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

Tasks performed:

- In this report, we are going to implement, ***2 Activation function***, which are:
  1) ***ISRU ACTIVATION FUNCTION***
  2) ***TRAINABLE LEAKY RELU***

ISRU ACTIVATION FUNCTION

Introduction

Inverse Square Root Unit (ISRU) Activation function has the following characteristics:

1) Slow lag period as x builds
2) A rapid Growth phase, including an inflexion period at the middle where concavity changes.

Equation of ISRU activation Function:

y = (x /sqrt (1 + a * (x^2)))

- a is a constant

a>=0

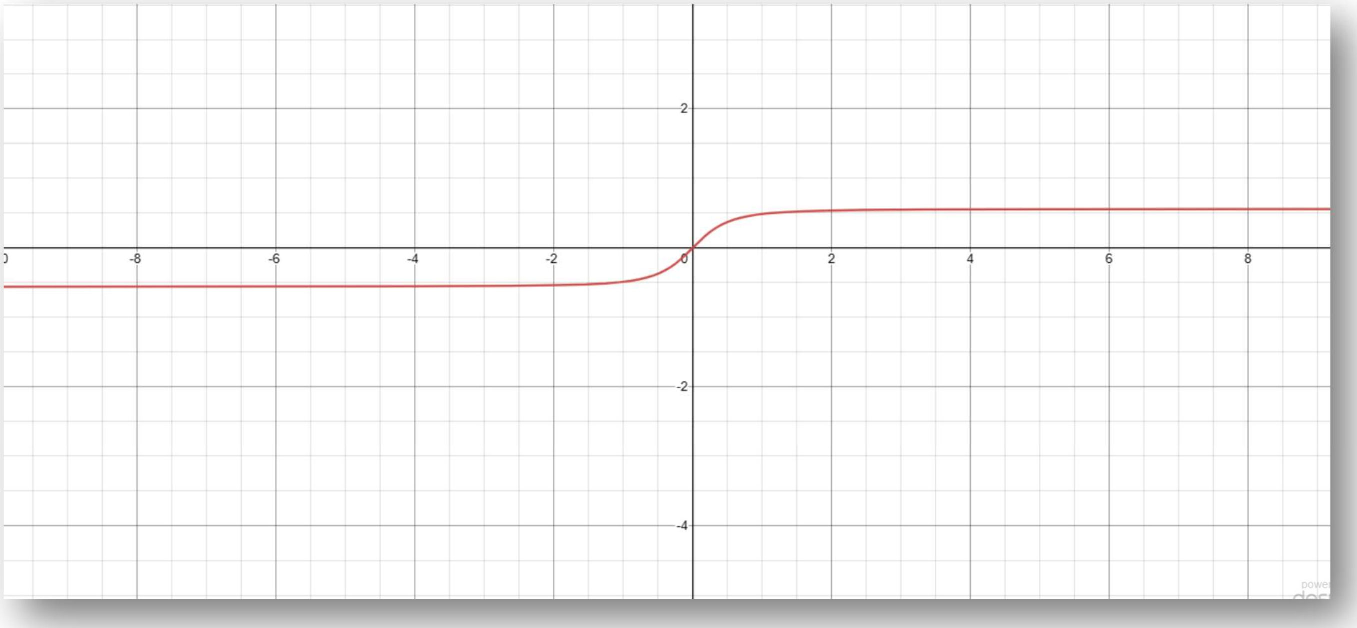
- Output: -1/ sqrt(a) to 1/sqrt(a), including both upper and lower bound

Benefits of using ISRU Activation Function

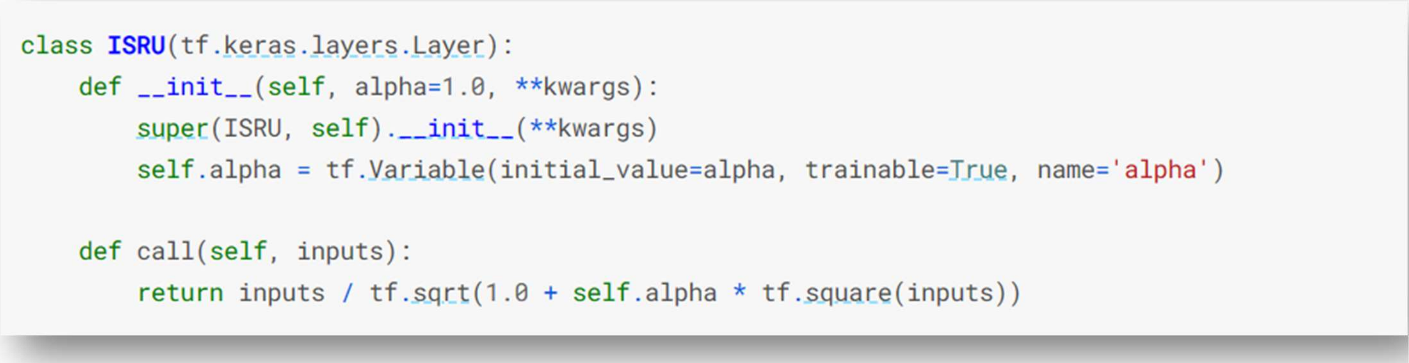
- When We compare the ***ISRU activation function with sigmoidal activation*** function. In Sigmoidal activation function, the output is fixed between 0 and 1, including both upper and lower bound. ***But In ISRU we can vary it by changing the parameter a, which can be seen from the output of ISRU.***

