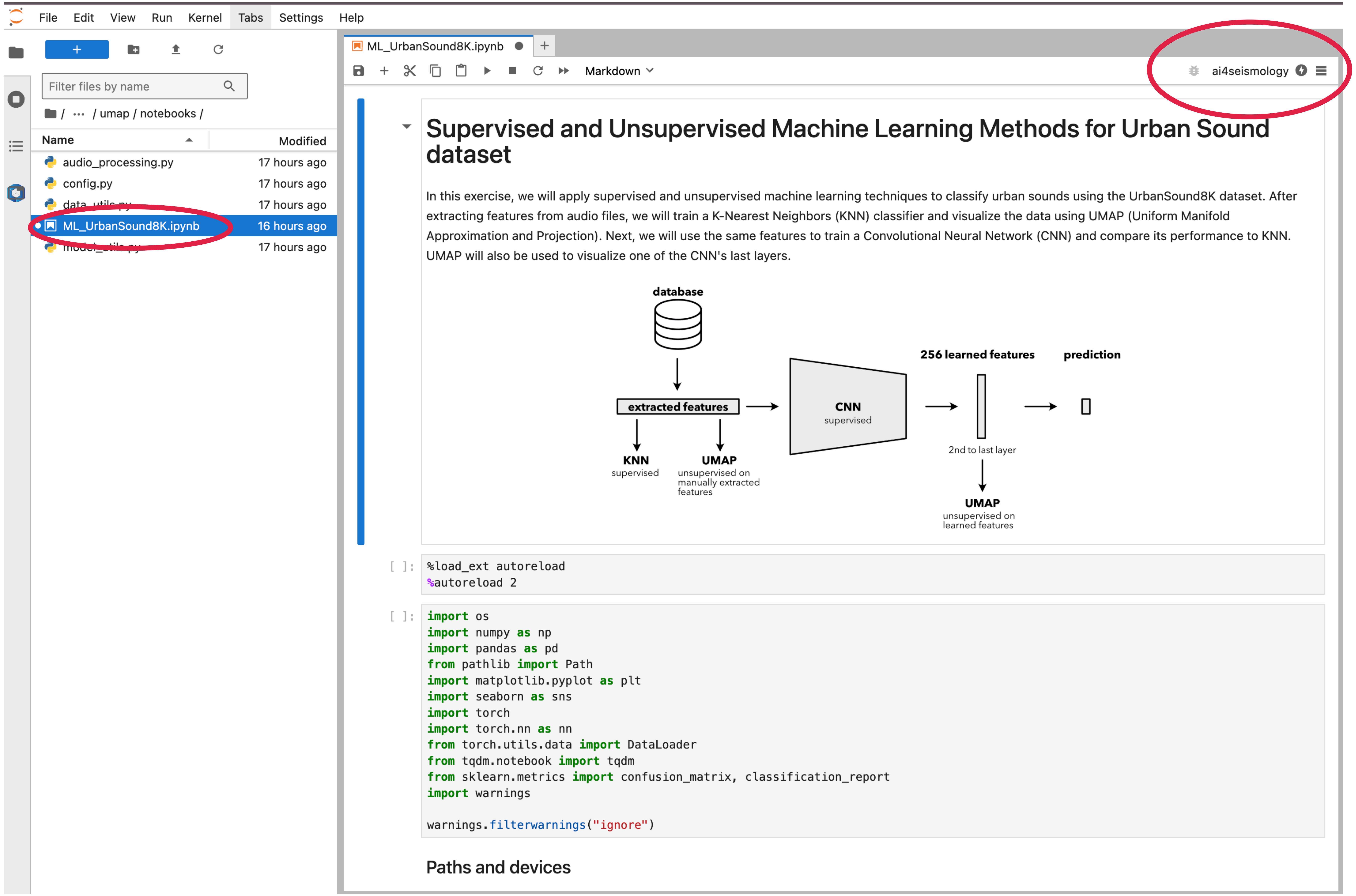
# Before we start…



# Before we start...

ML_UrbanSound8K.ipynb | config.py

```python
"""
Configuration parameters for audio processing, datasets, and model training.
"""

# Audio processing parameters
SR = 22050              # Sampling rate
DURATION = 4            # Duration of the audio in seconds
N_MELS = 128            # Number of Mel bands
N_FFT = 2048            # FFT window size
HOP_LENGTH = 512        # Hop length for STFT
N_MFCC = 13             # Number of MFCC coefficients

# Model parameters
NUM_CLASSES = 10        # Number of audio classes

# Training parameters
BATCH_SIZE = 128        # Batch size for training
LEARNING_RATE = 0.001 # Initial learning rate
EPOCHS = 20             # Maximum number of training epochs
# XXX TEST:
# EPOCHS = 3             # Maximum number of training epochs
EARLY_STOPPING_PATIENCE = 3 # Early stopping patience
EPOCHS_FOR_PRETRAINED = 2  # Epochs for fine-tuning pretrained models
NUMBER_WORKERS = 4      # Number of workers for data loading
# Scheduler parameters
SCHEDULER_STEP_SIZE = 5  # Step size for learning rate scheduler
SCHEDULER_GAMMA = 0.5     # Gamma for learning rate scheduler

# Model architecture parameters
DROPOUT_RATE = 0.5      # Dropout rate for CNN
FC_SIZE = 256           # Size of fully connected layer

# Other settings
RANDOM_SEED = 42        # Random seed for reproducibility
MODEL_SAVE_PATH = "best_model.pth"  # Path to save the best model
```

