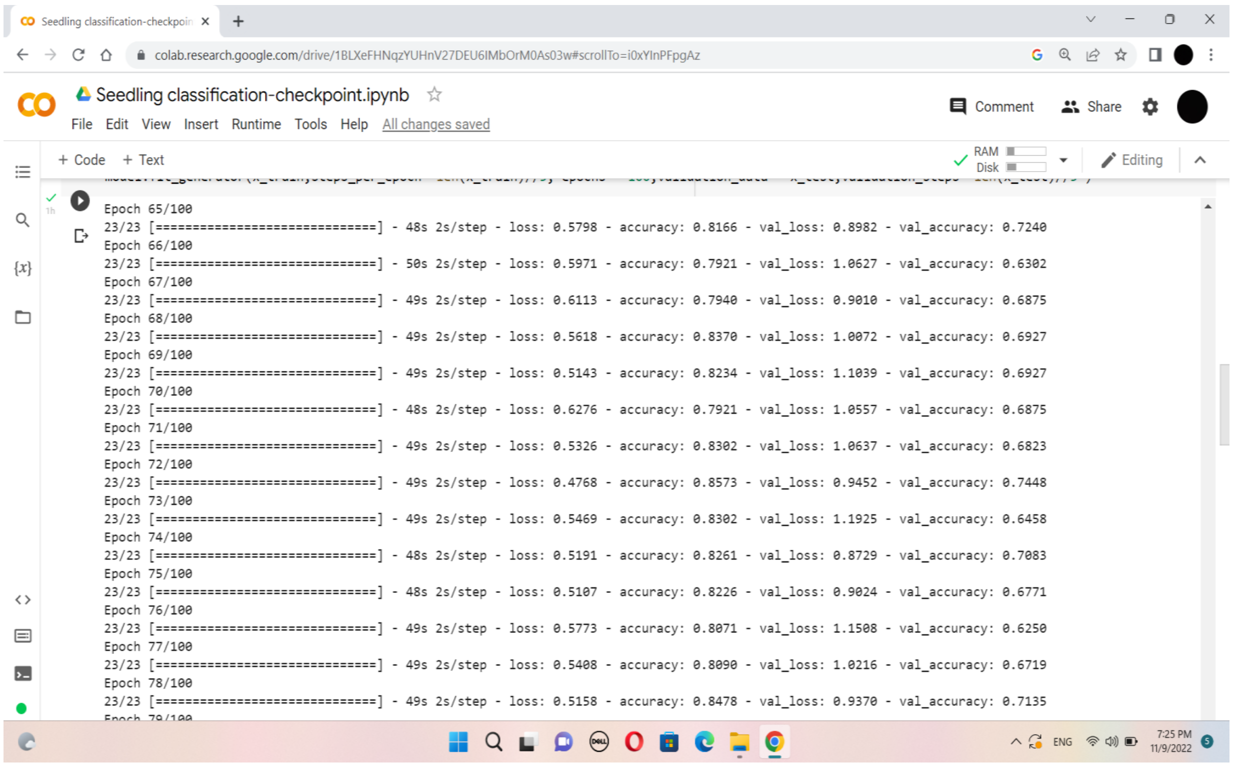
```
Epoch 23/100
23/23 [==============================] - 46s 2s/step - loss: 1.0095 - accuracy: 0.6630 - val_loss: 1.1629 - val_accuracy: 0.6354
Epoch 24/100
23/23 [==============================] - 48s 2s/step - loss: 1.0016 - accuracy: 0.6685 - val_loss: 1.0988 - val_accuracy: 0.6198
Epoch 25/100
23/23 [==============================] - 46s 2s/step - loss: 0.9483 - accuracy: 0.6848 - val_loss: 1.0141 - val_accuracy: 0.6354
Epoch 26/100
23/23 [==============================] - 52s 2s/step - loss: 1.0710 - accuracy: 0.6495 - val_loss: 1.1013 - val_accuracy: 0.6042
Epoch 27/100
23/23 [==============================] - 48s 2s/step - loss: 0.9339 - accuracy: 0.6821 - val_loss: 1.1229 - val_accuracy: 0.6094
Epoch 28/100
23/23 [==============================] - 49s 2s/step - loss: 0.9330 - accuracy: 0.6997 - val_loss: 0.9757 - val_accuracy: 0.6927
Epoch 29/100
23/23 [==============================] - 47s 2s/step - loss: 0.8939 - accuracy: 0.7024 - val_loss: 1.0113 - val_accuracy: 0.6667
Epoch 30/100
23/23 [==============================] - 47s 2s/step - loss: 0.9483 - accuracy: 0.6658 - val_loss: 1.1133 - val_accuracy: 0.6354
Epoch 31/100
23/23 [==============================] - 47s 2s/step - loss: 0.9517 - accuracy: 0.6889 - val_loss: 1.0082 - val_accuracy: 0.6562
Epoch 32/100
23/23 [==============================] - 48s 2s/step - loss: 0.8732 - accuracy: 0.7160 - val_loss: 1.0934 - val_accuracy: 0.6354
Epoch 33/100
23/23 [==============================] - 49s 2s/step - loss: 0.8357 - accuracy: 0.7283 - val_loss: 1.1114 - val_accuracy: 0.6823
Epoch 34/100
23/23 [==============================] - 49s 2s/step - loss: 0.8326 - accuracy: 0.7108 - val_loss: 1.2172 - val_accuracy: 0.5677
Epoch 35/100
23/23 [==============================] - 55s 2s/step - loss: 0.8994 - accuracy: 0.6985 - val_loss: 1.0470 - val_accuracy: 0.6406
Epoch 36/100
23/23 [==============================] - 48s 2s/step - loss: 0.8510 - accuracy: 0.7147 - val_loss: 0.9294 - val_accuracy: 0.6458
```


