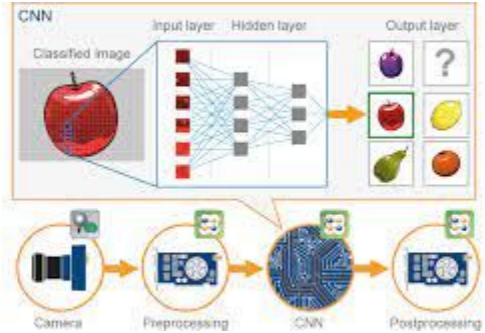
Capstone

Fresh Fruit Classification With CNN

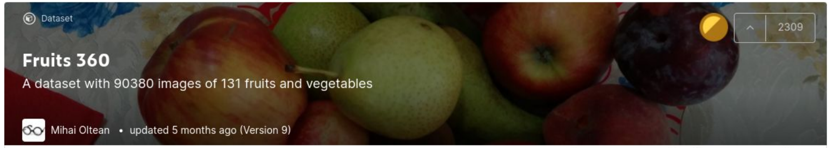
Overview


CNN
Input layer
Hidden layer
Output layer
Classified image
?
Camera
Preprocessing
CNN
Postprocessing

Why It Matters

About This Dataset

kaggle



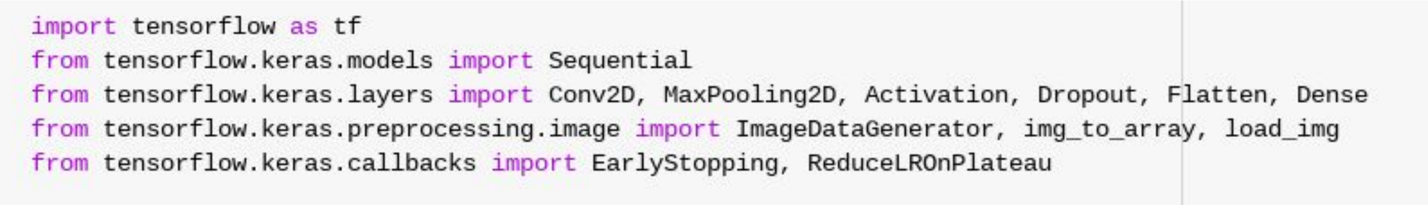
Fruits 360

A dataset with 90380 images of 131 fruits and vegetables

Mihai Oltean • updated 5 months ago (Version 9)

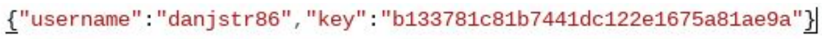
- The total number of images: 90483.
- Training set size: 67692 images.
- Test set size: 22688 images.
- The number of classes: 131 (fruits and vegetables).
- Image size: 100x100 pixels.

Importing Dependencies



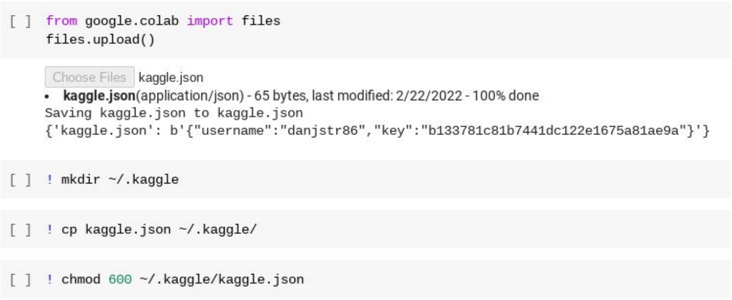
```python
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Activation, Dropout, Flatten, Dense
from tensorflow.keras.preprocessing.image import ImageDataGenerator, img_to_array, load_img
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
```

Kaggle API



```
{"username":"danjstr86","key":"b133781c81b7441dc122e1675a81ae9a"}
```

