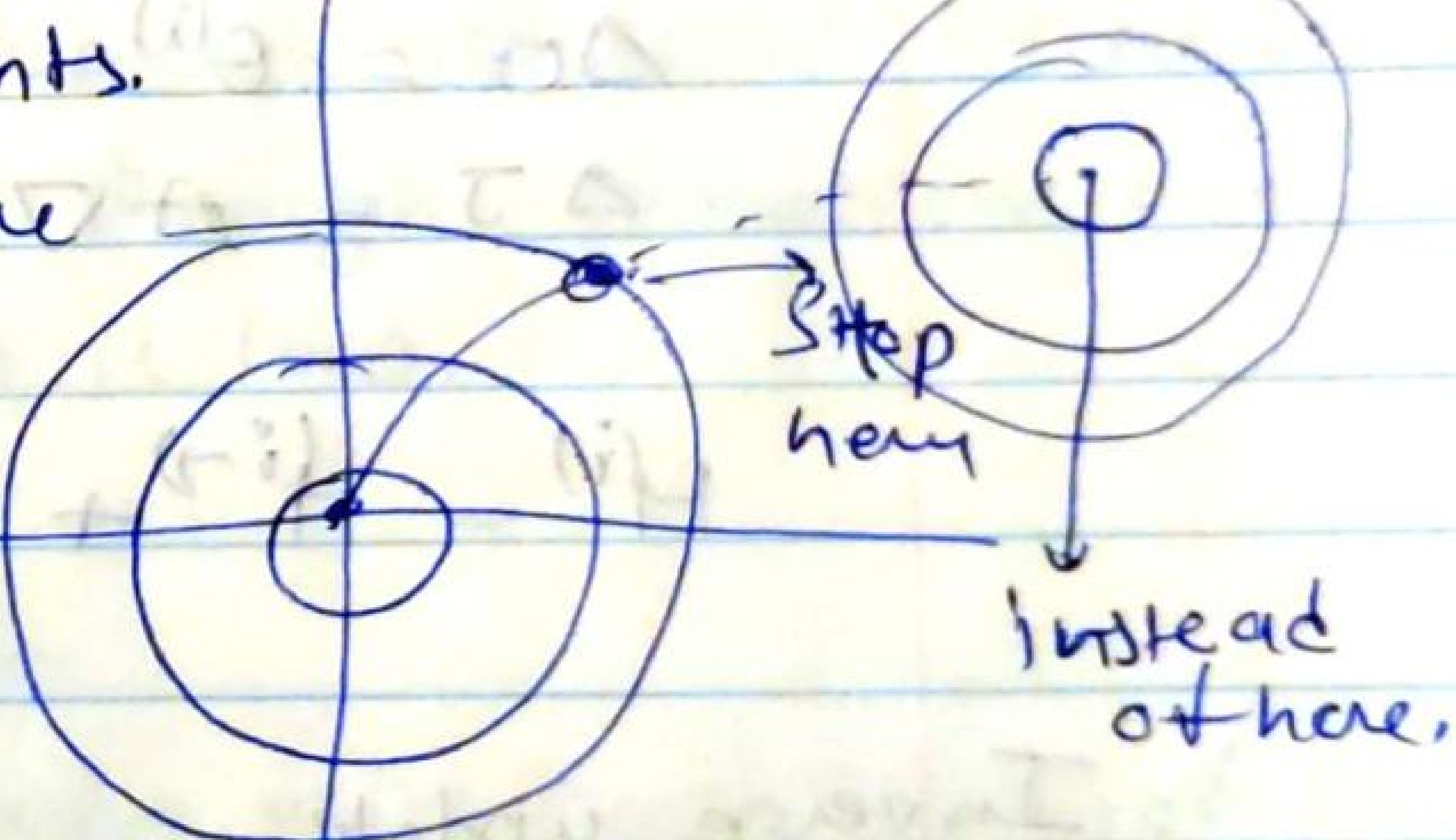
Regularization :→

Q ① Weight Decay: In while traing neural networks it is common to use "weight decay" where after each update, the weight are multiplied by a factor $P \in [0,1]$. As itration progres weights that are not reinforced decay to 0. This prevent the weight from growing too large and can be seen as adding regularization term to loss fuction.