Seedling classification-checkpoi
colab.research.google.com/drive/1BLXeFHNqzYUHnV27DEU6IMbOrM0As03w#scrollTo=i0xYInPFpgAz
Seedling classification-checkpoint.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Editing
7:25 PM
11/9/2022

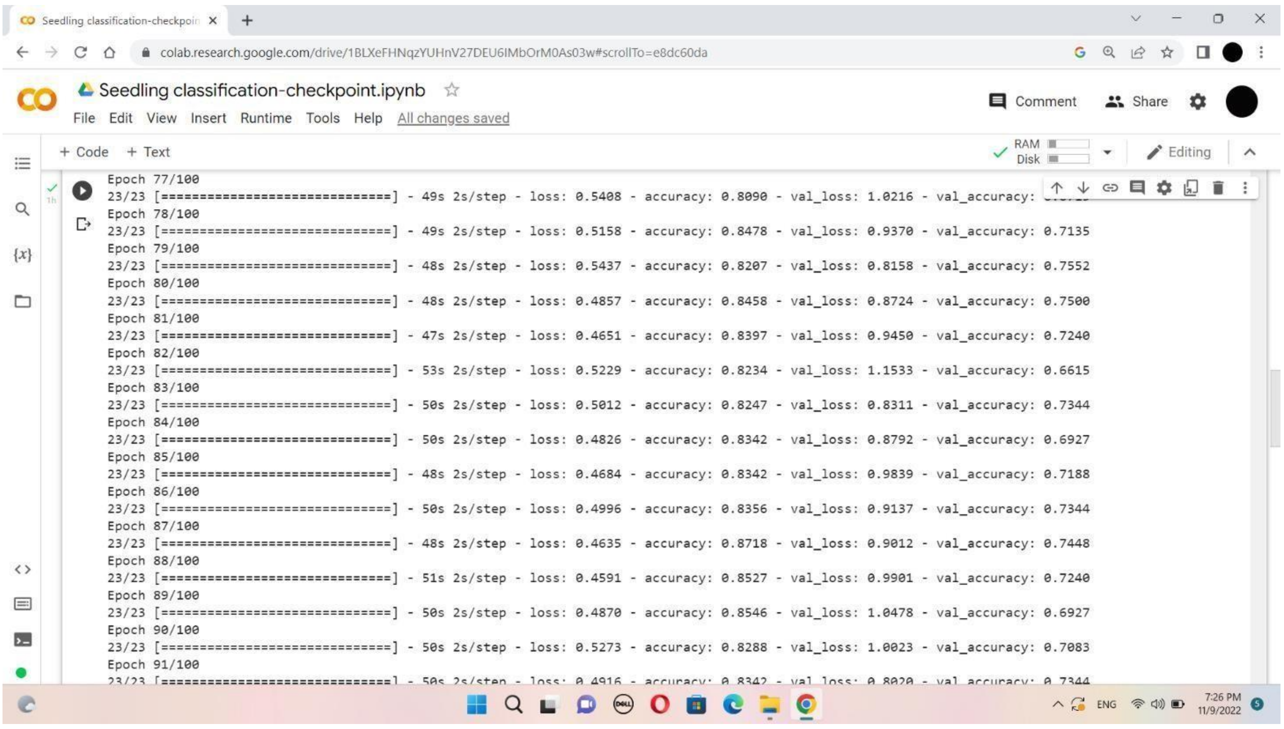
```
Epoch 37/100
23/23 [==============================] - 49s 2s/step - loss: 0.8249 - accuracy: 0.7255 - val_loss: 1.0691 - val_accuracy: 0.5990
Epoch 38/100
23/23 [==============================] - 49s 2s/step - loss: 0.7889 - accuracy: 0.7351 - val_loss: 1.0500 - val_accuracy: 0.6094
Epoch 39/100
23/23 [==============================] - 48s 2s/step - loss: 0.8330 - accuracy: 0.7188 - val_loss: 0.9575 - val_accuracy: 0.6510
Epoch 40/100
23/23 [==============================] - 49s 2s/step - loss: 0.7954 - accuracy: 0.7514 - val_loss: 1.2163 - val_accuracy: 0.5833
Epoch 41/100
23/23 [==============================] - 48s 2s/step - loss: 0.7909 - accuracy: 0.7473 - val_loss: 0.9991 - val_accuracy: 0.6615
Epoch 42/100
23/23 [==============================] - 49s 2s/step - loss: 0.7990 - accuracy: 0.7378 - val_loss: 1.1259 - val_accuracy: 0.6198
Epoch 43/100
23/23 [==============================] - 49s 2s/step - loss: 0.7371 - accuracy: 0.7364 - val_loss: 1.1335 - val_accuracy: 0.5885
Epoch 44/100
23/23 [==============================] - 54s 2s/step - loss: 0.7046 - accuracy: 0.7649 - val_loss: 0.9036 - val_accuracy: 0.7083
Epoch 45/100
23/23 [==============================] - 48s 2s/step - loss: 0.7592 - accuracy: 0.7514 - val_loss: 1.1569 - val_accuracy: 0.6198
Epoch 46/100
23/23 [==============================] - 47s 2s/step - loss: 0.6909 - accuracy: 0.7758 - val_loss: 1.0641 - val_accuracy: 0.6562
Epoch 47/100
23/23 [==============================] - 48s 2s/step - loss: 0.6776 - accuracy: 0.7785 - val_loss: 1.0287 - val_accuracy: 0.6667
Epoch 48/100
23/23 [==============================] - 49s 2s/step - loss: 0.7001 - accuracy: 0.7704 - val_loss: 0.8693 - val_accuracy: 0.7031
Epoch 49/100
23/23 [==============================] - 48s 2s/step - loss: 0.7228 - accuracy: 0.7595 - val_loss: 1.1221 - val_accuracy: 0.6615
Epoch 50/100
23/23 [==============================] - 49s 2s/step - loss: 0.7059 - accuracy: 0.7649 - val_loss: 1.1654 - val_accuracy: 0.6667
```


