Batch Normalization

Normalization:
Here you scale your data such it has **mean=0 and standard deviation=1**

So, the age and income values would be on same scale. This way no one feature would dominate other.

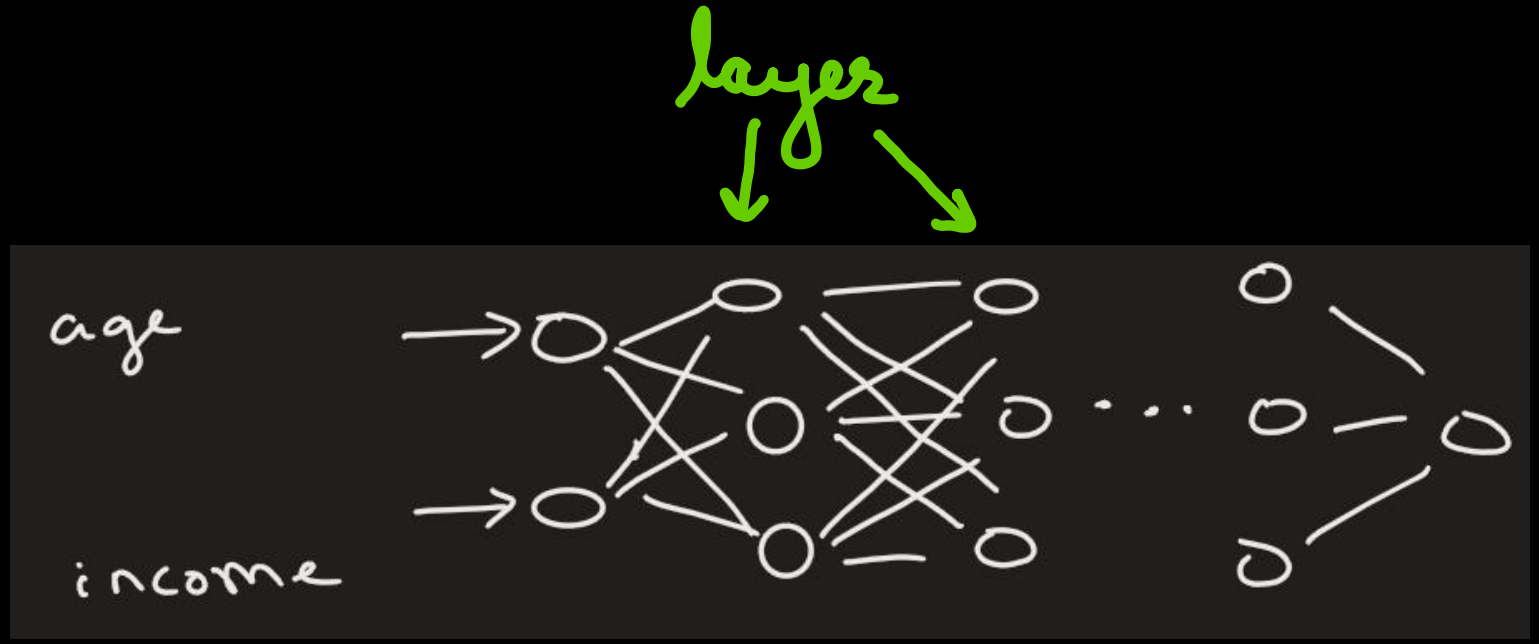
age	Income
39	60000
42	94000
27	120000
33	90000
…	….

