Plant Seedlings Classification Report

XIAOJING YE A1792345

The University of Adelaide, Faculty of ECMS, a1792345@student.adelaide.edu.au

1 INTRODUCTION

Plants are incredibly complex, and they have thousands of species. They are probably somewhere around half a million different species. There is a good project from Kaggle named Plant Seedlings Classification that can help us determine the seedling species from an image.

1.1 Problem

The goal of the project is to determine the specie of plant seedlings from an image.

1.2 Datasets

From the Kaggle, we got two sets of data, a training dataset, and a test dataset. In the training dataset, there are a total of 12 folders of images of plant seedlings at various stages of grown. In the test dataset, it contains bounds of images of different plant seedlings. Each image has its unique id.

2 METHODS

It is critical to understand image classification to achieve the goal of the experiment. Image classification is a process run by a computer that can analyze an image and identify the ‘class’ of the picture. A class can be seen as a label, such as ‘cat, ‘birds, ‘flowers, and so on. For example, we input an image of a cat, and the computer can tell us the picture is a cat through image classification.

One of the famous algorithms for classifying images in DL is a convolutional neural network [1], also known as CNN or ConvNet.

2.1 Import and Preprocess the Data

In this project, the data is already split into two sets of training and testing data. And according to the requirements of the project, training data should be spilt into 2 parts for training and validation. Besides, it still needs reshape dataset inputs to fit the model expects for training.

2.2 Build CNN model

The model type that the project used is Sequential. Sequential, a common way to build a model in Keras, allows to build a model layer by layer.

The figure bellowed shows what a convolution is. Suppose the input data is a 5×5 matrix $\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix}$, and kernel(filter) is a 3×3 matrix $\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$. Applying the kernel to the input image, matrix $\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$, gets the convolved feature “4”. This convolved feature will be passed on to the next layer.

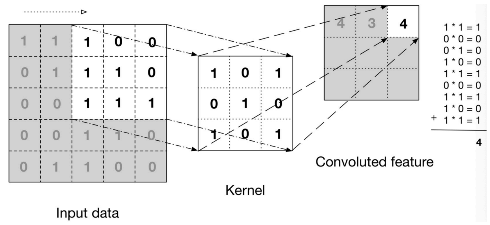

Figure 1: How CNN works

(https://editor.analyticsvidhya.com/uploads/750710_QS1ArBEUJjjySXhE.png)

There are several important layers in the model building. Convolution layers are used to deal with input images. Max Pooling describes the most activated feature. Batch Normalization simplifies the process of standardizing the inputs to a certain layer. Flatten is a connection between the dense layers and the convolution. Dense layer, a standard layer type, is used as output layer.

2.3 Compile model

Before the model is trained, it should be configured a learning process – compilation - which involves three parameters: optimizer, loss, and metrics.

Optimizers are used to adjust input weights by comparing prediction and the loss function. 'categorical_crossentropy' as a parameter of loss function in the model, it can show the performance of the model. The 'accuracy' matrix let the training model decide which metric should be used. Optimizers implements Adam algorithm in this project. According to an anticle written by Diederik P. Kingma and Jimmy Lei Ba, results demonstrate that Adam works well in practice and compares favorably to other stochastic optimization methods[2]. Figure 2 is the comparison of Adam to other optimization algorithms when training a multilayer perceptron, the training cost of Adam is lower than other algorithms.

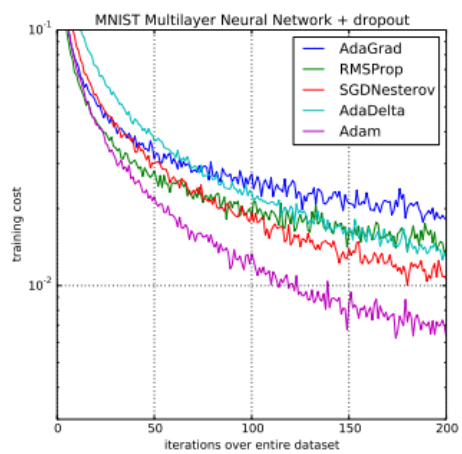

Figure 2: Comparison of Adam to other optimization algorithms when training a multilayer perceptron

taken from Adam: A Method for Stochastic Optimization, 2015.

