CNN & VGG16
Model Comparison

Content

Data

The type of dataset that we use and dataset properties with its preparation

Methods

The characteristics and usage of used methods in the project

Findings

The outcome of the program and how models differ

1. Data

Dataset

We use 3 types of apples in the dataset for comparison

1. Braeburn
2. Crimson Snow
3. Red Delicious

Braeburn

Crimson Snow

Red Delicious

Data Properties in Dataset

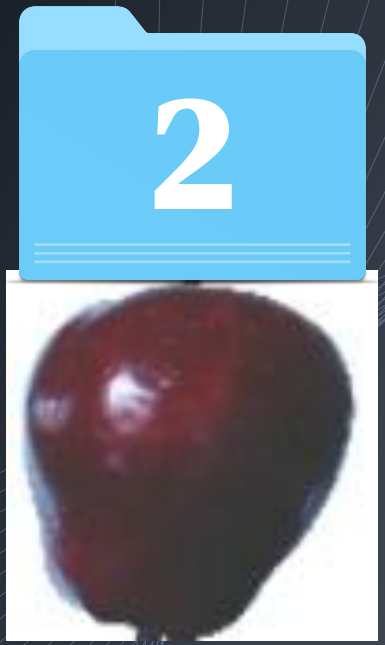
No Background

Taken From Many angles

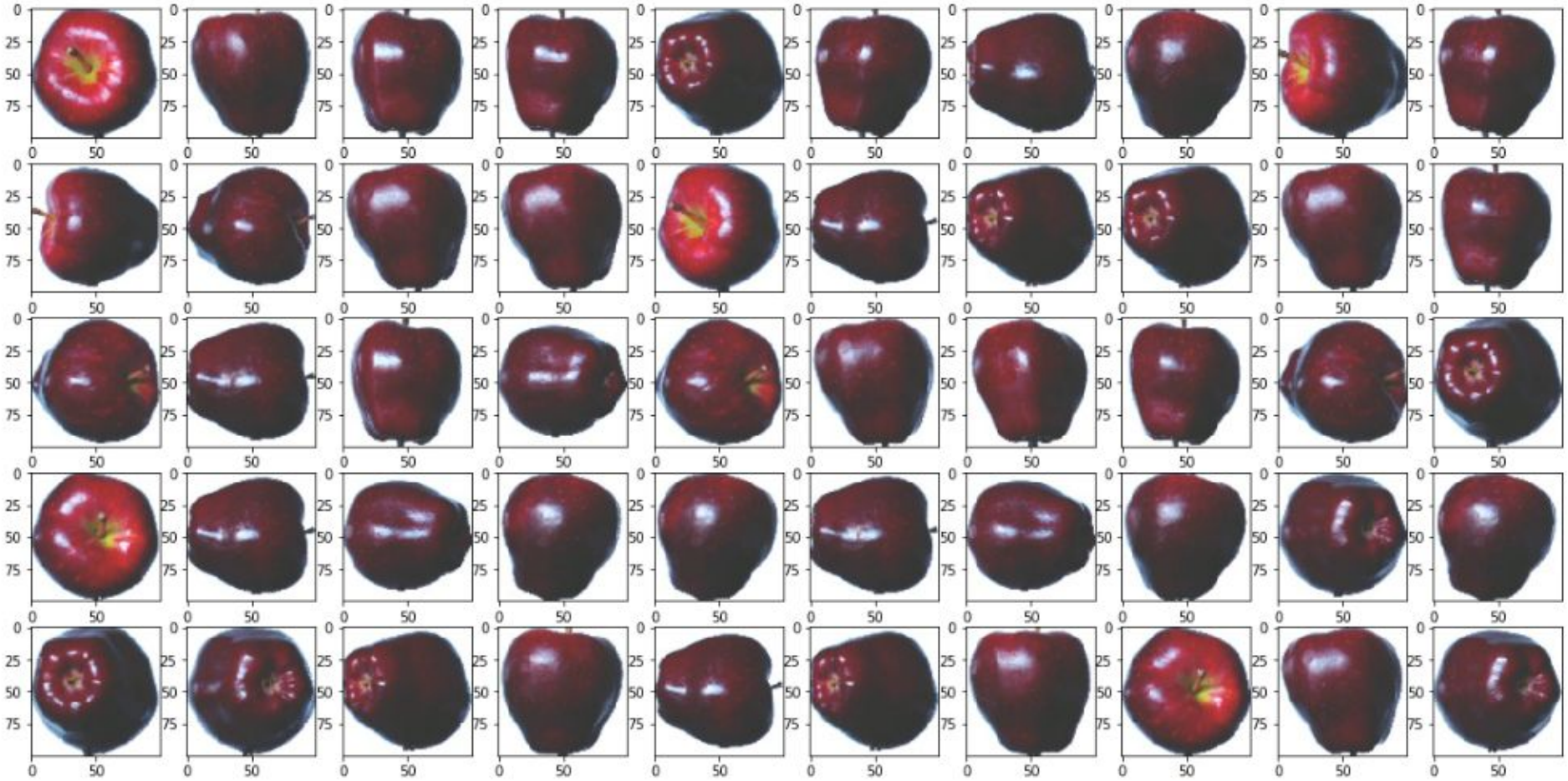
Data Preparation

Braeburn

Crimson Snow



Red Delicious



Randomize Data

```
['x_train_path'] image count: {0: 393, 1: 355, 2: 392} {0: 34.47, 1: 31.14, 2: 34.39}
['x_test_path'] image count: {0: 99, 1: 89, 2: 98} {0: 34.62, 1: 31.119999999999997, 2: 34.27}
```

	path	target
1295	Apple TrainDataset/2/r_241_100.jpg	2
238	Apple TrainDataset/0/314_100.jpg	0
1088	Apple TrainDataset/2/282_100.jpg	2
751	Apple TrainDataset/1/r_148_100.jpg	1
620	Apple TrainDataset/1/265_100.jpg	1
...	...	...
498	Apple TrainDataset/1/14_100.jpg	1
1177	Apple TrainDataset/2/97_100.jpg	2
135	Apple TrainDataset/0/221_100.jpg	0
1316	Apple TrainDataset/2/r_260_100.jpg	2
720	Apple TrainDataset/1/r_107_100.jpg	1

[1140 rows x 2 columns]

	path	target
151	Apple TrainDataset/0/236_100.jpg	0
1030	Apple TrainDataset/2/215_100.jpg	2
