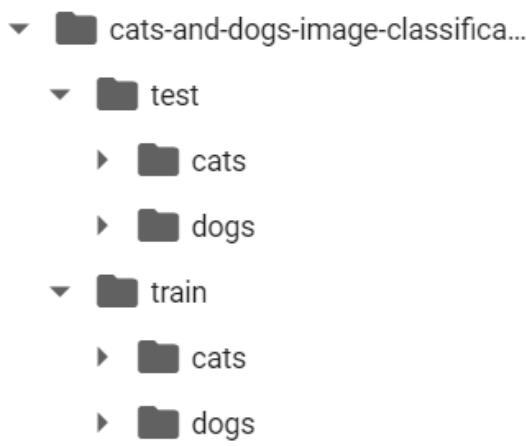
디지털 영상처리 연구실 연구보고서

김우헌

##이미지 분류

->Kaggle data set 이용




```
Found 557 images belonging to 2 classes.
Found 140 images belonging to 2 classes.
```

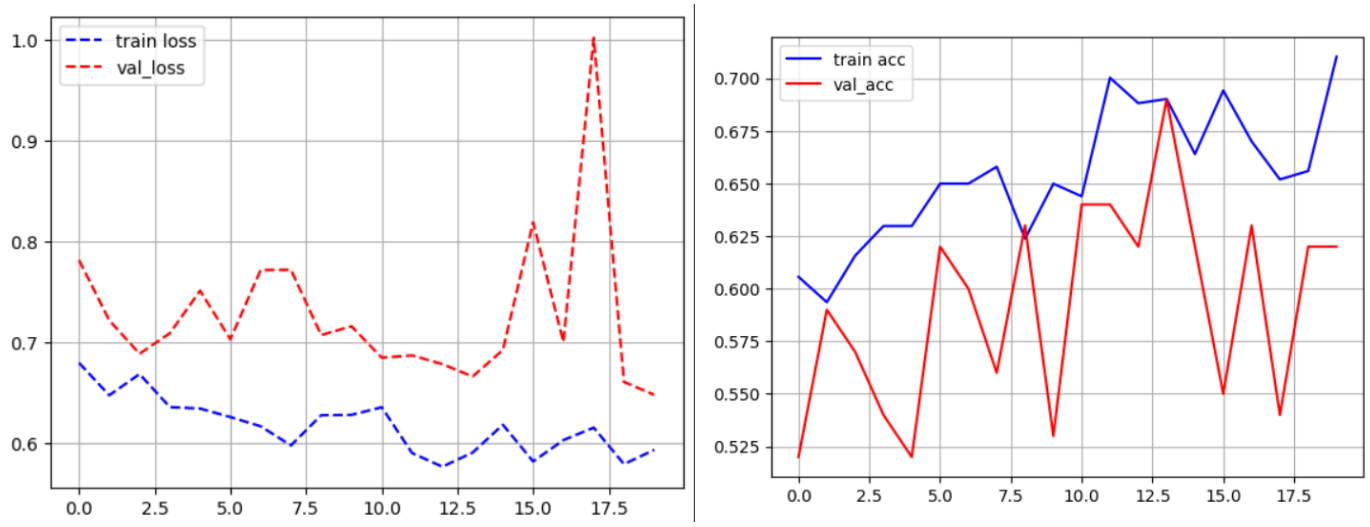
#일반적인 CNN

```
model2=Sequential()
model2.add(Conv2D(32,(3,3),input_shape=(150,150,3)))
model2.add(BatchNormalization())
model2.add(Activation(activation='relu'))
model2.add(MaxPool2D(2,2))

model2.add(Conv2D(32,(3,3)))
model2.add(BatchNormalization())
model2.add(Activation(activation='relu'))
model2.add(MaxPool2D(2,2))

model2.add(Conv2D(32,(3,3)))
model2.add(BatchNormalization())
model2.add(Activation(activation='relu'))
model2.add(MaxPool2D(2,2))

model2.add(Conv2D(32,(3,3)))
model2.add(BatchNormalization())
model2.add(Activation(activation='relu'))
model2.add(MaxPool2D(2,2))

model2.add(Flatten())
model2.add(Dropout(0.5))
model2.add(Dense(512,activation='relu'))
model2.add(Dense(1,activation='sigmoid'))
```

