Machine Learning

Activation Functions

Curtis Larsen

Utah Tech University—Computing

Spring 2025

Objectives

Objectives:

- Review - Activation Functions in Neural Networks
- Review - Back-propagation
- Understand - Vanishing and Exploding Gradients
- Identify - Common Activation Functions
- Initialize - Best Practices per Activation Function

Review

Activation Functions

$$\phi_w(x) = \sum_{i=0}^{n} w_i x_i$$

Loss and Gradient

Loss function

$$J(\boldsymbol{\theta})$$

Gradient w.r.t $\boldsymbol{\theta}$

$$\boldsymbol{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

Back-propagation

- Forward pass (compute)
- Backward pass (error propagation)
- Vanishing gradients
- Exploding gradients

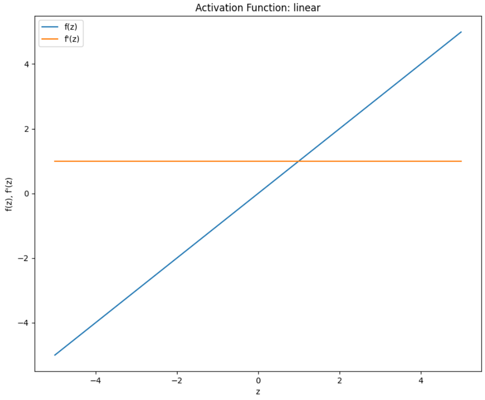
Common Activation Functions

Linear

$$f(x) = x$$

Notes:

- Smooth
- Fast
- `linear`

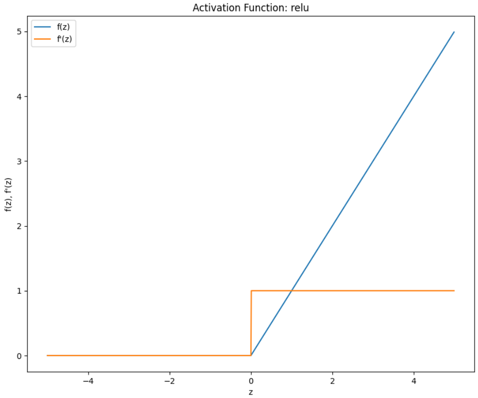


Rectified Linear

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases}$$

Notes:

- Not smooth
- Fast
- No gradient on left side (neurons "die")
- `relu`

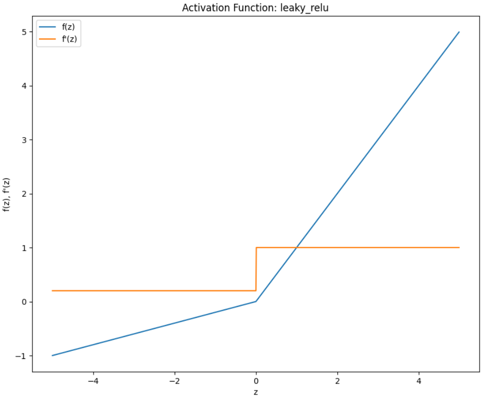


Leaky ReLU

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha x & \text{if } x \leq 0 \end{cases}$$

Notes:

- Not smooth
- Fast
- Small gradient on left side (neurons don't "die")
- `leaky_relu`

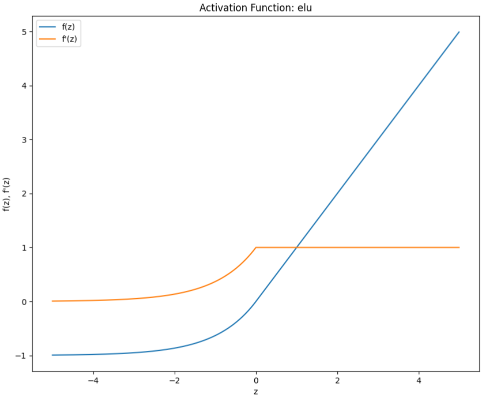


Exponential Linear Unit

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha(e^x - 1) & \text{if } x \leq 0 \end{cases}$$

Notes:

- Smooth if $\alpha = 1$
- Slower to compute
- Gradient descent may converge faster
- Small gradient on left side (neurons don't "die")
- `elu`