Importing Data With API



```
[ ] from google.colab import files
    files.upload()
```

kaggle.json

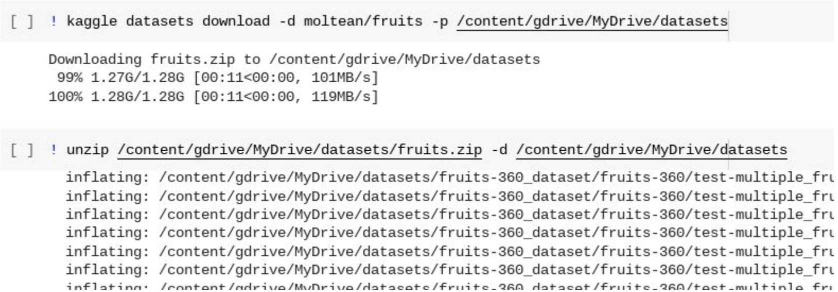
- **kaggle.json**(application/json) - 65 bytes, last modified: 2/22/2022 - 100% done

```
Saving kaggle.json to kaggle.json
{'kaggle.json': b'{"username":"danjstr86","key":"b133781c81b7441dc122e1675a81ae9a"}'}
```

```
[ ] ! mkdir ~/.kaggle
```

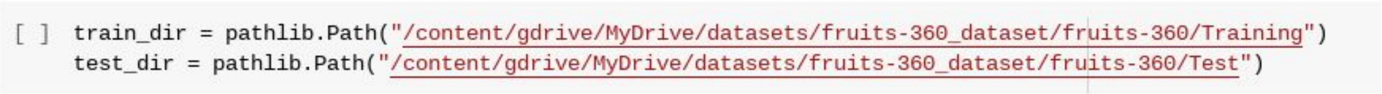
```
[ ] ! cp kaggle.json ~/.kaggle/
```

```
[ ] ! chmod 600 ~/.kaggle/kaggle.json
```



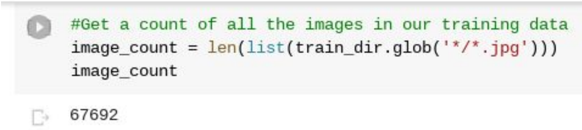
```
[ ] ! kaggle datasets download -d moltean/fruits -p /content/gdrive/MyDrive/datasets

    Downloading fruits.zip to /content/gdrive/MyDrive/datasets
     99% 1.27G/1.28G [00:11<00:00, 101MB/s]
    100% 1.28G/1.28G [00:11<00:00, 119MB/s]
```

```
[ ] ! unzip /content/gdrive/MyDrive/datasets/fruits.zip -d /content/gdrive/MyDrive/datasets
      inflating: /content/gdrive/MyDrive/datasets/fruits-360_dataset/fruits-360/test-multiple_fru
      inflating: /content/gdrive/MyDrive/datasets/fruits-360_dataset/fruits-360/test-multiple_fru
      inflating: /content/gdrive/MyDrive/datasets/fruits-360_dataset/fruits-360/test-multiple_fru
      inflating: /content/gdrive/MyDrive/datasets/fruits-360_dataset/fruits-360/test-multiple_fru
      inflating: /content/gdrive/MyDrive/datasets/fruits-360_dataset/fruits-360/test-multiple_fru
      inflating: /content/gdrive/MyDrive/datasets/fruits-360_dataset/fruits-360/test-multiple_fru
      inflating: /content/gdrive/MyDrive/datasets/fruits-360_dataset/fruits-360/test-multiple_fru
```



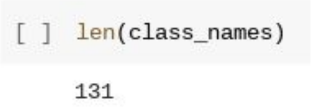
```
[ ] train_dir = pathlib.Path("/content/gdrive/MyDrive/datasets/fruits-360_dataset/fruits-360/Training")
    test_dir = pathlib.Path("/content/gdrive/MyDrive/datasets/fruits-360_dataset/fruits-360/Test")
```

