Hybrid Quantum Neural Networks for Image Classification

Olivia Watt
College of Engineering
College of Science
Virginia Tech
Blacksburg, VA
oliviawatt@vt.edu

Lucas Shadoyan
College of Engineering
Virginia Tech
Blacksburg, VA
lshad03@vt.edu

Thomas Nguyen
College of Engineering
Virginia Tech
Blacksburg, VA
thomasn03@vt.edu

Fig. 1. Snippet of the VMMRdb Dataset

I. Introduction

As we reach limits with classical compute power, it becomes necessary to investigate methods that execute faster and are capable of solving more difficult problems. By applying concepts from quantum mechanics to the computing space, we can leverage the power of superposition, entanglement, and interference to parallelize operations and operate at higher dimensions.

In this project, we compare classical machine learning with quantum machine learning through image-based car model classification algorithms. Specifically, our models predict vehicle *make*, *model* and *year*. This problem being chosen because it's already sufficiently difficult for a classic approach using convolutional neural networks due to the minute level of detail a model would need to extract to differentiate between model years. Our goal with this project is to determine whether quantum-enhanced models can achieve performance comparable to or superior to classical approaches in the context of image classification.

To more deeply observe these differences and learn more about different types of quantum circuits/processing, we decided to explore two different approaches. The first approach comparing the classical approach to one that uses a quanvolutional layer as a preprocessing step on a small subset of the Stanford Cars dataset. The other approach comparing the classical approach to one that adds a quantum circuit head on top of a CNN backbone, but this time on a larger dataset combining both the Stanford Cars and VMMRdb dataset.

II. Related Work

A main source of reference for this project was research done by Henderson et. al on quanvolutional neural networks for image recognition [1]. This was a study directly related to our project that trained on the MNIST dataset. We referenced this study to set up a simulation to get a baseline estimation of performance improvement using a hybrid quantum convolutional neural network over a purely classical one, and the results of their training data aligned with our results.

In recent years, hybrid quantum-classical architecture has become more popular as a way to reduce computational and parameter overhead while maintaining competitive classification accuracy. Senokosov et al. introduced the HQNN-Parallel architecture, which integrates classical convolutional layers for feature extraction with a classification layer consisting of multiple parameterized quantum circuits. The design utilizes angle embedding to map classical features onto a high-dimensional quantum Hilbert space where variational gates perform the classification tasks [2].

Another source of reference for this project was the Deep Residual Learning for Image Recognition paper by Kaiming He et. al. [3] This is the paper that introduced the ResNet architecture and popularized the use of residual connections in deep learning. In terms of our project, this paper was used to get a better grasp of the ResNet architecture, the layout, how residual connections work and how a bottleneck block works as well. This was especially helpful since we used ResNet as the backbone of our larger example/exploration.

III. Quantum Information Science Background

A. Bits vs Qubits

In classical computing, a bit is denoted as a 0 or a 1. It is either one or the other, and we know this with 100% certainty. In other words, values of these bits do not inherently depend on

the values of other bits. In contrast, a qubit, otherwise known as a quantum bit, can hold values in a superposition of states.

The most general form of a qubit is $\alpha|0\rangle + \beta|1\rangle$, where α and β are amplitude factors that represent the probability of the qubit being in a particular state. From the Born rule, and because we know these are probabilities, the normalized state implies that $|\alpha|^2+|\beta|^2 = 1$. If we perform many measurements on qubits in the same state, it is evident that we will measure $|0\rangle$ for $|\alpha|^2$ percent of the time and $|1\rangle$ for $|\beta|^2$ percent of the time. If $|\alpha|^2 = 1$, then we will always measure our qubit as being in the $|0\rangle$ state, as the $|1\rangle$ component would become 0, and vice versa if $|\beta|^2 = 1$. We say that $|0\rangle$ and $|1\rangle$ form a basis, namely the computational (or Z) basis. Since we are in a Hilbert space, we can take the inner product of our normalized basis states such that $\langle 0|0\rangle = \langle 1|1\rangle = 1$ and $\langle 0|1\rangle = \langle 1|0\rangle = 0$. We can apply gates to qubits to transform their states, changing their amplitudes and thus changing their probabilities.