Q2 Early Stopping ⇒ means to stop training the network when ~~vadt~~ validation error increase instead of when traing error stops decreasing. It can be interpreted as L_2 regularization limit the error to smaller neighbourhood wriets is live penalty on larger weights.

* for early stopping, we need the validation data to understand where to stop



* Strategy 1

Retain on all the data using the no of ~~increa~~ iterations determind from the validation. The iterations where validation loss stop decreasing on validation data.

Algorithm ⇒

① Let x^{train} and y^{train} be the training set.

② Split x^{train} and y^{train} into $(x^{sub\,train}, x^{valid})$ and $(y^{sub\text{-}train}, y^{valid})$

③ Run early stopping starting from random 'θ' using $x^{(subtrain)}$ and $y^{(subtrain)}$ for training data and x^{valid} and y^{valid} for validation data. This returns i^* the optimal no. of steps.

④ Set θ' to the random values again.

⑤ Train on x^{train} and y^{train} ~~for~~ i^* steps.

* Strategy 2:

Continue training from previous weights with entire data while validation loss is bigger than training loss.

① Let x^{train} and y^{train} be the training set

② ~~Split~~ split x^{train} and y^{train} into ($x^{subtrain}$, x^{valid}) and ($y^{subtrain}$, y^{valid})

③ Run early stopping algo. starting from random 'θ' using $x^{(subtrain)}$ and $y^{subtrain}$ for training data and x^{valid} and y^{valid} for validation data. This updates θ.

④ $\epsilon \leftarrow J(\theta, x^{subtrain}, y^{subtrain}) \rightarrow$ error

while $J(\theta, x^{valid}, y^{valid}) > \epsilon$ do

train on x^{train} and y^{train} for n steps

end while.

④ Data Augmentation → To prevent overfitting of the data

Synthetic data to increase variability in training better generalization

→ Augmentation can be done in feature space or data

→ Augment by interpolating between example or by adding noise (in data or feature domain).

→ Augment by transforming data by chopping, rotating, scaling the images.

→ popular in image classification do introduction scale/ illumination/rotation invariance.

Q4 Dropout. At every training stage dropout units in fully connected layers with probability of $(1-p)$, where p is hyper parameter.

→ Removed nodes are reinstated with original weights in the subsequent stage.

→ Advantages ⇒ Reduce node interaction (co-adaption) reduce overfitting, increase training speed. Reduce dependency on a single node, distribute feature across multiple nodes.

→ Disadvantage ⇒ longer training due to dropout (not all units are available at each step).

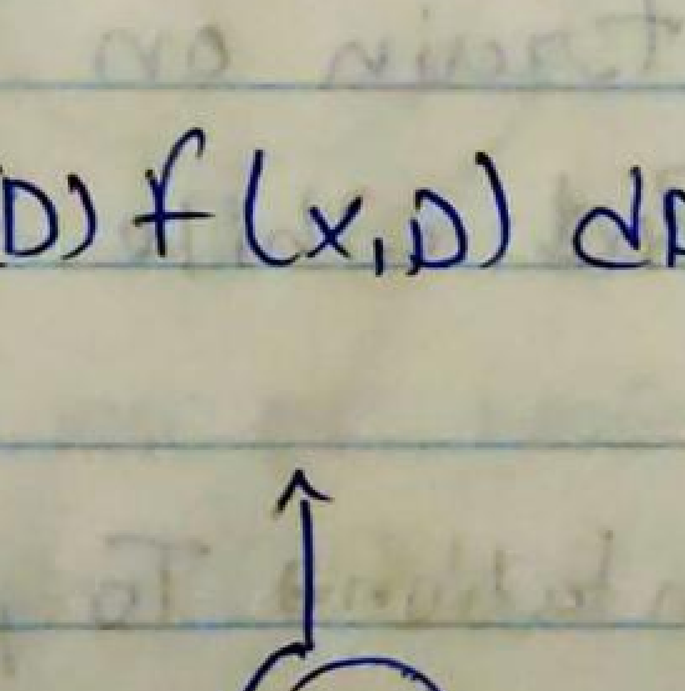
Q5 Multiplying the output of each node by p is equivalent to computing expected value of for 2^n dropped-out networks.

