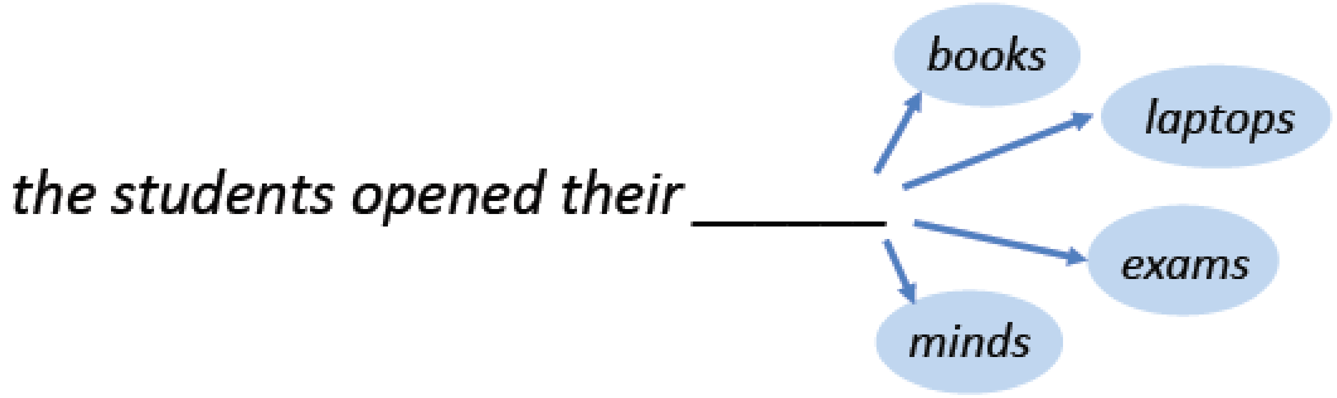
Neural
Language Models

Language Modeling

- **Language Modeling** is the task of predicting what word comes next.



- More formally: given a sequence of words $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(t)}$, compute the probability distribution of the next word $\boldsymbol{x}^{(t+1)}$:

$$P(\boldsymbol{x}^{(t+1)} | \; \boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(1)})$$

where $\boldsymbol{x}^{(t+1)}$ can be any word in the vocabulary $V = \{\boldsymbol{w}_1, ..., \boldsymbol{w}_{|V|}\}$

- A system that does this is called a **Language Model**.

Slide Credit: Abigail See

Language Modeling

- You can also think of a Language Model as a system that assigns probability to a piece of text.

- For example, if we have some text $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(T)}$, then the probability of this text (according to the Language Model) is:

$$
\begin{aligned}
P(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(T)}) &= P(\boldsymbol{x}^{(1)}) \times P(\boldsymbol{x}^{(2)} \mid \boldsymbol{x}^{(1)}) \times \cdots \times P(\boldsymbol{x}^{(T)} \mid \boldsymbol{x}^{(T-1)}, \ldots, \boldsymbol{x}^{(1)}) \\
&= \prod_{t=1}^{T} \underbrace{P(\boldsymbol{x}^{(t)} \mid \boldsymbol{x}^{(t-1)}, \ldots, \boldsymbol{x}^{(1)})}_{\text{This is what our LM provides}}
\end{aligned}
$$

Slide Credit: Abigail See

N-gram Language Models Example

Suppose we are learning a 4-gram Language Model.

$\cancel{\textit{as the proctor started the clock, the}}$ $\underbrace{\textit{students opened their}}_{\text{condition on this}}$ ______

discard

$$P(\boldsymbol{w}|\text{students opened their}) = \frac{\text{count}(\text{students opened their } \boldsymbol{w})}{\text{count}(\text{students opened their})}$$

<u>For example, suppose that in the corpus:</u>

- "students opened their" occurred 1000 times
- "students opened their books" occurred 400 times
  - → P(books | students opened their) = 0.4
- "students opened their exams" occurred 100 times
  - → P(exams | students opened their) = 0.1

Should we have discarded the "proctor" context?

Slide Credit: Abigail See

Sparsity Problems with n-gram Language Models

Sparsity Problem 1

Problem: What if *"students opened their $\boldsymbol{w}$"* never occurred in data? Then $\boldsymbol{w}$ has probability 0!

(Partial) Solution: Add small δ to the count for every $\boldsymbol{w} \in V$. This is called *smoothing*.

$$P(\boldsymbol{w}|\text{students opened their}) = \frac{\boxed{\text{count}(\text{students opened their } \boldsymbol{w})}}{\boxed{\text{count}(\text{students opened their})}}$$

Sparsity Problem 2

Problem: What if *"students opened their"* never occurred in data? Then we can't calculate probability for *any* $\boldsymbol{w}$!

(Partial) Solution: Just condition on *"opened their"* instead. This is called *backoff*.

Note: Increasing *n* makes sparsity problems *worse*. Typically we can't have *n* bigger than 5.

Slide Credit: Abigail See

Storage Problems with n-gram Language Models

Storage: Need to store count for all *n*-grams you saw in the corpus.

$$P(\boldsymbol{w}|\text{students opened their}) = \frac{\boxed{\text{count}(\text{students opened their } \boldsymbol{w})}}{\text{count}(\text{students opened their})}$$

Increasing *n* or increasing corpus increases model size!

