COMP90084 Research Project

Chenyang Dong[1] and Jiahui Luo[2]

[1]doncd@student.unimelb.edu.au | 1074314
[2]jialuo7@student.unimelb.edu.au | 1282009

Convolutional Neural Network (CNN) has been proven effective in many applications, especially for image classification and segmentation. With the rapid development of quantum machine learning, the quantum convolutional neural network (QCNN) has been proposed and applied to some real tasks, such as quantum phase recognition. In this project, we first give a review of QCNN and then apply the QCNN to classical image classification. We experimented with several different structures of QCNN on the Chinese-MNIST dataset and compared the results with CNN. We have shown that the QCNN models give a slightly lower accuracy but are much more stable than CNN.

1 Introduction

Convolutional neural network (CNN) is a widely used tool in many research domains such as computer vision, natural language processing, and signal analysis. However, in real industry, extreme computational power is required due to the large data size and high complexity of algorithms. Quantum algorithm may be helpful for solving this challenge due to its potential to solve problems faster than classical methods [2, 14].

Quantum machine learning has attracted substantial interest recently [4]. Many Research aims to use quantum machines to speed up or enhance the performance of classical algorithms with CNN being one of them. Due to the superposition and entanglement properties of quantum, QCNN could have higher storage capacity and computational efficiency compared to classical CNN [25].

QCNN was proposed to solve quantum problems initially, in which the input is quantum information. However, Due to its advantage in complexity, many researchers [15, 6, 16] have applied QCNN to classical problems by encoding the numerical data to quantum representation. Tak [15] et al. applied various QCNN models with different structures of circuits, data encoding methods, and pre-processing methods on MNIST and Fashion MNIST datasets. Their work demonstrates the superiority of QCNN compared to CNN. To investigate the effectiveness of QCNN on more classical data, we further experimented with various QCNNs based on Tak's work.

In this project, we applied variational quantum circuits based on QCNN to the Chinese-MNIST data set. Different encoding methods and circuit structures have been experimented with. In addition, we also build a CNN with a similar number of parameters for comparison. The results show that QCNN is much more stable than CNN while the accuracy is slightly lower than CNN.

This report is organized as follows: Section 1 gives an introduction and the motivation for this project. Section 2 introduces the CNN and QCNN in more detail, as well as some related work regarding QCNN. Section 3 shows the methods we used to build a QCNN for image classification. Section 4 illustrates the experiments. Section 5 shows the results and analysis. Finally, section 6 provides a summary of the results and suggests some future directions.

2 Literature Review

2.1 CNN

CNN has the ability to effectively capture spatial dependencies of the image and extract important features automatically [6]. A typical CNN consists of the convolutional layer, the pooling layer, and the fully connected layer. The key function of CNN is using the small rectangular matrix, known as the convolutional filter or kernel, to extract features from input by calculating the dot products. In contrast, the kernel is aligned to each input part. The pooling layer is used be-

tween successive convolutional layers to reduce the size of the feature map. Finally, the fully connected layer, which is a feed-forward neural network, is used to convert the feature to the expected number of outputs. There are various pooling layers with the max pooling being the most commonly used one. In addition, some activation functions are often inserted between layers to introduce non-linearity.

2.2 QCNN

There is no unified definition for QCNN. However, researchers usually use it to refer to the quantum analogue of a convolutional neural network that uses a structure similar to CNN, including convolutional layers, pooling layers, and fully connected layers. In order to apply the concept of classical convolution neural network to quantum, the convolution kernel is replaced by the parameterized quantum circuits. And within the circuit, the unitary operators will act on the qubits. The detailed implementation may vary depending on the design. In the training of CNN, the parameters are updated by the optimizers based on the cost function. While for the QCNN, we use variational circuits to obtain the current states and measure classical qubits as predictions to calculate the cost function and use a classical optimization algorithm to update parameters in each step.

QCNN is proposed by Cong [8] with a multi-scale entanglement renormalization Ansatz (MERA) based architecture to solve quantum problems, including quantum states classification and quantum error correction. They applied the proposed model to solve quantum problems, including quantum phase recognization and quantum error correction. In both tasks, QCNN outperforms the existing approaches.