B. Superposition

One of the most fundamental gates in quantum computing is the Hadamard gate, which puts a state into a superposition. The gate itself is represented by the following unitary matrix:

$$H = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

It is important to note that all quantum gates must be unitary to preserve reversibility, as original states would otherwise be unrecoverable after applying gates.

The Hermitian matrix, when applied to $|0\rangle$, can be represented in Dirac notation as $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle$, and applying it to $|1\rangle$ is represented in Dirac notation as $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle$. This math is shown below.

$$\begin{aligned} H|0\rangle &= \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ &= \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\ &= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \end{aligned}$$

$$\begin{aligned} H|1\rangle &= \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\ &= \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix} \\ &= \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \end{aligned}$$

From these results, we can see that applying a Hadamard gate changes the aplitudes of the $|0\rangle$ and $|1\rangle$ components, such that the probability of measuring either state become $\frac{1}{2}$. This means that before measurement, the qubit is simultaneously $|0\rangle$ and $|1\rangle$. It is only when we measure that we get a definite answer as to what state the qubit is in – and measuring will result in $|0\rangle$ half the time, but $|1\rangle$ the other half of the time. Upon measuring, the qubit cannot return to its original state of being both $|0\rangle$ and $|1\rangle$ at the same time. This is called collapsing the state via measurement.

C. Entanglement

Another fundamental gate in quantum computing is the Controlled-NOT (CNOT) gate. In a two-qubit system, if we apply a Hadamard gate to one qubit and then link it with another through a CNOT gate, this will create entanglement and generate one of the four bell states. Bell states are maximally entangled states, meaning that you cannot measure any one qubit without destroying the superposition of the state.

The four bell states are

$$\begin{aligned} \Phi^+ &= \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \\ \Phi^- &= \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle) \\ \psi^+ &= \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle) \\ \psi^- &= \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle) \end{aligned}$$

The CNOT gate with the first bit being called the "control qubit" is represented by the following matrix.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

If we apply this gate to a 2-qubit system, the second bit will get flipped if the control qubit is 1. For example,

$$\begin{aligned} &\text{CNOT}|00\rangle \rightarrow |00\rangle \\ &\text{CNOT}|01\rangle \rightarrow |01\rangle \\ &\text{CNOT}|10\rangle \rightarrow |11\rangle \\ &\text{CNOT}|11\rangle \rightarrow |10\rangle \end{aligned}$$

From here, it should become clear how we create entanglement by applying the CNOT gate to a system after it passes through a Hadamard.

D. Quantum Convolutional Neural Networks

In the case of image classification, classical neural networks consist of several kernels that go through the image and detect features with a 2D array of weights. We alternate between these types of layers and pooling layers, which have the purpose of down-sampling. A quantum convolutional network is built on the same idea, but it leverages the concept of superposition. From the earlier section on superposition, it was demonstrated that qubits have the ability to travel down multiple paths simultaneously, with different probabilities.

This enables parallelism in quantum convolution, allowing for the extraction of feature data at a much higher speed than classically. Then there is quantum pooling, which uses parameterized quantum circuits to perform unitary transformations on the data, for the same purpose of down-sampling. It is clear that quantum convolutional neural networks have the ability to far surpass the speed of their classical counterparts, but the issue lies with scalability and error correction. This will become evident throughout the rest of this report.

IV. Methodology

A. Dataset

We are using both the Vehicle Make and Model Recognition Dataset (VMMRdb) consisting of 9,170 classes and 291,752 images and the Stanford Cars Dataset, which has $16,185$ images of 196 classes of cars, split into $8,144$ training images and $8,041$ testing images. In both datasets, classes contain *make*, *model* and *year*, though a lot have body style, trim, among other things that we needed to remove to combine the datasets. All images have different lighting conditions, angles, photo quality, and dimensions. In terms of image type, the VMMRdb dataset stores images primarily as JPEGs, with a few thousand being stored as PNGs. The images in the Stanford Cars dataset are all stored as JPEGs.

