Deep Learning in Practice

Wednesday, October 23, 2024 8:02 AM

Architecture choices:

- DAG: # layers, etc
- Activation functions (logistic, tanh, ReLU, ...)
- Error or log-loss functions
- Input preprocessing

Data choices: Train, validation, test, cross-validation, ...

Algorithm choices:

- Initialization of weights & biases
- Gradient descent, SGD, minibatches, line search, momentum, Adam, ...
- Stopping criterion, weight regularization, dropout, ...

Deep Learning in Practice

Wednesday, October 23, 2024 8:02 AM

Architecture choices:

- DAG: # layers, etc
- Activation functions (logistic, tanh, ReLU, ...)
- Error or log-loss functions
- Input preprocessing

Data choices: Train, validation, test, cross-validation, ...

Algorithm choices:

- Initialization of weights & biases
- Gradient descent, SGD, minibatches, line search, momentum, Adam, ...
- Stopping criterion, weight regularization, dropout, ...

Weight Initialization (DL 8.4)

Wednesday, October 23, 2024 8:14 AM

- Recall that $\frac{\partial E}{\partial w_{lm}} \propto g'(x_m)$
- If $g'(x_m)$ is close to zero, it will be very difficult to change w_{lm}
  $\Rightarrow$ Initialize w's so $|g'(x_m)| > \epsilon$ ← Minimum derivative
- For $g = \tanh$, if we sample weights from a zero-mean distribution, then

$$E[x_m] = \sum_{i=1}^{m-1} E[w_{im}] E[g(x_i)] = 0 \quad (E[w_{im}] \to 0)$$

$$V[x_m] = \sum_{i=1}^{m-1} V[w_{im} g(x_i)] = \sum_{i=1}^{m-1} V[w_{im}] V[g(x_i)] \quad (V[g(x_i)] \leq 1)$$

$$\leq \sum_{i=1}^{m-1} V[w_{im}] = (\#\text{inputs to } x_m) \times V[w]$$

- So, sample w's with variance $= \frac{1}{\#\text{inputs}}$

Glorot & Bengio (DL 8.4)

Wednesday, October 23, 2024 8:35 AM

- Consider a fully connected layer with m inputs & n outputs

- Sample w's from $\text{Uniform}\left(-\sqrt{\frac{6}{m+n}}, \sqrt{\frac{6}{m+n}}\right)$

- Compromise between ensuring activation variance is the same across layers and all w's have same gradient variance

$$\text{Variance} \propto \frac{6}{m+n}$$

Deep Learning in Practice

Wednesday, October 23, 2024 8:02 AM

Architecture choices:

- DAG: # layers, etc
- Activation functions (logistic, tanh, ReLU, ...)
- Error or log-loss functions
- Input preprocessing

Data choices: Train, validation, ~~test~~, cross-validation, ...

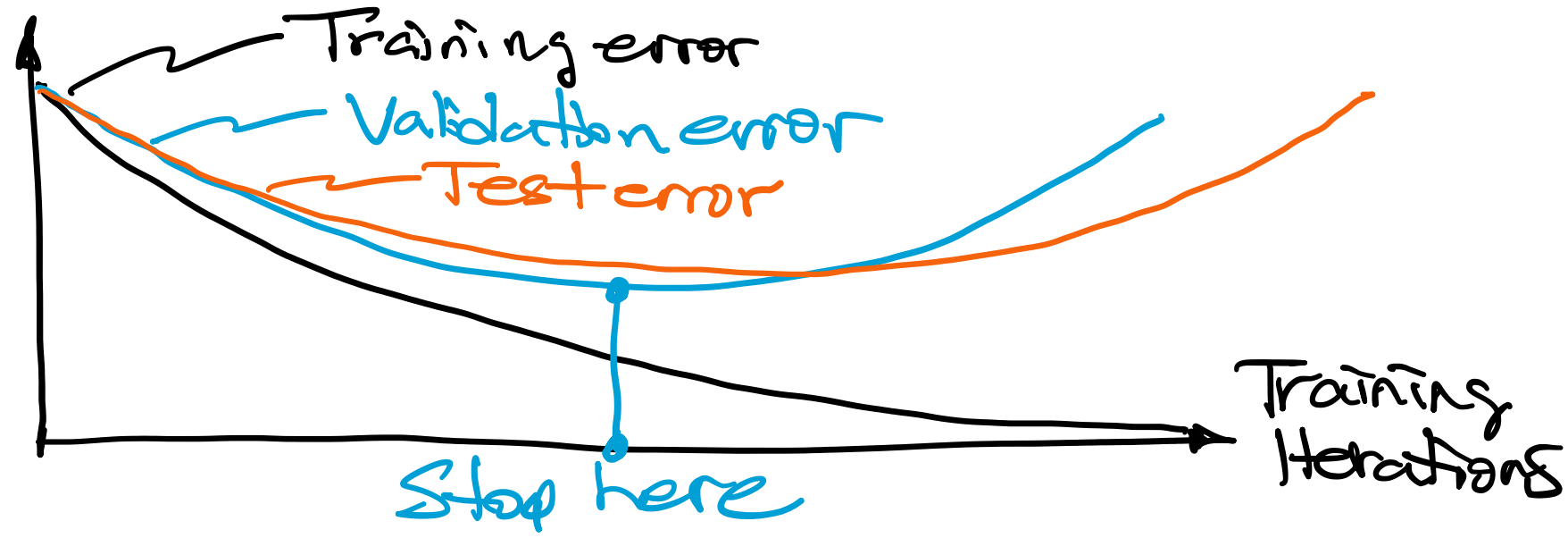
Algorithm choices:

- Initialization of weights & biases
- Gradient descent, SGD, minibatches, line search, momentum, Adam, ...
- Stopping criterion, weight regularization, dropout, ...

