Mobilenet | ibean (AI-Lab-Makerere)

Pigasin Dmitry

Dataset

Class	Examples
Healthy class	428
Angular Leaf Spot	432
Bean Rust	436
Total:	1,296

Structure

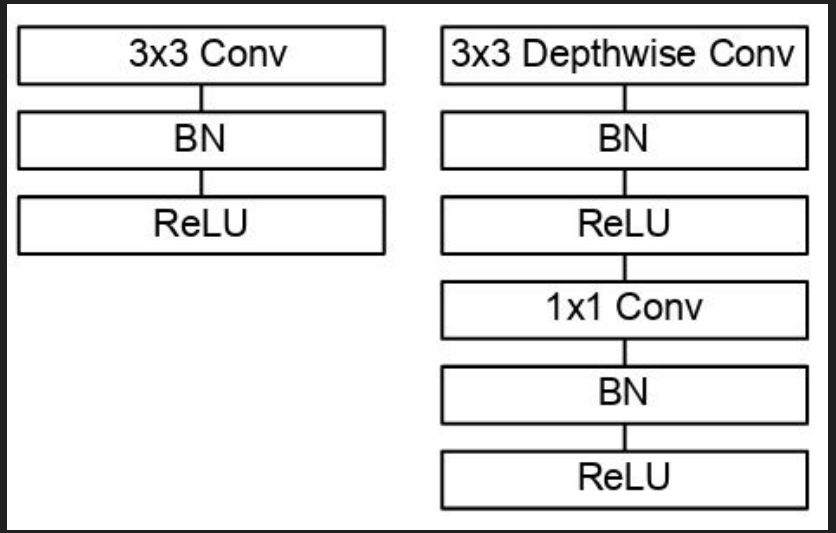
31 layers

```
Total params: 10,275
Trainable params: 9,795
Non-trainable params: 480
```

```
Input size | Output size

 0 conv2d
(None, 500, 500, 3) (None, 250, 250, 8)

 3 depthwise_conv2d
(None, 250, 250, 8) (None, 250, 250, 8)

 6 conv2d_1
(None, 250, 250, 8) (None, 250, 250, 16)

 9 depthwise_conv2d_1
(None, 250, 250, 16) (None, 125, 125, 16)

 12 conv2d_2
(None, 125, 125, 16) (None, 125, 125, 32)

 15 depthwise_conv2d_2
(None, 125, 125, 32) (None, 125, 125, 32)

 18 conv2d_3
(None, 125, 125, 32) (None, 125, 125, 32)

 21 depthwise_conv2d_3
(None, 125, 125, 32) (None, 63, 63, 32)

 24 conv2d_4
(None, 63, 63, 32) (None, 63, 63, 64)

 27 global_average_pooling2d
(None, 63, 63, 64) (None, 64)

 28 dense
(None, 64) (None, 64)

 29 dense_1
(None, 64) (None, 3)

 30 activation_9
(None, 3) (None, 3)
```

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Structure




```
def add_conv(filters):
    return (
        Conv2D(
            filters=filters,
            kernel_size=(1, 1),
            strides=(1, 1),
            padding='same'
        ),
        BatchNormalization(),
        Activation('relu'),
    )


def add_dw(strides):
    return (
        DepthwiseConv2D(
            kernel_size=(3, 3),
            strides=strides,
            padding='same'
        ),
        BatchNormalization(),
        Activation('relu'),
    )
```

```
def get_my_model(min_filter_count=8, max_filter_count=64, fc_layer_count=1):
    conv_layers = [
        Conv2D(filters=min_filter_count, kernel_size=(3, 3), strides=(2, 2),
               padding='same', input_shape=(img_width, img_height, 3)),
        BatchNormalization(),
        Activation('relu'),
    ]

    cur_filter_count = min_filter_count * 2
    while cur_filter_count < max_filter_count:
        conv_layers.extend([
            *add_dw(strides=1),
            *add_conv(filters=cur_filter_count),
        ])
        cur_filter_count *= 2
        conv_layers.extend([
            *add_dw(strides=2),
            *add_conv(filters=cur_filter_count),
        ])

    model = Sequential([
        *conv_layers,
        GlobalAveragePooling2D(),
        *[Dense(max_filter_count // (2 ** i), activation='relu') for i in range(fc_layer_count)],
        Dense(3),
        Activation('softmax'),
    ])
    model.compile(
        loss='categorical_crossentropy',
        optimizer='rmsprop',
        metrics=['accuracy'],
    )
    return model
```