MERA is a model introduced by Vidal [10, 23], which is designed to efficiently simulate many-body state quantum systems. The model is able to specify the layout of the hierarchical circuit and algorithms for learning the parameters. Unlike earlier classical techniques especially density matrix renormalization group (DMRG) used on 1D systems [24], MERA uses a tensor network instead, therefore, leading to address higher-dimension systems. The core idea of MERA is the entanglement renormalization group. In general, the model initially starts from a known deterministic state, then gradually injects new qubits and applies entangling unitary between existing qubits and new qubits injected. With such additional entanglers, short-range entanglement can be built between neighboring qubits and a broader range of quantum correlations can be captured. Further studies on entanglement renormalization such as algorithms and generalization have been explored in [9, 11, 13].

QCNN is inspired by MERA and runs in a reversed direction. By choosing an N-qubit operator in the layer for different dimension systems, it is a lot similar to increasing the kernel size in CNN. The importance of such a structure is that, for any given state $|\psi\rangle$ with a MERA representation, there is always a QCNN that recognizes the state $|\psi\rangle$ with deterministic measurement outcomes. Also, the model will reduce the size of the quantum system exponentially from the given data, which gives O(logN) parameters to classify N-qubit input, compared to a generic quantum circuit-based classifier [12, 20].

2.3 Related Work

Some related works that apply QCNN to classical data classification tasks are shown.

In [16], the authors introduced a quantum convolution product and quantum sampling technique to implement QCNN architecture that could accept exact same input and output as classical CNN. They also proposed the forward pass and backpropagation algorithms that offer a speedup compared to the corresponding CNN. Their proposed algorithms are tested using the MNIST dataset with manually simulated noise, and the results show that QCNN has the capability to learn despite the noise.

Li et al. [17] proposed a quantum deep convolutional neural network for image recognition and showed an exponential acceleration compared with classical CNN. To simulate the CNN, the proposed model has the quantum convolutional layer composed of the parametrized unitary and rotation gates and the quantum classified layer that contains unitary transformation and projection measurement. Their model was experimented on handwritten digital classification (MNIST) and German traffic sign recognition (GTSRB) tasks and gave competitive results.

Chen et al. [6] applied QCNN to the classification of high-energy physics events. Specifically,

they use QCNN to classify four different types of particles (muon, electron, π^+, and proton) from the images of their simulated activities. They first use amplitude encoding to convert the classical data to quantum states. The quantum states are then fed into the variational layer, which is a sequence of unitary transformations including the entanglement part and parameterized rotation part. The variational Layer is used as the kernel in the Convolutional layer that converts $n \times n$ dimensional vectors into a single value by measurement. Their results show that QCNN could achieve higher test accuracy compared to CNN in some situations.

3 Dataset

Since most of the previous studies investigate the performance of QCNN on MNIST dataset [16, 17], it is desired to see the model achievable on other tasks as well. However, due to the hardware limitation, the complexity of the task would be restricted. Thus, in this experiment, the dataset is curated from [19] which gives a collection of handwritten numbers in Chinese characters. Here we experiment proposed model with a binary classification problem by using only data with labels 0 and 1.

4 Methods

4.1 Data Preprocessing

Due to the environment limit, we only attempt to use at most 8 qubits. The dimensional reduction algorithms are needed to reduce input feature size. Therefore, principal component analysis (PCA) [1] and simple resizing are used.

PCA is one of the commonly used algorithms for dimensionality reduction. It is an orthogonal basis transformation that transforms data into a new coordinate system. The new bases are found by maximizing the variance of the scalar projection of data. It could reduce the dimension of data while preserving most various features. For the resizing method, it directly removes part of the pixels.

4.2 Encoding

To encode the classical data to quantum representation in a Hilbert space, we considered both Amplitude Encoding and Angular Encoding.

4.2.1 Amplitude Encoding

Amplitude encoding uses the measurement probability to express normalized classical data. It is a compact representation that can express N features by only log(N) qubits. For normalized input vectors $x = [x_0, ..., x_{N-1}]^T \in \mathbb{R}^N$ of dimension $N = 2^n$, we encode them using N-qubit quantum state $|\phi(x)\rangle$ as:

$$U_\phi(x) : x \in \mathbb{R}^N \to |\phi(x)\rangle = \frac{1}{\|x\|} \sum_{i=1}^{N} x_i |i\rangle$$

It is an effective method in terms of the number of qubits used. By using fewer qubits, the number of parameters is also reduced. However, the implementation of an amplitude encoding scheme may be complex due to the lack of an efficient way to preparing classical vectors into quantum registers [7], Quantum Random Access Memory (QRAM) may be needed [21].

