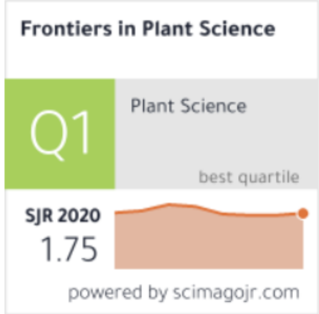
Pembelajaran Mesin Lanjut

Using Deep Learning for Image-Based Plant Disease Detection

Dewi Tresnawati - 33221003
Leni Fitriani - 33221034
Prof. Ir. Dwi Hendratmo Widyantoro, M.Sc., Ph.D
Sekolah Teknik Elektro dan Informatika



2016 – Q1 - Using Deep Learning for Image-Based Plant Disease Detection

Sharada P. Mohanty, David P. Hughes and Marcel Salathé

https://www.frontiersin.org/articles/10.3389/fpls.2016.01419/full

Problem

- Penyakit tanaman merupakan ancaman utama terhadap ketahanan pangan, tetapi identifikasi cepat penyakit ini masih sulit dilakukan di banyak bagian dunia karena kurangnya infrastruktur yang diperlukan.
- Kombinasi dari peningkatan penetrasi smartphone global dan kemajuan terbaru dalam Computer Vision yang dimungkinkan oleh Deep Learning telah membuka jalan bagi diagnosis penyakit yang dibantu oleh smartphone.

Contribution

- Menghasilkan model yang dapat dengan mudah diimplementasikan pada smartphone

Method

Choice of deep learning architecture

- AlexNet,
- GoogLeNet.

Choice of training mechanism:

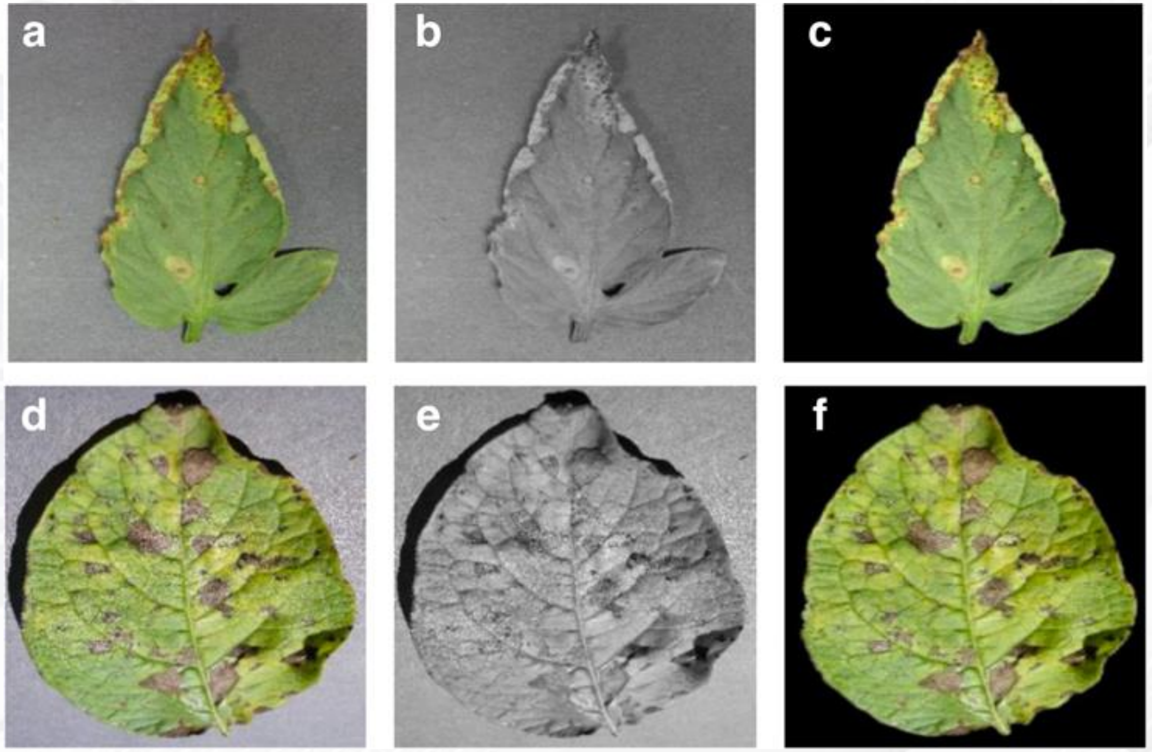
- Transfer Learning,
- Training from Scratch.

Choice of dataset type:

- Color,
- Gray scale,
- Leaf Segmented.

Choice of training-testing set distribution:

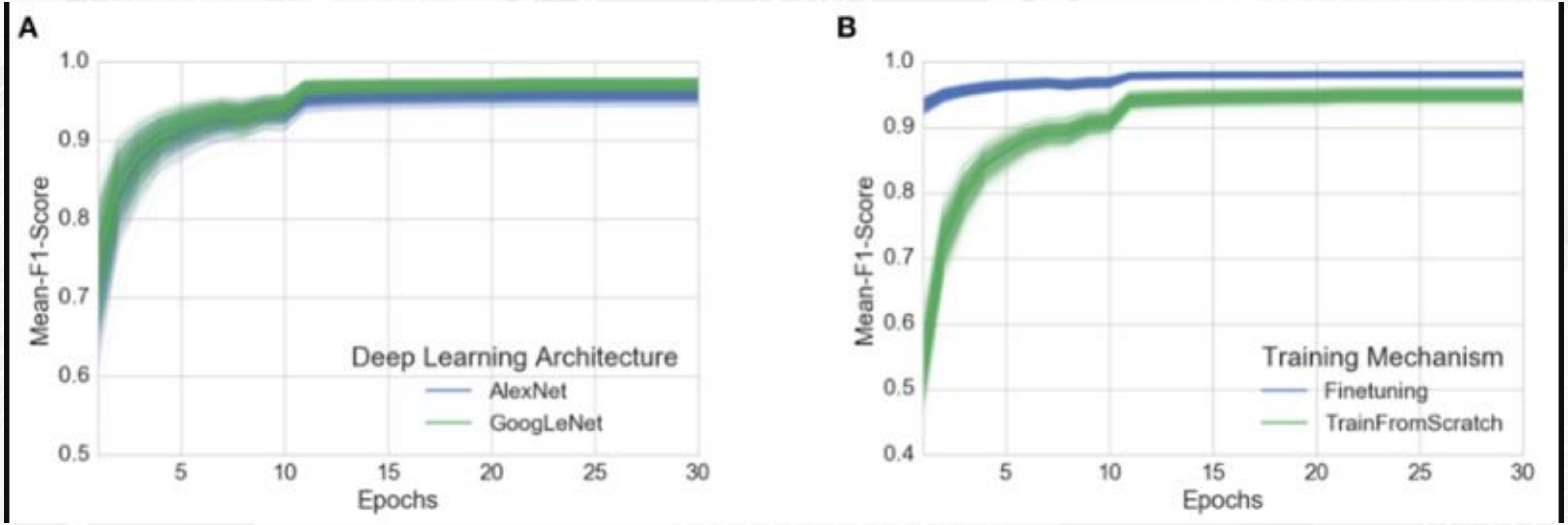
- Train: 80%, Test: 20%,
- Train: 60%, Test: 40%,
- Train: 50%, Test: 50%,
- Train: 40%, Test: 60%,
- Train: 20%, Test: 80%.



