Quanvolutional Neural Network

Shreevardhan Shah, Amol Harsh, Hriday Rathi Github

***Abstract*— This project explores the application of quantum frameworks within deep learning, specifically focusing on Quantum Convolutional Neural Networks (QCNNs) for computer vision tasks. Inspired by prior research (Henderson et al., 2019), we replicated and extended the concept of Quanvolutional Neural Networks, which integrate quantum circuits into the convolutional layers of traditional neural networks to process image data. Using the Fashion MNIST dataset, we conducted experiments to evaluate the performance of different quantum convolution kernel sizes (2x2, 3x3, and 4x4) on image recognition tasks. Our findings reveal that smaller quantum kernels, particularly 2x2, achieve higher validation accuracy by capturing intricate patterns more effectively than larger kernels or traditional convolutional neural networks.**

Despite computational constraints, our work demonstrates the promising capabilities of QCNNs and provides new insights into optimizing quantum-enhanced neural network architectures for improved performance in image recognition tasks. Future research will focus on leveraging more powerful quantum devices and extending our approach to colored images and larger datasets.

I. Introduction and Literature Review

The integration of quantum computing and deep learning marks a revolutionary shift in the landscape of artificial intelligence, particularly in the field of computer vision. The emerging field of Quantum Convolutional Neural Networks (QCNNs) exemplifies this convergence by combining classical convolutional neural networks (CNNs) with quantum circuits. This hybrid approach aims to leverage the unique capabilities of quantum computing to enhance the processing and interpretation of image data. The primary objective of this project is to delve into the potential of quantum frameworks within deep learning, focusing on the innovative concept of "quanvolutional" neural networks.

Challenges with Classical Computing

Quantum computing promises to address some of the most significant challenges faced by classical computing, particularly in handling complex, high-dimensional data. Quantum Convolutional Neural Networks (QCNNs) are designed to capitalize on the principles of quantum mechanics to improve the efficiency and accuracy of deep learning models. The foundational concept of QCNNs is extensively detailed in the seminal paper by Maxwell Henderson et al., titled "Quanvolutional Neural Networks: Powering Image Recognition with Quantum Circuits" (arXiv:1904.04767, 2019). This paper introduces the innovative approach of using quantum circuits to process input images, thereby creating new feature representations that can be fed into neural networks for training and prediction.We will briefly talk about this process below:

Process of Quanvolution

The process of quanvolution involves three key stages:

Encoding: This stage involves converting classical pixel values into quantum states. Techniques such as binary threshold encoding or rotation gates are employed to achieve this transformation. For instance, pixel values can be encoded into quantum states using rotation gates, where the pixel value determines the rotation angle applied to the quantum state.

Quantum Circuit Transformation: In this stage, quantum gates are applied to the encoded quantum states to transform them. These transformations leverage the principles of quantum superposition and entanglement to create new feature representations that classical methods might struggle to produce.

Decoding: The final stage involves measuring the transformed quantum states and converting them back into classical values. These values form a feature map that can be used by traditional neural networks for further processing and prediction.

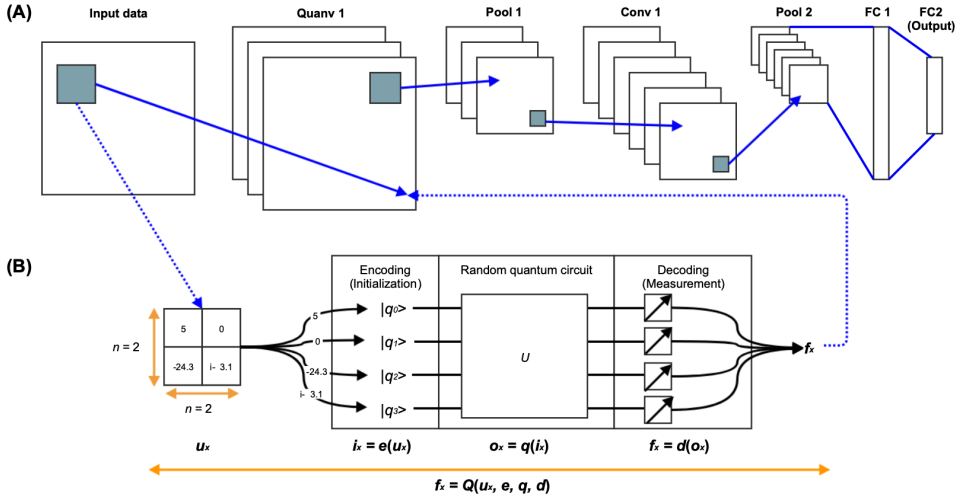


***Fig. 1.**: A. Simple example of a quanvolutional layer in a full network stack. The quanvolutional layer contains several quanvolutional filters (three in this example) that transform the input data into different output feature maps. B. An in-depth look at the processing of classical data into and out of the random quantum circuit in the quanvolutional filter.*