Confirming Our Imports

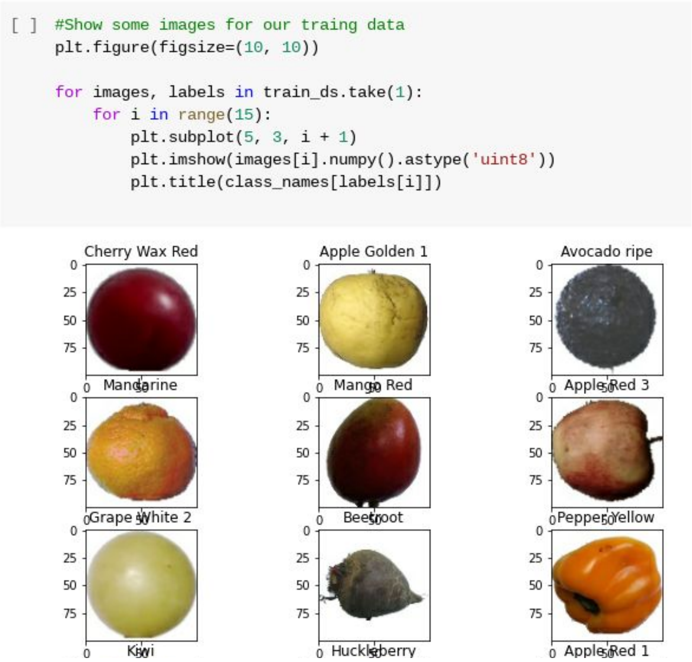


```
#Get a count of all the images in our training data
image_count = len(list(train_dir.glob('*/*.jpg')))
image_count
```

```
67692
```

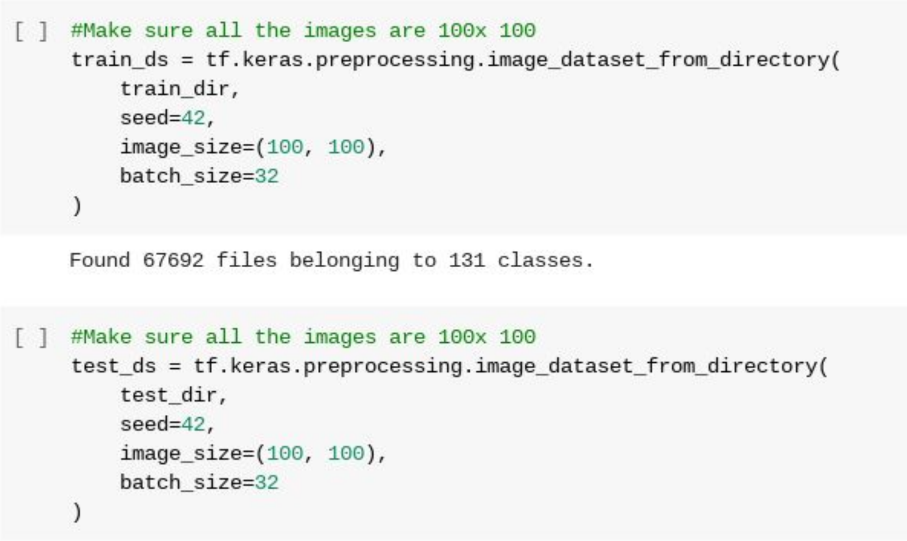


```
[ ]  len(class_names)

     131
```



```
[ ]  #Show some images for our traing data
     plt.figure(figsize=(10, 10))

     for images, labels in train_ds.take(1):
         for i in range(15):
             plt.subplot(5, 3, i + 1)
             plt.imshow(images[i].numpy().astype('uint8'))
             plt.title(class_names[labels[i]])
```

Creating Batches

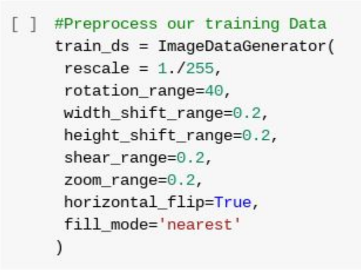


```
[ ]  #Make sure all the images are 100x 100
     train_ds = tf.keras.preprocessing.image_dataset_from_directory(
         train_dir,
         seed=42,
         image_size=(100, 100),
         batch_size=32
     )
```