166	Apple TrainDataset/0/24_100.jpg	0
875	Apple TrainDataset/1/r_273_100.jpg	1
363	Apple TrainDataset/0/r_205_100.jpg	0
...	...	...
973	Apple TrainDataset/2/140_100.jpg	2
651	Apple TrainDataset/1/293_100.jpg	1
764	Apple TrainDataset/1/r_161_100.jpg	1
308	Apple TrainDataset/0/r_156_100.jpg	0
319	Apple TrainDataset/0/r_166_100.jpg	0

[286 rows x 2 columns]

Train: 80%

Test: 20%

Reshape/Binarize For CNN and VGG

In [10]:
```
import cv2
def importimg(df):
    IMG_SIZE = 64
    all_images = []
    label = []
    for i in range(len(df)):
        image_path = df.path.values[i]
        labels = df.target.values[i]
        img = cv2.imread(image_path)
        img = cv2.resize(img,(IMG_SIZE,IMG_SIZE))
        all_images.append(img)
        label.append(labels)
    train = np.array(all_images)
    tlabel = np.array(label)
    return train,tlabel
x_train,y_train = importimg(x_train_path)
x_test,y_test = importimg(x_test_path)

print('x_train shape:',x_train.shape,'x_test shape:',x_train.shape)
print('y_train shape:',y_train.shape,'y_test shape:',y_train.shape)
```

```
x_train shape: (1140, 64, 64, 3) x_test shape: (1140, 64, 64, 3)
y_train shape: (1140,) y_test shape: (1140,)
```

```
from keras.utils import to_categorical
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
print(y_train.shape,y_test.shape)
```

```
(1140, 3) (286, 3)
```

