Paper Review: Learning Sparse Visual Representations via Spatial-Semantic Factorization

ICML 2026, Microsoft

Presented by Choeun Kim

July 16, 2026

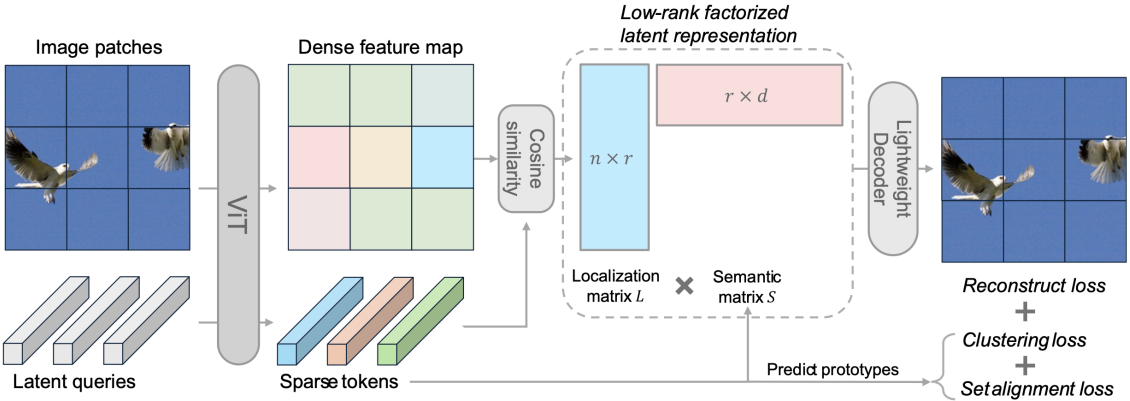

Figure: Overview of the STELLAR framework

Forward Pass

Given an input image, STELLAR first converts the image into dense patch tokens. It then appends learnable latent queries to extract sparse semantic tokens through a ViT encoder.

$$\text{Image: } X \in \mathbb{R}^{B\times 3\times H\times W}$$

$$\text{Patch tokens: } P \in \mathbb{R}^{B\times n\times d}, \qquad \text{Latent queries: } Q \in \mathbb{R}^{B\times r\times d}$$

$$\text{ViT input: } T_{\text{in}} = [P; Q] \in \mathbb{R}^{B\times (n+r)\times d}$$

$$\text{ViT output: } T_{\text{out}} \in \mathbb{R}^{B\times (n+r)\times d}$$

For simplicity, the optional CLS token is omitted in this notation.

Forward Pass

After passing through the ViT encoder, the output tokens are separated into dense patch features U and sparse semantic tokens S.

$$U \in \mathbb{R}^{B \times n \times d} \qquad S \in \mathbb{R}^{B \times r \times d}$$

Tensor	Shape	Meaning
U	$B \times n \times d$	Dense patch feature map
S	$B \times r \times d$	Sparse semantic tokens, "what"
r	usually 16	Number of sparse concept tokens
n	e.g., 196	Number of image patches

Computing the Localization Matrix L

STELLAR computes a localization matrix L by comparing dense patch features U with sparse semantic tokens S. This matrix describes **how each patch is softly assigned to the sparse concepts**.

First, both U and S are projected into a shared embedding space.

$$U' = UW_1 \in \mathbb{R}^{B \times n \times p}$$

$$S' = SW_2 \in \mathbb{R}^{B \times r \times p}$$

Then, pairwise cosine similarity is computed.

$$A = \text{cossim}(U', S') \in \mathbb{R}^{B \times n \times r}$$

Computing the Localization Matrix L

Finally, STELLAR applies a softmax over the sparse-token dimension.

$$L = \text{softmax}\left(\frac{A}{\tau_{\text{spatial}}}\right) \in \mathbb{R}^{B\times n\times r}$$

Each row L_i represents how patch i is softly assigned to the sparse tokens.

$$\sum_{j=1}^{r} L_{ij} = 1$$

Thus, L acts as the "where" component of the factorized representation.

Low-rank Dense Representation: $Z = LS$

The **localization matrix** L and **sparse semantic tokens** S are multiplied to construct a low-rank dense representation.

$$L \in \mathbb{R}^{B \times n \times r}, \qquad S \in \mathbb{R}^{B \times r \times d}$$

$$Z = LS \in \mathbb{R}^{B \times n \times d}$$

For example, if $n = 196$, $r = 16$, and $d = 768$,

$$L : B \times 196 \times 16, \qquad S : B \times 16 \times 768$$

$$Z : B \times 196 \times 768$$

The important point is that Z is reconstructed from only r sparse tokens.

$$\text{rank}(Z) \leq r$$

Loss 1: Reconstruction

The low-rank dense representation $Z = LS$ is passed to a decoder for reconstruction.

$$Z \in \mathbb{R}^{B \times n \times d} \quad \longrightarrow \quad D(Z) \in \mathbb{R}^{B \times n \times |\mathcal{V}|}$$

Here, $|\mathcal{V}|$ denotes the VQGAN codebook size. The decoder D predicts a discrete VQGAN token for each patch.

