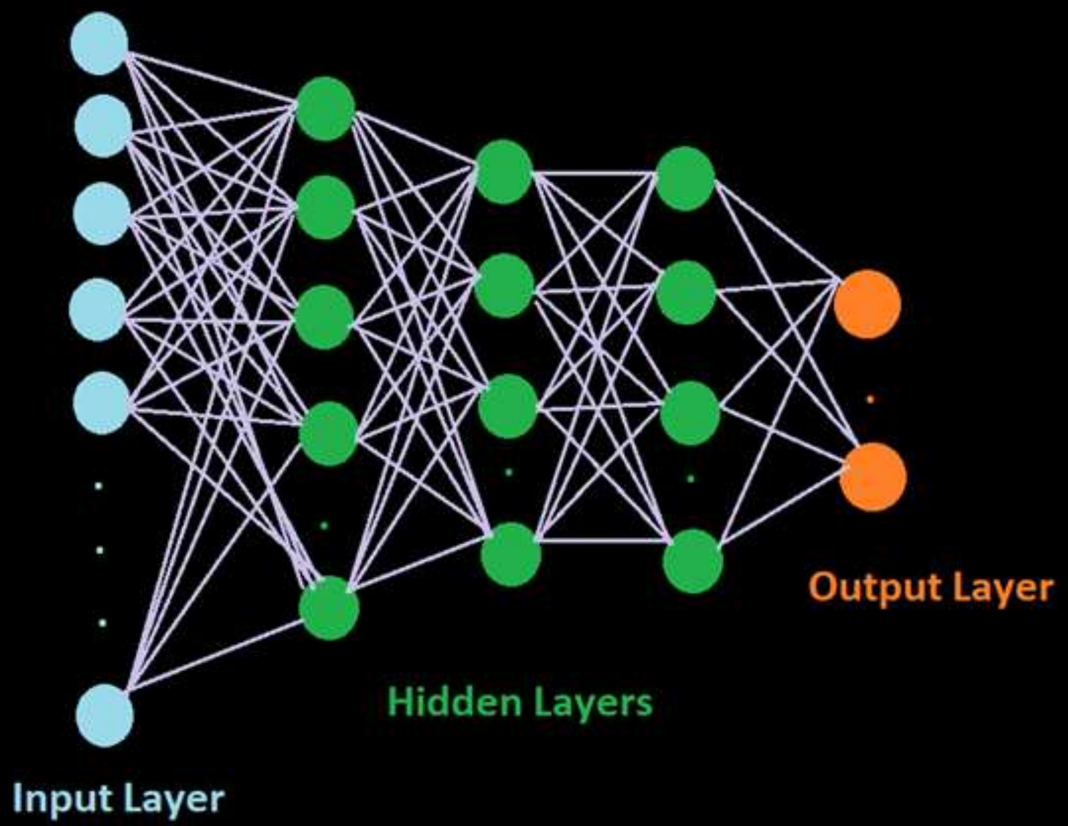
Dropout Regularization
and
Batch Normalization