2. Methods

Our CNN

1. 3 Layers - 2 Convolutional layers, 1 Dense Layer
2. Use adam optimizer

Our CNN

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_7 (Conv2D)            (None, 62, 62, 64)        1792
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 60, 60, 32)        18464
_________________________________________________________________
flatten_4 (Flatten)          (None, 115200)            0
_________________________________________________________________
dense_4 (Dense)              (None, 3)                 345603
=================================================================
Total params: 365,859
Trainable params: 365,859
Non-trainable params: 0
```

VGG16

1. 16 Layers - 13 Convolutional layers, 3 Dense layers
2. 3x3 Kernel size
3. Focus on depth of layers
4. Created by Visual Geometry Group (VGG)

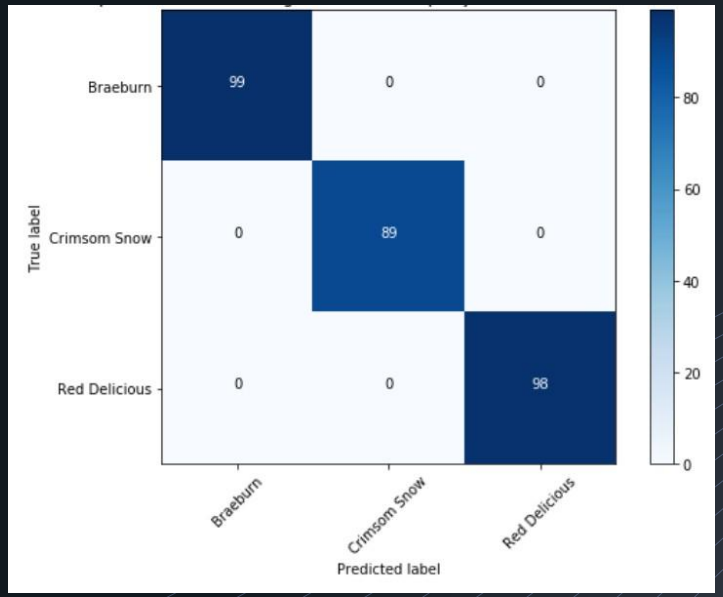
VGG16