Reference: https://arxiv.org/abs/1904.04767

Walkthrough of the Process:
For simplicity, let's say we are applying quanvolutional on a 2x2 patch from a 8 bit grayscale image.

$$u_x = \begin{bmatrix} 200 & 50 \\ 30 & 100 \end{bmatrix}$$

Step 1: Encoding: Convert the classical pixel values of the input patch into quantum states (qubits).

Process 1: Encode each pixel value into a quantum state. For simplicity, let's use a binary threshold encoding where pixel values greater than a threshold (128) are encoded as $|1\rangle$, and values less than or equal to the threshold are encoded as $|0\rangle$.

Process 2: We can map the values as follows:

- $200 > 128 \rightarrow |q_o\rangle = |1\rangle$
- $50 < 128 \rightarrow |q_1\rangle = |0\rangle$
- $30 < 128 \rightarrow |q_2\rangle = |0\rangle$
- $100 < 128 \rightarrow |q_3\rangle = |0\rangle$

Thus, the encoded quantum state vector is: $i_x = |1000\rangle$

Step 2: Quantum circuit transformation: Apply random quantum gates to the encoded quantum states to transform them , Here we

- Apply a CNOT gate where q0 controls q1:
  - $CNOT_{q0,q1}|1000\rangle = |1100\rangle$
- Apply a CNOT gate where q1 controls q2:
  - $CNOT_{q1,q2}|1100\rangle = |1110\rangle$

The final output quantum state after the circuit o_x: $o_x = q(i_x) = q(|1000\rangle) = |1110\rangle$

Step 3: Decoding:
Objective: convert the transformed quantum states back into classical values (scalars) to form the feature map.

- Process 1: Measurement: The measurement collapses the quantum state into a classical bitstring. For our example: Measure $|1110\rangle$ results in 1110
- Process 2: Post-Processing: Convert the bitstring to a scalar value. For simplicity, let's sum the measured bits:
  Sum of bits in 1110 = 1 + 1 + 1 + 0 = 3

The final decoded value f_x is: $f_x = d(o_x) = 3$

This same Quanvolutional is done for the entire image grid and then the newly created feature space is fed to the neural network. (The paper had a stride value of 1)

We also read and then implemented an article written by Andrea Mari from the PennyLane community (3). Where he also implemented Quanvolutional using the same three processes that we described above:
It was just that they used different methods under some of the above-mentioned processes. We have used the same mechanism described below for our own experimentations:

Let's take the help of the same **quanvolutional 2x2 patch to walk through their process:**

Step 1: Encoding:

- **Process 1** (Normalization) Normalize the pixel values to the range [0, 1] by dividing by 255:

$$\text{Normalized } u_x = \begin{bmatrix} \frac{200}{255} & \frac{50}{255} \\ \frac{30}{255} & \frac{100}{255} \end{bmatrix} = \begin{bmatrix} 0.784 & 0.196 \\ 0.118 & 0.392 \end{bmatrix}$$

- **Process 2** (Mapping to Quantum States): Encode each pixel value into a quantum state using an RY gate, where the pixel value is multiplied by π to determine the rotation angle.
  We can map the values as follows:
  - 0.784 → RY ($\pi \cdot 0.784$)
  - 0.196 → RY ($\pi \cdot 0.196$)
  - 0.118 → RY ($\pi \cdot 0.118$)
  - 0.392 → RY ($\pi \cdot 0.392$)

Thus, the encoded quantum state vector using RY gates is:
i_x = [RY ($\pi \cdot 0.784$) , RY ($\pi \cdot 0.196$) , RY ($\pi \cdot 0.118$) , RY ($\pi \cdot 0.392$)]