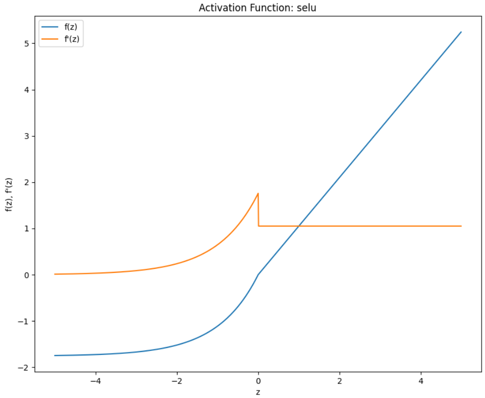


Scaled ELU

$$f(x) = \begin{cases} sx & \text{if } x > \\ s\alpha(e^x - 1) & \text{if } x \leq \end{cases}$$

Notes:

- Only useful on stacks of dense layers
- `selu`

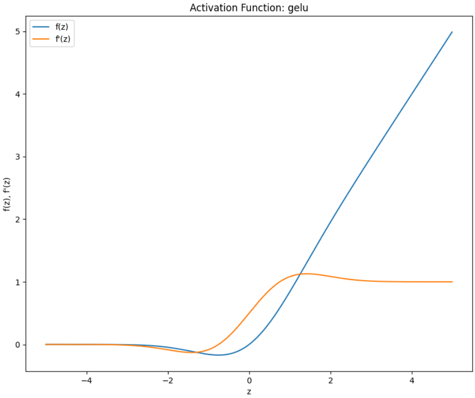


Gaussian ELU

$$f(x) = x\Phi(x)$$

Notes:

- $\Phi(x)$ is the Gaussian cumulative distribution function
- More computationally expensive
- Maybe faster convergence
- `gelu`

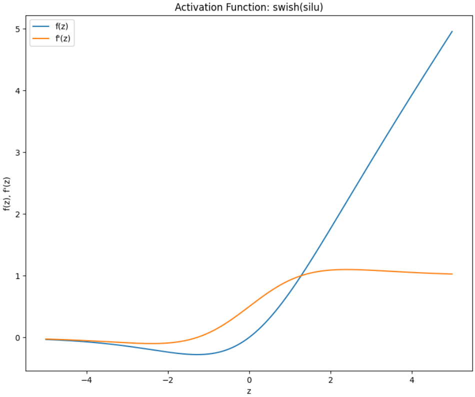


Sigmoid Linear Unit (Swish)

$$f(x) = x\sigma(x)$$

Notes:

- $\sigma(x)$ is the sigmoid function
- Sometimes better than GELU
- `silu`, `swish`

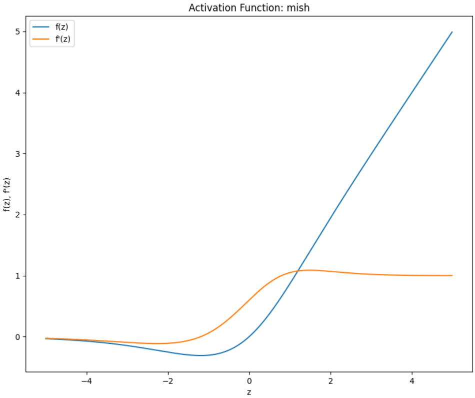


MISH

$$f(x) = x\tanh(\log(1+e^x))$$

Notes:

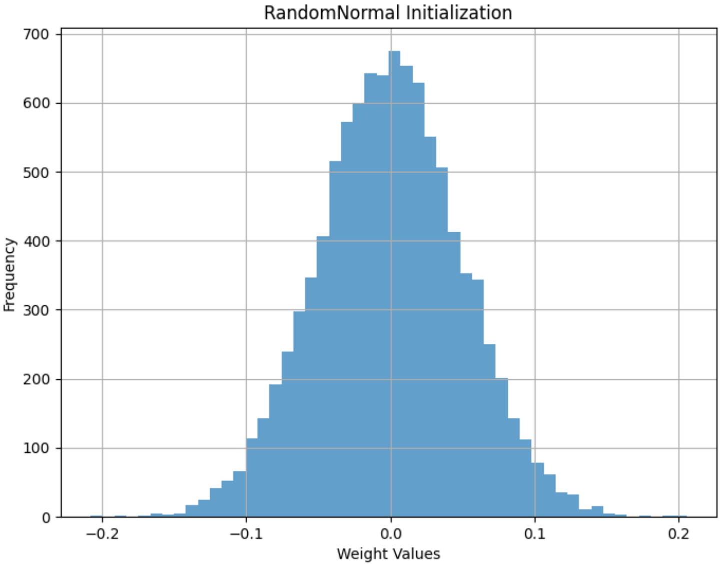
- Sometimes better than GELU, Swish
- `mish`