4.2.2 Angular Encoding

The angular encoding uses rotation angles of qubits to represent them. Each data point is expressed by a single qubit. Here the amplitude of this single qubit is a data point. For example, we could encode x_i as $|\phi(x_i)\rangle = cos(\frac{x_i}{2}) |0\rangle + sin(\frac{x_i}{2}) |1\rangle$ So, for input vectors $x = (x_0, ..., x_{N-1}))^T$, we encode them using N qubit quantum state $|\phi(x)\rangle$ as:

$$U_\phi(x) : x \in \mathbb{R}^N \to |\phi(x)\rangle = \bigotimes_{i=1}^{N} (cos(\frac{x_i}{2}) |0\rangle + sin(\frac{x_i}{2}) |1\rangle)$$

Compared to Amplitude Encoding, more qubits are required. In the contract, the computation for encoding is simpler. One of the simplest ways is just applying the rotation gate to each qubit using data that normalized to $[0, \pi]$ as rotation angles. and we get the desired state with information encoded. For instance, to encode 3 data points $\mathbf{x} = [1, 2, 3]$, we would use a circuit shown in Fig. 1.

4.2.3 Angular Compact Encoding

To save the qubits used for encoding, we also considered encoding two classical data points into

one qubit by applying the different rotation matrix to the input states twice. For instance, two data points [1, 2] could be encoded to 1 qubit by applying the circuit shown in Fig. 2.

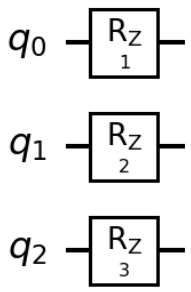

Figure 1: Circuit for encoding [1,2,3] with angular encoding

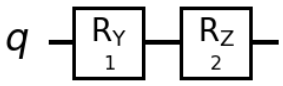

Figure 2: Circuit for encoding [1,2] with angular compact encoding

4.3 Unitary Ansatz

The ansatz used for the convolutional layer is shown in Fig. 3 which comes from [18, 22]. $U3(\theta, \phi, \lambda)$ in the circuit can be expressed as $R_z(\phi)R_x(\frac{-\pi}{2})R_z(\theta)R_x(\frac{-\pi}{2})R_z(\lambda)$. In total, there are 15 variational parameters composing this arbitrary SU(4) gate. The design of such circuit is inspired by the tight bound on the one-qubit gates and CNOT gates count to implement the generic two-qubit computation [5, 22]. With such design, it helps improve the efficiency of the computation to a great extent.

For the pooling layer, we chose the ansatz of a parametrized two-qubit circuit shown in Fig. 4, which applies two controlled rotation gates. By tracing out after this operation, the dimension will be decreased as the entanglement is reduced from two qubits to just one. However, since [22] proves the existence of tight bounds on the number of gates required for an optimally constructed circuit, two additional parameters used in the pooling layer might be redundant for the system. Thus, the impact of using extra parameters in the pooling layer after our convolutional layer would be experimented with and analyzed in the result.

4.4 QCNN Circuit

Our QCNN consists of convolutional layers and pooling layers. A convolutional layer consists of parameterized entangling unitary between neighboring qubits. While the pooling layer traces a single qubit out of a two qubits system, the number of qubits entangled in the following layers is halved. Since we have 8 qubits, two convolutional layers and two pooling layers are used, which ensures that the information is stored in one single qubit in the end. Finally, the probability distribution of the fourth qubit to be the results are measured for the predictions.

Besides, according to the consideration on pooling layer in Section 4.3, the circuit using pooling layers without parameters was also tested to see whether how it actually impacts on the model learning.

4.5 CNN

To have CNN using similar number of parameters as QCNN, we use a simple 1D CNN with 2 convolutional layers and 2 max-pooling layers. ReLU activation function is used between layers to introduce non-linearity.

4.6 Parameters

To fairly compare the performance of QCNN and CNN, we build them to have a similar number of parameters (See Tab. 1). These variational parameters are trained to minimize the cost function for optimisation.