Slide Credit: Abigail See

n-gram Language Models in practice

- You can build a simple trigram Language Model over a 1.7 million word corpus (Reuters) in a few seconds on your laptop*

Business and financial news

Try for yourself: https://nlpforhackers.io/language-models/

Generating text with a n-gram Language Model

You can also use a Language Model to generate text.

today the ________

condition on this

get probability distribution

company	0.153
bank	0.153
price	0.077
italian	0.039
emirate	0.039
…	

sample

Generating text with a n-gram Language Model

You can also use a Language Model to generate text.

today $\underbrace{\textit{the price}}_{\text{condition on this}}$ ________

get probability distribution

of	0.308	sample
for	0.050	
it	0.046	
to	0.046	
is	0.031	
…		

Slide Credit: Abigail See

Generating text with a n-gram Language Model

You can also use a Language Model to generate text.

today the $\underbrace{\textit{price of}}_{\text{condition on this}}$ ________

get probability distribution

the	0.072
18	0.043
oil	0.043
its	0.036
gold	0.018
…	

sample

Slide Credit: Abigail See

Generating text with a n-gram Language Model

You can also use a Language Model to generate text.

today the price of gold ________

Generating text with a n-gram Language Model

You can also use a Language Model to generate text.

today the price of gold per ton , while production of shoe lasts and shoe industry , the bank intervened just after it considered and rejected an imf demand to rebuild depleted european stocks , sept 30 end primary 76 cts a share .

Surprisingly grammatical!

...but **incoherent.** We need to consider more than three words at a time if we want to model language well.

But increasing *n* worsens sparsity problem, and increases model size...

Slide Credit: Abigail See

Advantages of Neural Language Models:

- Don’t need smoothing,
- Can handle much longer histories,
- Can generalize over contexts of similar words.

How to build a *neural* Language Model?

Recall the Language Modeling task:

- Input: sequence of words $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(t)}$
- Output: prob dist of the next word $P(\boldsymbol{x}^{(t+1)} | \, \boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(1)})$

How about a window-based neural model?

A fixed-window neural Language Model

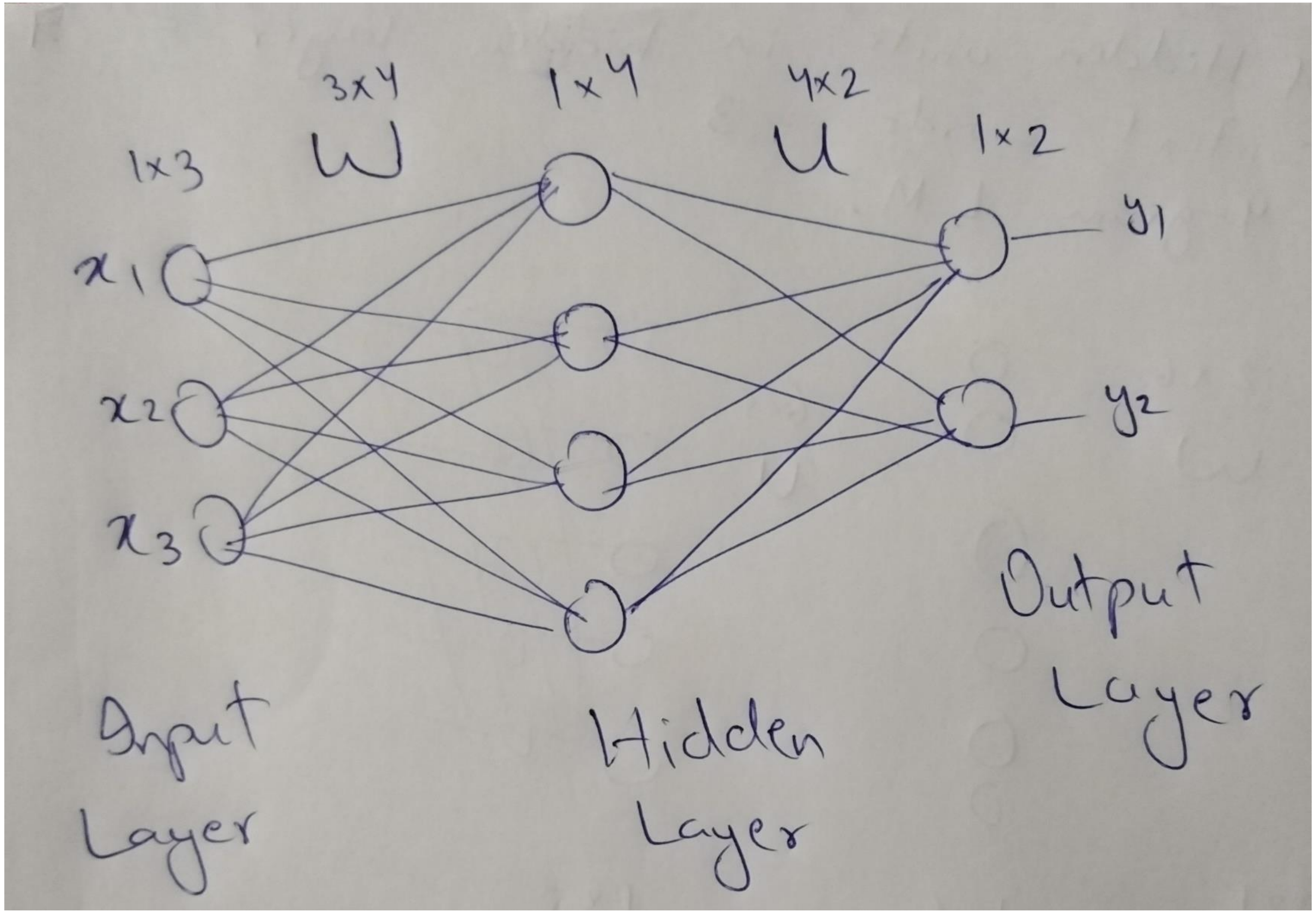
~~*as the proctor started the clock*~~ *the students opened their* ______