Avoiding overfitting

Saturday, November 2, 2024 5:08 PM

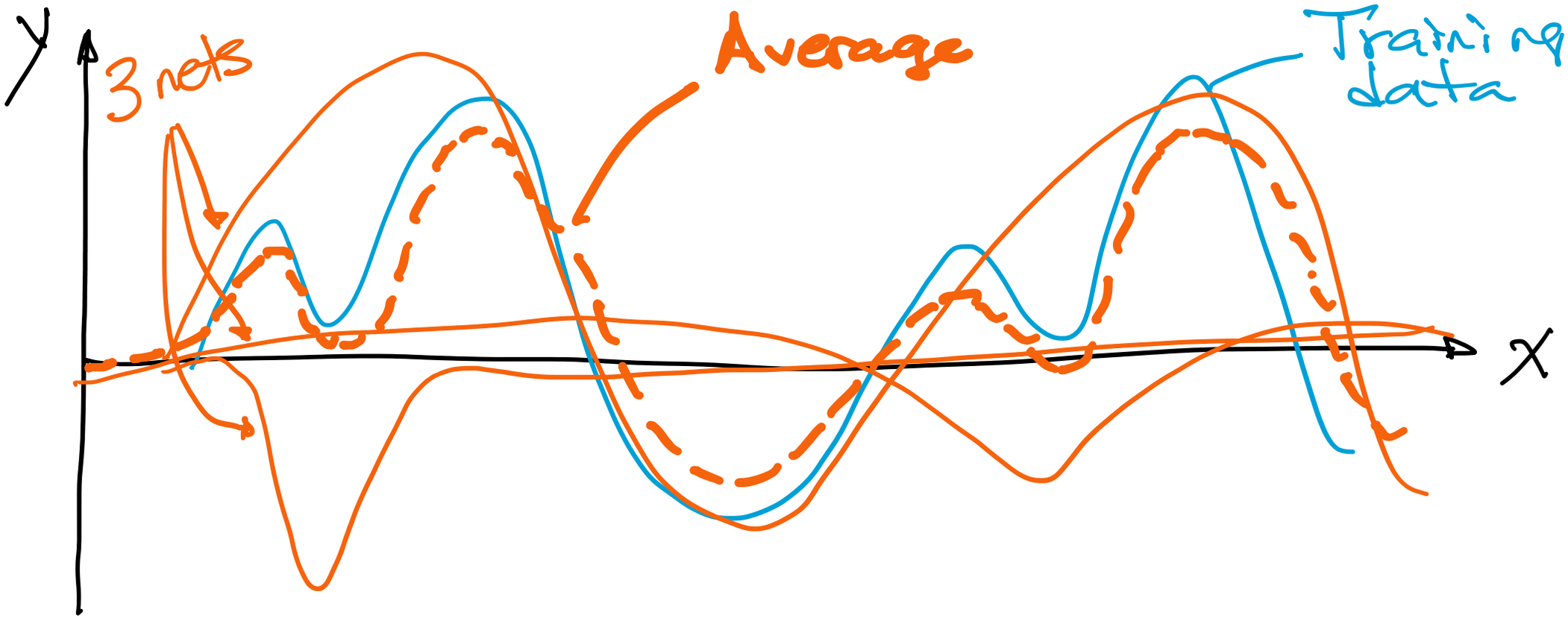
- Weight penalties (DL 7.1): Minimize $E(\text{Data}, w) + \lambda \|w\|$
  → Favors small weights & neural nets that make "smooth" predictions
- Early stopping (DL 7.8): Start with tiny weights monitor a validation set during training, and stop when the error increases



Avoiding overfitting

Saturday, November 2, 2024 5:14 PM

- Bagging (DL 7.11): Train many (eg, 10) neural nets, and allow them to overfit. To make a prediction for a ~~test~~ case, average the predictions from the ensemble of neural nets.



Wasted/stuck units

Saturday, November 2, 2024 5:25 PM

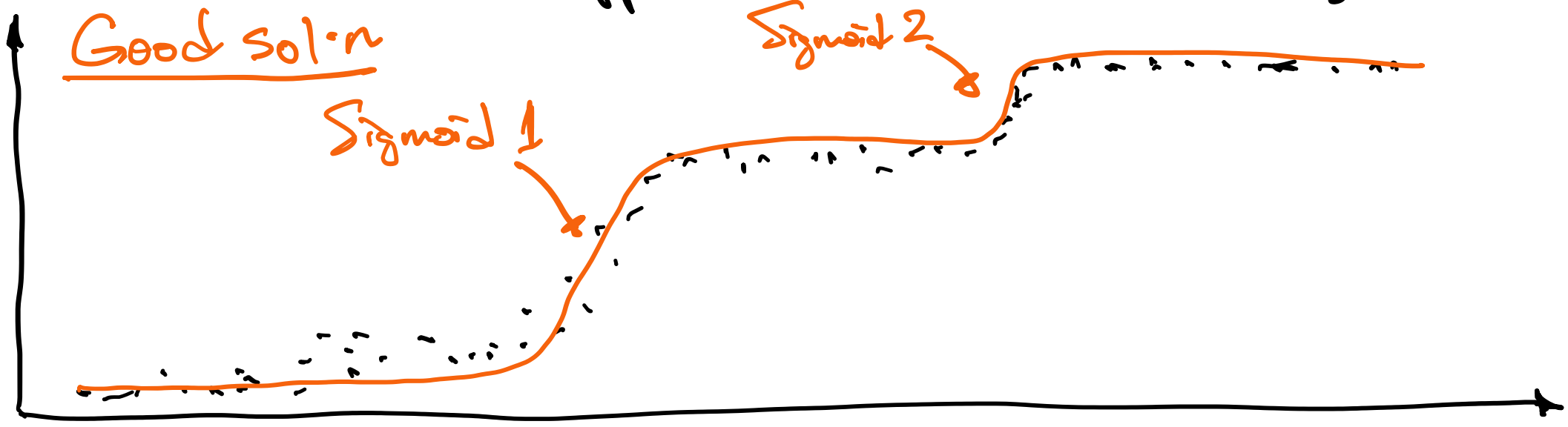
- The main challenge for large neural nets is not just overfitting, but that most hidden units get <u>stuck where they aren't needed</u> & are thus <u>wasted</u>

