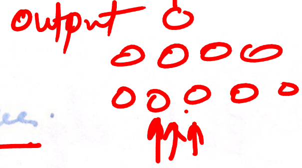


The top layer V_T is called the output layer.

In simple prediction problems, the output layer contains a single neuron whose output is the output of the network. $h(x) \in \mathbb{R}$

We refer to T as the number of layers/depth of the network. The size of the network is $|V|$ #nodes

The width of the network is $\max_t |V_t|$

Suppose we have calculated the outputs of the neurons at V_t. We then computes the activation value for every neuron j as

E: edges of the graph

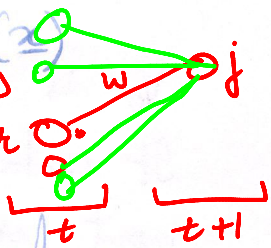
layer, neuron id.

Affine transform

scalar value

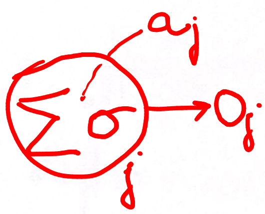
(layer#, node id)

$$\underbrace{a_{t+1,j}(x)}_{\text{activation computed by node } j \text{ in layer } t+1} = \sum_{r:\,\underbrace{(v_{t,r},\, v_{t+1,j}) \in E}} \underbrace{w(v_{t,r},\, v_{t+1,j})}_{\text{edge weights}}\ \underbrace{o_{t,r}}_{\text{o/p of the prev layer}}(x)$$

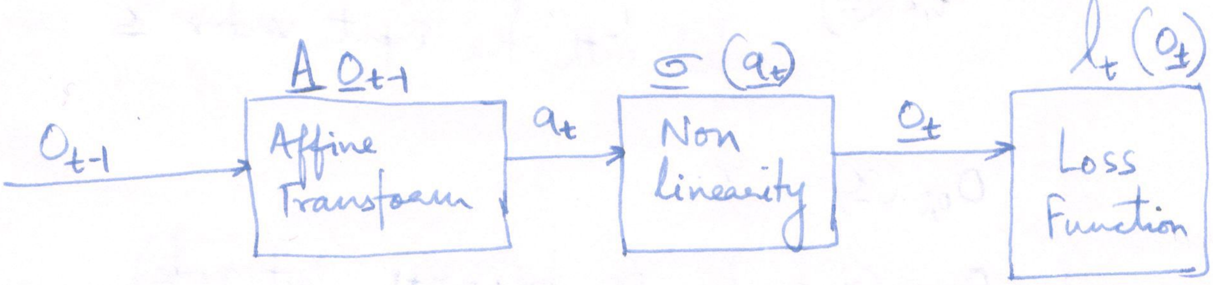


Applying the non-linearity to the activation value:

$$o_{t+1,j}(x) = \sigma(a_{t+1,j}(x))$$



Forward Propagation

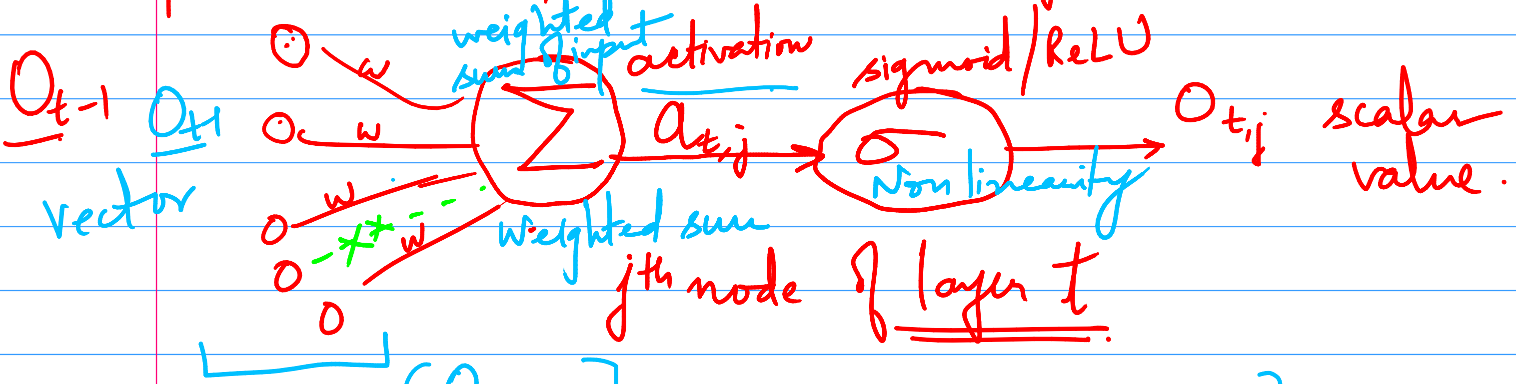


Matrix $\underline{A}$ is formulated using layer weights $\underline{W}_{t-1}$

Activations $\underline{a}_t = \underline{A}\ \underline{o}_{t-1} = \underline{B}\ \underline{W}_{t-1}$ where matrix $\underline{B}$ is formulated using $\underline{o}_{t-1}$