After Normalization

age	Income
-1.2	1.4
1.4	0.7
0.4	2.3
0.9	1.8
…	…

Internal Covariate Shift

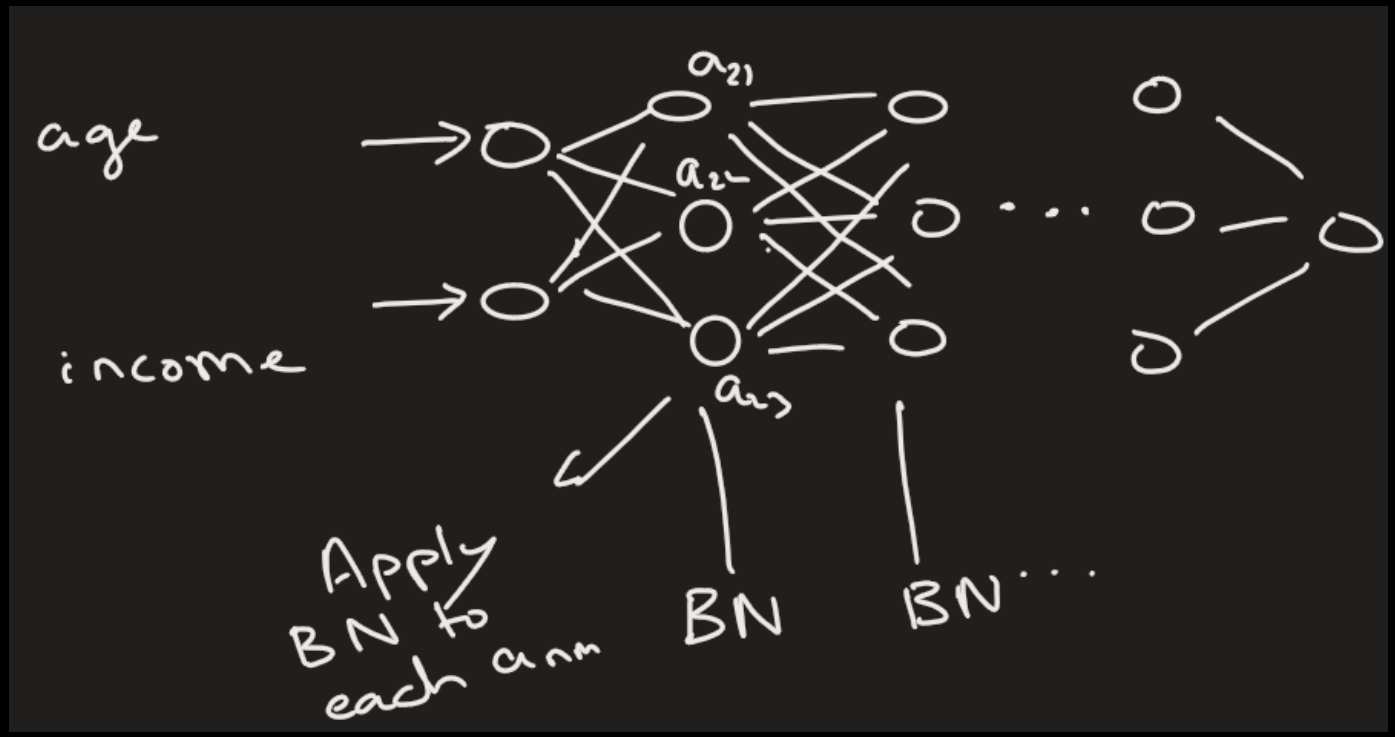
Suppose you feed your normalized data to 1st input layer of deep neural networks.
When training DNN:
Each layer's output becomes the next layer's input.
As weights update in each layer, the distribution of inputs to deeper layers keeps changing.
This makes training **slow and unstable.** This problem is called **Internal Covariate Shift**



Batch Normalization

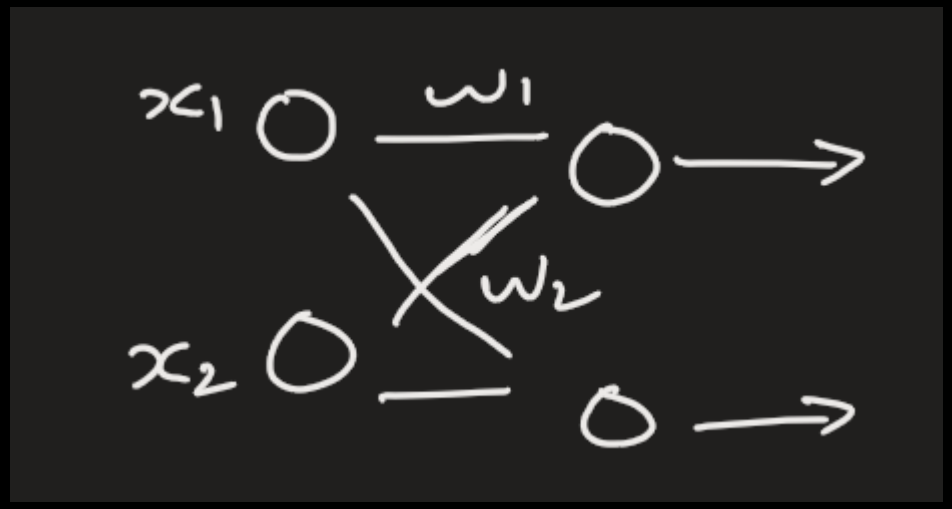
Batch Normalization is a technique used in **neural networks** to
- **normalize the activations of a layer** so that they have a **stable distribution** during training.

