AI Bootcamp

NLP & LLMs

DENSE WORD EMBEDDINGS USING Word2Vec

Dr Usman Zia

Asst Professor

SINES, NUST

usman.zia@sines.nust.edu.pk

One Hot Encoding

- Each word is represented as a binary vector where each element in the vector corresponds to one of the word,
- and is either 0 or 1 depending on whether the data point belongs to that category or not.

- **Document1 — We are learning Natural Language Processing**

	0	1	2	3	4	5	6	7	8	9
We	1	0	0	0	0	0	0	0	0	0
are	0	1	0	0	0	0	0	0	0	0
learning	0	0	1	0	0	0	0	0	0	0
Natural	0	0	0	1	0	0	0	0	0	0
Language	0	0	0	0	1	0	0	0	0	0
Processing	0	0	0	0	0	1	0	0	0	0

Sparse vs. Dense Word Embeddings

Sparse:

- Very high-dimensional vectors
- Lots of empty (zero-valued) cells
- Out of Vocabulary (OOV) problem
- No capturing of semantic meaning

Dense:

- Lower-dimensional (50-1000 cells) vectors
- Most cells have non-zero values

Which embedding type is better for NLP tasks?

Dense vectors!

Why?

- Easier to include as **features** in machine learning systems
  - Classifiers have to learn ~100 weights instead of ~50,000
- Fewer **parameters** → lower chance of overfitting
  - May generalize better to new data
- Better at capturing **synonymy**
  - Words are not distinct dimensions; instead, dimensions correspond to meaning components

Methods for getting short dense vectors

- **Singular Value Decomposition (SVD)**
- **"Neural Language Model" – inspired by predictive models**

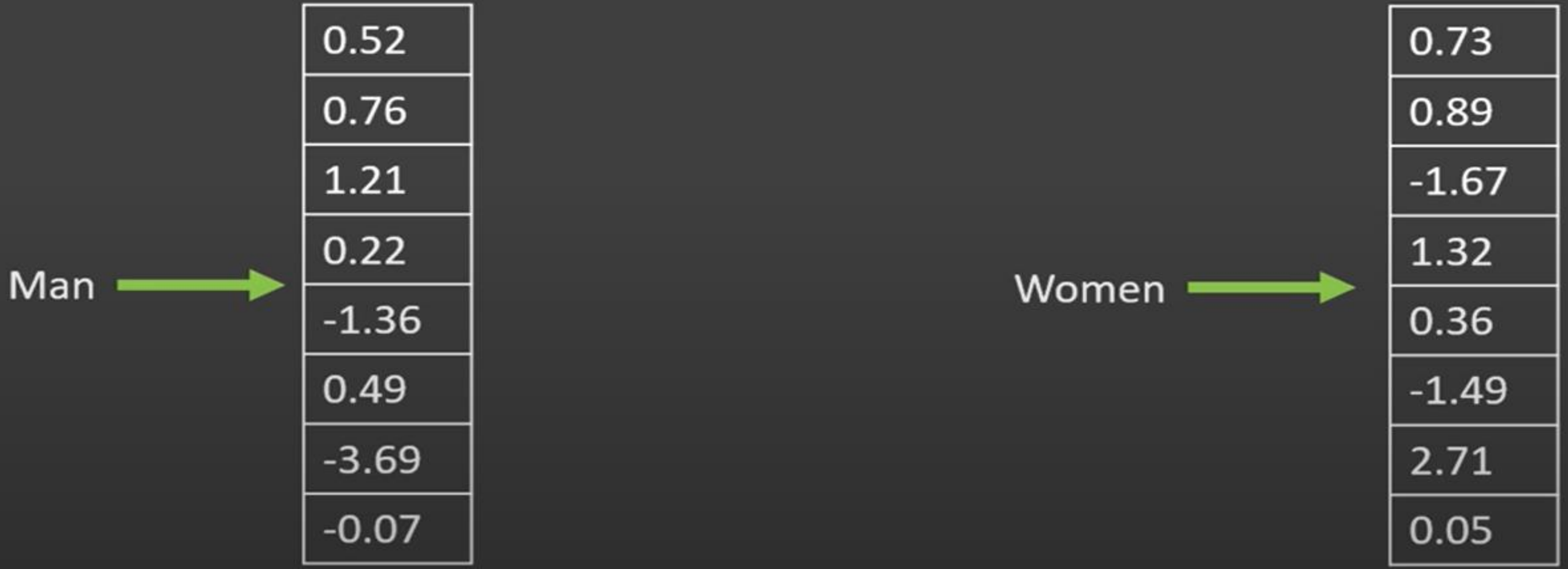
WORD2VEC

Word2Vec Intuition

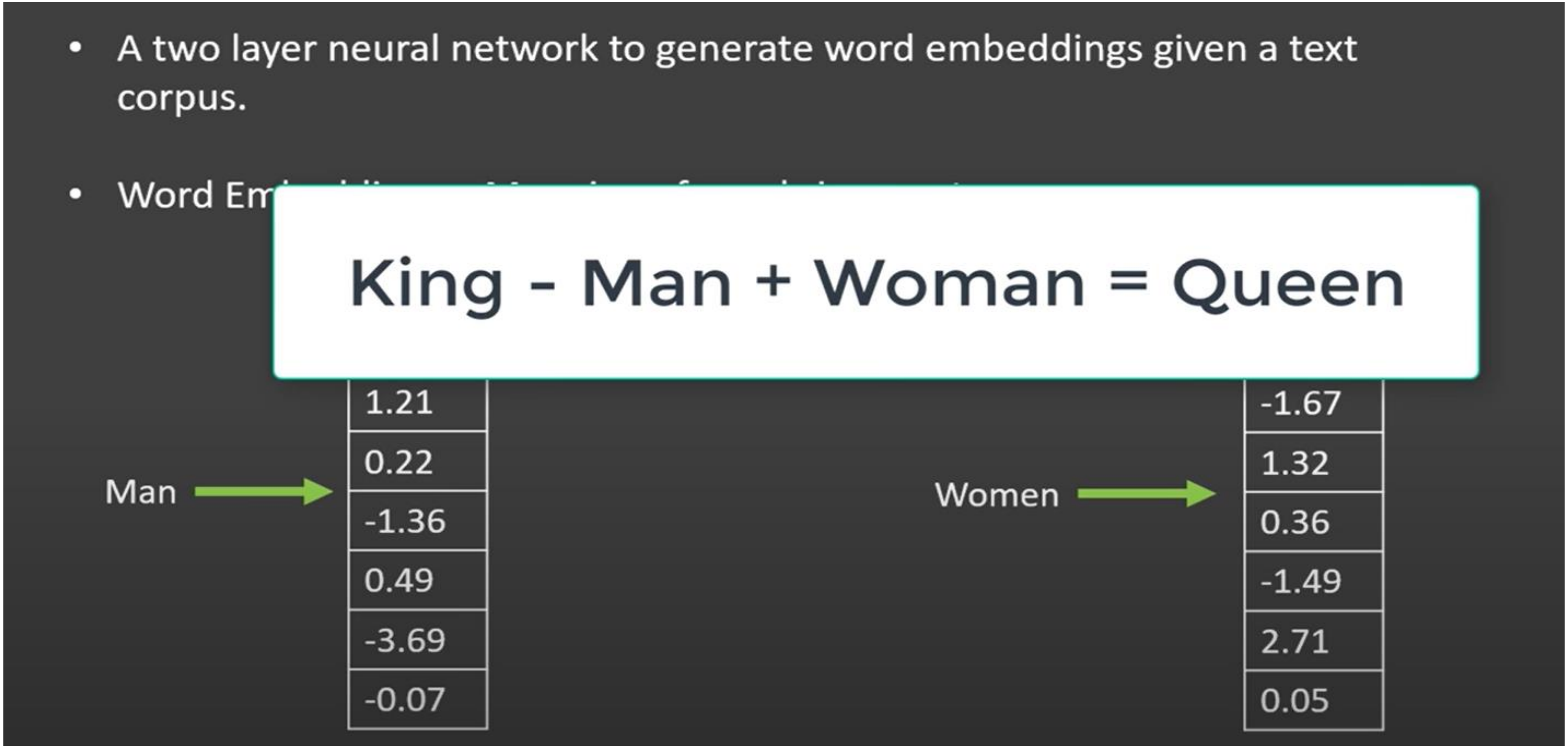
- Instead of counting how often each word occurs near each context word, train a classifier on a **binary prediction task**
  - Is word *w* likely to occur near context word *c*?
- The twist: **We don’t actually care about the classifier!**
- We use the **learned classifier weights** from this prediction task as our word embeddings

Word2Vec Explained

- A two layer neural network to generate word embeddings given a text corpus.
- Word Embeddings – Mapping of words in a vector space.



