Dense(int(trainX.shape[1]/2), activation=ACTIVATION, input\_dim=trainX.shape[1])

keras.layers.Dense(units,

activation=**None**,

use\_bias=**True**,

kernel\_initializer='glorot\_uniform',

bias\_initializer='zeros',

kernel\_regularizer=**None**,

bias\_regularizer=**None**,

activity\_regularizer=**None**,

kernel\_constraint=**None**,

bias\_constraint=**None**)

Source: <https://stackoverflow.com/questions/56299770/units-in-dense-layer-in-keras>

Neural Networks are basically matrix multiplications, the drop you are talking about in the first part is not due to an Activation function, it's only happen because of the nature of matrix multiplication :

The calculation here is : input \* weights = output

so -> [BATCHSIZE, 784] \* [784, 32] = [BATCHSIZE, 32] -> output dimension

With that logic we can easily explain how we can have an input shape << size of units, it will give this calcul :

-> [BATCHSIZE, 20] \* [20, 64] = [BATCHSIZE, 64] -> output dimension

Hope that helped you !