So, each activation that becomes input to next layer is also normalized.



2 ways to do batch normalization

1) Calculate z first for each neuron, then normalize the z value, and then calculate the activation a for this normalized z



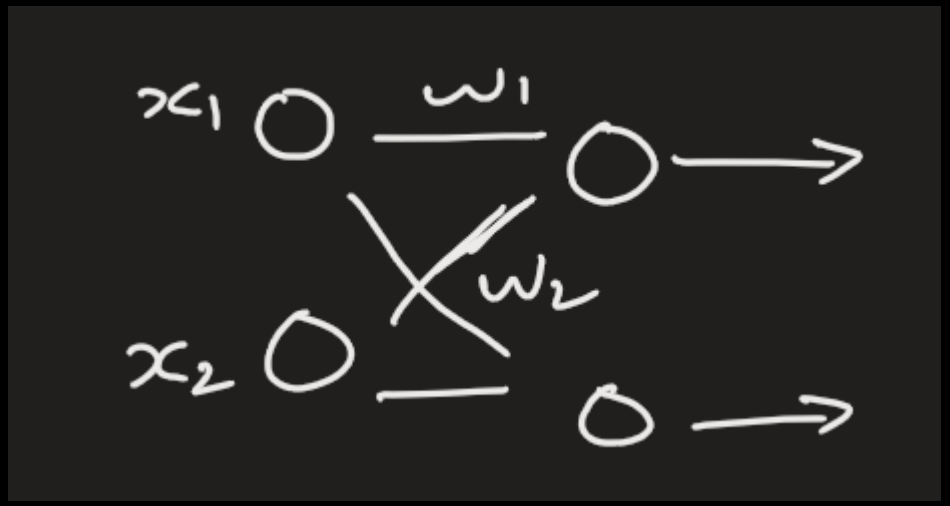
1) $z_{11} = x_1 w_1 + x_2 w_2 + b$

Normalize z_{11} → apply $g(\,)$

$z_{11} \rightarrow z_{11}^{N} \xrightarrow[g(\,)]{\text{apply}} g(z_{11}^{N}) = a_{11}$

2 ways to do batch normalization

2) Calculate z first for each neuron, then calculate the activation a for this z , and then normalize the activation a.



2) $z_{11} = x_1 w_1 + x_2 w_2 + b$

Apply $g()$ → Normalize a

$z_{11} \rightarrow g(z_{11}) = a_{11} \rightarrow a_{11}^{N}$

Question: How do you normalize z ?