# Before we start...

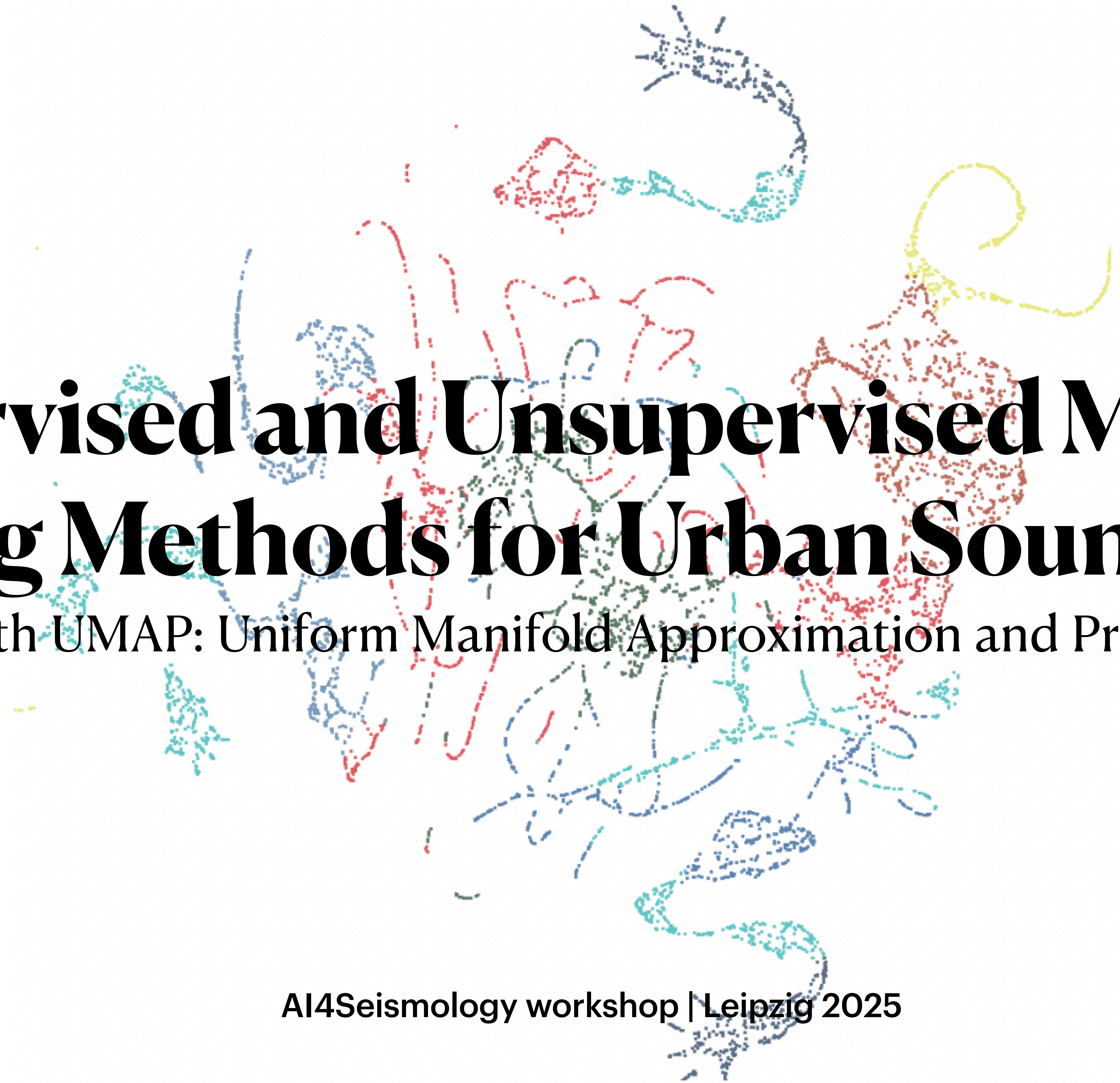
- Open your terminal and type:
  1. *cd ai4seismology-2025/day3/umap/*
  2. *mkdir data*
  3. *cd data*
  4. *cp -r /data/horse/ws/s4122485-ai4seismology/data/umap/* .*
- *OR:*
  -

```python
# data_path = Path("..") / "data"
data_path = Path("..") / "/data/horse/ws/s4122485-ai4seismology/data/umap"
metadata_path = data_path / "UrbanSound8K.csv"

# # load device depending on your system
if torch.cuda.is_available():
    device = torch.device("cuda")        # NVIDIA GPU
elif torch.backends.mps.is_available():
    device = torch.device("mps")         # Apple Silicon
else:
    device = torch.device("cpu")         # CPU fallback

print(f"Using device: {device}")
# One liner:
# device = torch.device("cuda" if torch.cuda.is_available() else "mps" if torch.backends.mps.is_available() else "cpu")
```

***Execute the notebook.***



# Supervised and Unsupervised Machine Learning Methods for Urban Sound dataset

## With UMAP: Uniform Manifold Approximation and Projection

AI4Seismology workshop | Leipzig 2025

# What is UMAP?

- UMAP is a **dimensionality reduction** technique that helps visualise high-dimensional data in 2D or 3D, preserving both global structure and local neighbourhoods better than other methods (e.g. t-SNE).
- **Key Features:**
    - **Manifold Learning**: Assumes data lies on a low-dimensional manifold within a high-dimensional space.
    - **Graph-based Approach**: Builds a k-nearest neighbour graph, then optimises a low-dimensional layout that preserves this structure.
    - **Fast and Scalable**: Faster and more memory-efficient than t-SNE, works well with large datasets.
    - **Preserves Local and Global Structure**: Better at maintaining the overall shape of the data compared to t-SNE.
    - **Versatile**: Works for visualisation, clustering, preprocessing, and more.

# UMAP is not a clustering algorithm

- Short Answer:
  UMAP is a dimensionality reduction technique, not a clustering algorithm. It **reveals structure** in the data, but it doesn’t assign labels or group data points into clusters on its own.
- Longer Answer:
  - UMAP projects high-dimensional data into a lower-dimensional space by preserving the topological relationships between points – i.e., it tries to maintain "who is close to whom."
  - It creates a neighbourhood graph in high-dimensional space, then tries to lay that out in a lower-dimensional space in a way that preserves the structure.
  - The result often shows clusters visually, but these clusters are an emergent property, not something UMAP explicitly calculates or labels.

# What Is Manifold Learning?

- Manifold learning is a type of nonlinear dimensionality reduction based on the idea that high-dimensional data often lies on a lower-dimensional curved surface (a "manifold") embedded in that high-dimensional space.
- Example:
  - Imagine a sheet of paper (2D) that's been twisted into a Swiss roll shape in 3D space.
  - Although it exists in 3D, all the points on the roll really live on a 2D surface.
  - So, we can “unroll” it and find a 2D representation that captures the true structure.
  - UMAP assumes something similar: your high-dimensional data actually lies on a much lower-dimensional manifold (e.g., 2D or 3D).

# What is UMAP?

## Two important parameters when working with UMAP

**n_neighbors**

Approximate nearest neighbours are used to construct the initial **high-dimensional graph**

Controls how UMAP **balances local versus global structure**

- Low values will push UMAP to focus more on local structure by constraining the number of neighbouring points considered when analysing the data in high dimensions
- High values will push UMAP towards representing the big-picture structure while losing fine detail.