Sample images from the three different versions. (A) Leaf 1 color, (B) Leaf 1 grayscale, (C) Leaf 1 segmented, (D) Leaf 2 color, (E) Leaf 2 gray-scale, (F) Leaf 2 segmented.

Main Results

- Di antara arsitektur AlexNet dan GoogLeNet, GoogLeNet secara konsisten berkinerja lebih baik daripada AlexNet (Gambar A), dan berdasarkan metode pelatihan, pembelajaran transfer selalu memberikan hasil yang lebih baik (Gambar B).

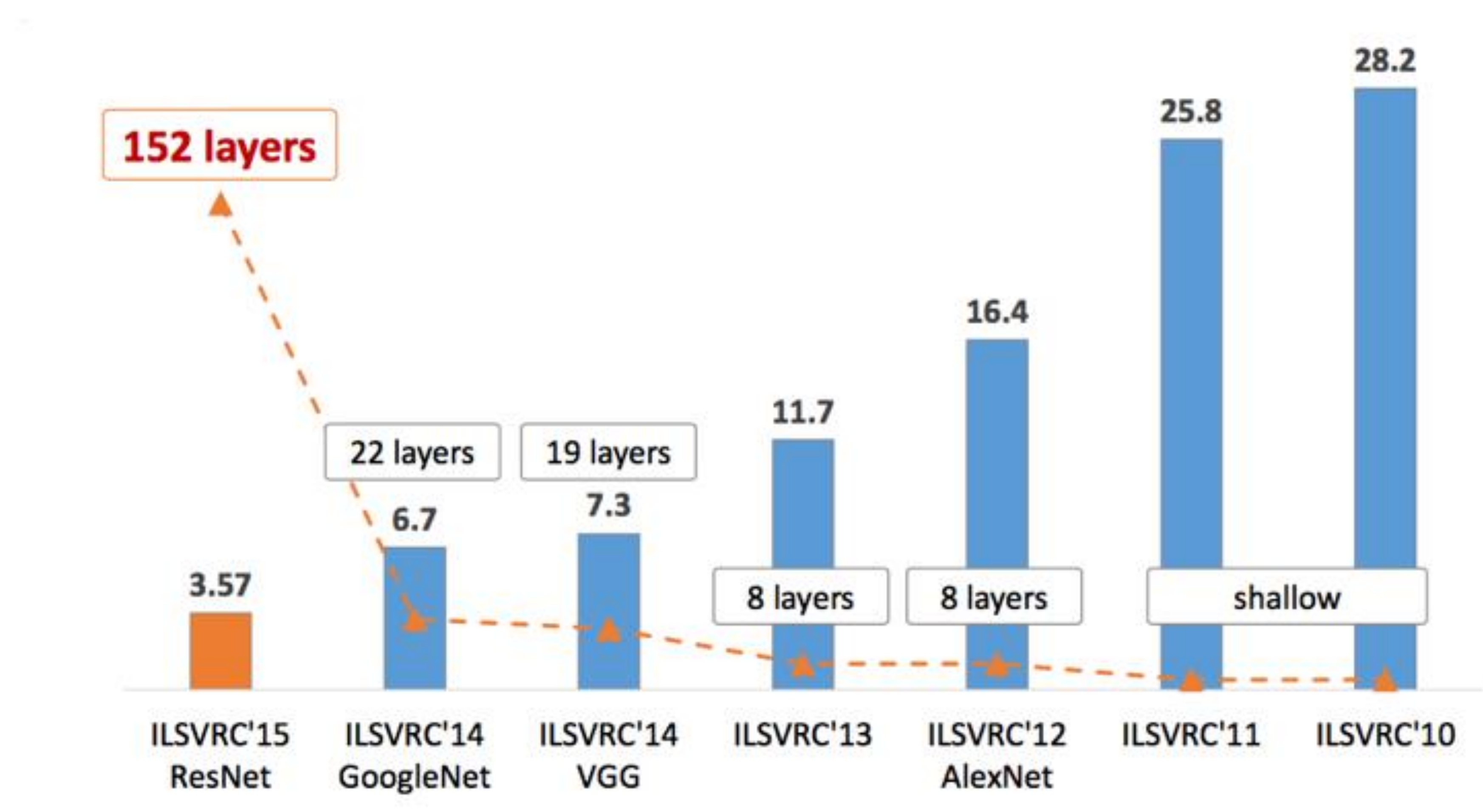


Limitation

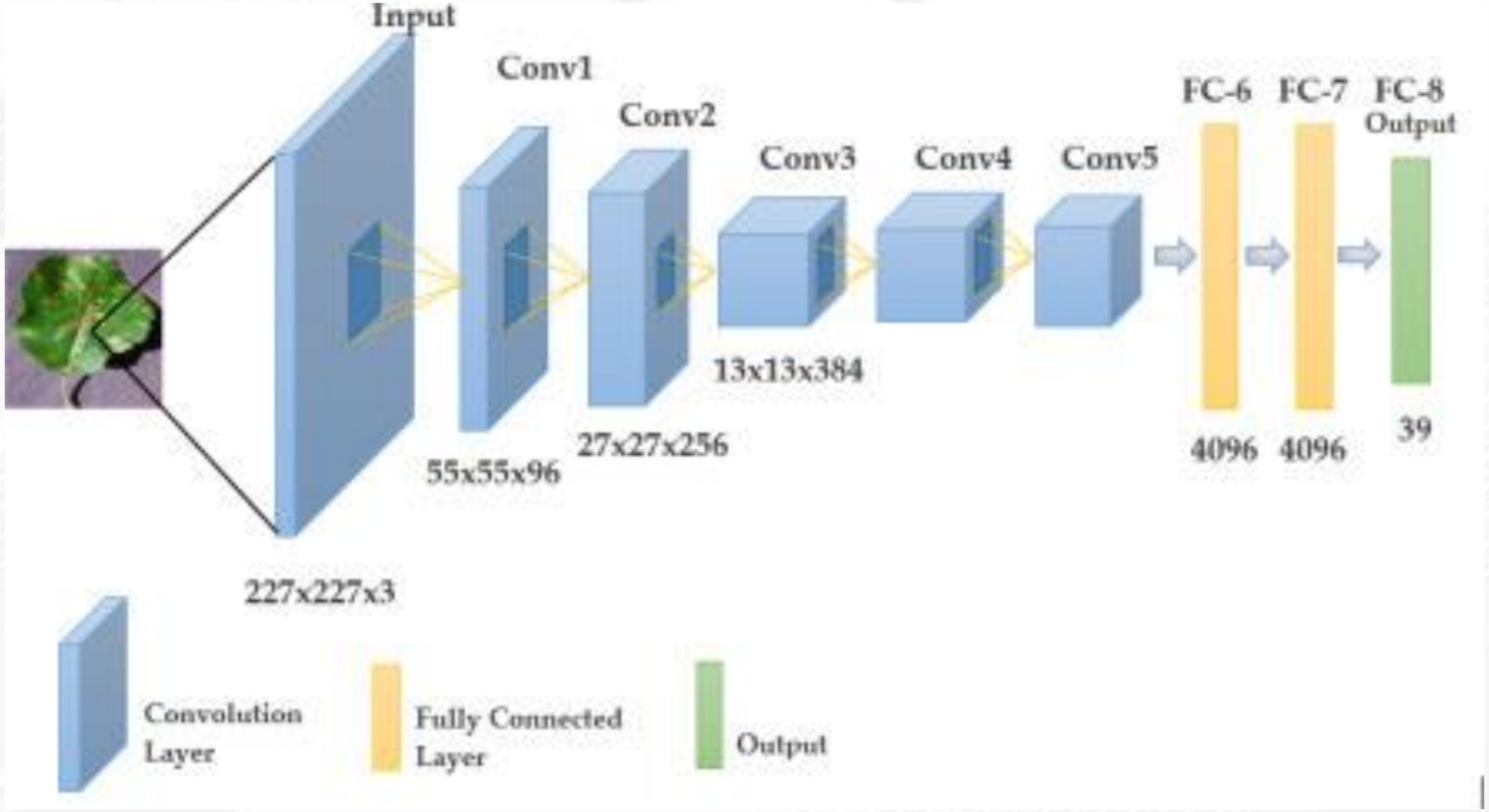
Pertama, ketika diuji pada sekumpulan gambar yang diambil dalam kondisi yang berbeda dari gambar yang digunakan untuk pelatihan, akurasi model berkurang secara substansial, hingga sedikit di atas 31%.

