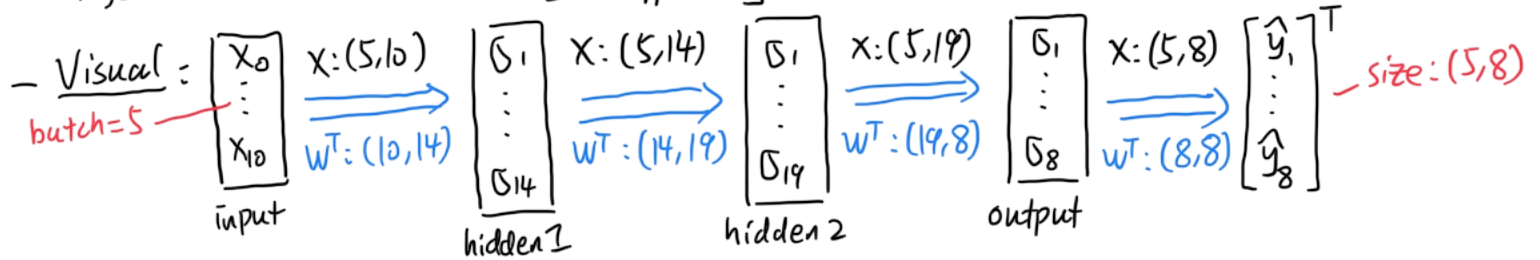




15) Weight Inits

Weight Matrix

- Given $y = xW^T$, where $W = \begin{bmatrix} -w_1^T & - \\ -w_2^T & - \\ \vdots & \\ -w_n^T & - \end{bmatrix}$
 - \rightarrow each row w_i = weights for node i
 - $\# \text{ column} = \# \text{ nodes in this layer}$
 - \rightarrow so $W.\text{shape} = (\text{output}, \text{input})$



Weight Initialization

- **Weight Symmetry**: issue where model has same params (~ 0 gradients)
- Solution: Random Weights provides model the **variability** & **direction** to learn
- so draw weight values from ① Normal \mathcal{N}  ② Uniform \mathcal{U} 

Weight-Init Algorithms:

- ① **Kaiming**: draw $w \in \mathcal{U}(-\sigma, \sigma)$ with $\sigma^2 = \frac{1+a^2}{N_{in}}$ \leftarrow slope of activation (negative part) = 1 for linear, 0 for ReLU
- ② **Xavier**: draw $w \in \mathcal{N}(0, \sigma^2)$ with $\sigma^2 = \frac{2}{N_{in} + N_{out}}$ \leftarrow # input & output features

Freezing Weights

- Deactivate gradient descent in a layer (for transfer learning)
- How?: set `requires_grad = False` for that layer

Quantify Weight Changes

- Weight matrix changes over time (its distribution widens)

- metrics:
 - ① **Frobenius norm** $d = \sqrt{\sum_{i=1}^M \sum_{j=1}^n (W_{i,j}^{(t)} - W_{i,j}^{(t-1)})^2}$ \leftarrow large d = large change in weights = a lot of learning
 - ② **Condition number** $k = \sigma_{\max} / \sigma_{\min}$ \leftarrow large k = sparse weight matrix = layer learned specific features