The paper describes a Quantum Hybrid Neural Network approach to solve the escape routing problem. The network is trained on data obtained from running node-wise Djiksta's algorithm on a simulated earthquake scenario.

The model of the Earthquake evolves dynamically as the parameter they've chosen to represent the area of effect of the earthquake changes with time and distance to the epicenter of the earthquake.

Using data obtained from a real city, representing its road layouts with a directed weighted graph, the model simulates an earthquake by randomly sampling a point as the earthquake's epicenter and having fixed exits. Then the simulation takes random starting points and runs Node-wise Djikstra's algorithm in order to find the quickest path to an exit point.

The weights of the graph take into account, both the area of effects of the earthquake and traffic pile-up at exits. It does so by employing the following formulas:

- $r_{\text{epi}} = 0.5 + \sqrt{0.0002 \times t}.$
  Describes the expansion of the area of effect of the Earthquake. i.e. a radius that increases over time.
- $$w \leftarrow \begin{cases} w \times 5, & \text{if } d_{\text{epi}} \leq 0.3\ r_{\text{epi}} \\ w \times 2, & \text{if } 0.3\ r_{\text{epi}} < d_{\text{epi}} \leq 0.75\ r_{\text{epi}} \\ w \times 1.3, & \text{if } 0.75\ r_{\text{epi}} < d_{\text{epi}} \leq r_{\text{epi}} \\ w, & \text{otherwise.} \end{cases}$$
  The weights of the edges of the graph initially are biased based on varying degrees according to euclidean distance from the epicenter.
- $$w \leftarrow \begin{cases} \min\{w \times \sqrt{0.003 \times t + 1}\ ,\ 5\}, \\ \quad \text{if } d_{\text{epi}} \leq 0.3\ r_{\text{epi}} \\ \min\{w \times \sqrt{0.002 \times t + 1}\ ,\ 4\}, \\ \quad \text{if } 0.3\ r_{\text{epi}} < d_{\text{epi}} \leq 0.75\ r_{\text{epi}} \\ \min\{w \times \sqrt{0.001 \times t + 1}\ ,\ 3\}, \\ \quad \text{if } 0.75\ r_{\text{epi}} < d_{\text{epi}} \leq r_{\text{epi}} \\ w, \quad \text{otherwise,} \end{cases}$$
  The subsequent increase of weights due to earthquake effects are governed by these equations. Note that all the weights have caps in order to avoid their values blowing up to large numbers.

$$w \leftarrow \begin{cases} \min\{w \times \sqrt{0.03 \times t + 1}\ ,\ 5\}, \\ \qquad \text{if } d_{\text{exit}} \leq 0.5\ r_{\text{exit}} \\ \min\{w \times \sqrt{0.02 \times t + 1}\ ,\ 4\}, \\ \qquad \text{if } 0.5\ r_{\text{exit}} < d_{\text{exit}} \leq 0.75\ r_{\text{exit}} \\ \min\{w \times \sqrt{0.01 \times t + 1}\ ,\ 3\}, \\ \qquad \text{if } 0.75\ r_{\text{exit}} < d_{\text{exit}} \leq r_{\text{exit}} \\ w, \qquad \text{otherwise} \end{cases}$$

with $r_{\text{exit}} = \sqrt{0.00075 \times t}$.

- Similarly, the traffic effect at exit points are governed by these equations.

Two heuristic indicators are chosen to indicate global information. These can be conceptualized as the following questions:

- How far away from the exit point am I? – The euclidean distance

$$d(p, q) = \sqrt{(q_x - p_x)^2 + (q_y - p_y)^2}$$

- Am I traveling in the right direction? – The cosine distance

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\,\|\mathbf{B}\|}$$

For a node p and a neighbor node q we define A and B as: $A = (q_x - p_x, q_y - p_y)$ $B = (\text{exit}_x - p_x, \text{exit}_y - p_y)$.

The inputs for the model are detailed as follows:

TABLE I: Model input summary

1) Earthquake coordinates	$x_{\text{epi}}, y_{\text{epi}}$
2) Start node coordinates	$x_{\text{start}}, y_{\text{start}}$
3) Destination coordinates	$x_{\text{dest}}, y_{\text{dest}}$
4) End of edge coordinates	$x_{\text{edge}_n}, y_{\text{edge}_n}$
5) Required travel time	w_n
6) Edge betweenness centrality	e_n
7) Euclidian distance	d_n
8) Cosine distance	c_n

The features from 4 to 8 have values of n from 1 to 5 as the degree of the graph is 5, i.e. a given node can have upto 5 edges, i.e. 5 options the model has to choose from. Thus the model has 36 inputs (6+5*6).

The model is evaluated on the basis of:

$$\text{Arrival rate} = \frac{\text{No. instances path found}}{\text{No. sampled node pairs}}$$

- The probability of finding a path between a random start node and exit node.

$$\text{Accuracy} = 1 - \left| 1 - \frac{\sum_{\text{weight}} \text{path}_{\text{Dij}}}{\sum_{\text{weight}} \text{path}_{\text{model}}} \right|$$

- The performance of the model relative to the results from Node-wise Dijkstra's algorithm it is trained on.

The model can choose to include the earthquake's epicenter coordinates in the same input layer as the rest of the parameters. But this would cause the model to generate output for fixed earthquake coordinates, thus reducing the variability of the model.. To combat this multiple neural networks can be employed for each earthquake coordinate, but that is obviously resource intensive.