Word2Vec Explained



Working of word2Vec

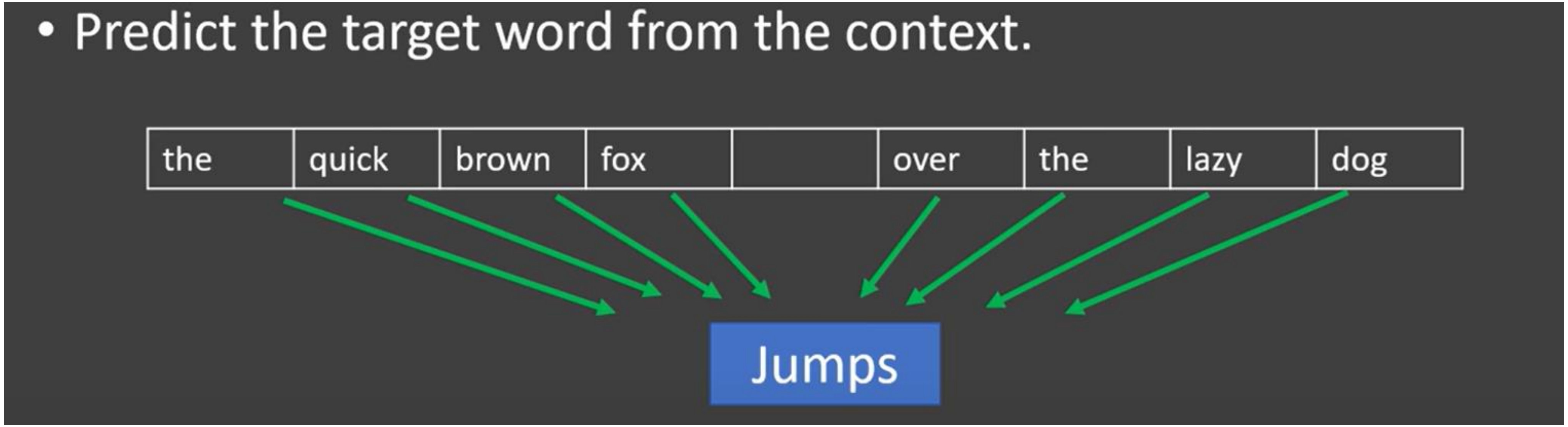
- The word2vec objective function causes the words that occur in similar contexts to have similar embeddings.

Example: The **kid** said he would grow up to be superman.

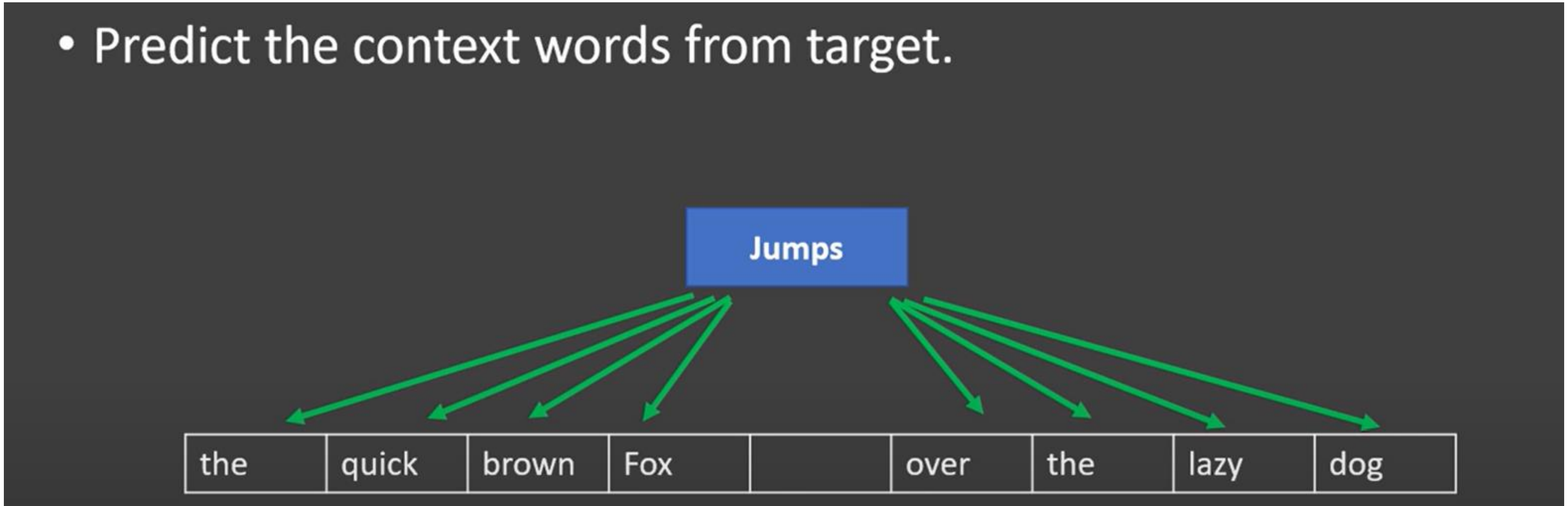
The **child** said he would grow up to be superman.

The words kid and child will have similar word vectors due to a similar context.

Continuous Bag of Words (CBOW)



Skip Gram



CBOW - Working

Hope can set you free.

CBOW - Working

Hope can set you free.

