Introduction

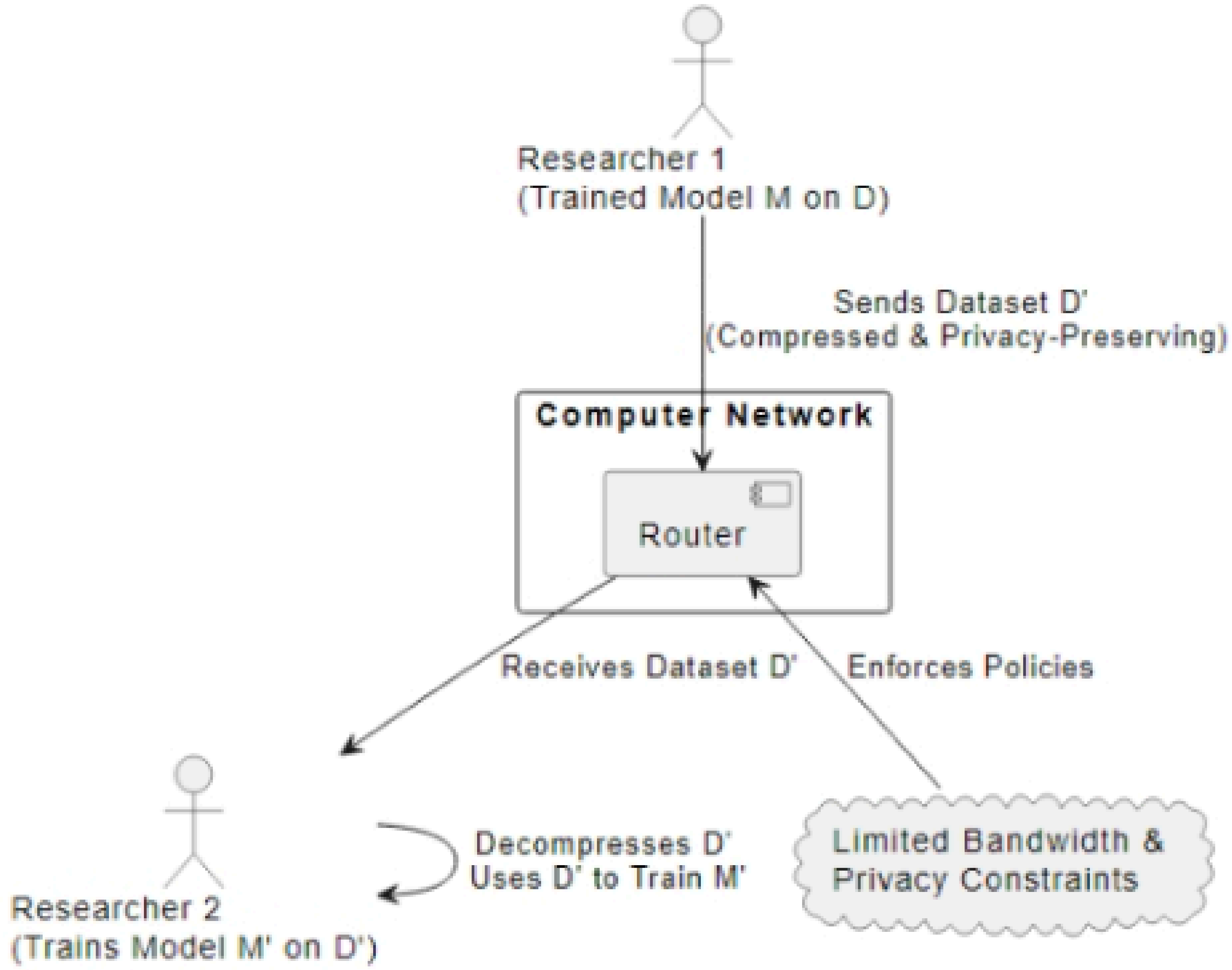
Latent Gaussian Compression

Problem Setup

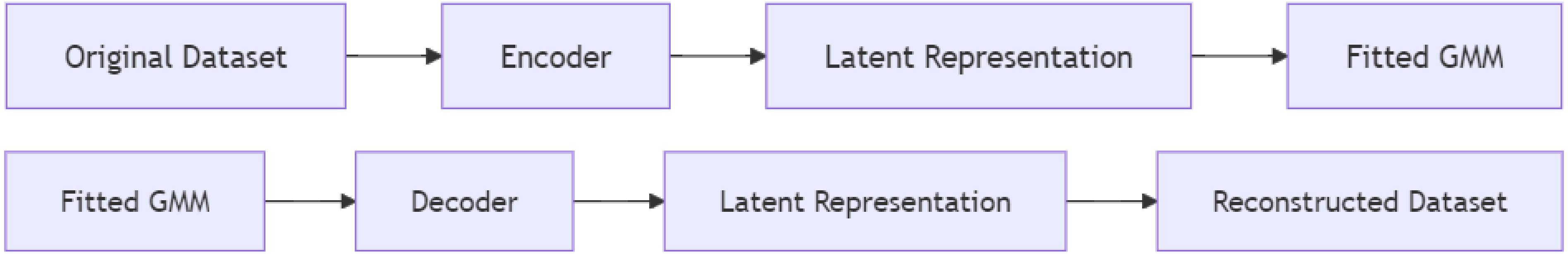
Suppose we have a dataset $D = \{Cat, Dog\}$ with two classes and we want to train a classifier.

