LeakyReLU [source]

```
keras.layers.advanced_activations.LeakyReLU(alpha=0.3)
```

Special version of a Rectified Linear Unit that allows a small gradient when the unit is not active: `f(x) = alpha * x for x < 0`, `f(x) = x for x >= 0`.

Input shape

Arbitrary. Use the keyword argument `input_shape` (tuple of integers, does not include the samples axis) when using this layer as the first layer in a model.

Output shape

Same shape as the input.

Arguments

- **alpha**: float >= 0. Negative slope coefficient.

PReLU [source]

```
keras.layers.advanced_activations.PReLU(init='zero', weights=None)
```

Parametric Rectified Linear Unit: `f(x) = alphas * x for x < 0`, `f(x) = x for x >= 0`, where `alphas` is a learned array with the same shape as x.

Input shape

Arbitrary. Use the keyword argument `input_shape` (tuple of integers, does not include the samples axis) when using this layer as the first layer in a model.

Output shape

Same shape as the input.

Arguments

- **init**: initialization function for the weights.
- **weights**: initial weights, as a list of a single Numpy array.

References

- Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification

---

ELU [source]

```
keras.layers.advanced_activations.ELU(alpha=1.0)
```

Exponential Linear Unit: `f(x) =  alpha * (exp(x) - 1.) for x < 0`, `f(x) = x for x >= 0`.

Input shape

Arbitrary. Use the keyword argument `input_shape` (tuple of integers, does not include the samples axis) when using this layer as the first layer in a model.

Output shape

Same shape as the input.

Arguments

- **alpha**: scale for the negative factor.

References

- Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)

ParametricSoftplus [source]

```
keras.layers.advanced_activations.ParametricSoftplus(alpha_init=0.2, beta_init=5.0, weights=Non
```

Parametric Softplus: `alpha * log(1 + exp(beta * x))`

Input shape

Arbitrary. Use the keyword argument `input_shape` (tuple of integers, does not include the samples axis) when using this layer as the first layer in a model.

Output shape

Same shape as the input.

Arguments

- **alpha_init**: float. Initial value of the alpha weights.
- **beta_init**: float. Initial values of the beta weights.
- **weights**: initial weights, as a list of 2 numpy arrays.

References

- Inferring Nonlinear Neuronal Computation Based on Physiologically Plausible Inputs

---

ThresholdedReLU [source]

```
keras.layers.advanced_activations.ThresholdedReLU(theta=1.0)
```

Thresholded Rectified Linear Unit: `f(x) = x for x > theta` `f(x) = 0 otherwise`.

Input shape

Arbitrary. Use the keyword argument `input_shape` (tuple of integers, does not include the samples axis) when using this layer as the first layer in a model.

Output shape

Same shape as the input.

Arguments

- **theta**: float >= 0. Threshold location of activation.

References

- Zero-Bias Autoencoders and the Benefits of Co-Adapting Features

---

SReLU [source]

```
keras.layers.advanced_activations.SReLU(t_left_init='zero', a_left_init='glorot_uniform', t_rig
```

S-shaped Rectified Linear Unit.

Input shape

Arbitrary. Use the keyword argument `input_shape` (tuple of integers, does not include the samples axis) when using this layer as the first layer in a model.

Output shape

Same shape as the input.

Arguments

- **t_left_init**: initialization function for the left part intercept
- **a_left_init**: initialization function for the left part slope
- **t_right_init**: initialization function for the right part intercept
- **a_right_init**: initialization function for the right part slope

References

- Deep Learning with S-shaped Rectified Linear Activation Units