Finding Aesthetic Knife/Glove Combos in CS2 Using Embedding Similarity

A Comparison of ResNet50 and CLIP

André Menezes

November 23, 2025

Abstract

This reports looks into the use of deep visual embeddings to suggest a list of visually appealing knife-glove combinations for Counter-Strike 2 (CS2). Both ResNet50 and CLIP can be used to extract image embeddings for knives and gloves. Then, a score is calculated byconsidering the cosine distance between embeddings and other factors.

1 Introduction

With a current marketcap of roughly $5,170,000,000, cosmetic skins, specially knives and gloves, are a very important part of player entertainment and expression in CS2. With over 500 individual knife finishes across all the types, manually finding aesthetically matching combinations can be very time-consuming and subjective. In addition to that, other than set lists of possible combinations, users are limited to a try and repeat approach when looking for a new pair of glove/knife.

In this project, I used deep learning models to compute visual similarity between knife and gloves skins and automatically suggest top matching pairs. Initially, I used an approach with a classical CNN (Resnet50), and then moved on to a more recent vision language model (CLIP) to better understand their strengths and weaknesses for this task.

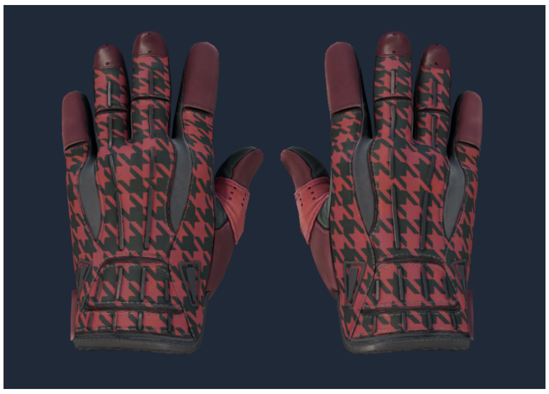
2 Dataset

I collected individual screenshots of 72 gloves and 586 knives. Each image is preprocessed to a fixed resolution (224×224 pixels) and normalized to the standard ImageNet mean and standard deviation.

3 Methods

3.1 Removing image background

Each image contained a dark blue background (image (a)). The background color can affect the cosine distance between image embeddings, which can lead to higher cosine distance even if the images are similar. To address this issue, all images had their background removed to ensure that embeddings focus on the actual object.


(a) Raw gloves image

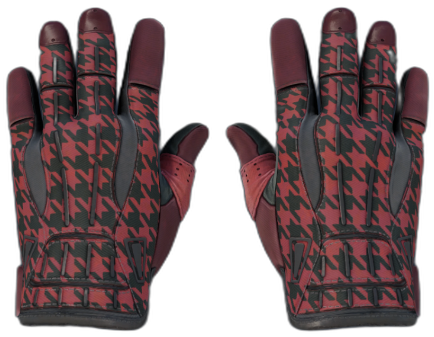

(b) Cleaned gloves image

Figure 1: Comparison of original and background-removed images

3.2 Embedding Extraction with ResNet50

A pretrained ResNet50 model (trained on ImageNet) is used. In order to fit the purpose of this experiment, I removed the classification head and set the model to evaluation mode. These changes to the model are needed to use ResNet50 as a feature extractor rather than a classifier.

For each image, I extract the output from the gloval average pooling layer. That is, taking the output from the layer just before the classification head in ResNet-50. The image is transformed into a single, fixed-length vector (2048 values for ResNet50).

These vectors summarizes the most important visual features existent in the image, which will then bet used to compare images or represent them in a high-dimensional space for tasks like similarity search.

3.3 Embedding Extraction with CLIP

While the model works relatively well with most skins. It came to my attention that the model struggles with skins that have some texture to it (for example, metalic finish to Doppler skins). As an alternative to ResNet50, I also use a pretrained CLIP model (ViT-B/32). This model is expected to perform better overall as it was trained on a larger, more diverse dataset or images and text. CLIP is able to associate visual patterns (such as reflections, colors, and textures) with descriptive concepts.

For each skin image, we apply CLIP's image encoder to obtain an embedding vector. CLIP is trained jointly on images and text (differently to ResNet50), which often makes its image embeddings more aligned with human judgments of similarity and style.