Seedling classification-checkpoint.ipynb
colab.research.google.com/drive/1BLXeFHNqzYUHnV27DEU6IMbOrM0As03w#scrollTo=i0xYInPFpgAz
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM Disk
Editing
7:25 PM 11/9/2022

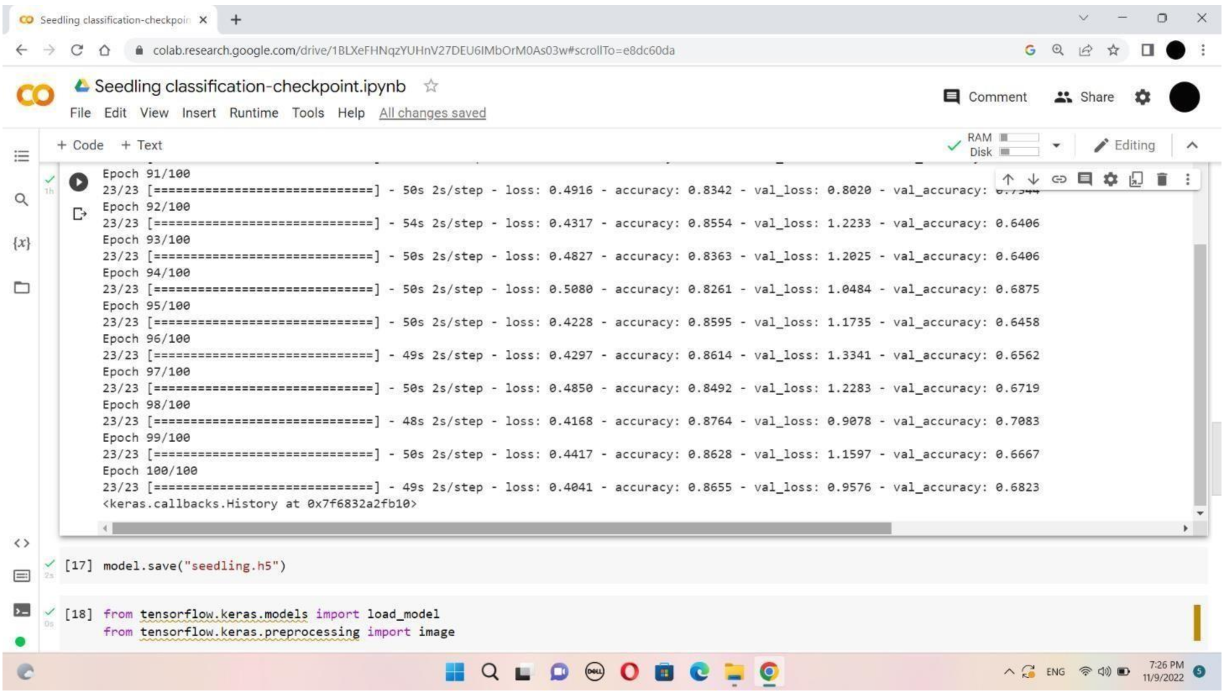
```
Epoch 51/100
23/23 [==============================] - 47s 2s/step - loss: 0.7134 - accuracy: 0.7772 - val_loss: 1.0443 - val_accuracy: 0.6250
Epoch 52/100
23/23 [==============================] - 48s 2s/step - loss: 0.7120 - accuracy: 0.7544 - val_loss: 1.0064 - val_accuracy: 0.6719
Epoch 53/100
23/23 [==============================] - 48s 2s/step - loss: 0.6386 - accuracy: 0.7812 - val_loss: 1.1797 - val_accuracy: 0.6458
Epoch 54/100
23/23 [==============================] - 49s 2s/step - loss: 0.7290 - accuracy: 0.7473 - val_loss: 1.1085 - val_accuracy: 0.6510
Epoch 55/100
23/23 [==============================] - 49s 2s/step - loss: 0.7837 - accuracy: 0.7378 - val_loss: 1.2471 - val_accuracy: 0.6667
Epoch 56/100
23/23 [==============================] - 49s 2s/step - loss: 0.6246 - accuracy: 0.7976 - val_loss: 0.9866 - val_accuracy: 0.6562
Epoch 57/100
23/23 [==============================] - 48s 2s/step - loss: 0.6325 - accuracy: 0.7913 - val_loss: 0.8458 - val_accuracy: 0.6719
Epoch 58/100
23/23 [==============================] - 47s 2s/step - loss: 0.6010 - accuracy: 0.8043 - val_loss: 1.0625 - val_accuracy: 0.6667
Epoch 59/100
23/23 [==============================] - 48s 2s/step - loss: 0.6678 - accuracy: 0.7804 - val_loss: 1.1093 - val_accuracy: 0.6406
Epoch 60/100
23/23 [==============================] - 48s 2s/step - loss: 0.6676 - accuracy: 0.7690 - val_loss: 1.1657 - val_accuracy: 0.6354
Epoch 61/100
23/23 [==============================] - 48s 2s/step - loss: 0.6188 - accuracy: 0.7962 - val_loss: 1.0065 - val_accuracy: 0.6771
Epoch 62/100
23/23 [==============================] - 48s 2s/step - loss: 0.6487 - accuracy: 0.7948 - val_loss: 0.8574 - val_accuracy: 0.7031
Epoch 63/100
23/23 [==============================] - 50s 2s/step - loss: 0.6279 - accuracy: 0.7894 - val_loss: 0.9546 - val_accuracy: 0.6615
Epoch 64/100
23/23 [==============================] - 48s 2s/step - loss: 0.7104 - accuracy: 0.7527 - val_loss: 1.0306 - val_accuracy: 0.6823
```