Keras Tuner

```
def build_model(hp):
    min_filter_count = hp.Choice('min_filter_count',
                                 values=[4, 8, 16, 32, 64])
    max_filter_count = hp.Choice('max_filter_count',
                                 values=[32, 64, 128, 256, 512, 1024])
    fc_layer_count = hp.Choice('fc_layer_count',
                               values=[1, 2, 3, 4])
    return get_my_model(min_filter_count, max_filter_count, fc_layer_count)


tuner = Hyperband(
    build_model,
    objective='val_accuracy',
    max_epochs=30,
    directory='models',
    project_name='lab2',
)
```

Search space summary

|-Default search space size: 3

min_filter_count (Choice)

|-default: 4
|-ordered: True
|-values: [4, 8, 16, 32, 64]

max_filter_count (Choice)

|-default: 32
|-ordered: True
|-values: [32, 64, 128, 256, 512, 1024]

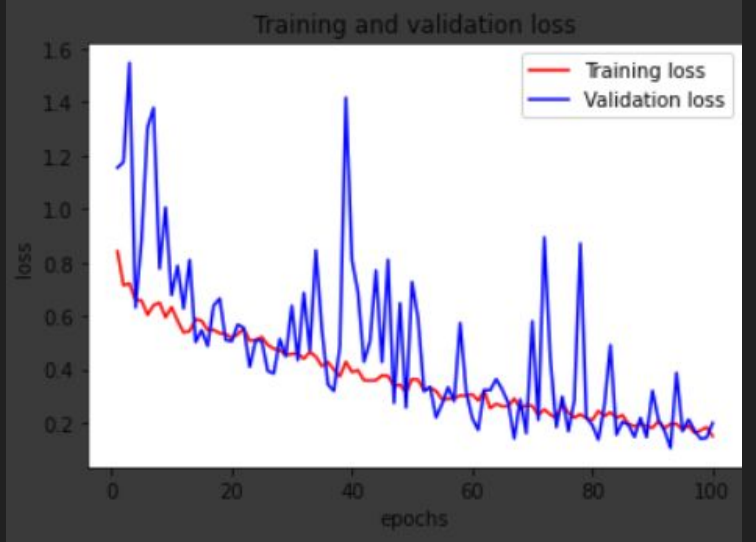
fc_layer_count (Choice)

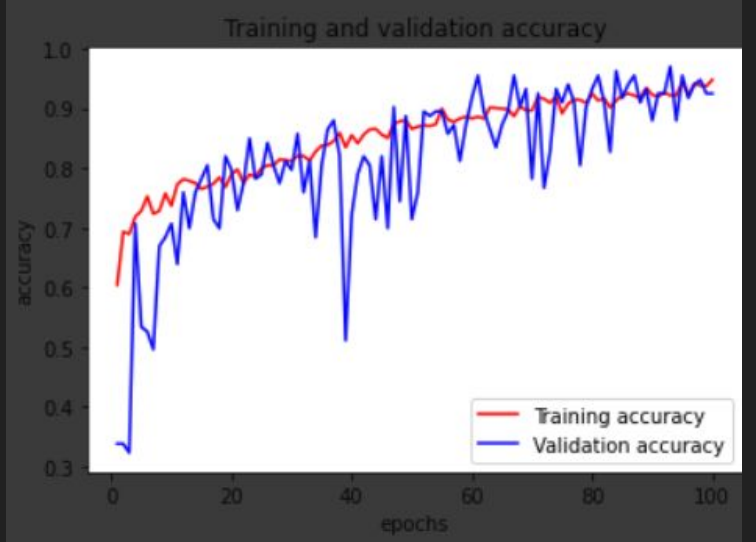
|-default: 1
|-ordered: True
|-values: [1, 2, 3, 4]

Model training

```
checkpoint = ModelCheckpoint(
    os.path.join(lab2_root, 'best_epoch.h5'),
    save_best_only=True,
    monitor='val_accuracy',
    mode='max',
    verbose=2,
)

fit_history = model.fit(
    train_generator,
    epochs=100,
    validation_data=validation_generator,
    verbose=2,
    callbacks=[checkpoint],
)
```

```
100 epochs - loss: 0.2237 - accuracy: 0.9141
Best_epoch - loss: 0.3018 - accuracy: 0.8906
```