conv2d_7 (Conv2D)	(None, 15, 15, 32)	9248
batch_normalization_3 (BatchNormalization)	(None, 15, 15, 32)	128
activation_3 (Activation)	(None, 15, 15, 32)	0
max_pooling2d_7 (MaxPooling2D)	(None, 7, 7, 32)	0
flatten_1 (Flatten)	(None, 1568)	0
dropout_1 (Dropout)	(None, 1568)	0
dense_2 (Dense)	(None, 512)	803328
dense_3 (Dense)	(None, 1)	513

```
model2.compile(optimizer=Adam(),loss='binary_crossentropy',metrics=['acc'])
```

```
history=model2.fit(train_generator,
                   steps_per_epoch=50,
                    epochs=20,
                    batch_size=256,
                    validation_data=test_generator,
                    validation_steps=10,
                    verbose=2)
```



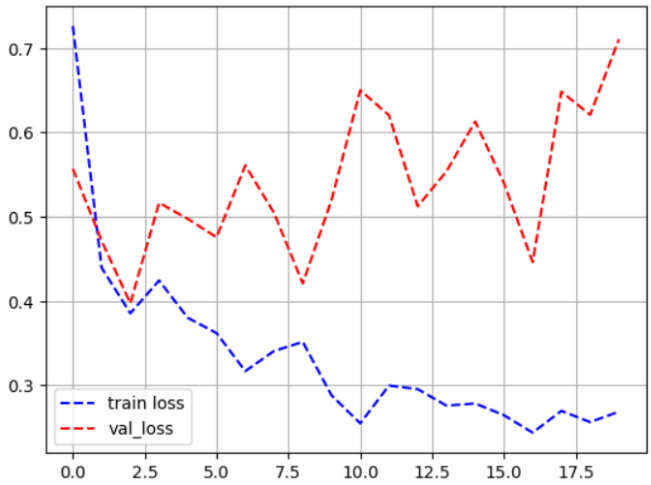
#VGG NET 사용

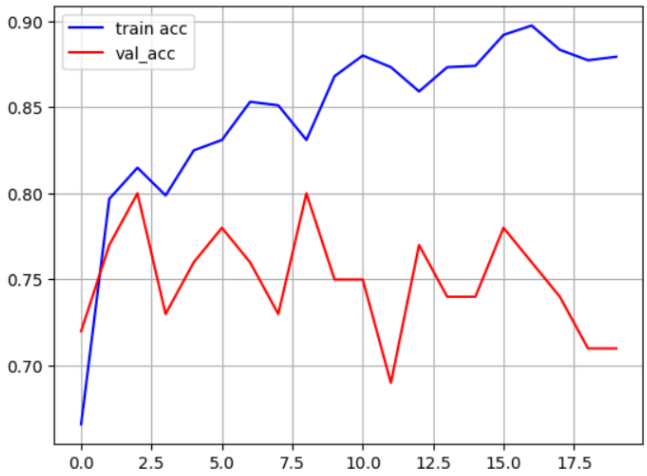
```
conv_base=VGG16(weights='imagenet',input_shape=(150,150,3),include_top=False)
```

```
model=Sequential()
model.add(conv_base)
model.add(Flatten())
model.add(Dense(256,activation='relu'))
model.add(Dense(1,activation='sigmoid'))

model.compile(optimizer=Adam(),loss='binary_crossentrpoy',metrics=['acc'])
```

