QENN: A Quantum Entanglement-Inspired Neural Network for Interaction and Relationship Prediction in Story Videos

Supplementary Material

A. Feature Extraction Details

This section details the pipeline for extracting three core features from i-th video segment (v_i): the visual feature ($\boldsymbol{f}_v$), the character feature ($\boldsymbol{f}_{c_j}$), and the text feature ($\boldsymbol{f}_t$).

A.1. Character Identification and Tracking

To accurately localize and track key characters, we devise a comprehensive pipeline comprising face recognition, full-body tracking, and identity association.

A.1.1. Face Detection and Identity Recognition

We utilize the InsightFace for face detection and recognition. The process relies on a pre-established character gallery, which is manually created for each film and contains named images of key characters.

The face analysis model within InsightFace processes each image in the gallery to generate a 512-dimensional feature embedding. These embeddings constitute the probe gallery for the film, along with a mapping from character names to unique numerical IDs. The FaceAnalysis model is run on every frame of the video to detect all faces and extract their corresponding 512-D embeddings.

To identify a detected face, we compute the dot product between its embedding and all embeddings in the probe gallery. The identity is then assigned to the character corresponding to the highest similarity score via an argmax operation.

A.1.2. Full-body Detection and Tracking

Concurrently, we perform multi-object tracking (MOT) to obtain anonymous full-body tracks. We leverage the MMTracking toolbox, which integrates a Faster R-CNN detector with a Feature Pyramid Network (FPN) and the DeepSORT algorithm. DeepSORT ensures track stability by combining a Kalman Filter for motion prediction with a deep learning-based model for appearance feature matching, enabling robust re-identification across occlusions.

A.1.3. Identity Association

The final step associates the identities obtained from face recognition (Sec. A.1.1) with the anonymous full-body tracks (Sec. A.1.2). This is achieved through a track-level weighted voting mechanism.

1. **Confidence Filtering:** We first discard all face matches with a similarity score below a threshold of 0.3.
2. **Track-level Voting:** For each anonymous body track, we iterate through its frames. In each frame, we compute an overlap metric between the body bounding box and all identified face boxes, defined as Intersection Area/Face Box Area. A vote score is then calculated for each potential character ID as:

$$\text{Score} = \text{Overlap} \times \text{Similarity Score}$$

   This score is accumulated for each character ID over the entire track.
3. **Final Identity Assignment:** After processing all frames of a track, we compute the average score for each candidate character ID. The identity with the highest average score is definitively assigned to the track.

A.2. Dialogue Extraction and Scene Alignment

Dialogue is extracted by WhisperX. A multi-step preprocessing pipeline is applied to clean the raw text by removing non-dialogue annotations (e.g., [Music]), standardizing the format by merging multi-line entries, and merging subtitle segments that are temporally close or overlapping. The cleaned dialogue is then temporally aligned with predefined scene boundaries.

A.3. Feature Vector Generation

A.3.1. Visual and Character Features

We employ a Temporal Shift Module (TSM) model with a ResNet-50 backbone, pre-trained on the Kinetics-400 dataset, for spatiotemporal feature extraction.

To represent the overall visual context, the video content of an entire scene is fed into the TSM model. The model extracts a sequence of features across the temporal dimension. A max pooling operation is then applied to this sequence to aggregate them into a single feature vector $\boldsymbol{f}_v$ for the scene.

To represent a specific character, we use their identified track from Sec. A.1. The bounding box regions are cropped from the original frames to create a character-centric temporal image sequence. This sequence is fed into the same TSM model, and the resulting feature sequence is similarly aggregated via max pooling to produce a single feature vector $\boldsymbol{f}_{c_j}$ for character c_j.

A.3.2. Text Feature

Text features are generated using the Bert model. For each scene, we process its dialogue on a sentence-by-sentence basis. Each sentence is independently fed into the BERT model, and we extract the 768-dimensional embedding of the [CLS] token. All sentence embeddings from the scene are then stacked and aggregated via max pooling to produce a single 768-dimensional text feature vector $\boldsymbol{f}_t$.