Model	# of parameters
CNN - PCA8	44
CNN - PCA16	56
QCNN without Pooling	45
QCNN	51

Table 1: Number of Parameters for CNN and QCNN-PCA16

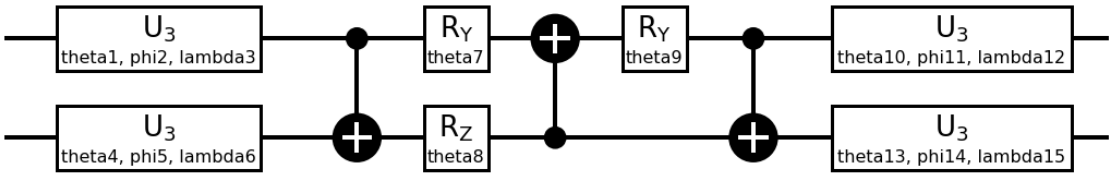

Figure 3: Unitary Ansatz for Convolution Layer

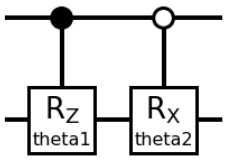

Figure 4: Unitary Ansatz for Pooling Layer

5 Experiments

Our experiments are conducted using the state simulator of qubit-based quantum circuit architectures provided by Pennylane [3]. In order to have more reliable statistics and show the stability of the model performance, four repetitions of the experiment are conducted to compute the mean and standard deviation.

5.1 Data Embedding

For PCA embedding, we use the scikit-learn implementation in the decomposition package. Both 8 and 16 dimensions features have been experimented with. While for resizing, tf.image.resize from TensorFlow is used. Since we use at most 8 qubits and the amplitude encoding could use n qubits to encode n^2 features, we choose to resize the original 28×28 image to 256 dimensions.

5.2 Encoding

For both amplitude encoding and angular encoding, the implementation from Pennylane.templates.embeddings is used. For angular encoding, we apply it to 8-dimension inputs and 16-dimension inputs. For inputs of 8 features, we apply Y rotation to the initial states by the angle of input data. While for 16 feature inputs, we apply X rotation using the first 8 data points and then apply Y rotation using the following 8 data points. For amplitude encoding, due to its ability to represent n^2 using n qubits, we apply it to 8-dimension inputs, 16-dimension inputs, and 256-dimension inputs using 3, 4, and 8 qubits separately.

5.3 Cost function

In the experiment, the cross-entropy loss is used to test the performance of our models. Cross entropy loss is mainly used to determine how close the actual output is to the desired output. That is, the smaller the value of cross-entropy, the closer the two probability distributions are. The weights of the model would be adjusted to minimize this value during training, which leads to better performance. For optimizer, Nesterov Momentum is used on QCNN model while Adam is used on CNN, because of higher efficiency on corresponding model after a series of trial.

6 Results and Analysis

The classification accuracy of models with different data pre-processing and encoding methods is shown in Tab. 2. From the results, CNN performs the best in any data pre-processing method with over 96% classification accuracy. The model of QCNN using resizing method to preprocess the data gives the worst performance. Besides, no huge gap is found between the other models.

Two sets of designs are used to test the performance of the circuit structure without pooling layers. One uses amplitude encoding with PCA-8 as the pre-preprocessing method, while the other is using angular-compact encoding with PCA-16. From the result, the accuracy for the former design decreases to a great extent by 3.6% and becomes more fluctuated after removing the pooling layers. However, with the latter design, it performs better by 0.3% and becomes more stable. At this stage, the function of the pooling layer is

	Data Pre-processing	Encoding	Classification Accuracy
QCNN	RESIZE-256	Amplitude	90.1 ±4.0
	PCA8	Amplitude	95.9 ±0.6 92.3 ±3.8*
	PCA8	Angular	95.3 ±1.0
	PCA16	Angular-compact	94.8 ±1.4 95.1 ±0.6*
CNN	PCA8		96.5 ±1.5
	PCA16		96.1 ±1.2

* Circuit without pooling layer

Table 2: The Classification Accuracy of Models with Different Methods

doubted. One possible reason may be that the complexity of our task is relatively low and we already have enough parameters in the convolutional layers. So the extra parameterized pooling layer makes the model harder to learn.