Computations in a layered Network

Report the computation at every node



$$\underline{O}_{t-1} \equiv \begin{bmatrix} O_{t-1,1} \\ O_{t-1,2} \\ \vdots \\ O_{t-1,k_{t-1}} \end{bmatrix} \qquad \underline{w} = \begin{bmatrix} w(v_{t-1,1}, v_{t,j}) \\ w(v_{t-1,2}, v_{t,j}) \\ \vdots \end{bmatrix}$$

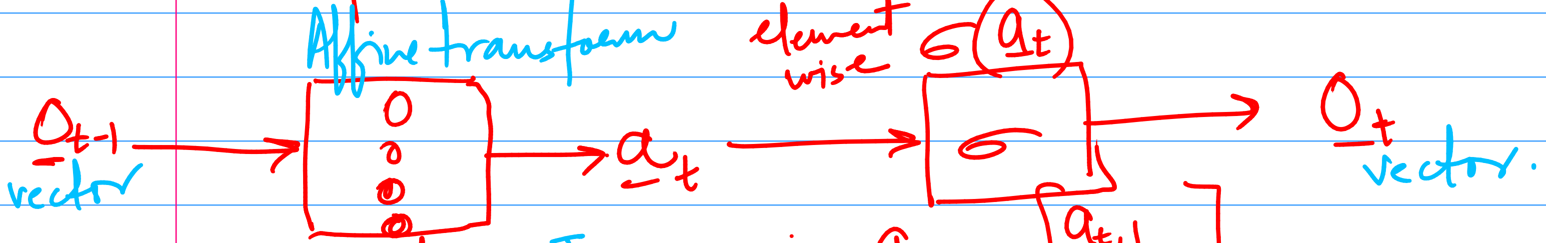
k_{t-1} neurons

$$a_{t,j} = \underbrace{\underline{w}^T \underline{O}_{t-1}}_{\text{inner product}} = \underline{O}_{t-1}^T \underline{w}$$

$\mathbb{R}$. scalar output

$$O_{t,j} \equiv \sigma(a_{t,j})$$

Computations carried out by a layer



$$\underline{a}_t = \begin{bmatrix} a_{t,1} \\ a_{t,2} \\ a_{t,3} \\ \vdots \\ a_{t,k_t} \end{bmatrix}$$

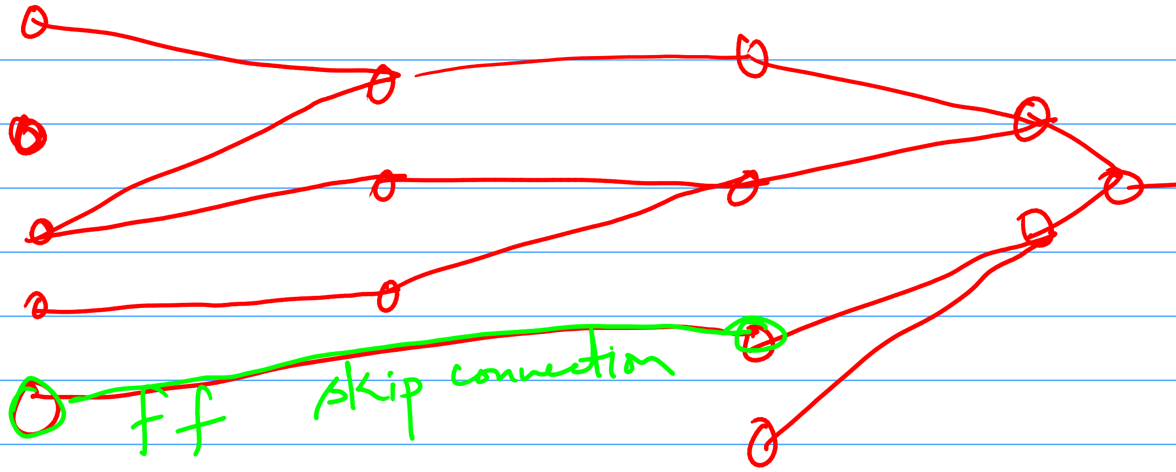
All Nodes compute in parallel $\underline{W}_{t-1}^T \underline{O}_{t-1}$

$\underline{W}_{t-1}$ weight matrix

k_t = # nodes in the t^{th} layer

Architecture (V, E)

layers.