Seedling classification-checkpoint.ipynb
colab.research.google.com/drive/1BLXeFHNqzYUHnV27DEU6IMbOrM0As03w#scrollTo=i0xYInPFpgAz
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM Disk
Editing
7:25 PM 11/9/2022

```
Epoch 65/100
23/23 [==============================] - 48s 2s/step - loss: 0.5798 - accuracy: 0.8166 - val_loss: 0.8982 - val_accuracy: 0.7240
Epoch 66/100
23/23 [==============================] - 50s 2s/step - loss: 0.5971 - accuracy: 0.7921 - val_loss: 1.0627 - val_accuracy: 0.6302
Epoch 67/100
23/23 [==============================] - 49s 2s/step - loss: 0.6113 - accuracy: 0.7940 - val_loss: 0.9010 - val_accuracy: 0.6875
Epoch 68/100
23/23 [==============================] - 49s 2s/step - loss: 0.5618 - accuracy: 0.8370 - val_loss: 1.0072 - val_accuracy: 0.6927
Epoch 69/100
23/23 [==============================] - 49s 2s/step - loss: 0.5143 - accuracy: 0.8234 - val_loss: 1.1039 - val_accuracy: 0.6927
Epoch 70/100
23/23 [==============================] - 48s 2s/step - loss: 0.6276 - accuracy: 0.7921 - val_loss: 1.0557 - val_accuracy: 0.6875
Epoch 71/100
23/23 [==============================] - 49s 2s/step - loss: 0.5326 - accuracy: 0.8302 - val_loss: 1.0637 - val_accuracy: 0.6823
Epoch 72/100
23/23 [==============================] - 49s 2s/step - loss: 0.4768 - accuracy: 0.8573 - val_loss: 0.9452 - val_accuracy: 0.7448
Epoch 73/100
23/23 [==============================] - 49s 2s/step - loss: 0.5469 - accuracy: 0.8302 - val_loss: 1.1925 - val_accuracy: 0.6458
Epoch 74/100
23/23 [==============================] - 48s 2s/step - loss: 0.5191 - accuracy: 0.8261 - val_loss: 0.8729 - val_accuracy: 0.7083
Epoch 75/100
23/23 [==============================] - 48s 2s/step - loss: 0.5107 - accuracy: 0.8226 - val_loss: 0.9024 - val_accuracy: 0.6771
Epoch 76/100
23/23 [==============================] - 49s 2s/step - loss: 0.5773 - accuracy: 0.8071 - val_loss: 1.1508 - val_accuracy: 0.6250
Epoch 77/100
23/23 [==============================] - 49s 2s/step - loss: 0.5408 - accuracy: 0.8090 - val_loss: 1.0216 - val_accuracy: 0.6719
Epoch 78/100
23/23 [==============================] - 49s 2s/step - loss: 0.5158 - accuracy: 0.8478 - val_loss: 0.9370 - val_accuracy: 0.7135
```


Seedling classification-checkpoint.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Editing

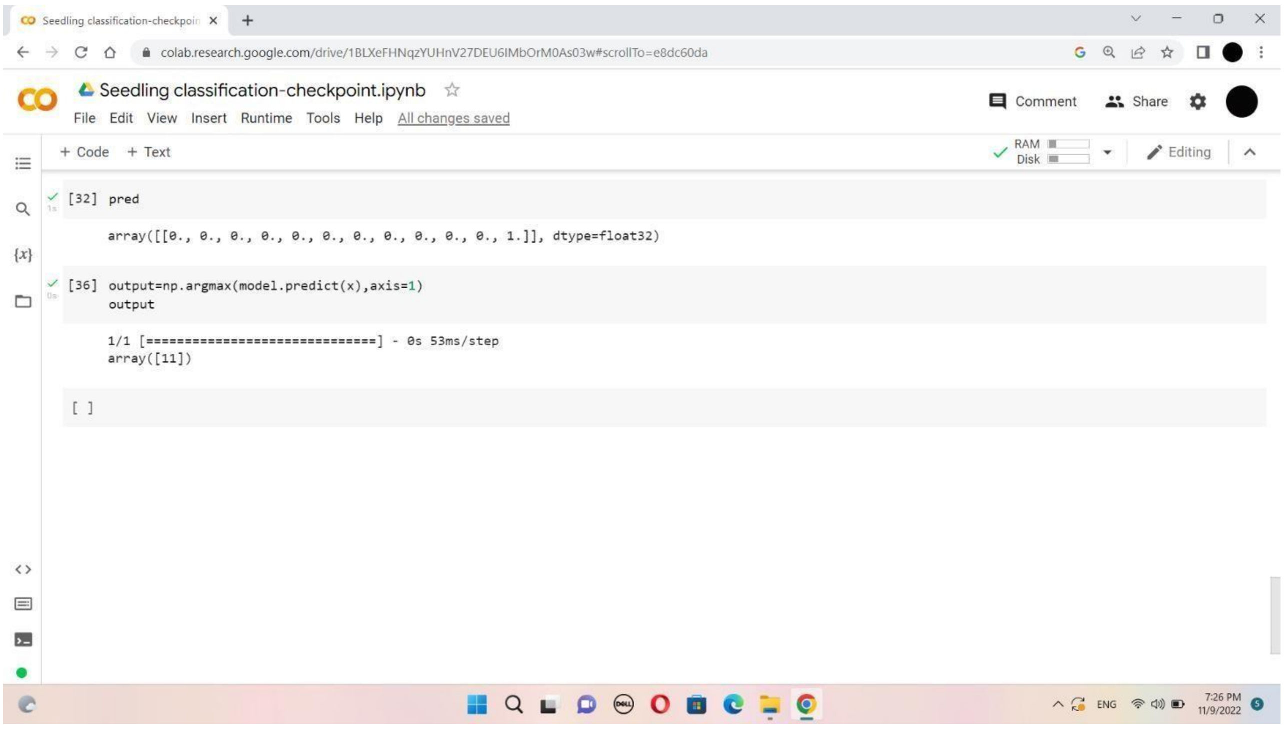
```
Epoch 77/100
23/23 [==============================] - 49s 2s/step - loss: 0.5408 - accuracy: 0.8090 - val_loss: 1.0216 - val_accuracy:
Epoch 78/100
23/23 [==============================] - 49s 2s/step - loss: 0.5158 - accuracy: 0.8478 - val_loss: 0.9370 - val_accuracy: 0.7135
Epoch 79/100
23/23 [==============================] - 48s 2s/step - loss: 0.5437 - accuracy: 0.8207 - val_loss: 0.8158 - val_accuracy: 0.7552
Epoch 80/100
23/23 [==============================] - 48s 2s/step - loss: 0.4857 - accuracy: 0.8458 - val_loss: 0.8724 - val_accuracy: 0.7500
Epoch 81/100
23/23 [==============================] - 47s 2s/step - loss: 0.4651 - accuracy: 0.8397 - val_loss: 0.9450 - val_accuracy: 0.7240
Epoch 82/100
23/23 [==============================] - 53s 2s/step - loss: 0.5229 - accuracy: 0.8234 - val_loss: 1.1533 - val_accuracy: 0.6615
Epoch 83/100
23/23 [==============================] - 50s 2s/step - loss: 0.5012 - accuracy: 0.8247 - val_loss: 0.8311 - val_accuracy: 0.7344
Epoch 84/100
23/23 [==============================] - 50s 2s/step - loss: 0.4826 - accuracy: 0.8342 - val_loss: 0.8792 - val_accuracy: 0.6927
Epoch 85/100
23/23 [==============================] - 48s 2s/step - loss: 0.4684 - accuracy: 0.8342 - val_loss: 0.9839 - val_accuracy: 0.7188
Epoch 86/100
23/23 [==============================] - 50s 2s/step - loss: 0.4996 - accuracy: 0.8356 - val_loss: 0.9137 - val_accuracy: 0.7344
Epoch 87/100
23/23 [==============================] - 48s 2s/step - loss: 0.4635 - accuracy: 0.8718 - val_loss: 0.9012 - val_accuracy: 0.7448
Epoch 88/100
23/23 [==============================] - 51s 2s/step - loss: 0.4591 - accuracy: 0.8527 - val_loss: 0.9901 - val_accuracy: 0.7240
Epoch 89/100
23/23 [==============================] - 50s 2s/step - loss: 0.4870 - accuracy: 0.8546 - val_loss: 1.0478 - val_accuracy: 0.6927
Epoch 90/100
23/23 [==============================] - 50s 2s/step - loss: 0.5273 - accuracy: 0.8288 - val_loss: 1.0023 - val_accuracy: 0.7083
Epoch 91/100
23/23 [==============================] - 50s 2s/step - loss: 0.4916 - accuracy: 0.8342 - val_loss: 0.8020 - val_accuracy: 0.7344
```


