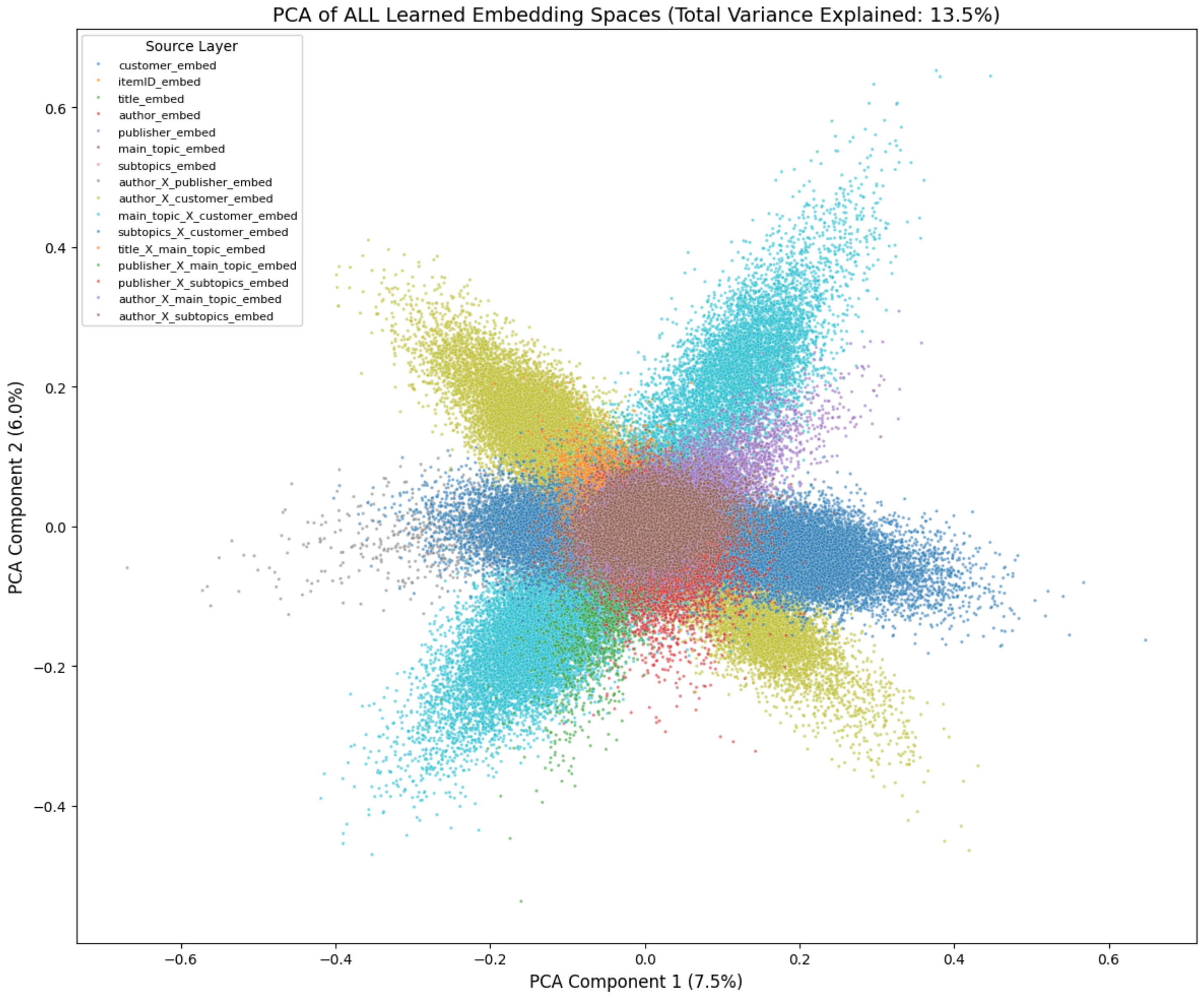
```python
# This code is adapted from Google Gemini.
# It retrives all our embedding layers from the trained model and plots them according to their PCA significance.
# That is, the features are reduced to the two most important PCA components and the weights from 4 layers are plotted.
# Each color represents an important feature.

from sklearn.decomposition import PCA
from tensorflow import keras
import warnings
warnings.filterwarnings('ignore')

ALL_EMBEDDING_LAYERS = [
    'customer_embed', 'itemID_embed', 'title_embed', 'author_embed',
    'publisher_embed', 'main_topic_embed', 'subtopics_embed',
    'author_X_publisher_embed', 'author_X_customer_embed',
    'main_topic_X_customer_embed', 'subtopics_X_customer_embed',
    'title_X_main_topic_embed', 'publisher_X_main_topic_embed',
    'publisher_X_subtopics_embed', 'author_X_main_topic_embed',
    'author_X_subtopics_embed'
]
TARGET_DIM = 64

all_embeddings = []
all_source_labels = []
all_layer_names = []

# Extraction and Projection
print(f"Extracting and projecting {len(ALL_EMBEDDING_LAYERS)} embedding spaces to {TARGET_DIM} dimensions...")