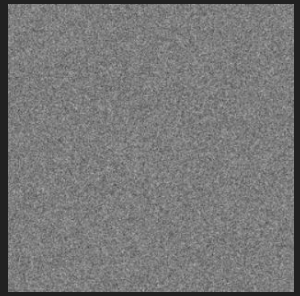
Test

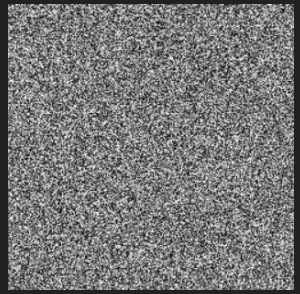
```
def preprocess_image(image):
    return np.expand_dims(img_to_array(image), axis=0) / 255.


image_paths = [os.path.join(dir, random.choice(os.listdir(dir))) for dir in image_dirs * 3]
classes = ['angular_leaf_spot', 'bean_rust', 'healthy']

for path in image_paths:
    out = model.predict(preprocess_image(load_img(path)))[0]
    print(dict(zip(classes, [round(x, 2) for x in out])), path.split('/')[-1])
```

```
-----------------------------------------out-----------------------------------------
{'angular_leaf_spot': 0.98, 'bean_rust': 0.02, 'healthy': 0.0} angular_leaf_spot_train.259.jpg
{'angular_leaf_spot': 0.01, 'bean_rust': 0.99, 'healthy': 0.0} bean_rust_train.80.jpg
{'angular_leaf_spot': 0.0, 'bean_rust': 0.03, 'healthy': 0.97} healthy_train.41.jpg
{'angular_leaf_spot': 0.99, 'bean_rust': 0.0, 'healthy': 0.01} angular_leaf_spot_train.17.jpg
{'angular_leaf_spot': 0.0, 'bean_rust': 1.0, 'healthy': 0.0} bean_rust_train.41.jpg
{'angular_leaf_spot': 0.0, 'bean_rust': 0.0, 'healthy': 1.0} healthy_train.231.jpg
{'angular_leaf_spot': 0.82, 'bean_rust': 0.18, 'healthy': 0.0} angular_leaf_spot_train.214.jpg
{'angular_leaf_spot': 0.0, 'bean_rust': 1.0, 'healthy': 0.0} bean_rust_train.75.jpg
{'angular_leaf_spot': 0.0, 'bean_rust': 0.0, 'healthy': 1.0} healthy_train.230.jpg
```

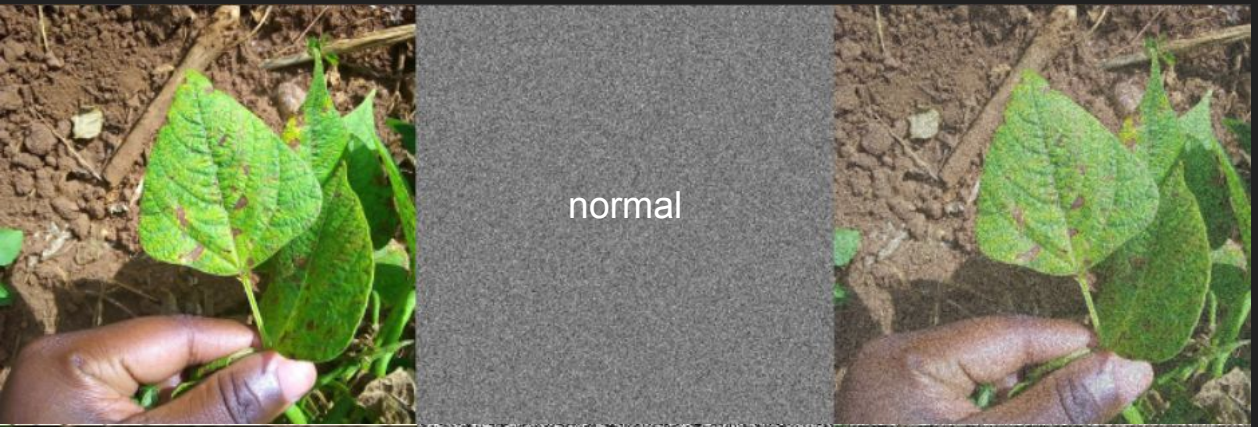
Test with noise

```
S = 1

def get_normal_noise(image):
    noise = np.random.normal(128, 20, (image.shape[0], image.shape[1]))
    return np.dstack((noise, noise, noise)).astype(np.uint8)

def add_normal_noise(image):
    noise = get_normal_noise(image) * S
    noise_image = cv2.add(image.astype(np.float64), noise.astype(np.float64))
    cv2.normalize(noise_image, noise_image, 0, 255, cv2.NORM_MINMAX)
    return noise_image

def get_uniform_noise(image):
    noise = np.random.uniform(0, 255, (image.shape[0], image.shape[1]))
    return np.dstack((noise, noise, noise)).astype(np.uint8)

def add_uniform_noise(image):
    noise = get_uniform_noise(image) * S
    noise_image = cv2.add(image.astype(np.float64), noise.astype(np.float64))
    cv2.normalize(noise_image, noise_image, 0, 255, cv2.NORM_MINMAX)
    return noise_image
```





