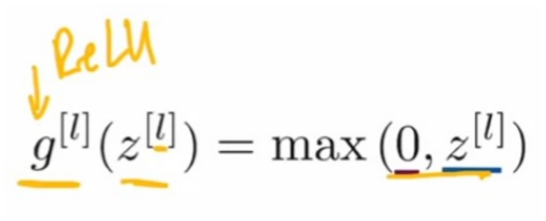
2, Common Activation Functions

Notes

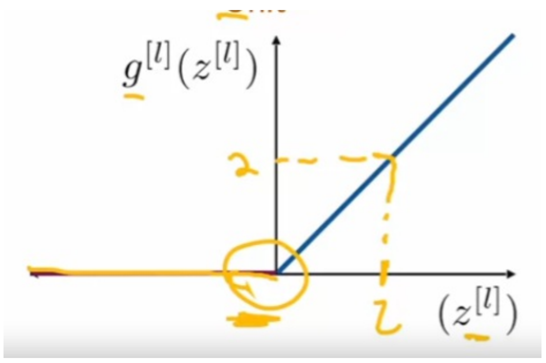
four commonly used activation functions that you'll be using in your GAN.

- ▼ ReLU
  - ▼ formula


ReLU

$g^{[l]}(z^{[l]}) = \max(0, z^{[l]})$

  - ▼ case study



- if input is 2, then output will be 2 also
  - ▼ if input is negative, the output will be zero (which make it is non linear)

    > no negative value is allowed

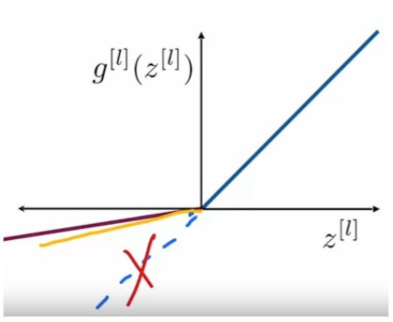
  - ▼ if input is zero, the output will be zero(might cause it have non-derivative result, that might cause dying ReLU)

    > Linear would mean a single straight line. You might notice that ReLU is strictly speaking not differentiable at z equals zero. But by convention and implementation, the derivative of ReLU at z equals zero is often set to zero here.

- ▼ Dying ReLU problem

  > The flat part of the ReLU activation function when **z is negative always has a derivative equal to zero**. This can be problematic because the learning process depends on the derivative to provide important information on how to update the weights. With a zero derivative some nodes get stuck on the same value and their weights stops learning.

- ▼ Leaky ReLU (variant of ReLU)



▼ formula

$$g^{[l]}(z^{[l]}) = max(\underline{az^{[l]}}, \underline{z^{[l]}})$$

▼ case study (explanation)

> What the Leaky ReLU does is that it maintains the same form of ReLU, and it maintains the same form as ReLU for the case when z is positive, which means that it keeps the same positive value again as whatever the input is, but it adds a little leak or a slope in the line when z is less than zero, when z is negative down here, and it's still nonlinear with a bend in the slope at z equals zero, but now it has this non-zero derivative when z is negative. **This slope is intended to be less than one** so it doesn't have to form a line with this positive side.(to prevent the red dotted line in pic might happen)

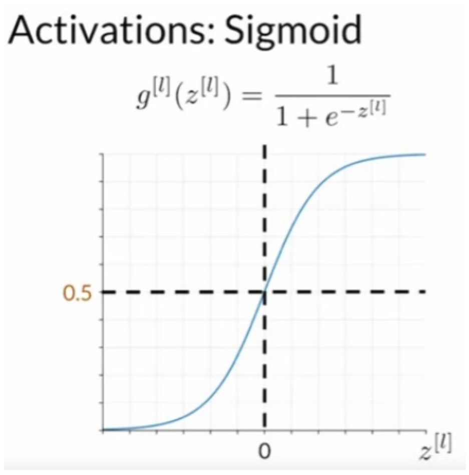
▼ leaky?

$$g^{[l]}(z^{[l]}) = max(\underset{\text{Solves the dying ReLU problem}}{\underline{\overset{0.1}{a}z^{[l]}}}, \underline{z^{[l]}})$$