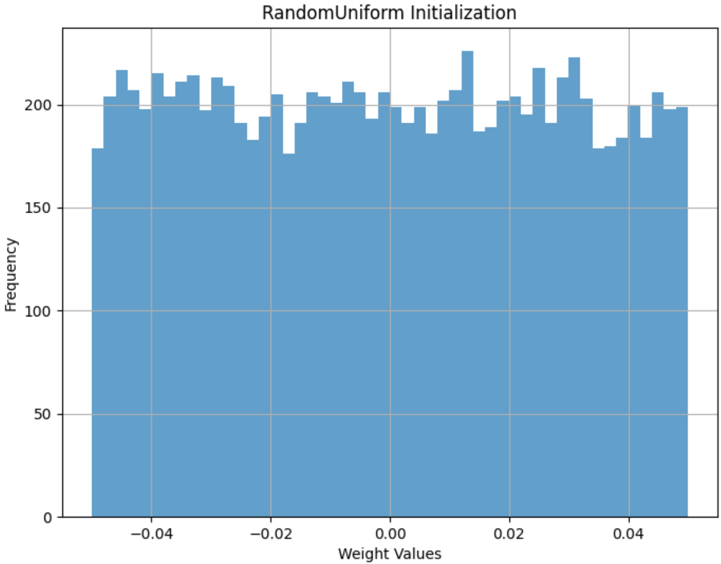


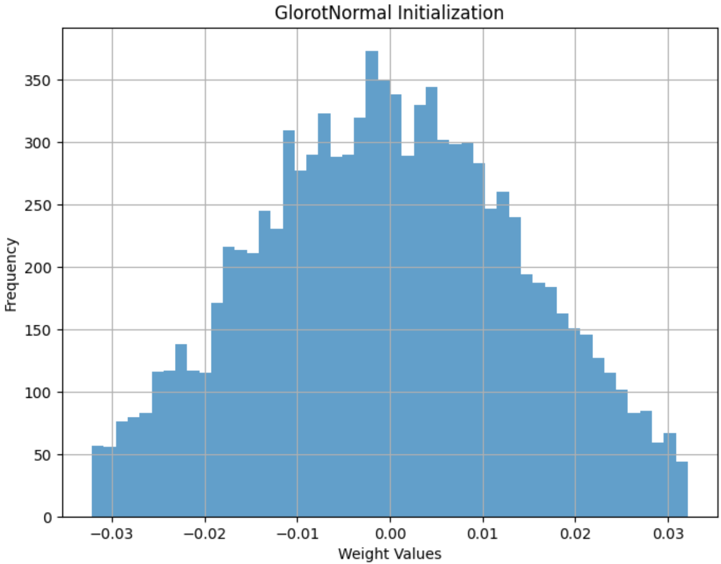
Implementation

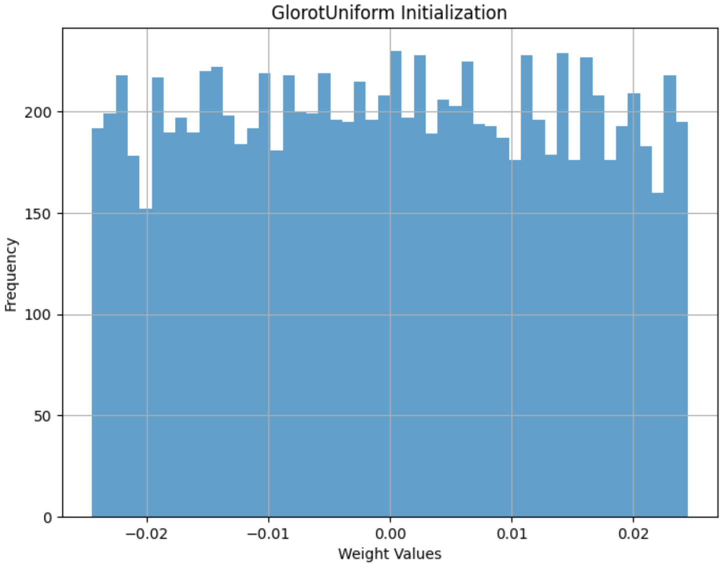
```
model.add(keras.layers.Dense(units, activation="leaky_relu"))
```

