Q. Since SkipGram Negative sampling during each training step, we take 1 positive pair and k negative pairs. So each training step involves 1 + k dot products and sigmoid operations—not millions. This makes it very less resources expensive. But does not it make the training process slow as for suppose, 400000 words we have each positive pair, so the total number of training steps significantly increases.

✅ Recap: Why SGNS Is Efficient

Yes, in **each training step**, we:

- Take **1 real (center, context)** pair (positive sample),
- Sample **k random words** as negative examples,
- Do **1 + k dot products** and sigmoid calculations.

🔹 This avoids the **softmax denominator**, which would require a dot product over **all 400,000+ vocabulary words** — extremely expensive.

🤔 But Doesn't That Mean Too Many Training Steps?

Yes, but it's still efficient overall — here's why:

🔹 1. Training Over Corpus, Not All Word Pairs

- SGNS doesn't loop over every **possible pair** in the vocabulary.
- It only considers **(center, context)** pairs that actually occur in the corpus.
- So even if the vocabulary is 400,000, we don't generate $400{,}000^2$ training examples.

👉 **Training data size = number of word tokens × context window size**

For example:

- A corpus with 1 billion tokens
- Window size = 5
- ~5 billion training examples

Yes, that's a lot — **but still way fewer** than 400,000 × 400,000 = **160 billion** combinations you'd need for co-occurrence matrix-based training like in GloVe.

🔹 2. Training Is Local and Parallelizable

- Each (center, context) pair can be processed **independently**.
- Can be **massively parallelized** across GPUs/CPUs.
- Very amenable to **mini-batch SGD**.

🔹 3. k (Negative Samples) Is Small

- In practice, kkk is typically **5 to 20**.
- That's tiny compared to evaluating a full softmax over 400k+ words.
- You're trading off:
    - **Fewer computations per step**, but
    - **More steps** to converge.

✅ But **each step is cheap**, so you can do many steps **very fast**.

🔹 4. You Can Subsample Frequent Words

Word2Vec includes a **subsampling trick**:

- Frequent words like “the,” “of,” “and” are downsampled.
- This reduces the number of training steps **without hurting quality**.

⚖️ Summary: Trade-Off Comparison

Approach	Computation per Step	Number of Steps	Total Efficiency
Softmax (full vocab)	Very high (~400k ops)	Fewer	❌ Too expensive
SGNS	Low (1 + k ops)	More	✅ Efficient overall
GloVe (count matrix)	One pass, but matrix is huge	Needs SVD	⚠️ Scales poorly for large vocab

✅ Conclusion

Yes, SGNS involves many training steps, but:

- Each step is **extremely lightweight**,
- Training is **fast and parallelizable**,
- And the total cost is **much lower than full softmax or dense matrix methods**.

👉 **That’s why SGNS remains one of the most efficient and scalable methods for training word vectors**.