Let's see one example of ϕ_0 how we are applying the Ry gates

$$\mathrm{RY}(\theta) = \begin{bmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{bmatrix}$$

$$\theta_0 = \pi . 0.784 = 2.463\, radians$$

The RY gate applied to the qubit $|q_0\rangle$:

$$\mathrm{RY}(2.463) = \begin{bmatrix} \cos(2.463/2) & -\sin(2.463/2) \\ \sin(2.463/2) & \cos(2.463/2) \end{bmatrix}$$

Similarly, we will apply it to other ϕ_1, ϕ_2, and ϕ_3 values.
And get the following Quantum State after applying the RY Gates:

Let's consider the initial state of each qubit to be $|0\rangle$

Qubit $|q0\rangle$: RY(2.463 radians) $|0\rangle$ = cos(1.231) $|0\rangle$+sin(1.231) $|1\rangle$

Similarly, we will get the results for the other qubit $|q_1\rangle, |q_2\rangle, and\ |q_3\rangle$ values.

$|q_1\rangle$ = cos(0.308) $|0\rangle$ + sin(0.308) $|1\rangle$

$|q_2\rangle$ = cos(0.1855)$|0\rangle$+sin(0.1855)$|1\rangle$

$|q_3\rangle$ = cos(0.6155)$|0\rangle$+sin(0.6155)$|1\rangle$

Thus, the combined state of the system after encoding is:
$|q\rangle=|q0\rangle\otimes|q1\rangle\otimes|q2\rangle\otimes|q3\rangle$
$|q\rangle=(\cos(1.231)|0\rangle+\sin(1.231)|1\rangle)\otimes(\cos(0.308)|0\rangle+\sin(0.308)|1\rangle)\otimes(\cos(0.1855)|0\rangle+\sin(0.1855)|1\rangle)\otimes(\cos(0.6155)|0\rangle+\sin(0.6155)|1\rangle)$

Let's focus on the key terms for simplicity: $|\psi\rangle = \alpha_{0000}|0000\rangle + \alpha_{0001}|0001\rangle + \alpha_{0010}|0010\rangle + \cdots + \alpha_{1111}|1111\rangle$ Let this be our encoded quantum state

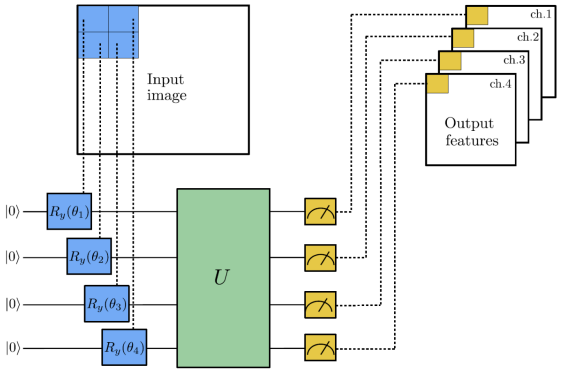
Step 2: Quantum circuit transformation: Apply random quantum gates to the encoded quantum states to transform them. Similar to how we did previously. Let that transformed quantum state be called: $|\psi_{final}\rangle$

Step 3: Decoding:
After applying these operations, we measure the qubits in the Z-basis. This means we measure whether each qubit is in state $|0\rangle$ or $|1\rangle$.
The expectation values of Pauli-Z for each qubit give us a measure of the "balance" between the $|0\rangle$ and $|1\rangle$ states: $\langle Z_i \rangle = \langle \psi_{final} | Z_i | \psi_{final} \rangle$
The expectation value can range between -1 and +1:
⟨Zi⟩ = +1: The qubit is always measured in state |0⟩.
⟨Zi⟩ = -1: The qubit is always measured in state |1⟩.
⟨Zi⟩ = 0: The qubit is equally likely to be measured in state |0⟩ or |1⟩ (i.e., the qubit is in a superposition state with equal probability)

Where Z_i is the Pauli-Z operator acting on qubit *i*. For example, if the measurements yield:
$\langle Z_0 \rangle = 0.6$, $\langle Z_1 \rangle = -0.2$, $\langle Z_2 \rangle = 0.1$, $\langle Z_3 \rangle = -0.4$,

The final decoded value vector f_x is: f_x = [0.6 , − 0.2 , 0.1 , − 0.4]

Each value from the above decoded value vector is then transferred to its own individual channel which acts like a feature embedding for the future neural network. As shown in the image below. We have followed the same above preprocessing steps. Moreover, we have curated this iterative way of comprehension which was not to be found in the article or the research paper.



