DEEP–LEARNING(Concepts)

***U-Net Code – Summary - Malignant Image Segmentation***

Theory – Understanding:

* + - It’s a type of neural network that helps in identifying objects within an image. (For example: Tumor in brain).
    - Widely used in medical Industry.

How does it work?

* + - **Encoder** – It downscales the image, thus learning small details. Simple 🡪 Complex image.
    - **Decoder** – It upscales the image. Formed Complex image 🡪 Original Size
    - **Unique Feature** – Use of Skip Connections 🡪 Ensure to keep informations from getting lost , helping the model to understand the borders around the object.

Pseudo – Code:

**(Encoder)**

(256 \* 256) 🡪applied 3\*3 convolution(2 layers)🡪output-S1 🡪applied Max Pooling 2\*2 🡪newly formed input layer(128 \* 128).

(256 \* 256) 🡪applied 3\*3 convolution(2 layers)🡪output-S2 🡪applied Max Pooling 2\*2 🡪newly formed input layer(64 \* 64).

(64 \*64) 🡪applied 3\*3 convolution(2 layers)🡪output-S3 🡪applied Max Pooling 2\*2 🡪newly formed input layer(32 \* 32).

(32 \* 32) 🡪applied 3\*3 convolution(2 layers)🡪output-S4 🡪applied Max Pooling 2\*2 🡪newly formed input layer(16 \* 16).

**(Decoder)**

(16 \* 16) 🡪up convolution(2 \* 2)🡪output-d1 🡪 applied 3\*3 convolution(2 layers)🡪newly formed output layer(32 \*32).

(32 \*32) 🡪up convolution(2 \* 2)🡪output-d2 🡪 applied 3\*3 convolution(2 layers)🡪newly formed output layer(64 \* 64).

(64 \* 64) 🡪up convolution(2 \* 2)🡪output-d3 🡪 applied 3\*3 convolution(2 layers)🡪newly formed output layer(128 \*128).

(128 \*128) 🡪up convolution(2 \* 2)🡪output-d4 🡪 applied 3\*3 convolution(2 layers)🡪newly formed output layer(256 \*256).

**(Skip Connections)**

After Upscaling each output image, It does concatenation with the corresponding feature map from the encoding path . (d1,s4)

**U-Net Model Code:**

def unet\_model():

inputs = Input((256,256,1))

s1 = Conv2D(64,3,activation = 'relu' , padding = 'same')(inputs)

s1 = Conv2D(64,3,activation = 'relu' , padding = 'same')(s1)

p1 = MaxPooling2D(pool\_size = (2,2))(s1)

s2 = Conv2D(128,3,activation = 'relu' , padding = 'same')(p1)

s2 = Conv2D(128,3,activation = 'relu' , padding = 'same')(s2)

p2 = MaxPooling2D(pool\_size = (2,2))(s2)

s3 = Conv2D(256,3,activation = 'relu' , padding = 'same')(p2)

s3 = Conv2D(256,3,activation = 'relu' , padding = 'same')(s3)

p3 = MaxPooling2D(pool\_size = (2,2))(s3)

s4 = Conv2D(512,3,activation = 'relu' , padding = 'same')(p3)

s4 = Conv2D(512,3,activation = 'relu' , padding = 'same')(s4)

p4 = MaxPooling2D(pool\_size = (2,2))(s4)

b1 = Conv2D(1024,3,activation = 'relu' , padding = 'same')(p4)

b1 = Conv2D(1024,3,activation = 'relu' , padding = 'same')(b1)

d1 = Conv2D(512,2,activation = 'relu' , padding = 'same')(UpSampling2D(size = (2,2))(b1))

d1 = concatenate([s4,d1], axis = 3)

d1 = Conv2D(512,3,activation = 'relu' , padding = 'same')(d1)

d1 = Conv2D(512,3,activation = 'relu' , padding = 'same')(d1)

d2 = Conv2D(256,2,activation = 'relu' , padding = 'same')(UpSampling2D(size = (2,2))(d1))

d2 = concatenate([s3,d2], axis = 3)

d2 = Conv2D(256,3,activation = 'relu' , padding = 'same')(d2)

d2 = Conv2D(256,3,activation = 'relu' , padding = 'same')(d2)

d3 = Conv2D(128,2,activation = 'relu' , padding = 'same')(UpSampling2D(size = (2,2))(d2))

d3 = concatenate([s2,d3], axis = 3)

d3 = Conv2D(128,3,activation = 'relu' , padding = 'same')(d3)

d3 = Conv2D(128,3,activation = 'relu' , padding = 'same')(d3)

d4 = Conv2D(64,2,activation = 'relu' , padding = 'same')(UpSampling2D(size = (2,2))(d3))

d4 = concatenate([s1,d4], axis = 3)

d4 = Conv2D(64,3,activation = 'relu' , padding = 'same')(d4)

d4 = Conv2D(64,3,activation = 'relu' , padding = 'same')(d4)

d4 = Conv2D(2,3,activation = 'relu' , padding = 'same')(d4)

out = Conv2D(1,1,activation = 'sigmoid')(d4)

model = Model(inputs,out)

return model