$$\hat{y} = E_D[f(x, D)] = \int p(D) f(x, D)\, dD$$

D: mask for all n nods.

$$\hat{y} = E_D[f(x, D)] = \int p(D) f(x, D)\, dD.$$

$$\hat{y} = \frac{1}{4}\sigma(0x_1 + 0x_2) + \frac{1}{4}\sigma(0x_1 + 1x_2) + \frac{1}{4}\sigma(1x_1 + 0x_2) + \frac{1}{4}\sigma(1x_1 + 1x_2)$$



θ_1	0	0	1	1
θ_2	0	1	0	1

So during the testing phase, all nodes are utilized which leads to higher values than during the training phase with ~~drop~~ dropouts. To approximate this the output in the training phase are multiplied by p or $(1-p)$ → probability of node being dropped.

Q6 Batch Normalization → $\{x^{(i)}\}_{i=1}^{m} \rightarrow \{\hat{x}^{(i)}\}_{i=1}^{m}$

* Input Normalization :

$$\hat{x}_j = \frac{x_j^{(i)} - \mu_j}{\sigma_j} \quad \mu_j \rightarrow \text{mean}, \ \sigma_j \rightarrow SD$$

$$\mu_j = \frac{1}{m} \sum_{i=1}^{m} x_j^{(i)}$$

$$\sigma_j = \left(\frac{1}{m} \sum_{i=1}^{m} (x_j^{(i)} - \mu_j)^2\right)^{1/2}$$

→ why to do this Input Normalization :-

1. Give equal importance to features with different scales.
2. To make sure that activations are not saturated (eg values too large given current weight).
3. avoid all gradients having the same sign due to all positive or all negative inputs.

Batch output $\{z^{(i)}\}_{i=1}^{q} \rightarrow \{\hat{z}^{(i)}\}_{i=1}^{q}$ $\quad j \in [1, n]$, $i \in [1, q]$

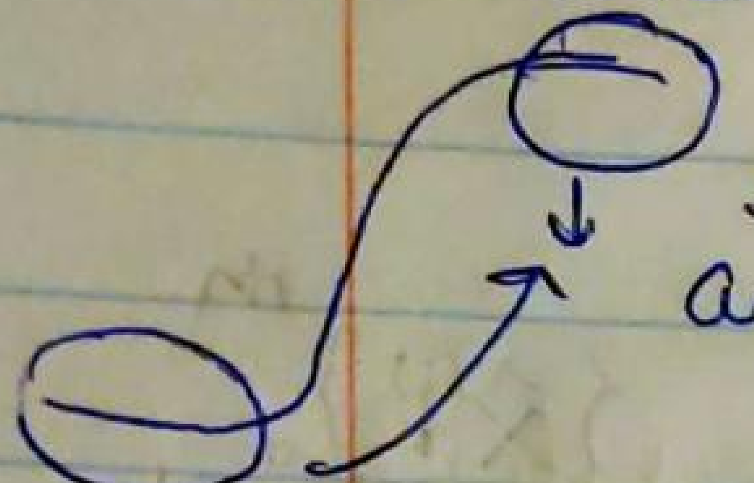
$$\hat{z}_j = \frac{z_j^{(i)} - \mu_j}{\sigma_j} \quad \mu_j = \frac{1}{q} \sum_{i=1}^{q} z_j^{(i)}$$

output of j-th unit for i-th batch example.