The VRMMdb dataset was found on GitHub and later on Kaggle. [4]. The Stanford Cars dataset was also found and downloaded from Kaggle as well, and more information about the dataset is available in PyTorch documentation [5]. Both datasets were downloaded using the kagglehub library.

The exploration on the smaller dataset was done solely using a subset of the Stanford Cars dataset, while the larger example/exploration was done using both the VMMRdb dataset and the Stanford Cars dataset.

To combine both of the datasets for this larger example, we first needed to normalize all of the different class names in both datasets. To normalize class names, we removed any identifiers such as body style, vehicle type, vehicle trim, and even powertrain type. These identifiers didn't pertain to make, model, or year and prevented class overlap between the two datasets, so they were removed.

In addition to normalizing the classes from both datasets, we also removed any classes that had too few images. A lot of classes only had 1 - 20 images, and we eventually went with removing any classes with less than 40 images as that still preserved most of the classes from the Stanford Cars dataset (The dataset that seemed to contain more difficult angles and lighting conditions). This helped the class imbalance, where the classes contained one or hundreds of images. However, it didn't fix it completely, to further fix this imbalance, we limited classes to contain at most 100 images from each dataset. This left us with 2023 classes to work with, a much more reasonable number of classes to predict both our ResNet and HQNN models.

We processed and augmented our dataset by first resizing all images to 224×224. This is a standard input size and was helpful when we later attempted to improve the accuracy of our HQNN by fine-tuning a pre-trained ResNet-50 model. In terms of randomization, we added horizontal flips, color jitter (brightness, contrast, saturation, and hue), and rotation to our images. All of this was in an effort to improve the robustness of our training data and help prevent our model from overfitting. [6] In addition, we normalized our input images using the mean and standard deviation of all 3 input channels of the train images (RGB in this case). This normalization helped speed up convergence and stabilize training. [7]

Fig. 2. Example of the Augmentation without Normalization

In running our small experiment of comparing a classical convolutional neural network to a hybrid quantum neural network, via something called "quanvolution," we chose to only use the Stanford Cars dataset, as training a hybrid quantum model requires much more processing power, i.e. a very large dataset would take overly long to train. Since we only needed a simple overall evaluation of the two approaches, we found that using a subset of the smaller dataset provided sufficient information to draw performance conclusions. Similar data augmentation was performed on the Stanford Cars subset.

B. Machine Learning Model

We are using two models for our larger example, ResNet-50 (A traditional CNN model) and a Hybrid Quantum Convolutional Neural Network.

ResNet-50 is a shallower ResNet model that only uses 50 layers. It's just the right balance between speed and robustness. It's more accurate than a ResNet-18 and while it is slightly less accurate than a ResNet-152 model, it's a lot more efficient during training and inference.

At a very basic level, ResNet models group convolutional layers into blocks. ResNet-50 calls these blocks bottleneck blocks. They're called bottleneck blocks because they intentionally bottleneck/compress information by reducing the

channel dimensionality passing into the block before doing the bulk of the computation. This reduces the computational requirement and enables for a deeper network. To do this, the first convolutional layer in these blocks uses 1×1 filters and decreases the depth / channel dimensionality of the input. The second convolutional layer uses 3×3 filters and does most of the heavy computation on this decreased input. The third convolutional layer increases the channel dimension back to its original input size or even increases it further using $1x1$ filters. Each of these blocks also use a residual connection or skip connection at the very end (between blocks). These residual connections allow information to bypass certain layers and be directly propagated to deeper layers. These connections create paths for the gradient to flow through, which directly helps to mitigate the vanishing gradient problem.