discard

fixed window

Slide Credit: Abigail See

2 Layer Neural Network (How to use it as a classifier for language modeling or how to treat language modeling as a classification problem)



A fixed-window neural Language Model

output distribution

$$\hat{\boldsymbol{y}} = \mathrm{softmax}(\boldsymbol{U}\boldsymbol{h} + \boldsymbol{b}_2) \in \mathbb{R}^{|V|}$$

hidden layer

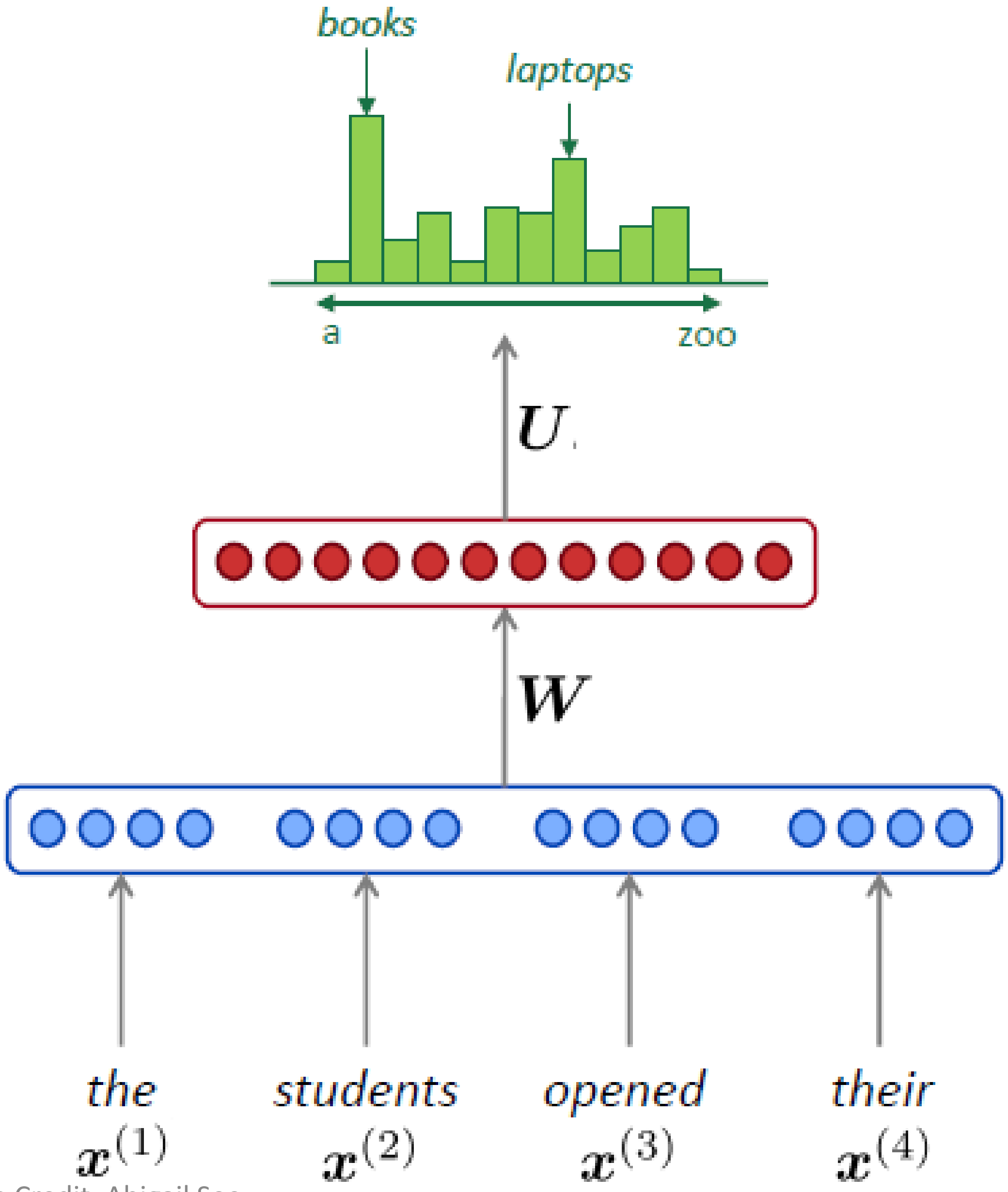
$$\boldsymbol{h} = f(\boldsymbol{W}\boldsymbol{e} + \boldsymbol{b}_1)$$

concatenated word embeddings

$$\boldsymbol{e} = [\boldsymbol{e}^{(1)}; \boldsymbol{e}^{(2)}; \boldsymbol{e}^{(3)}; \boldsymbol{e}^{(4)}]$$