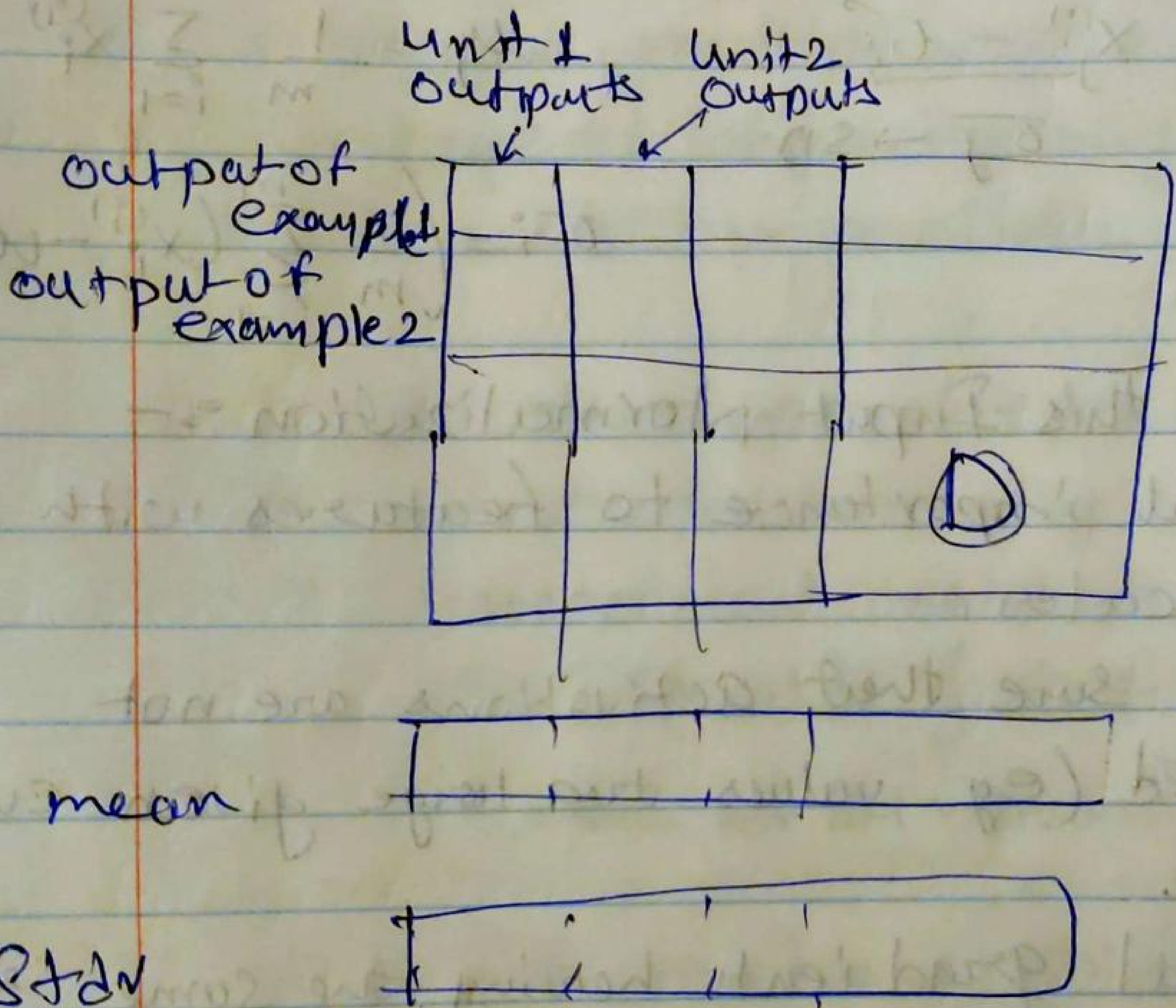
$$\sigma_j = \left(\frac{1}{q} \sum_{i=1}^{m} (z^{(i)} - \mu_j)^2\right)^{1/2}$$

Advanges of Batch Normalizator :-

1. makes sure activations are not saturated
2. Normalization values computed for each batch in training.
3. Normalization is differentiable → suitable for back propogation.