Layer (type)	Output Shape	Param #
vgg16 (Functional)	(None, 4, 4, 512)	14714688
flatten_7 (Flatten)	(None, 8192)	0
dense_14 (Dense)	(None, 256)	2097408
dense_15 (Dense)	(None, 1)	257

```
history2=model.fit(train_generator,
                   steps_per_epoch=50,
                    epochs=20,
                    batch_size=256,
                    validation_data=test_generator,
                    validation_steps=10,
                    verbose=2)
```

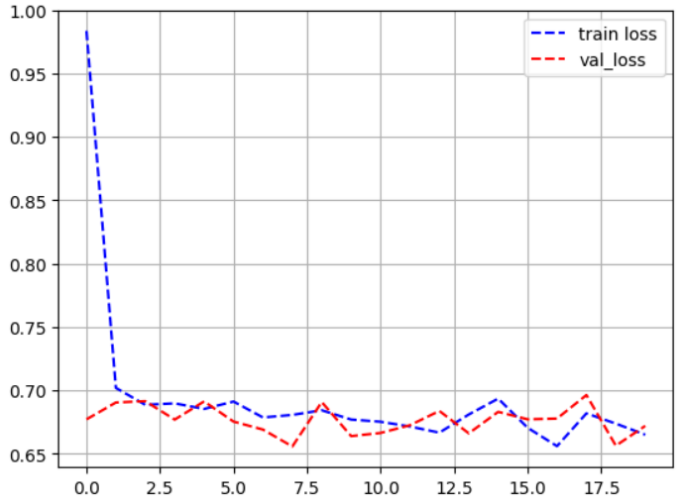


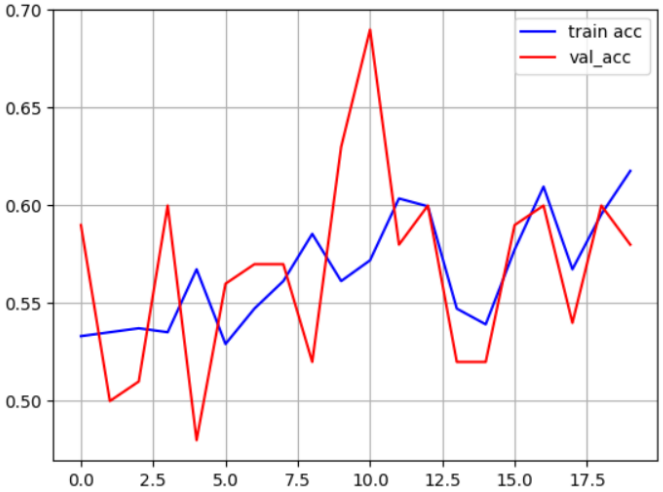


#ResNet사용

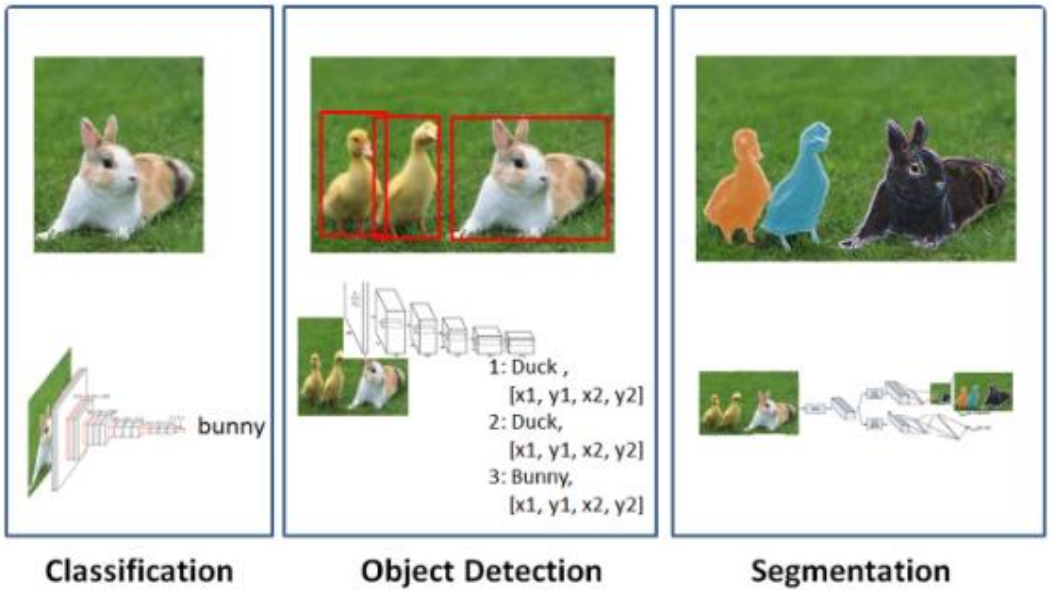
```
resnet101=ResNet101(weights='imagenet',input_shape=(150,150,3),include_top=False)
```