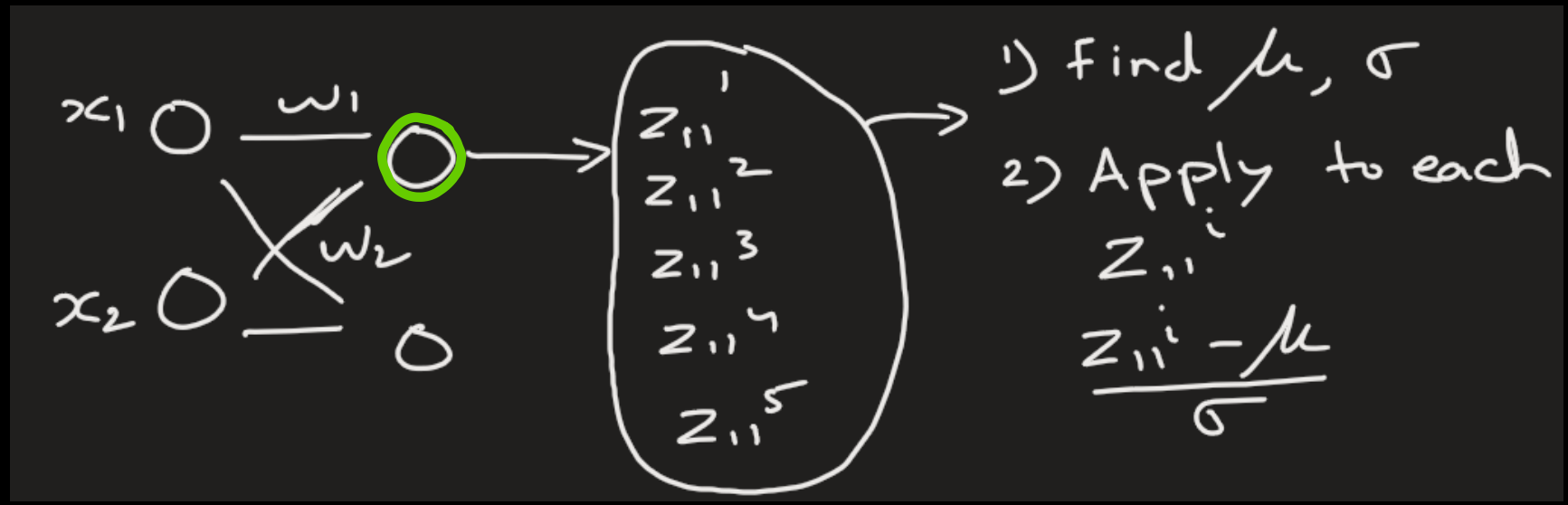
Let's focus on first top neuron.

Step 1)

Since we are working with batches (i.e. batch normalization), so we feed a batch of, say 5, input values and calculate the z for all 5 values.

Then we normalize these 5 values like we do any dataset.

age	income	
-1	2	batch of 5
2.1	0	
0.4	1.1	
-0.5	0.5	
2.1	0.9	
-1	1.2	
⋮	⋮	



Step 2)

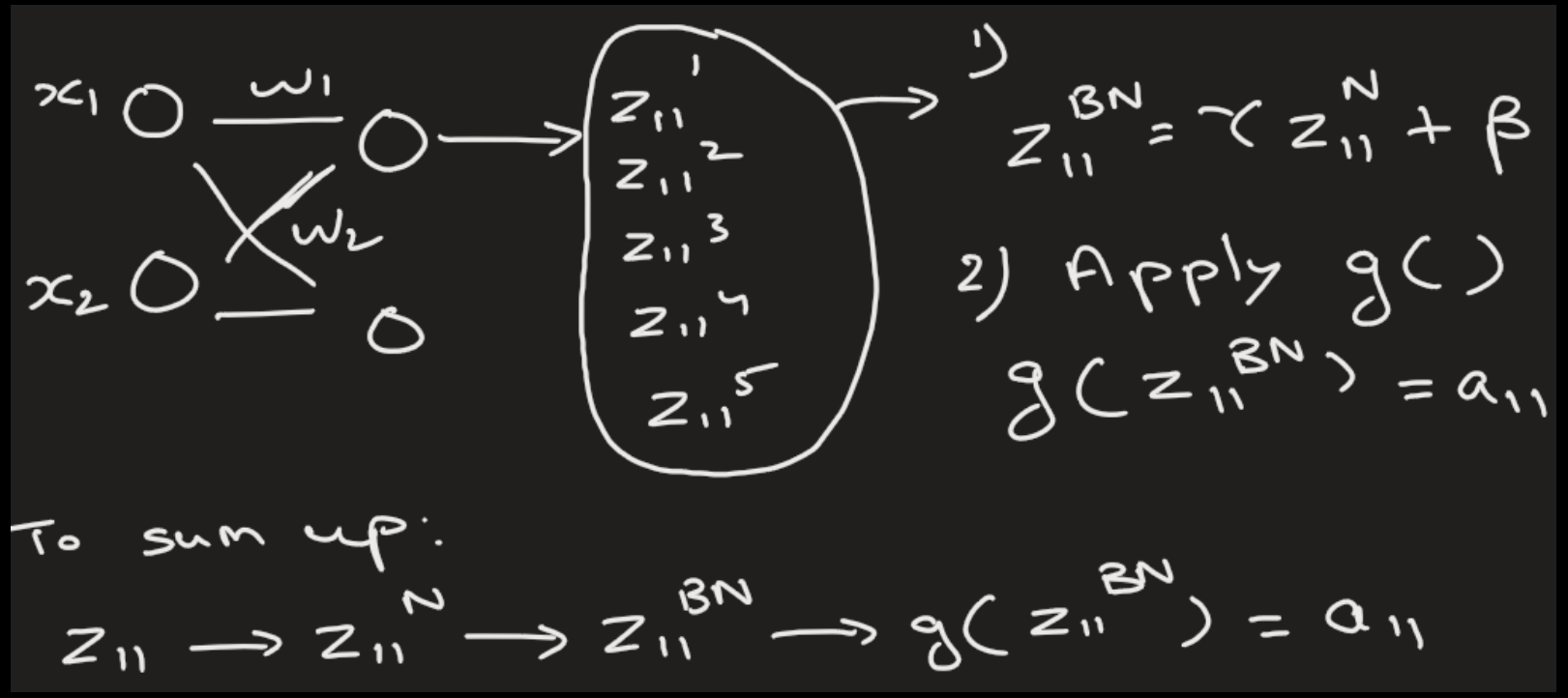
Then we multiply this new normalized z with learning parameter γ and add β