While the research paper discusses how increasing the number of filters can enhance the performance of Quanvolutional neural networks, as shown in Figure 2, it does not address how the size of the image patch affects the validation accuracy. Therefore, we are building on the research conducted by previous researchers and presenting fresh, new insights on this topic. This will help future quantum deep learning experts make more informed decisions about architectures to achieve higher accuracy levels.

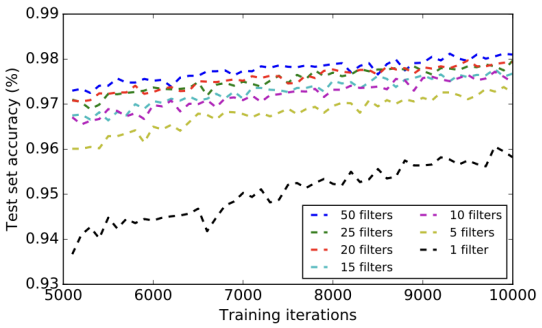


***Fig. 2.**: QNN MODEL test set accuracy results using a variable number of quanvolutional filters.*

Experiment Setup:

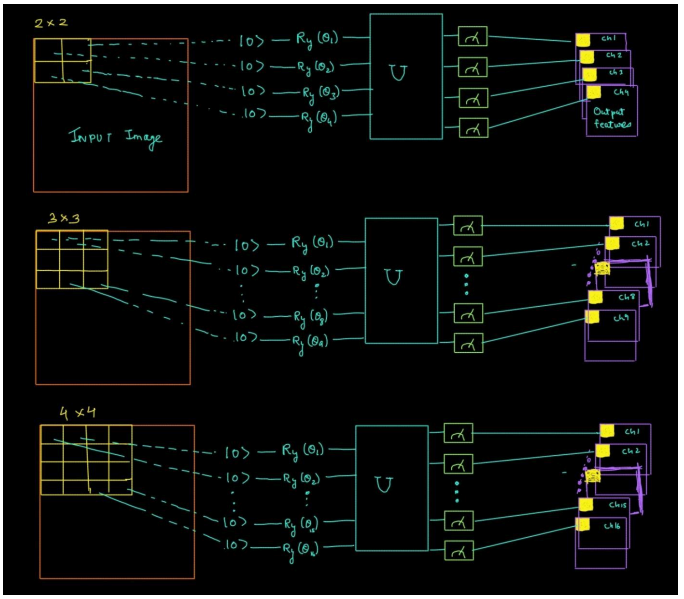
In our previous implementation of the QCNN network, we had chosen the image patch size to be 2x2, as demonstrated in the prior classification of the MNIST dataset. We wanted to explore the effects of enlarging the image patch size to 3x3, 4x4, etc., to see its impact on classification testing accuracy using Quanvolutional networks. We implement this through parametrized rotations applied to the qubits initialized in the ground state, similar to those discussed previously. Moreover, this time we are using the fashion-MNIST dataset, which is a more challenging dataset than the one used in the research paper.

Quanvolutional Filter Architecture:

We implemented three different kernel filter sizes: 2x2, 3x3, and 4x4. Each configuration employs a unique quantum circuit that applies rotations parameterized by angles derived from the pixel values of the input image patch. The configurations are detailed as follows:

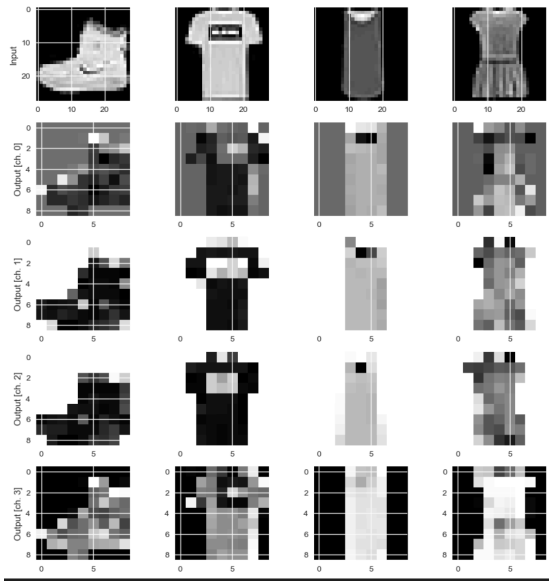
2x2 Filters: The quantum circuit for the 2x2 filter divides the input image into patches of 2x2 pixels. Each pixel value is used to parameterize a rotation on a qubit, resulting in a 4-qubit system per patch. Resulting in a total of 4 channels.