```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Model)                (None, 2, 2, 512)         14714688
_________________________________________________________________
flatten_6 (Flatten)          (None, 2048)              0
_________________________________________________________________
dense_6 (Dense)              (None, 256)               524544
_________________________________________________________________
dense_7 (Dense)              (None, 3)                 771
=================================================================
Total params: 15,240,003
Trainable params: 15,240,003
Non-trainable params: 0
```

3. Evaluation

Confusion Matrix

CNN



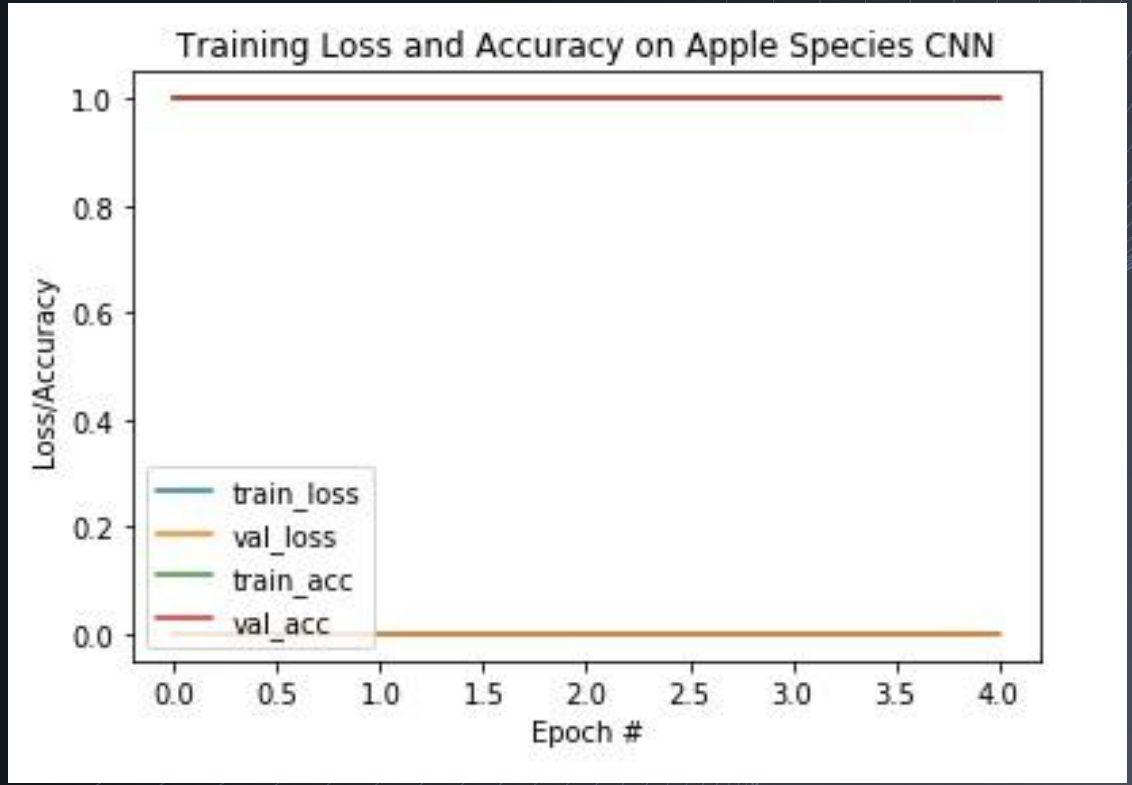
Results

CNN

	precision	recall	f1-score	support
Braeburn	1.00	1.00	1.00	99
Crimsom Snow	1.00	1.00	1.00	89
