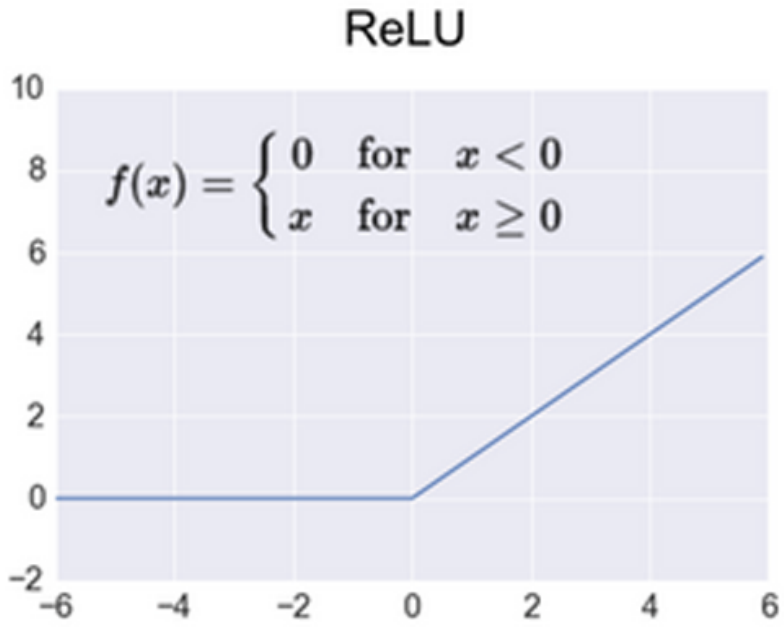
3. Relu

- By Using of Sigmoid And Tanh function there is vanishing gradient problem occure so the convergence rate slow down.
- To overcome slighlty we use Relu Function. it not totally overcome this problem but here the convergence rate is faster than sigmoid and tanh.
- Value Range :- [0, inf)
- Its Nature non-linear, which means we can easily backpropagate the errors and have multiple layers of neurons being activated by the ReLU function.



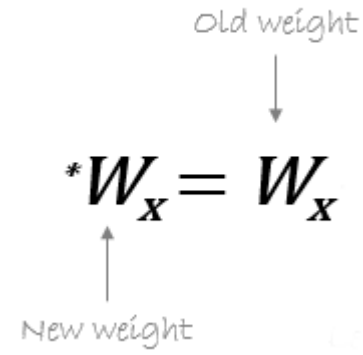
Point:

1. Vanishing Gradient Problem occure due to multiple number of derivative.
2. As sigmoid derivative 0 to 0.25 so by multiple by this sigmoid derivative the result might vanishing gradient as number of hidden layer increase.
3. But in Relu here its derivative 0 to 1 so here no problem of any vanishing gradient problem as its derivative cant be 0.2,0.3 like.
4. But as its derivative can be 0 so here a problem aries that called Dead_Activation

Uses :- ReLu is less computationally expensive than tanh and sigmoid because it involves simpler mathematical operations. At a time only a few neurons are activated making the network sparse making it efficient and easy for computation.

What is Dead_Activation ?

- When Derivative equal to 0 in Relu Then New Weight = Old Weight :



which is not good for any model.Here Weight Cant be update. it occure when value of z is negative.This state called Dead Activation state. to overcome this we use Leaky Relu.