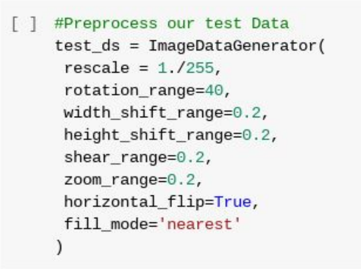
```
Found 67692 files belonging to 131 classes.
```

```
[ ]  #Make sure all the images are 100x 100
     test_ds = tf.keras.preprocessing.image_dataset_from_directory(
         test_dir,
         seed=42,
         image_size=(100, 100),
         batch_size=32
     )
```

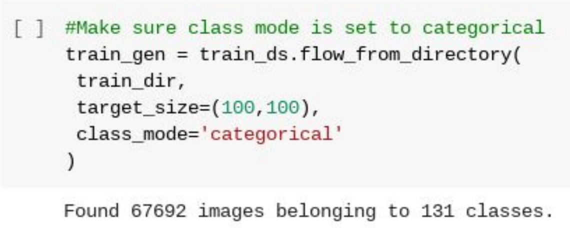
Data Preparation



```
#Preprocess our training Data
train_ds = ImageDataGenerator(
 rescale = 1./255,
 rotation_range=40,
 width_shift_range=0.2,
 height_shift_range=0.2,
 shear_range=0.2,
 zoom_range=0.2,
 horizontal_flip=True,
 fill_mode='nearest'
)
```

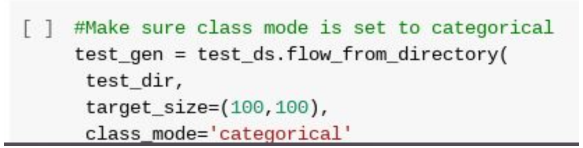


```
#Preprocess our test Data
test_ds = ImageDataGenerator(
 rescale = 1./255,
 rotation_range=40,
 width_shift_range=0.2,
 height_shift_range=0.2,
 shear_range=0.2,
 zoom_range=0.2,
 horizontal_flip=True,
 fill_mode='nearest'
)
```



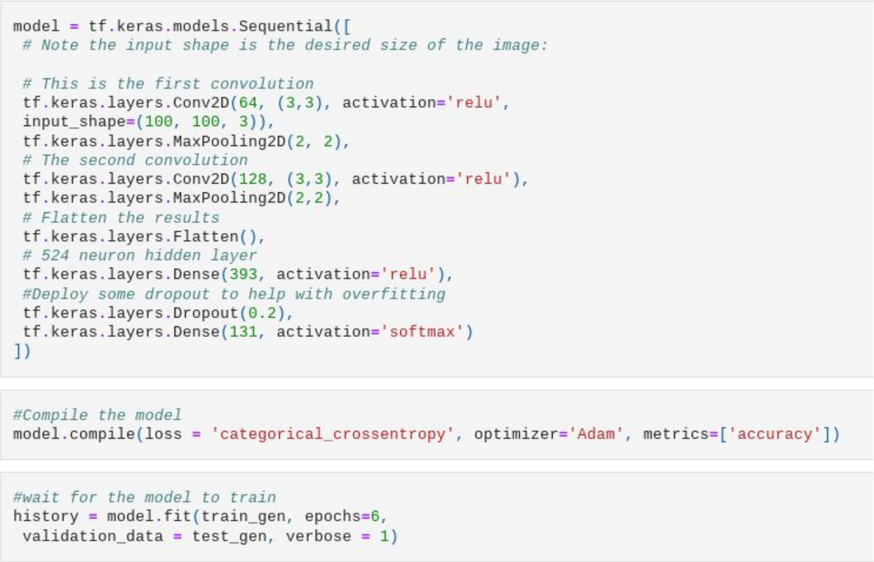
```
#Make sure class mode is set to categorical
train_gen = train_ds.flow_from_directory(
 train_dir,
 target_size=(100,100),
 class_mode='categorical'
)
```

```
Found 67692 images belonging to 131 classes.
```



```
#Make sure class mode is set to categorical
test_gen = test_ds.flow_from_directory(
 test_dir,
 target_size=(100,100),
 class_mode='categorical'
```