These 2 are calculated during training phase.

γ is also called scale parameter and β is also called shift parameter.

age	income
-1	2
2.1	0
0.4	1.1
-0.5	0.5
2.1	0.9
-1	1.2
:	:

batch of 5

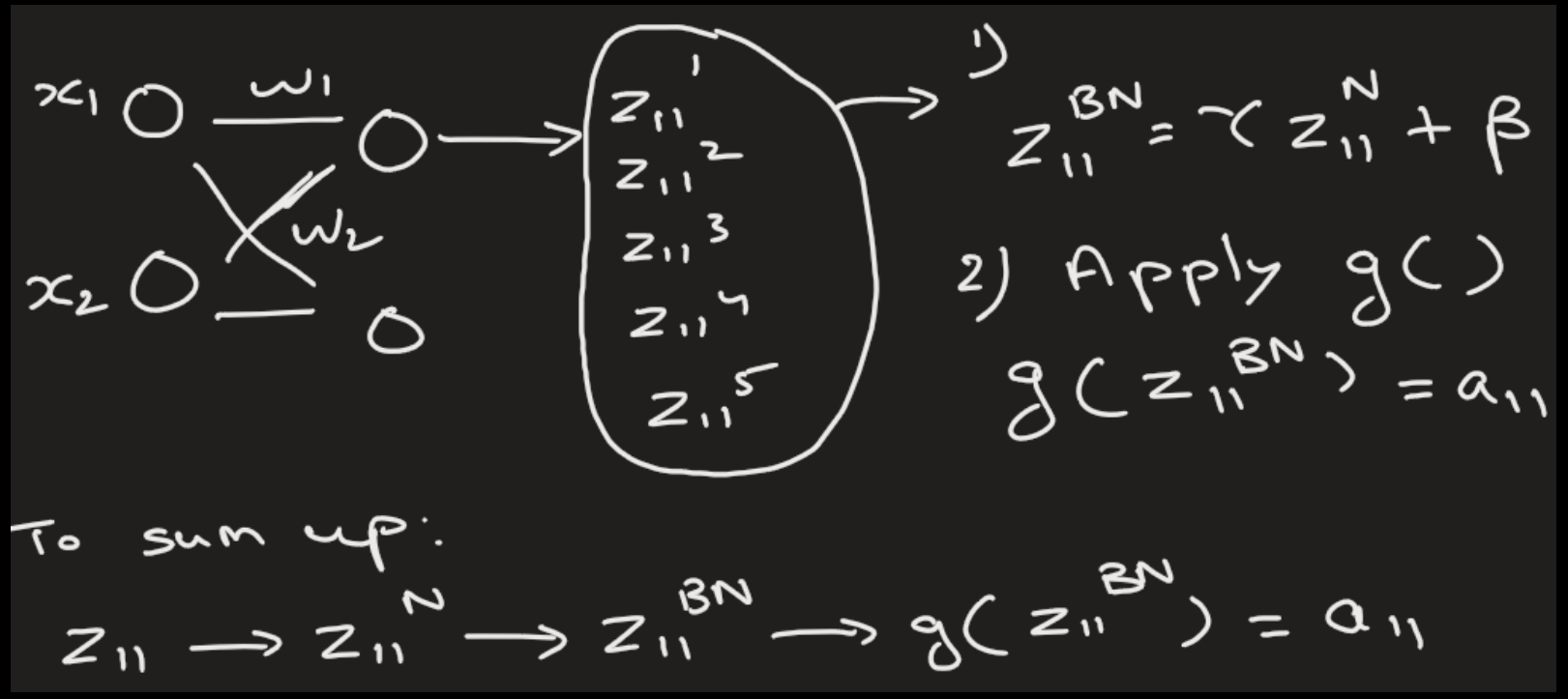


Batch Normalization

Note:

- We do these steps for all neurons.
- Each neuron has its own learning parameter, γ and β.

age	income	
-1	2	batch of 5
2.1	0	
0.4	1.1	
-0.5	0.5	
2.1	0.9	
-1	1.2	
:	:	



Question: How do you update parameter γ and β?

We use gradient descent.