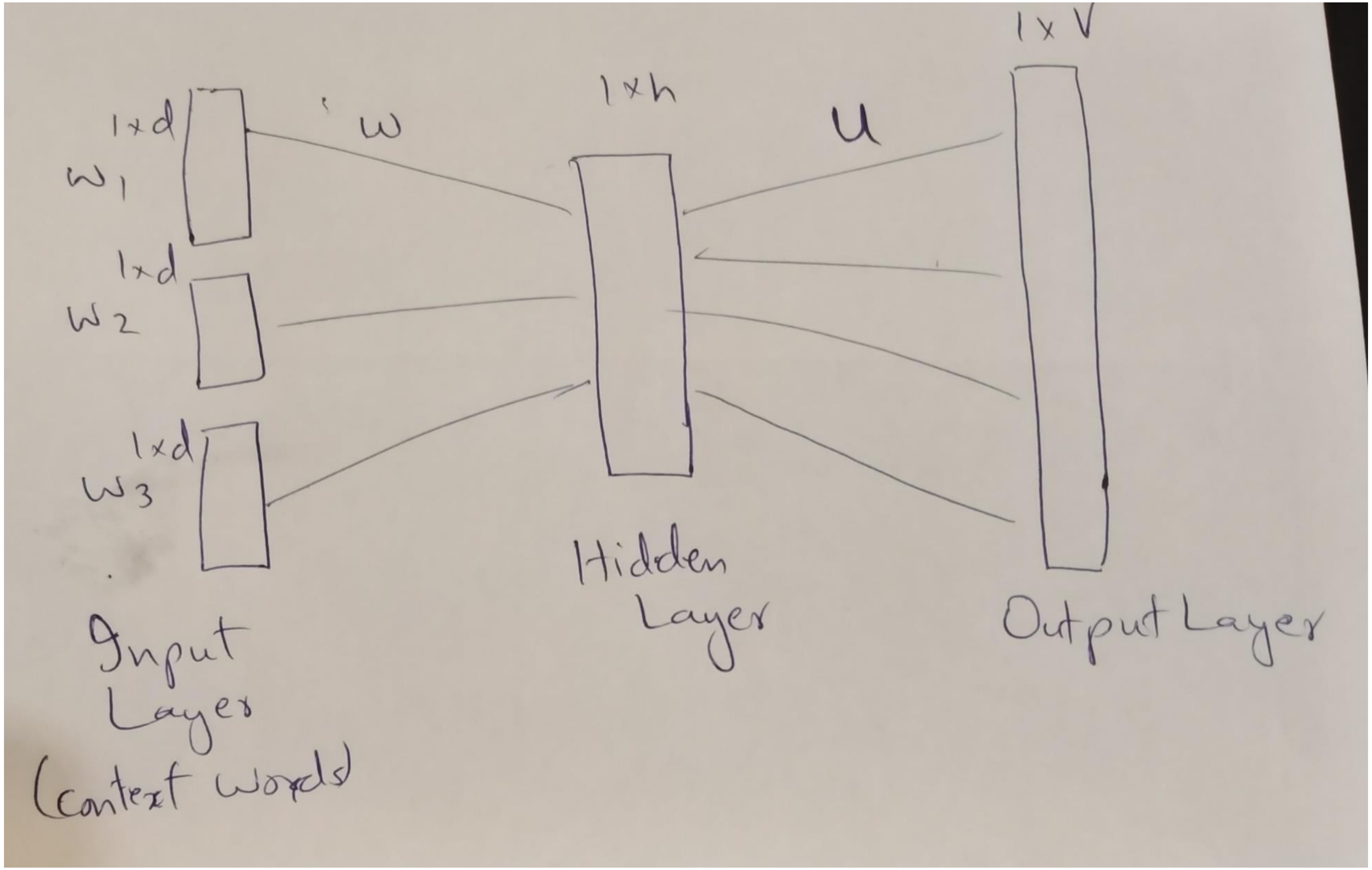
words / one-hot vectors

$$\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(3)}, \boldsymbol{x}^{(4)}$$