In Fig. 5, QCNN with different data preprocessing and encoding methods are compared by the average cross entropy loss trend. As demonstrated, PCA-8 with angular encoding is able to outperform PCA-8 with amplitude encoding from the perspective of training loss. By using the Resizing method to preprocess the data, amplitude encoding represents 256 features to 8 qubits. When using PCA-8 encoding, only 3 qubits are used. The lower loss obtained by resizing 256 may indicate that though amplitude could encode the N feature in $log(N)$, some information may be lost.

Among four sets of methods, PCA16 with angular compact encoding converges fastest and reaches the lowest loss at the end. This shows the compact angular encoding, which encodes N features by N/2 qubits, could effectively represent the data.

To further compare the performance between CNN and QCNN with different structures, we compare the cross entropy loss trend between them. From Tab. 1, we have seen the models we built using similar number of parameters for fair competition. Since the trend of CNN with PCA-16 does not have a significant difference from using PCA-8, only CNN using PCA-8 (44 parameters) is used to compare with QCNN models. Two QCNN models both use PCA-16 and angular-compact encoding methods, but either keep pooling layers or not (51 or 45 parameters). Fig. 6

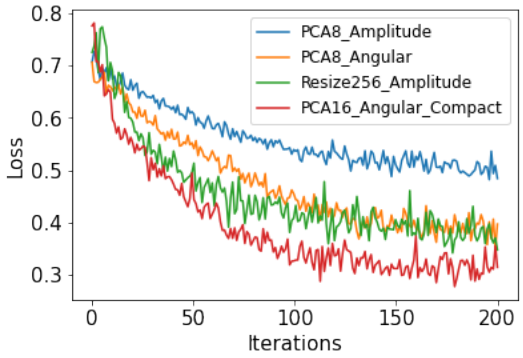

Figure 5: Cross Entropy Loss of QCNN with Different Encoding Methods

represents the performance of the models with cross-entropy loss during the training using the mean and standard deviation from repeated experiment. By comparing the trends, one can see that under such circumstances QCNN are always trained faster at the early stage. Besides, the training on QCNN is always more stable than CNN since they have a significantly smaller standard deviation.

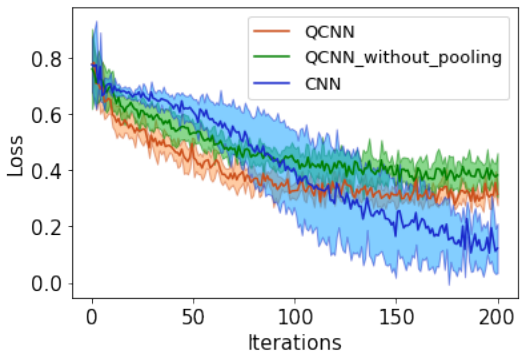

Figure 6: Cross Entropy Loss of QCNN with PCA-16 Dense Encoding and CNN with PCA-8

7 Conclusion and Future Directions

In conclusion, this project proposed a QCNN model for binary classification on images. Three encoding methods, amplitude encoding, angular encoding, and angular-compact encoding have been experimented with. Also, the effectiveness of parameterized pooling layer after a well-built convolutional layer is also examined. Experiments showed that the compact-angular encoding outperforms the others and the redundant parameters in the pooling layer makes few differences in the accuracy or even worse. In addition, we also compared the results of our model with classical CNN and showed that the QCNN performs slightly worse than CNN but is much more stable and has lower time complexity.

Due to the computation limit, we only applied QCNN and CNN to the simple task using only 8 qubits. To examine the real potential of QCNN, a harder task with a more complex circuit and more qubits may be needed. In addition, some proposed QCNNs use a Quantum Error Correction mechanism that measures some qubits in the pooling layer, which may help improve the performance and could be conducted as future work.

References

[1] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.

[2] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 2019.

[3] Ville Bergholm, Josh Izaac, Maria Schuld, Christian Gogolin, M Sohaib Alam, Shahnawaz Ahmed, Juan Miguel Arrazola, Carsten Blank, Alain Delgado, Soran Jahangiri, et al. Pennylane: Automatic differentiation of hybrid quantum-classical computations. *arXiv preprint arXiv:1811.04968*, 2018.

[4] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549(7671):195–202, 2017.

[5] Stephen S. Bullock and Igor L. Markov. Arbitrary two-qubit computation in 23 elementary gates. *Phys. Rev. A*, 68:012318, Jul 2003.