- Weight regularization & early stopping don't help
- Bagging somewhat helps, but
  - The number of nets needed grows <u>rapidly</u> with the complexity of the data
- When training each net in the ensemble, most units still get stuck.

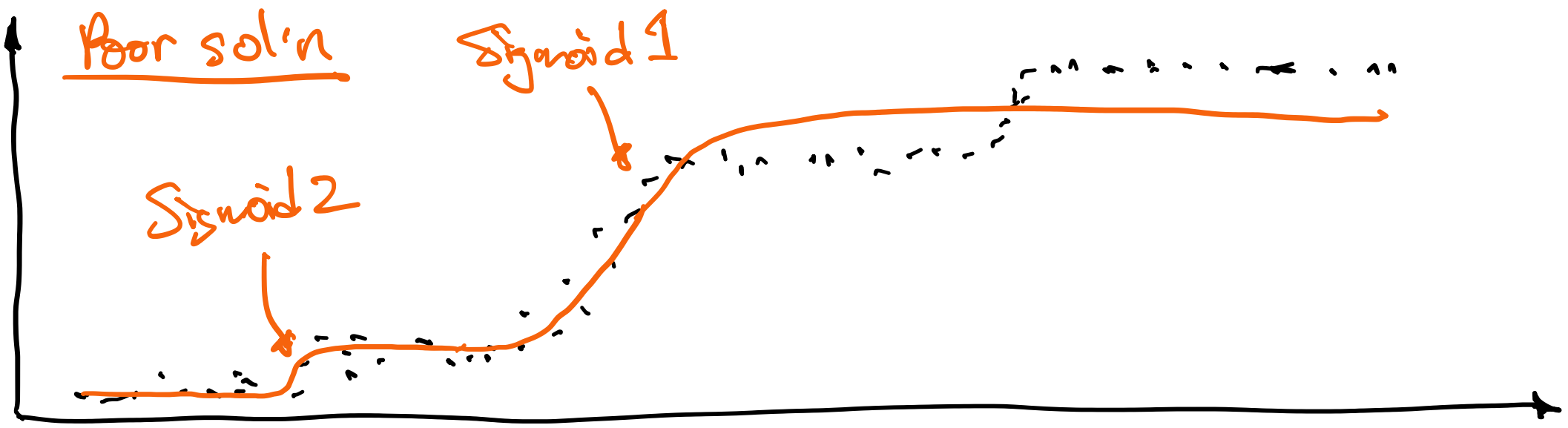
Example: Stuck units

Sunday, November 3, 2024 10:32 AM

Consider function approximation using 2 logistic units



Here's a local minimum where sigmoid 2 is "stuck"



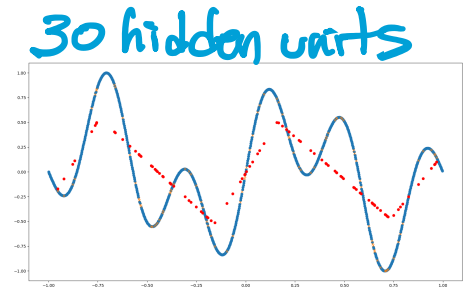
Dropout (DL 7.12)

Wednesday, October 23, 2024 8:32 AM

Eg, GPT-3 uses dropout

- Goal: Train very large networks while avoiding overfitting and wasting most hidden units

