Neural Network:

torch.nn → contains everything that is required to make a neural network namely https://pytorch.org/docs/stable/nn.html#

class torch.nn.Module → base class for all neural network modules. The models that we will be building should be a subclass to this class. Modules can also be nested within another module.

Convolution layers →

Pooling layers →

Padding layers →

Non-linear Activations →

Normalization layers →

Recurrent Layers →

Linear Layers →

Dropout Layer →

Sparse Layers →

Vision Layers →

torch.nn.function → contains all the functional interfaces required to build a neural network. https://pytorch.org/docs/stable/nn.html#torch-nn-functional

Convolution functions →

Pooling functions →

Non-linear activation functions →

Normalization functions →

Linear functions →

Dropout functions →

Sparse functions →

Distance functions →

Loss functions →

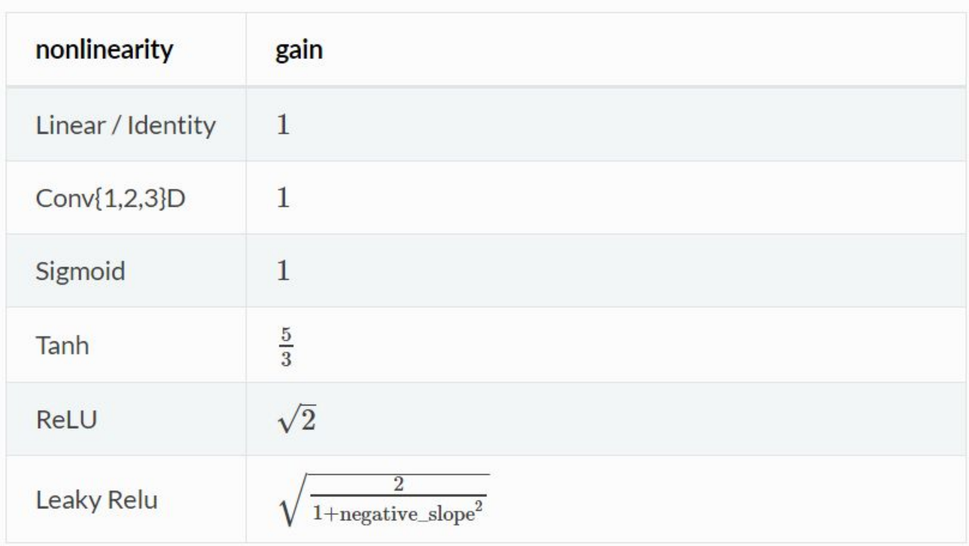
Vision functions →

A neural network without an activation function is essentially just a linear regression model. The activation function does the non-linear transformation to the input making it capable to learn and perform more complex tasks. a(1) is the vectorized form of any linear function.

Torch.nn.init → Return the recommended gain value for the given nonlinearity function. The syntax is -

torch.nn.init.calculate_gain(nonlinearity, param=None)

The following recommended values of the given non-linear functions are returned -



nonlinearity	gain
Linear / Identity	1
Conv{1,2,3}D	1
Sigmoid	1
Tanh	$\frac{5}{3}$
ReLU	$\sqrt{2}$
Leaky Relu	$\sqrt{\frac{2}{1+\text{negative\_slope}^2}}$

Image source: PyTorch Documentation

To know more what torch.nn.init has to offer, refer the documentation page here - https://pytorch.org/docs/stable/nn.html#torch-nn-init