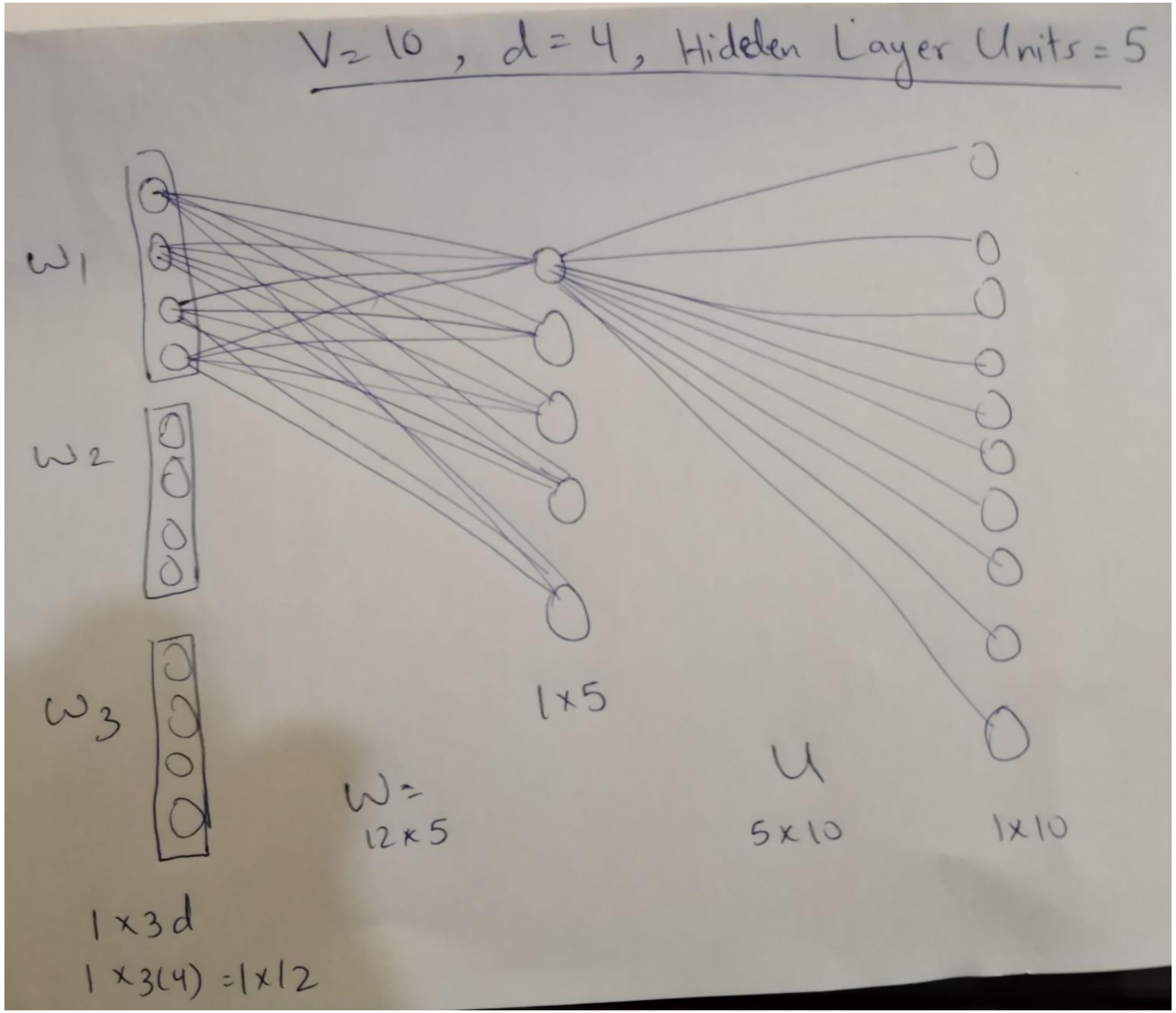


Slide Credit: Abigail See

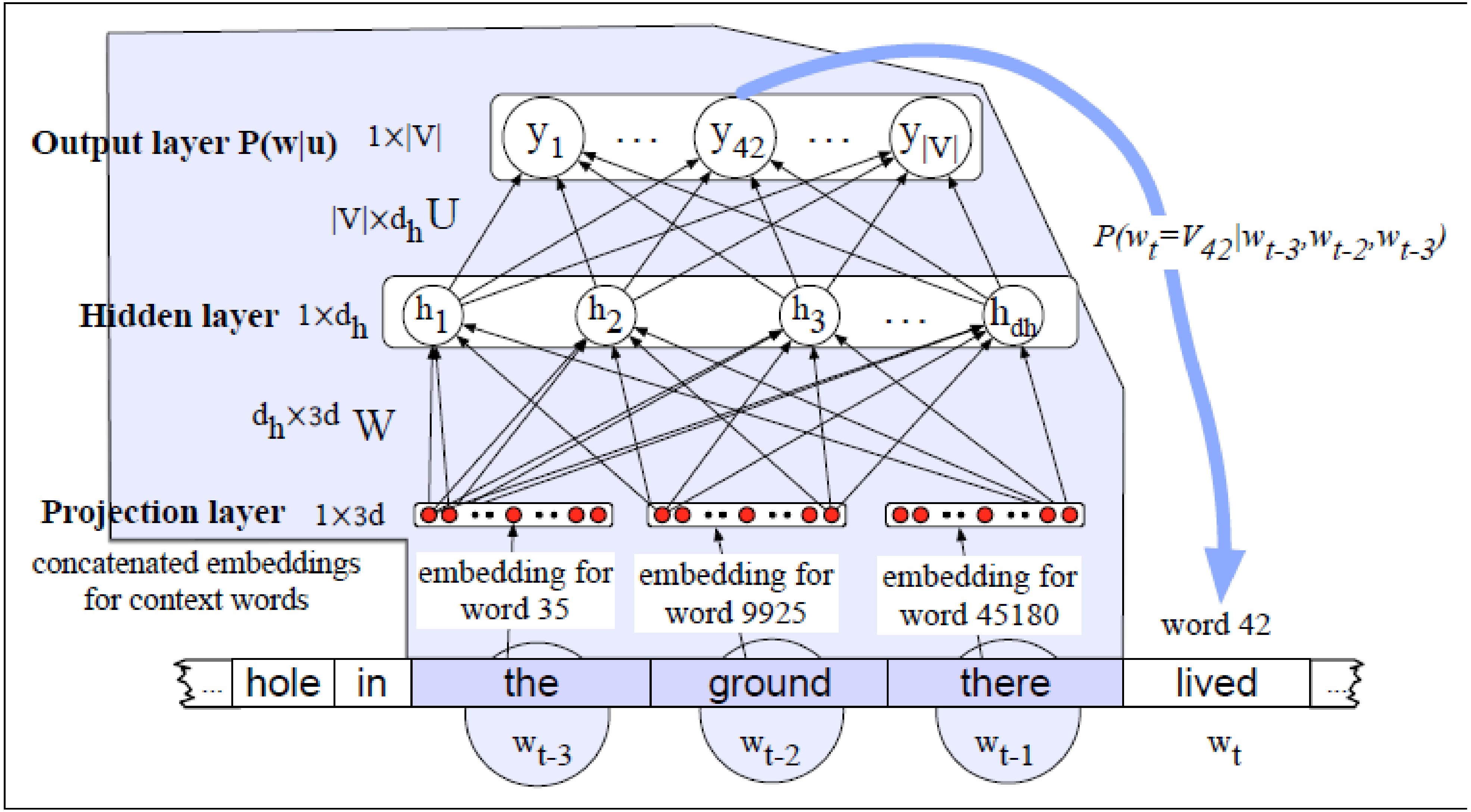
High Level Architecture of Neural LM



High Level Architecture of Neural LM

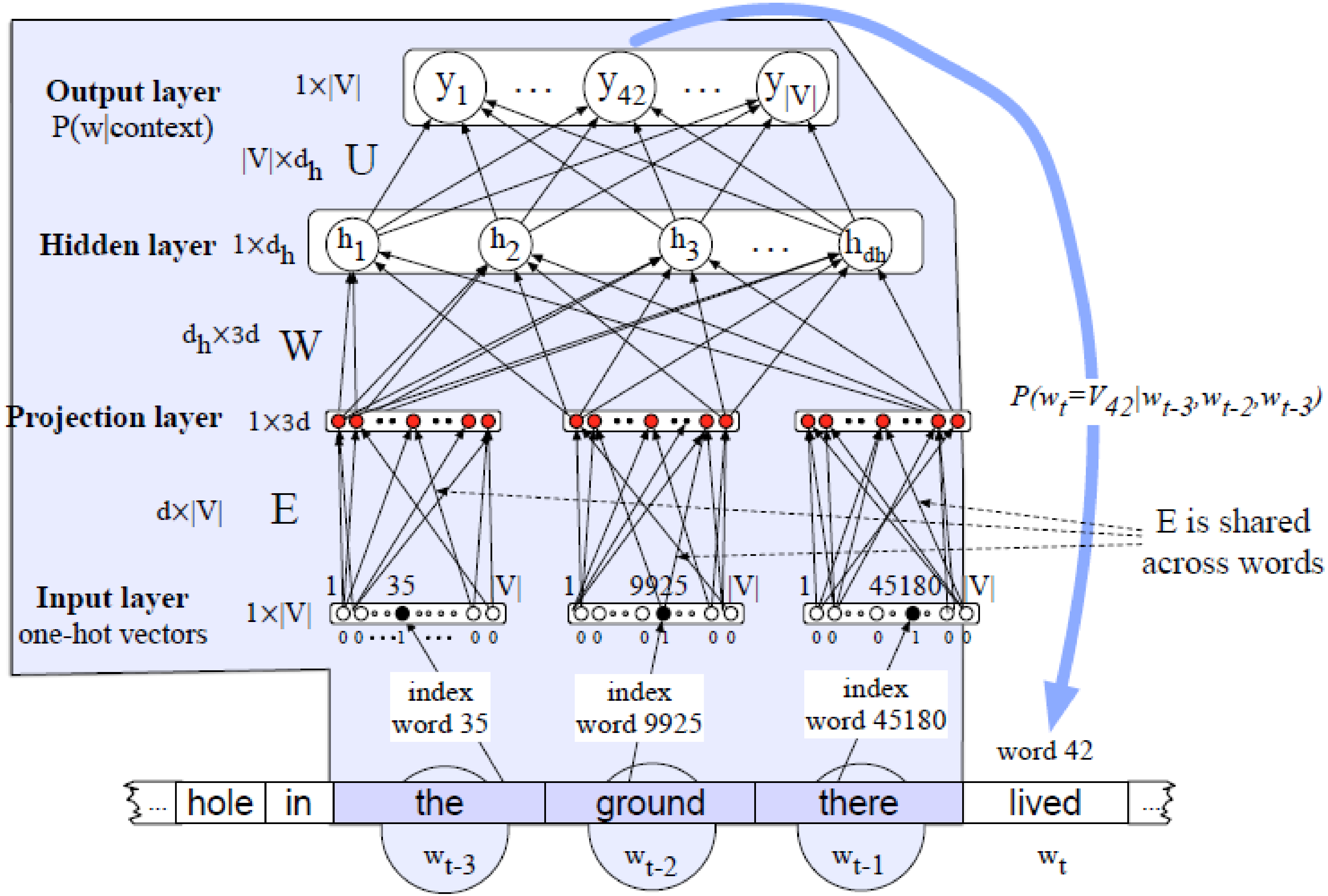


Feedforward neural language model moving through a text



Embedding as input for LM

- I have to make sure when I get home to feed the cat.
- I have to make sure when I get home to feed the __________
- Word “dog” will also have high probability to appear here even if it is not seen in text with same context.