Building Our Models

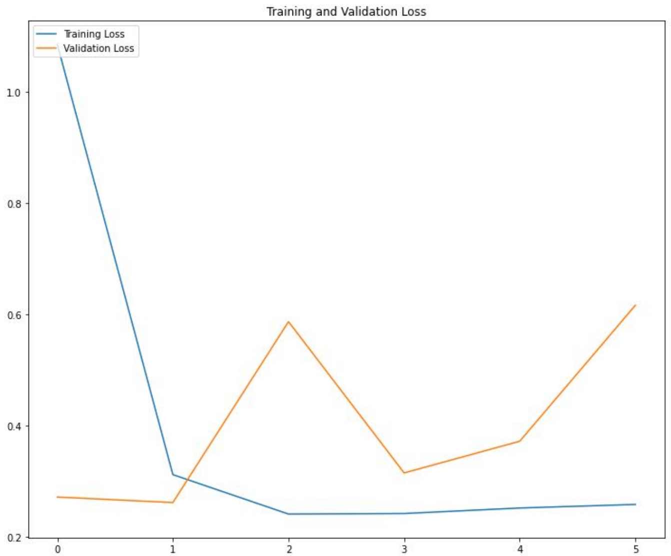


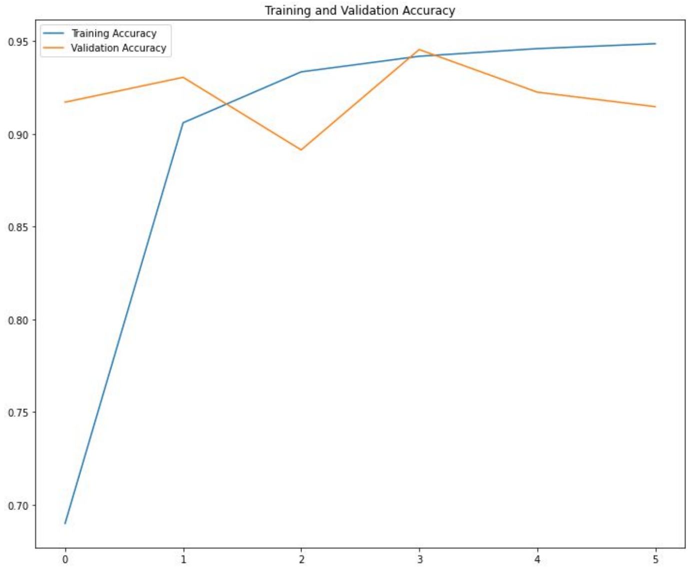
```
model = tf.keras.models.Sequential([
 # Note the input shape is the desired size of the image:

 # This is the first convolution
 tf.keras.layers.Conv2D(64, (3,3), activation='relu',
 input_shape=(100, 100, 3)),
 tf.keras.layers.MaxPooling2D(2, 2),
 # The second convolution
 tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
 tf.keras.layers.MaxPooling2D(2,2),
 # Flatten the results
 tf.keras.layers.Flatten(),
 # 524 neuron hidden layer
 tf.keras.layers.Dense(393, activation='relu'),
 #Deploy some dropout to help with overfitting
 tf.keras.layers.Dropout(0.2),
 tf.keras.layers.Dense(131, activation='softmax')
])
```

```
#Compile the model
model.compile(loss = 'categorical_crossentropy', optimizer='Adam', metrics=['accuracy'])
```