3.4 Cosine Similarity and Ranking

For each glove embedding, we compute the cosine similarity to all knife embeddings. Given two embedding vectors u and v, the cosine similarity is

$$\text{cos\_sim}(u, v) = \frac{u \cdot v}{\|u\| \, \|v\|}. \tag{1}$$

For each knife, we rank all gloves by decreasing cosine similarity and take the top-5 as suggested matches. The final score consists of three elements. First, and most importantly, the cosine similarity explained above. Next, we check add a boost of 0.25 if the combo already exists in the list of combos (combos.csv). Lastly, if the combo does not exist in combos.csv but the same combo exists with different skins (that is, that combo exists with a different knife of the same finish), an indirect boost of 0.15 is added.

4 Experiments and Results

4.1 Experimental Setup

Let's look into how both models on the same set of knife and glove images. All embeddings are computed offline, cosine distances are calculated using standard linear algebra operations, and direct/indirect boosting is calculated by looking at the list of combos.

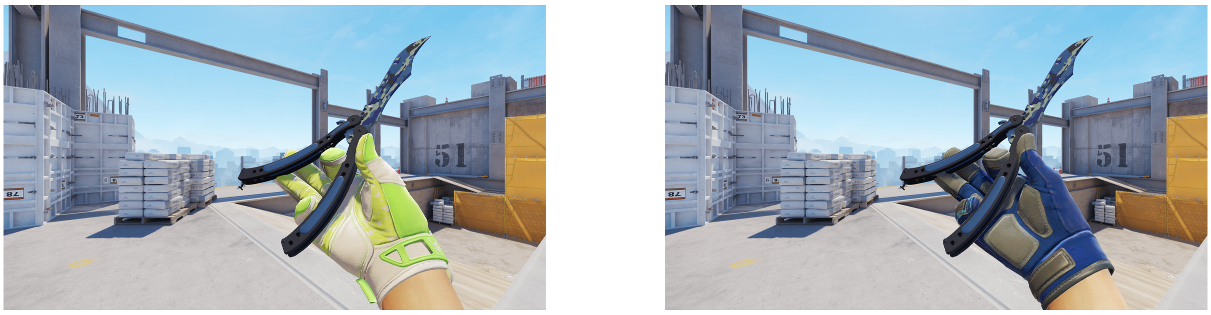
4.2 Sample Results

Table 1 shows an example of top-3 knife matches for a given glove using both models.

Table 1: Example top-3 matches for a specific glove using ResNet50 and CLIP. (Replace placeholders with actual skin names.)

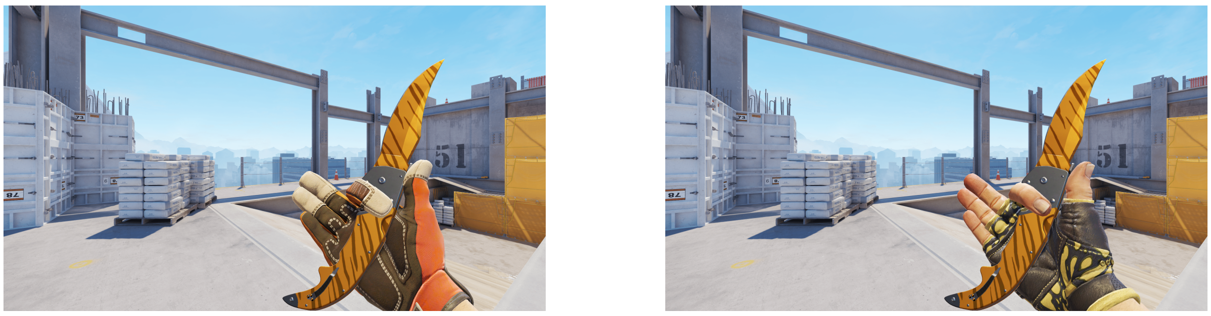
Knife	Top-3 Gloves (ResNet50)	Top-3 Knives (CLIP)
Butterfly Bright Water	Sport Hedge Maze (0.4275) Moto Boom (0.3439) Hydra Emerald (0.3359)	Specialist Mogul (0.6927) Moto Polygon (0.6831) Sport Hedge-Maze (0.4761)
Falchion Tiger Tooth	Specialist Tiger Strike (0.2738) Specialist Forest DDPAT (0.2733) Hydra Emerald (0.2553)	Broken Fang Yellow Banded (0.6297) Specialist Tiger Strike (0.3861) Specialist Forest DDPAT (0.3759)

Below, we look at example image of a recommendation combo from each model to illustrate differences between them.