CBOW - Working

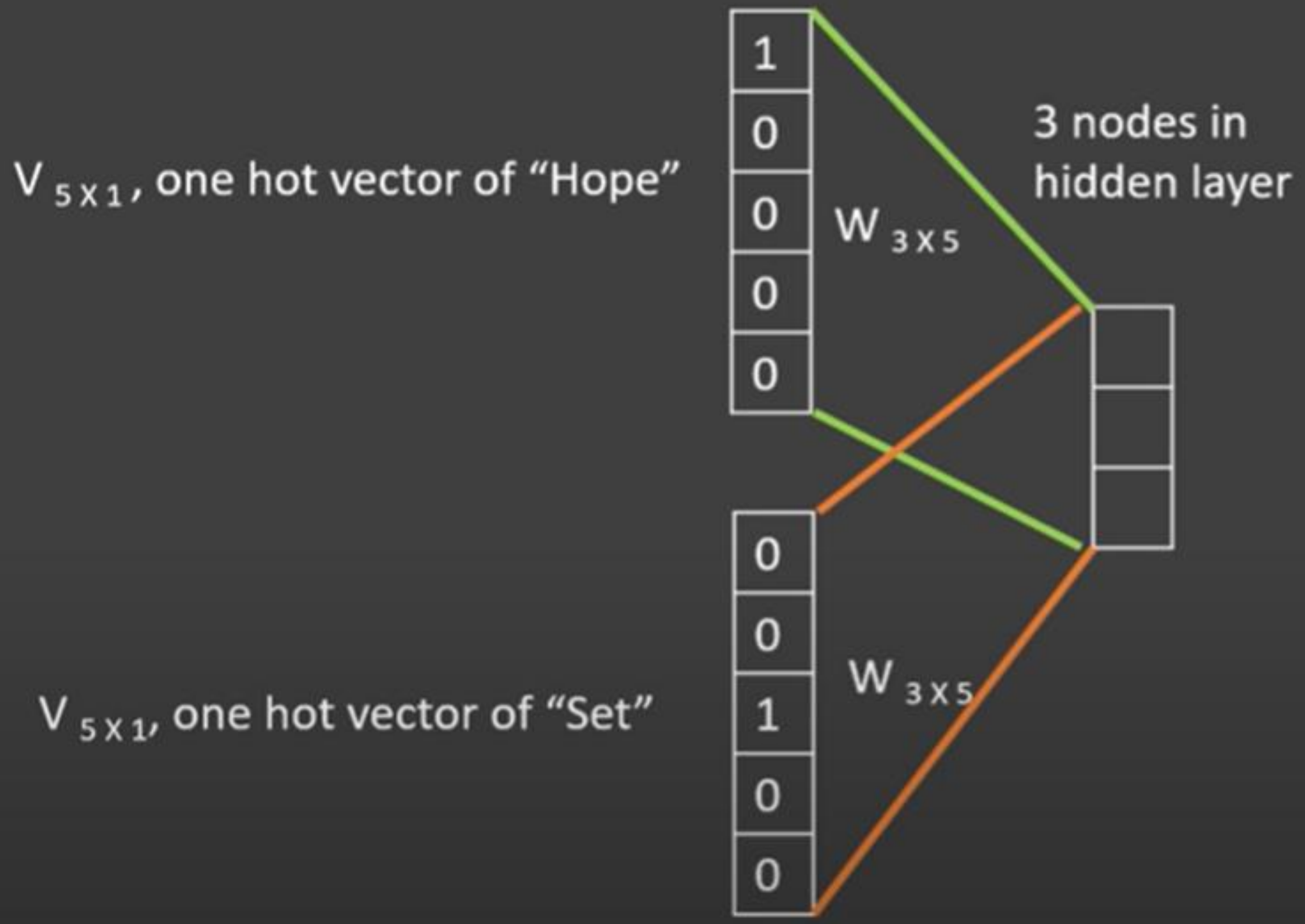
$\boxed{\text{Hope can set}}$ you free.

$V_{5 \times 1}$, one hot vector of "Hope" $\begin{bmatrix}1\\0\\0\\0\\0\end{bmatrix}$

$V_{5 \times 1}$, one hot vector of "Set" $\begin{bmatrix}0\\0\\1\\0\\0\end{bmatrix}$

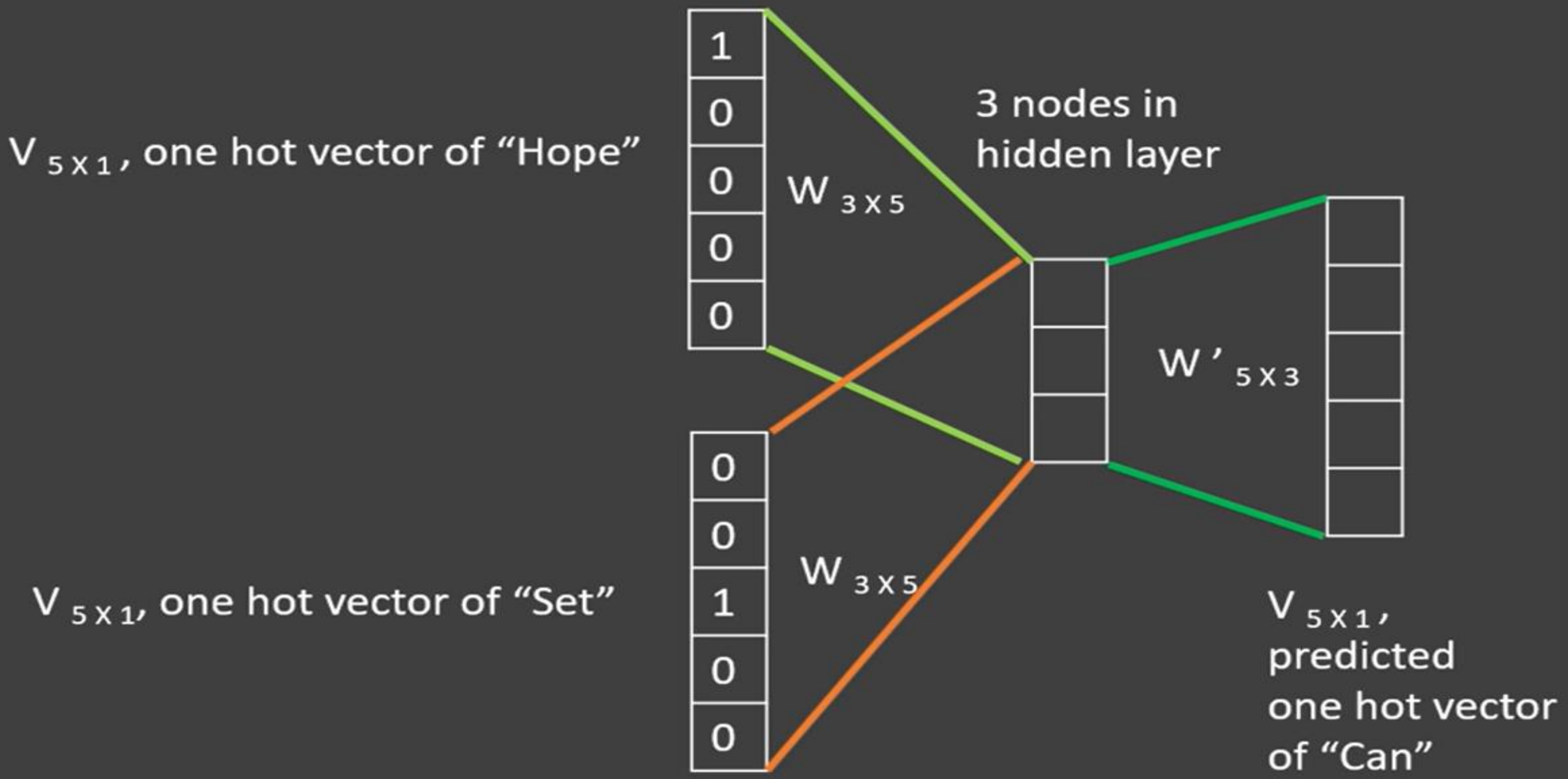
CBOW - Working

Hope can set you free.