$$\gamma = \gamma - \eta \frac{\partial C}{\partial \gamma}$$

C: Cost function

η: learning rate

$$\beta = \beta - \eta \frac{\partial C}{\partial \beta}$$

BN advantage

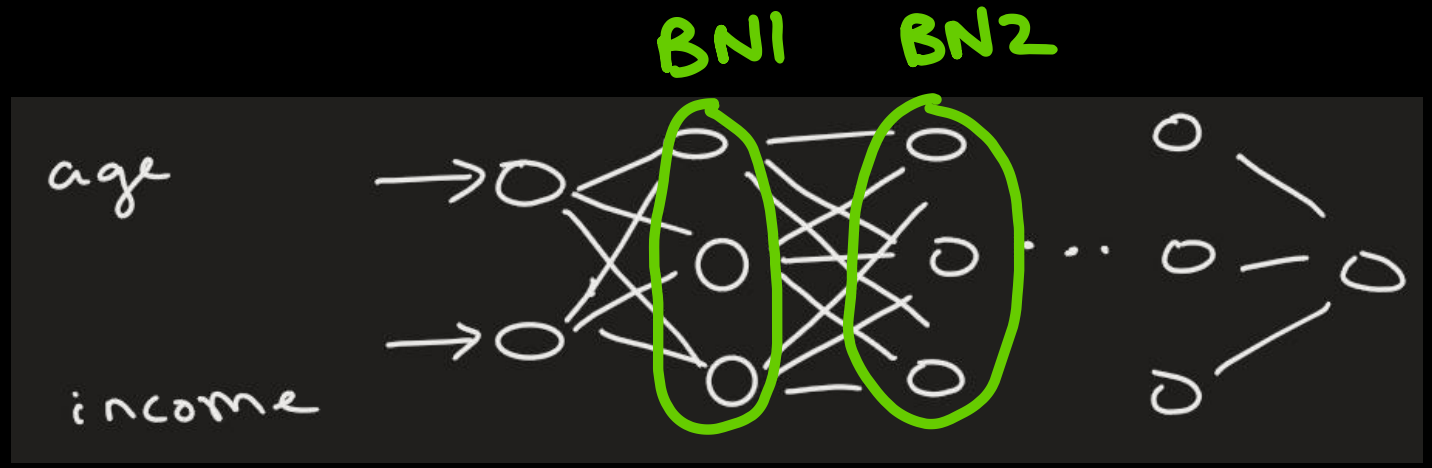
BN does not necessarily increase accuracy, but it helps in following:

1) Faster Training: Allows **higher learning rates** and **converges faster.**

2) Reduces Internal Covariate Shift: Prevents input distributions to deeper layers from changing drastically as previous layers' weights update.