2.4 Train model

The function fit () indicates the train, and the function with the following parameters: training data, validation data, and the number of epochs. The number of epochs is the number of times the model will cycle through the input data.

2.5 Make prediction

The number of epochs refers to the number of times that the model will run through the data. Each epoch will improve until it reaches a certain point. To prevent overfitting, a function named callbacks is defined to stop running through the data.

3 FINDLINGS AND RESULTS

3.1 findings

3.1.1 Visualization with Python

Python is an open-source platform that enables users to create interactive and detailed data visualizations with kinds of libraries. Matplotlib is a comprehensive library for creating static, animated, and interactive visualization.

The subplot () function in pyplot module are powerful visualizations that help easy comparisons between plots, and the imshow () function in pyplot module is used to display data as an image. When showing the image, the size of image can also be adjusted through figure () function. Code shown in figure 3 shows how to print nine pictures in a loop. And figure 4 is the nine pictures converted by input data.

```
species_names = training_dataset.class_names
plt.figure(figsize=(8, 8))
for images, names in training_dataset.take(1):
  for i in range(9):
    ax = plt.subplot(3, 3, i + 1)
    plt.imshow(images[i].numpy().astype("uint8"))
    plt.title(species_names[names[i]])
    plt.axis("off")
```

Figure 3: Code shows how data visualization

Figure 4: Display data as image

When building a deep neural network using the Keras framework, one of the most critical steps is ensuring that everything is working properly. Luckily, Keras provides a text-based overview of the model by calling the summary () function on summaries model. As what shown in figure 5, the summary contains the layers, the output shape of each layer, the number of parameters (weights) in each layer, and the total number of parameters (weights) in the model.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 62, 62, 64)        1792
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 31, 31, 64)        0
_________________________________________________________________
batch_normalization (BatchNo (None, 31, 31, 64)        256
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 29, 29, 64)        36928
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 14, 14, 64)        0
_________________________________________________________________
batch_normalization_1 (Batch (None, 14, 14, 64)        256
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 12, 12, 128)       73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 6, 6, 128)         0
_________________________________________________________________
batch_normalization_2 (Batch (None, 6, 6, 128)         512
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 4, 4, 128)         147584
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 2, 2, 128)         0
_________________________________________________________________
global_max_pooling2d (Global (None, 128)               0
_________________________________________________________________
flatten (Flatten)            (None, 128)               0
_________________________________________________________________
dense (Dense)                (None, 128)               16512
_________________________________________________________________
dropout (Dropout)            (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                8256
_________________________________________________________________
batch_normalization_3 (Batch (None, 64)                256
_________________________________________________________________
dense_2 (Dense)              (None, 12)                780
=================================================================
Total params: 286,988
Trainable params: 286,348
Non-trainable params: 640
```

Figure 5: Model Summarization in Text

When the model is being trained, python only displays the text information of the running result automatically. It cannot visually display the trend of the data. As it is shown in figure 6. The picture tells some information about accuracy and loss in each epoch. However, it is hard to visually display the trend of the data. Then, the subplot () function plays an important role here. Figure 7, charts show comparison of accuracy, loss between training and accuracy and the changes are clear.