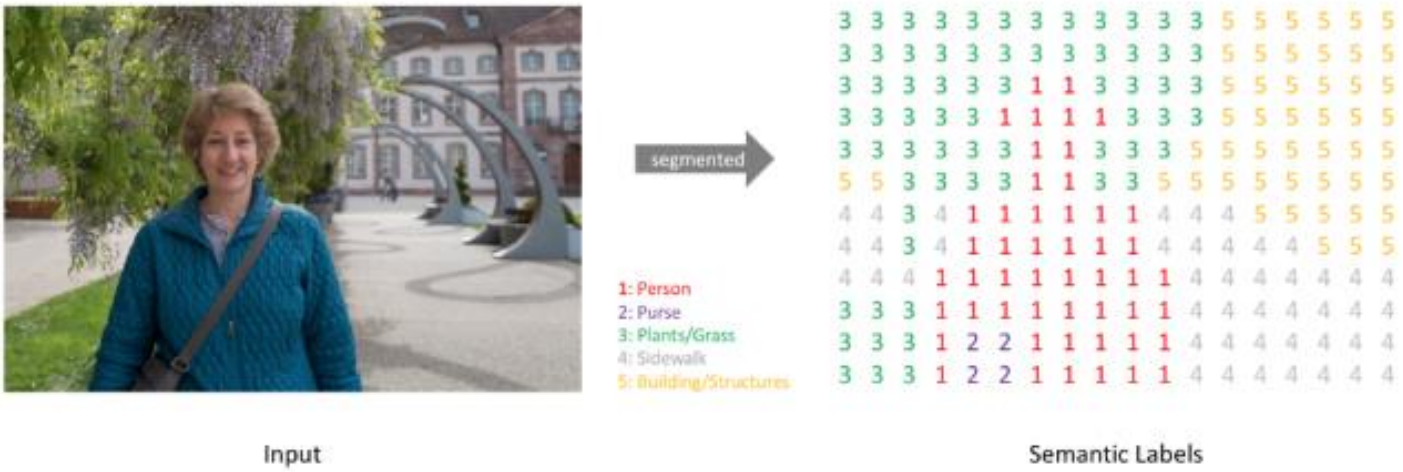
Layer (type)	Output Shape	Param #
resnet101 (Functional)	(None, 5, 5, 2048)	42658176
flatten_5 (Flatten)	(None, 51200)	0
dense_10 (Dense)	(None, 256)	13107456
dense_11 (Dense)	(None, 1)	257