Seedling classification-checkpoint.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Editing

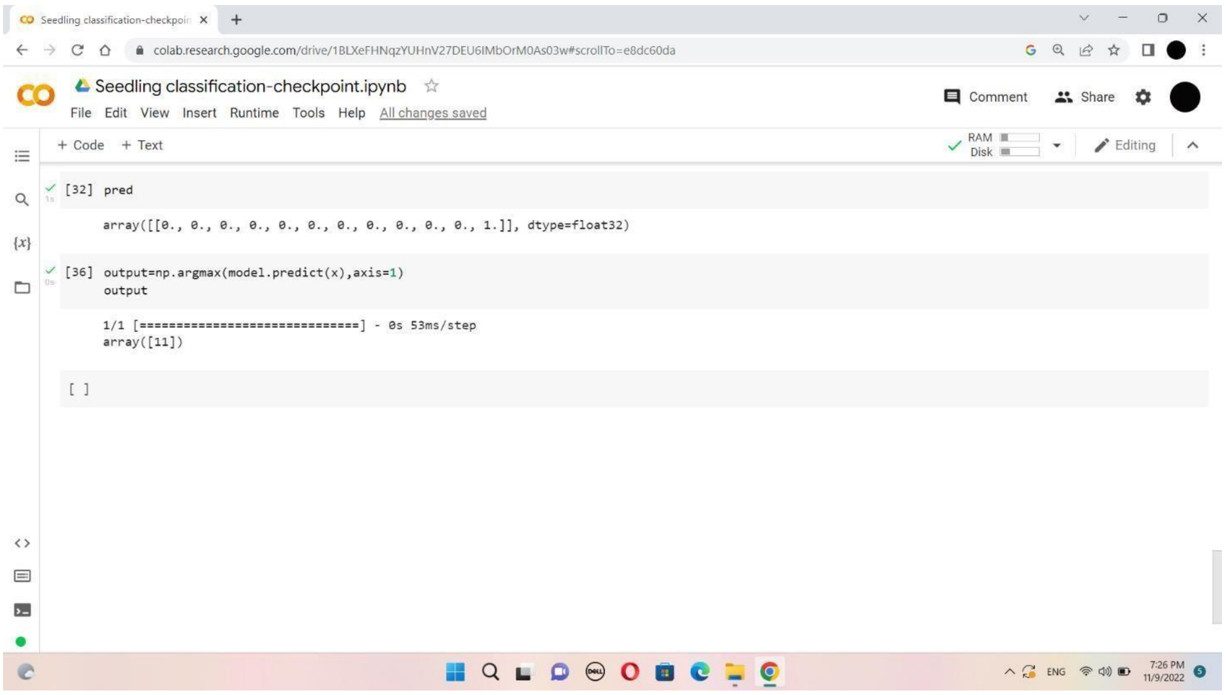
```
Epoch 91/100
23/23 [==============================] - 50s 2s/step - loss: 0.4916 - accuracy: 0.8342 - val_loss: 0.8020 - val_accuracy:
Epoch 92/100
23/23 [==============================] - 54s 2s/step - loss: 0.4317 - accuracy: 0.8554 - val_loss: 1.2233 - val_accuracy: 0.6406
Epoch 93/100
23/23 [==============================] - 50s 2s/step - loss: 0.4827 - accuracy: 0.8363 - val_loss: 1.2025 - val_accuracy: 0.6406
Epoch 94/100
23/23 [==============================] - 50s 2s/step - loss: 0.5080 - accuracy: 0.8261 - val_loss: 1.0484 - val_accuracy: 0.6875
Epoch 95/100
23/23 [==============================] - 50s 2s/step - loss: 0.4228 - accuracy: 0.8595 - val_loss: 1.1735 - val_accuracy: 0.6458
Epoch 96/100
23/23 [==============================] - 49s 2s/step - loss: 0.4297 - accuracy: 0.8614 - val_loss: 1.3341 - val_accuracy: 0.6562
Epoch 97/100
23/23 [==============================] - 50s 2s/step - loss: 0.4850 - accuracy: 0.8492 - val_loss: 1.2283 - val_accuracy: 0.6719
Epoch 98/100
23/23 [==============================] - 48s 2s/step - loss: 0.4168 - accuracy: 0.8764 - val_loss: 0.9078 - val_accuracy: 0.7083
Epoch 99/100
23/23 [==============================] - 50s 2s/step - loss: 0.4417 - accuracy: 0.8628 - val_loss: 1.1597 - val_accuracy: 0.6667
Epoch 100/100
23/23 [==============================] - 49s 2s/step - loss: 0.4041 - accuracy: 0.8655 - val_loss: 0.9576 - val_accuracy: 0.6823
<keras.callbacks.History at 0x7f6832a2fb10>
```

```
[17] model.save("seedling.h5")
```