Large output matrix



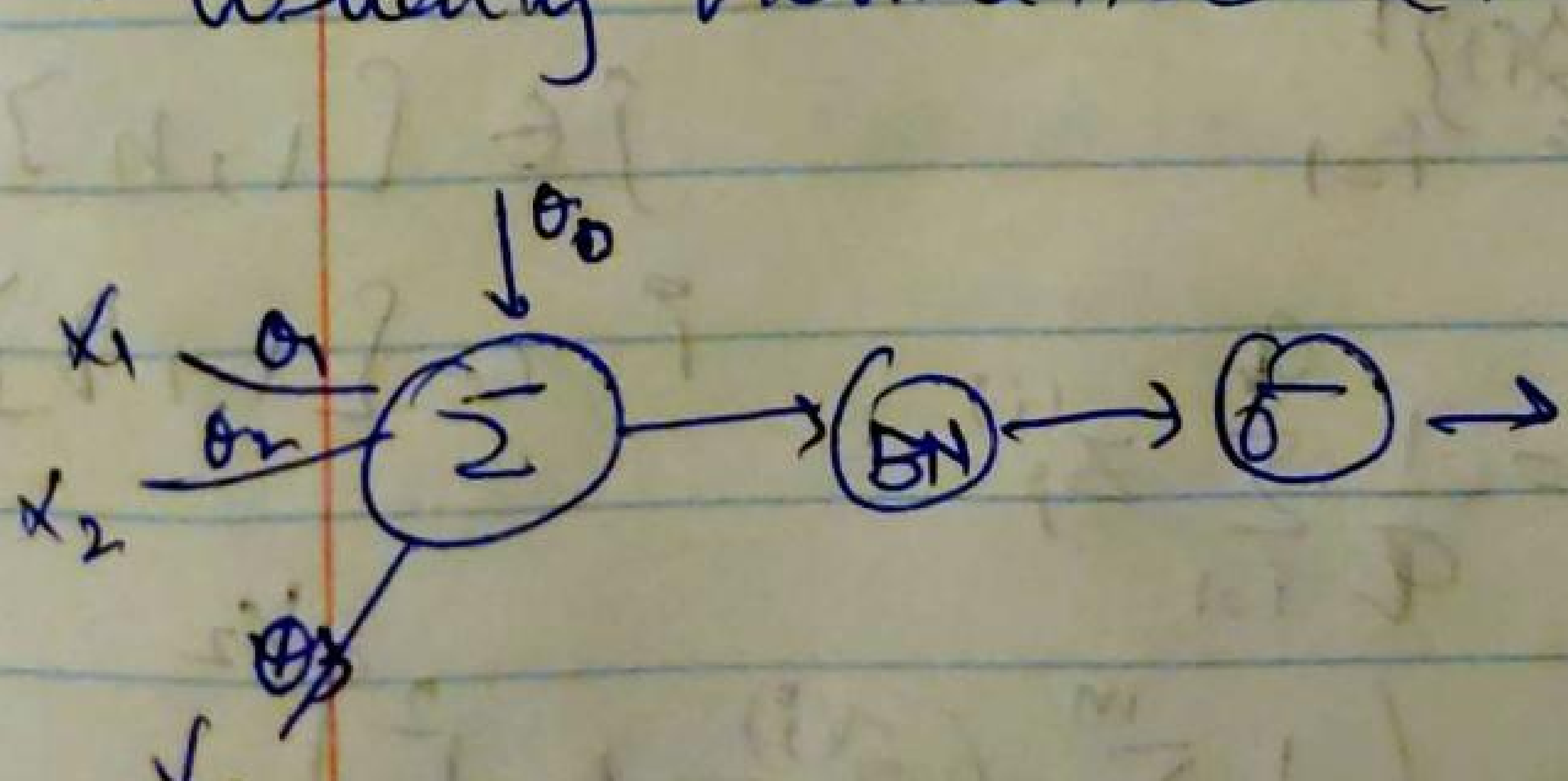
m = mean (axis=0)