Output layer
P(w|context)
1×|V|
y1 … y42 … y|V|
|V|×dh U
Hidden layer 1×dh
h1 h2 h3 … hdh
dh×3d W
Projection layer 1×3d
d×|V| E
E is shared across words
Input layer one-hot vectors
1×|V|
1 35 |V|
1 9925 |V|
1 45180 |V|
index word 35
index word 9925
index word 45180
word 42
P(wt=V42|wt-3,wt-2,wt-3)
… hole in the ground there lived …
wt-3 wt-2 wt-1 wt

Forward Pass

$$
\begin{aligned}
e &= (Ex_1, Ex_2, \ldots, Ex) \\
h &= \sigma(We + b) \\
z &= Uh \\
y &= \text{softmax}(z)
\end{aligned}
$$

Forward Pass

$$
\begin{aligned}
e &= (Ex_1, Ex_2, ..., Ex) \\
h &= \sigma(We + b) \\
z &= Uh \\
y &= \text{softmax}(z)
\end{aligned}
$$

1. **Select three embeddings from E:** the projection layer for input w will be $Ex_i = e_i$, the embedding for word i.
2. **Multiply by W:** We now multiply by weight matrix W (and add b) and pass through activation function to get the hidden layer h.
3. **Multiply by U:** h is now multiplied by U, U is also weight matrix
4. **Apply softmax:** After the softmax, each node i in the output layer estimates the probability $P(w_t = i | w_{t-1}, w_{t-2}, w_{t-3})$

Training of Model

Loss Function $$L = -\log p(w_t | w_{t-1}, \ldots, w_{t-n+1})$$

The gradient for this loss is

$$\theta_{t+1} = \theta_t - \eta \frac{\partial - \log p(w_t | w_{t-1}, \ldots, w_{t-n+1})}{\partial \theta}$$

Training the neural language model

This gradient can be computed in any standard neural network framework which will then backpropagate through U,W, b, E.

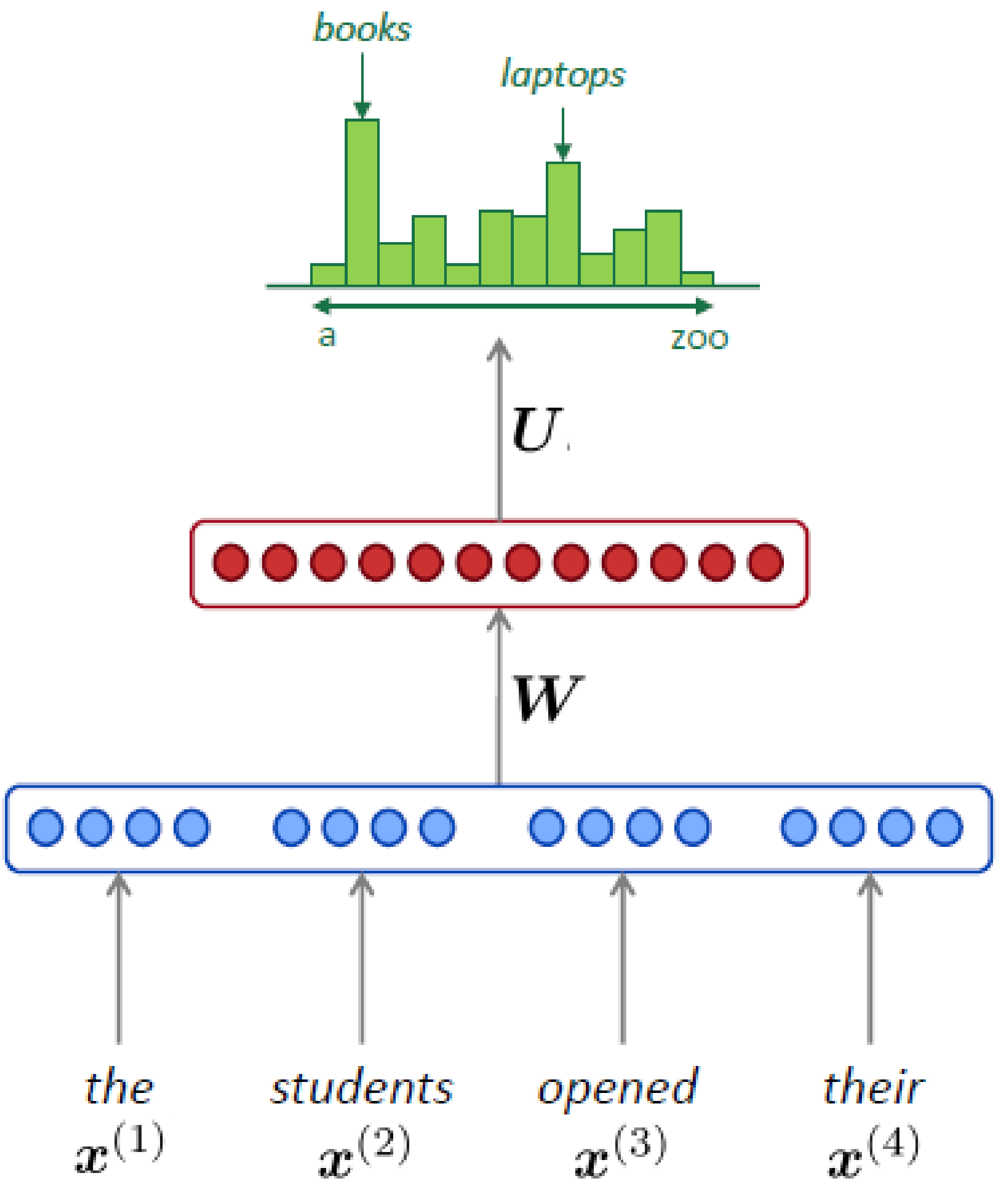
Training the parameters to minimize loss will result both in an algorithm for language modeling (a word predictor) but also a new set of embeddings E that can be used as word representations for other tasks.