Keterbatasan kedua adalah dibatasi pada klasifikasi daun tunggal, menghadap ke atas, dengan latar belakang yang homogen. Meskipun ini adalah kondisi langsung, aplikasi dunia nyata harus dapat mengklasifikasikan gambar penyakit karena muncul sendiri secara langsung pada tanaman.

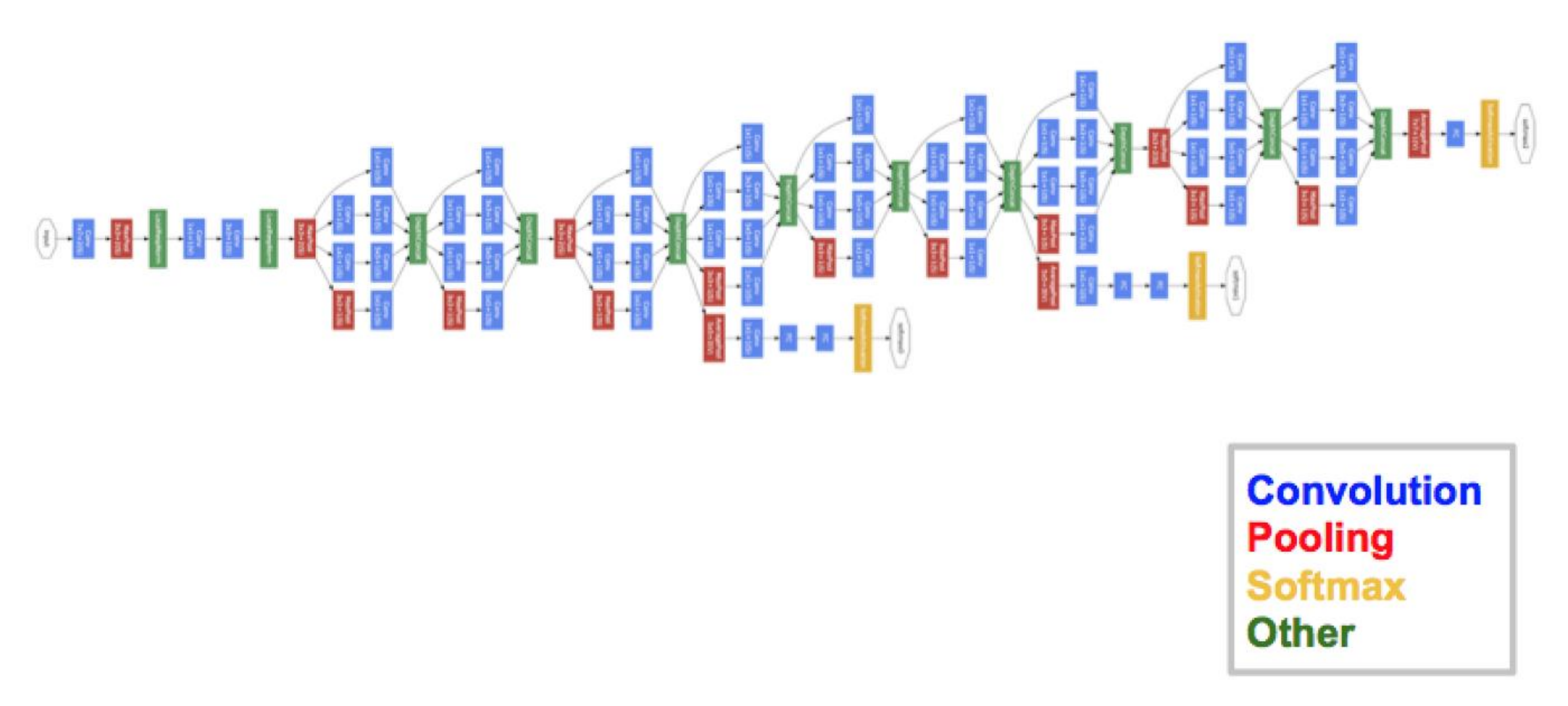
ImageNet Large Scale Visual Recognition Challenge (ILSVRC)



AlexNet

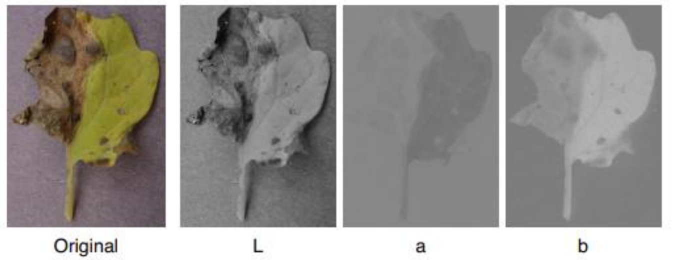

Input
Conv1
Conv2
Conv3
Conv4
Conv5
FC-6
FC-7
FC-8
Output
227x227x3
55x55x96
27x27x256
13x13x384
4096
4096
39
Convolution Layer
Fully Connected Layer
Output