D = D − m

S = D.std (axis=0)

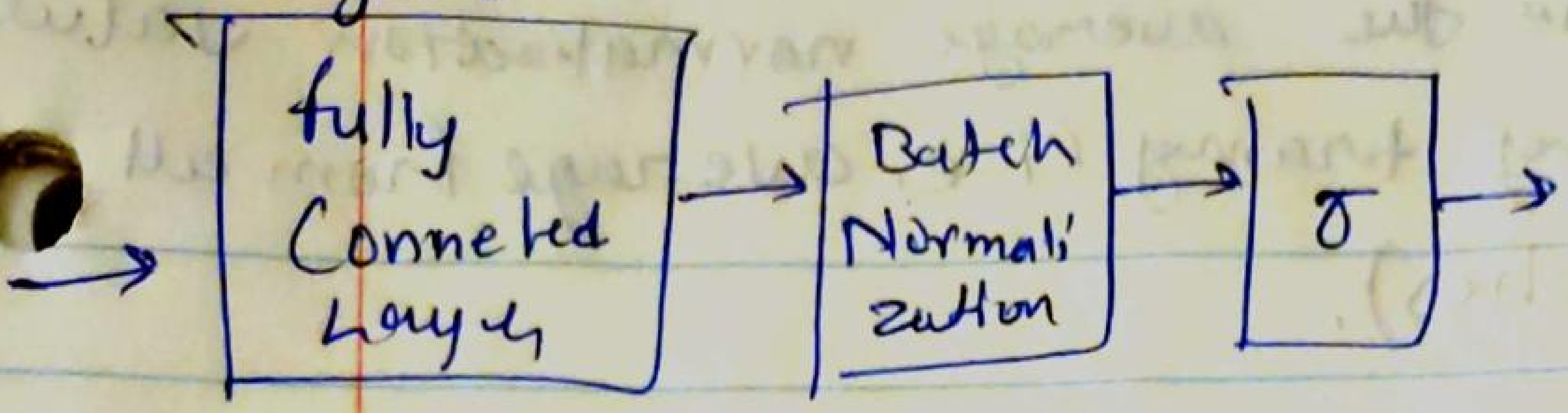
D = D/S

* usually normalize before activation :-



After sigmoid activation is applied values are already between 0 & 1 so doing ~~no~~ normalization after this doesn't make any sense

Layers



we can do BM after activation, however for Sigmoid its nor recommended to do before activation.

Q7 Some Saturation is needed in the network learning after a while so that the network Converges to certain values for these we Scale and shift after normalization.

$$\{Z^{(i)}\}_{i=1}^{q} \to \{\hat{Z}^{(i)}\}_{i=1}^{q} \to \{\tilde{Z}^{(i)}\}_{i=1}^{q}$$

$$\tilde{Z}_j^{(i)} = \gamma_j \hat{Z}_j^{(i)} + \beta_j$$

γ_j and β_j are learned. the network can learn to cancel BM if there is no need for it. Eg:

$$\gamma_j = \sigma_j \Rightarrow \tilde{Z}_j^{(i)} = \sigma_j \hat{Z}^{(i)} + \mu_j = \cancel{\sigma_j} \frac{Z_j^{(i)} - \mu_j}{\cancel{\sigma_j}} + \mu_j$$

$$= Z_j^{(i)}$$

$$\{\beta_j = \mu_j\}$$

During the training we do BN because batches are random BN adds randomness into the training and so reduces overfitting. During prediction we

