Group Name: Innovators Group 9

ML+CV Combined Project: Cell Segmentation

Group Members:

Aadya Chinubhai AU2040232

Abhishu Oza AU2040027

Razin Karimi AU2040230

Nihar Jani AU2040205

Task Performed:

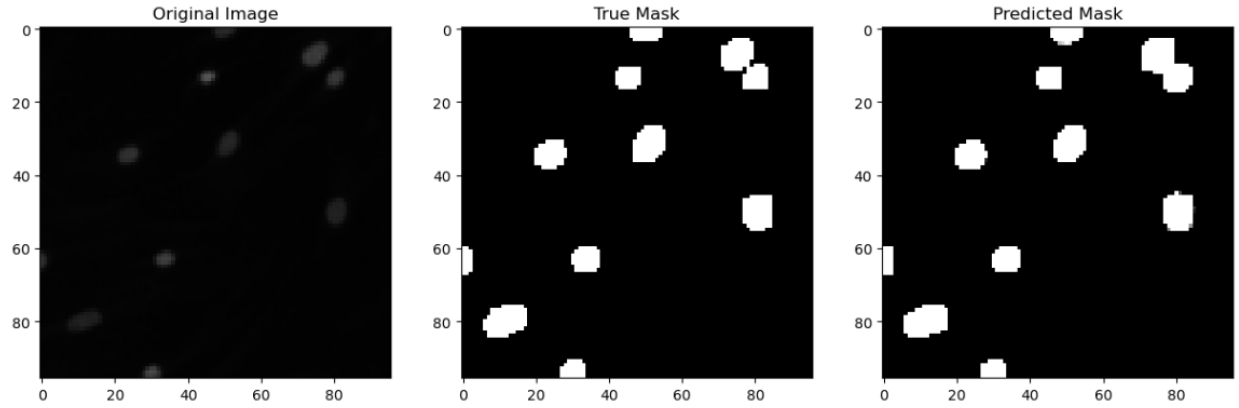
We implemented five layered UNet architecture as a part of hyper parameter training and implemented all the activation functions on the architecture

```
        c1 = Conv2D(16, (3, 3), activation= self.activation, kernel_initializer=kernel_initializer, padding='same')(c1)
        p1 = MaxPooling2D((2, 2))(c1)

        c2 = Conv2D(32, (3, 3), activation= self.activation, kernel_initializer=kernel_initializer, padding='same')(p1)
#         c2 = Dropout(0.1)(c2)
        c2 = Conv2D(32, (3, 3), activation= self.activation, kernel_initializer=kernel_initializer, padding='same')(c2)
        p2 = MaxPooling2D((2, 2))(c2)

        c3 = Conv2D(64, (3, 3), activation= self.activation, kernel_initializer=kernel_initializer, padding='same')(p2)
#         c3 = Dropout(0.2)(c3)
        c3 = Conv2D(64, (3, 3), activation= self.activation, kernel_initializer=kernel_initializer, padding='same')(c3)
        p3 = MaxPooling2D((2, 2))(c3)

        c4 = Conv2D(128, (3, 3), activation= self.activation, kernel_initializer=kernel_initializer, padding='same')(p3)
#         c4 = Dropout(0.2)(c4)
        c4 = Conv2D(128, (3, 3), activation= self.activation, kernel_initializer=kernel_initializer, padding='same')(c4)
        p4 = MaxPooling2D(pool_size=(2, 2))(c4)

        c_x = Conv2D(128, (3, 3), activation= self.activation, kernel_initializer=kernel_initializer, padding='same')(p3)
#         c4 = Dropout(0.2)(c4)
        c_x = Conv2D(128, (3, 3), activation= self.activation, kernel_initializer=kernel_initializer, padding='same')(c4)
        p_x = MaxPooling2D(pool_size=(2, 2))(c4)

        c5 = Conv2D(256, (3, 3), activation= self.activation, kernel_initializer=kernel_initializer, padding='same')(p_x)
#         c5 = Dropout(0.3)(c5)
        c5 = Conv2D(256, (3, 3), activation= self.activation, kernel_initializer=kernel_initializer, padding='same')(c5)

        #Expansive path
        u_x = Conv2DTranspose(32, (2, 2), strides=(2, 2), padding='same')(c5)
        u_x = concatenate([u_x, c_x])
        c_x = Conv2D(32, (3, 3), activation= self.activation, kernel_initializer=kernel_initializer, padding='same')(u_x)
#         c8 = Dropout(0.1)(c8)
        c_x = Conv2D(32, (3, 3), activation= self.activation, kernel_initializer=kernel_initializer, padding='same')(c_x)

        u6 = Conv2DTranspose(128, (2, 2), strides=(2, 2), padding='same')(c5)
        u6 = concatenate([u6, c4])
        c6 = Conv2D(128, (3, 3), activation= self.activation, kernel_initializer=kernel_initializer, padding='same')(u6)
#         c6 = Dropout(0.2)(c6)
        c6 = Conv2D(128, (3, 3), activation= self.activation, kernel_initializer=kernel_initializer, padding='same')(c6)

        u7 = Conv2DTranspose(64, (2, 2), strides=(2, 2), padding='same')(c6)
        u7 = concatenate([u7, c3])
        c7 = Conv2D(64, (3, 3), activation= self.activation, kernel_initializer=kernel_initializer, padding='same')(u7)
#         c7 = Dropout(0.2)(c7)
        c7 = Conv2D(64, (3, 3), activation= self.activation, kernel_initializer=kernel_initializer, padding='same')(c7)

        u8 = Conv2DTranspose(32, (2, 2), strides=(2, 2), padding='same')(c7)
        u8 = concatenate([u8, c2])
        c8 = Conv2D(32, (3, 3), activation= self.activation, kernel_initializer=kernel_initializer, padding='same')(u8)
#         c8 = Dropout(0.1)(c8)
        c8 = Conv2D(32, (3, 3), activation= self.activation, kernel_initializer=kernel_initializer, padding='same')(c8)


        u9 = Conv2DTranspose(16, (2, 2), strides=(2, 2), padding='same')(c8)
        u9 = concatenate([u9, c1])
        c9 = Conv2D(16, (3, 3), activation= self.activation, kernel_initializer=kernel_initializer, padding='same')(u9)
#         c9 = Dropout(0.1)(c9)
        c9 = Conv2D(16, (3, 3), activation= self.activation, kernel_initializer=kernel_initializer, padding='same')(c9)
```

Outcomes:

We plotted predictions for every activation functions

Eg:Swish:



Eg:TrainableLeakyRelu