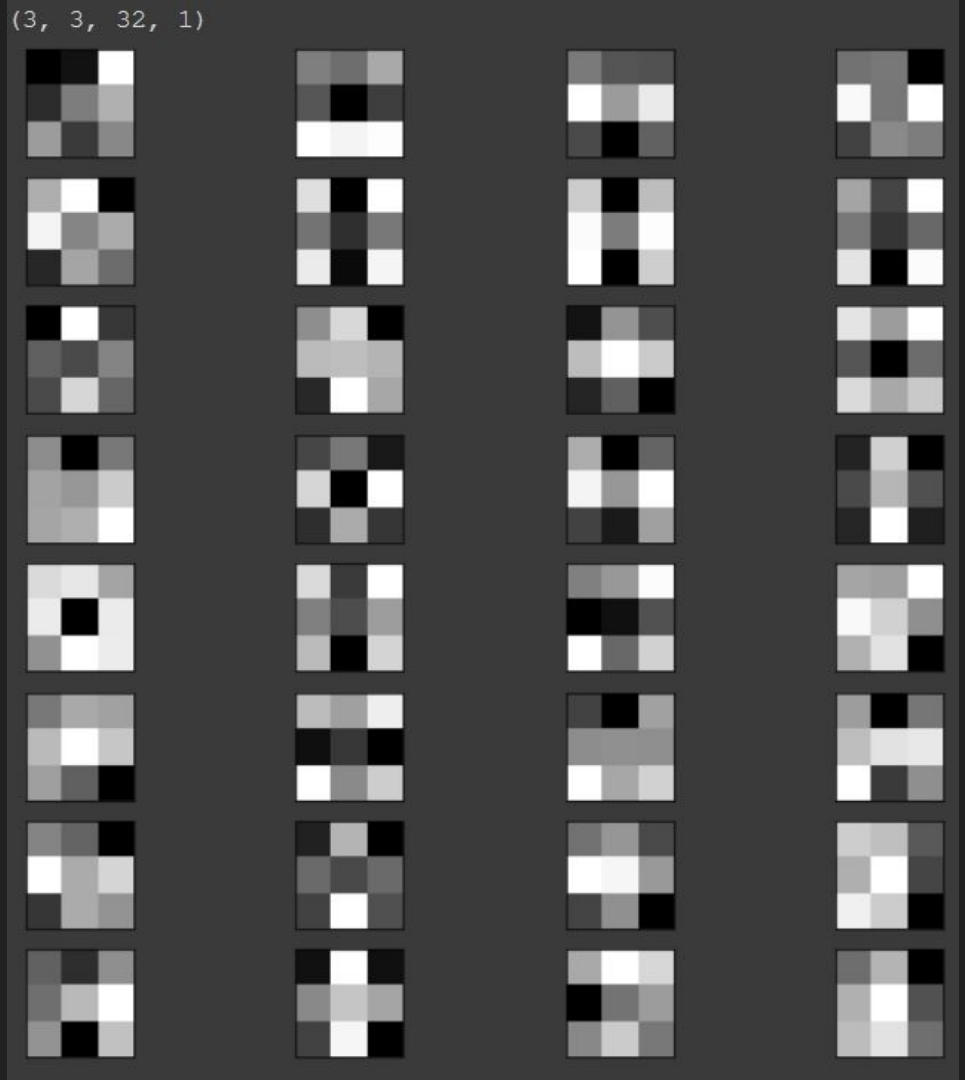
Test with noise



loss: 1.1743 - accuracy: 0.5703

loss: 2.5685 - accuracy: 0.4844

Conv filters



Feature maps