3x3 Filters: This setup increases the number of qubits to 9 per patch, with each qubit corresponding to a pixel in the 3x3 image patch. The circuit applies a rotation to each qubit based on the pixel intensity. Resulting in a total of 9 channels.



Kernel filter architecture for our 3 experimental setup

4x4 Filters: The largest configuration uses 16 qubits per image patch, allowing for a more detailed encoding of the input image into the quantum state.

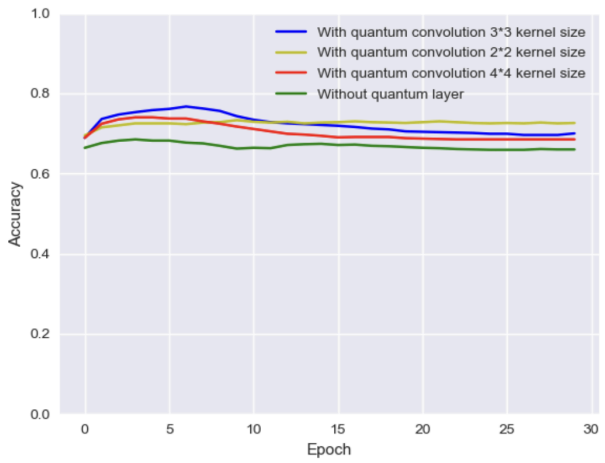


Fashion-MNIST with 2*2 filter: 4 channels

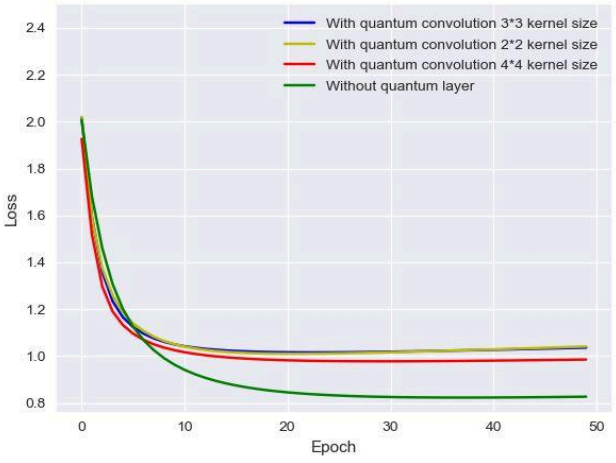
Dataset:
The fashion-MNIST dataset, which consists of 70,000 images of 10 different fashion items, was chosen for its complexity compared to the standard MNIST dataset. Each image in the dataset is 28x28 pixels, grayscale, and labeled across 10 categories. Due to limited computational resources, we used only 600 images for training and 50 for testing.

Evaluation Metrics:
The primary metric for evaluation was the classification testing accuracy. We compared the performance across different filter sizes to assess how changes in the quantum convolution (Quanvolution) layer influenced the overall effectiveness of the network in handling a more challenging dataset.

The code and further details of the implementation can be found in the attached Jupyter notebook (code in attached ipynb).



Validation Accuracy vs Epoch



Validation Loss vs Epoch

Result Overview:

From our experimentation, we observed that performing Quanvolution on smaller image patch sizes captures more intricate patterns in contextual images through state invariance, thereby yielding higher validation accuracy. In our architecture, the quantum-preprocessed images are flattened and then fully connected through a single classical neural network layer with 10 neurons, ending with a softmax layer. The actual code can be found in our GitHub repository.

Concise Analysis of Quanvolutional Experiment Results:

The plot displays the validation accuracy over 30 training epochs for four models:

- Yellow Line: QNN with a 2x2 quantum convolution kernel (image patch size).
- Blue Line: QNN with a 3x3 quantum convolution kernel.
- Red Line: QNN with a 4x4 quantum convolution kernel.
- Green Line: CNN without a quantum layer.