for layer_name in ALL_EMBEDDING_LAYERS:
    try:
        # Retrieve the layer from the model
        embedding_layer = inference_model.get_layer(layer_name)
        weights = embedding_layer.get_weights()[0]
        embeddings = weights[1:] # Skip OOV token (index 0)

        current_dim = embeddings.shape[1]
        final_embeddings = embeddings

        # Project layers with dimension != 64D to the TARGET_DIM (64D)
        if current_dim != TARGET_DIM:
            print(f"  - Projecting {layer_name}: {current_dim}D -> {TARGET_DIM}D...")

            # Create a temporary sequential model for linear projection
            projection_model = Sequential([
                Input(shape=(current_dim,)),
                Dense(TARGET_DIM, use_bias=False)
            ])

            # Project using an untrained dense layer to resize the vector space
            final_embeddings = projection_model.predict(embeddings, verbose=0)

        # Store the resulting embeddings and corresponding labels
        all_embeddings.append(final_embeddings)
        all_layer_names.append(layer_name)

        # Create labels for the plot
        labels = [layer_name] * final_embeddings.shape[0]
        all_source_labels.extend(labels)

    except Exception as e:
        # Catch layers that might not be embeddings (like 'click' or 'basket' if the list included them)
        # or layers that weren't defined in the final model structure.
        print(f"Warning: Skipping layer '{layer_name}'. Error during extraction: {e}")
        continue

# Concatenation and PCA
embedding_matrix = np.concatenate(all_embeddings, axis=0)
print(f"\nTotal embedding vectors collected: {embedding_matrix.shape[0]:,}")
print(f"Final vector dimension: {embedding_matrix.shape[1]}D")

N_COMPONENTS = 2
print(f"Applying PCA to reduce {embedding_matrix.shape[1]}D vectors to {N_COMPONENTS}D...")

pca = PCA(n_components=N_COMPONENTS)
weights_pca = pca.fit_transform(embedding_matrix)

# Visualization
explained_variance = pca.explained_variance_ratio_.sum()

plt.figure(figsize=(12, 10))
sns.scatterplot(
    x=weights_pca[:, 0],
    y=weights_pca[:, 1],
    hue=all_source_labels,
    s=5,
    alpha=0.6,
    palette=sns.color_palette("tab10", len(all_layer_names))
)

plt.title(f'PCA of ALL Learned Embedding Spaces (Total Variance Explained: {explained_variance:.1%})', fontsize=14)
plt.xlabel(f'PCA Component 1 ({pca.explained_variance_ratio_[0]:.1%})', fontsize=12)
plt.ylabel(f'PCA Component 2 ({pca.explained_variance_ratio_[1]:.1%})', fontsize=12)
plt.legend(title='Source Layer', loc='best', fontsize=8)
plt.tight_layout()
plt.show()
```

```
Extracting and projecting 16 embedding spaces to 64 dimensions...
  - Projecting author_embed: 55D -> 64D...
  - Projecting publisher_embed: 25D -> 64D...
  - Projecting main_topic_embed: 21D -> 64D...
  - Projecting author_X_publisher_embed: 100D -> 64D...
  - Projecting author_X_customer_embed: 128D -> 64D...
  - Projecting main_topic_X_customer_embed: 128D -> 64D...
  - Projecting subtopics_X_customer_embed: 128D -> 64D...
  - Projecting title_X_main_topic_embed: 100D -> 64D...
  - Projecting publisher_X_main_topic_embed: 70D -> 64D...
  - Projecting publisher_X_subtopics_embed: 100D -> 64D...
  - Projecting author_X_main_topic_embed: 100D -> 64D...
  - Projecting author_X_subtopics_embed: 100D -> 64D...

Total embedding vectors collected: 253,228
Final vector dimension: 64D
Applying PCA to reduce 64D vectors to 2D...
```



Each cluster represents how the embedding of that feature is represented in this space, what categeory the embedding encodes for. For example, author_embed (red), author_x_main_topics_embed (purple), and author_x_subtopics_embed (brown) are all embedded very near / on top of each other, indicating that they're pretty similar in category. This implies that the author feature is the more important part of the crossed features (though the purple one does shoot off in a different direction, showing that the cross did teach the model some complex relationship). Dark blue and light blue are almost orthogoal, indicating that these features encode much different categories (here its the topic vs the subtopic). Finally, the fact that they're so close together shows that these features are highly related to each other.