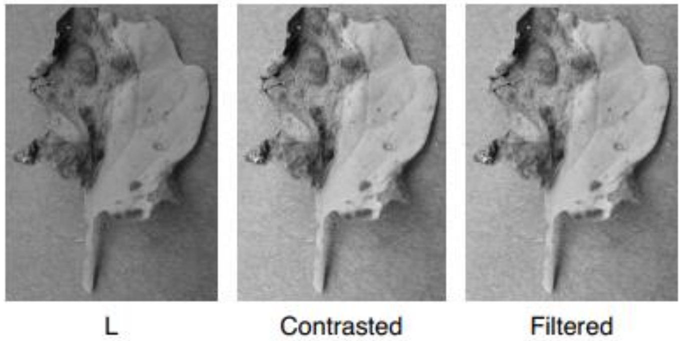
GoogLeNet



Segmentation

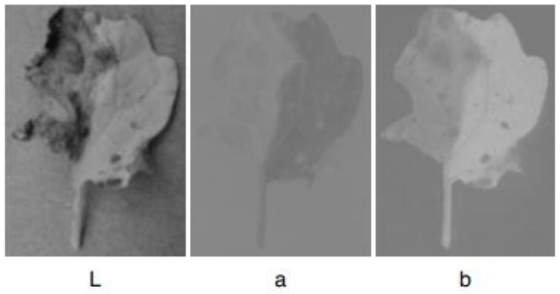
Conversion to Lab color space



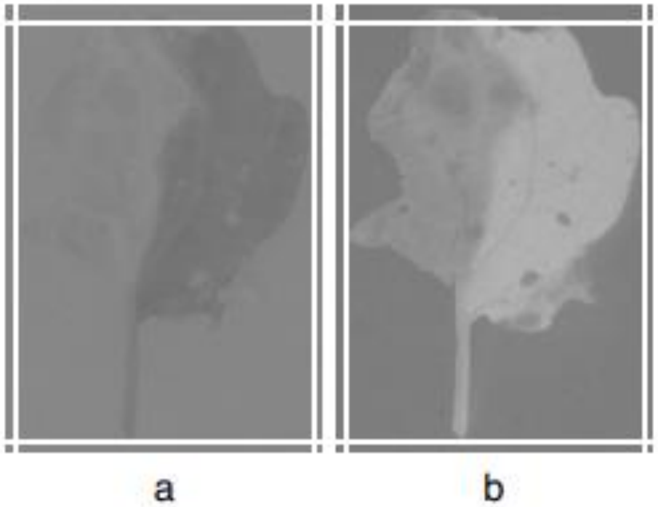


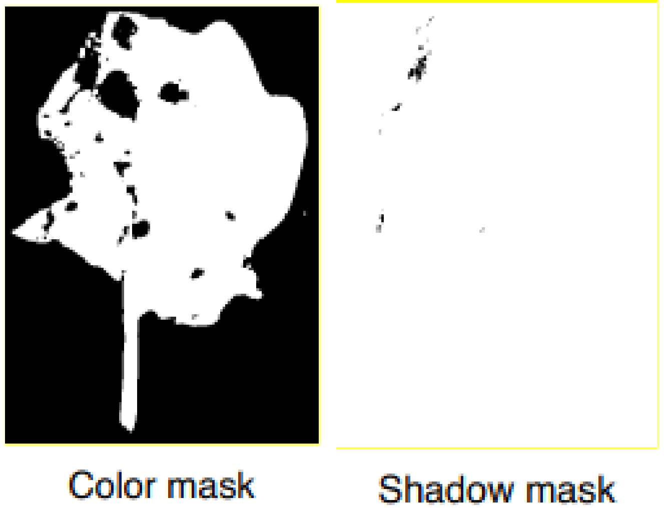
Blurring

Channels L, a and b are slightly blurred to decrease noise:

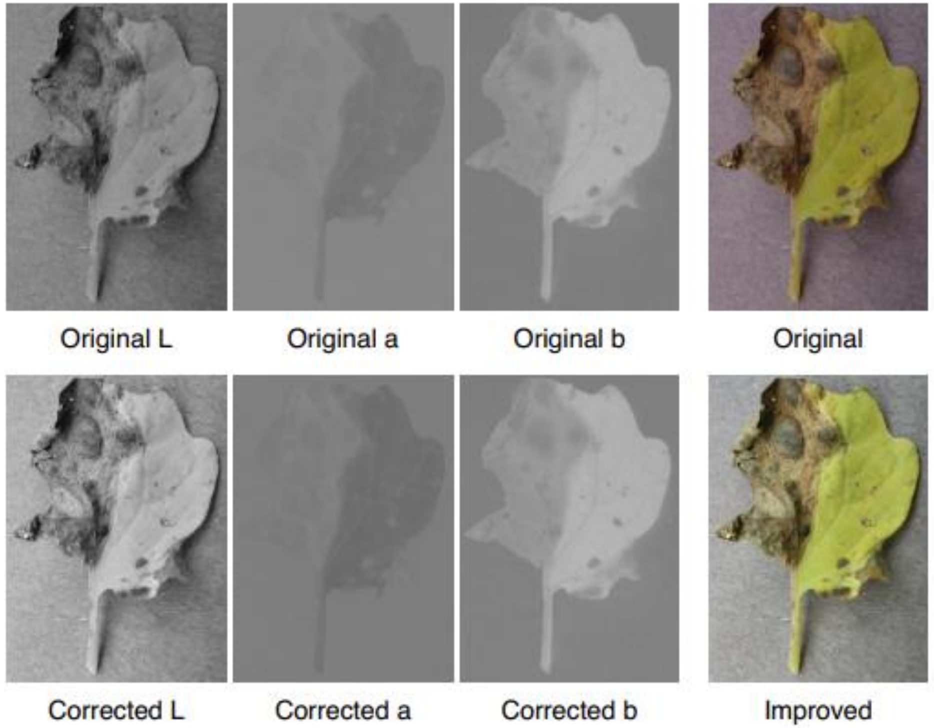


Edge cutting, Median Color



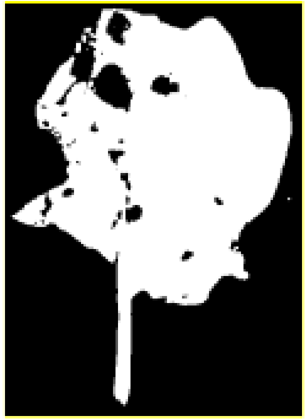


a and b channel correction and reconstruction



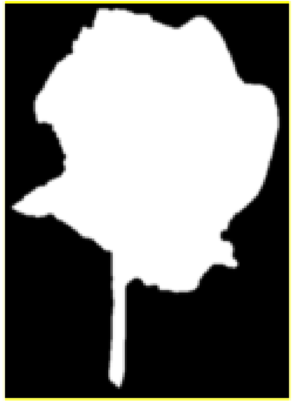
Final steps

Combine the masks together



Intermediate mask

Remove artefacts and detect leaf borders



Intermediate mask

Mask the corrected image

Original Corrected Final result

54,306 images of diseased and healthy plant leaves

Data Explorer

342.23 MB

- PlantVillage
  - Pepper__bell___Bacter...
  - Pepper__bell___healthy
  - Potato___Early_blight
  - Potato___Late_blight
  - Potato___healthy
  - Tomato_Bacterial_spot
  - Tomato_Early_blight
  - Tomato_Late_blight
  - Tomato_Leaf_Mold
  - Tomato_Septoria_leaf...
  - Tomato_Spider_mites...
  - Tomato__Target_Spot
  - Tomato__Tomato_Yell...
  - Tomato__Tomato_mos...
  - Tomato_healthy