**min_dist**

Minimum distance between points in a **low-dimensional space**

Controls how tightly UMAP **clumps points** together

- Low values leading to more tightly packed embeddings
- Larger values will make UMAP pack points together more loosely (preservation of the broad topological structure)

Links to check out:

- https://pair-code.github.io/understanding-umap/
- https://umap-learn.readthedocs.io/en/latest/how_umap_works.html

# n_neighbours

## n_neighbors = 3

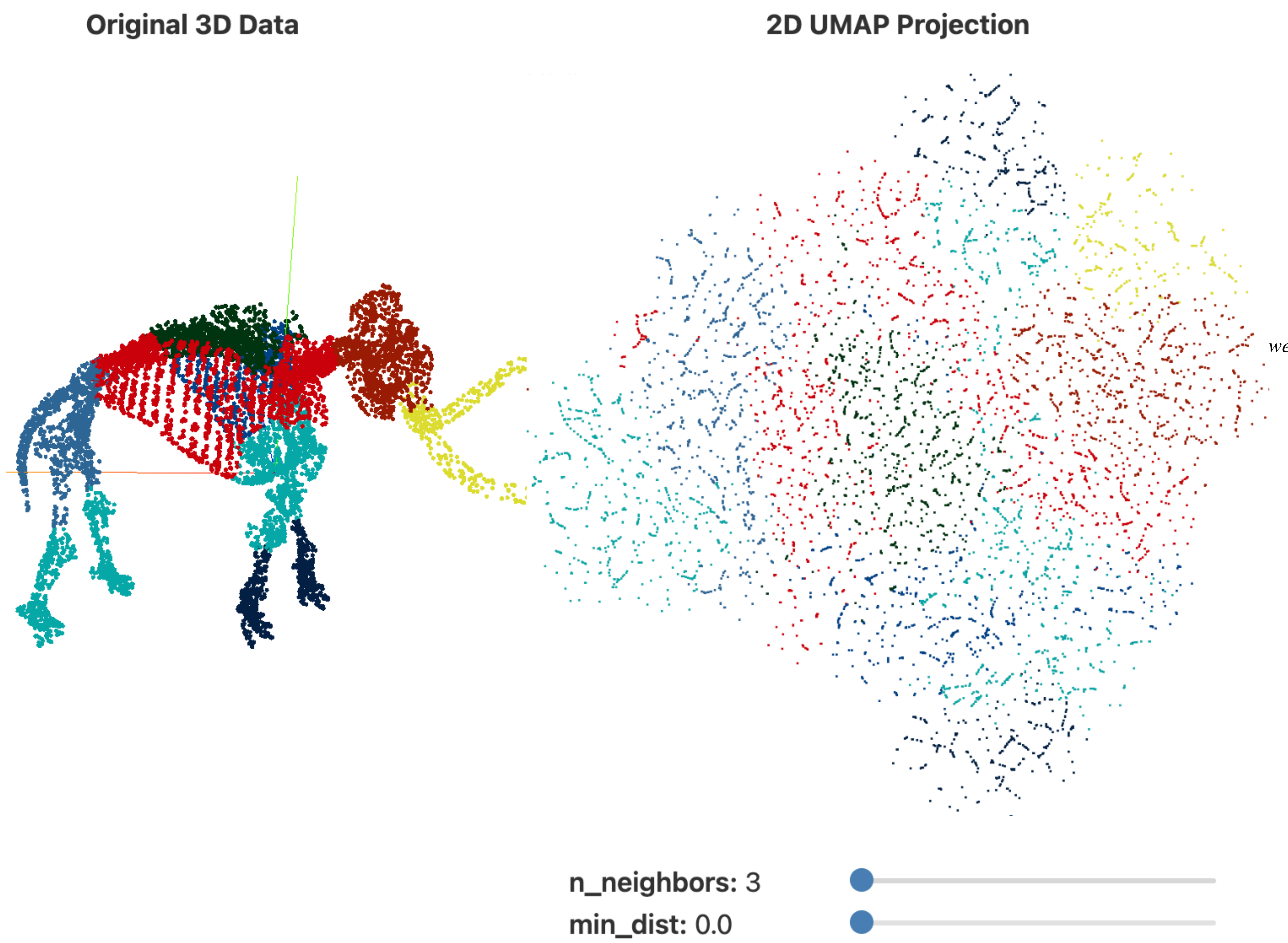



*Figure 5:* *UMAP projections of a 3D woolly mammoth skeleton (50k points, 10k shown) into 2 dimensions, with various settings for the `n_neighbors` and `min_dist` parameters.*

## n_neighbors = 200

*we're not preserving distances directly — we're preserving local relationships and the shape of the manifold.*