ResNet-50 uses 16 of these blocks, each with 3 convolutional layers as stated above. There is an additional convolutional layer at the start of the network that immediately downsamples the image input. On top of that, there is a fully connected layer at the end of the network that maps the convolutional layer output to the number of output classes. With that we have the 50 layers used in the ResNet-50 model - 48 used in the blocks, 1 at the beginning and 1 at the end. At a high level, the biggest and most important difference between the ResNet architecture and something like the VGG architecture, which also organizes layers into blocks, is the addition of residual connections or skip connections. [8]

The Hybrid QNN model uses the ResNet-50 architecture as a backbone to extract features from input images. The images pass through the convolution stages, global average pooling, and flattening, producing a 2048-dimensional feature vector which is then fed to a small classical MLP adapter which is scaled to form rotation angles. The angles are then encoded and processed through a variational quantum circuit using PennyLane's simulator (AngleEmbedding, StronglyEntanglingLayers). The circuit returns one classical value per qubit, which are then mapped to class logits by a single fully connected layer. Training uses cross-entropy loss on the logits with AdamW used as an optimizer on the final linear layer. For the fine-tuned HQNN, layer4 of the CNN is trained together with the hybrid head on raw images with a lower learning rate.

Residual connections: A neural network that doesn't use residual connections would have layers that learn this mapping:

$$H(x) = F(x)$$

Where x is the input and $F(x)$ is what the model learns to try to directly fit the desired mapping $H(x)$. Residual connections learn this mapping instead:

$$H(x) = F(x) + x$$

Instead of trying to directly fit the desired mapping $H(x)$, the model learns $F(x) = H(x) - x$, the difference between the input x and the desired mapping $H(x)$. This also explains why these connections mitigate the vanishing gradient problem because the residual connection introduces a direct path for the gradient to flow through. During backpropagation the gradient becomes the following:

$$\frac{dF}{dx} + 1$$

This means that even if $\frac{dF}{dx}$ becomes very, very small the gradient is still able to flow.

Convolution: Convolution is an element wise matrix multiplication operation. One of the matrices is the image and the other is a 3D filter or 2D kernel. When training a CNN, the weights and bias associated with each filter are modified. This modification allows filters to learn to extract different aspects/features from the input images.

Variational Quantum Circuit(VQC): A VQC is a version of a quantum circuit that utilizes two gates to manipulate qubits; a fixed and parameterized gate. The fixed gate takes classical inputs and applies rotation gates to create an input-dependent quantum state. The parameterized gate represents the tunable portion of the circuit. This applies parameterized single-qubit rotations and entangling gates across qubits, where weights are updated by gradient descent. Overall, these circuits have trainable parameters that are optimized using classical techniques using PennyLane. [9]

V. Results & Evaluation

A. VMMRdb and Stanford Cars Dataset Results

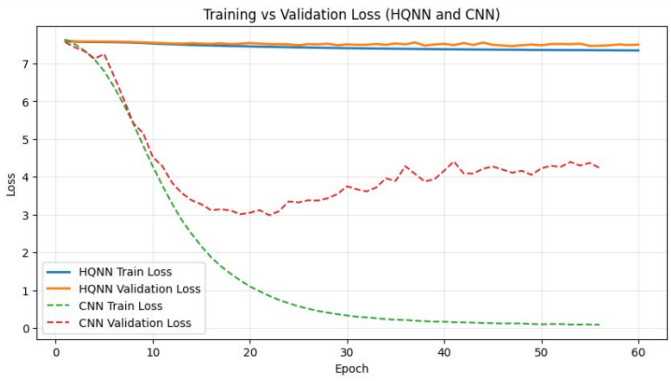

Fig. 3. Training & Validation Loss between HQNN and CNN on the Combined Dataset

Using PyTorch to implement and train a ResNet-50 model gave us a pretty solid baseline and we were able to see a top-1 accuracy of 35.78% and a top-5 accuracy of 65.50% on the testing set. To achieve higher accuracy, a deeper ResNet model like ResNet-152 is likely be needed. On top of a model modification, additional vehicle data and an even more robust dataset would likely be needed too. Additionally, as seen in figure 3, during training after about epoch 20, the validation loss begins to rise, which indicates that the model actually began to overfit the training dataset at that point, even if the validation accuracy was still rising slightly. The validation accuracy, though still rising slightly as seen in figure 4, also indicates that the model was overfitting after epoch 20 too,

as the curve becomes much more jagged as more and more peaks and troughs are formed. It also never quite hit 40% top-1 accuracy.

