Skip-Gram vs CBOW — Deep Comparison (Expanded)

A complete, readable, and visually organized LaTeX note (explanations + tables + worked examples)

Purpose and scope

This document gives an exhaustive comparison of the two classic Word2Vec-style predictive models: **Continuous Bag-of-Words (CBOW)** and **Skip-Gram (SG)**. It integrates intuition, concrete step-by-step forward/backward math, implementation practicalities (negative sampling, hierarchical softmax), and multiple clear tables showing differences. Worked examples for both models are visually separated so you can print and annotate them.

1 Quick executive summary

- **CBOW:** Given a bag of context words (the neighbors), predict the missing center word. Fast, stable, good for frequent-word quality.
- **Skip-Gram:** Given a center word, predict each context word. Slower per training example but yields better vectors for rare words.
- Mathematically they use the same operations (lookup, dot product, softmax / contrastive loss); the only difference is the *direction of prediction* (which words serve as input and which are targets).

2 Notation (keep this visible)

- $|V|$ or m – vocabulary size.
- d – embedding dimension.
- C – number of context words (window size or number of neighbors used by CBOW / number of positions predicted by Skip-Gram).
- $V \in \mathbb{R}^{m \times d}$ – **input** embedding matrix. Row v_w is the input/context vector for word w.
- $U \in \mathbb{R}^{m \times d}$ – **output** / classifier matrix. Row u_w is the output vector for word w (used to score candidates).
- h – hidden/context summary vector. In CBOW $h = \frac{1}{C} \sum_{i=1}^{C} v_{c_i}$. In Skip-Gram $h = v_{center}$.
- $z_w = u_w^\top h$ – score (logit) for candidate word w.
- $p(w|\cdot)$ – probability after softmax, or use sigmoid for negative sampling.

3 High-level conceptual intuition

Think of learning embeddings as shaping a geometric space where words with similar usage (co-occurrence patterns) lie near each other. Both CBOW and Skip-Gram create the same geometry by repeatedly applying a *pull/push* rule:

- Pull together pairs of vectors that co-occur (positive examples).
- Push apart vectors that are unlikely to co-occur (negatives or the rest of vocabulary via softmax).

The difference is: CBOW compresses multiple neighbors into one summary then asks "which word fits here?"; Skip-Gram uses one center to ask "what neighbors should appear around this word?"

4 CBOW — full explanation (step-by-step)

CBOW: forward + loss + gradients

Goal: predict center word o from context $c_1, \dots, c_C$.

Forward:

1. Lookup: for each context word c_i fetch v_{c_i} from V.
2. Pool: $h = \frac{1}{C}\sum_{i=1}^{C} v_{c_i}$.
3. Score: $z_w = u_w^\top h$ for all $w \in V$.
4. Softmax: $p(w|context) = \dfrac{e^{z_w}}{\sum_{w'} e^{z_{w'}}}$.
5. Loss: $L = -\log p(o|context)$.

Backprop / gradients (full softmax):

$$\frac{\partial L}{\partial u_w} = (p(w) - t_w)\, h,$$

$$\frac{\partial L}{\partial h} = \sum_w (p(w) - t_w)\, u_w = U^\top (p - t),$$

$$\frac{\partial L}{\partial v_{c_i}} = \frac{1}{C}\frac{\partial L}{\partial h}.$$

Here $t_w = 1$ if $w = o$, else 0. Intuition: move u_o toward h, push other u_w away; change each v_{c_i} so their average becomes a better predictor of o.

5 Skip-Gram — full explanation (step-by-step)

Skip-Gram: forward + loss + gradients

Goal: predict context words $c_1, \dots, c_C$ from center word o.

Forward for a single pair (center o, one context c):

1. Lookup: $h = v_o$ (row for center from V).
2. Score: $z_w = u_w^\top h$ for all $w \in V$.
3. Softmax: $p(w|o) = \dfrac{e^{z_w}}{\sum_{w'} e^{z_{w'}}}$.
4. Loss for this position: $-\log p(c|o)$. Sum these for all positions in the window.

Gradients (per pair, full softmax):

$$\frac{\partial L}{\partial u_w} = (p(w) - t_w)\, h,$$

$$\frac{\partial L}{\partial v_o} = \sum_w (p(w) - t_w)\, u_w.$$

Sum over context positions to get the gradient for v_o when multiple targets exist.