Initial Accuracy:

All models begin with an accuracy of approximately 0.64 at epoch 0, suggesting similar initial conditions.
Learning Dynamics:

2x2 Kernel QNN (Yellow):
Achieves the highest accuracy, peaking around 75% and maintaining this level throughout the epochs.
3x3 Kernel QNN (Blue):
Slightly below the 2x2 kernel but still shows high efficacy, with a validation accuracy of around 73%.
4x4 Kernel QNN (Red):
Demonstrates lower accuracy compared to the smaller kernels at around 70%, suggesting a less effective capture of fine details.
Classical CNN (Green):
Consistently performs the worst, stabilizing at around 65% accuracy.

Final Accuracy:

- 2x2 Kernel QNN: Highest accuracy (75%).
- 3x3 Kernel QNN: Slightly lower, yet effective (73%).
- 4x4 Kernel QNN: Lower efficacy for capturing fine details (70%).
- Classical CNN: Lowest performance (65%).

<u>Technical Insights</u>

Effectiveness of Smaller Kernels:

Smaller kernels (2x2, 3x3) prove more effective in capturing intricate patterns, likely due to their ability to process high-frequency components in the images.

State Invariance:
Quantum circuits exhibit state invariancy, which is crucial in capturing invariant features that are beneficial for image recognition, suggesting an edge over classical convolutional approaches in certain aspects. It shows state invariancy through the use of symmetry operations and unitary gates that preserve key properties of the quantum state. This invariance means that important features remain unchanged even when transformations are applied to the data.

Hybrid Architecture:
The quantum-preprocessed images are flattened and linked to a classical neural network layer. This hybrid approach leverages the quantum feature extraction capability, significantly enhancing the classification performance.

Conclusion:

The results confirm that integrating quantum convolutional layers, especially with smaller kernel sizes, significantly enhances neural network performance in image recognition tasks. The 2x2 quantum kernels are the most effective, achieving the highest accuracy, while the 3x3 kernels also perform commendably but with slight concessions in accuracy. The 4x4 kernels, although useful, show diminished effectiveness, particularly in capturing fine details. Conversely, the classical CNN demonstrates the least capability, underscoring the advantages of quantum enhancements in neural architectures. This analysis underscores the potential of quantum convolutional approaches in refining feature extraction processes, thereby improving image classification outcomes.

Future Work:

In future work, we plan to address several areas to enhance and extend our experiments.
Firstly, our team attempted to replicate the results that we have discussed in our report so far on a real IBM quantum device. However, due to computational constraints, processing one 2x2 patch of an image took approximately one minute, and even after one hour, we were unable to preprocess a single image fully. Additionally, IBM's quantum computer usage is limited to 10 minutes per month, posing significant limitations (because of this we got an error which can be seen in the last cell of the juypter notebook uploaded on github). Despite these challenges, we successfully wrote the entire code intended for the IBM server, which can be utilized once faster quantum devices become available. (please refer to github link)

Secondly, we aim to test our quantum convolutional approach on colored images to evaluate its effectiveness across all three RGB channels, instead of just 8 bit grayscale images.

Third, given access to more powerful computational resources, we would like to run our experiments on the entire Fashion MNIST dataset to validate our results on a larger scale..

Fourth, we plan to test our insights on Quantum Neural Networks, as opposed to just Classical Neural Networks.

Fourth, we could also explore and test the effect of strides in quantum convolutional imaging. We are interested in examining how preprocessing an image with a patch size of 2x2, but with strides of 1, 2, or 3, will affect accuracy.

These steps will help us further understand the potential and limitations of quantum-enhanced neural networks

Reference:

(1) Henderson, M., Shakya, S., Pradhan, S., & Cook, T. (2019). Quanvolutional Neural Networks: Powering Image Recognition with Quantum Circuits. arXiv preprint arXiv:1904.04767.
(2) Xanadu. (n.d.). Quantum Convolutional Neural Network (QCNN). Retrieved from https://pennylane.ai/qml/glossary/qcnn/.
(3) Xanadu. (n.d.). Tutorial: Quanvolutional Neural Networks. Retrieved from https://pennylane.ai/qml/demos/tutorial_quanvolution/.