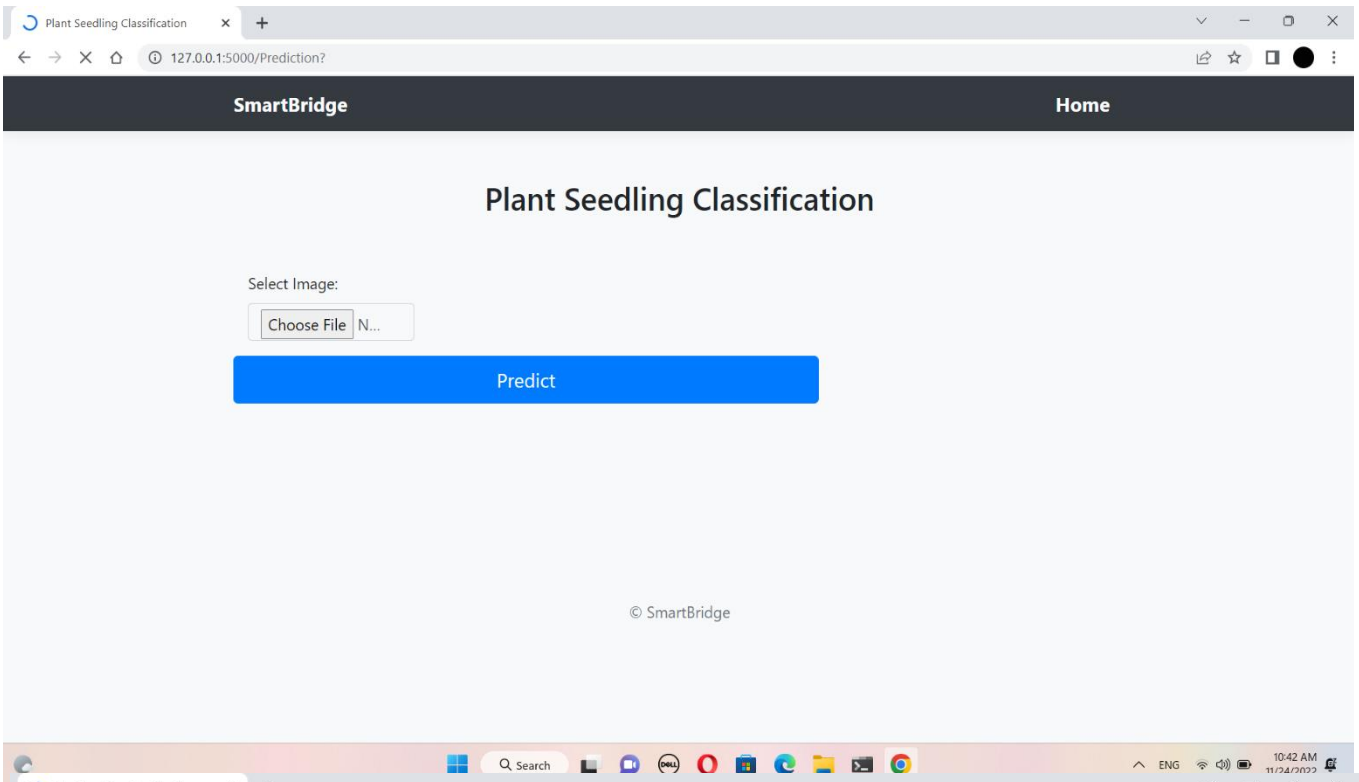
```
[18] from tensorflow.keras.models import load_model
     from tensorflow.keras.preprocessing import image
```