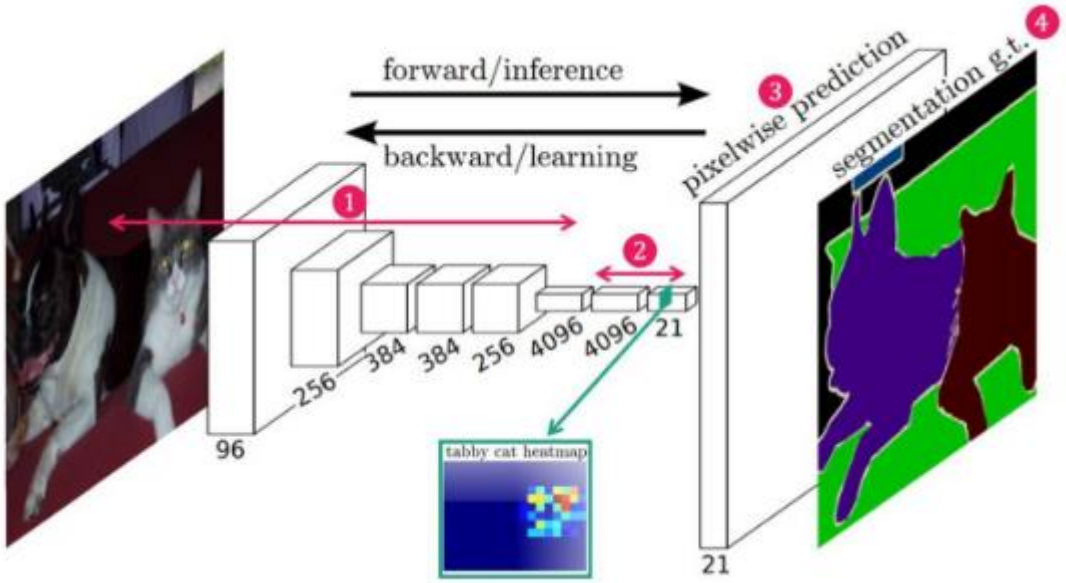




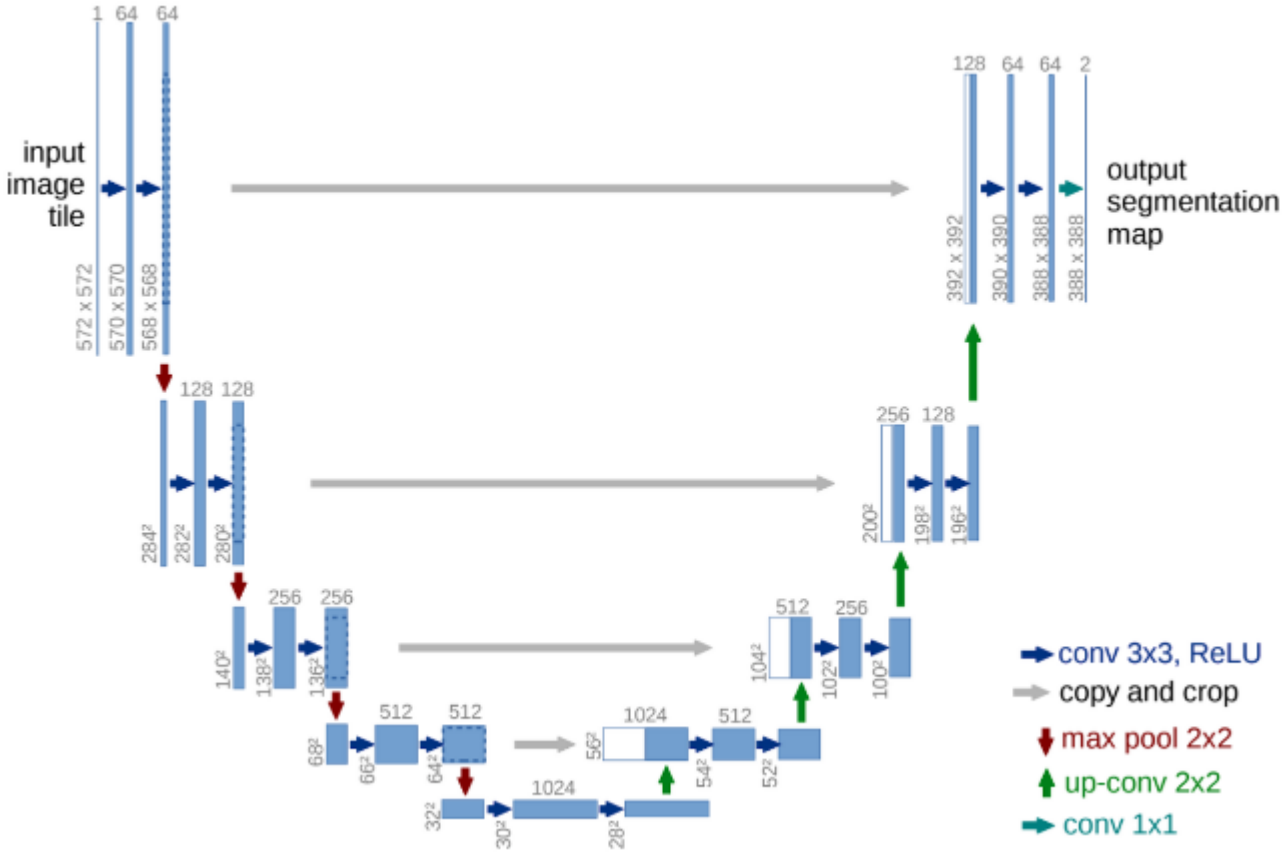
##image segmentation



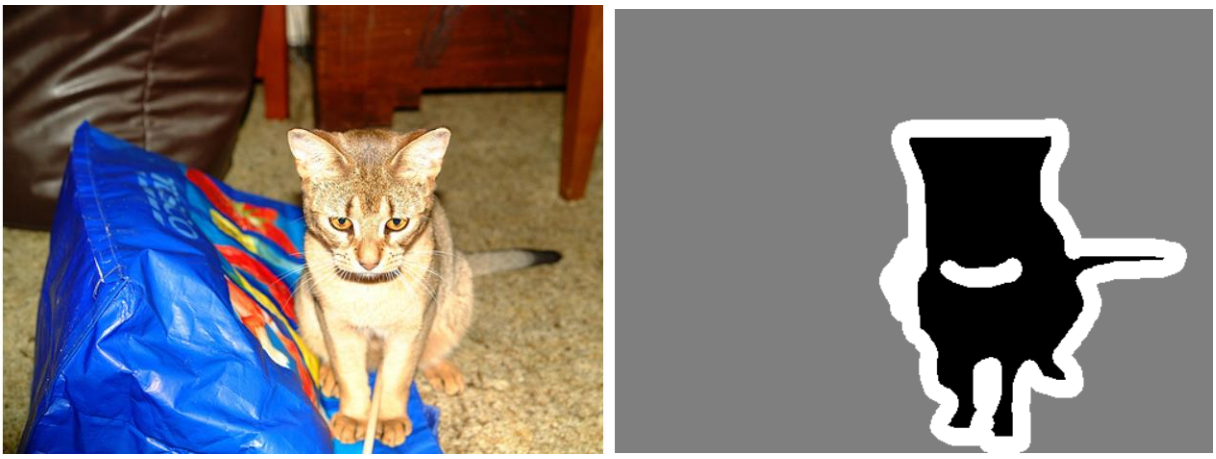




#U-Net 기반 image segmentation

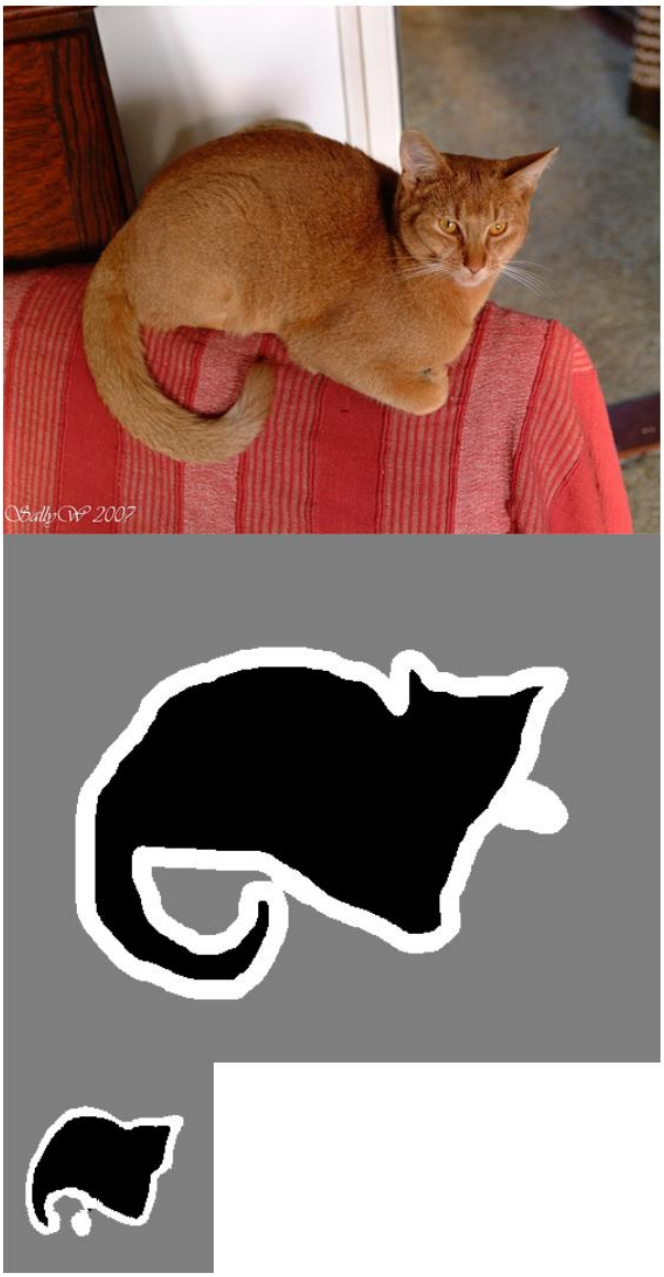


->Oxford-IIIT Pets 데이터셋 사용



->7000개 정도의 data set

```
def get_model(img_size,num_class):
  inputs =keras.Input(shape=img_size+(3,))

  x=layers.Conv2D(32,3,strides=2,padding='same')(inputs)
  x=layers.BatchNormalization()(x)
  x=layers.Activation('relu')(x)

  previous_block_activation=x

  for filters in [62,128,256]:
    x=layers.Activation('relu')(x)