Performance of the baseline HQNN and fine-tuned HQNN on the pretrained ResNet50 with the merged dataset displays a reduction of training and validation loss over 20 epochs, seen in figure 3. The baseline HQNN reaches a final training loss of 6.1064, a validation loss of 6.4021, and a best validation accuracy of 0.0061 (0.61%). The fine-tuned version with only layer four unfrozen achieved improved validation behavior, ending with a final training loss of 6.2643, a validation loss of 6.3259, and a best validation accuracy of 0.0102 (1.02%), an improvement of 0.41% accuracy over the baseline. Although not impressive, it still performs better than random guessing, which would give you an accuracy of 0.049% given 2023 classes. This indicates that fine-tuning improved generalization, yet absolute classification performance still remained low. Low top-1 and top-5 accuracy values suggest the combined dataset creates a challenging multi-class classification problem. The label space is large and fine-grained, requiring the model to differentiate between specific combinations of vehicle make, model and year. Since many classes are visually similar, accurately predicting the exact class is difficult.

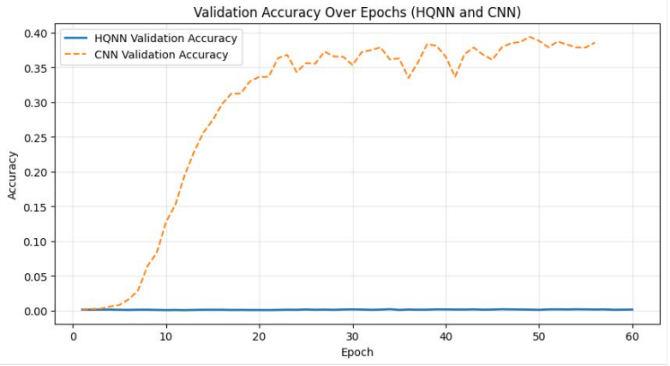

Fig. 4. Validation Accuracy between HQNN and CNN

B. Stanford Cars Subset Results

Since the Stanford Cars dataset was smaller, we found it to be a good choice for initial training with the hybrid quantum neural network. Using a quanvolutional template from PennyLane, we were able to integrate a small subset of the Stanford Cars dataset into this, modeling the work done by Henderson et. al. After running several tests, adjusting the amount of augmentation, number of classes, neural network layers, and image resolution, it became clear that the quanvolutional layer showed the most improvements when

1) Images were normalized
2) Fewer classes were used (we chose a minimum of 3)
3) A ReLU (Rectified Linear Unit) layer was added between the original two linear layers
4) Image resolution was increased