- When we compare ***ISRU activation function with Relu activation function***. In Relu, if the value if greater than certain linearity is achieved, which may loss the data, ***but in ISRU activation function, no-linearity is achieved at any point of time.***

Graphical Plot of ISRU Activation

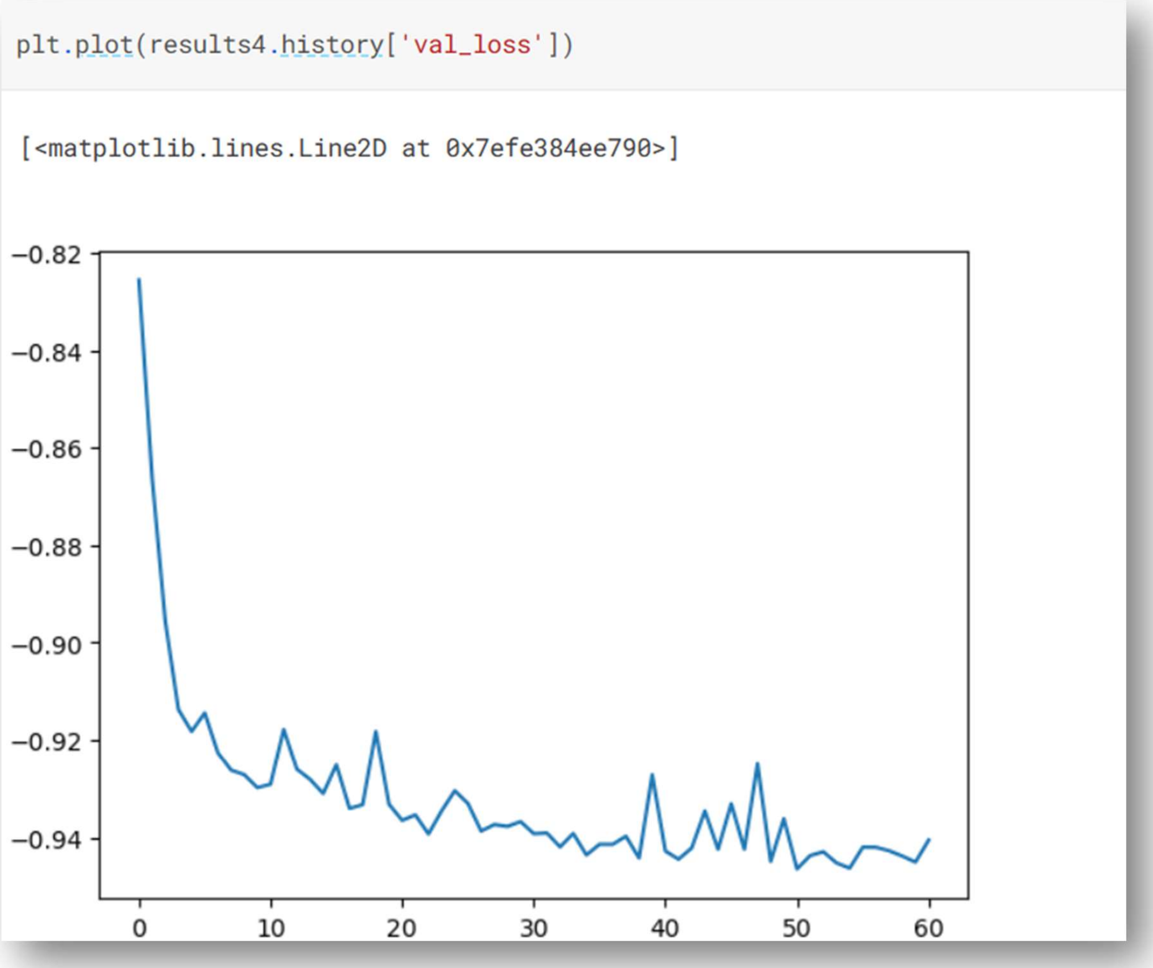


Code



```
class ISRU(tf.keras.layers.Layer):
    def __init__(self, alpha=1.0, **kwargs):
        super(ISRU, self).__init__(**kwargs)
        self.alpha = tf.Variable(initial_value=alpha, trainable=True, name='alpha')

    def call(self, inputs):
        return inputs / tf.sqrt(1.0 + self.alpha * tf.square(inputs))
```