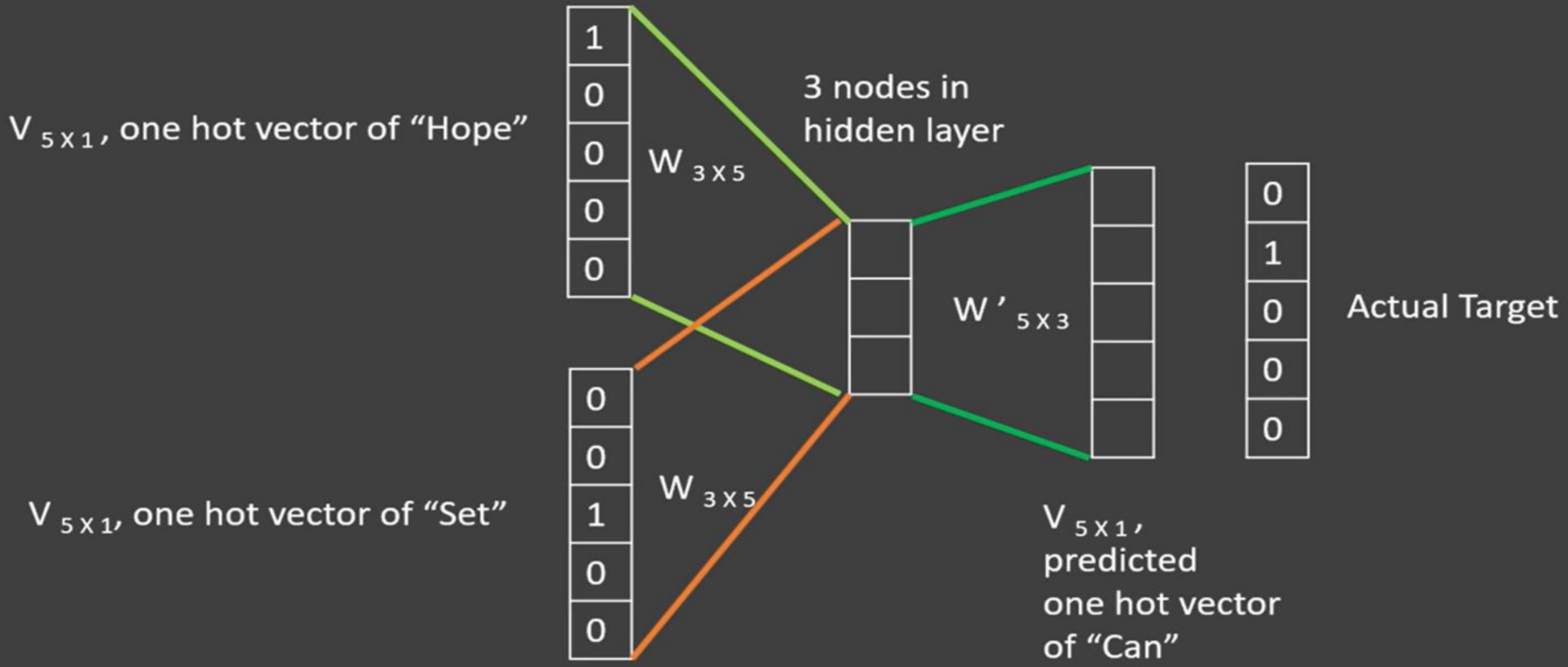
CBOW - Working

Hope can set you free.



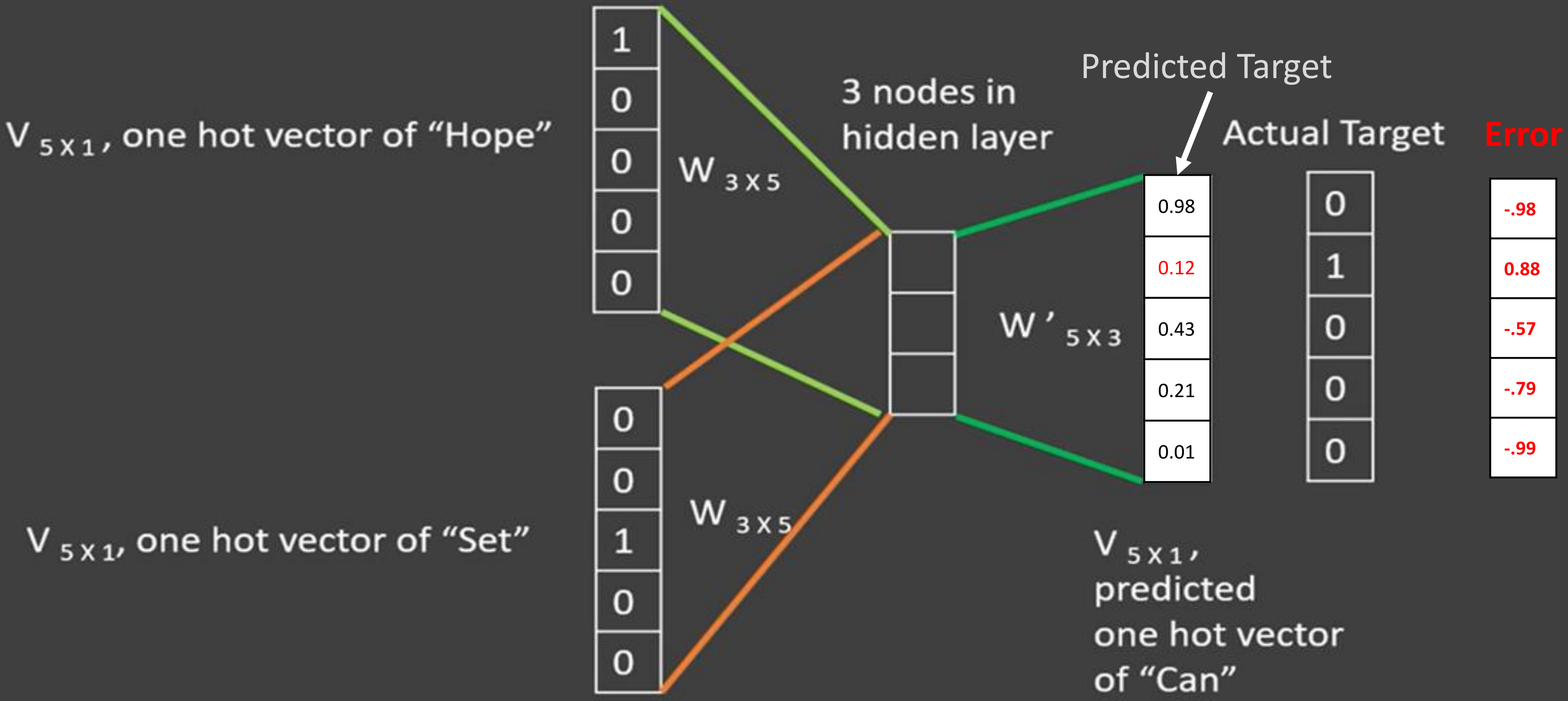
CBOW - Working

Hope can set you free.

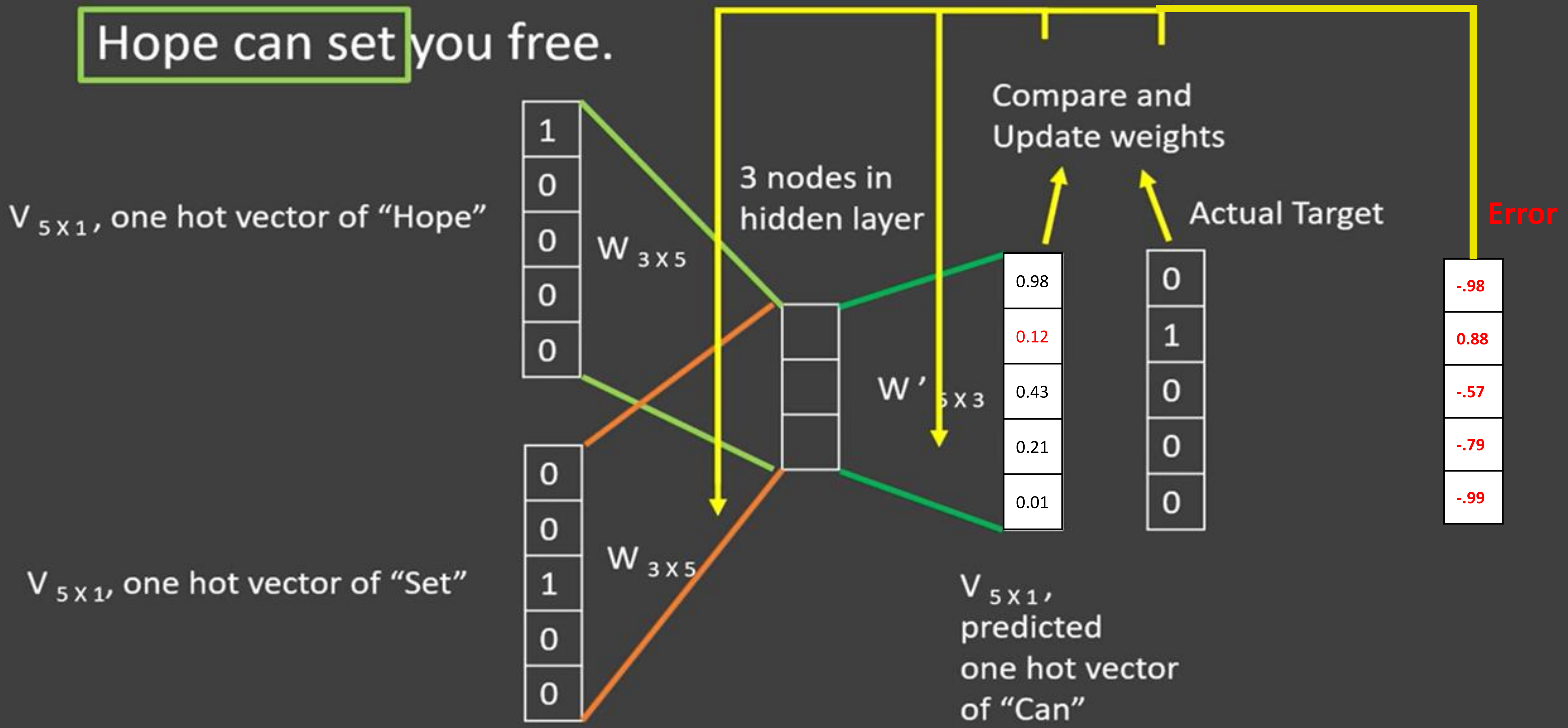


CBOW - Working

Hope can set you free.