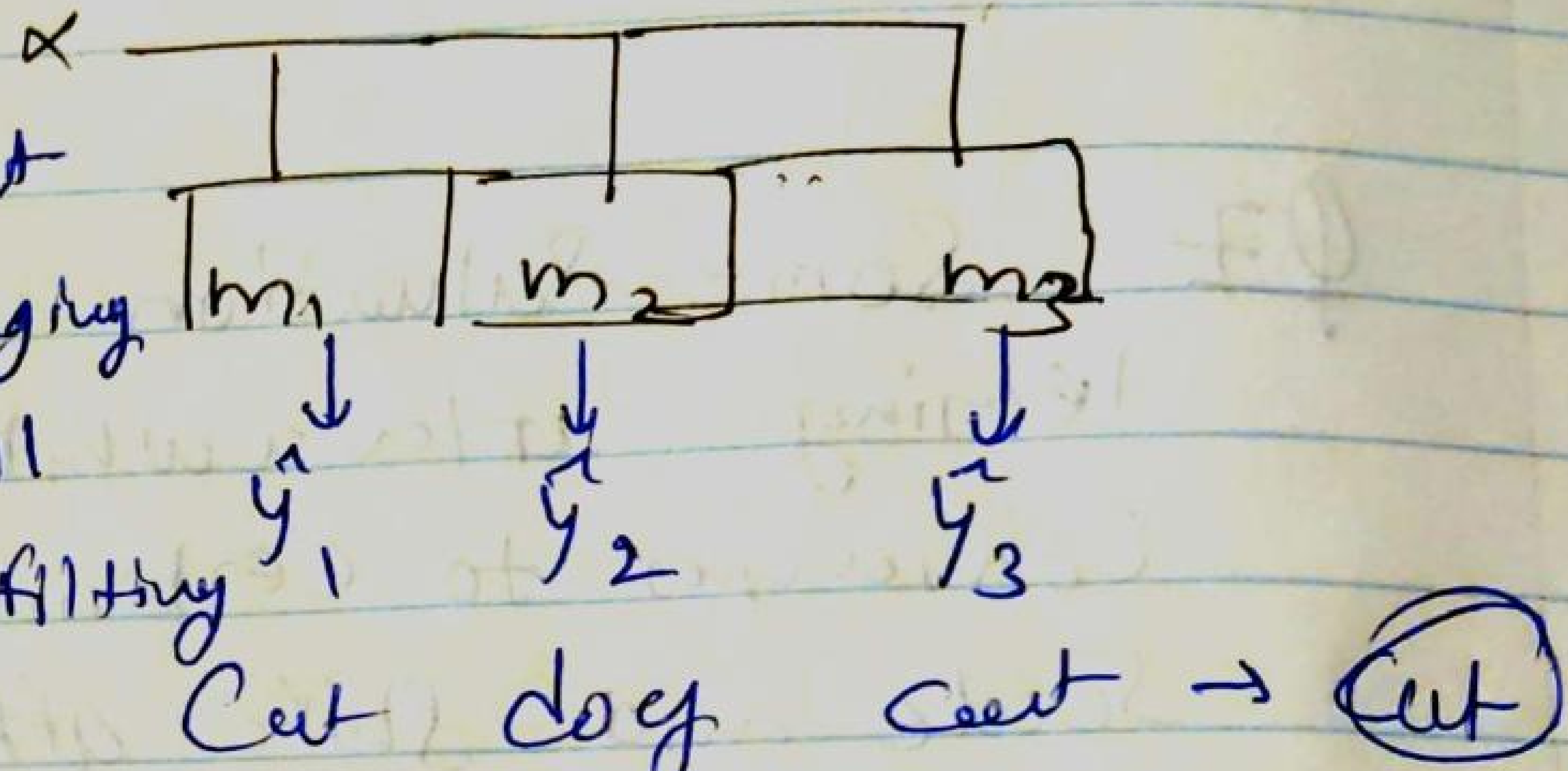
During Prediction we use the average normalization values computed during training (i.e. average from all training batches).

Q8 Ensemble Classifiers →

① train multiple independent models.

② use majority vote or average during testing.

* Individually the models might ~~have~~ overfits but while averaging out the final result, it will damper the effect of overfitting



* To obtain multiple models :→
  - Change data → randomization (Seed)
  - → change parameter (no of layers, units, learning rate)
  - → Record multiple Snapshots of the model during training. (vary learning rate).
  - → There is no loss or tuning of ensemble,
  - → for performance measure, we can use cross entropy to measure predicted & observed results.