**How to Represent / Analyze shared\_latent (128-dim vector)**

Once your model is trained, for each input sequence window, you’ll extract a **128-dimensional vector** like this:

python

CopyEdit

latent\_vector = encoder.predict([trigger\_seq, pattern\_seq, structure\_seq])

Now to **understand or visualize these vectors**, here are your main options:

**🔹 1. Dimensionality Reduction for Visualization**

Since 128 dimensions is too large to plot directly, use techniques like:

**✅ UMAP (best for structure preservation)**

python

CopyEdit

from umap import UMAP

import matplotlib.pyplot as plt

umap\_model = UMAP(n\_components=2)

latent\_2d = umap\_model.fit\_transform(latent\_vectors)

plt.scatter(latent\_2d[:, 0], latent\_2d[:, 1])

plt.title("2D UMAP Projection of Latent Vectors")

plt.show()

**✅ t-SNE (good for local clusters)**

python

CopyEdit

from sklearn.manifold import TSNE

tsne = TSNE(n\_components=2, perplexity=30)

latent\_2d = tsne.fit\_transform(latent\_vectors)

You’ll start seeing **clusters of similar patterns**.

**🔹 2. Clustering to Discover Pattern Types**

Use the latent vectors to group similar structures:

python

CopyEdit

from sklearn.cluster import KMeans

kmeans = KMeans(n\_clusters=10)

labels = kmeans.fit\_predict(latent\_vectors)

# Visualize with cluster colors

plt.scatter(latent\_2d[:, 0], latent\_2d[:, 1], c=labels)

You can now **label clusters** based on visual inspection of samples from each group.

**🔹 3. Cosine Similarity / Distance Search**

Find similar patterns using cosine distance:

python

CopyEdit

from sklearn.metrics.pairwise import cosine\_similarity

# Compare one pattern to all others

ref\_vec = latent\_vectors[0]

similarities = cosine\_similarity([ref\_vec], latent\_vectors)

most\_similar\_indices = similarities.argsort()[0][-5:] # Top 5 most similar

Useful to **find analog patterns** from history for forecasting or validation.

**🔹 4. Heatmap or PCA Plot of Latent Space**

Quick insights into which of the 128 dimensions carry the most variation:

python

CopyEdit

import seaborn as sns

sns.heatmap(latent\_vectors[:100], cmap="viridis")

**✅ Summary**

| **Goal** | **Method** |
| --- | --- |
| Visualize pattern space | UMAP / t-SNE 2D projection |
| Group similar patterns | KMeans / HDBSCAN clustering |
| Match new patterns to known ones | Cosine similarity search |
| Interpret dominant latent features | PCA / Heatmaps |