3) Less Sensitive to Initialization: Reduces dependency on careful initial weight settings.

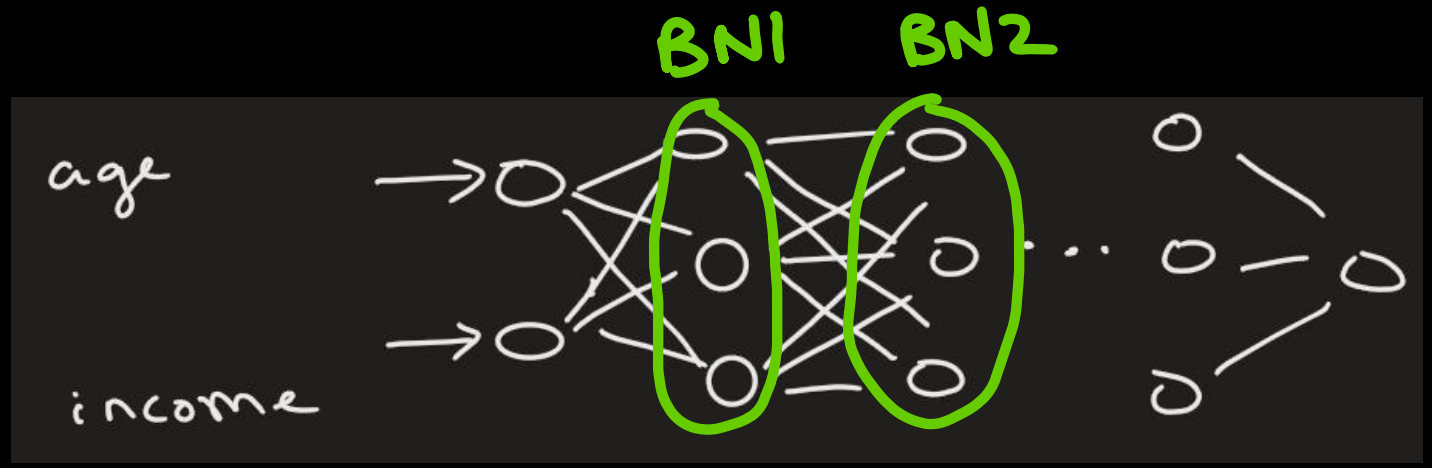
4) Helps avoid vanishing or exploding gradients.



Limitations of BN

- **Small Batches:** Performance can degrade with very small mini-batches because batch statistics become less reliable. For example, mean is more reliable when samples are large.

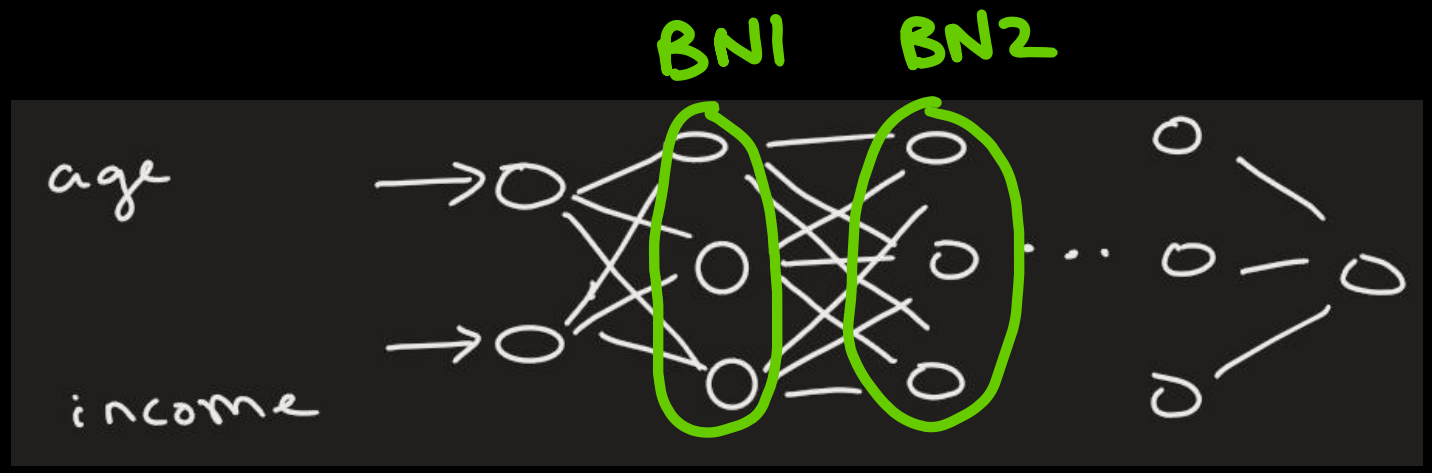
- **RNNs:** Difficult to apply directly to Recurrent Neural Networks (RNNs) due to variable sequence lengths, though variants exist.