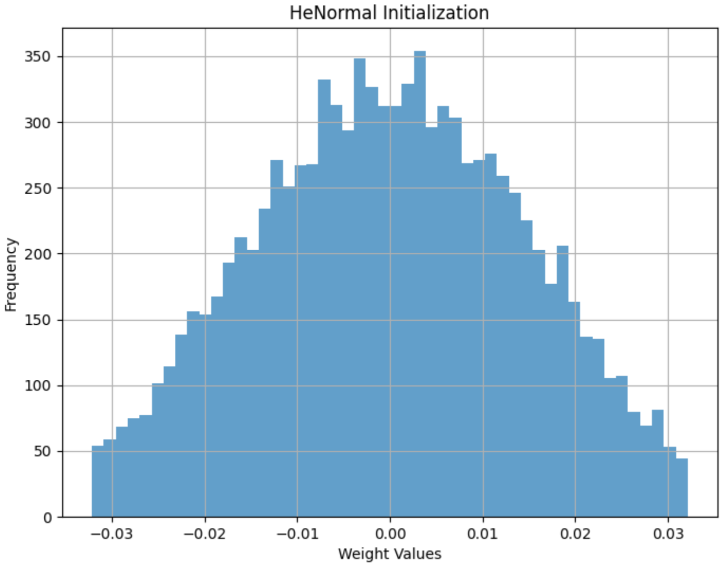
Weight Initialization

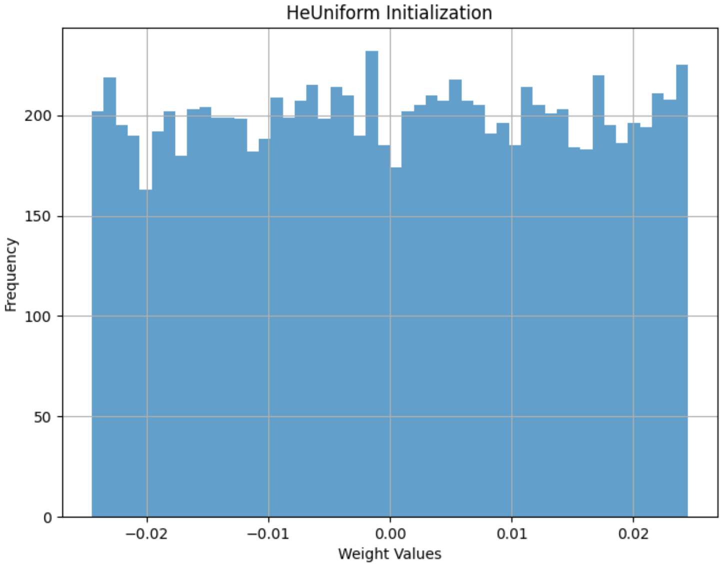

RandomNormal Initialization
Frequency
Weight Values
0
100
200
300
400
500
600
700
−0.2
−0.1
0.0
0.1
0.2

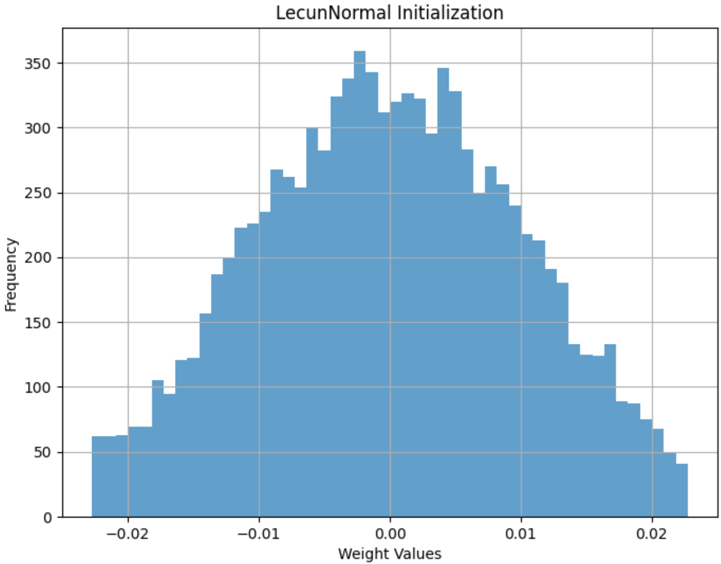

RandomUniform Initialization
Frequency
0
50
100
150
200
−0.04
−0.02
0.00
0.02
0.04
Weight Values

