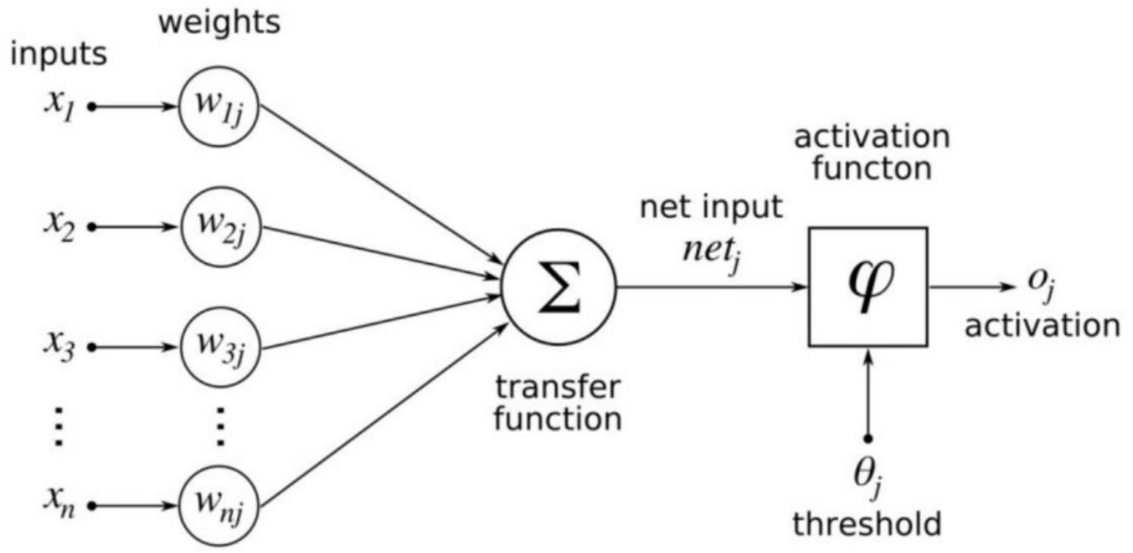
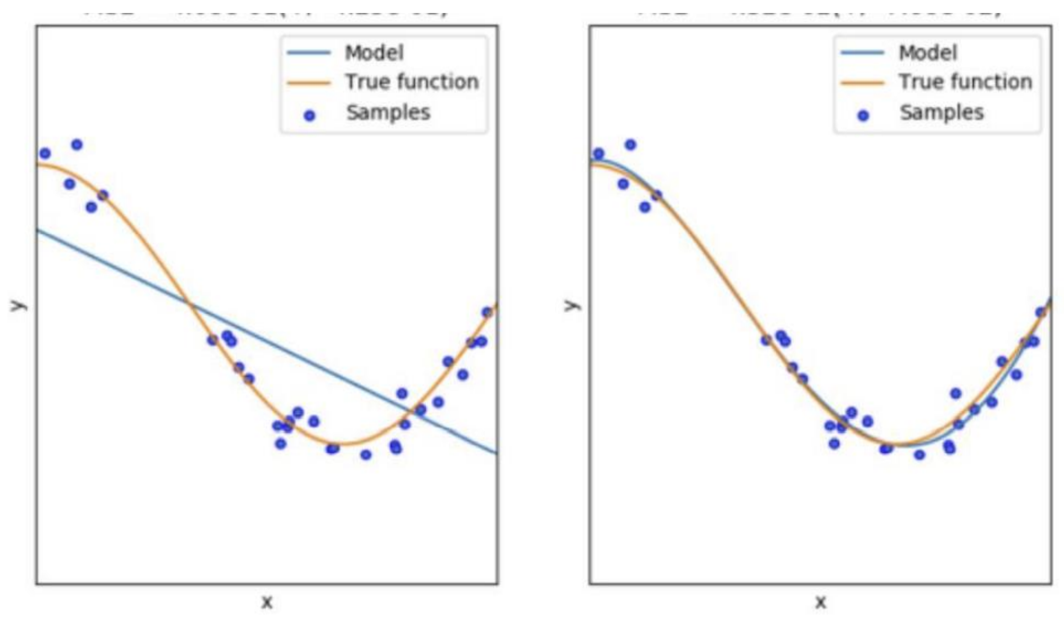
Activation Function
ReLu

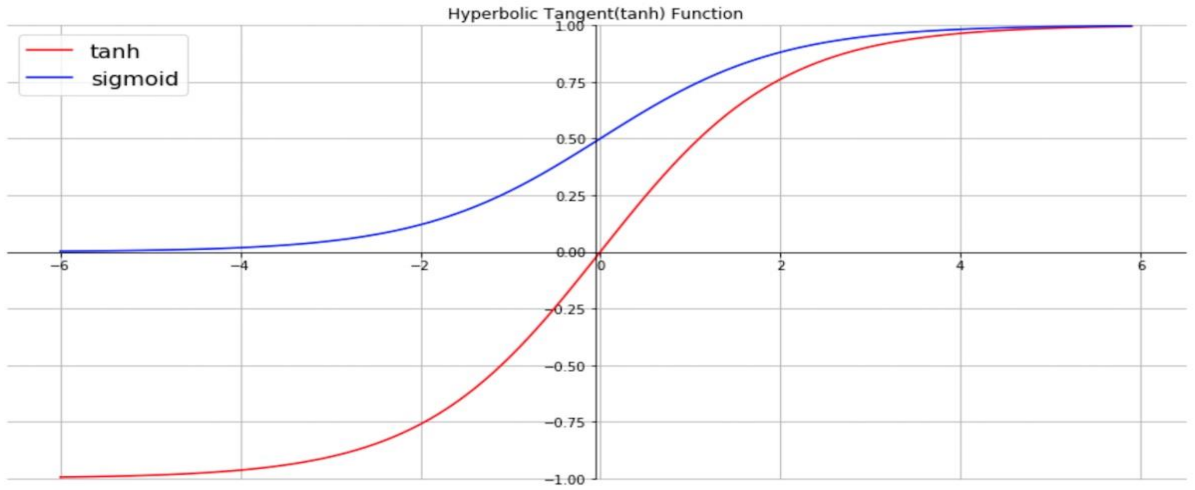
1. Activation Function
2. ReLu
3. ReLu Family