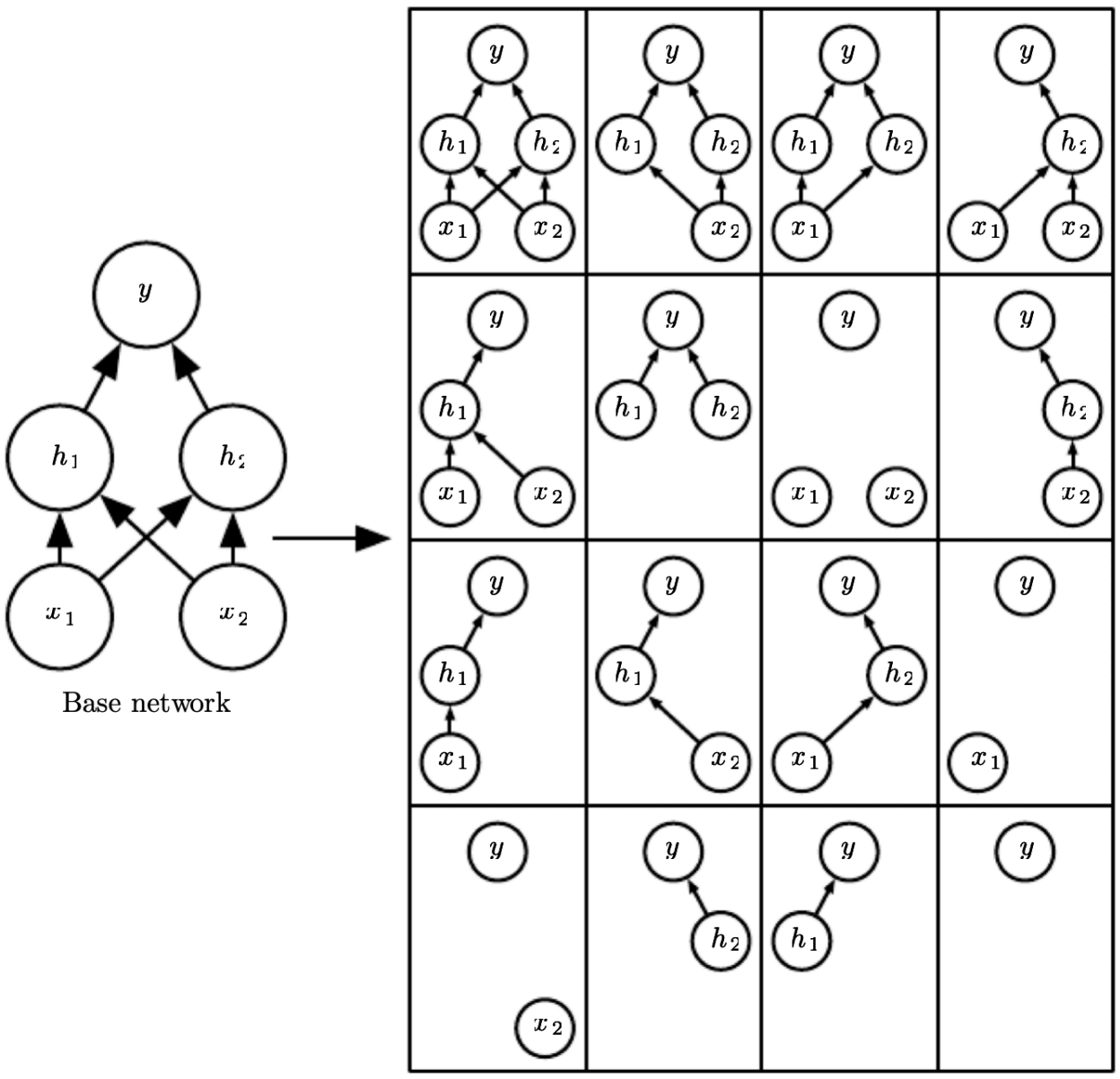
Recall the wasted hidden units for a simple task



- Method
  - Set dropout probabilities $\{\mu_i\}$, eg, for input units $\mu_i = 0.8$ & for hidden units $\mu_i = 0.5$.
  - During training, for each forward pass, set the activity of x_i to zero with probability $1-\mu_i$ & independent of other units
  - During testing, set $w_{ij} \leftarrow w_{ij} \cdot \mu_i \ \forall_{ij}$.

Dropout: Cooperation & exponential bagging (DL 7.12)

Saturday, November 2, 2024 5:49 PM



Base network

Ensemble of subnetworks

These are the nets that get sampled in the forward pass.

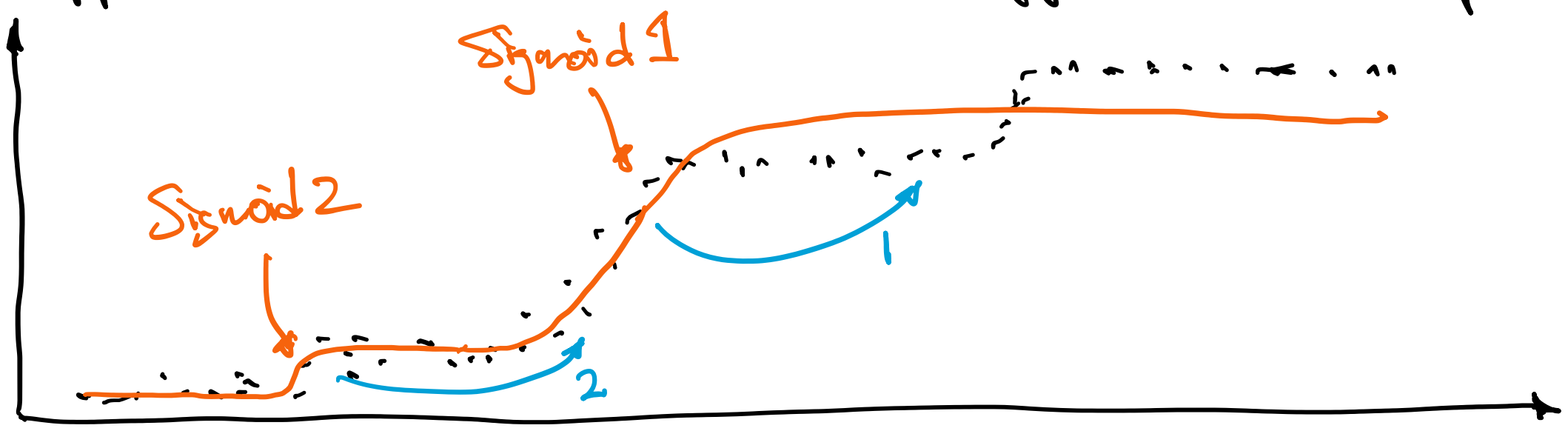
- Dropout simulates an exponential number of neural architectures
  → Exponential bagging.

- Because the weights are shared across all networks and they are trained jointly, the units can cooperate during training to get unstuck.

Dropout Example

Sunday, November 3, 2024 10:32 AM

Suppose each hidden unit is dropped out with $p = 0.5$



- 1/4 of the time only sigmoid 2 is used to predict the output, so it will be pressured to take over for sigmoid 1, which can then capture the incline on the far right

Deep Learning in Practice

Wednesday, October 23, 2024 8:02 AM

Architecture choices:

- DAG: # layers, etc
- Activation functions (logistic, tanh, ReLU, ...)
- Error or log-loss functions
- Input preprocessing

Data choices: Train, validation, test, cross-validation, ...