“Batch Normalization is not magic. On easy problems with modern optimizers, you may not see dramatic improvements.”

“But on deep or poorly conditioned networks, BatchNorm becomes essential.”

“If you don’t see BatchNorm helping, your problem is probably too easy.”



```python
model_no_bn = Sequential()

# Flatten 28x28 image to 1D
model_no_bn.add(Flatten(input_shape=(28, 28)))

# Hidden layers
model_no_bn.add(Dense(128))
model_no_bn.add(ReLU())

model_no_bn.add(Dense(64))
model_no_bn.add(ReLU())

# Output layer (10 classes)
model_no_bn.add(Dense(10, activation='softmax'))
```

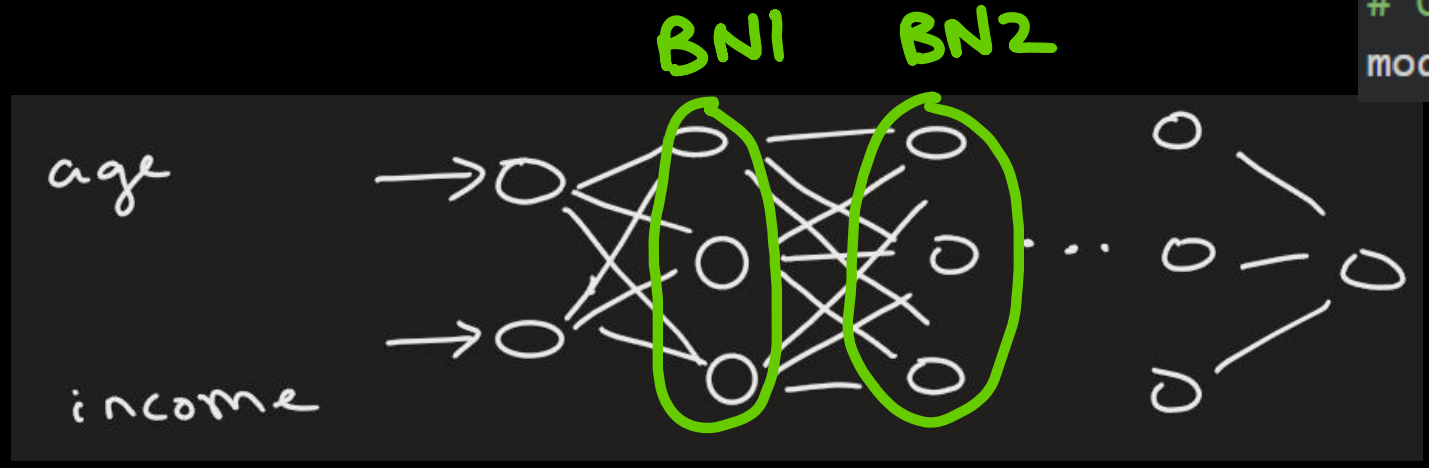
```python
model_bn = Sequential()

# Flatten layer
model_bn.add(Flatten(input_shape=(28, 28)))

# Hidden layer 1
model_bn.add(Dense(128))
model_bn.add(BatchNormalization())
model_bn.add(ReLU())

# Hidden layer 2
model_bn.add(Dense(64))
model_bn.add(BatchNormalization())
model_bn.add(ReLU())

# Output layer
model_bn.add(Dense(10, activation='softmax'))
```



Heading Goes Here

Fhdsklf
Fjdsklf
Fjskldf