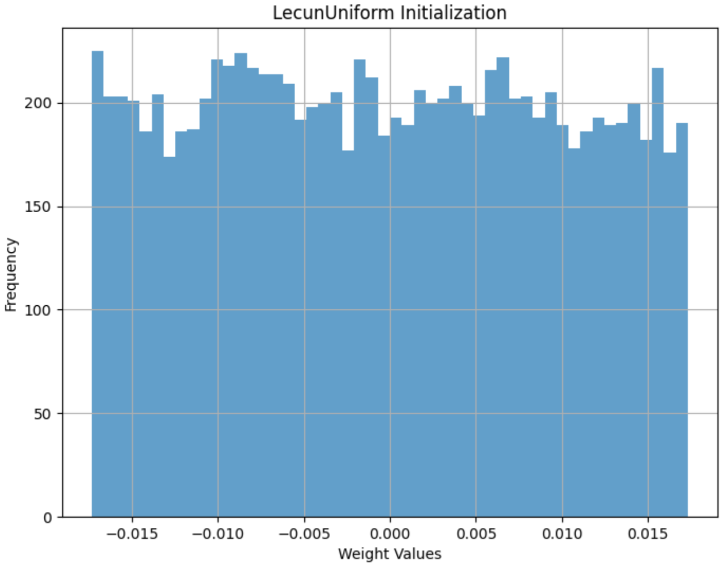

GlorotNormal Initialization
Frequency
350
300
250
200
150
100
50
0
−0.03
−0.02
−0.01
0.00
0.01
0.02
0.03
Weight Values


GlorotUniform Initialization
Frequency
0
50
100
150
200
−0.02
−0.01
0.00
0.01
0.02
Weight Values


HeNormal Initialization
Frequency
Weight Values
350
300
250
200
150
100
50
0
−0.03
−0.02
−0.01
0.00
0.01
0.02
0.03


HeUniform Initialization
Frequency
200
150
100
50
0
−0.02
−0.01
0.00
0.01
0.02
Weight Values


LecunNormal Initialization
Frequency
0
50
100
150
200
250
300
350
−0.02
−0.01
0.00
0.01
0.02
Weight Values


LecunUniform Initialization
Frequency
200
150
100
50
0
−0.015
−0.010
−0.005
0.000
0.005
0.010
0.015
Weight Values

Activation	Recommended Initialization
Linear	Glorot
Sigmoid	Glorot
Hyperbolic Tangent	Glorot
Rectified Linear Unit	He
Leaky ReLU	He
Exponential Linear Unit	He
Scaled Exponential Linear Unit	LeCun
Gaussian Error Linear Unit	He
Sigmoid Linear Unit, Swish	He
Mish	He

Implementation

```
model.add(keras.layers.Dense(units, activation="leaky_relu",
      kernel_intializer="he_normal"))
```

Summary

Topics

- Activation Functions
- Initializations

Activation Functions

Function	Keras Name
Linear	`linear`
Sigmoid	`sigmoid`
Hyperbolic Tangent	`tanh`
Rectified Linear Unit	`relu`
Leaky ReLU	`leak_relu`
Exponential Linear Unit	`elu`
Scaled Exponential Linear Unit	`selu`
Gaussian Error Linear Unit	`gelu`
Sigmoid Linear Unit, Swish	`swish,silu`
Mish	`mish`

Initializations

Function	Keras Name
Zeros	`zeros`
Ones	`ones`
Random Normal	`random_normal`
Random Uniform	`random_uniform`
Glorot (Xavier) Normal	`glorot_normal`
Glorot (Xavier) Uniform	`glorot_uniform`
He Normal	`he_normal`
He Uniform	`he_uniform`
Lecun Normal	`lecun_normal`
Lecun Uniform	`lecun_uniform`