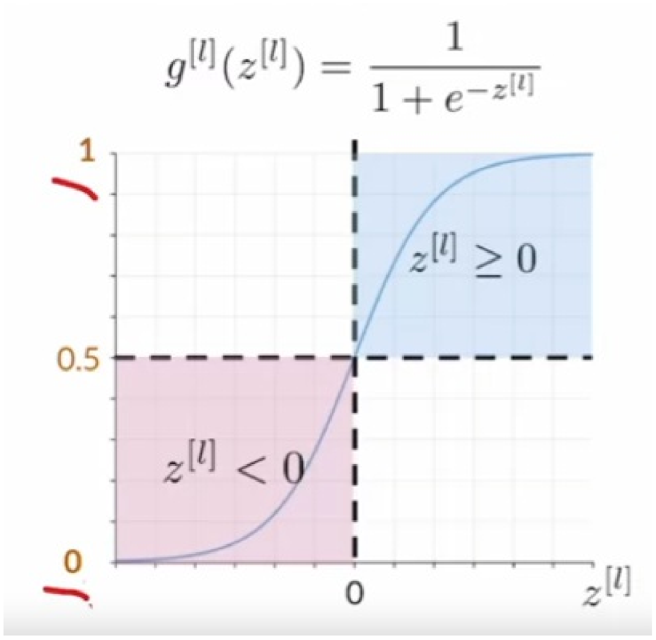
>> compare to ReLU, when the input value is neagtive, the derivative is zero, not having small value (0.1) like in Leaky ReLU, so now, it is kinda 'leaky', not die die zero anymore

> Broadly the slope is treated as a hyperparameter and that's a here, but it's typically set to 0.1, meaning the leak is quite small relative to the positive slope still. This solves the dying ReLU problem by enlarge.

▼ Sigmoid



▼ case study (output values between 0 and 1)

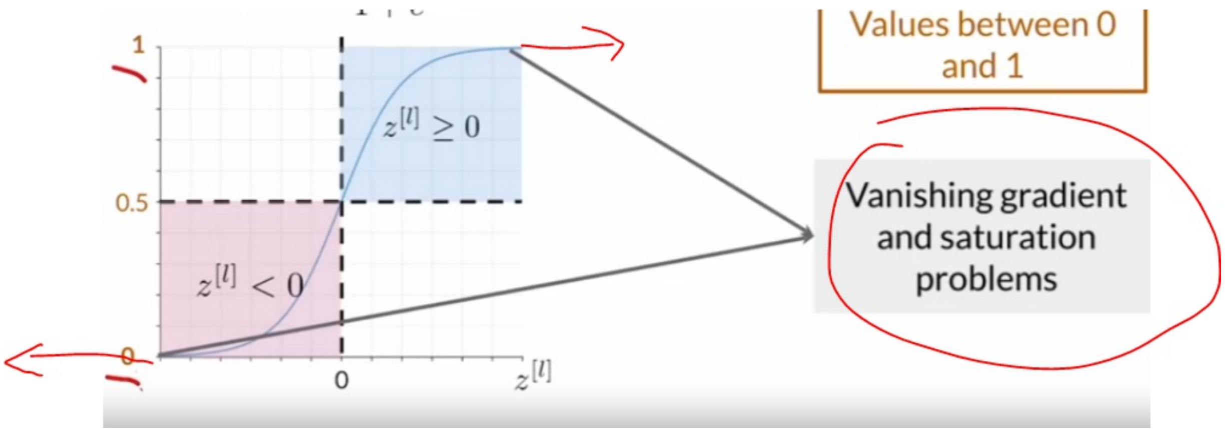


- when z is smaller than zero, the derivative value is ranged between 0~0.5
- when z is bigger than zero, the derivative value is ranged between 0.5 ~ 1

▼ Why Sigmoid always used in binary classification problem?