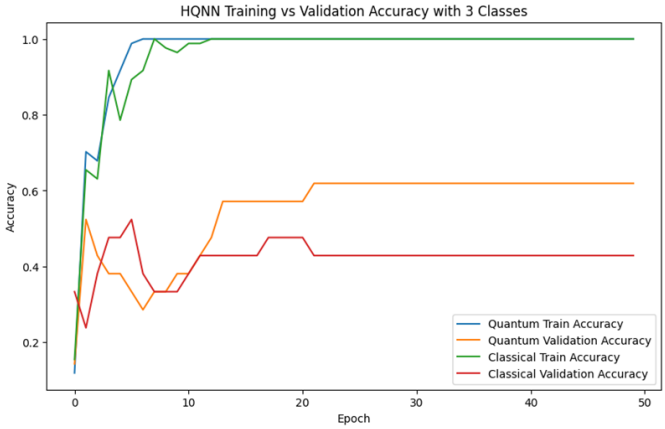

Fig. 5. HQNN Training vs. Validation Accuracy with 3 Classes

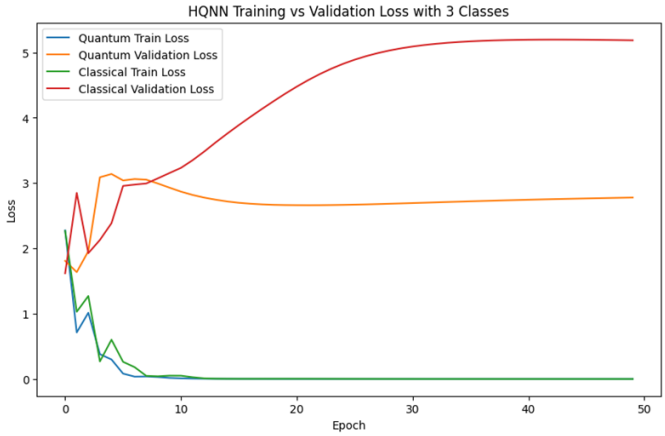

Fig. 6. HQNN Training vs. Validation Loss with 3 Classes

As shown in figures 5 and 6, with these parameters, the classical model exhibits significantly more loss and a notable drop in validation accuracy.

VI. Conclusion & Future Work

In terms of the HQNN in our larger example, the projection from the 2048 dimensional feature space that ResNet outputs to the six qubits used in our quantum circuit head is the fundamental bottleneck in this model. This is likely the reason for its low top-1 and top-5 validation across epochs. Using more qubits would most likely improve our situation slightly, but not by much given that the more qubits we add the more computationally intensive the training and inference would become (especially since we're simulating quantum circuits using PennyLane).

Only using a ResNet-50 model with a linear layer as the head performed much better than the HQNN for the large example, but not as much as we had expected, given the top-1 test accuracy score was only 35.78%. This is once again likely due to the fine-grained nature of predicting make, model and year with a large class amount.

With the HQNN small-scale model, it was evident that adding the quanvolutional layer had noticeable improvements over a solely classical implementation. However, this was only tested for a very specific case – the case of having an insanely small dataset. While we did test on a larger dataset and see that the hybrid model was actually hurting performance, there is still room to investigate this, as the HQNN used with the combined dataset was not based on the same implementation as the small-scale model, i.e. the two models had different architectures. We initially didn't think this would have an impact because we thought that just the *concept* of adding a quantum circuit to a neural network would improve performance, but after firsthand experience tweaking the model construction and layers, it is clear that overall performance depends on a variety of factors, architecture included.

In terms of future work, training a quanvolutional layer or layers with a ResNet-50 model on the larger dataset would be the first on our to-do list. This was something we ran out of time to do, and would be the most important thing to do to round off and fully complete this project. From there, training a quanvolutional layer with a ResNet-50 model (or an equivalent architecture) along with quantum circuit head would be another interesting avenue of research too. To add onto this, specifically trying to combat the projection bottleneck we hit with our current HQNN model on the larger dataset, a multi-headed architecture/multi-task learning with multiple quantum circuit heads might help. The potential use of a ResNet-152 model as a backbone instead of a ResNet-50 model might provide some interesting results.

It would also be worth maintaining consistency by ensuring that the model used to train the combined dataset follows the exact same architecture as the one used with the Stanford Cars dataset. In completing this project, we were thinking about how to work in parallel and didn't think that different implementations of adding a quanvolutional layer would impact performance very much; however, this was a massive oversight and major point of improvement for future work.

Lastly, as all of our training and simulations were done on classical hardware, it would be interesting to see how this performance would compare to using a quantum computer. There are several open-source frameworks available online for running simulations on real quantum hardware, so it could be worth seeing if this supports machine learning applications.

VII. Ethical Consideration

The main ethical dilemma that we have to take into account is privacy. A model that predicts, make, model, and year of a vehicle could also be used to potentially track vehicles if used in a fully developed pipeline. Our models of course aren't part of a vehicle recognition and tracking pipeline, so the real ethical consideration that we have to deal with surrounds license plates. Our models, especially the ResNet-50 base model, could be very easily integrated into a small website or app to identify vehicles. If pictures of vehicles were uploaded to such a website, the website would have to make sure that any pictures are completely secure. A fine-tuned YOLO (You Only Look Once) license plate detection model (or something similar) could be used to detect and blur license plates. An even better solution would be to never store any images that the users upload, maybe even blurring the license plate as the first step in the pipeline just to be safe.