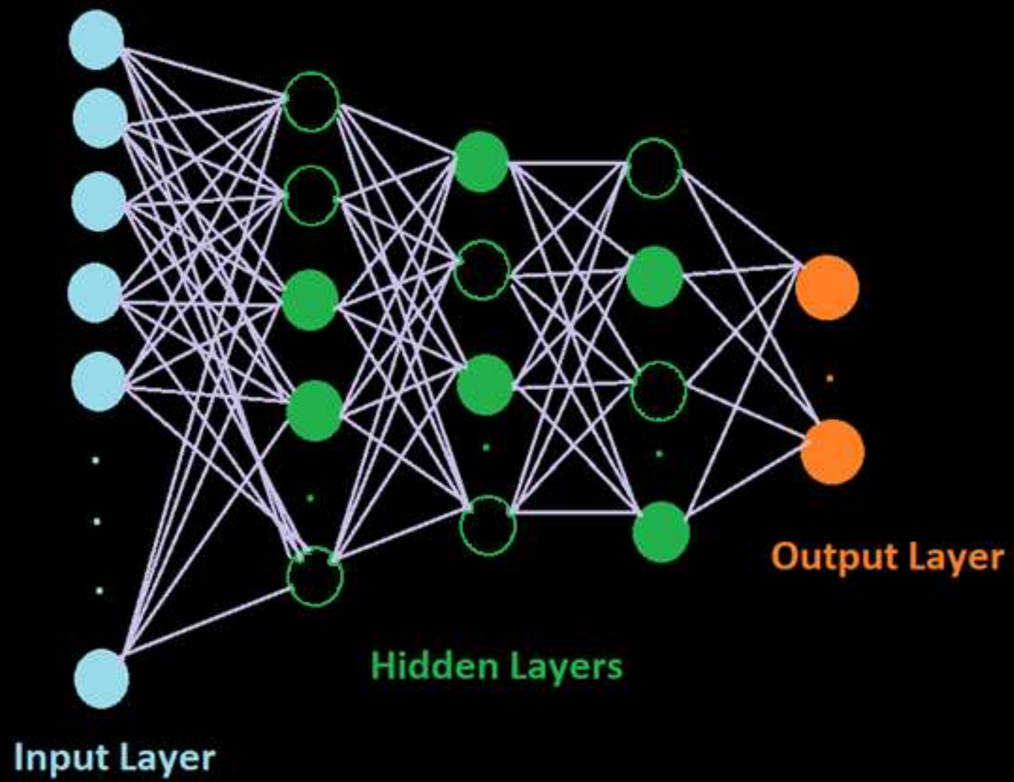
Dropout Regularization

Model Before Dropout

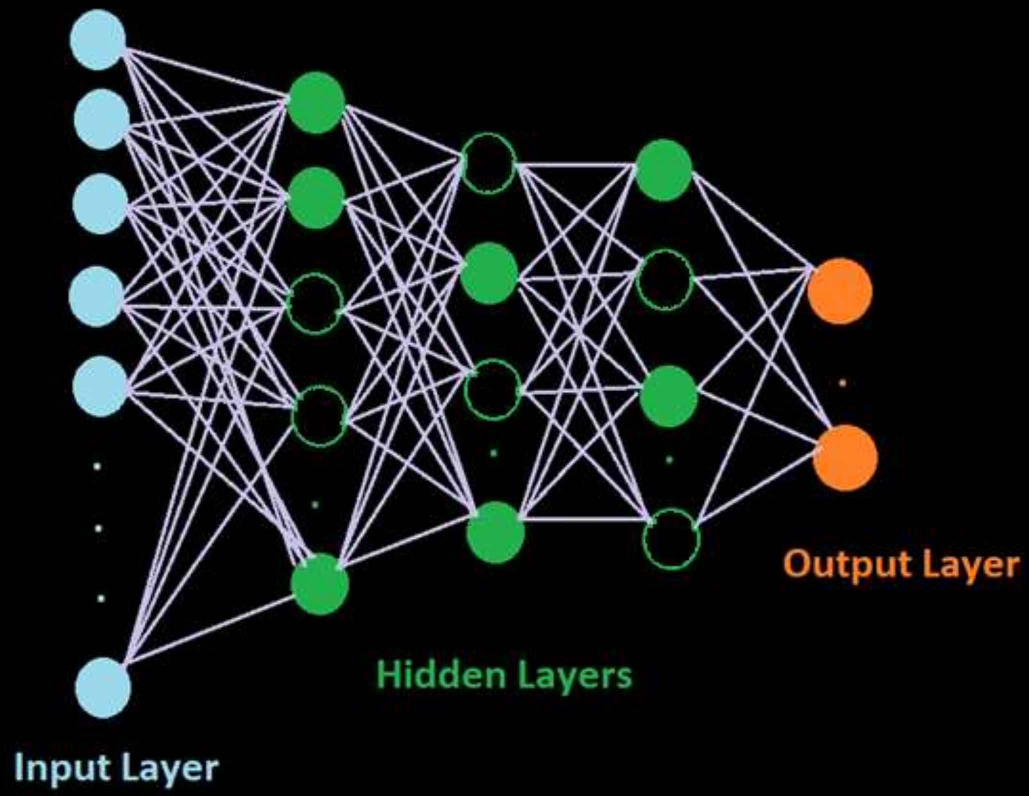


P = 0.5 (Dropout probability)