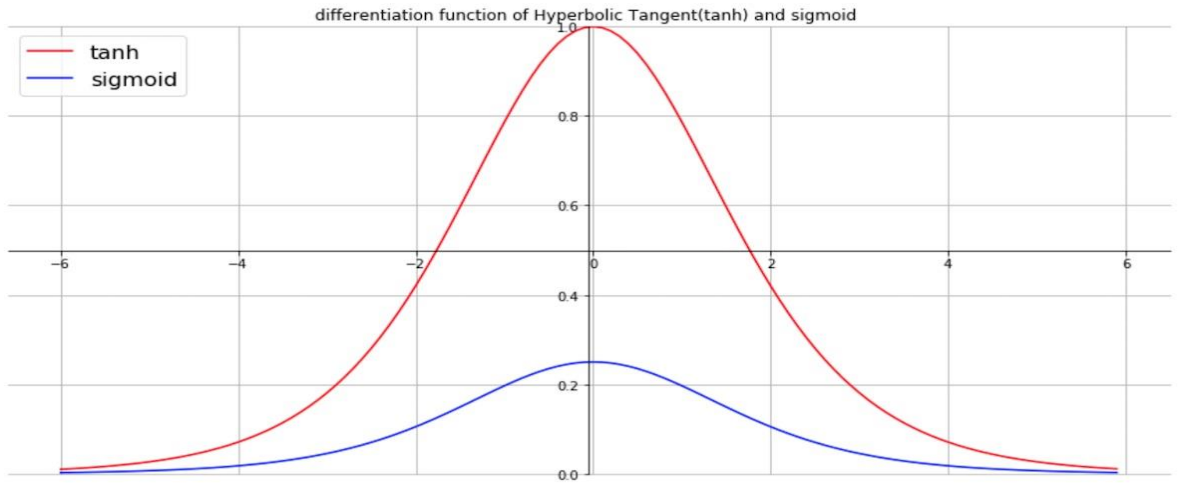
1. Activation Function





Sigmoid, tanh





Saturation Problem

- Large values snap to 1.0, small values to 0 and -1.

- Sensitive to changes around their mid-point of their input.

- Fail to receive useful gradient information, challenging for the learning weights to improve the performance of the model.

Sigmoid, tanh -> ReLu

- The sigmoid and hyperbolic tangent activation functions cannot be used in networks with many layers due to the vanishing gradient problem(Saturation).

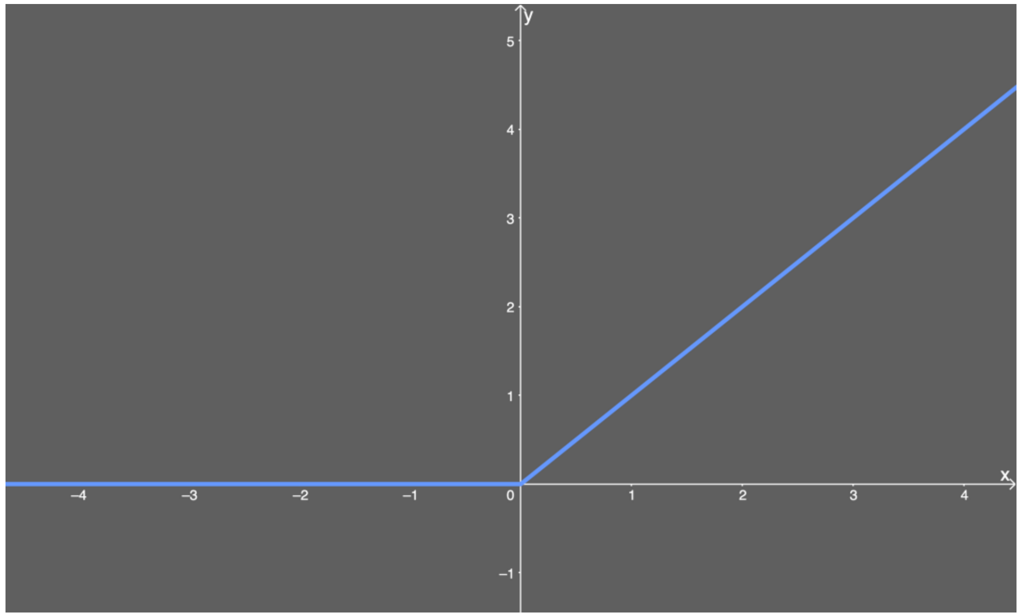
- The ReLu activation function overcomes the vanishing gradient problem, allowing models to learn faster and perform better.

Desirable Property

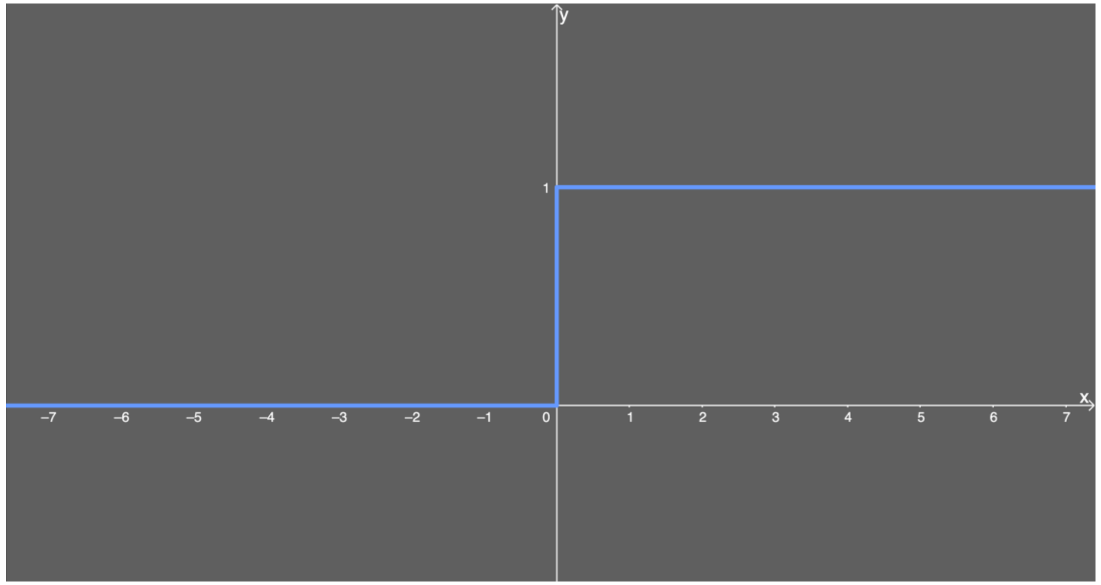
- 1. Unbounded above
  - Preventing saturation which generally causes training to drastically slow down due to near-zero gradients
- 2. Bounded below
  - Strong regularization effects.
- 3. Non-monotonic
  - Causes small negative inputs to be preserved as negative outputs, which improves expressivity and gradient flow.
- 4. Continuity / smooth
  - Improves expressivity and gradient flow, and helping with effective optimization and generalization
  - not continuously differentiable causing some undesired problems in gradient-based optimization.

2. ReLU (Rectified Linear Unit)

$f(x) = \max(0, x)$



Bounded below, Unbounded above,
Monotonic, non-continuous



1. Computational simplicity
   - No exp() func
2. Representational Sparsity
   - True zero value
3. Not saturating (Unbounded above)
   - $[0, \infty)$
4. Linear behavior
   - easy to optimize

ReLu Problem

1. Non-differentiable at zero

not continuously differentiable causing some undesired problems in gradient-based optimization.

2. Not zero-centered output

Slow updata problem

3. Vanishing gradient problem (- region)

All negative input will be 0.