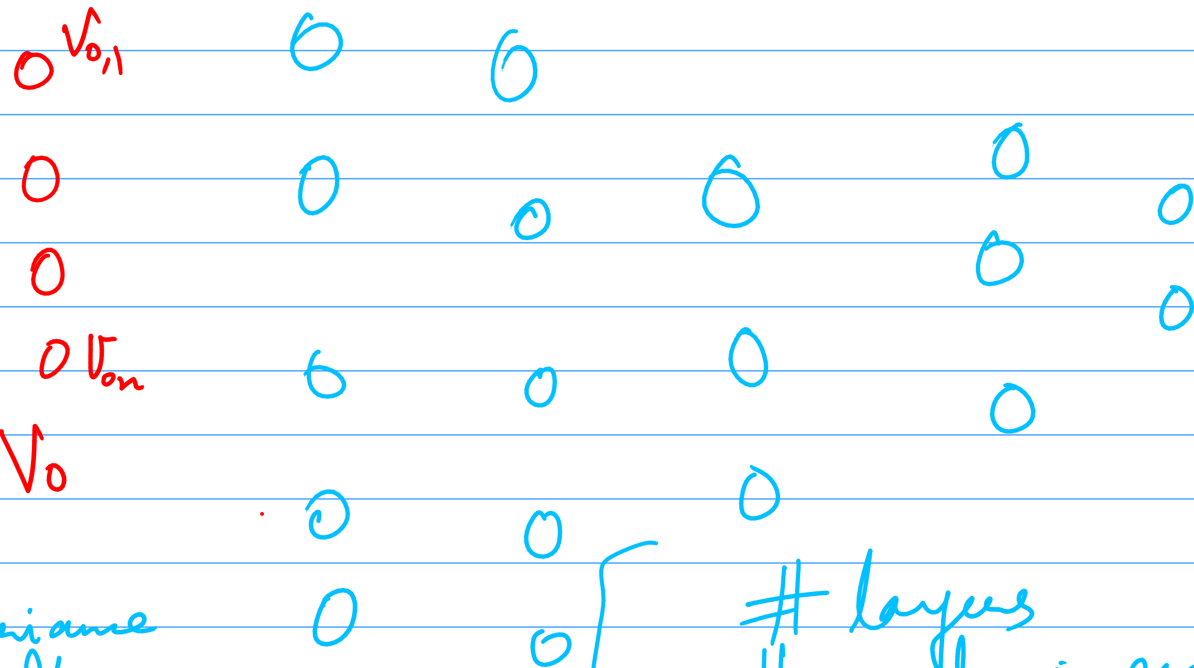


Dense Connection

Sparse Connection

(V, E)

w: learnable parameters



n.

Bais-Variance trade off.

?? →

. . →

Architecture
- # layers
- # nodes in each layer
- (edges) Connections between nodes

$\to \boxed{A} \to \boxed{\sigma} \to \boxed{B} \to \boxed{\sigma} \to \boxed{C} \xrightarrow{\sigma}$ output

$W = ABC$

$\to \boxed{A\ B\ C} \to$

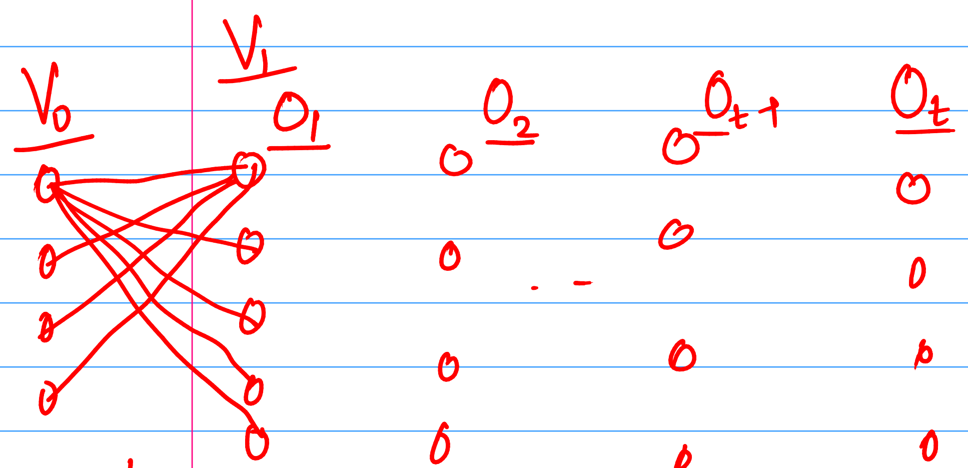
provided $\sigma(a) = I$ identity

Behaves as a linear transformation

Universal Approximators] Non linearity is important.