Model After Dropout (epoch = 1)



Model After Dropout (epoch = 2)

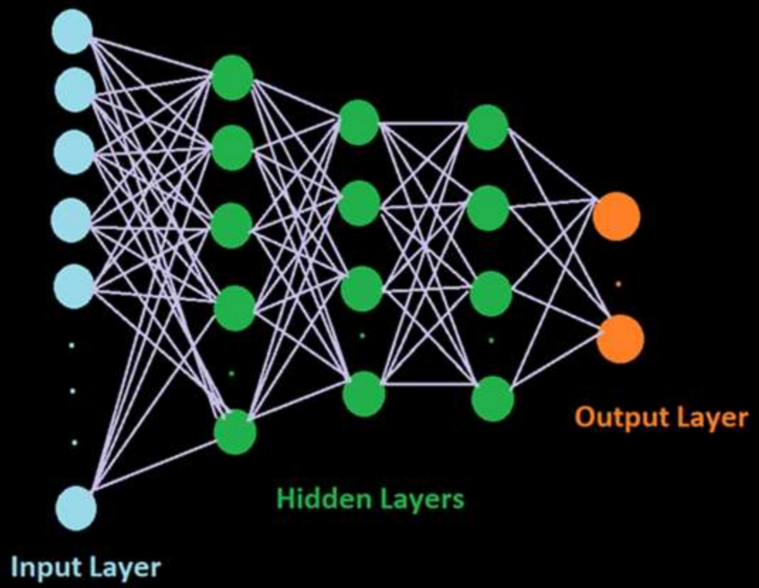


Important Points

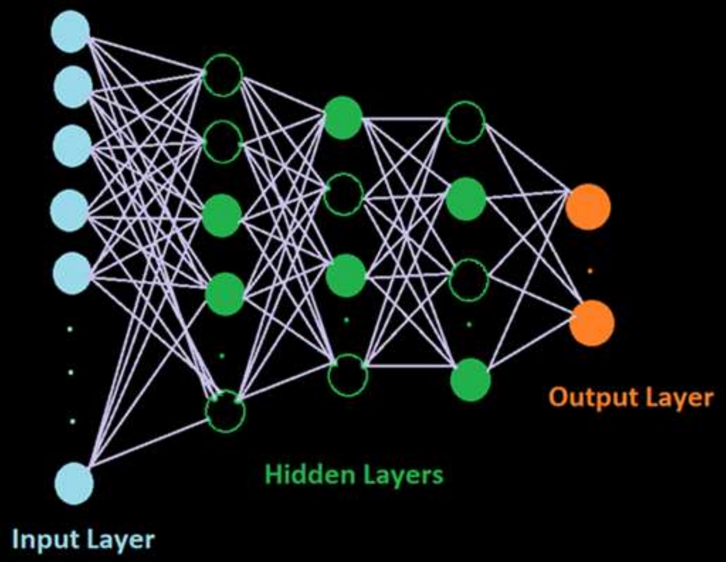
- The unit are not physically out of the model but their activations are forced to be zero.
- In every epoch, the probability of drop out is same but units are selected randomly and it continues till the end.
- No dropout during testing.

Problem

We do not want dropout during testing. So, the problem is input to the node is higher during the testing than the training.