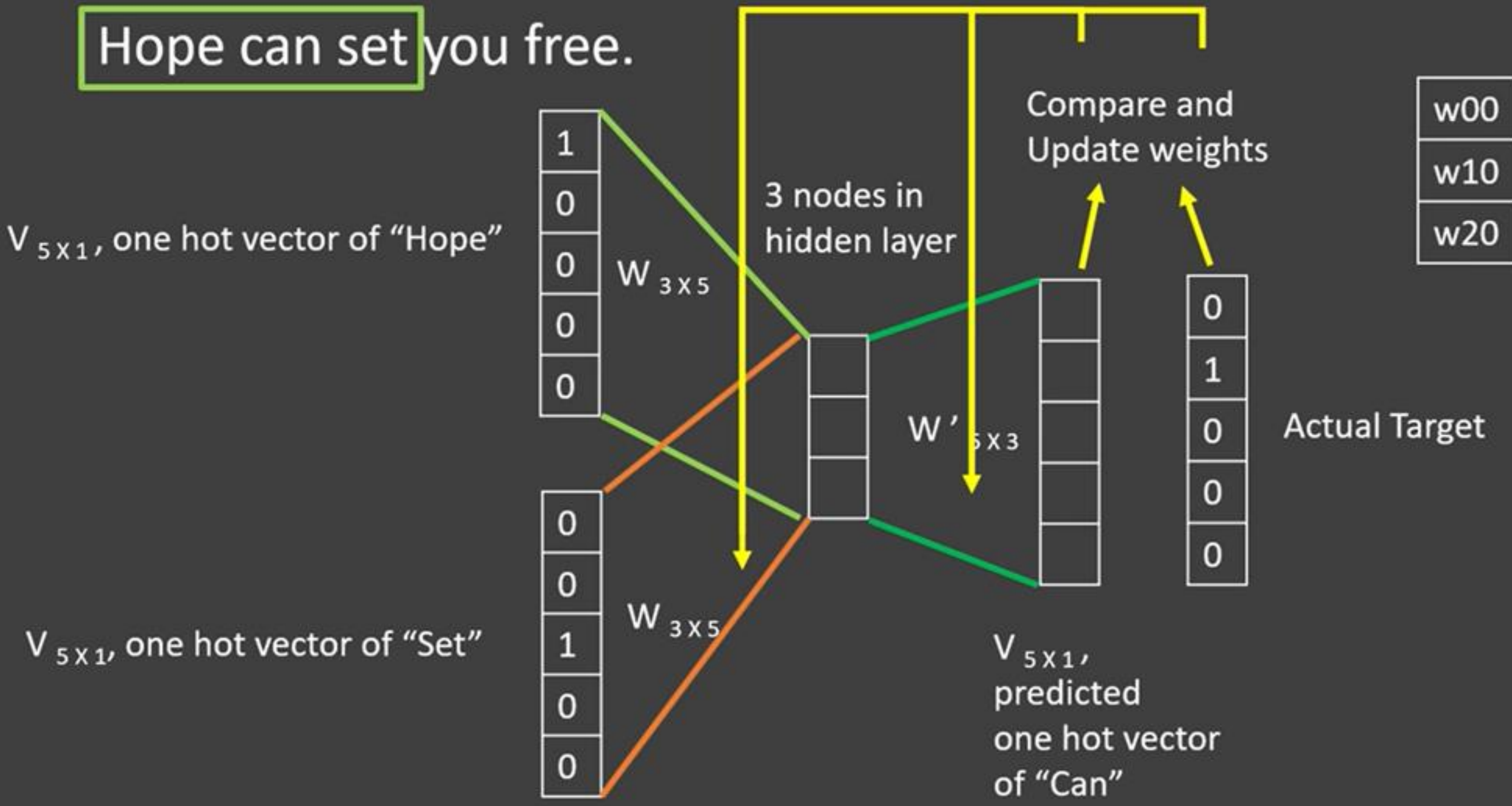


CBOW - Working



CBOW - Working

Hope can set you free.