Output:



```
plt.plot(results4.history['val_loss'])
```

```
[<matplotlib.lines.Line2D at 0x7efe384ee790>]
```

Trainable Leaky Relu Activation Function

Introduction

Trainable Leaky Relu Activation function has following characteristic:

1) It prevents dying problem
2) There has small variation of positive or negative constant has compared to Relu Activation function.

Equation of Trainable Leaky Relu Activation Function

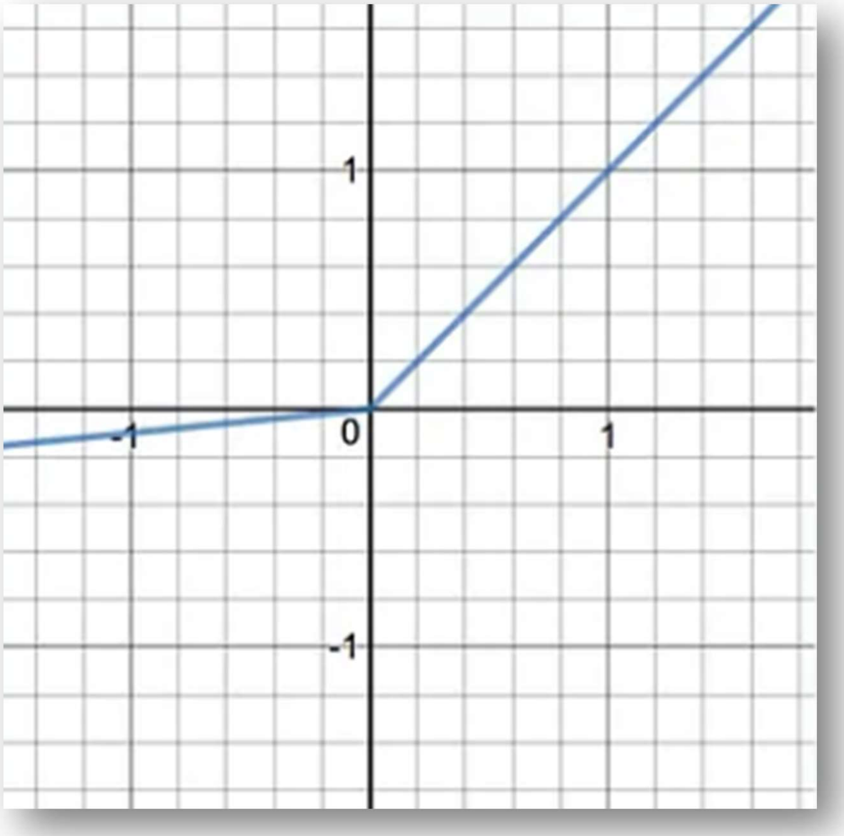
y = {z, z<0

az, z>=0}

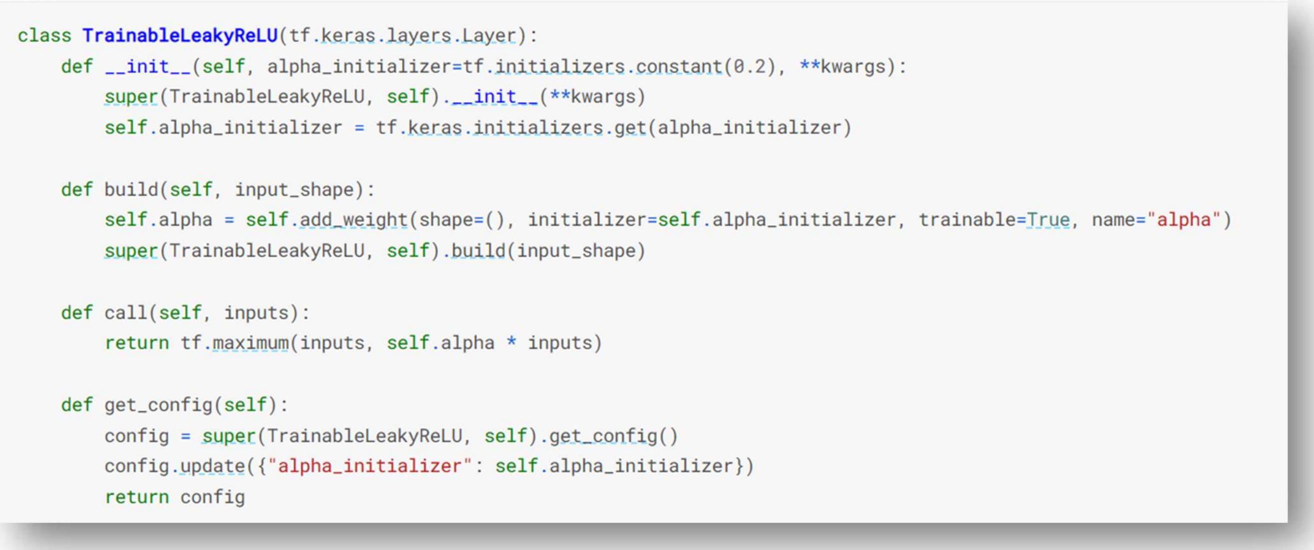
Benefits of Trainable Leaky Relu Activation Function