Testing

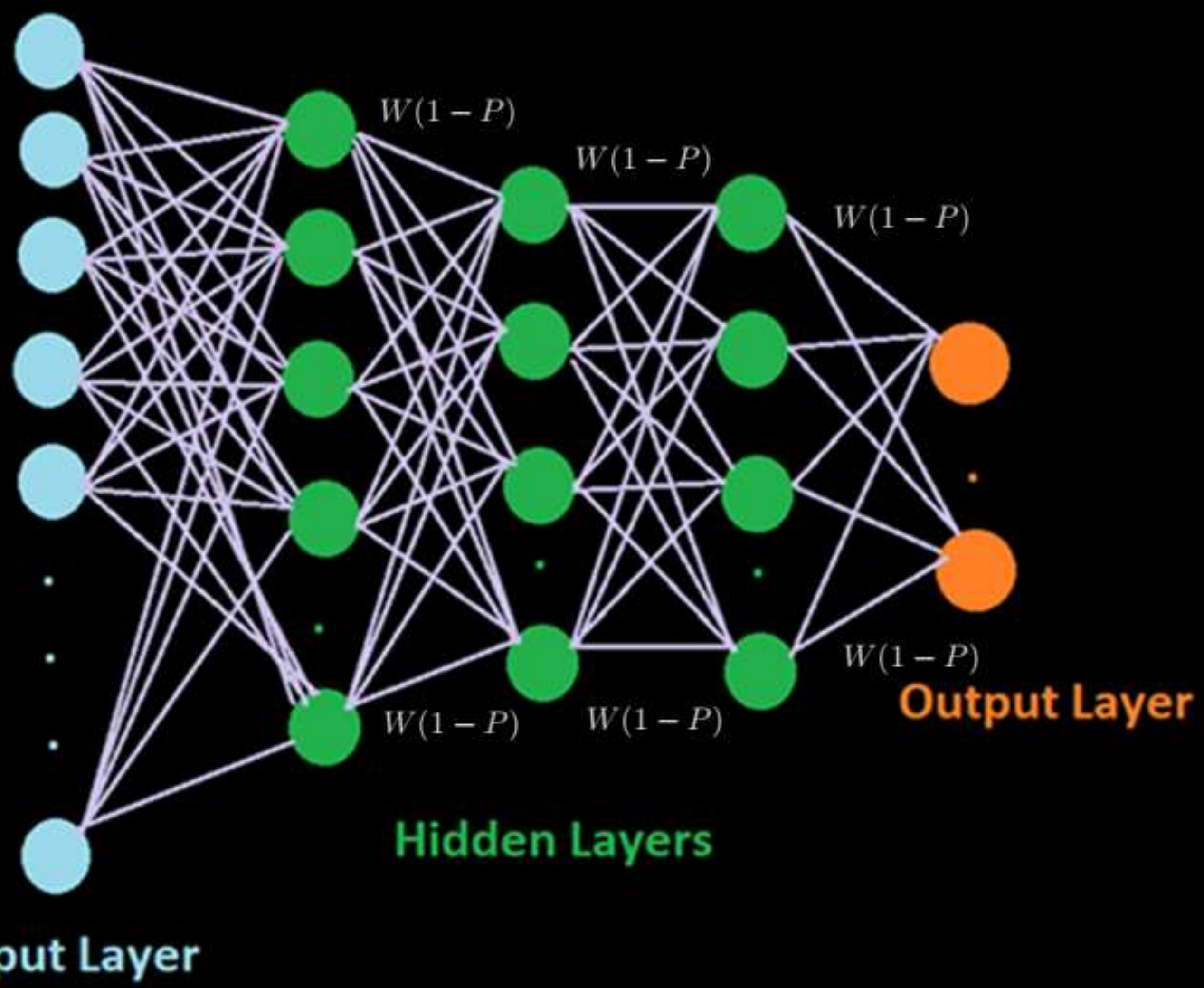


Training

Solutions of the Problem

- Scale down the weights during testing.
- Scale up the weights during training.