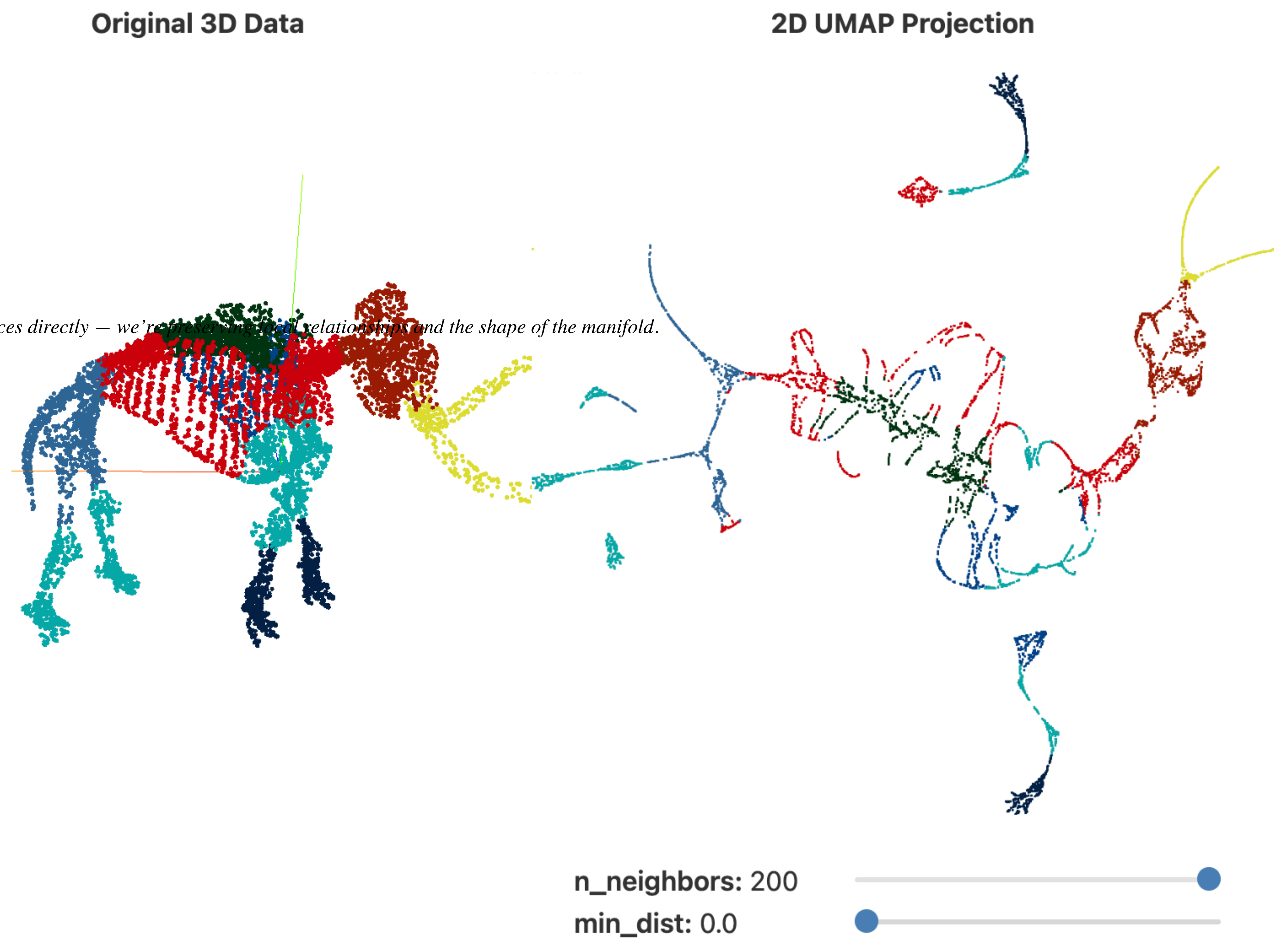



*Figure 5:* *UMAP projections of a 3D woolly mammoth skeleton (50k points, 10k shown) into 2 dimensions, with various settings for the `n_neighbors` and `min_dist` parameters.*

# min_dist

## n_neighbors = 3 | min_dist = 0.99

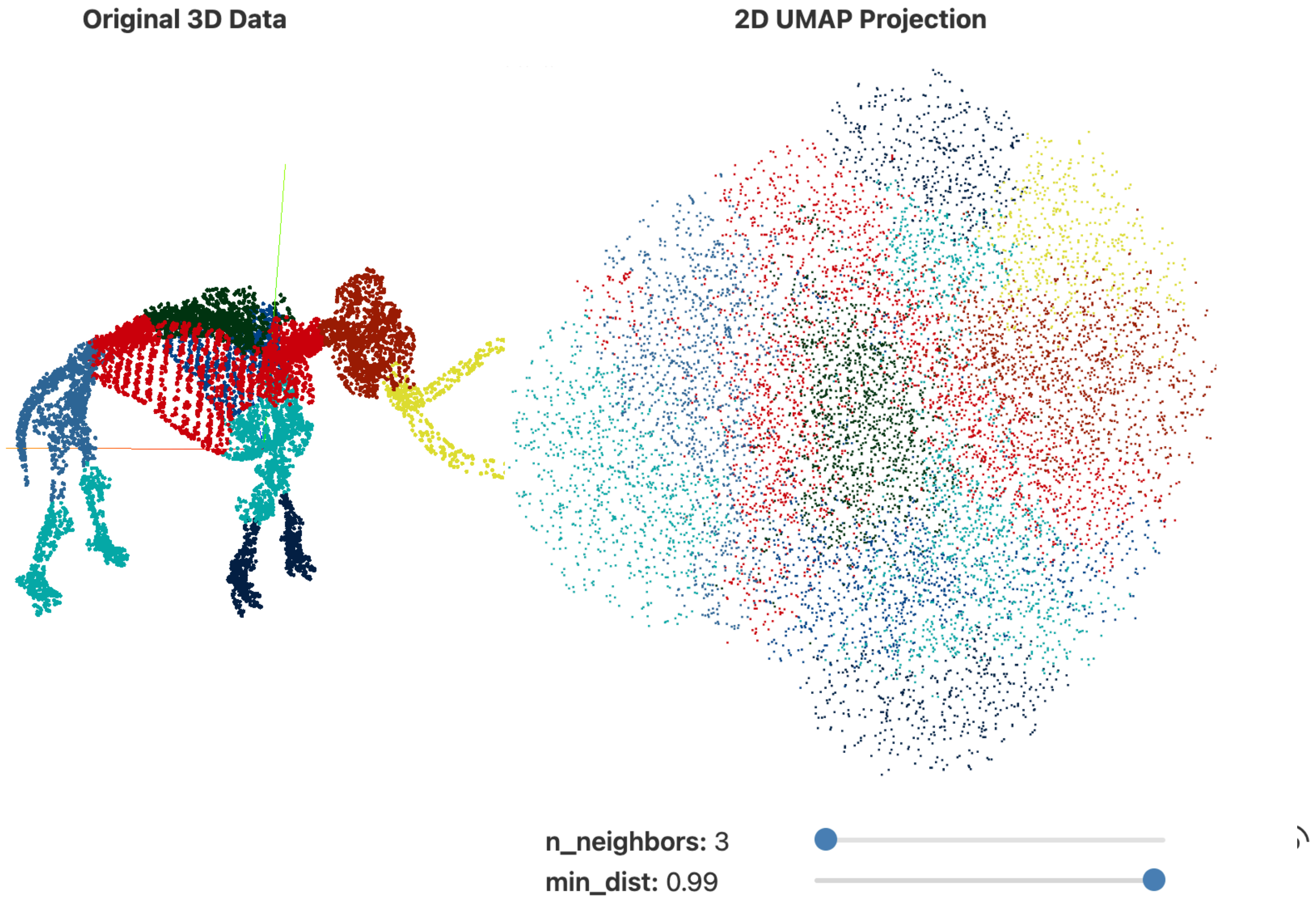



*Figure 5: UMAP projections of a 3D woolly mammoth skeleton (50k points, 10k shown) into 2 dimensions, with various settings for the `n_neighbors` and `min_dist` parameters.*