VIII. Contribution Statement and Peer Evaluation

Olivia: For this project, I came up with the general idea of comparing something classical to something quantum, as it very much relates to the field of study I wish to pursue in the future. I have a decent amount of experience/background with quantum mechanics and quantum information science, so thought that this idea could make for a unique project. I figured it would align with Option B, of which " one or more existing machine learning techniques could be effective" and suggested that we do something simple, like image classification, for this comparison. Since the Option B track requires having us use a "new dataset," I first worked on training a classical convolutional neural network with the Stanford Cars dataset using a custom Resnet model, while Lucas trained with the VMMRdb dataset, with the idea that these two datasets would eventually be combined to give us our custom "new dataset." I did this to get a better idea of how ResNet worked. Then I tackled the idea of comparing a classical implementation to one with a quantum circuit layer and followed PennyLane documentation on quanvolutional neural networks, which was based off of our main related work reference by Henderson et. al. From here, I was able to come to the main conclusion that hybrid convolutional neural networks outperform classical neural networks for very small datasets and that in terms of general scalability, at least for this application, quantum neural networks do not scale well. This is because it takes much more processing power to run images through quantum circuits, so the waiting time just gets longer with larger datasets. As I mentioned in the conclusions section, it would be interesting to see if this wait-time decreases when using real quantum hardware, or how an actual model trained with qubits would compare to one trained with simulated qubits.

Lucas: While Olivia was working on training a classical convolutional neural network (a modified ResNet model) with the Stanford Cars dataset, I decided to work on getting a working custom PyTorch dataset class up and running for the VMMRdb data so we'd be able to use PyTorch data loaders with VMMRdb dataset during training. Along with creating the custom dataset class, I also worked on trying to better understand the ResNet model by coding up a working ResNet-18 model to predict make and model, and decided to not yet predict year after seeing its poor performance on the validation set, especially given it's even smaller than a ResNet-50 model. After milestone 1, with Olivia working on the smaller dataset exploration and Thomas working on getting the HQNN fully working, I decided to try and finally tackle combining both the VMMRdb dataset and the Stanford Cars dataset together to try and create a more robust dataset. With this even larger dataset to work with, we decided as a group to try and predict

make, model, and year again on the larger example, but this time with a ResNet-50 model instead.

Thomas: I built on top of Lucas' and Olivia's CNNs they had trained on the Stanford Cars dataset and the combined dataset by creating and training a HQNN hybrid head. I then worked on tweaking the HQNN to improve accuracy, first by using a pretrained ResNet-50 model and then by fine-tuning only the last layer.

References

[1] M. Henderson, S. Shakya, S. Pradhan, and T. Cook, "Quanvolutional neural networks: Powering image recognition with quantum circuits." [Online]. Available: http://arxiv.org/abs/1904.04767

[2] A. Senokosov, A. Sedykh, A. Sagingalieva, B. Kyriacou, and A. Melnikov, "Quantum machine learning for image classification," vol. 5, no. 1, p. 015040. [Online]. Available: http://arxiv.org/abs/2304.09224

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," Dec 2015. [Online]. Available: https://arxiv.org/abs/1512.03385

[4] F. Tafazzoli, K. Nishiyama, and H. Frigui, "A large and diverse dataset for improved vehicle make and model recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2017.

[5] StanfordCars — torchvision main documentation. [Online]. Available: https://docs.pytorch.org/vision/main/generated/torchvision.datasets.StanfordCars.html

[6] PyTorch, "Transforming and augmenting images — torchvision 0.13 documentation," 2017, accessed: 2026-03-16. [Online]. Available: https://docs.pytorch.org/vision/0.13/transforms.html

[7] P. Kashyap, "Image normalization in pytorch: From tensor conversion to scaling — medium," Dec 2024. [Online]. Available: https://medium-com.translate.goog/@piyushkashyap045/image-normalization-in-pytorch-from-tensor-conversion-to-scaling-3951b6337bc8?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc

[8] K. He, X. Zhang, S. Ren, and J. Sun. [Online]. Available: https://arxiv.org/pdf/1512.03385

[9] S. Shapiro, "Hybrid quantum-classical machine learning with pennylane: A comprehensive guide for computational research," 2025. [Online]. Available: https://arxiv.org/abs/2511.14786