>> because the output value is between 0 and 1 (i.e fake or real, cat or not cat)

▼ Why sigmoid activation function isn't used very often in hidden layers? (vanishing gradient and saturation problems)

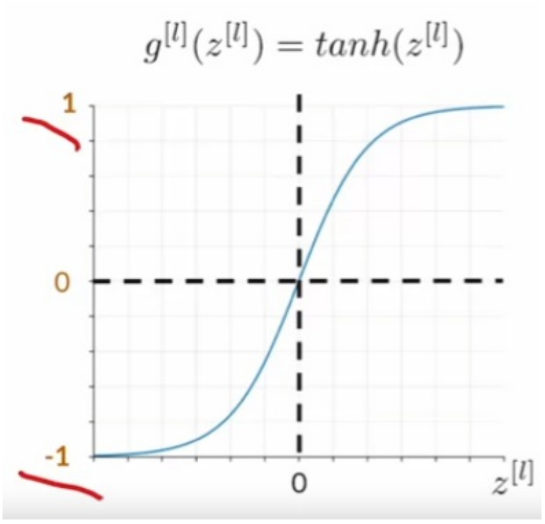


> vanishing gradient >> gradient value often get too close to 1 or to close to zero

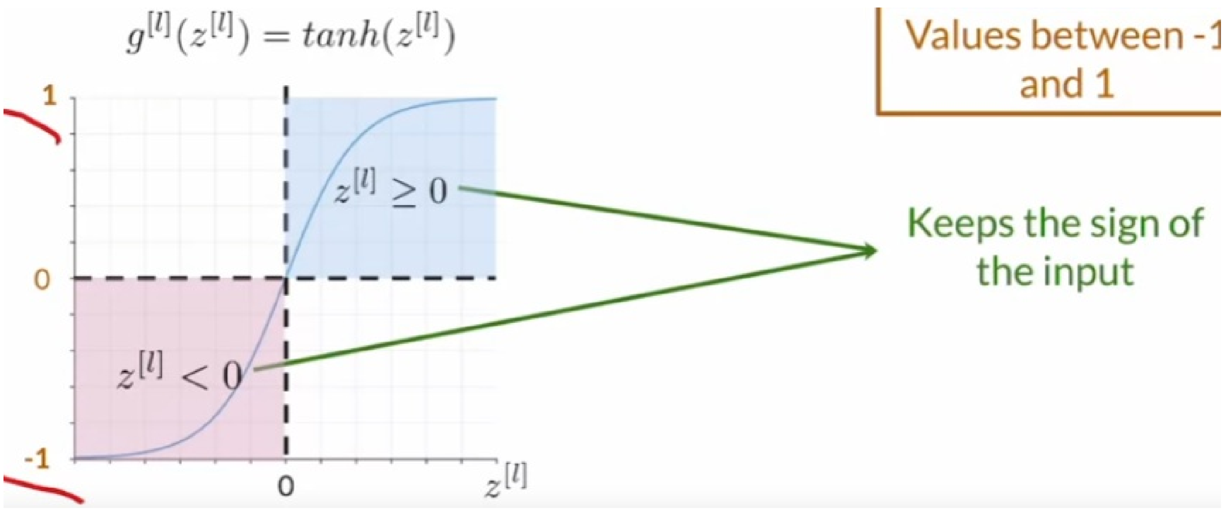
>> ReLu got dying ReLU problem, Sigmoid got vanishing gradient problem (concept is the same, gradient getting vanished)

▼ Tanh (hyperbolic tangent) (has similar shape with Sigmoid)

(has similar shape with Sigmoid>> means has same issues with Sigmoid)