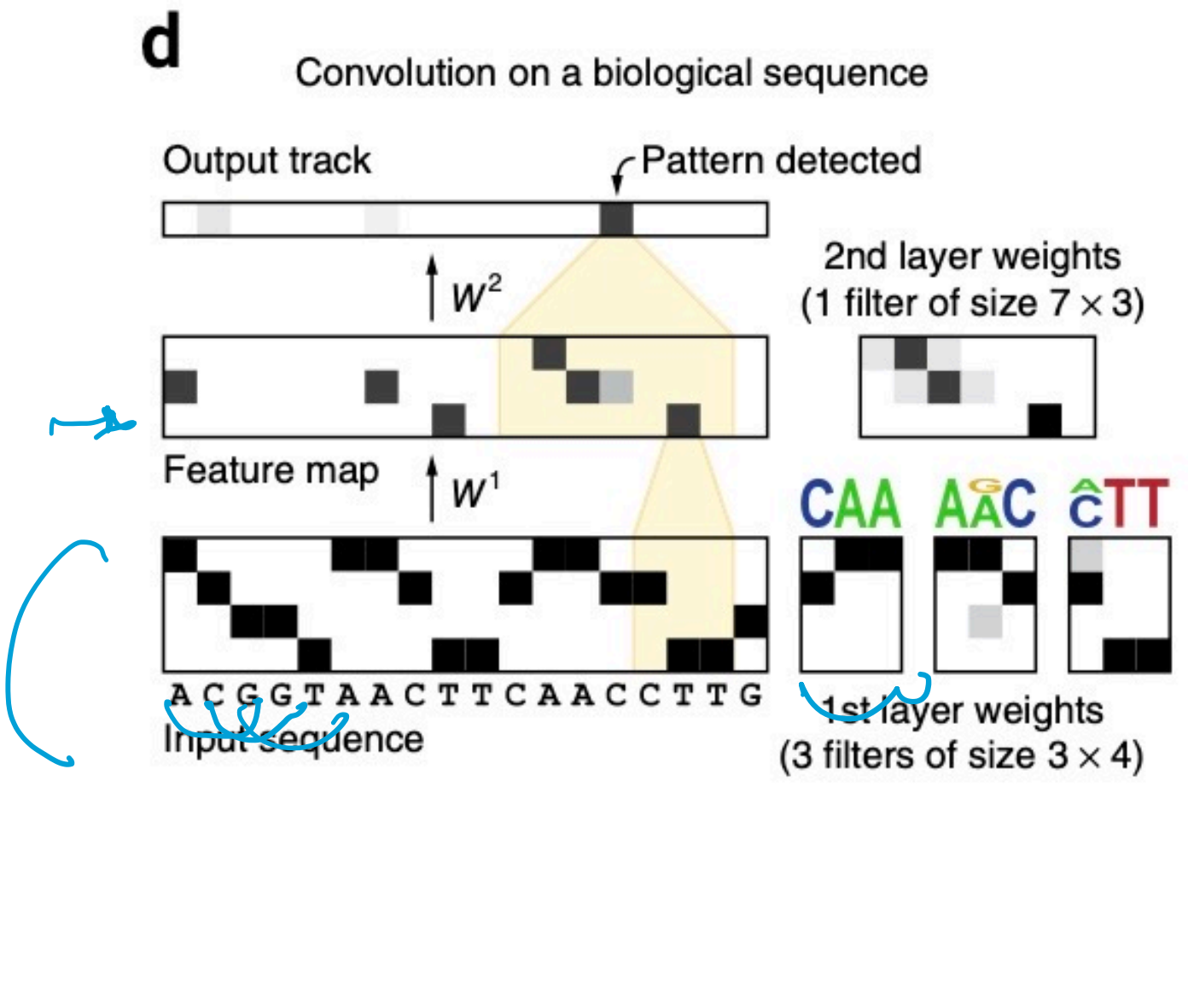
Algorithm choices:

- Initialization of weights & biases
- Gradient descent, SGD, minibatches, line search, momentum, Adam, ...
- Stopping criterion, weight regularization, dropout, ...

Convolutional neural nets (CNNs)

Saturday, November 2, 2024 5:56 PM

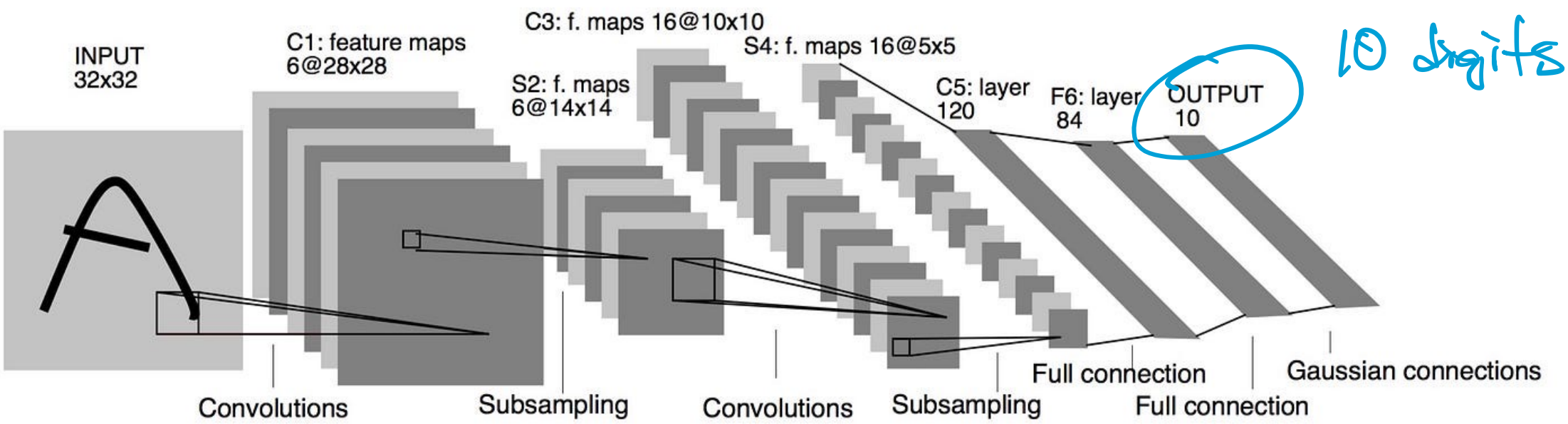
- To avoid overfitting and ensure cooperation of hidden units during training, employ <u>modularity</u>
- Use convolution modules: Recall from before:



LeCun, 1989

Saturday, November 2, 2024 6:02 PM

- Best result on handwritten digit recognition in 1989



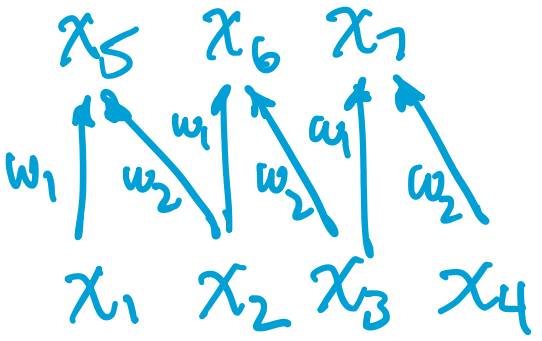
10 digits

2-D convolutions

Weight sharing (DL 7.9)

Saturday, November 2, 2024 6:13 PM

- In CNNs and other architectures we can have a situation where many connections share the same, single weight
- Let $K(i,j)$ be the index of the weight on connection $i \to j$. Instead of $x_j = \sum_{i=1}^{j-1} w_{ij}\, g_i(x_i)$ we have $x_j = \sum_{i=1}^{j-1} w_{K(i,j)}\, g_i(x_i)$
- Example



$K(1,5)=1,\ K(2,6)=1,\ K(3,7)=1$

$K(2,5)=2,\ K(3,6)=2,\ K(4,7)=2$

Weight sharing: Learning (DL 7.9)

Saturday, November 2, 2024 6:20 PM

- Before, for $\partial E / \partial w_{lm}$, we had that w_{lm} influences E only via x_m, so $\frac{\partial E}{\partial w_{lm}} = \frac{\partial E}{\partial x_m}\frac{\partial x_m}{\partial w_{lm}} = \frac{\partial E}{\partial x_m} g_l(x_l)$

- Now, w_k influences E via all x_m s.t. $K(l,m) = k$, so

Use back prop to compute this

$$\frac{\partial E}{\partial w_k} = \sum_{\substack{l,m: \\ K(l,m)=k}} \frac{\partial E}{\partial x_m}\frac{\partial x_m}{\partial w_k} = \sum_{\substack{l,m: \\ K(l,m)=k}} \frac{\partial E}{\partial x_m} g_l(x_l)$$

- Algorithm
  - Forward & back propagation are same as before
  - The weight derivatives are computed as above.

Weight sharing: Learning (DL 7.9)

Sunday, November 3, 2024 10:09 AM

Forward propagation: for $j = I+1 \dots M$, $x_j = \sum_{i=1}^{j-1} w_{K(i,j)} g_i(x_i)$

Output unit error derivatives:

For $m = M \dots M-O+1$: $\frac{\partial E}{\partial x_m} = -2(y_m - g_m(x_m))g'_m(x_m)$

Backpropagation:

For $m = M-O \dots M-O-H+1$: $\frac{\partial E}{\partial x_m} = \sum_{k=m+1}^{M} \frac{\partial E}{\partial x_k} w_{K(m,k)} g'_m(x_m)$

Weight derivatives:

For $l = 1 \dots M-O+1$, for $m = l+1 \dots M$

$$\frac{\partial E}{\partial w_k} = \sum_{l,m:} \frac{\partial E}{\partial x_m} g_l(x_l)$$

$K(l,m) = k$

Learning rate

$$w_{lm} \leftarrow w_{lm} - \eta \frac{\partial E}{\partial w_{lm}}$$

Deep Learning in Practice

Wednesday, October 23, 2024 8:02 AM

Architecture choices:

- DAG: # layers, etc
- Activation functions (logistic, tanh, ReLU, ...)
- Error or log-loss functions
- Input preprocessing

Data choices: Train, validation, test, cross-validation, ...

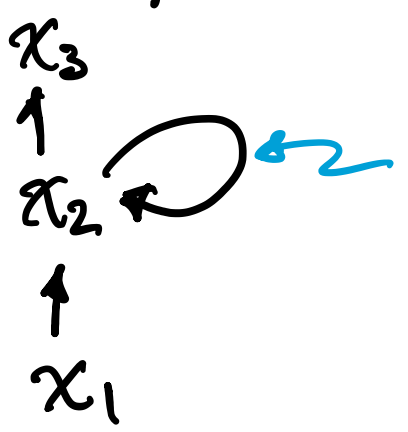
Algorithm choices:

- Initialization of weights & biases
- Gradient descent, SGD, minibatches, line search, momentum, Adam, ...
- Stopping criterion, weight regularization, dropout, ...