[10] A. Li, W. Huang, X. Lan, J. Feng, Z. Li, and L. Wang, "Boosting few-shot learning with adaptive margin loss," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[11] J. Yu, Y. Dai, X. Liu, J. Huang, Y. Shen, K. Zhang, R. Zhou, E. Adhikarla, W. Ye, Y. Liu, Z. Kong, K. Zhang, Y. Yin, V. Namboodiri, B. D. Davison, J. H. Moore, and Y. Chen, "Unleashing the power of multi-task learning: A comprehensive survey spanning traditional, deep, and pretrained foundation model eras," 2024.

[12] A. Mao, M. Mohri, and Y. Zhong, "Cross-entropy loss functions: Theoretical analysis and applications," 2023. [Online]. Available: https://arxiv.org/abs/2304.07288

[13] V. Bergholm, J. Izaac, M. Schuld, C. Gogolin, S. Ahmed, V. Ajith, M. S. Alam, G. Alonso-Linaje, B. AkashNarayanan, A. Asadi, J. M. Arrazola, U. Azad, S. Banning, C. Blank, T. R. Bromley, B. A. Cordier, J. Ceroni, A. Delgado, O. D. Matteo, A. Dusko, T. Garg, D. Guala, A. Hayes, R. Hill, A. Ijaz, T. Isacsson, D. Ittah, S. Jahangiri, P. Jain, E. Jiang, A. Khandelwal, K. Kottmann, R. A. Lang, C. Lee, T. Loke, A. Lowe, K. McKiernan, J. J. Meyer, J. A. Montañez-Barrera, R. Moyard, Z. Niu, L. J. O'Riordan, S. Oud, A. Panigrahi, C.-Y. Park, D. Polatajko, N. Quesada, C. Roberts, N. Sá, I. Schoch, B. Shi, S. Shu, S. Sim, A. Singh, I. Strandberg, J. Soni, A. Száva, S. Thabet, R. A. Vargas-Hernández, T. Vincent, N. Vitucci, M. Weber, D. Wierichs, R. Wiersema, M. Willmann, V. Wong, S. Zhang, and N. Killoran, "Pennylane: Automatic differentiation of hybrid quantum-classical computations," 2022. [Online]. Available: https://arxiv.org/abs/1811.04968

[14] E. Aïmeur, G. Brassard, and S. Gambs, "Machine learning in a quantum world," in *Advances in Artificial Intelligence*, A. Y. Tawfik and S. D. Goodwin, Eds. Springer Berlin Heidelberg, vol. 3060, pp. 431–442, series Title: Lecture Notes in Computer Science. [Online]. Available: http://link.springer.com/10.1007/11766247_37

[15] S. B. Ramezani, A. Sommers, H. K. Manchukonda, S. Rahimi, and A. Amirlatifi, "Machine learning algorithms in quantum computing: A survey," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, pp. 1–8. [Online]. Available: https://ieeexplore.ieee.org/document/9207714/

[16] Y. Kwak, W. J. Yun, S. Jung, and J. Kim, "Quantum neural networks: Concepts, applications, and challenges," in *2021 Twelfth International Conference on Ubiquitous and Future Networks (ICUFN)*. IEEE, pp. 413–416. [Online]. Available: https://ieeexplore.ieee.org/document/9528698/

[17] F. V. Massoli, L. Vadicamo, G. Amato, and F. Falchi, "A leap among quantum computing and quantum neural networks: A survey," vol. 55, no. 5, pp. 1–37. [Online]. Available: https://dl.acm.org/doi/10.1145/3529756

[18] E. N. Evans, D. Byrne, and M. G. Cook, "A quick introduction to quantum machine learning for non-practitioners," version Number: 1. [Online]. Available: https://arxiv.org/abs/2402.14694

[19] M. Henderson, S. Shakya, S. Pradhan, and T. Cook, "Quanvolutional neural networks: powering image recognition with quantum circuits," vol. 2, no. 1, p. 2. [Online]. Available: http://link.springer.com/10.1007/s42484-020-00012-y