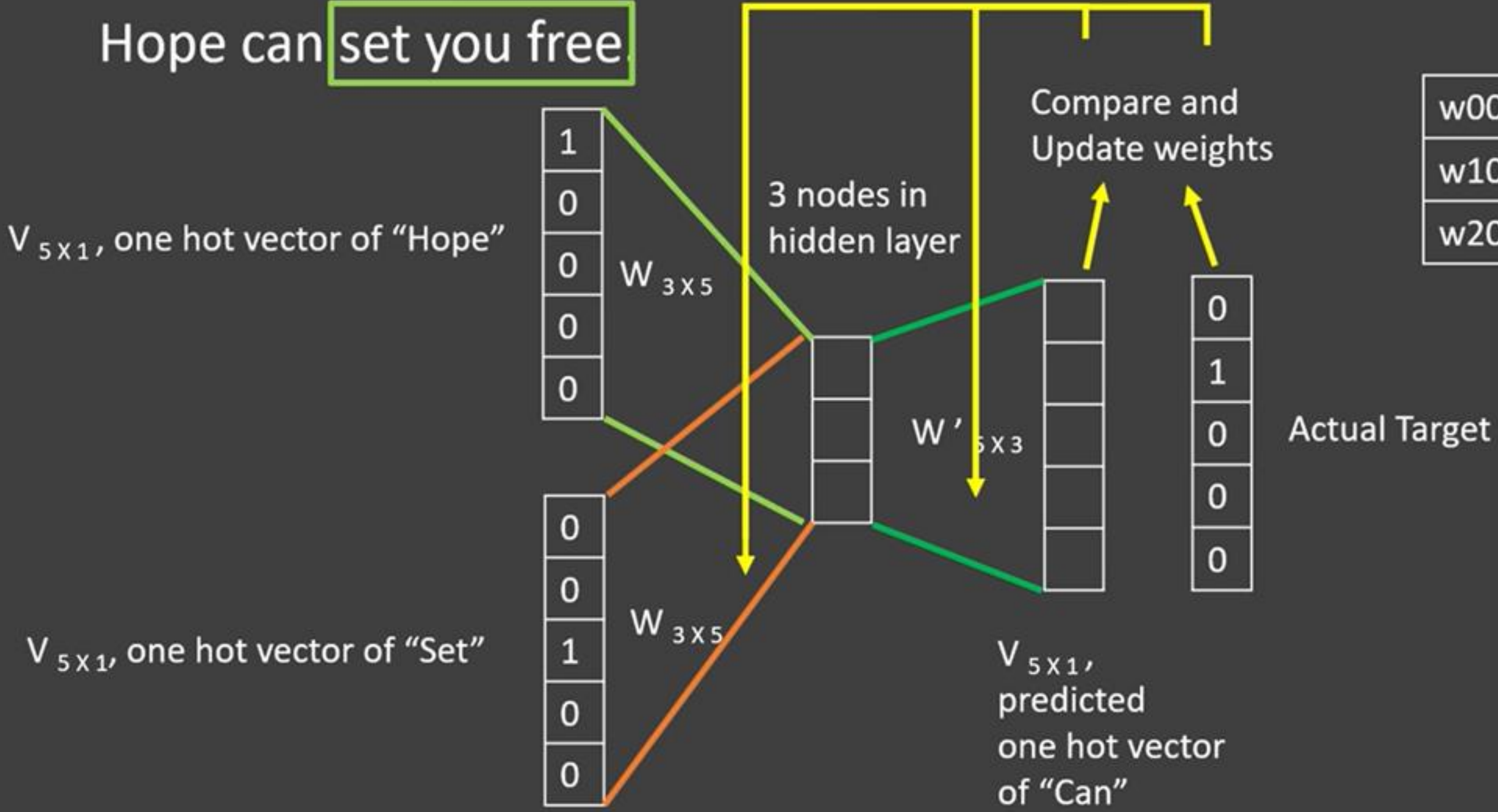


w00	w01	w02	w03	w04
w10	w11	w12	w13	w14
w20	w21	w22	w23	w24

$W_{3 \times 5}$

CBOW - Working

Hope can set you free

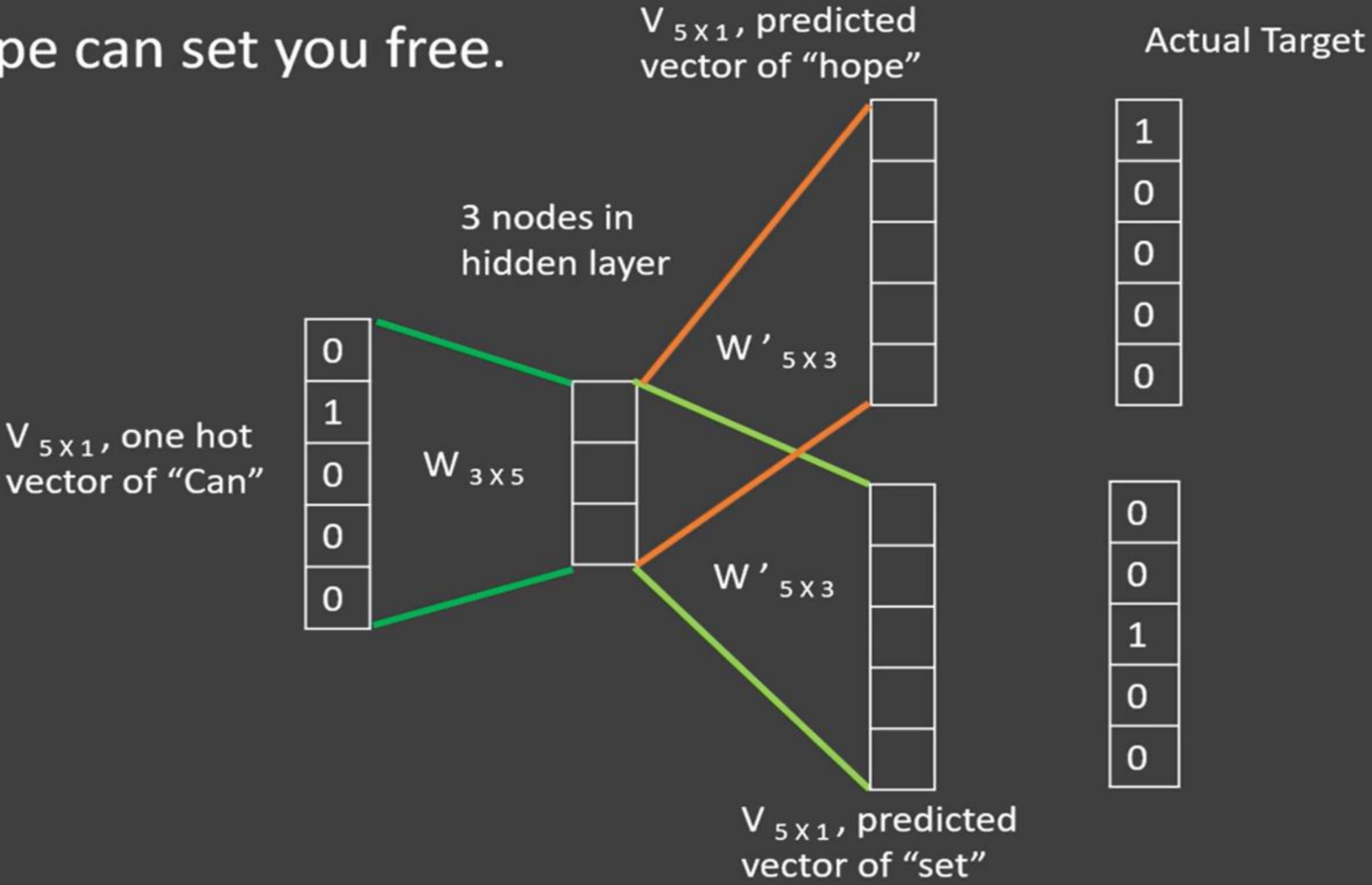


w00	w01	w02	w03	w04
w10	w11	w12	w13	w14
w20	w21	w22	w23	w24

$W_{3 \times 5}$

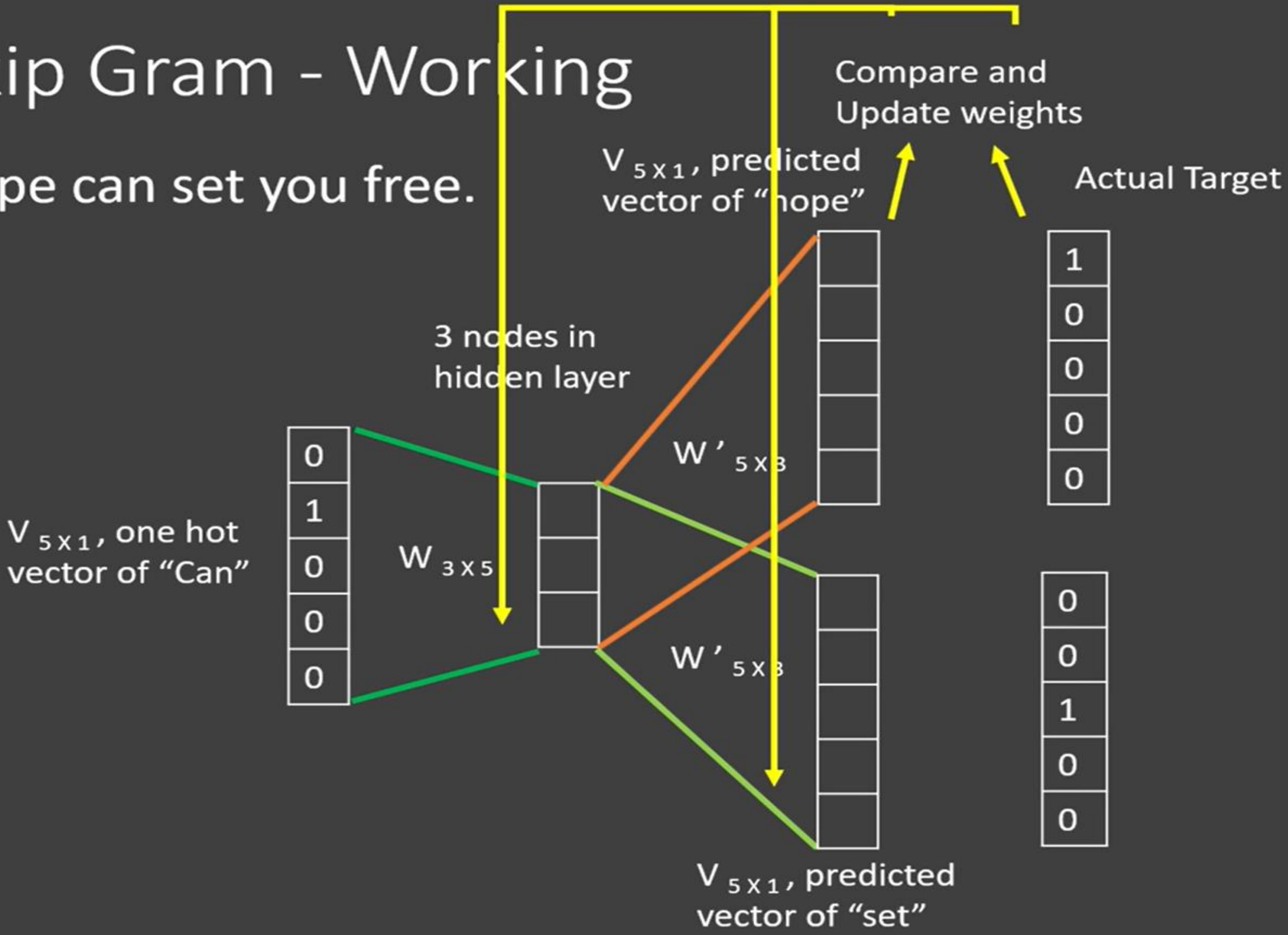
Skip Gram - Working

Hope can set you free.