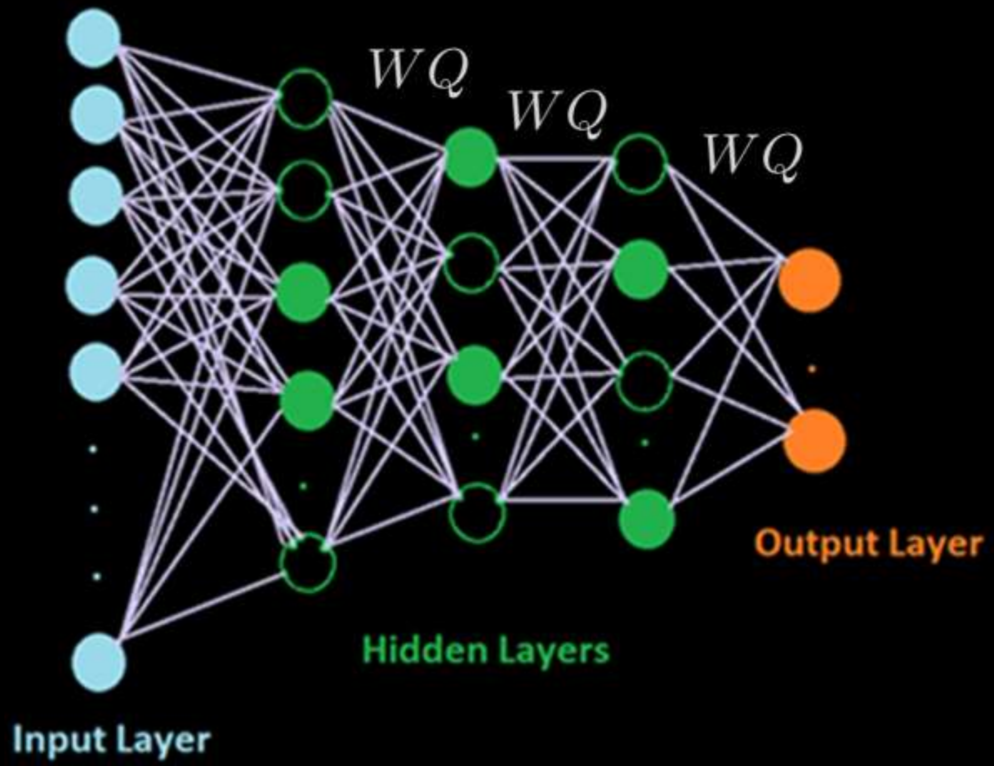
Scaling down the weights during testing



$\boxed{(1-P) = \text{Probability of Keeping the node}}$

Testing

Scaling up the weights during training



Training

$$\boxed{Q = \frac{1}{1-P}}$$

Observations About Dropout Regularization

- It prevents the units of NN to learn more than necessary.
- Works better on deep than shallow Networks.
- Work better with Large data.
- Results in more smooth training.
- Makes the model less dependent on certain units.
- Adding dropout may decrease the training accuracy but increase the generalization capacity of the model.

Tips About Dropout Regularization

- A good starting point is to start with 20% and gradually increase if the model is not learning.
- It is more likely to observe better results on a deeper Neural Networks than shallow ones
- Dropout can be used to input layers as well as hidden layers.