(a) Best combo ResNet50 (b) Best combo CLIP

Figure 2: Model comparison for Butterfly Bright Water



(a) Best combo ResNet50 (b) Best combo CLIP

Figure 3: Model comparison for Falchion Tiger Tooth

5 Discussion

By looking at the sample results from above. ResNet50 struggles with different textures, as seen in the first example. The model's top recommendation is a green glove for a blue knife. In general, the ResNet50 tends to favor matches with similar dominant colors and basic textures, as observed in the second example with the Falchion Tiger Tooth. Frequently, some combinations look less coherent to a human observer, suggesting that low-level features alone are not always sufficient for subjective aesthetic matching.

CLIP, on the other hand, often proposes combinations that better capture overall style and theme (for example, matching similar patterns, wear levels, or "vibes"). As observed in the first example, the CLIP suggested blue gloves for a blue knife, which meets the goal of providing a visually pleasing combo. This likely comes from its training on large-scale image-text data, making its embeddings more aligned with semantic and stylistic concepts.

For the second example, since we have a mostly yellow skin with little texture, both models perform somewhat similarly. Although they have different top picks, out of the three top choices, two are the same.

6 Next Steps

In general, CLIP consistently offers good combos for any knife. However, my next step is understanding why the model suggests a green glove for a yellow knife as the third choice, even when there are more than 3 yellow gloves. This is likely related with the color yellow not being the predominant one.

Furthermore, it came to my attention that hand wraps never perform well, regardless of how well they look with a knife. This was observed when printing out all scores and hand wraps were constantly at the bottom.

Lastly, since CLIP was trained on both images and text, I want to analyze how giving gloves/knives a more descriptive name affects the model performance.

7 Conclusion

This is simple working system for suggesting knife–glove combinations in CS2 using deep visual embeddings and cosine distance. While both ResNet50 and CLIP can produce reasonable matches, CLIP generally yields combinations that feel more consistent with human aesthetic judgment. Future work could include understanding how to increase performance, why hand wraps perform poorly, and how remaning gloves/knives affect performance.

A Code Snippets

Listing 1: Example Python code for preprocessing images

```python
import torchvision.transforms as transforms

# Transfrom pipeline for image preprocessing
transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224,
        0.225]),
])
```

Listing 2: Example Python code for cosine similarity.

```python
import numpy as np

def cosine_similarity(vec1: np.ndarray, vec2: np.ndarray) -> float:
    """
    vec1, vec2: 1D numpy arrays (any length, e.g. 512 for CLIP)
    """
    dot = float(np.dot(vec1, vec2))
    norm1 = float(np.linalg.norm(vec1))
    norm2 = float(np.linalg.norm(vec2))

    if norm1 == 0.0 or norm2 == 0.0:
        return 0.0

    return dot / (norm1 * norm2)
```

Listing 3: Example Python code for ranking gloves, given a knife.

```python

def rank_gloves_for_knife(
    knife_id: str,
    knife_folder: str,
    glove_folder: str,
    combos_csv: str = "metadata/combos.csv",
    top_k: int = 5,
    direct_boost: float = 0.25,
    indirect_boost_scale: float = 0.15,
) -> List[Tuple[str, float]]:
    """
    direct_boost: added if exact (glove, knife) is in combos
    indirect_boost_scale: multiplied by max similarity to any combo-
       knife for that glove
    """
    knife_embeds = load_embeddings(knife_folder)
    glove_embeds = load_embeddings(glove_folder)

    combos, glove_to_knives, knife_to_gloves = load_combos(combos_csv)

    if knife_id not in knife_embeds:
        raise KeyError(f"Knife ID '{knife_id}' not found in {
            knife_folder}")

    query = knife_embeds[knife_id]
    scores: List[Tuple[str, float]] = []

    for glove_id, glove_vec in glove_embeds.items():
        base_sim = cosine_similarity(query, glove_vec)
        score = base_sim

        # 1) Direct combo boost
        if (glove_id, knife_id) in combos:
            score += direct_boost

        # 2) Indirect combo boost via similar knives
        related_knives = glove_to_knives.get(glove_id, [])
        if related_knives:
            sims = []
            for related_knife_id in related_knives:
                rk_vec = knife_embeds.get(related_knife_id)
                if rk_vec is None:
                    continue
                sims.append(cosine_similarity(query, rk_vec))

            if sims:
                max_sim_to_combo_knife = max(sims)
                # e.g. if max_sim is 0.9, indirect bonus ~ 0.9 * 0.15
                score += indirect_boost_scale * max_sim_to_combo_knife

        scores.append((glove_id, score))

    scores.sort(key=lambda x: x[1], reverse=True)
    return scores[:top_k]
```