4. Dying ReLu

Large learning rate and large negative BIAS makes the input negative -> relu(x) = 0

These problems negatively affect the model training -> relu family

3. ReLu Family

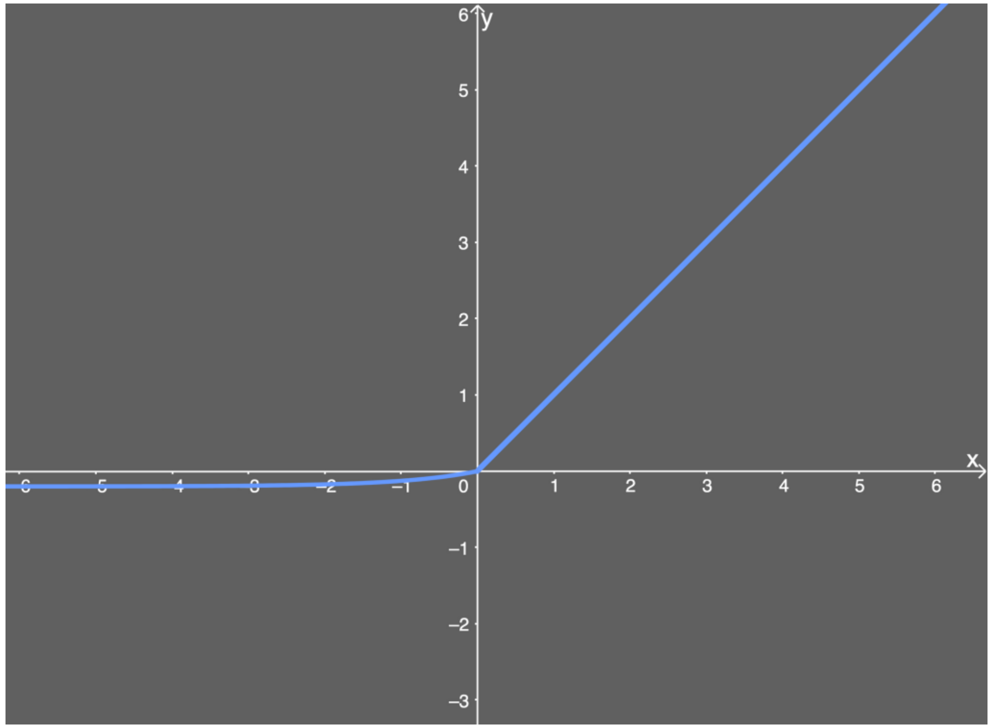
- ELU
- Leaky ReLu
- SELU

- GELU
- Mish
- Swish

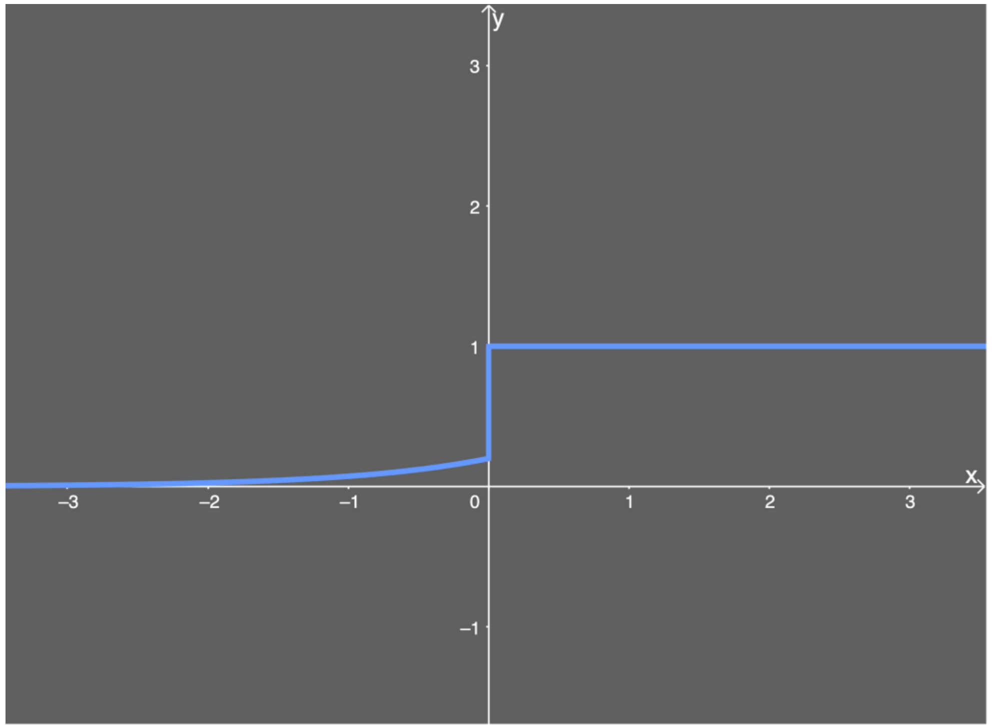
ELU (Exponential Linear Unit)

ELU(x) = x if x>0

α(e^x−1) if x<0



Bounded below, unbounded above,
Monotonic, non-continuous



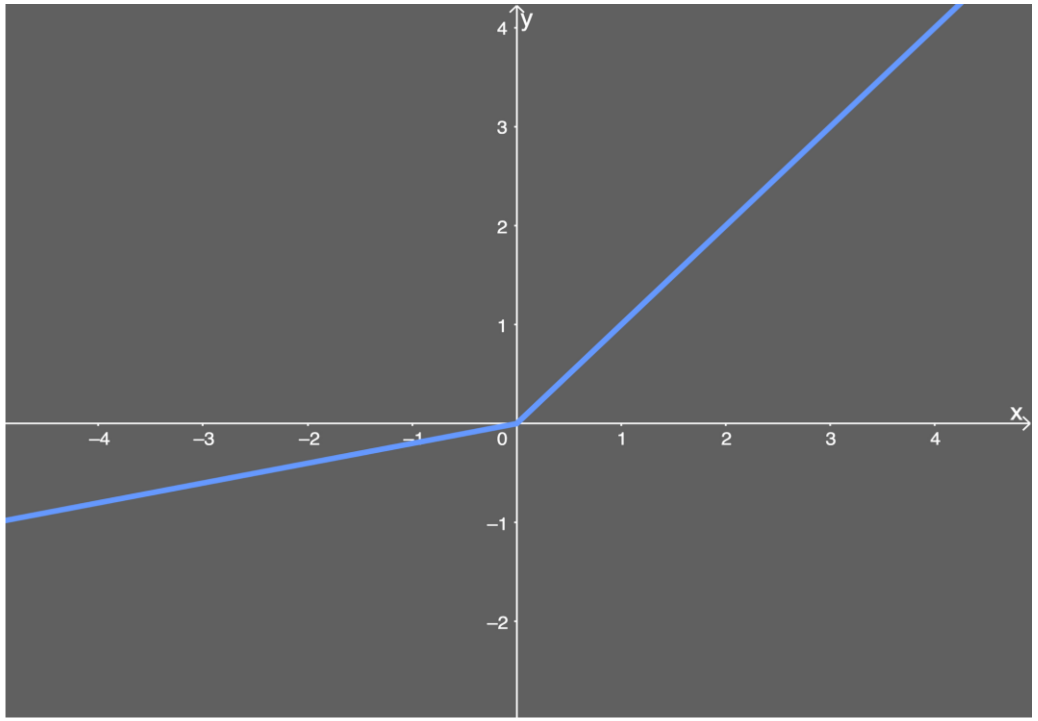
Produce negative output,
Not zero gradient when x<0
-> Preventing Vanishing gradient
-> Avoiding the dying relu.