Recurrent neural networks (RNN, DL 10.1-10.2)

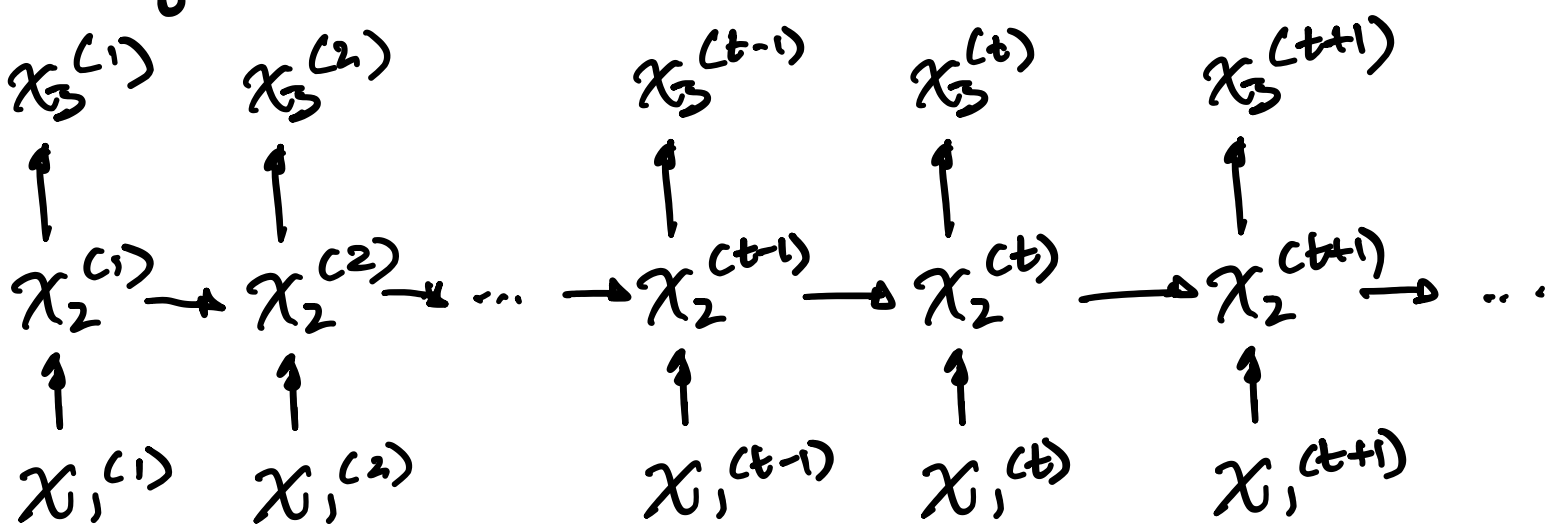
Saturday, November 2, 2024 6:11 PM

- Instead of a DAG, we allow cycles to denote signals feeding back <u>across time</u>:



x_2 at time t depends on x_2 at time $t-1$ and x_1 at time t.

- Unrolling an RNN

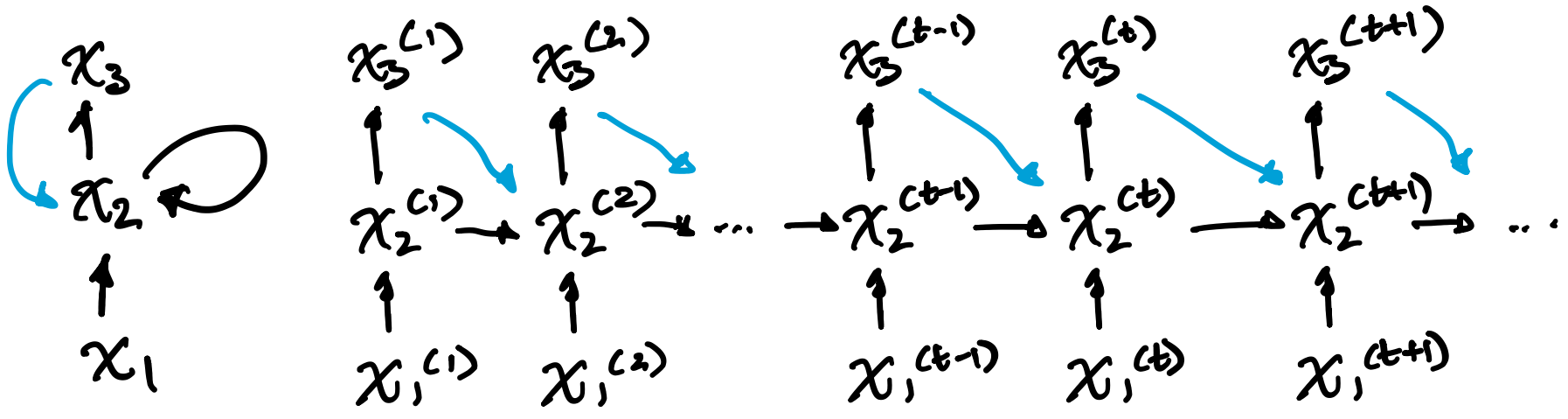


- Learning: This is <u>a DAG NN with weight sharing</u>, so we can use the method described previously.