B. Optimization on the Unitary Manifold

This appendix details the Riemannian optimization procedure used to train the learnable unitary matrices $\mathbf{U}$ in our model, ensuring the constraint $\mathbf{U}^\dagger\mathbf{U} = \mathbf{I}$ is strictly maintained. We treat the space of unitary matrices as a Riemannian manifold and employ a Riemannian gradient descent algorithm.

The core algorithm consists of three steps at each iteration: (1) calculating the Riemannian gradient by projecting the standard Euclidean gradient onto the tangent space of the manifold, (2) determining the update direction, and (3) moving along the manifold via a retraction operation.

B.1. Riemannian Gradient Calculation

The first step involves computing the standard Euclidean gradient of the loss function $f(\mathbf{U})$, denoted as $\nabla_E f(\mathbf{U})$, which is obtained via backpropagation. However, this gradient does not reside in the tangent space $T_{\mathbf{U}}\mathcal{U}(n)$ of the unitary manifold at point $\mathbf{U}$. The tangent space, which contains all valid "directions" of movement from $\mathbf{U}$, is defined as:

$$T_{\mathbf{U}}\mathcal{U}(n) = \{\mathbf{Z} \in \mathbb{C}^{n\times n} \mid \mathbf{U}^\dagger\mathbf{Z} + \mathbf{Z}^\dagger\mathbf{U} = 0\} \tag{13}$$

The Riemannian gradient, $\mathrm{grad} f(\mathbf{U})$, is the orthogonal projection of the Euclidean gradient $\nabla_E f(\mathbf{U})$ onto this tangent space. The projection is calculated as follows:

$$\mathrm{grad} f(\mathbf{U}) = \nabla_E f(\mathbf{U}) - \mathbf{U} \cdot \mathrm{sym}(\mathbf{U}^\dagger \nabla_E f(\mathbf{U})) \tag{14}$$

where $\mathrm{sym}(\mathbf{A}) = \frac{1}{2}(\mathbf{A} + \mathbf{A}^\dagger)$ is the Hermitian part of a matrix $\mathbf{A}$. This calculation maps the Euclidean gradient to the Riemannian gradient.

B.2. Update via Retraction

With the Riemannian gradient $\mathrm{grad} f(\mathbf{U})$ computed, we determine the update vector in the tangent space, $\boldsymbol{\eta}_k = -\alpha_k \mathrm{grad} f(\mathbf{U}_k)$, where α_k is the learning rate.

A simple additive update $\mathbf{U}_k + \boldsymbol{\eta}_k$ would move the point off the manifold. Instead, we use a retraction, a map $R_{\mathbf{U}_k}(\cdot)$ that takes the tangent vector $\boldsymbol{\eta}_k$ and maps it back to the manifold, yielding the updated point $\mathbf{U}_{k+1}$.

$$\mathbf{U}_{k+1} = R_{\mathbf{U}_k}(\boldsymbol{\eta}_k) \tag{15}$$

Our implementation utilizes an efficient and robust retraction based on the QR decomposition:

$$R_{\mathbf{U}}(\mathbf{Z}) = \mathrm{qr}(\mathbf{U} + \mathbf{Z}) \tag{16}$$

In the final step, a phase correction is applied by multiplying $\mathbf{Q}$ with the phases of the diagonal elements of $\mathbf{R}$ to ensure a unique and stable retraction.

B.3. Algorithm Summary

The complete optimization process is summarized in Algorithm 1.

Algorithm 1 Riemannian Gradient Descent on the Unitary Manifold

Initialize: Choose an initial unitary matrix $\mathbf{U}_0 \in \mathcal{U}(n)$.
for $k = 0, 1, 2, \ldots$ **do**
  Compute the Euclidean gradient: $G_k = \nabla_E f(\mathbf{U}_k)$ via backpropagation.
  Project to tangent space to get the Riemannian gradient (Eq. 14):
    $g_k = G_k - \mathbf{U}_k \cdot \mathrm{sym}(\mathbf{U}_k^\dagger G_k)$.
  Choose a learning rate α_k.
  Compute the update vector in the tangent space: $\boldsymbol{\eta}_k = -\alpha_k g_k$.
  Update the matrix by retracting to the manifold (Eq. 16):
    $\mathbf{U}_{k+1} = \mathrm{qr}(\mathbf{U}_k + \boldsymbol{\eta}_k)$.
end for
return $\mathbf{U}$.