- calculate exp() -> time expensive
- The NN does not learn the alpha value

Leaky ReLu

$f(x) = \max(\alpha x, x)$

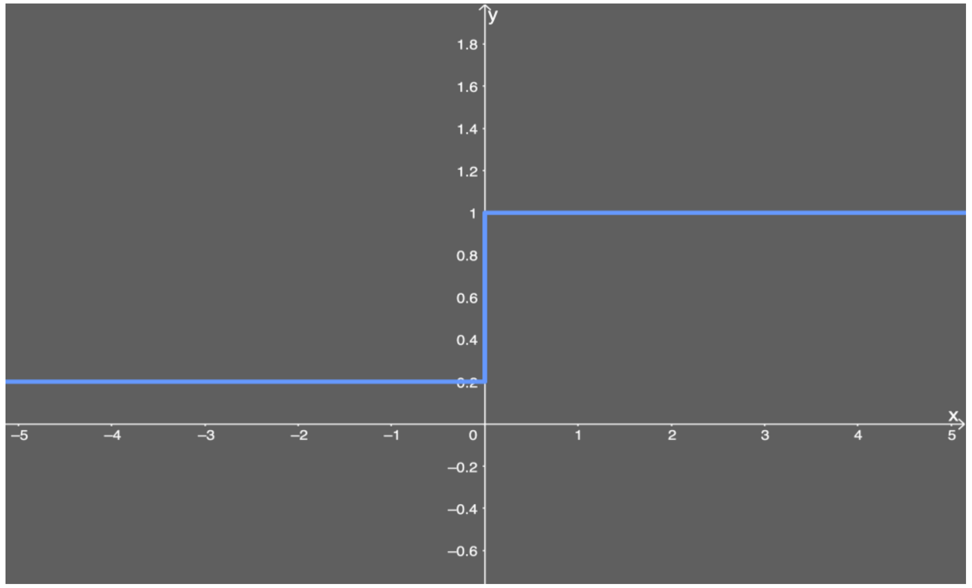
Where $\alpha = 0.2$



Unbounded above,

Unbounded below,

Monotonic, non-continuous



Produce negative output,

Not zero gradient when $x<0$

-> Preventing Vanishing gradient

-> Avoiding the dying relu.

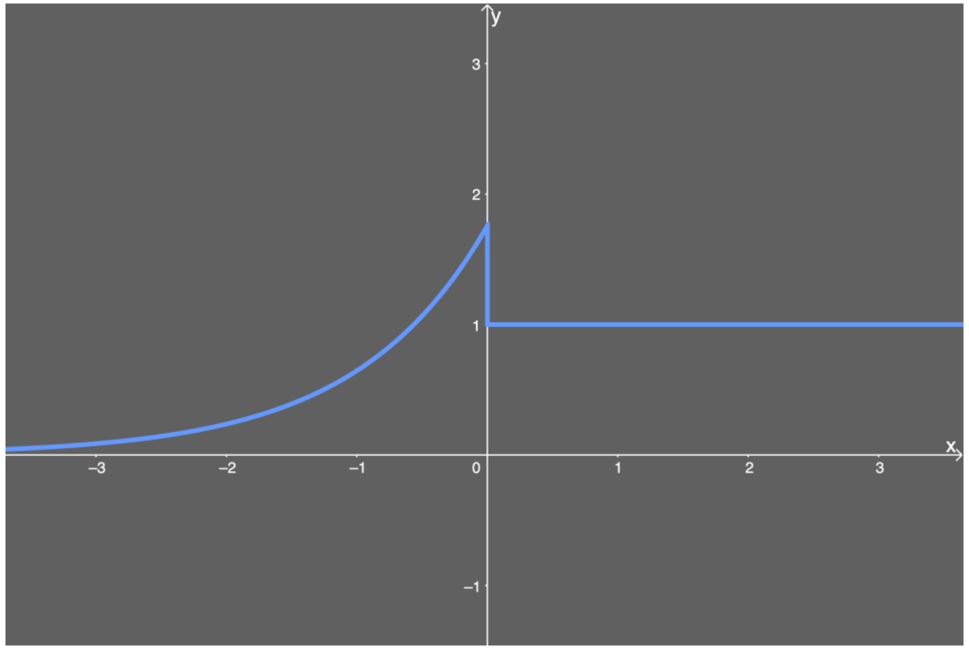
Faster than ELU (not exp)

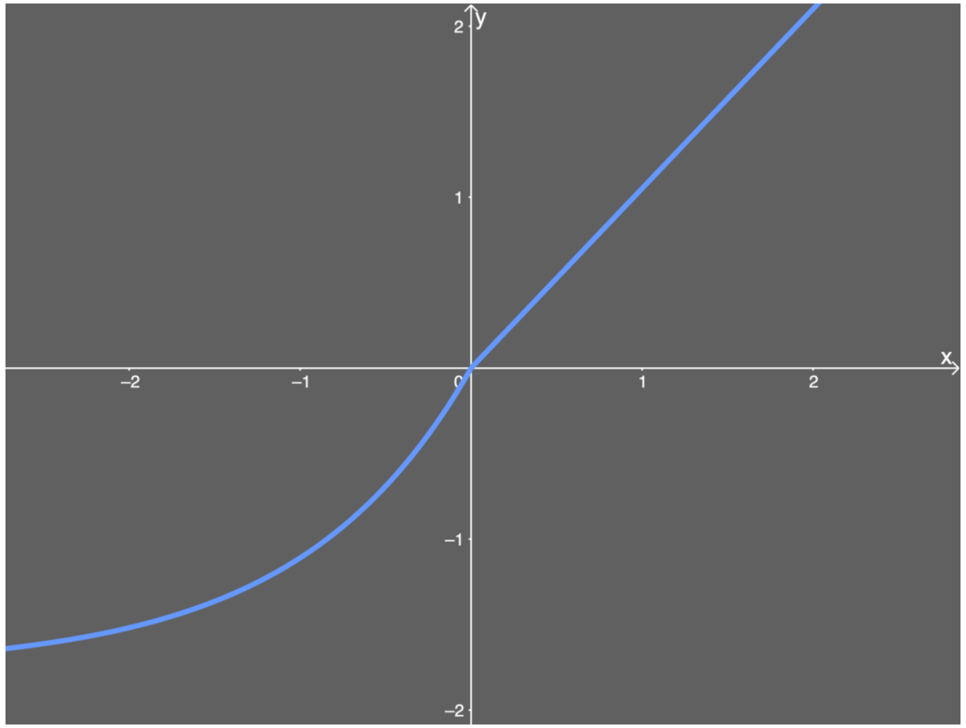
- The NN does not learn the alpha value.
- Becomes a linear function, when it is differentiated, whereas ELU is partly linear and nonlinear.
- Unbounded below

SELU (Scaled Exponential Linear Unit)

SELU(x) = λx if x>0

λα(e^x−1) if x≤0





•Internal self-normalization is faster than external normalization, which means the network converges faster.