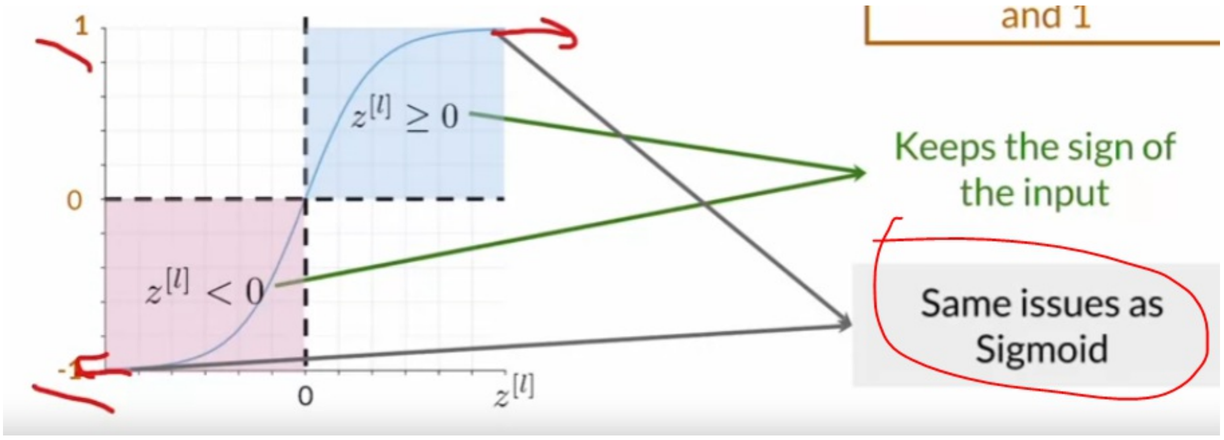


▼ difference betweenn Sigmoid and Tanh?



> One key difference from the sigmoid is that tanh keeps the sign of the input z, so **negatives are still negative**. That **can be useful in some applications**.

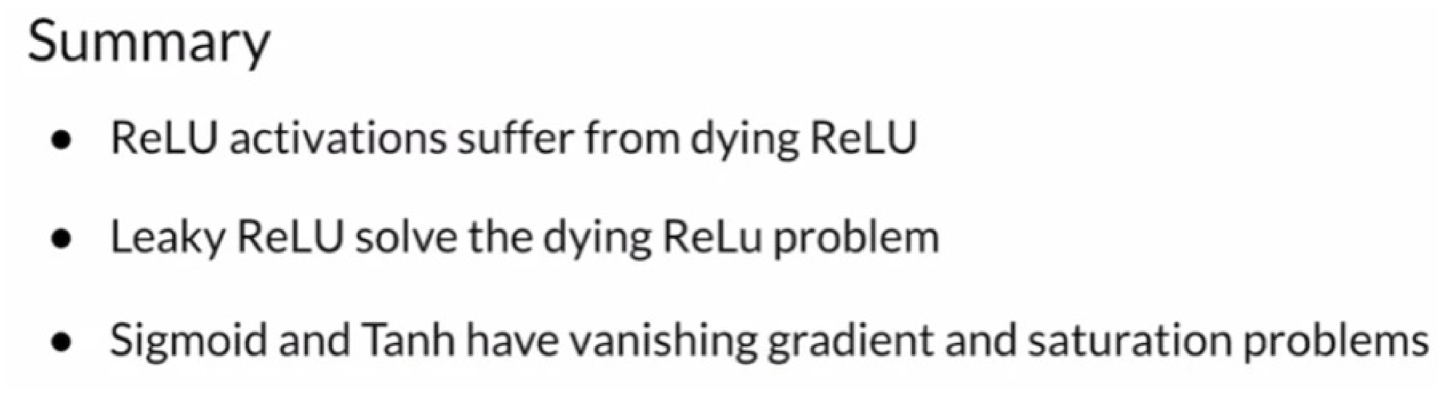
▼ tanh also has same issue with Sigmoid (vanishing gradient and saturation problems)



> vanishing gradient >> gradient value often get too close to 1 or to close to -1 (similar to Sigmoid, just value diff)

▼ activation function isn't used very often in hidden layers

□Summary



Summary

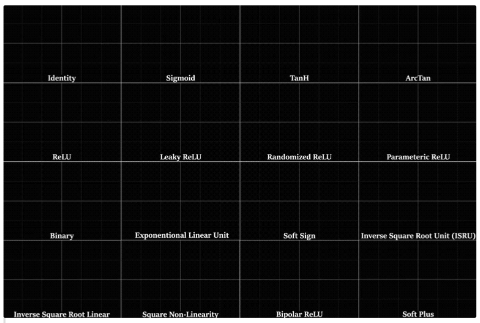
- ReLU activations suffer from dying ReLU
- Leaky ReLU solve the dying ReLu problem
- Sigmoid and Tanh have vanishing gradient and saturation problems