$\mathcal{H}_{V,E,\sigma}$

$h \in \mathcal{H}$ which can correctly model it

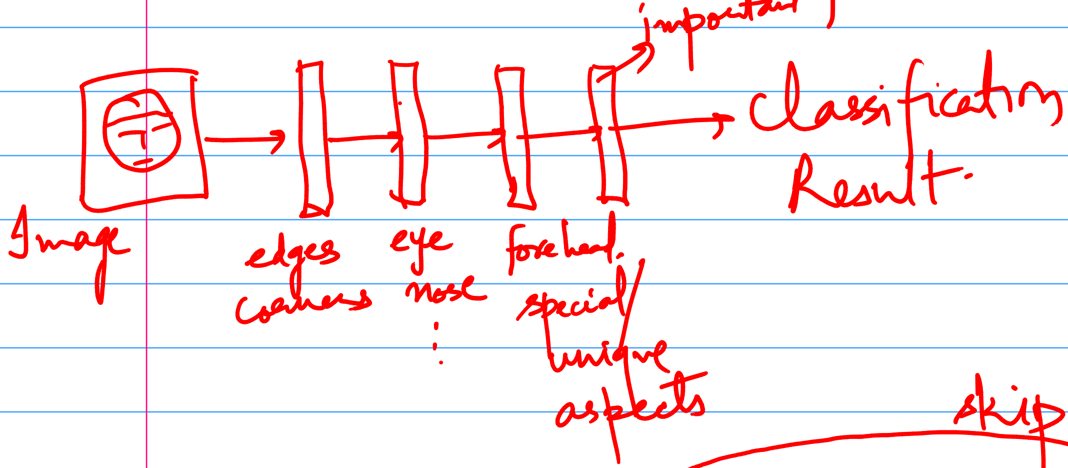


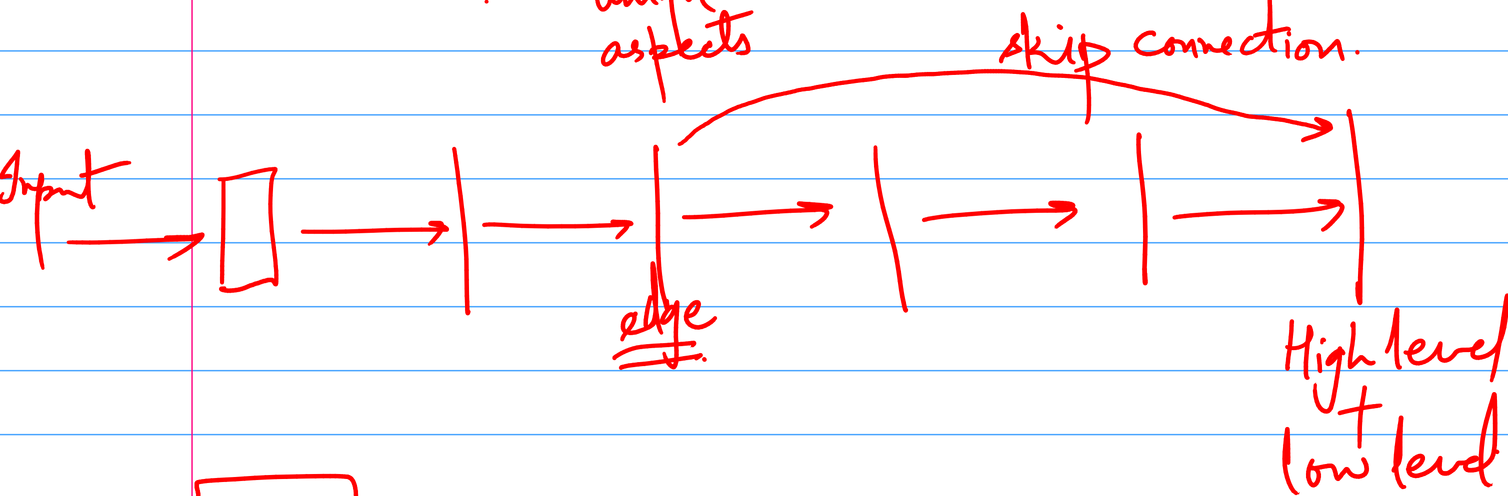
input layer
no computations

$\longrightarrow$ feed forward network.

Features of higher level of abstraction are computed by the deeper layers

low level features are composed to form high level features





Backpropagation computes the gradients of the loss w.r.t. parameters of each & every layer.

Weight update using gradients

SGD

Learning features automatically.

features $\longrightarrow$ $\boxed{\text{ML model}}$ $\longrightarrow$ h

Engineered / Handcrafted / designed. | <u>Learned.</u>

<u>Representation Learning</u>

$$\underline{x} \begin{bmatrix} - \\ - \\ - \\ - \end{bmatrix} \longrightarrow \begin{bmatrix} - \\ - \\ - \\ - \end{bmatrix}$$