6 Negative sampling (why and how)

Full softmax requires computing scores across the entire vocabulary and is thus $O(|V|)$ per training target. For large vocabularies (100k+) this is infeasible. Negative sampling (NS) converts the multi-class prediction into several binary classification tasks: is (center,context) a real pair or noise?

NS objective for one positive pair (CBOW or a single Skip-Gram pair):

$$L_{NS} = -\log \sigma(u_o^\top h) - \sum_{k=1}^{K} \log \sigma(-u_{w_k}^\top h),$$

where w_k are negative samples drawn from noise distribution $P_n(\cdot)$ (commonly unigram$^{3/4}$).

NS gradients:

$$\frac{\partial L}{\partial u_o} = (\sigma(u_o^\top h) - 1)\, h,$$

$$\frac{\partial L}{\partial u_{w_k}} = \sigma(u_{w_k}^\top h)\, h,$$

$$\frac{\partial L}{\partial h} = (\sigma(u_o^\top h) - 1)\, u_o + \sum_{k=1}^{K} \sigma(u_{w_k}^\top h)\, u_{w_k}.$$

7 Worked numeric examples (tiny) — visually separated

CBOW numeric toy

Vocabulary $|V| = 5$, $d = 3$, context indices 1,2 ($C = 2$):

```
V (input rows): v1=[0.2,0.1,0.0], v2=[0.1,0.0,0.2]
h = (v1+v2)/2 = [0.15,0.05,0.10]
U (output rows):
u0=[0.1,0.0,0.0] score=0.015
u1=[0.0,0.2,0.1] score=0.020
u2=[0.0,0.1,0.0] score=0.005
u3=[0.3,0.1,0.2] score=0.070 (true center)
u4=[0.2,0.0,0.1] score=0.040
Softmax -> p(true) ~ 0.2081, loss ~ 1.57
Updates: push u3 closer to h; update v1,v2 by (1/2)*grad_h.
```

Skip-Gram numeric toy

Use same sized matrices but now center = word 3 with v3 used as h. Scores will differ because h differs. For each context position you compute scores and update u$_{p}osandv_{c}enteraccordingly(oruseNStoupdateonlypos + Knegatives)$.

8 Deep comparison tables

Compact comparison

	CBOW	**Skip-Gram**
Input	multiple context words	single center word
Output	center word	context words
Hidden h	avg(context V-rows)	center V-row
Predictions / example	1	C
Complexity (neg samp)	O(K)	O(K*C)
Good for	speed; frequent words	rare words; nuance
Typical K	5–20	5–20

Exhaustive comparison table

Aspect	CBOW	Skip-Gram
Direction	context → center	center → context
Loss per example	one loss	sum over positions
Training speed	faster	slower per example
Rare-word behavior	weaker	stronger
Gradient stability	smoother (averaged ctx)	noisier but richer
Typical use	quick baseline, big corpora	high-quality embeddings (SGNS)
Final embedding	V rows or (V+U)/2	V rows or (V+U)/2
When to tie U=V	possible	possible

9 Practical advice and hyperparameters

- Window size: 2–5 typical for syntactic similarity; larger windows capture topic.
- Embedding dim d: 50–300 typical depending on corpus size.
- Negative samples K: 5–20 common; increase for smaller corpora or very large vocabularies.
- Negative distribution: unigram$^{3/4}$ recommended.
- Subsampling frequent words improves training speed and vector quality for rarer words.
- Final embedding: use V or average $(V + U)/2$.

10 FAQ and conceptual clarifications

Q: Are the math ops different? A: No — only data flow differs.

Q: Which matrix is used where? A: Inputs lookup from V; candidate scoring uses U. In CBOW context words are inputs; in Skip-Gram center is input.

Q: Can I mix them or tie them? A: You can tie $U = V$ (reduces params) or average them after training. Mixed architectures or positional variants are possible.

Conclusion

Both models are compact and powerful; choose based on your dataset and accuracy vs speed trade-offs. The mathematical core is identical; your decision is an engineering one based on training time, corpus size, and the kind of words you care about.