- **The Problem**:
  - Cannot store or transmit full dataset D because of
    - Network bandwidth constraints.
    - Space constraints
    - Privacy constraints.
- Can we share compressed dataset D' instead?

Problem Assumptions



Dataset Assumptions

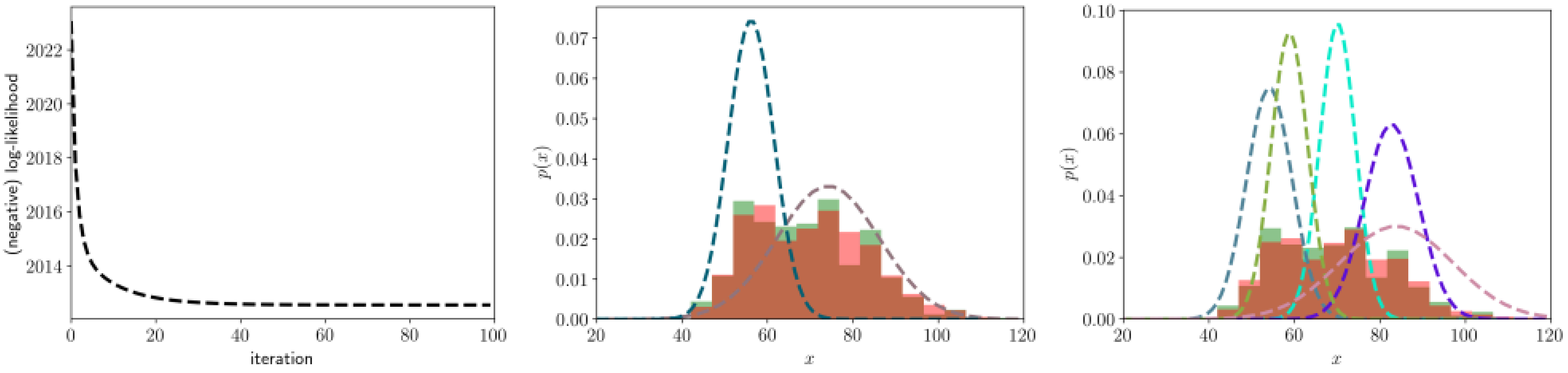


Gaussian Mixture Modeling

- In the reduced space $R^k(A', B')$, the data looks smoother than in R^n.
- We can approximate the class distributions using **Gaussian Mixture Models (GMMs)**:
  - Represent the class distributions as linear combinations of Gaussian distributions.

$$N(\mu_A, \Sigma_A), \quad N(\mu_B, \Sigma_B)$$

Visualizing GMM Distribution Learning



- The image shows the learning of Gaussian Mixture Models (GMMs) with different numbers of clusters ($k = 2$ and $k = 5$).
- **GMM Objective**:
  - Learn the parameters μ_k and Σ_k to best fit the data distributions.

Compression with Autoencoders

- **Goal**: Reduce the dimensionality of the input while retaining essential information.
- Latent space acts as a compressed representation.
- Applications:
  - Image compression.
  - Dimensionality reduction.

Encoder-Decoder Architecture

Step 1: Projection

1. Use an **encoder** to project the dataset $R^n(A, B)$ into a lower-dimensional space R^k where $k \ll d$.
2. Reconstruct the dataset back to R^n.

Compression Step

To compress the data:

- Learn μ_A, Σ_A, μ_B, Σ_B using **Gaussian Mixture Model Distribution Learning (GMM)**.
- Compressed dataset becomes:
  - $\mu_A, \Sigma_A, \mu_B, \Sigma_B$.
- The decoder maps $R^k \to R^n$.

Reconstruction Step

Given $\mu_A, \Sigma_A, \mu_B, \Sigma_B$, and the decoder:

Step 1: Sampling

- Sample from $N(\mu_A, \Sigma_A)$ and $N(\mu_B, \Sigma_B)$.

Step 2: Pass Samples Through Decoder

- The samples are passed through the decoder to reconstruct the dataset in R^n.

Reconstructed Dataset

- The reconstructed dataset approximates the original $R^n(A, B)$.
- This approach provides an efficient way to compress, store, and reconstruct datasets.