In [1]:
```
import tensorflow as tf
from tensorflow.keras import models, layers
import matplotlib.pyplot as plt
```

In [2]:
```
IMAGE_SIZE = 256
BATCH_SIZE = 32
CHANNELS=3
EPOCHS=50
```

In [3]:
```
dataset = tf.keras.preprocessing.image_dataset_from_directory(
"PlantVillage",
shuffle=True,
image_size = (IMAGE_SIZE,IMAGE_SIZE),
batch_size = BATCH_SIZE
)
```

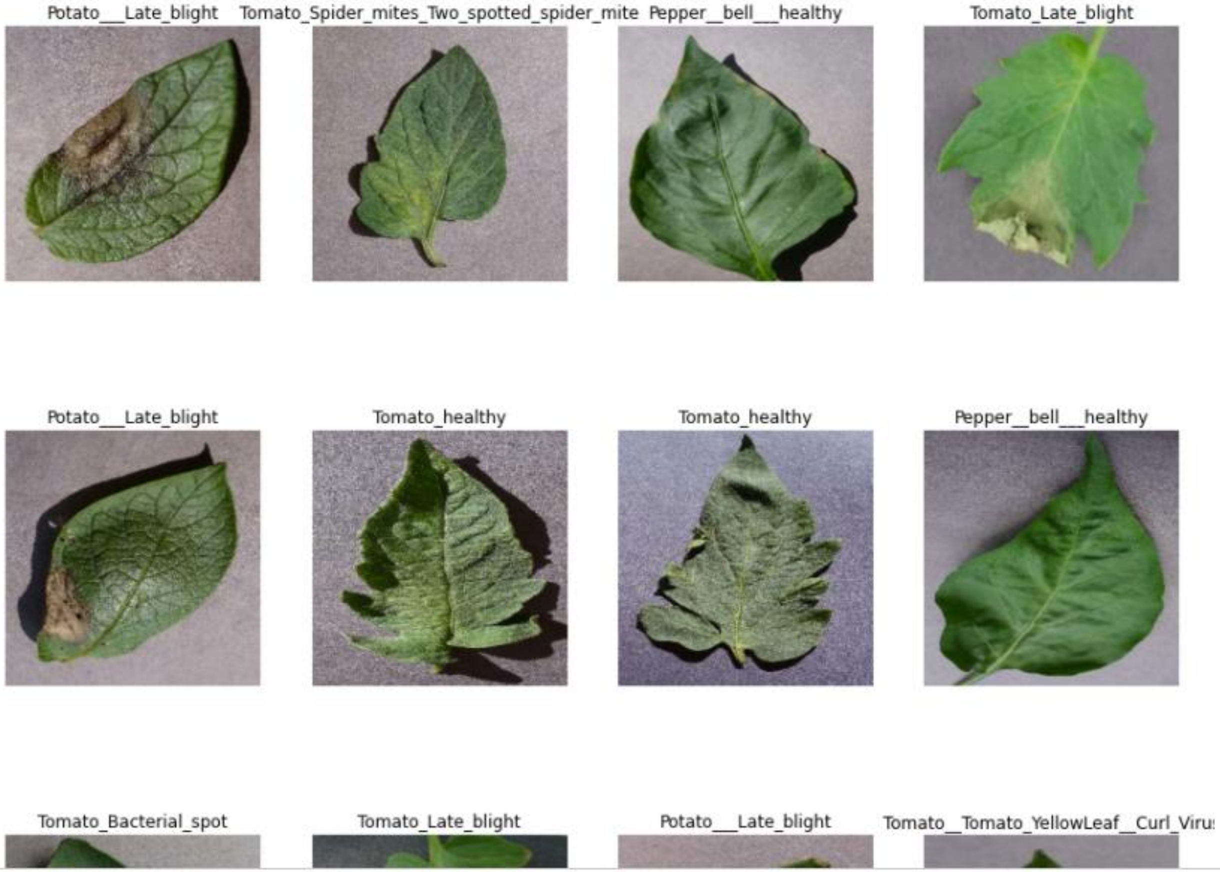
```
Found 20638 files belonging to 15 classes.
```

In [5]:
```
class_names = dataset.class_names
class_names
```

Out[5]:
```
['Pepper__bell___Bacterial_spot',
 'Pepper__bell___healthy',
 'Potato___Early_blight',
 'Potato___Late_blight',
 'Potato___healthy',
 'Tomato_Bacterial_spot',
 'Tomato_Early_blight',
 'Tomato_Late_blight',
 'Tomato_Leaf_Mold',
 'Tomato_Septoria_leaf_spot',
 'Tomato_Spider_mites_Two_spotted_spider_mite',
 'Tomato__Target_Spot',
 'Tomato__Tomato_YellowLeaf__Curl_Virus',
 'Tomato__Tomato_mosaic_virus',
 'Tomato_healthy']
```

```
80% ==> training
20% ==> 10% validation, 10% test
```

```python
plt.figure(figsize=(15, 15))
for image_batch, labels_batch in dataset.take(1):
    for i in range(12):
        ax = plt.subplot(3, 4, i + 1)
        plt.imshow(image_batch[i].numpy().astype("uint8"))
        plt.title(class_names[labels_batch[i]])
        plt.axis("off")
```




```python
#Applying Data Augmentation to Train Dataset
train_ds = train_ds.map(
    lambda x, y: (data_augmentation(x, training=True), y)
).prefetch(buffer_size=tf.data.AUTOTUNE)
```

```python
#Model Architecture
#use convolutional neural network (CNN)
input_shape = (BATCH_SIZE, IMAGE_SIZE, IMAGE_SIZE, CHANNELS)
n_classes = 3

model = models.Sequential([
    resize_and_rescale,
    layers.Conv2D(32, kernel_size = (3,3), activation='relu', input_shape=input_shape),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64,  kernel_size = (3,3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64,  kernel_size = (3,3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(n_classes, activation='softmax'),
])

model.build(input_shape=input_shape)

```

In [30]:

```
#Compiling the Model
#use adam Optimizer, SparseCategoricalCrossentropy for losses, accuracy as a metric
model.compile(
    optimizer='adam',
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
    metrics=['accuracy']
)
```

In [*]:

```
history = model.fit(
    train_ds,
    batch_size=BATCH_SIZE,
    validation_data=val_ds,
    verbose=1,
    epochs=50,
)
```

```
Epoch 1/50
```