```
Epoch 1/20
30/30 [==============================] - 12s 260ms/step - loss: 2.6166 - accuracy: 0.1832 - val_loss: 7.4855 - val_
accuracy: 0.0558
Epoch 2/20
30/30 [==============================] - 4s 145ms/step - loss: 2.0435 - accuracy: 0.3137 - val_loss: 3.1824 - val_a
ccuracy: 0.1274
Epoch 3/20
30/30 [==============================] - 4s 145ms/step - loss: 1.6897 - accuracy: 0.4389 - val_loss: 2.3919 - val_a
ccuracy: 0.2347
Epoch 4/20
30/30 [==============================] - 4s 146ms/step - loss: 1.3856 - accuracy: 0.5495 - val_loss: 2.0566 - val_a
ccuracy: 0.3305
Epoch 5/20
30/30 [==============================] - 4s 149ms/step - loss: 1.1913 - accuracy: 0.6242 - val_loss: 2.2531 - val_a
ccuracy: 0.1853
Epoch 6/20
30/30 [==============================] - 4s 148ms/step - loss: 1.0434 - accuracy: 0.6611 - val_loss: 1.6217 - val_a
ccuracy: 0.4305
Epoch 7/20
30/30 [==============================] - 4s 151ms/step - loss: 0.8113 - accuracy: 0.7495 - val_loss: 1.9229 - val_a
ccuracy: 0.3547
Epoch 8/20
30/30 [==============================] - 4s 148ms/step - loss: 0.6526 - accuracy: 0.8011 - val_loss: 1.2840 - val_a
ccuracy: 0.5021
Epoch 9/20
30/30 [==============================] - 4s 151ms/step - loss: 0.5098 - accuracy: 0.8579 - val_loss: 1.2238 - val_a
ccuracy: 0.5800
Epoch 10/20
30/30 [==============================] - 5s 157ms/step - loss: 0.4284 - accuracy: 0.8874 - val_loss: 1.1364 - val_a
ccuracy: 0.6168
Epoch 11/20
30/30 [==============================] - 5s 160ms/step - loss: 0.2957 - accuracy: 0.9400 - val_loss: 1.0823 - val_a
ccuracy: 0.6284
Epoch 12/20
30/30 [==============================] - 5s 155ms/step - loss: 0.2674 - accuracy: 0.9421 - val_loss: 1.1357 - val_a
ccuracy: 0.6179
Epoch 13/20
30/30 [==============================] - 4s 150ms/step - loss: 0.1913 - accuracy: 0.9768 - val_loss: 1.3860 - val_a
ccuracy: 0.5337
Epoch 14/20
30/30 [==============================] - 5s 168ms/step - loss: 0.1533 - accuracy: 0.9726 - val_loss: 1.3912 - val_a
ccuracy: 0.5484
```

Figure 6: Results of training

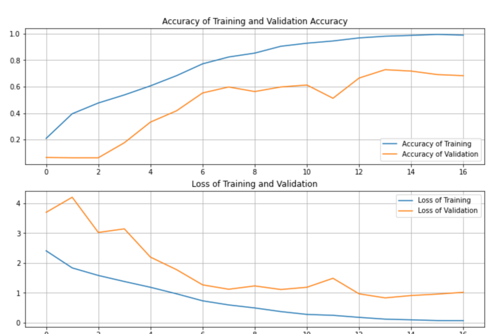

Figure 7: Comparison of accuracy, loss between training and accuracy

3.2 results

In conclusion, through the training of massive data, the machine enables the model to grasp the underlying data and accurately classify or predict the newly input data.

The upper limit of the number of epochs is 20. And in this experiment, the number of epochs stopped at 14 for avoid overfitting. After 14 epochs, the model got 54.84% accuracy on validation set.

Recall the figure 6, first, as the number of epochs increases, the time for the model to analyze data gets faster and faster. It shows that the learning ability of the model is getting stronger and stronger.

After 14 rounds of learning, in general, whether it is training or validation, the value of Accuracy shows an upward trend. Although at the 12th time, the value dropped slightly. The change of loss and Accuracy is inversely proportional. This is as it should be.

When the test data is input into the model, the model predicts it. Part of the prediction results are shown in the figure 8, one file corresponds to one category.

submission

file	species
1b490196c.png	Shepherds Purse
85431c075.png	Loose Silky-bent
506347cfe.png	Scentless Mayweed
7f46a71db.png	Scentless Mayweed
668c1007c.png	Charlock
71f5323c5.png	Loose Silky-bent
1f3f44563.png	Charlock
beebe5f4e.png	Fat Hen
780defa2e.png	Common Chickweed
df521c0c0.png	Loose Silky-bent
466bb6d3b.png	Scentless Mayweed
98d819587.png	Shepherds Purse
223e4af09.png	Cleavers
abc331628.png	Common wheat
eef131644.png	Loose Silky-bent
b7a7f6390.png	Fat Hen
7d3045fc3.png	Sugar beet
1926e82fd.png	Loose Silky-bent

Figure 8: Results of Prediction

REFERENCES

[1] Seetala K., Birdsong W., Reddy Y.B. (2019) Image Classification Using TensorFlow. In: Latifi S. (eds) 16th International Conference on Information Technology-New Generations (ITNG 2019). Advances in Intelligent Systems and Computing, vol 800. Springer, Cham. https://doi.org/10.1007/978-3-030-14070-0_67

[2] Kingma, D.P. and Ba, J., 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.