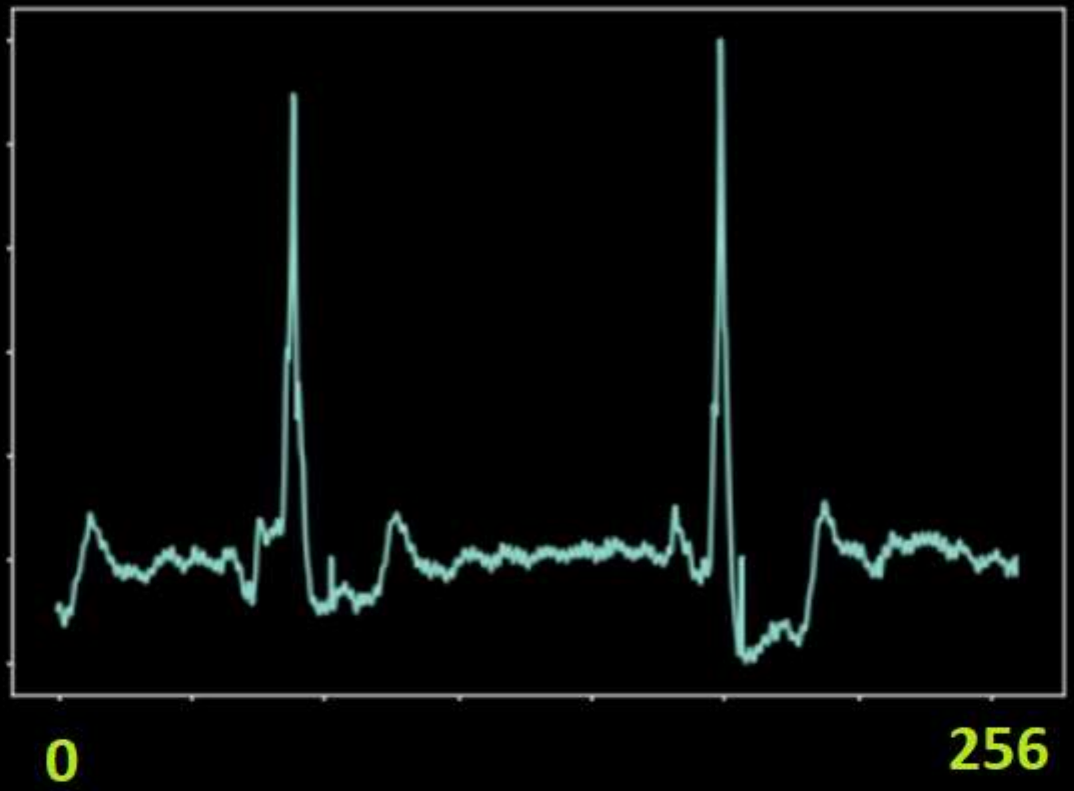
Introducing WESAD, a Multimodal Dataset for Wearable Stress and Affect Detection

Philip Schmidt*
Robert Bosch GmbH
Corporate Research, Germany
firstname.lastname@de.bosch.com

Attila Reiss, Robert Dürichen, Claus Marberger
Robert Bosch GmbH
Corporate Research, Germany

Kristof Van Laerhoven
University of Siegen
Siegen, Germany
kvl@eti.uni-siegen.de

- user2
- user3
- user4
- user5
- user6
- user7
- user8
- user9
- user10
- user11
- user13
- user14
- user15
- user16
- user17