    x=layers.SeparableConv2D(filters,3,padding='same')(x)
    x=layers.BatchNormalization()(x)

    x=layers.Activation('relu')(x)
    x=layers.SeparableConv2D(filters,3,padding='same')(x)
    x=layers.BatchNormalization()(x)

    x=layers.MaxPooling2D(3,strides=2,padding='same')(x)

    residual=layers.Conv2D(filters,1,strides=2,padding='same')(previous_block_activation)
    x=layers.add([x,residual])
    previous_block_activation=x


  for filters in [256,128,64,32]:
    x=layers.Activation('relu')(x)
    x=layers.Conv2DTranspose(filters,3,padding='same')(x)
    x=layers.BatchNormalization()(x)

    x=layers.Activation('relu')(x)
    x=layers.Conv2DTranspose(filters,3,padding='same')(x)
    x=layers.BatchNormalization()(x)

    x=layers.UpSampling2D(2)(x)

    residual=layers.UpSampling2D(2)(previous_block_activation)
    residual=layers.Conv2D(filters,1,padding='same')(residual)
    x=layers.add([x,residual])
    previous_block_activation=x

  outputs=layers.Conv2D(num_classes,3,activation='softmax',padding='same')(x)
  model=keras.Model(inputs,outputs)
  return model

model=get_model(img_size,num_classes)
```


Sally W 2007