In [1]:
```python
import tensorflow as tf
from tensorflow.keras import models, layers
import matplotlib.pyplot as plt
```

In [2]:
```python
IMAGE_SIZE = 256
BATCH_SIZE = 32
CHANNELS=3
EPOCHS=50
```

In [3]:
```python
dataset = tf.keras.preprocessing.image_dataset_from_directory(
"PlantVillage",
shuffle=True,
image_size = (IMAGE_SIZE,IMAGE_SIZE),
batch_size = BATCH_SIZE
)
```

```
Found 2152 files belonging to 3 classes.
```

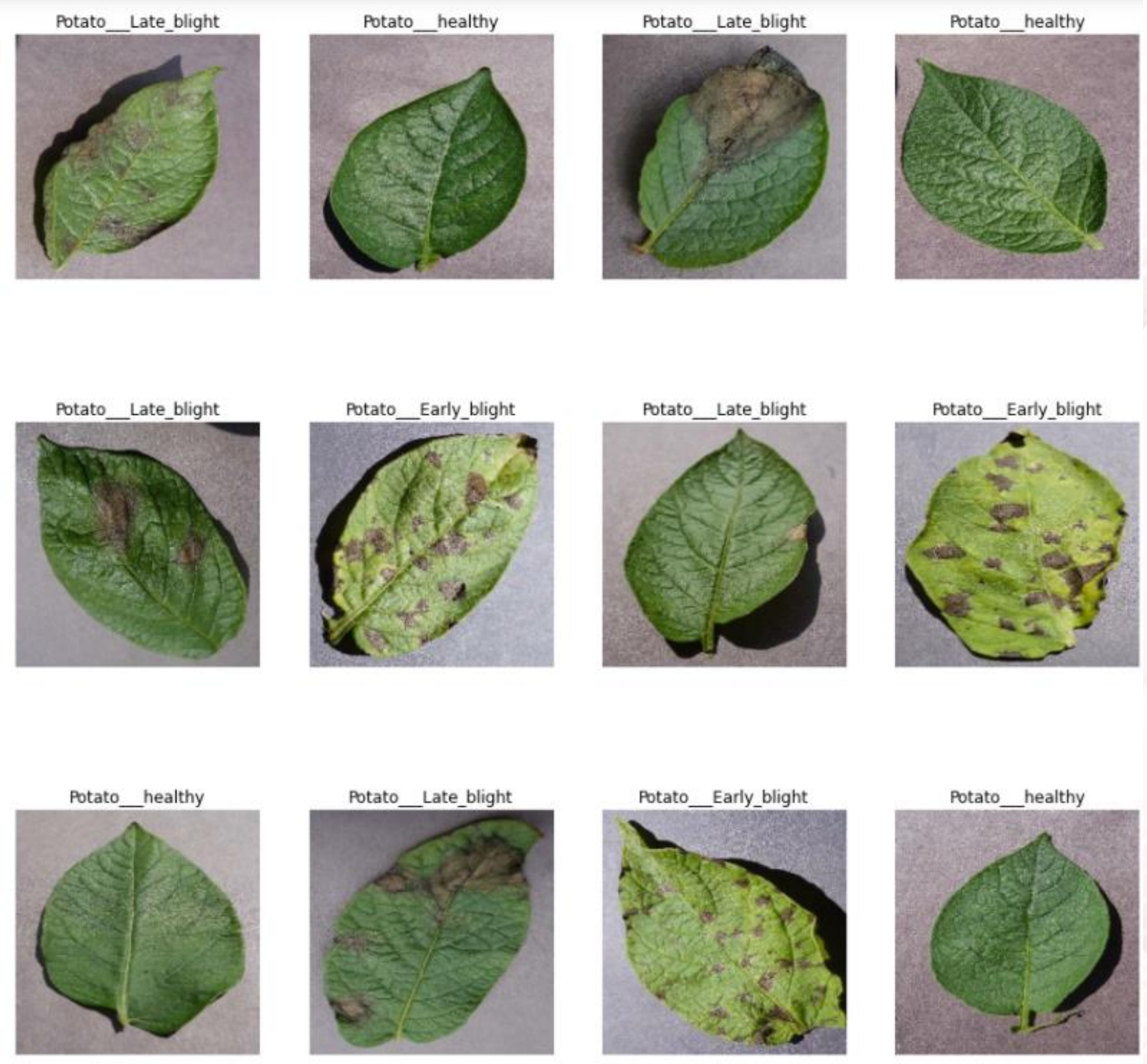
In [4]:
```python
class_names = dataset.class_names
class_names
```

Out[4]:
```
['Potato___Early_blight', 'Potato___Late_blight', 'Potato___healthy']
```

In [5]:
```python
len(dataset)
```

Out[5]:
```
68
```

In [7]:
```python
plt.figure(figsize=(15, 15))
for image_batch, labels_batch in dataset.take(1):
    for i in range(12):
        ax = plt.subplot(3, 4, i + 1)
        plt.imshow(image_batch[i].numpy().astype("uint8"))
        plt.title(class_names[labels_batch[i]])
        plt.axis("off")
```



In [17]:
```python
def get_dataset_partitions_tf(ds, train_split=0.8, val_split=0.1, test_split=0.1, shuffle=True, shuffle_size=10000):
    assert (train_split + test_split + val_split) == 1

    ds_size = len(ds)

    if shuffle:
        ds = ds.shuffle(shuffle_size, seed=12)

    train_size = int(train_split * ds_size)
    val_size = int(val_split * ds_size)

    train_ds = ds.take(train_size)
    val_ds = ds.skip(train_size).take(val_size)
    test_ds = ds.skip(train_size).skip(val_size)

    return train_ds, val_ds, test_ds
```

In [18]:
```python
train_ds, val_ds, test_ds = get_dataset_partitions_tf(dataset)
```

In [19]:
```python
len(train_ds)
```

Out[19]: 54

In [20]:
```python
len(val_ds)
```

Out[20]: 6

In [21]:
```python
len(test_ds)
```

Out[21]: 8

In [23]:
```python
#Building the Model
#Creating a Layer for Resizing and Normalization
resize_and_rescale = tf.keras.Sequential([
  layers.experimental.preprocessing.Resizing(IMAGE_SIZE, IMAGE_SIZE),
  layers.experimental.preprocessing.Rescaling(1./255),
])
```

In [24]:
```python
#Data Augmentation
data_augmentation = tf.keras.Sequential([
  layers.experimental.preprocessing.RandomFlip("horizontal_and_vertical"),
  layers.experimental.preprocessing.RandomRotation(0.2),
])
```

In [25]:
```python
#Applying Data Augmentation to Train Dataset
train_ds = train_ds.map(
    lambda x, y: (data_augmentation(x, training=True), y)
).prefetch(buffer_size=tf.data.AUTOTUNE)
```