Features = 256

ECG Data

Three Stress Classes in WESAD Dataset

0 : First Level of stress

1 : Second Level of stress

2 : Third Level of stress

Batch Normalization

Why Batch Normalization

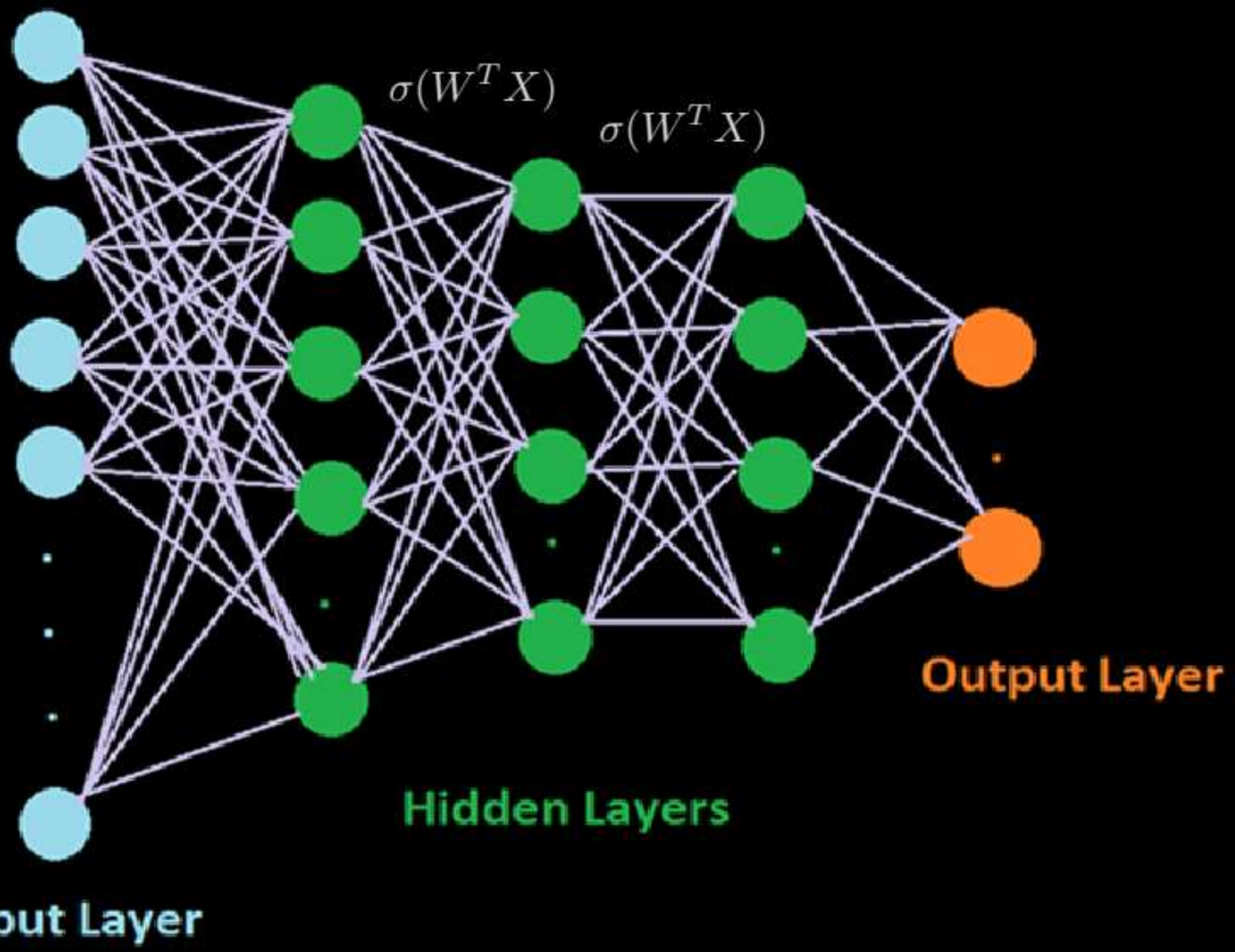
- After data Normalization, we pass the data through the model. But it is not likely that this data remains normal while travelling through the nodes of the neural network.
- While moving from layer to layer, it gets scaled by taking the weighted combination with the weight matrix.

Why Batch Normalization

- Activation functions are non-linear. For example, ReLU function clips off the negative values. Similary activation functions also do the scaling and shifting according to their variance and mean value of distribution. To solve this problem, we use batch normalization.

$$\boxed{y = \sigma(W^T X)}$$

Batch Normalization



Batch Normalization

- Batch normalization normalizes each feature independently across the batch. Since batch normalization is dependent on batch size, therefore, it is not much effective for small batches.
- We want the data to be normalized for each layer.
- For input layer of Neural Network, we normalize the data first and then pass it to the input layer, however for hidden layers we also need to normalize the data, therefore, we do batch normalizations.

Batch Normalization

- Batch Normalization is only applied during the training because during test sample we may have very small batch size such as 1 if we test one sample. Also with one sample, there is no variance.
- When we set model.eval(), then Pytorch switch off batch normalization.
- Batch normalization is a form of Regularization because it prevents over-fitting.

Thank you!

Thank you!