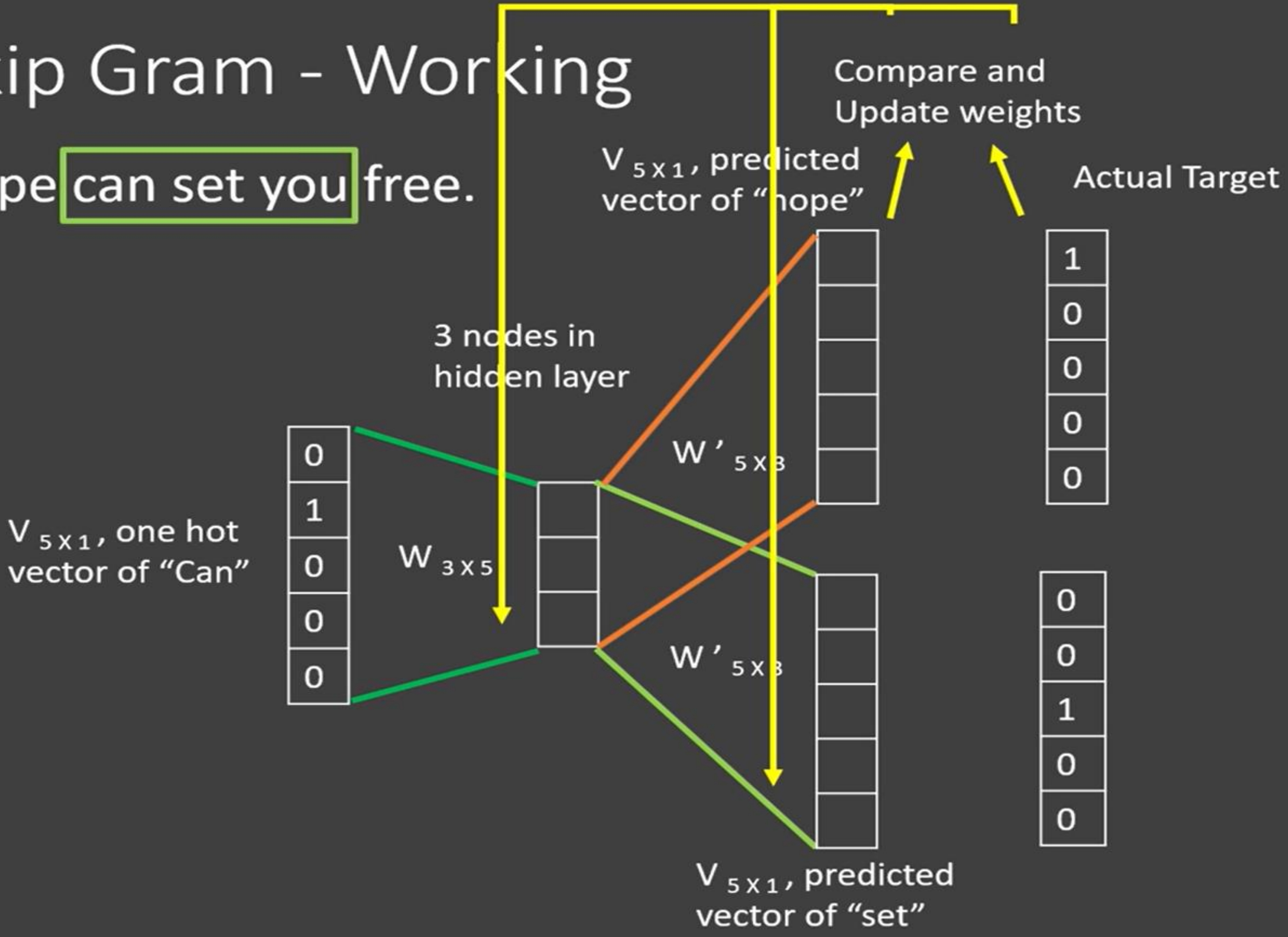
Skip Gram - Working

Hope can set you free.



Skip Gram - Working

Hope can set you free.



Getting word embeddings

Weights after training

$W_{3 \times 5}$

w00	w01	w02	w03	w04
w10	w11	w12	w13	w14
w20	w21	w22	w23	w24

One Hot vector of words $V_{5 \times 1}$

$$\underset{\text{Hope}}{\begin{bmatrix}1\\0\\0\\0\\0\end{bmatrix}} \quad \underset{\text{can}}{\begin{bmatrix}0\\1\\0\\0\\0\end{bmatrix}} \quad \underset{\text{set}}{\begin{bmatrix}0\\0\\1\\0\\0\end{bmatrix}} \quad \underset{\text{you}}{\begin{bmatrix}0\\0\\0\\1\\0\end{bmatrix}} \quad \underset{\text{free}}{\begin{bmatrix}0\\0\\0\\0\\1\end{bmatrix}}$$

Word Vector for hope $= W_{3 \times 5} \times V_{5 \times 1}$

$$\begin{bmatrix} w00 & w01 & w02 & w03 & w04 \\ w10 & w11 & w12 & w13 & w14 \\ w20 & w21 & w22 & w23 & w24 \end{bmatrix} \times \begin{bmatrix}1\\0\\0\\0\\0\end{bmatrix} = \begin{bmatrix} w00 \\ w10 \\ w20 \end{bmatrix}$$

$V_{3 \times 1}$

Word Vector for Hope

Improving the accuracy

- Choice of Model architecture (CBOW / Skipgram)
  - Large Corpus, higher dimensions, slower– Skipgram
  - Small Corpus, Faster - CBOW
- Increasing the training dataset.
- Increasing the vector dimensions
- Increasing the windows size.

Why Word2Vec?