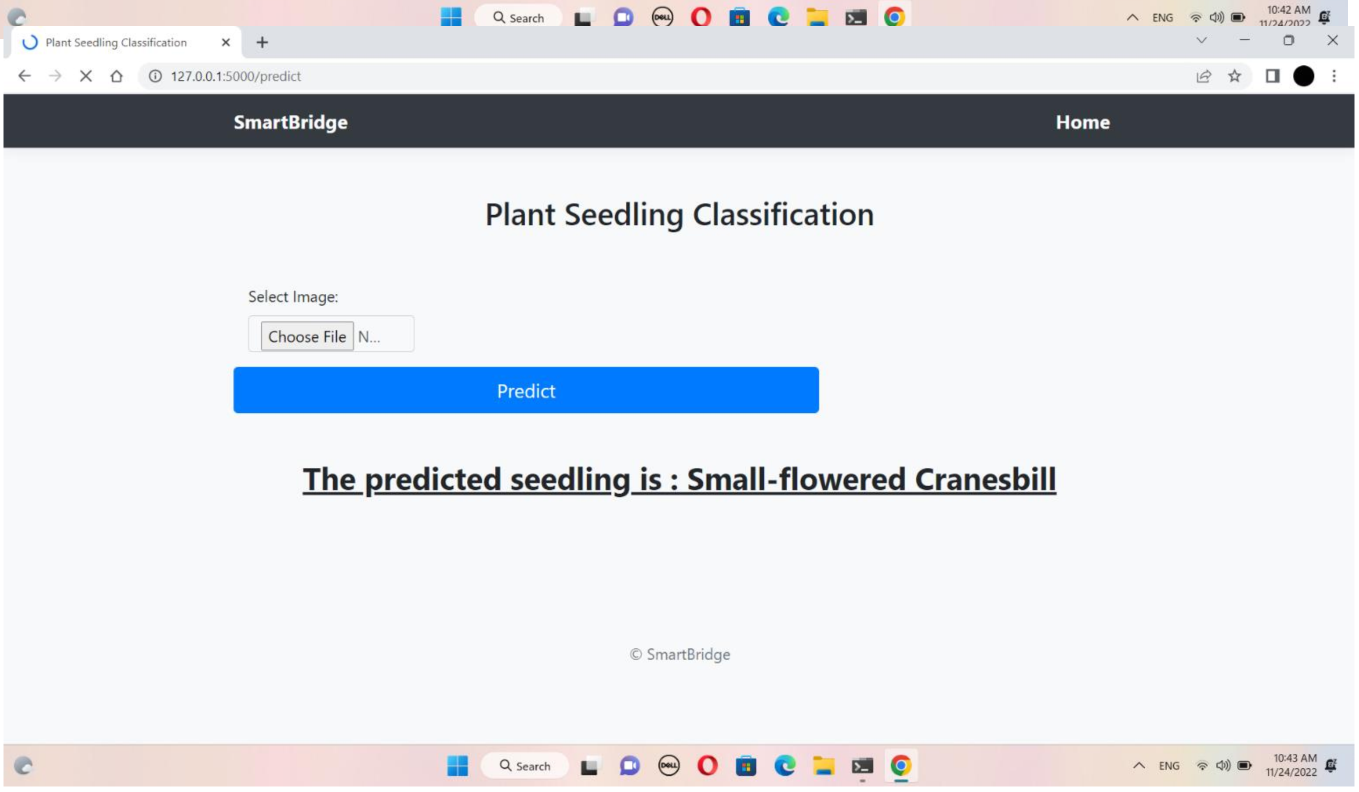

Seedling classification-checkpoint.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk
Editing

```
[32] pred

     array([[0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1.]], dtype=float32)

[36] output=np.argmax(model.predict(x),axis=1)
     output

     1/1 [==============================] - 0s 53ms/step
     array([11])

[ ]
```



```
[32] pred

     array([[0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1.]], dtype=float32)

[36] output=np.argmax(model.predict(x),axis=1)
     output

     1/1 [==============================] - 0s 53ms/step
     array([11])

[ ]
```


Plant Seedling Classification
127.0.0.1:5000/Prediction?
SmartBridge
Home
Plant Seedling Classification
Select Image:
Choose File N...
Predict
© SmartBridge



The predicted seedling is : Small-flowered Cranesbill

4.1 Model evaluation and confusion matrix

As shown in the dataset section, we split data into training data, validation data and testing data to train and evaluate the model.

- **Accuracy and loss**

Loss function is categorical cross entropy. Suppose there are N samples and K labels. y is the true labels, p is the probability of the prediction and L is the cross entropy.

$$L(y,p) = -\frac{1}{N}\sum_{i=0}^{N-1}\sum_{j=0}^{K-1} y_{i,j}log(p_{i,j})$$

Accuracy is a common metric for classifiers; it takes into account all true prediction with equal weight.

$$accuracy = \frac{true\ prediction}{dataset\ size}$$

- **Prediction time**

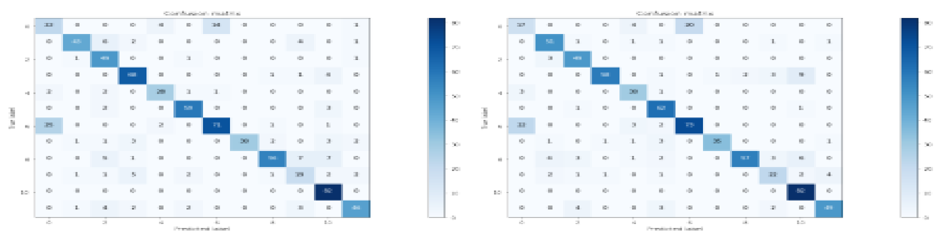
we use the prediction time of the test data to evaluate the prediction speed of our models. Here is the summary of all models we used.