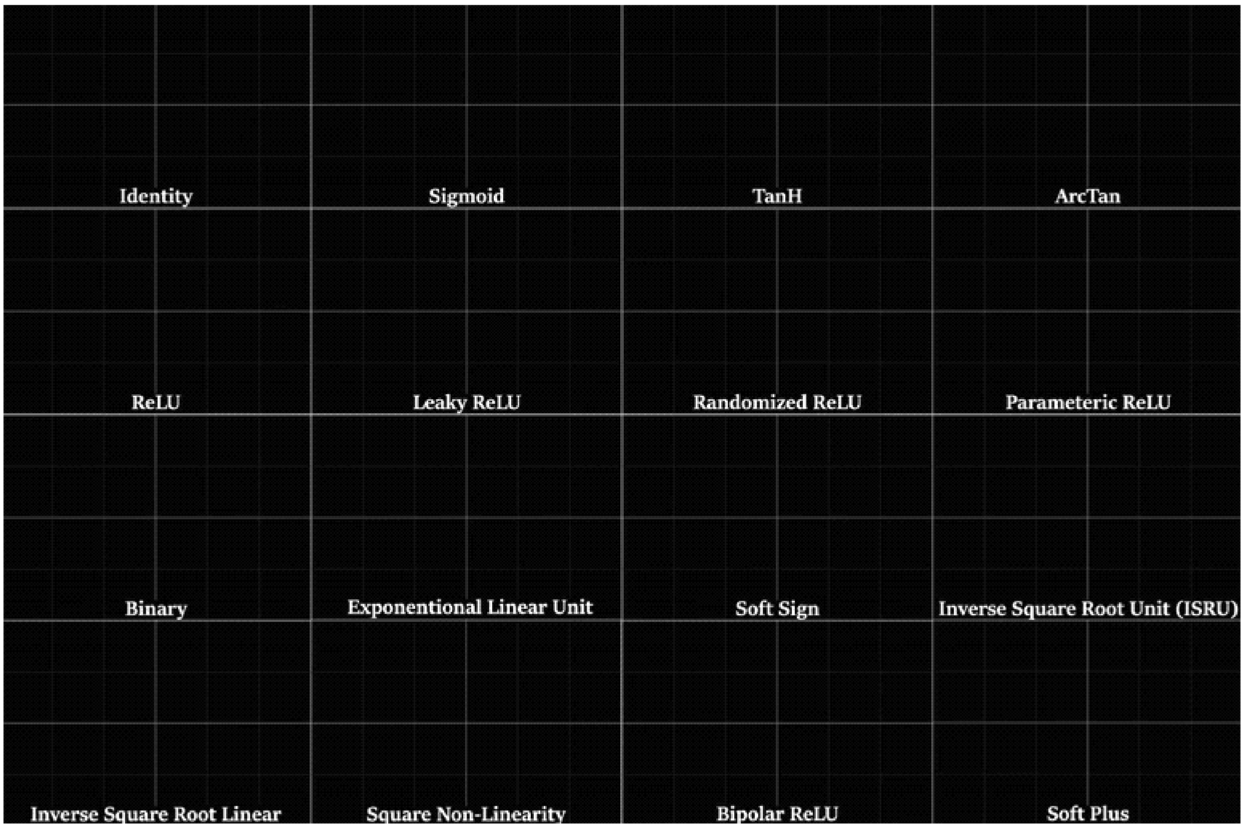
URL

Activation Functions Explained - GELU, SELU, ELU, ReLU and more

During the calculations of the values for activations in each layer, we use an activation function right before deciding what exactly the activation value should be. From the previous activations, weights and biases in each layer, we calculate a value for every activation in the next layer.

https://mlfromscratch.com/activation-functions-explained/#small-overview




Identity
Sigmoid
TanH
ArcTan
ReLU
Leaky ReLU
Randomized ReLU
Parameteric ReLU
Binary
Exponentional Linear Unit
Soft Sign
Inverse Square Root Unit (ISRU)
Inverse Square Root Linear
Square Non-Linearity
Bipolar ReLU
Soft Plus