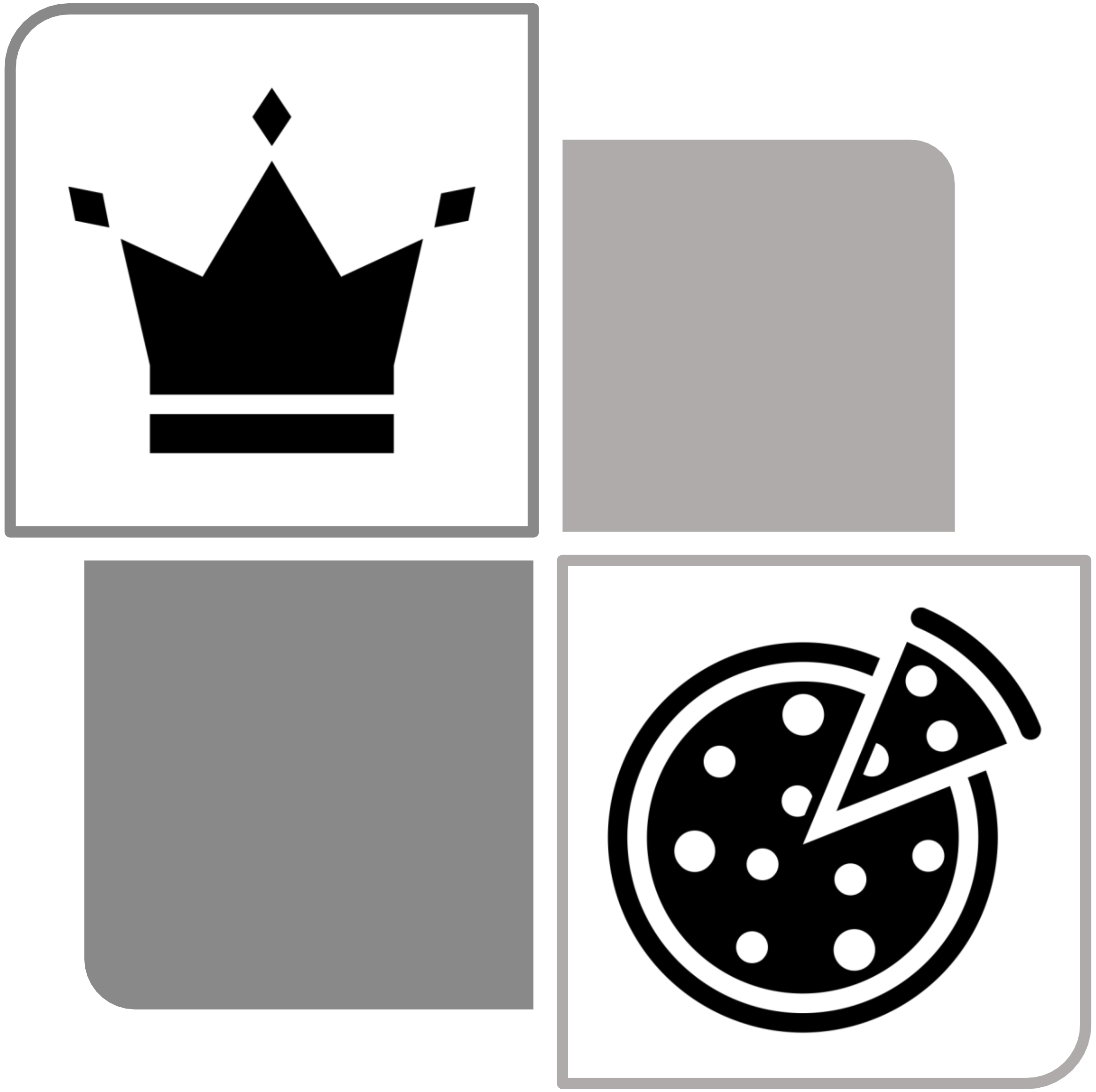
- Preserves relationship between words.
- Deals with addition of new words in the vocabulary.
- Better results in lots of deep learning applications.

Context window size can impact performance.

- Because of this, context window size is often tuned on a development set
- Larger window size → more required computations (important to consider when using very large datasets!)

Semantic Properties of Embeddings

- Major advantage of dense word embeddings: Ability to capture elements of meaning
- Context window size impacts what type of meaning is captured
  - Shorter context window → more **syntactic representations**
    - Information is from immediately nearby words
    - Most similar words tend to be semantically similar words with the same parts of speech
  - Longer context window → more **topical representations**
    - Information can come from longer-distance dependencies
    - Most similar words tend to be topically related, but not necessarily similar (e.g., waiter and menu, rather than spoon and fork)



Analogy

- Word embeddings can also capture **relational meanings**
- This is done by computing the offsets between values in the same columns for different vectors
- Famous examples (Mikolov et al., 2013; Levy and Goldberg, 2014):
  - king - man + woman = queen
  - Paris - France + Italy = Rome

Bias and Embeddings

The good:

- Word embeddings automatically learn semantic properties and relationships from text

The bad:

- They also end up reproducing the implicit biases and stereotypes that are latent in the text

Bias and Embeddings

- Recall: king - man + woman = queen
- Word embeddings trained on news corpora also produce:
  - man - computer programmer + woman = homemaker
  - doctor - father + mother = nurse
- Issues like these are problematic in real- world applications!
  - E.g., algorithms may automatically assign lower scores to resumes containing common female names when ranking them for technical positions

Bias and Embeddings

- Embeddings also encode **implicit associations**
- Many implicit associations are harmless, and even useful for sentence processing
  - flowers → pleasant
  - roaches → unpleasant
- However, other implicit associations are very harmful
  - Caliskan et al. (2017) identified the following known, harmful implicit associations in GloVe embeddings:
    - African-American names were more closely associated with unpleasantness than European-American names
    - Male names were more closely associated with mathematics than female names
    - Female names were more closely associated with the arts than male names
    - Names common among older adults were more closely associated with unpleasantness than those common among younger adults

Bias and Embeddings

- Thus, learning word representations poses an increasingly important ethical dilemma!
- Recent research has begun examining ways to **debias** word embeddings by:
  - Developing transformations of embedding spaces that remove gender stereotypes but preserves definitional gender
  - Changing training procedures to eliminate these issues before they arise
- Although these methods reduce bias, they do not eliminate it

Evaluating Vector Models

Extrinsic Evaluation

- Add the vectors as features in a downstream NLP task, and see whether and how this changes performance relative to a baseline model
- Most important evaluation metric for word embeddings!
  - Word embeddings are rarely needed in isolation
  - They are almost solely used to boost performance in downstream tasks

Intrinsic Evaluation

- Performance at predicting word similarity

Evaluating Similarity Performance

- Compute the **cosine similarity** between vectors for pairs of words
- Compute the **correlation** between those similarity scores and word similarity ratings for the same pairs of words manually assigned by humans
- Corpora for doing this:
  - WordSim-353
  - SimLex-999
  - TOEFL Dataset
    - *Levied* is closest in meaning to: (a) imposed, (b) believed, (c) requested, (d) correlated

A (Very Brief) Overview of Other Embedding Methods

GloVe

ELMo

BERT

Global Vectors for Word Representation (GloVe)

- While Word2Vec is a **predictive model** (it learns to predict whether words belong in a target word’s context), GloVe is a **count-based model** (it’s basically a fancy co-occurrence matrix)
- In a nutshell, GloVe embeddings are constructed by:
  - Building a huge word x context co-occurrence matrix
  - Performing dimensionality reduction on the matrix to reduce it to a more manageable size
- Still non-contextual

Embeddings from Language Models (ELMo)

- **Contextual** (predicts different vectors for different word senses)
- Does this by concatenating information from multiple layers of a **bidirectional** neural language model
- Accepts character inputs instead of words, which enables the model to predict embeddings for out-of-vocabulary words
- Predicts an embedding for a target word given its context

Bidirectional Encoder Representations from Transformers (BERT)

- Also contextual
- Learns embeddings for **subwords** (more than a character, but less than a full word)
- This allows the model to also predict embeddings for out-of-vocabulary words
- Uses a bidirectional neural language model to do this, similar to ELMo
  - The specific type of neural language model differs (**Transformer** rather than **LSTM**)

Which embeddings are best?

- It depends on your data!
- In general, Word2Vec and GloVe produce similar embeddings
- Word2Vec → slower to train but less memory intensive
- GloVe → faster to train but more memory intensive
- Word2Vec and Glove both produce context-independent embeddings
- ELMo and BERT produce context-dependent embeddings
- Both can also predict embeddings for new words
- BERT (or variants thereof) is the current state of the art
- ELMo may be better in cases with lots of obscure words that aren’t easily chunked into subwords

Summary: Word Embeddings

- **Dense vectors** are generally better for NLP tasks
- **Word2Vec**, **GloVe**, **ELMo**, and **BERT** are all examples of dense word embeddings
- Word2Vec (specifically, the **skip-gram** variant) learns a classifier that predicts whether a word is in the context of a target word
- The weights from the hidden layer of this classifier are the learned **word embeddings**
- These embeddings are learned using a formula that maximizes the similarity between vectors for words that occur in the same context
- Word embeddings capture **semantic properties** but they also capture **harmful stereotypes …..** coming up with good debiasing methods is still an open problem
- Word embeddings are best evaluated in **extrinsic tasks**, but can also be evaluated based on their ability to accurately capture **word similarity**
- Different word embeddings are good for different tasks (in general, **BERT is the current state of the art**)