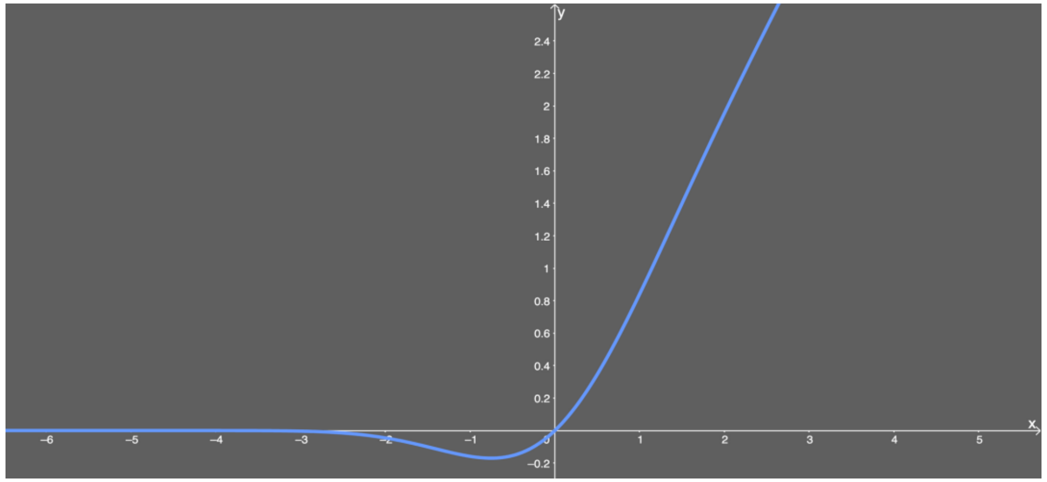
•Vanishing and exploding gradient problem is impossible.

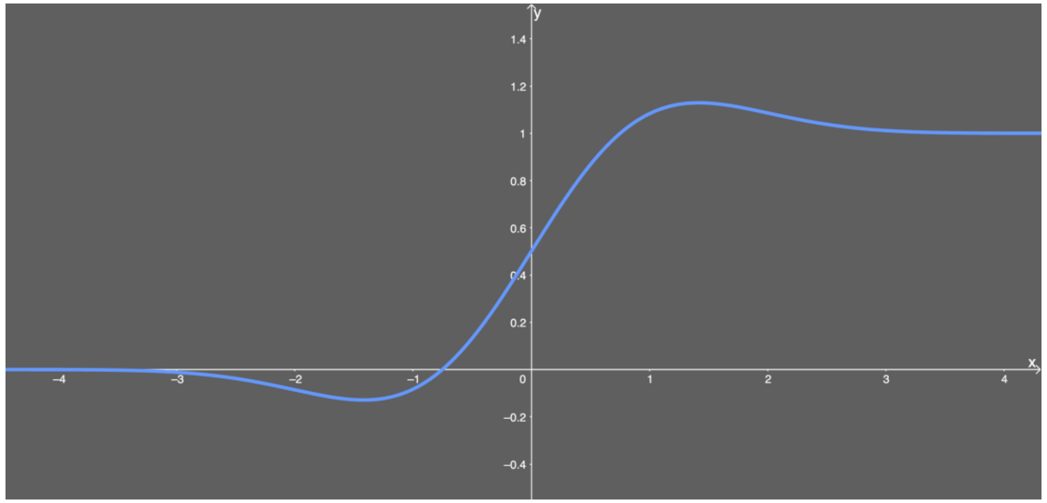
- Fairly new in practical use

Bounded below, unbounded above,

Monotonic, non-continuous

GELU (Gaussian Error Linear Unit)





GELU = dropout + ReLu

$X \sim N(0,1)$

$$\mathrm{GELU}(x) = x * P(X<x) + 0 * (1-P(X<x)) = x\Phi(x)$$
$$= 0.5x(1+\tanh(\sqrt{2/\pi}(x+0.044715x^3)))$$

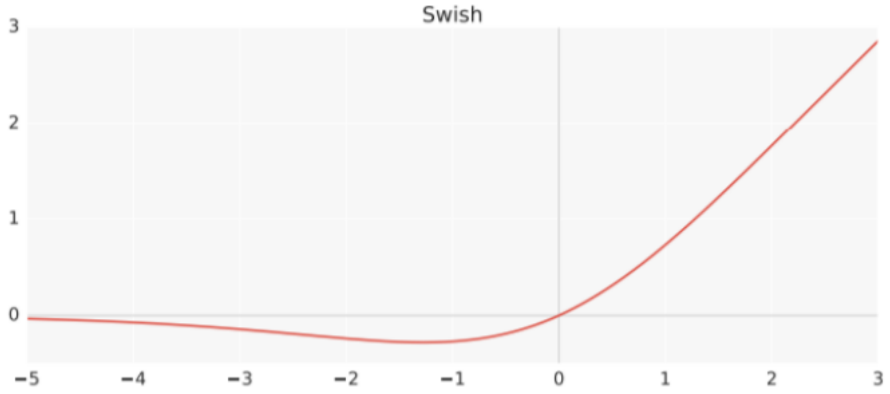
Bounded below, Unbounded above
Non-monotonic, Smooth / continuity

