





#the input values goes from 0 to 255 in an image. so, dividing it by 255 in a way that converting into your floating each pixel into afloating point number

s = tf.keras.layers.Lambda(lambda x: x / 255)(inputs)

#CONTRACTION- PATH

c1 =tf.keras.layers.Conv2D(16, (3,3), activation='relu', kernel\_initializer='he\_normal', padding='same')(s)

c1 =tf.keras.layers.Dropout(0.1)(c1)

c1 =tf.keras.layers.Conv2D(16, (3,3), activation='relu', kernel\_initializer='he\_normal', padding='same')(c1)

p1 = tf.keras.layers.MaxPooling2D((2,2))(c1)

c2 =tf.keras.layers.Conv2D(32, (3,3), activation='relu', kernel\_initializer='he\_normal', padding='same')(p1)

c2 =tf.keras.layers.Dropout(0.1)(c2)

c2 =tf.keras.layers.Conv2D(32, (3,3), activation='relu', kernel\_initializer='he\_normal', padding='same')(c2)

p2 = tf.keras.layers.MaxPooling2D((2,2))(c2)

c3 =tf.keras.layers.Conv2D(64, (3,3), activation='relu', kernel\_initializer='he\_normal', padding='same')(p2)

c3 =tf.keras.layers.Dropout(0.2)(c3)

c3 =tf.keras.layers.Conv2D(64, (3,3), activation='relu', kernel\_initializer='he\_normal', padding='same')(c3)

p3 = tf.keras.layers.MaxPooling2D((2,2))(c3)

c4 =tf.keras.layers.Conv2D(128, (3,3), activation='relu', kernel\_initializer='he\_normal', padding='same')(p3)

c4 =tf.keras.layers.Dropout(0.2)(c4)

c4 =tf.keras.layers.Conv2D(128, (3,3), activation='relu', kernel\_initializer='he\_normal', padding='same')(c4)

p4 = tf.keras.layers.MaxPooling2D((2,2))(c4)

c5 =tf.keras.layers.Conv2D(256, (3,3), activation='relu', kernel\_initializer='he\_normal', padding='same')(p4)

c5 =tf.keras.layers.Dropout(0.3)(c5)

c5 =tf.keras.layers.Conv2D(256, (3,3), activation='relu', kernel\_initializer='he\_normal', padding='same')(c5)

#EXPANSION- PATH

u6 = tf.keras.layers.Conv2DTranspose(128, (2, 2), strides=(2, 2), padding='same')(c5)

u6 = tf.keras.layers.concatenate([u6, c4])

c6 = tf.keras.layers.Conv2D(128, (3, 3), activation='relu', kernel\_initializer='he\_normal', padding='same')(u6)

c6 = tf.keras.layers.Dropout(0.2)(c6)

c6 = tf.keras.layers.Conv2D(128, (3, 3), activation='relu', kernel\_initializer='he\_normal', padding='same')(c6)

u7 = tf.keras.layers.Conv2DTranspose(64, (2, 2), strides=(2, 2), padding='same')(c6)

u7 = tf.keras.layers.concatenate([u7, c3])

c7 = tf.keras.layers.Conv2D(64, (3, 3), activation='relu', kernel\_initializer='he\_normal', padding='same')(u7)

c7 = tf.keras.layers.Dropout(0.2)(c7)

c7 = tf.keras.layers.Conv2D(64, (3, 3), activation='relu', kernel\_initializer='he\_normal', padding='same')(c7)

u8 = tf.keras.layers.Conv2DTranspose(32, (2, 2), strides=(2, 2), padding='same')(c7)

u8 = tf.keras.layers.concatenate([u8, c2])

c8 = tf.keras.layers.Conv2D(32, (3, 3), activation='relu', kernel\_initializer='he\_normal', padding='same')(u8)

c8 = tf.keras.layers.Dropout(0.1)(c8)

c8 = tf.keras.layers.Conv2D(32, (3, 3), activation='relu', kernel\_initializer='he\_normal', padding='same')(c8)

u9 = tf.keras.layers.Conv2DTranspose(16, (2, 2), strides=(2, 2), padding='same')(c8)

u9 = tf.keras.layers.concatenate([u9, c1], axis=3)

c9 = tf.keras.layers.Conv2D(16, (3, 3), activation='relu', kernel\_initializer='he\_normal', padding='same')(u9)

c9 = tf.keras.layers.Dropout(0.1)(c9)

c9 = tf.keras.layers.Conv2D(16, (3, 3), activation='relu', kernel\_initializer='he\_normal', padding='same')(c9)

outputs = tf.keras.layers.Conv2D(1, (1, 1), activation='sigmoid')(c9)

model = tf.keras.Model(inputs=[inputs], outputs=[outputs])

model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

model.summary()