## n_neighbors = 200 | min_dist = 0.99

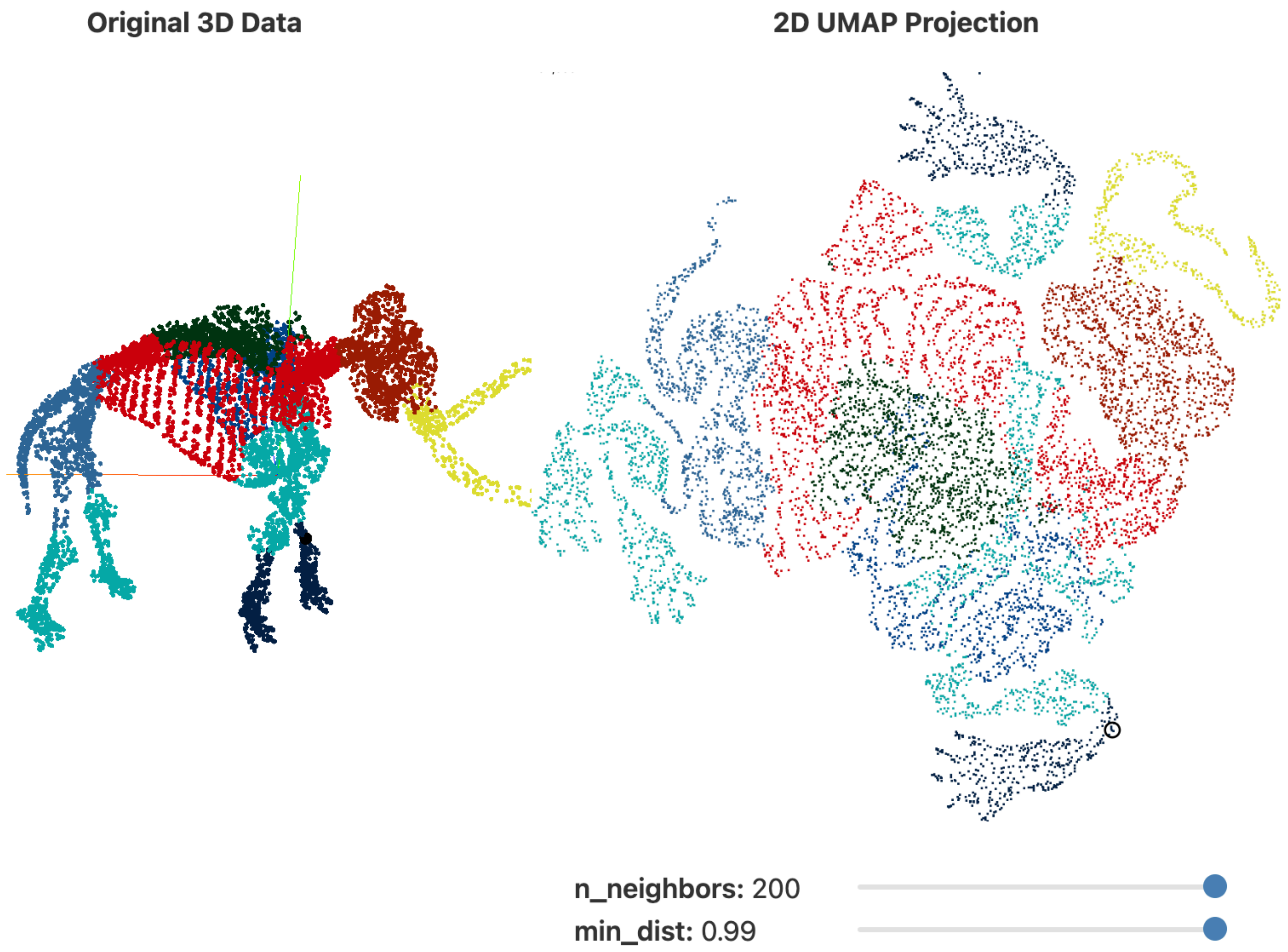