$$Y_{\text{VQ}} \in \mathbb{R}^{B \times n}$$

$$\mathcal{L}_{\text{recon}} = \text{CE}\left(D(Z), Y_{\text{VQ}}\right)$$

Sparse tokens are projected using a learnable projector $h : \mathbb{R}^d \to \mathbb{S}^{p-1}$ and assigned to learnable visual prototypes $C \in \mathbb{R}^{K \times p}$. ($K = 16{,}384$)

$$h(S) \in \mathbb{R}^{B \times r \times p}$$

$$\lambda = h(S)C^\top \in \mathbb{R}^{B \times r \times K}$$

$$q = \text{softmax}(\lambda/\tau) \in \mathbb{R}^{B \times r \times K}$$

The balanced target assignment is obtained using Sinkhorn-Knopp.

$$\tilde{q} = \text{Sinkhorn}(q)$$

The clustering loss is then defined as[1]

$$\mathcal{L}_{\text{cluster}} = -\frac{1}{Br} \sum_{i=1}^{B} \sum_{j=1}^{r} \sum_{k=1}^{K} \tilde{q}_{j,k}^{i} \log q_{j,k}^{i}$$

[1] Stop gradient is applied to $\tilde{q}$.

Loss 3: Set Concept Alignment

The sparse tokens are treated as an unordered set. Therefore, tokens from a global view and a transformed view cannot be aligned by their indices.

Let

$$S^g \in \mathbb{R}^{B \times r \times d}$$

be the sparse tokens from a global view, and

$$S^v \in \mathbb{R}^{B \times r \times d}$$

be the sparse tokens from a transformed view.

The goal is to align semantic concepts across views, not token positions.

Loss 3: Set Concept Alignment

The pairwise cost matrix is computed as

$$\Theta_{j'j} = \|s^v_{j'} - s^g_j\|_2$$

$$\Theta \in \mathbb{R}^{B\times r\times r}$$

STELLAR solves entropy-regularized optimal transport using the Sinkhorn algorithm.

$$P \in \mathbb{R}^{B\times r\times r}$$

$$\sigma(j') = \arg\max_j P_{j'j}$$

The transformed token $s^v_{j'}$ is trained to predict the prototype assignment of the matched global token $s^g_{\sigma(j')}$.

$$\mathcal{L}_{\text{align}} = -\frac{1}{r}\sum_{j'=1}^{r}\sum_{k=1}^{K}\tilde{q}^g_{\sigma(j'),k}\log q^v_{j',k}$$

Pseudo-code Summary

```
# X: [B, 3, H, W]

patch_tokens = patch_embed(X)              # [B, n, d]
latent_queries = repeat(Q, B)              # [B, r, d]

tokens = concat([patch_tokens, latent_queries], dim=1)
out = vit(tokens)

U = out[:, :n]                              # [B, n, d]
S = out[:, n:n+r]                           # [B, r, d]

U_proj = normalize(U @ W1)                  # [B, n, p]
S_proj = normalize(S @ W2)                  # [B, r, p]

sim = cosine_similarity(U_proj, S_proj)     # [B, n, r]
L = softmax(sim / tau_spatial, dim=-1)      # [B, n, r]

Z = L @ S                                   # [B, n, d]

logits_rec = decoder(Z)                     # [B, n, vocab_size]
loss_recon = CE(logits_rec, vqgan_tokens)

proto_logits = projector(S) @ C.T           # [B, r, K]
q = softmax(proto_logits / tau, dim=-1)
q_tilde = sinkhorn(q).detach()

loss_cluster = CE(q, q_tilde)
```

Pseudo-code: View Alignment

```
# Global view and transformed view
S_global = encoder(X_global).sparse_tokens  # [B, r, d]
S_view   = encoder(X_view).sparse_tokens    # [B, r, d]

# Pairwise token distance
cost = pairwise_l2(S_view, S_global)        # [B, r, r]

# Sinkhorn OT matching
P = sinkhorn_ot(cost)                       # [B, r, r]
matching = argmax(P, dim=-1)                # [B, r]

# Prototype assignment target from global view
q_global = prototype_prediction(S_global)   # [B, r, K]
q_tilde_global = sinkhorn(q_global).detach()

target = gather(q_tilde_global, matching)   # [B, r, K]

# Predict matched concept assignments from transformed view
q_view = prototype_prediction(S_view)       # [B, r, K]

loss_align = CE(q_view, target)

loss = (
    0.05 * loss_recon
    + loss_cluster
    + loss_align
)
```

Core Intuition

The key idea of STELLAR is to avoid representing an image only as a dense grid of patch features. Instead, the representation is factorized into two components.

$$S \in \mathbb{R}^{B \times r \times d} \quad : \text{sparse semantic tokens, “what”}$$

$$L \in \mathbb{R}^{B \times n \times r} \quad : \text{localization matrix, “where”}$$

$$Z = LS \quad : \text{low-rank dense representation}$$

Why this works

- S is encouraged to contain view-invariant semantic concepts.
- L captures where each concept appears in the image.
- Reconstruction forces S and L to explain the full image.
- Clustering and alignment prevent S from becoming a mere reconstruction code.

Therefore, STELLAR separates semantic invariance and spatial equivariance into different factors.