In [44]:
```python
#Model Architecture
#use convolutional neural network (CNN)
input_shape = (BATCH_SIZE, IMAGE_SIZE, IMAGE_SIZE, CHANNELS)
n_classes = 3

model = models.Sequential([
    resize_and_rescale,
    layers.Conv2D(32, kernel_size = (3,3), activation='relu', input_shape=input_shape),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64,  kernel_size = (3,3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64,  kernel_size = (3,3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(n_classes, activation='softmax'),
])

model.build(input_shape=input_shape)

```

In [28]:
```python
#Compiling the Model
#use adam Optimizer, SparseCategoricalCrossentropy for losses, accuracy as a metric
model.compile(
    optimizer='adam',
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
    metrics=['accuracy']
)
```

In [29]:
```python
history = model.fit(
    train_ds,
    batch_size=BATCH_SIZE,
    validation_data=val_ds,
    verbose=1,
    epochs=50,
)
```

```
Epoch 1/50
54/54 [==============================] - 240s 4s/step - loss: 0.9092 - accuracy: 0.4659 - val_loss: 0.8229 - val_accuracy:
0.5156
Epoch 2/50
54/54 [==============================] - 231s 4s/step - loss: 0.7396 - accuracy: 0.6157 - val_loss: 0.5617 - val_accuracy:
0.7760
Epoch 3/50
54/54 [==============================] - 236s 4s/step - loss: 0.5750 - accuracy: 0.7402 - val_loss: 0.3903 - val_accuracy:
0.8177
Epoch 4/50
54/54 [==============================] - 261s 5s/step - loss: 0.4327 - accuracy: 0.7963 - val_loss: 0.3699 - val_accuracy:
0.8490
Epoch 5/50
54/54 [==============================] - 248s 5s/step - loss: 0.3862 - accuracy: 0.8385 - val_loss: 0.3214 - val_accuracy:
0.8906
Epoch 6/50
54/54 [==============================] - 249s 5s/step - loss: 0.3122 - accuracy: 0.8744 - val_loss: 0.2421 - val_accuracy:
0.9115
Epoch 7/50
```

In [30]:
```
scores = model.evaluate(test_ds)
```
8/8 [==============================] - 15s 1s/step - loss: 0.2242 - accuracy: 0.9492

In [31]:
```
scores
```
Out[31]: [0.22419798374176025, 0.94921875]

In [32]:
```
#Plotting the Accuracy and Loss Curves
history
```
Out[32]: <keras.callbacks.History at 0x296e9f1fa90>

In [33]:
```
history.params
```
Out[33]: {'verbose': 1, 'epochs': 50, 'steps': 54}

In [34]:
```
history.history.keys()
```
Out[34]: dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])

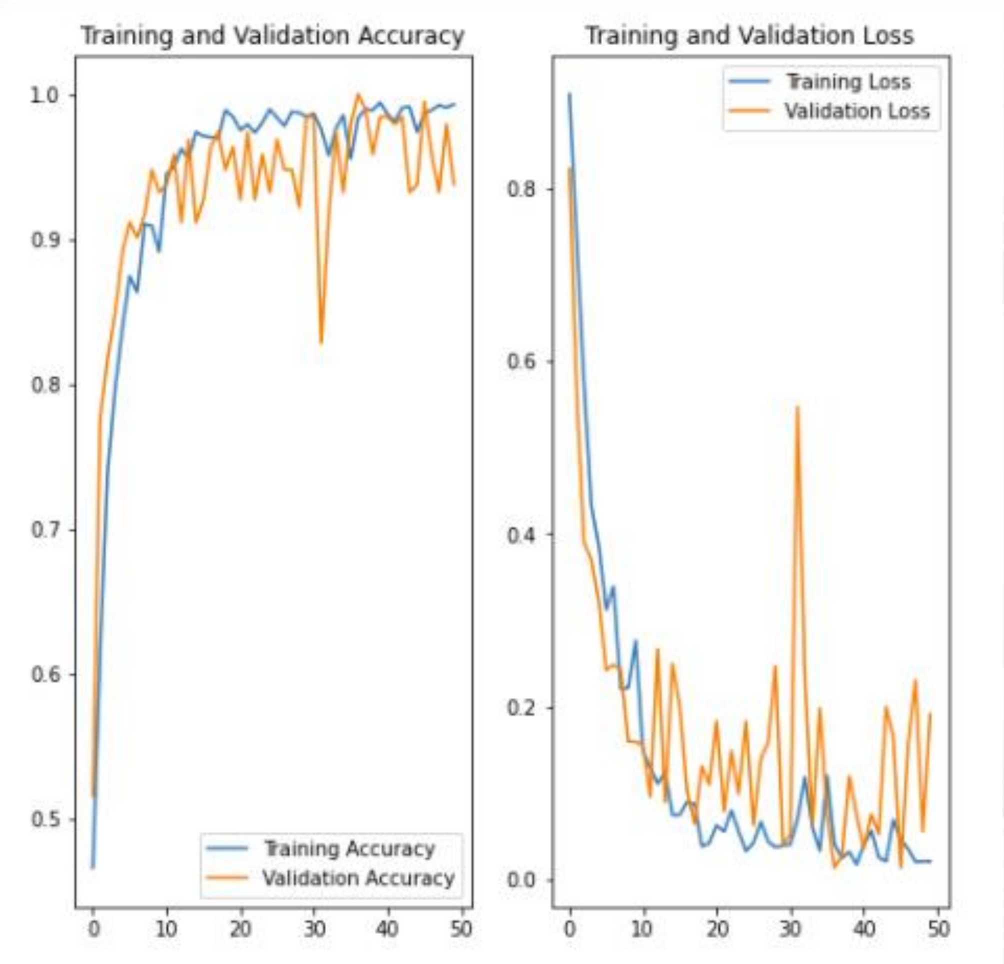
In [35]:
```
#loss, accuracy, val loss etc are a python list containing values of loss, accuracy etc at the end of each epoch
type(history.history['loss'])
```
Out[35]: list

In [36]:
```
len(history.history['loss'])
```
Out[36]: 50

In [37]:
```
history.history['loss'][:5] # show loss for first 5 epochs
```
Out[37]:
```
[0.9091859459877014,
 0.7396358847618103,
 0.5749866962432861,
 0.43272167444229126,
 0.3861580193042755]
```

In [38]:
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']
```

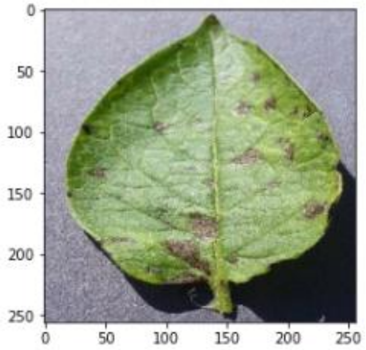
In [39]:
```
plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(range(EPOCHS), acc, label='Training Accuracy')
plt.plot(range(EPOCHS), val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(range(EPOCHS), loss, label='Training Loss')
plt.plot(range(EPOCHS), val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```



In [40]:

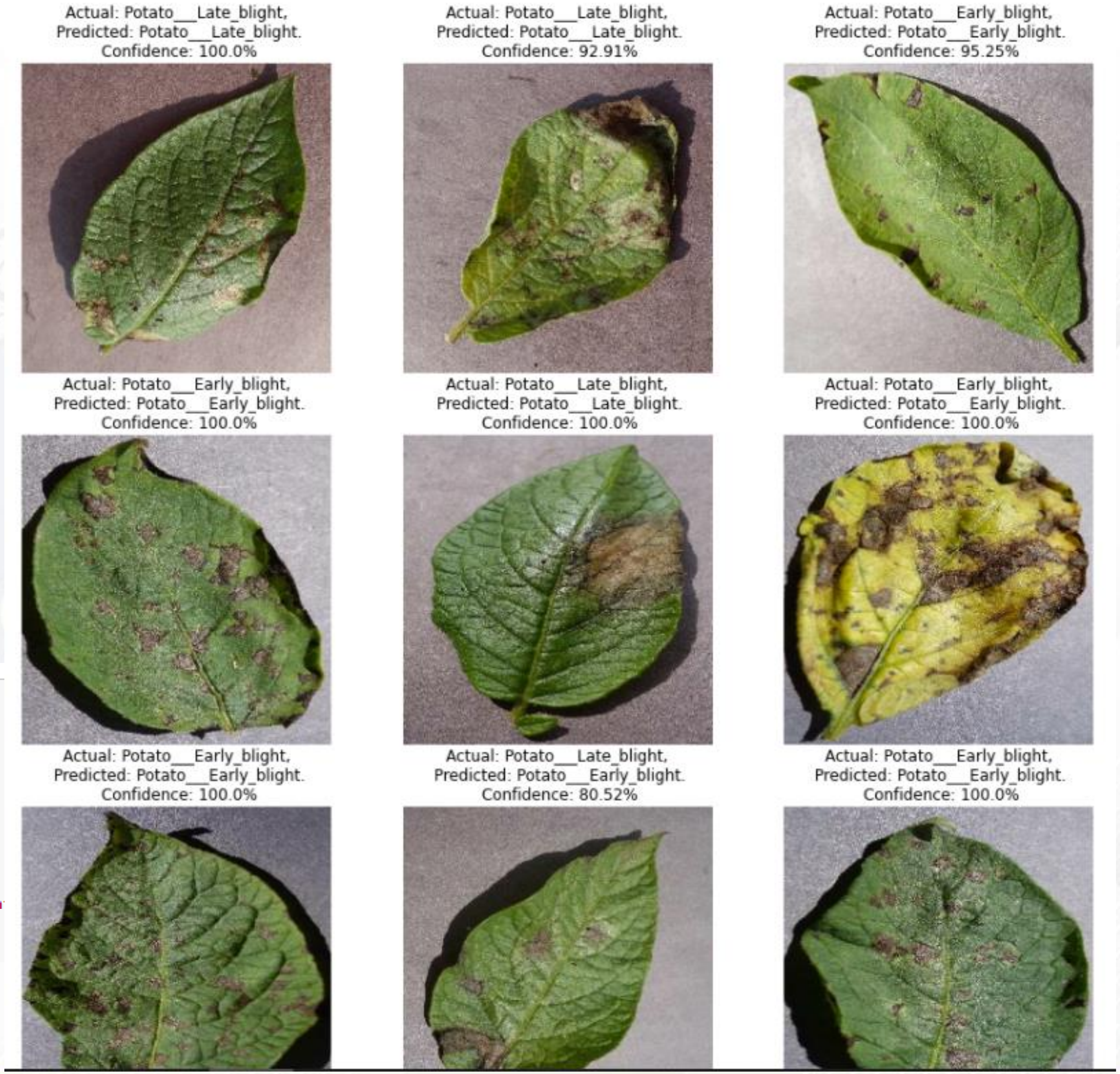
```python
#Run prediction on a sample image
import numpy as np
for images_batch, labels_batch in test_ds.take(1):

    first_image = images_batch[0].numpy().astype('uint8')
    first_label = labels_batch[0].numpy()

    print("first image to predict")
    plt.imshow(first_image)
    print("actual label:",class_names[first_label])

    batch_prediction = model.predict(images_batch)
    print("predicted label:",class_names[np.argmax(batch_prediction[0])])
```

```
first image to predict
actual label: Potato___Early_blight
predicted label: Potato___Early_blight
```



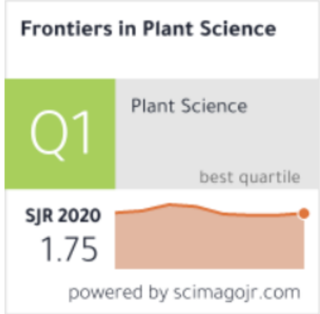
In [42]:

```python
#Now run inference on few sample images
plt.figure(figsize=(15, 15))
for images, labels in test_ds.take(1):
    for i in range(9):
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(images[i].numpy().astype("uint8"))

        predicted_class, confidence = predict(model, images[i].numpy())
        actual_class = class_names[labels[i]]

        plt.title(f"Actual: {actual_class},\n Predicted: {predicted_class}.\n Con

        plt.axis("off")
```



Within the PlantVillage data set of 54,306 images containing 38 classes of 14 crop species and 26 diseases (or absence thereof), this goal has been achieved as demonstrated by the top accuracy of 99.35%.

PlantVillage data set of 2152 images containing 3 classes of 1 crop species, this goal has been achieved as demonstrated by the top accuracy of 94.92%.



2016 – Q1 - Using Deep Learning for Image-Based Plant Disease Detection

Sharada P. Mohanty, David P. Hughes and Marcel Salathé

Within the PlantVillage data set of 54,306 images containing 38 classes of 14 crop species and 26 diseases (or absence thereof), this goal has been achieved as demonstrated by the top accuracy of 99.35%.

- PlantVillage data set of 2152 images containing 3 classes of 1 crop species, this goal has been achieved as demonstrated by the top accuracy of 94.92%. (CNN 50 epoch),
- AlexNet 10 epoch accuracy 66,41%

```
Epoch 7/10
54/54 [==============================] - 334s 6s/step - loss: 0.4447 - accuracy: 0.8692 - val_loss: 2.6832 - val_accuracy: 0.6172
Epoch 8/10
54/54 [==============================] - 334s 6s/step - loss: 0.4612 - accuracy: 0.8756 - val_loss: 0.8251 - val_accuracy: 0.7734
Epoch 9/10
54/54 [==============================] - 339s 6s/step - loss: 0.3005 - accuracy: 0.9074 - val_loss: 0.6854 - val_accuracy: 0.7461
Epoch 10/10
54/54 [==============================] - 338s 6s/step - loss: 0.2897 - accuracy: 0.9051 - val_loss: 2.8526 - val_accuracy: 0.6641
```

```
[31] scores = model.evaluate(test_ds)

8/8 [==============================] - 12s 1s/step - loss: 2.8526 - accuracy: 0.6641
```

AlexNet 10 epoch akurasi 66,41%

SEKIAN DAN TERIMA KASIH