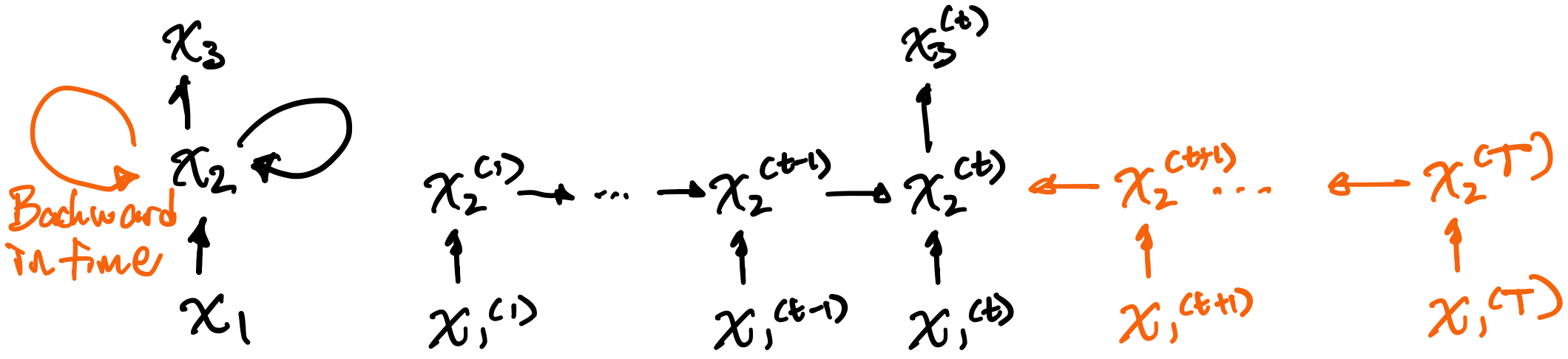
Improving RNN as sequence generators (DL 10.2)

Sunday, November 3, 2024 10:52 AM

- **Conditional RNNs**: Output (t) "sees" outputs (1) ... (t-1)



- **Bidirectional RNN's**: Output (t) "sees" inputs (1) ... (T)



The problem of vanishing gradients

Sunday, November 3, 2024 11:26 AM

- As gradients propagate through more hidden units, or through time, they get more and more suppressed (vanish) or amplified (explode)

- This makes it hard to learn RNNs with long sequences, or very deep NNs.

- A general approach is to add mechanisms that propagate derivatives directly across time, or layers.

LSTMs: Long short-term memory cells (SLP 8.5)

Saturday, November 2, 2024 6:56 PM

- Create gates that can copy over the previous cell state to the next cell state

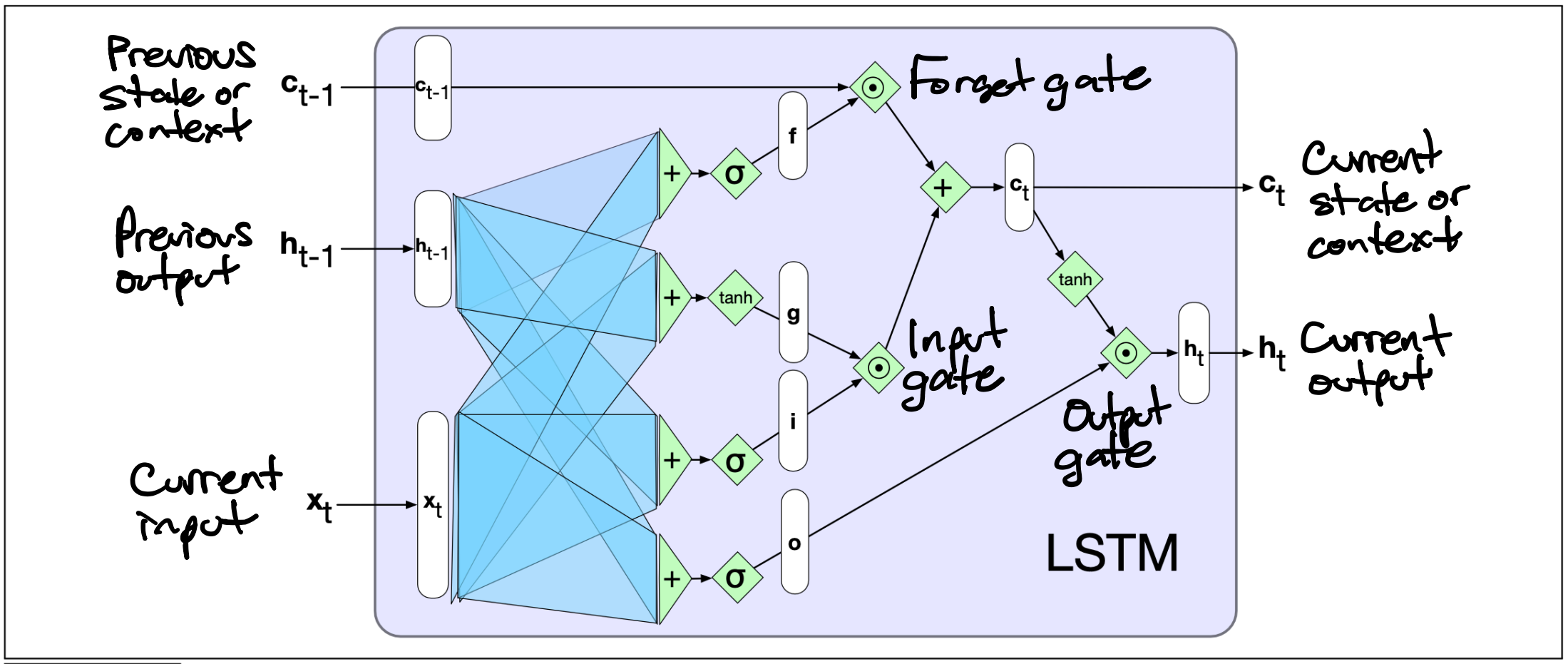


Figure 8.13 A single LSTM unit displayed as a computation graph. The inputs to each unit consists of the current input, x, the previous hidden state, h_{t-1}, and the previous context, c_{t-1}. The outputs are a new hidden state, h_t and an updated context, c_t.

* It's all differentiable!!!