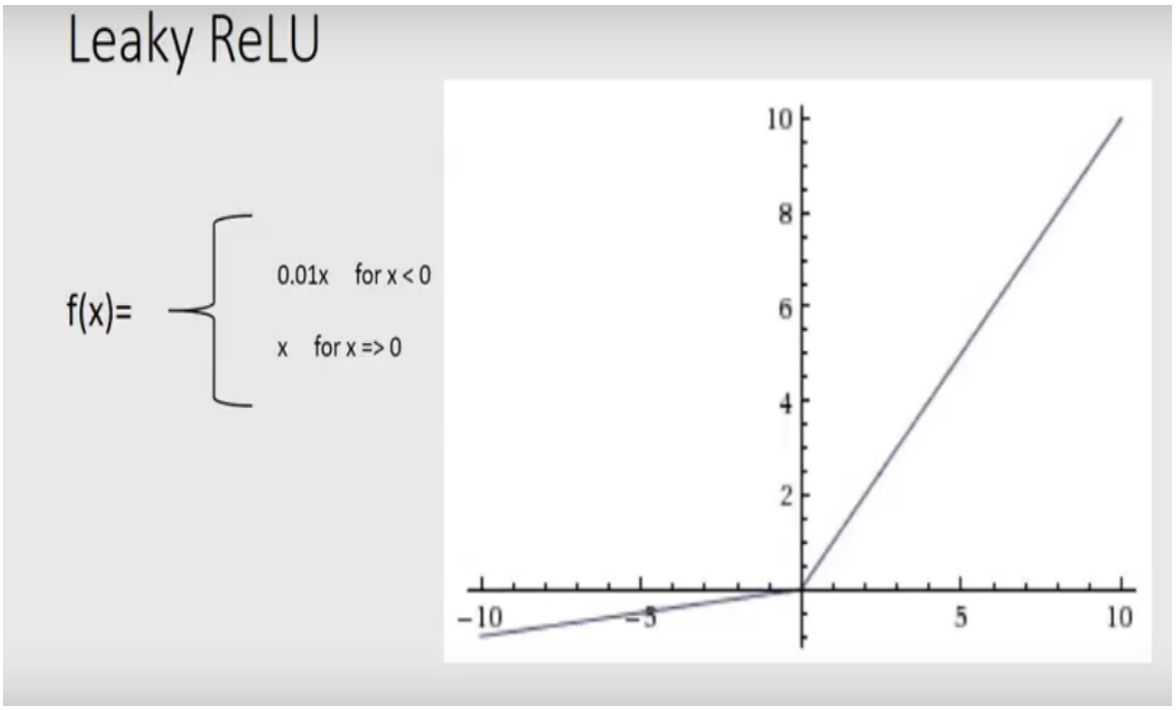
```
Epoch 1/25
32428/32428 [==============================] - 41s 1ms/step - loss: nan - accuracy: 0.5900 - val_loss: nan - val_accuracy: 0.59
44
Epoch 2/25
32428/32428 [==============================] - 39s 1ms/step - loss: nan - accuracy: 0.5907 - val_loss: nan - val_accuracy: 0.59
44
Epoch 3/25
32428/32428 [==============================] - 36s 1ms/step - loss: nan - accuracy: 0.5907 - val_loss: nan - val_accuracy: 0.59
44
Epoch 4/25
32428/32428 [==============================] - 34s 1ms/step - loss: nan - accuracy: 0.5907 - val_loss: nan - val_accuracy: 0.59
44
Epoch 5/25
32428/32428 [==============================] - 36s 1ms/step - loss: nan - accuracy: 0.5907 - val_loss: nan - val_accuracy: 0.59
44
Epoch 6/25
32428/32428 [==============================] - 37s 1ms/step - loss: nan - accuracy: 0.5907 - val_loss: nan - val_accuracy: 0.59
44
Epoch 7/25
32428/32428 [==============================] - 36s 1ms/step - loss: nan - accuracy: 0.5907 - val_loss: nan - val_accuracy: 0.59
44
Epoch 8/25
32428/32428 [==============================] - 38s 1ms/step - loss: nan - accuracy: 0.5907 - val_loss: nan - val_accuracy: 0.59
```

- See here no updation happen the value remain same and ‘nan’ for validation loss is an unexpected very large or very small number. This is dead activation state to overcome this we use leaky relu

4. Leaky Relu

- Leaky ReLU is an improved version of the ReLU function.
- ReLU function, the gradient is 0 for x < 0(-ve), which made the neurons die for activations in that region.
- Leaky ReLU is defined to address this problem. Instead of defining the Relu function as 0 for x less than 0, we define it as a small linear component of x.
- Leaky ReLUs are one attempt to fix the Dying ReLU problem. Instead of the function being zero when x < 0, a leaky ReLU will instead have a small negative slope (of 0.01, or so). That is, the function computes:

$$f(x) = 1(x < 0)(\alpha x) + 1(x >= 0)(x)$$



Sigmoid,Tanh,Relu,Leaky Relu