A fixed-window neural Language Model

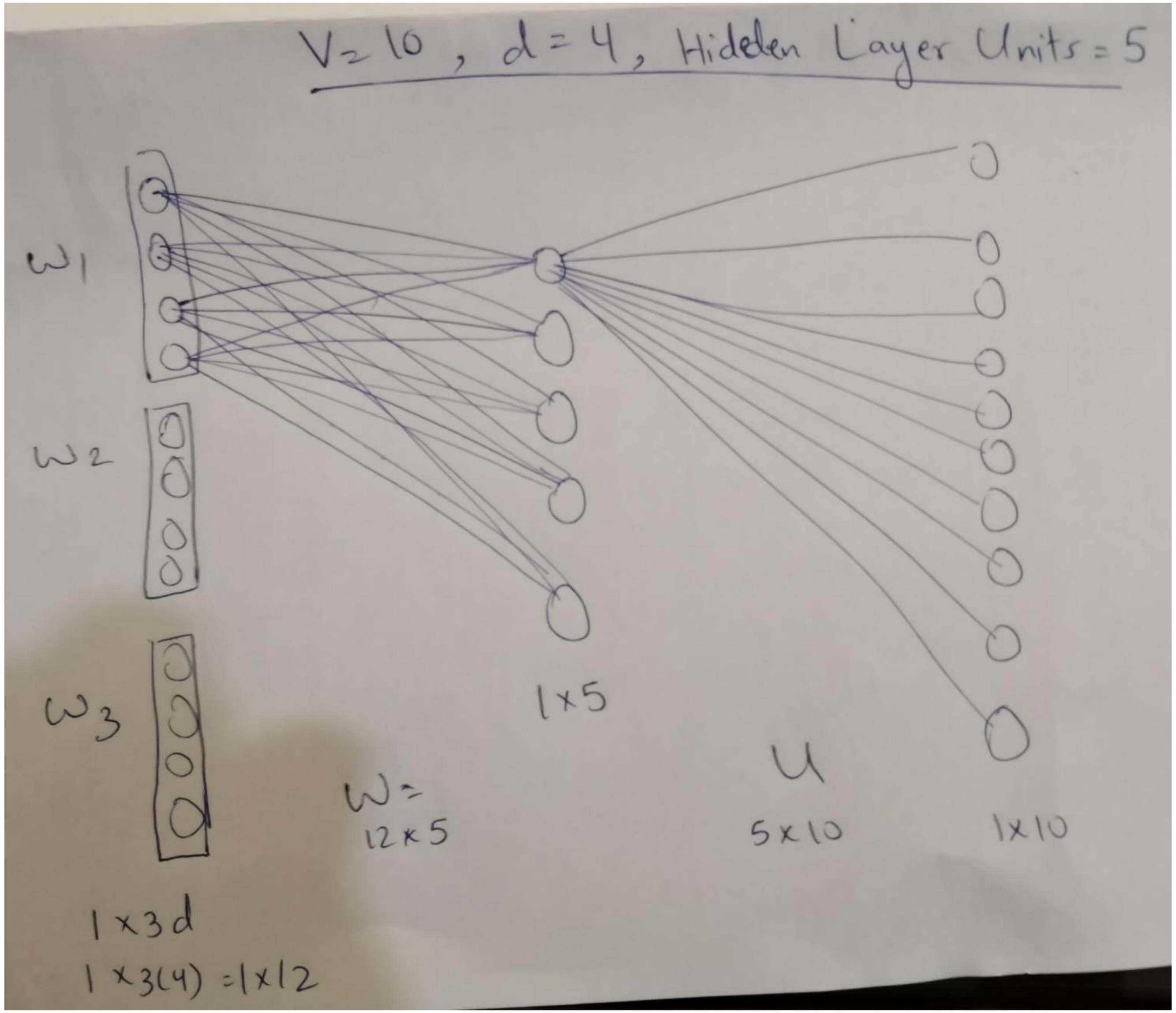
Improvements over *n*-gram LM:
- No sparsity problem
- Don't need to store all observed *n*-grams

Remaining **problems**:
- Fixed window is too small
- Enlarging window enlarges $\boldsymbol{W}$
- Window can never be large enough!
- $\boldsymbol{x}^{(1)}$ and $\boldsymbol{x}^{(2)}$ are multiplied by completely different weights in $\boldsymbol{W}$. No symmetry in how the inputs are processed.

We need a neural architecture that can process *any length input*



High Level Architecture of Neural LM
(Different weights for each word are used in Matrix W in first layer)



Reading

- Chapter 7, Speech and Language Processing. Daniel Jurafsky & James H. Martin. Third edition

  https://web.stanford.edu/~jurafsky/slp3/7.pdf