- Perform best in NLP model.
- Avoids vanishing gradients problem

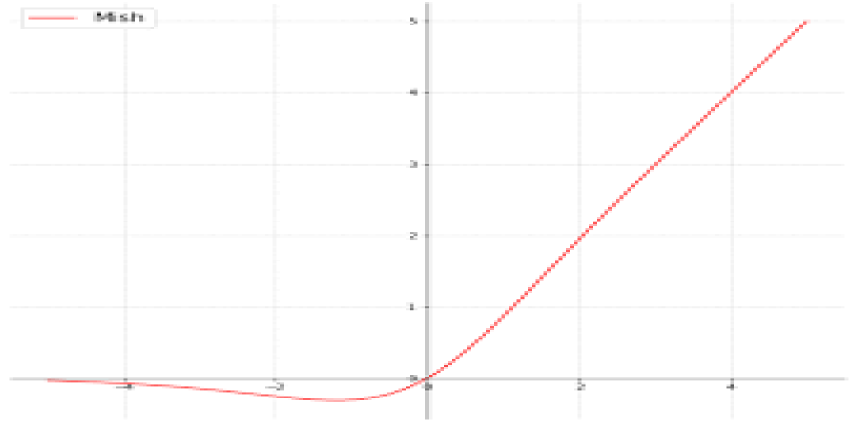
- Fairly new in practical use

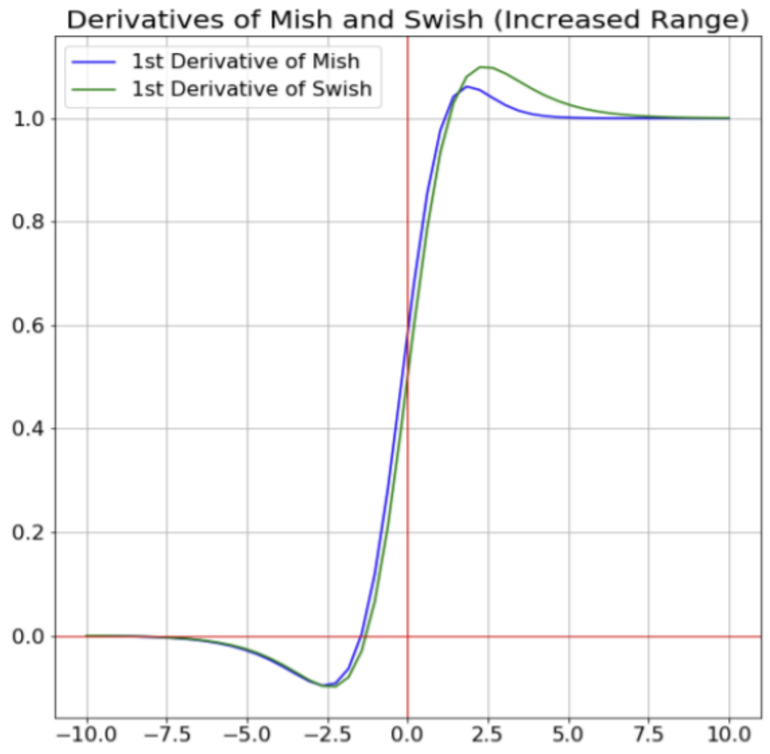
Swish, Mish

Swish(x) = x * sigmoid(x)



Mish(x) = $x \cdot tanh$(ln(1 + e^x))





Bounded below
Unbounded above
Non-monotonic
Smooth / continuity

Swish, Mish (non-monotonic)

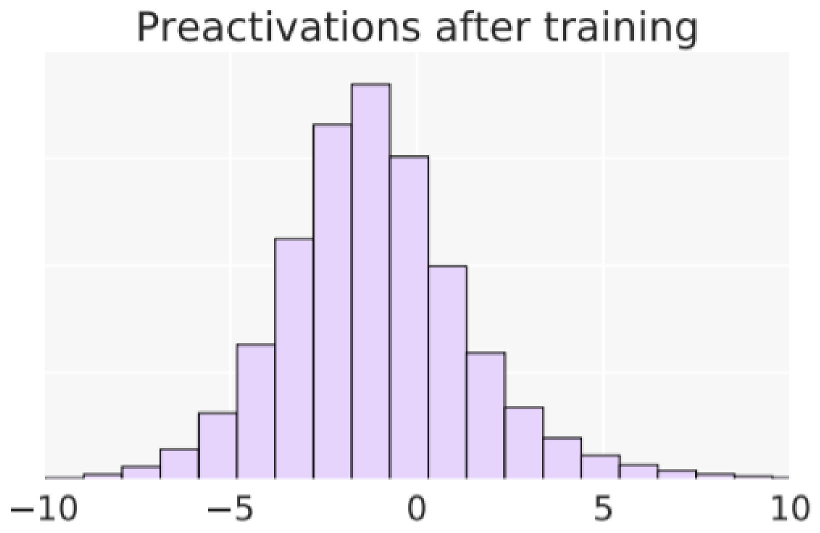


Figure 6: Preactivation distribution after training of Swish with $\beta = 1$ on ResNet-32.

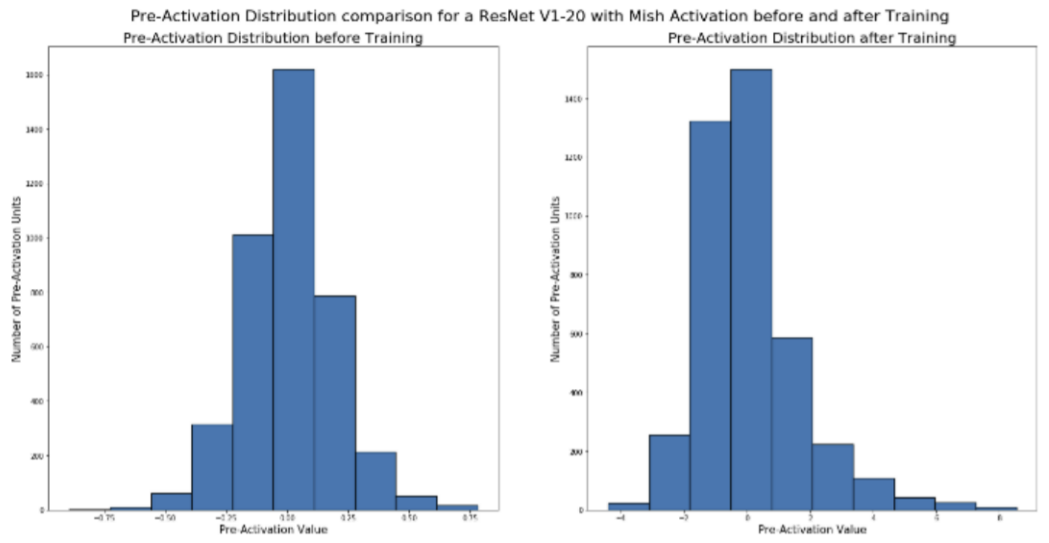


Figure 4. Pre-Activation Distribution comparison before and after training for a ResNet v1-20 with Mish.

The most striking difference with ReLU is the non-monotonic “bump” of them when $x < 0$. A large percentage of ‘pre-activations’ fall inside the domain of the bump ($-5 \le x \le 0$), which indicates that the non-monotonic bump is an important aspect of them.

Pre-activation?

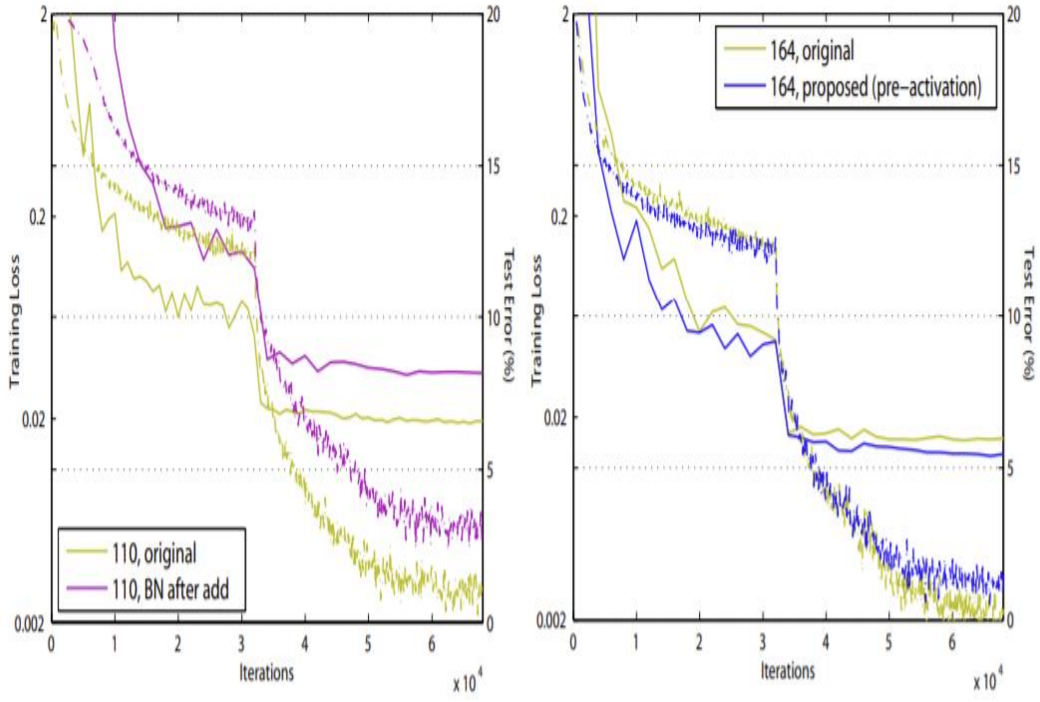


Figure 6. Training curves on CIFAR-10. **Left**: BN after addition (Fig. 4(b)) using ResNet-110. **Right**: pre-activation unit (Fig. 4(e)) on ResNet-164. Solid lines denote test error, and dashed lines denote training loss.