TABLE I: Results summary

Model	*CNN*	*Data augmentation*	*ResNet*	*MobileNet*	*DenseNet*	*VGG*
Accuracy	0.7921	0.8230	0.9059	0.8947	0.8610	0.8722
Loss	0.4416	0.5209	0.2601	0.3214	0.4115	0.8290
Time	0.5594	0.8207	4.5817	1.0843	3.7533	2.5685

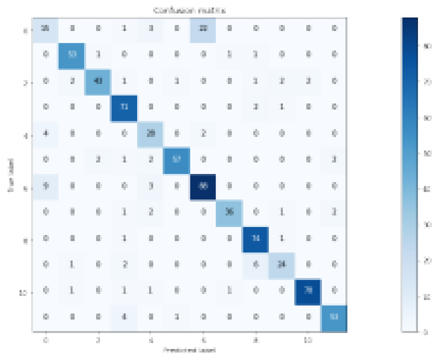
- **confusion matrix**

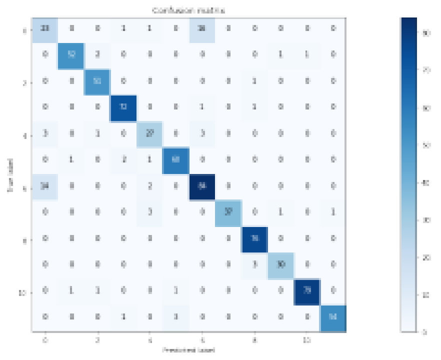
A confusion matrix C is such that C i, j is equal to the number of observations known to be in group i but predicted to be in group j. The confusion matrix could be very helpful to see the model drawbacks on classes of plant. Fig.5 - Fig.6 is the confusion matrices of our models.

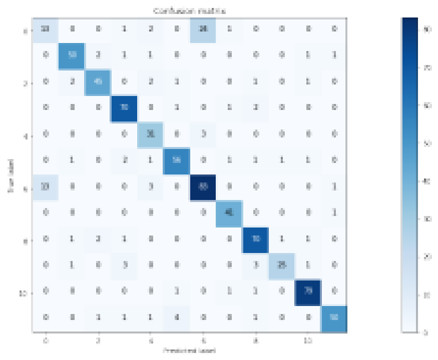


(a) Without augmentation (b) With augmentation

Fig. 5: self-built CNN model confusion matrix


(a) Confusion matrix(VGG)


(b) Confusion matrix(ResNet)


(c) Confusion matrix(DenseNet)

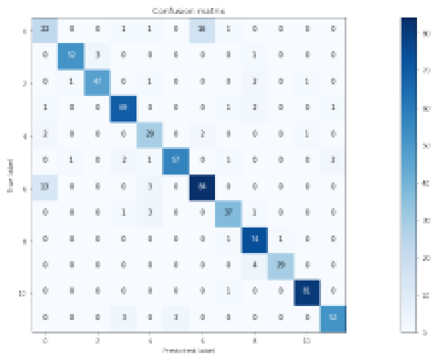

(d) Confusion matrix(MobileNet)

Fig. 6: Confusion matrices of different models using transfer learning

CHAPTER 6
CONCLUSION

First ,we try the self-built CNN model illustrated above, while we found that the accuracy is not very ideal. Then we try the ResNet model to improve the accuracy and get the expected result. Further, we want to decrease computation time and apply MobileNet model to the dataset.

Computation time reduce a lot as we expect and the accuracy still remains high. To have a better understanding of the performance, we try the other two models listed in the above sections. In general, we prefer MobileNet model with high accuracy and low latency. According to the results shown in the above section, the prediction of the model is acceptable for the application, and from the confusion matrix, we could know more about the precision of the prediction. This model could be applied to help farmers to automatically classify the seedling plants and weed plants

CHAPTER 7
REFERENCES

[1] T. Brendel, J. Schwanke, P. F. Jensch, and R. Megnet, “Knowledge based object recognition for different morphological classes of plants,” in Optics in Agriculture, Forestry, and Biological Processing, vol. 2345, pp. 277–284, International Society for Optics and Photonics, 1995.

[2] E. Neilson, A. Edwards, C. Blomstedt, B. Berger, B. Møller, and R. Gleadow, “Utilization of a high- throughput shoot imaging system to examine the dynamic phenotypic responses of a c4 cereal crop plant to nitrogen and water deficiency over time,” 2015.

[3] X. Luo, D. Jayas, and S. Symons, “Comparison of statistical and neural network methods for classifying cereal grains using machine vision,” Transactions of the ASAE, vol. 42, no. 2, p. 413, 1999.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in Advances in neural information processing systems, pp. 1097– 1105, 2012.

[5] T. M. Giselsson, H. S. Midtiby, and R. N. Jørgensen, “Seedling discrimination with shape features derived from a distance transform,” Sensors, vol. 13, no. 5, pp. 5585–5602, 2013.

[6] B. A. Ashqar, B. S. Abu-Nasser, and S. S. Abu-Naser, “Plant seedlings classification using deep learning,” 2019.

[7] C.-C. Andrea, B. B. M. Daniel, and J. B. J. Misael, “Precise weed and maize classification through convolutional neuronal networks,” in 2017 IEEE Second Ecuador Technical Chapters Meeting (ETCM), pp. 1–6, IEEE, 2017.

[8] A. Johannes, A. Picon, A. Alvarez-Gila, J. Echazarra, S. RodriguezVaamonde, A. D. Navajas, and A. Ortiz-Barredo, “Automatic plant disease diagnosis using mobile capture devices, applied on a wheat use case,” Computers and electronics in agriculture, vol. 138, pp. 200–209, 2017.

[9] G. L. Grinblat, L. C. Uzal, M. G. Larese, and P. M. Granitto, “Deep learning for plant identification using vein morphological patterns,” Computers and Electronics in Agriculture, vol. 127, pp. 418–424, 2016.

[10] S. Skovsen, M. Dyrmann, A. K. Mortensen, K. A. Steen, O. Green, J. Eriksen, R. Gislum, R. N. Jørgensen, and H. Karstoft, “Estimation of the botanical composition of clover-grass leys from rgb images using data simulation and fully convolutional neural networks,” Sensors, vol. 17, no. 12, p. 2930, 2017.

[11] Computer vision with seedlings, 2020. https://www.kaggle.com/allunia/computer-vision-with- seedlings.