A solution to both these issues is presented by the paper by proposing the use of Feature wise Linear Modulation a.k.a a FiLM layer. This allows us to take random samples of earthquake coordinates and modulate them into the overall neural network.

"The FiLM layer exploits the earthquake coordinates and modulates the traditional neural network layer to guide the prediction of the subsequent routing node."

The Hybrid Quantum Neural Network (HQNN) combines a classical NN with a variational quantum circuit (VQC). The VQC takes in the state features and the earthquake as input and outputs a list of expected values of the variational quantum states. The FiLM inputs are the random earthquake coordinates.

The FiLM section contains two qubits (earthquake x and y coordinates) and accepts the coordinates using data reuploading with the help of Z gates. Encoding gates are repeated 5 times (for edges) and interlaced with variational unitaries using 4 sub-layers of X gates and CNOT gates. This is the Basic Entangler Layer (BEL).

The main section accepts the rest of the variables by embedding the state data into a 5 qubit feature parameter quantum depth-infused layer. This layer breaks down the main input vector into subvectors of the size of the number of qubits such that - $n_{\text{features}} \leq n_{\text{qubit}} \times n_{\text{subvec}}$.

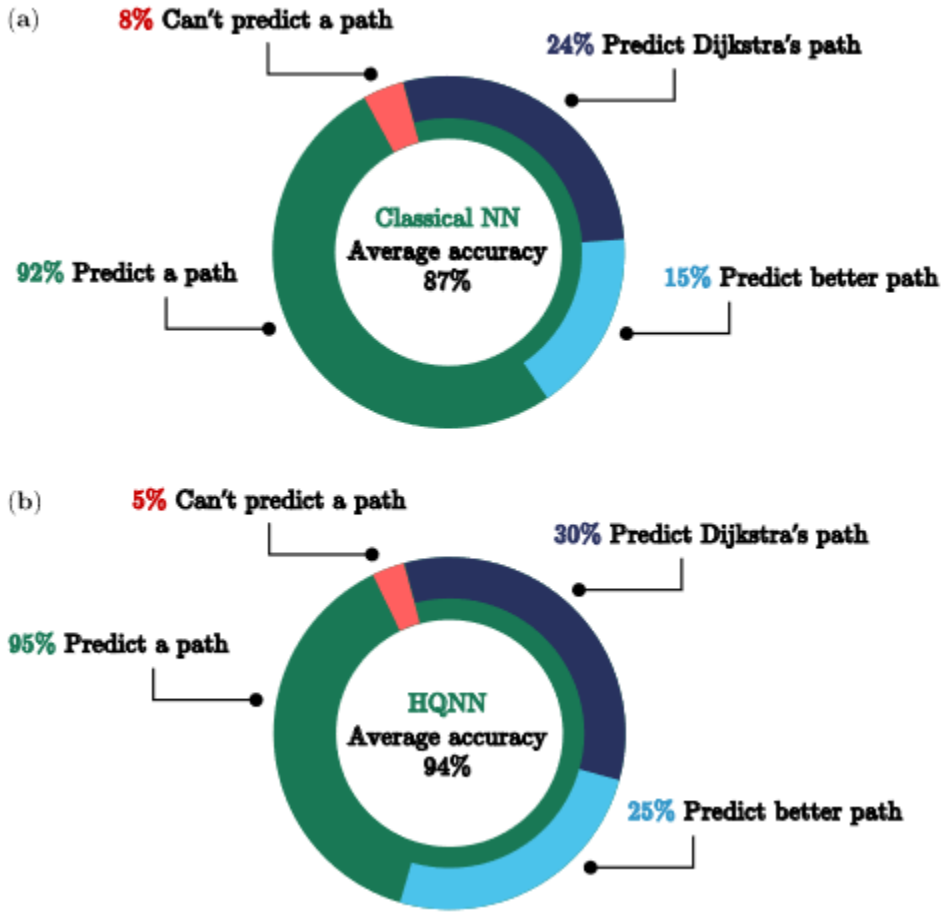
A layer of Z gates encodes the first layer of the first feature subvector, followed by another variational layer and this is repeated till all subvectors are encoded.

The two FiLM qubits are then entangled with the main section using CNOT gate, controlled using the two qubits and the NOT is applied to all the qubits in the state. Finally, a main BEL is applied to the state cubits.

The five main cubits are then measured many times to produce a list of bit strings, each of length five. Then the expected value of each qubit is measured separately.

Finally, the classical network will have five output nodes, among which the neighboring node corresponding to the highest node is chosen as the node to travel to.

Dijkstra's algorithm has a time complexity of O(E+VlogV) while HQNN only has a time complexity of O(V). An analysis and comparison of the results of a Classical NN and a HQNN are shown below:



The paper concludes that the hybrid supervised learning approach shows promising results and future research needs to focus on testing this approach on different and larger graphs.

Potential Critiques:

The paper is not based on data obtained from real life disasters, but rather a mathematically ideal representation of an Earthquake. Whether the simulated modeling of the earthquake the paper has provided is viable to apply to real earthquakes is debatable. The paper also chose to focus on earthquakes while neglecting natural disasters that are geometrically complex, like tsunamis, flash floods, etc.

The values used to set up the simulated earthquake are seemingly arbitrary with no explanation as to why those particular values were chosen.