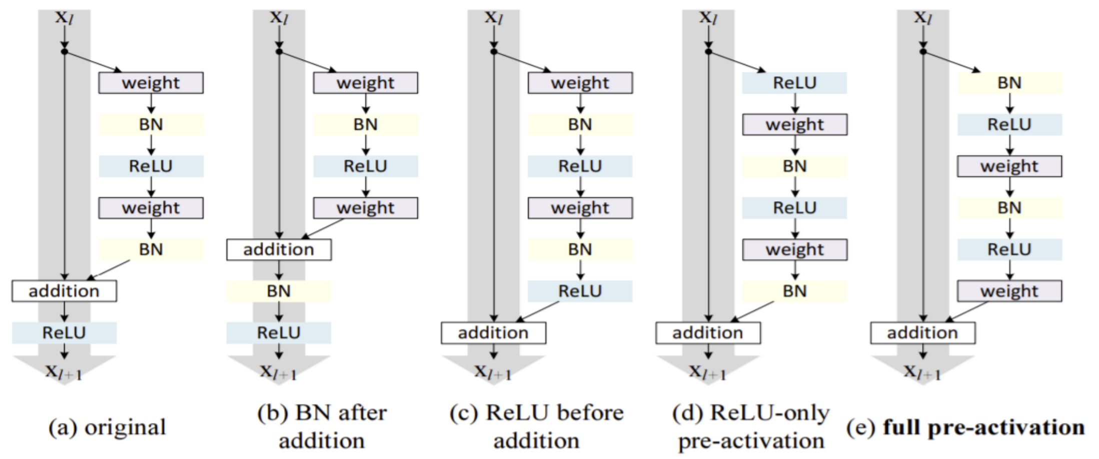
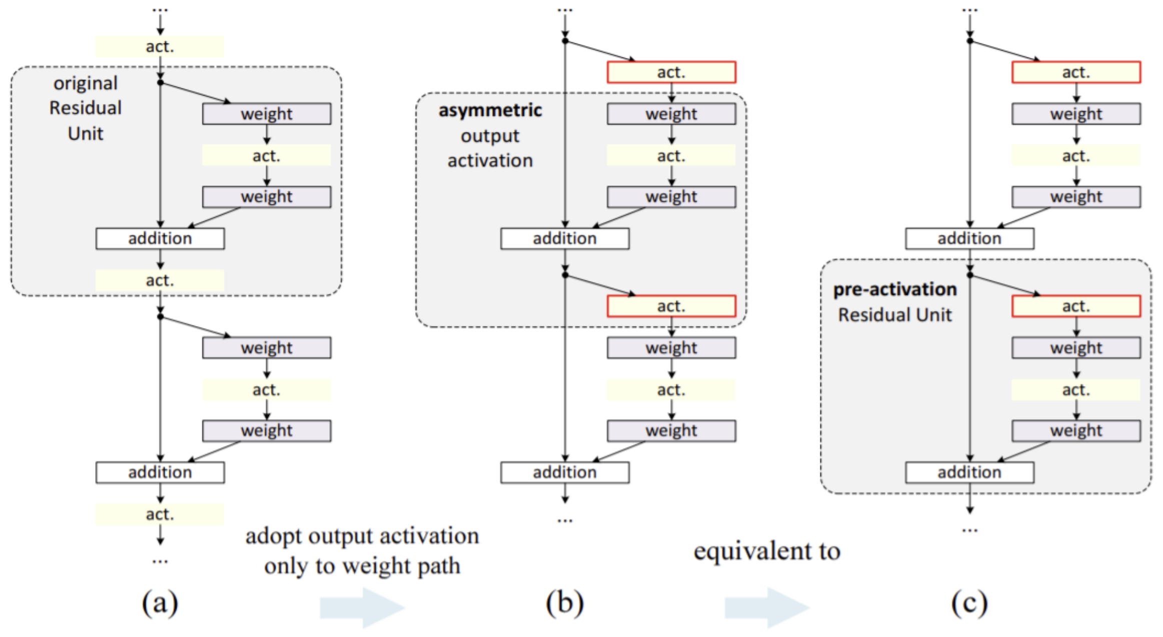


Figure 4. Various usages of activation in Table 2. All these units consist of the same components — only the orders are different.



original Residual Unit
asymmetric output activation
pre-activation Residual Unit
act.
weight
addition
adopt output activation only to weight path
equivalent to
(a)
(b)
(c)

Figure 5. Using asymmetric after-addition activation is equivalent to constructing a *pre-activation* Residual Unit.

Swish, Mish (smooth)

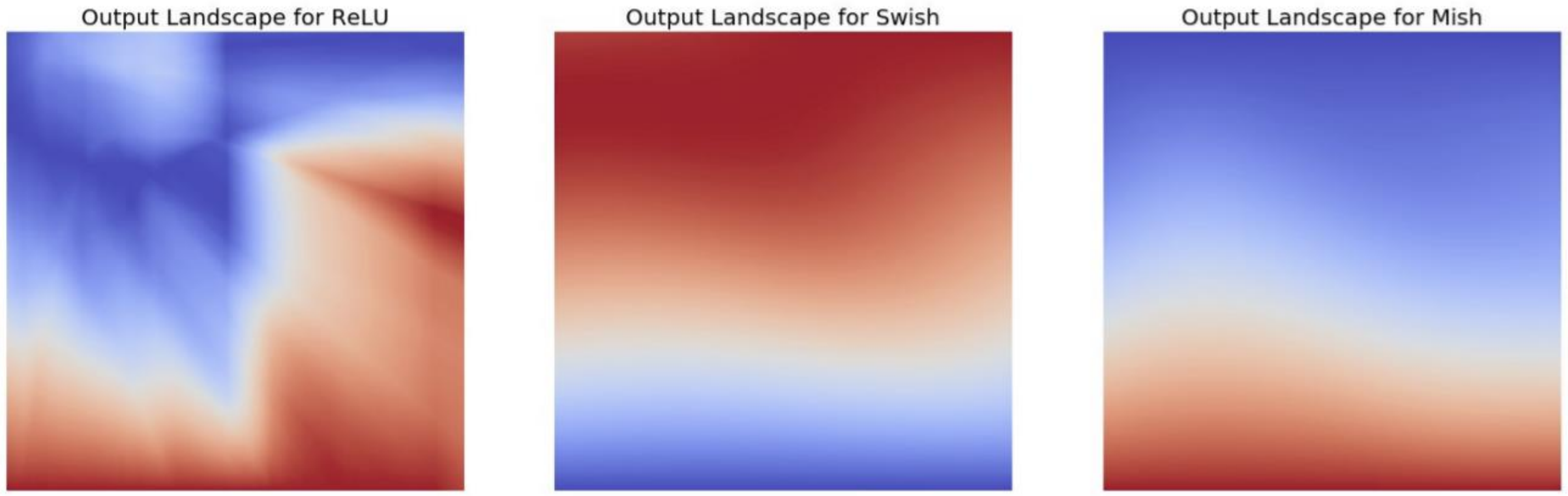


Figure 5. Output Landscape of a Random Neural Network with ReLU, Swish and Mish

Smooth function helps with effective optimization and generalization.
The output landscape of 5 layer randomly initialized neural network was compared for ReLU, Swish, and Mish. The Figure 5 clearly depicts the sharp transition between the scalar magnitudes for the coordinates of ReLU as compared to Swish and Mish.
Smoother transition results in smoother loss functions which are easier to optimize and hence the network generalizes better.