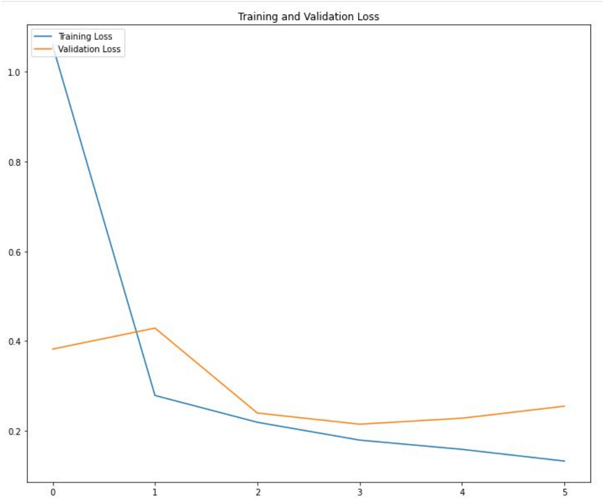
```
#wait for the model to train
history = model.fit(train_gen, epochs=6,
 validation_data = test_gen, verbose = 1)
```

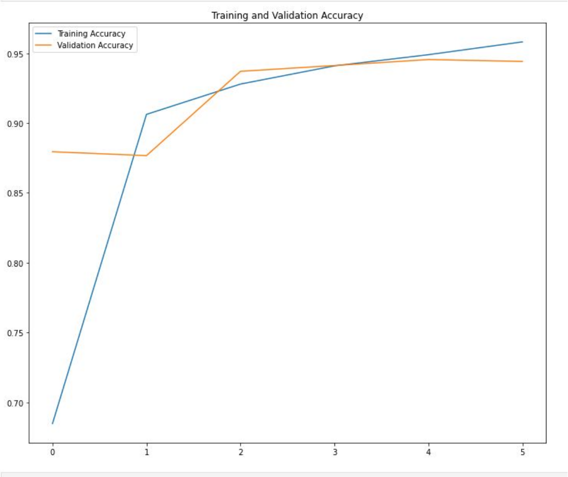
Model 1 - Optimizer RMSProp


Training and Validation Loss

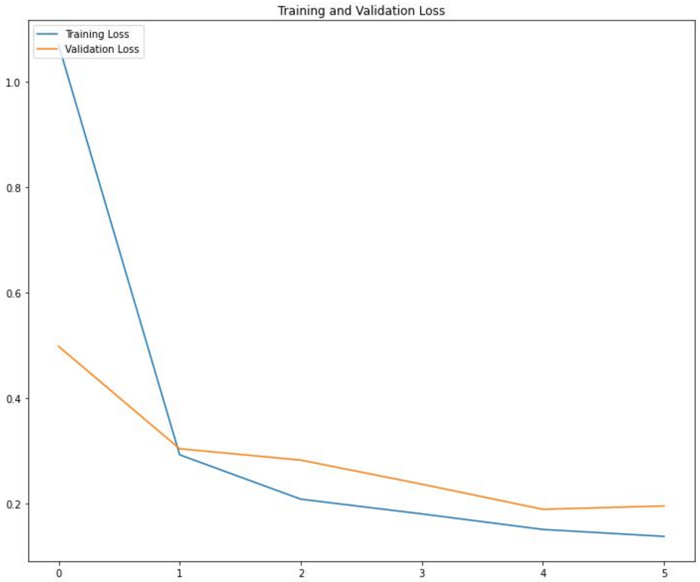

Training and Validation Accuracy

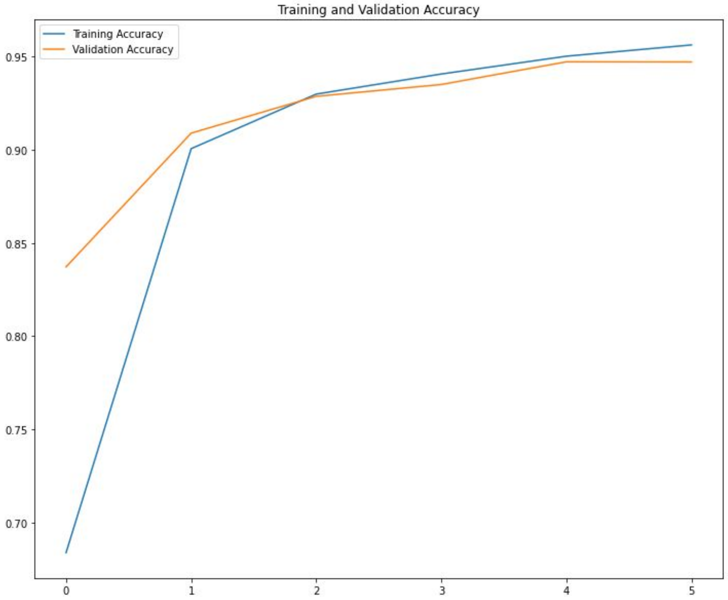
Model - 2 Optimizer ADAM V1





Model - 2 Optimizer ADAM V2





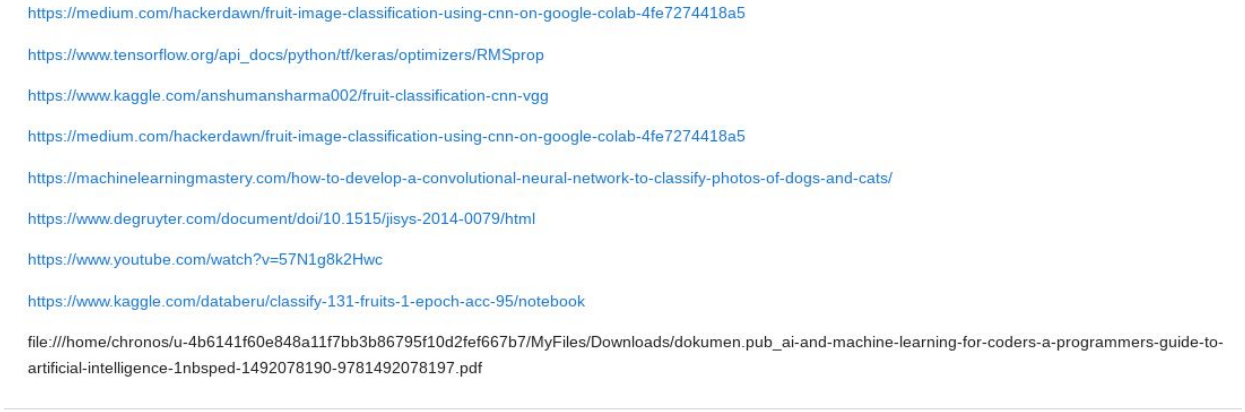
Conclusion

RMSprop - loss: 0.2587 - accuracy: 0.9485 - val_loss: 0.6163 - val_accuracy: 0.914

Adam V1 - loss: 0.1316 - accuracy: 0.9581 - val_loss: 0.2540 - val_accuracy: 0.944

Adam V2 - loss: 0.1380 - accuracy: 0.9561 - val_loss: 0.1957 - val_accuracy: 0.9470

Citations



https://medium.com/hackerdawn/fruit-image-classification-using-cnn-on-google-colab-4fe7274418a5

https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/RMSprop

https://www.kaggle.com/anshumansharma002/fruit-classification-cnn-vgg

https://medium.com/hackerdawn/fruit-image-classification-using-cnn-on-google-colab-4fe7274418a5

https://machinelearningmastery.com/how-to-develop-a-convolutional-neural-network-to-classify-photos-of-dogs-and-cats/

https://www.degruyter.com/document/doi/10.1515/jisys-2014-0079/html

https://www.youtube.com/watch?v=57N1g8k2Hwc

https://www.kaggle.com/databeru/classify-131-fruits-1-epoch-acc-95/notebook

file:///home/chronos/u-4b6141f60e848a11f7bb3b86795f10d2fef667b7/MyFiles/Downloads/dokumen.pub_ai-and-machine-learning-for-coders-a-programmers-guide-to-artificial-intelligence-1nbsped-1492078190-9781492078197.pdf