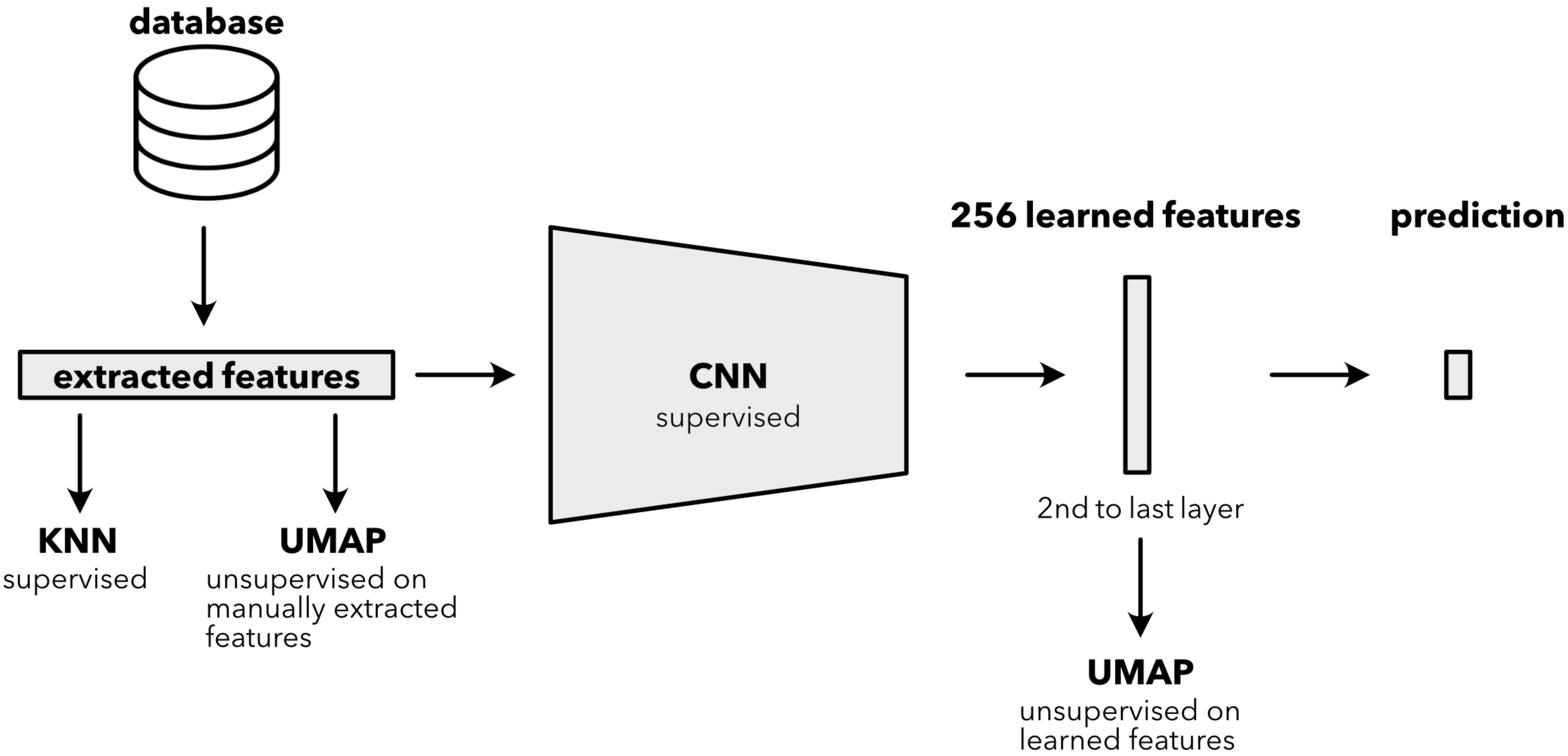
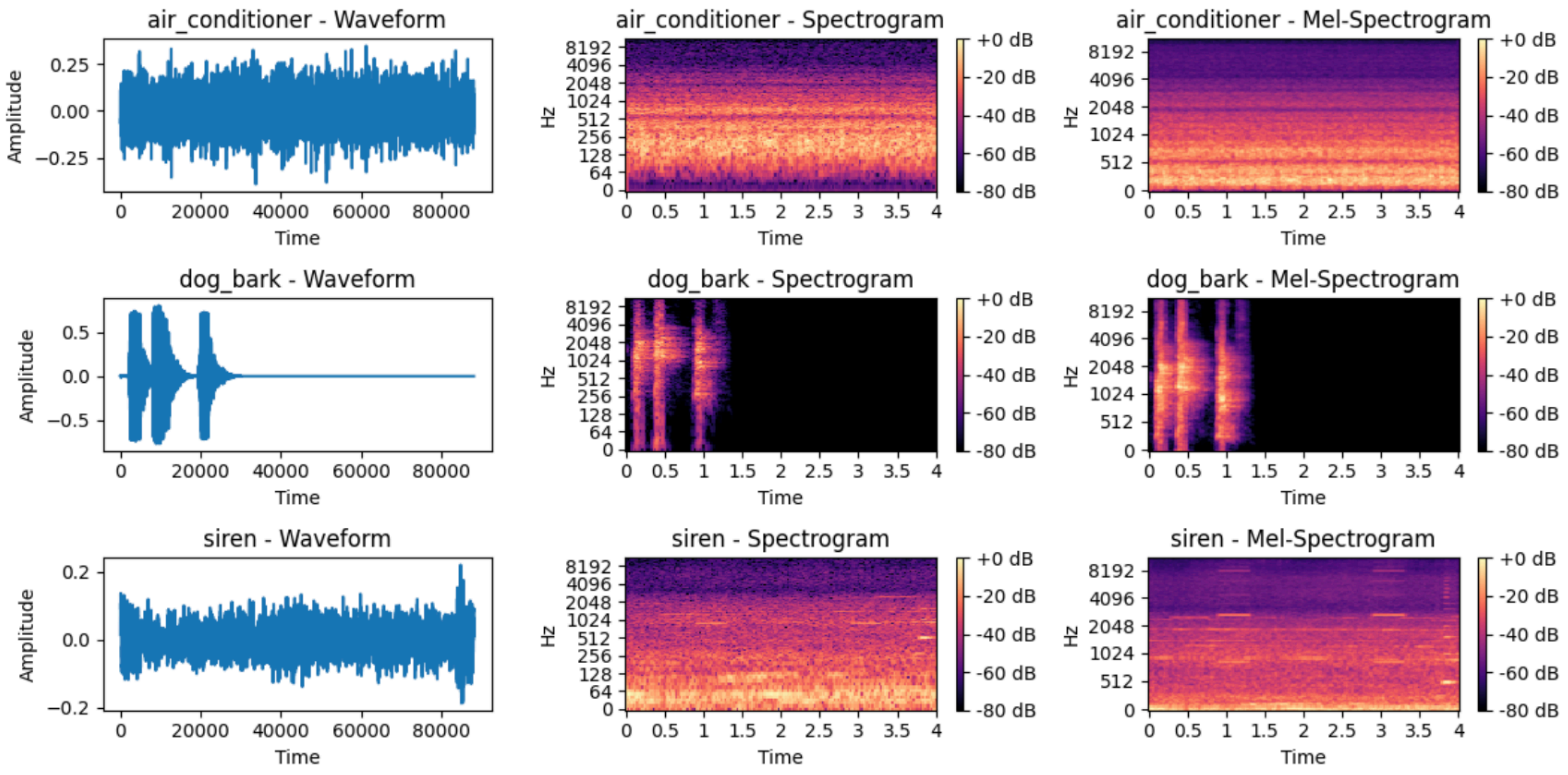



