Background:
Hyperbolic embeddings achieve excellent performance for hierarchical data structures, but are limited by numerical error when the space is represented by floating-points (the NaN problem). Standard models have unbounded error as points get far from the origin, which most likely prevents hyperbolic embeddings from widely adopted.

Motivation:
Use more integer computations to avoid float arithmetic. Consider solving the same problem in the Euclidean plane. Construct a constant-error representation by using the integer-lattice square tiling. Represent any point \( x \) in the plane by
1. integer coordinates of the square where \( x \) is located,
2. float-point coordinates of \( x \) within the square.

Representation error will be bounded and proportional to the machine epsilon of the floating-point.

Strategy:
Propose new models in various dimensions based on integer tilings, which represent any point in the space with provably bounded numerical error, further more, distance and gradient computation error is independent of how far the points are from the origin, hence solves the NaN problem.

\( L \)-tiling model is defined as the Riemannian manifold \((\mathcal{L}_n^2, g_L)\), where \( g_L \) and associated distance function are given as
\[
\mathbb{L}^d = \{ x \in \mathbb{R}^d : x^T g_L x = -1, r_\epsilon > 0 \}, \quad d_L((x, y)) = \text{arccosh}( -x^T g_L y ) .
\]

Worst case representation error using floating point arithmetic (with error bounded by \( r_\epsilon \)) is \( d_L = -\text{arccosh}( 1 + n (2\epsilon_1/2\epsilon_0 - 1) ) \), which becomes \( d_L = 4\epsilon_1 + \log(r_\epsilon) + o(r_\epsilon) \) if \( d = O(-\log(r_\epsilon)) \), where \( d_L \) is the hyperbolic distance to origin.

Empirical Results:
1. Compress embeddings: tiling-based models enable us to store the resulting embeddings with fewer bits, offer algorithms to significantly compress hyperbolic embeddings for real-world datasets (down to 2% of a Poincaré embedding on WordNet Nouns.)
2. Learn embeddings: tiling-based models enable us to compute more efficiently in integers and learn high-precision embeddings without using BigFloats. offer algorithms to learn embeddings for real-world datasets. report some results on learning, which dataset, dimension.