Red Delicious	1.00	1.00	1.00	98
accuracy			1.00	286
macro avg	1.00	1.00	1.00	286
weighted avg	1.00	1.00	1.00	286

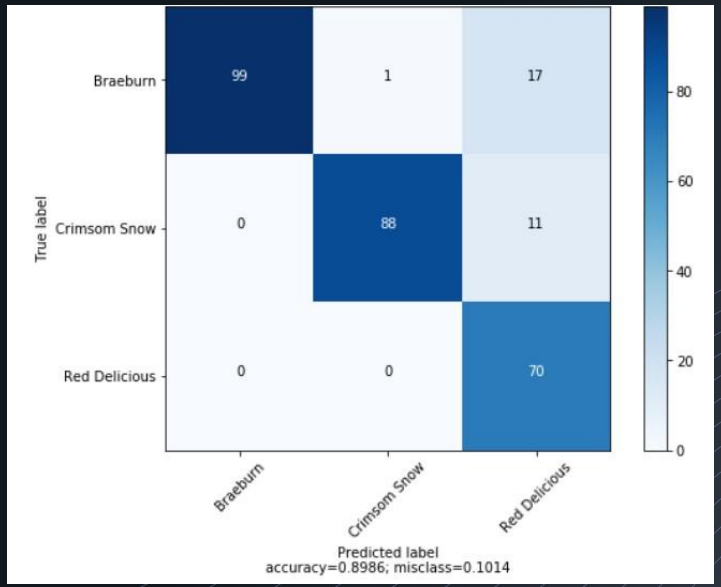
Loss

CNN


Training Loss and Accuracy on Apple Species CNN
Loss/Accuracy
Epoch #
train_loss
val_loss
train_acc
val_acc
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

Confusion Matrix

VGG16



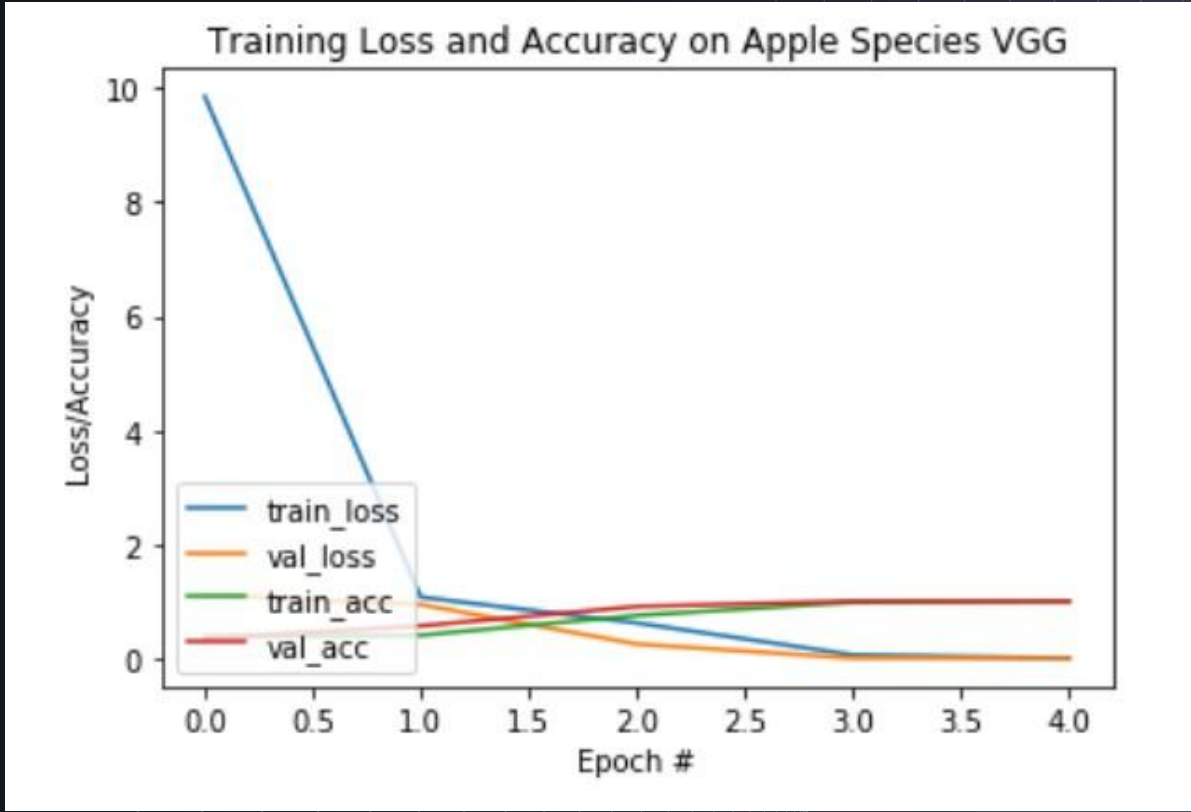
Results

VGG16

	precision	recall	f1-score	support
Braeburn	0.99	0.99	0.99	99
Crimsom Snow	0.96	0.99	0.97	86
Red Delicious	1.00	0.97	0.98	101
accuracy			0.98	286
macro avg	0.98	0.98	0.98	286
weighted avg	0.98	0.98	0.98	286

Loss

VGG16


Training Loss and Accuracy on Apple Species VGG
Loss/Accuracy
Epoch #
train_loss
val_loss
train_acc
val_acc
0
2
4
6
8
10
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

Demo