Swish, Mish

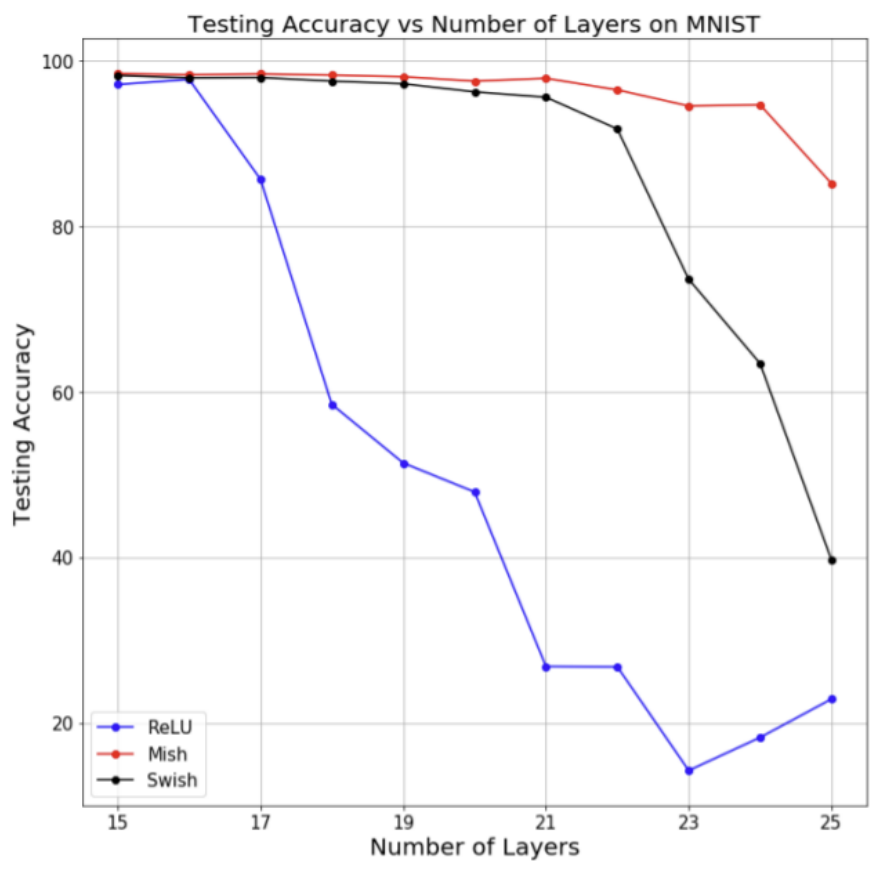


Figure 6. Testing Accuracy v/s Number of Layers on MNIST for Mish. Swish and ReLU.

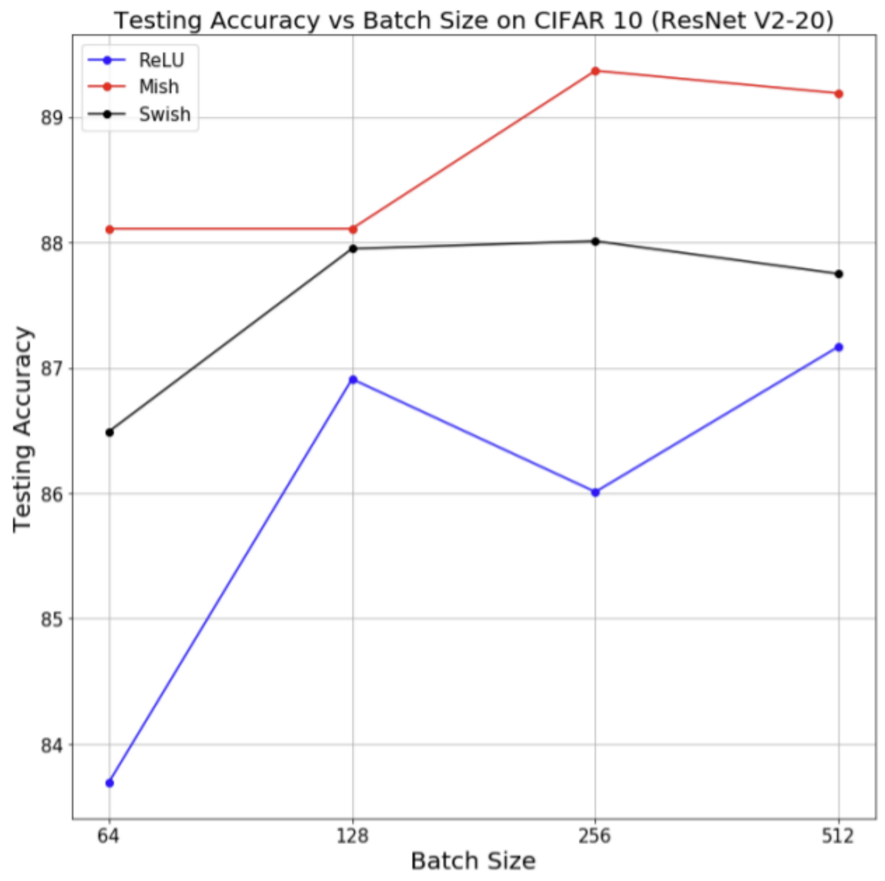


Figure 7. Test Accuracy v/s Batch Size on CIFAR-10 for Mish, Swish and ReLU.

Good for more complex model

References

SEARCHING FOR ACTIVATION FUNCTIONS (https://arxiv.org/pdf/1710.05941.pdf)

WHY RELU UNITS SOMETIMES DIE: ANALYSIS OF SINGLE-UNIT ERROR BACKPROPAGATION IN NEURAL NETWORKs (https://arxiv.org/pdf/1812.05981v1.pdf)

Mish: A Self Regularized Non-Monotonic Neural Activation Function (https://arxiv.org/vc/arxiv/papers/1908/1908.08681v2.pdf)

Self-Normalizing Neural Networks (https://arxiv.org/pdf/1706.02515.pdf)

Identity Mappings in Deep Residual Networks (https://arxiv.org/pdf/1603.05027.pdf)

https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/
https://mlfromscratch.com/activation-functions-explained/#/