*Figure 5: UMAP projections of a 3D woolly mammoth skeleton (50k points, 10k shown) into 2 dimensions, with various settings for the `n_neighbors` and `min_dist` parameters.*

# The exercise

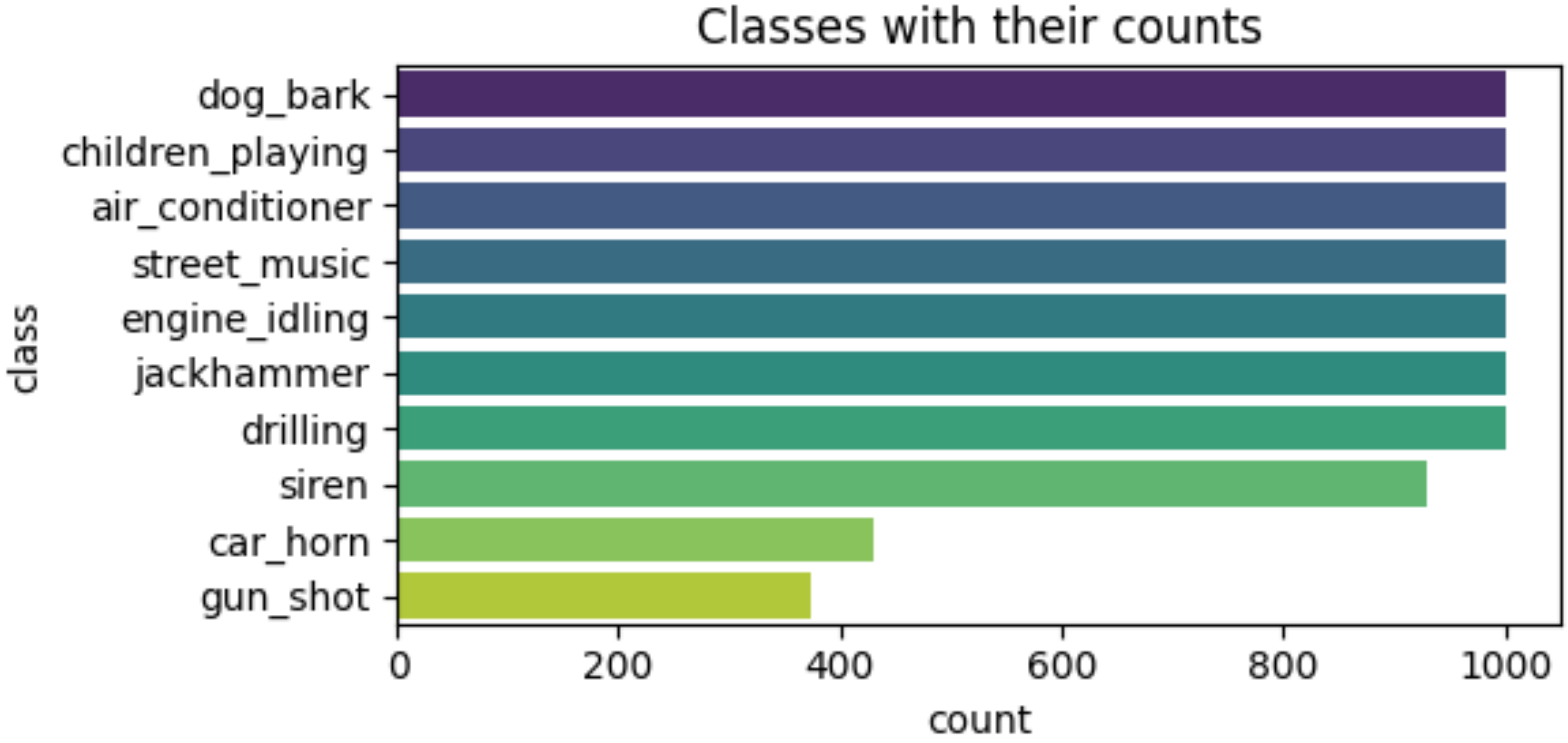


# The database



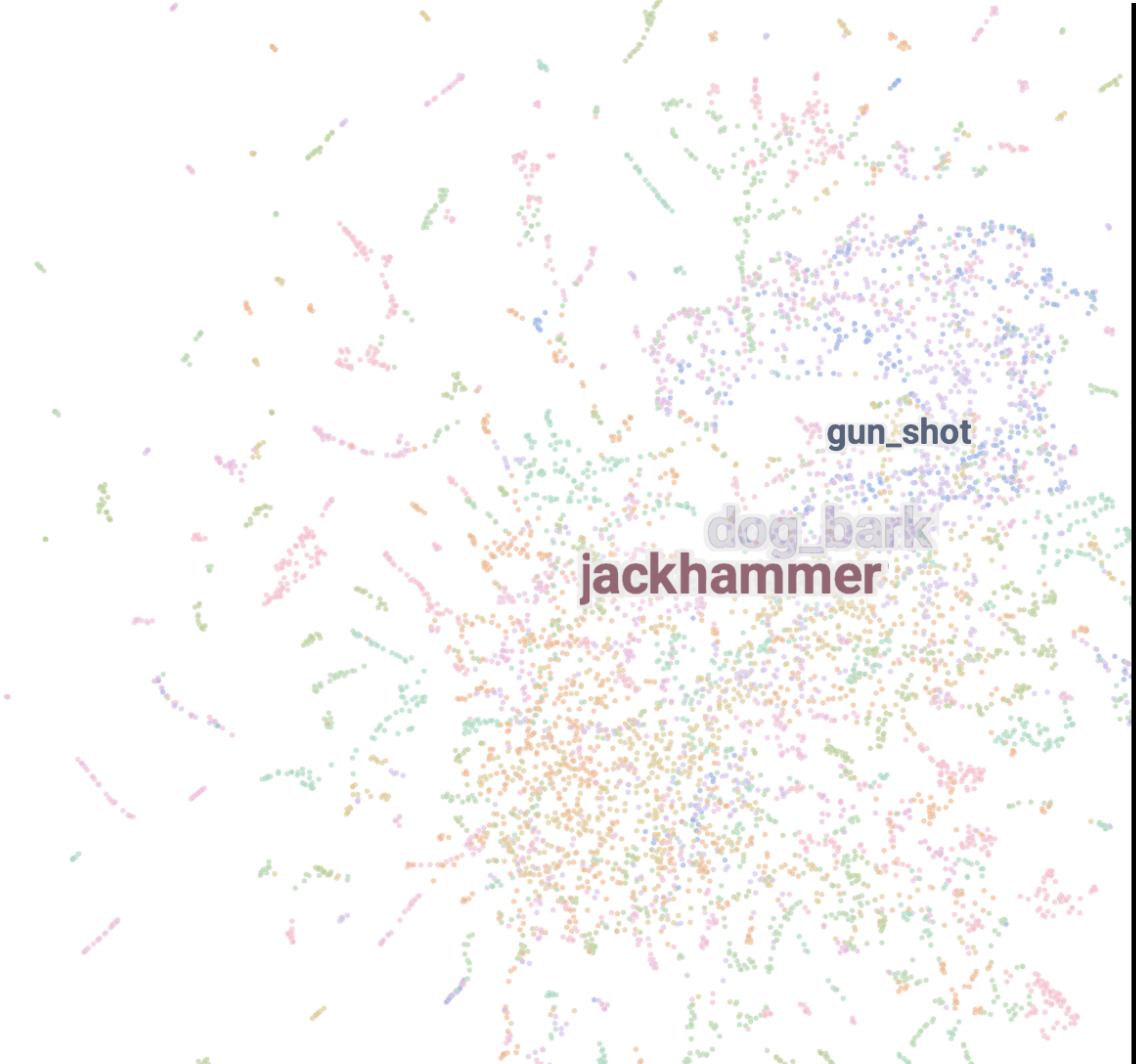

https://www.kaggle.com/datasets/chrisfilo/urbansound8k?resource=download