[6] Samuel Yen-Chi Chen, Tzu-Chieh Wei, Chao Zhang, Haiwang Yu, and Shinjae Yoo. Quantum convolutional neural networks for high energy physics data analysis. *Physical Review Research*, 4(1):013231, 2022.

[7] Samuel Yen-Chi Chen, Chao-Han Huck Yang, Jun Qi, Pin-Yu Chen, Xiaoli Ma, and Hsi-Sheng Goan. Variational quantum circuits for deep reinforcement learning. *IEEE Access*, 8:141007–141024, 2020.

[8] Iris Cong, Soonwon Choi, and Mikhail D Lukin. Quantum convolutional neural networks. *Nature Physics*, 15(12):1273–1278, 2019.

[9] G. Evenbly and G. Vidal. Algorithms for entanglement renormalization. *Phys. Rev. B*, 79:144108, Apr 2009.

[10] G. Evenbly and G. Vidal. Entanglement renormalization in two spatial dimensions. *Phys. Rev. Lett.*, 102:180406, May 2009.

[11] G. Evenbly and G. Vidal. Scaling of entanglement entropy in the (branching) multiscale entanglement renormalization ansatz. *Physical Review B*, 89(23), jun 2014.

[12] Edward Farhi and Hartmut Neven. Classification with quantum neural networks on near term processors. *arXiv: Quantum Physics*, 2018.

[13] Jutho Haegeman, Tobias J. Osborne, Henri Verschelde, and Frank Verstraete. Entanglement renormalization for quantum fields in real space. *Physical Review Letters*, 110(10), mar 2013.

[14] Aram W Harrow and Ashley Montanaro. Quantum computational supremacy. *Nature*, 549(7671):203–209, 2017.

[15] Tak Hur, Leeseok Kim, and Daniel K Park. Quantum convolutional neural network for classical data classification. *Quantum Machine Intelligence*, 4(1):1–18, 2022.

[16] Iordanis Kerenidis, Jonas Landman, and Anupam Prakash. Quantum algorithms for

deep convolutional neural networks. *arXiv preprint arXiv:1911.01117*, 2019.

[17] YaoChong Li, Ri-Gui Zhou, RuQing Xu, Jia Luo, and WenWen Hu. A quantum deep convolutional neural network for image recognition. *Quantum Science and Technology*, 5(4):044003, 2020.

[18] Ian MacCormack, Conor Delaney, Alexey Galda, Nidhi Aggarwal, and Prineha Narang. Branching quantum convolutional neural networks. *Physical Review Research*, 2020.

[19] K Nazarpour and M Chen. Handwritten Chinese Numbers. 1 2017.

[20] Maria Schuld, Alex Bocharov, Krysta Marie Svore, and Nathan Wiebe. Circuit-centric quantum classifiers. *Physical Review A*, 2018.

[21] Maria Schuld, Mark Fingerhuth, and Francesco Petruccione. Implementing a distance-based classifier with a quantum interference circuit. *EPL (Europhysics Letters)*, 119(6):60002, 2017.

[22] Farrokh Vatan and Colin Williams. Optimal quantum circuits for general two-qubit gates. *Physical Review. A*, 69(3), 3 2004.

[23] G. Vidal. Class of quantum many-body states that can be efficiently simulated. *Phys. Rev. Lett.*, 101:110501, Sep 2008.

[24] Steven R. White. Density matrix formulation for quantum renormalization groups. *Phys. Rev. Lett.*, 69:2863–2866, Nov 1992.

[25] Renxin Zhao and Shi Wang. A review of quantum neural networks: methods, models, dilemma. *arXiv preprint arXiv:2109.01840*, 2021.

Contribution

Chenyang Dong:

- Background research on the topic;
- Architecture and conclusion part of the presentation;
- Dataset exploration on Mnist, FashionMnist, MedMnist;
- Experiment including model modification, result visualization;
- Report on literature review, circuit visualisation, experiment and result analysis part.

Jiahui Luo:

- Background research on the topic;
- Application part of the presentation;
- Experiments on MedMnist, Pima Indians Diabetes Database, and Bank Note Authentication dataset;
- Modify the model and take Experiments;
- Abstract, Introduction, Literature review, Methods, Experiments on Report;