Instead of multiplying x with a constant term we can multiply it with a hyper-parameter which seems to work better the leaky ReLU. This extension to leaky ReLU is known as **Parametric ReLU**. While we compare Leaky-ReLU with ReLU, then It shows clear concept of difference between them.

Graphical Plot of Trainable Leaky Relu Activation Function

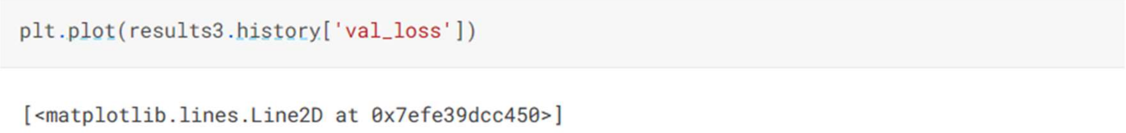


Code:



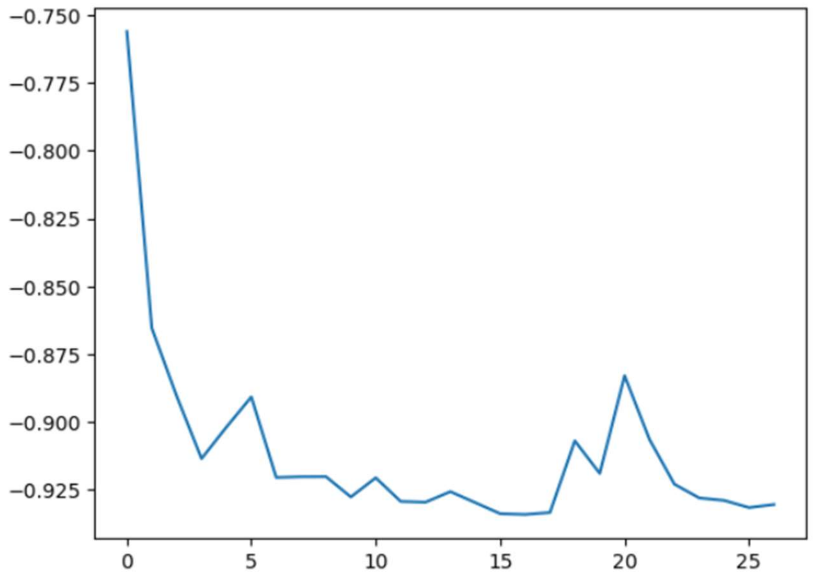
```
class TrainableLeakyReLU(tf.keras.layers.Layer):
    def __init__(self, alpha_initializer=tf.initializers.constant(0.2), **kwargs):
        super(TrainableLeakyReLU, self).__init__(**kwargs)
        self.alpha_initializer = tf.keras.initializers.get(alpha_initializer)

    def build(self, input_shape):
        self.alpha = self.add_weight(shape=(), initializer=self.alpha_initializer, trainable=True, name="alpha")
        super(TrainableLeakyReLU, self).build(input_shape)

    def call(self, inputs):
        return tf.maximum(inputs, self.alpha * inputs)

    def get_config(self):
        config = super(TrainableLeakyReLU, self).get_config()
        config.update({"alpha_initializer": self.alpha_initializer})
        return config
```

Output:



```
plt.plot(results3.history['val_loss'])
```

```
[<matplotlib.lines.Line2D at 0x7efe39dcc450>]
```



Task To Be Performed

We will train ***Mish*** and ***Gated Mish*** in the upcoming Week