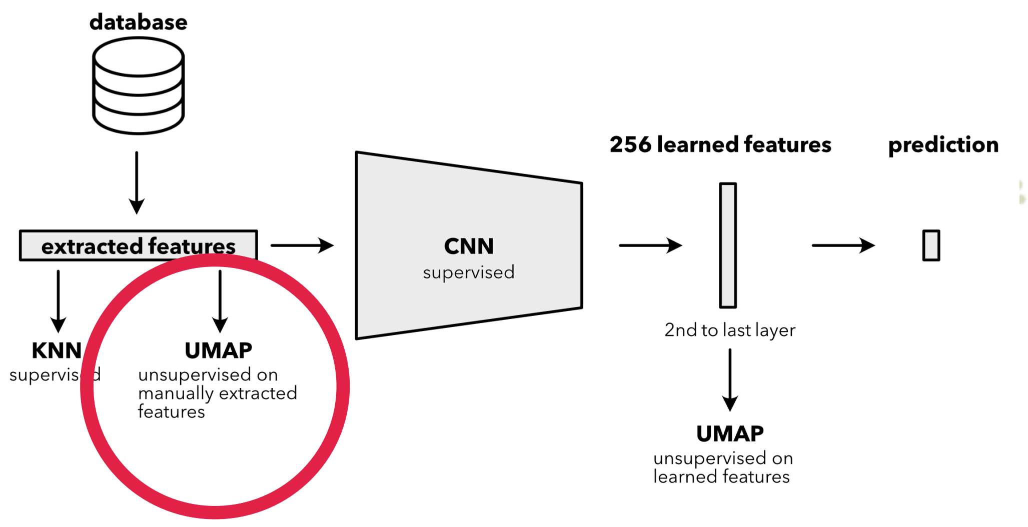
# The dataset

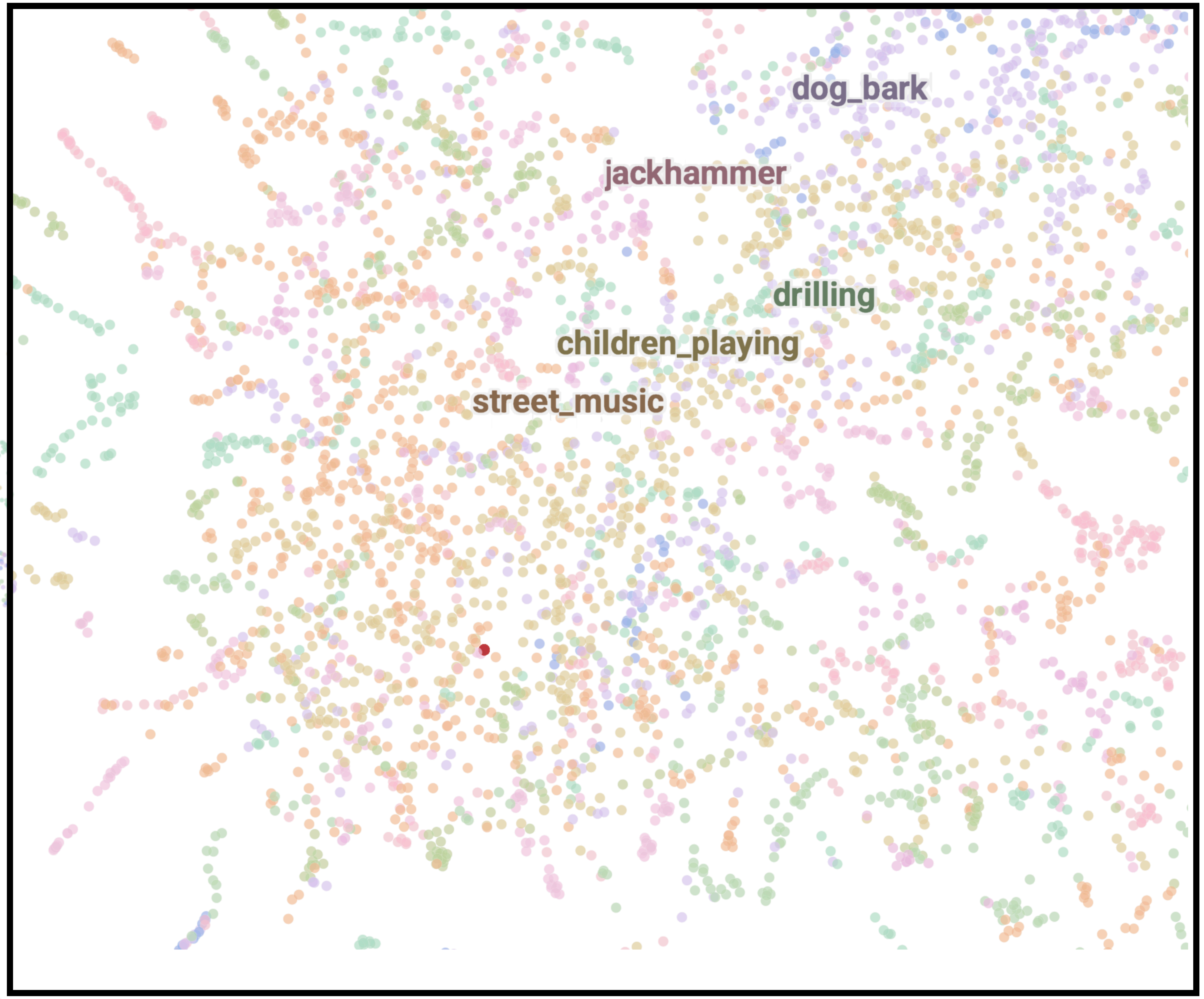


Classes with their counts
dog_bark
children_playing
air_conditioner
street_music
engine_idling
jackhammer
drilling
siren
car_horn
gun_shot
class
0
200
400
600
800
1000
count

# UMAP with extracted features



# UMAP with learned features