Final Project – Transfer Learning Tutorial

IDS-705 Principles of Machine Learning

Yuan Feng, Sebastián Soriano Pérez, Vishaal Venkatesh,
Abhiraj Vinnakota, Roderick Whang

Abstract – This tutorial explains the concepts and motivations behind transfer learning and seeks to provide an example of the technique used in practice. When tackling a machine learning problem, we often do not have enough training data in the domain we are interested in, but there is sufficient data available in a related domain. Transfer learning takes advantage of the available data to train a model for our target domain. Transfer learning has been shown to improve performance and reduce the costs of acquiring more labeled data. This tutorial will focus on applying transfer learning to deep neural networks for image classification problems, using a feature extraction (embedding) approach with classical computing and a classical-quantum computing hybrid approach. We will first explain the methods and formal concepts behind transfer learning, and then provide an example to measure and understand its performance.

1. Introduction

In the supervised learning context, deep neural networks have become more and more accurate in recent years. Convolutional neural networks using a residual learning network are particularly good at finding the optimal model parameters while improving accuracy [1]. However, it is not possible to apply these deep models into a different situation or phenomena other than what they were trained and designed for. It is important to have this flexibility in real-world scenarios where we rarely encounter data structured in the same way as our training data. Transfer learning allows us to transfer some of the information learned from an ideal situation into a different and related situation for which our data is more limited.

Quantum computing has been extensively featured in the news recently. While the field itself has made considerable progress since the 90s, we are still a long way from having quantum computers that can perform useful tasks. Although Google managed to controversially establish quantum supremacy with their 53-qubit Sycamore chip [8] by performing a task that no existing classical supercomputer could perform in reasonable time we are still a long way from solving useful problems in biology and chemistry using quantum computers. We are in the pre-transistor era of quantum computers – also known as the Noisy-Intermediate-Scale-Quantum (NISQ) era. However, with rapid advancements in quantum technology we are expected to have useful quantum computers in the next decade or so. To put it into perspective, in order to classically replicate the processing power of a 300-qubit quantum computer, we would require all the atoms in the known universe.

In this paper we will define a common transfer learning technique for computer vision and image classification problem: inductive transfer learning using feature extraction or embedding. By using a model previously trained on a robust training dataset, we can take advantage of its optimal parameters as a way to extract features for new data, reducing training time and improving performance considerably. First, we shall implement this method classically. Later, we will discuss one particular cutting-edge approach to this problem using quantum computing for a hybrid transfer learning method. In this approach, we replace a neural network's output layer with a dressed quantum circuit that we then train for our target domain and task [7].

Before we dive deeper into quantum circuits, we need to understand the basics of quantum computing as they are considerably different from that of classical computing. We would encourage the reader to visit Appendix A.1 where we have provided a brief introduction to the fundamentals of quantum computing.

2. Background

Pan, Sinno Jialin, and Qiang Yang have provided the following framework to understanding transfer learning [2]. In a machine learning, the **domain** $\mathcal{D} = \{\mathcal{X}, P(X)\}$ consists of the feature space $\mathcal{X}$ in the

dataset, with p features or predictors ($X = \{x_1, \dots, x_p\} \in \mathcal{X}$), and its marginal probability distribution $P(X)$. The marginal probability distribution $P(X)$ is the probability distribution of each of the variables $x_1, \dots, x_p$ with no reference to the values of the rest of the variables. A **task** $\mathcal{T} = \{\mathcal{Y}, f\}$ consists of the label space $\mathcal{Y}$ and an objective predictive function f, which can be used to predict the class label $f(x)$ and corresponds to the conditional probability distribution $P(Y|X)$ learned from the training data $\{X, Y\}$ (where $Y \in \mathcal{Y}$) [2].

We first train a machine learning model with data having a source domain $\mathcal{D}_S$ and a source task $\mathcal{T}_S$. For a binary classification problem, the training dataset would consist of n pairs of $\{\boldsymbol{x}_i, y_i\}$ values, where $i = 1, \dots, n$, the feature vector is $\boldsymbol{x}_i = [x_{i,1} \quad \cdots \quad x_{i,p}] \in X$, and $y_i \in Y$. It is assumed that there is sufficient training data available to train the machine learning model, and this data is usually balanced. Then, we use the information learned from this model and apply it to train new model with data having a related target domain $\mathcal{D}_T$ and task $\mathcal{T}_T$. Transfer learning aims to improve the learning of the target predictive function f_T using the knowledge previously acquired, when $\mathcal{D}_S \neq \mathcal{D}_T$ or $\mathcal{T}_S \neq \mathcal{T}_T$. Therefore, it is possible to estimate the target conditional probability distribution $P(Y_T|X_T)$ by using the information found through the training of the source model (See Figure 2.2). Some authors have proposed ways to quantify the relationships between different tasks, such as affinity matrices [5].

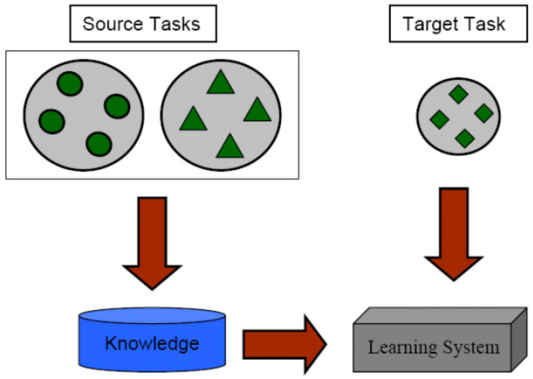

Figure 2.1:
Knowledge transfer in Transfer Learning [2].

Inductive transfer learning is applied when the source and target tasks are different, regardless if their respective domains are the same or not (when the source and target label spaces are different, $\mathcal{Y}_S \neq \mathcal{Y}_T$, or when the conditional probability distributions are different, $P(Y_S|X_S) \neq P(Y_T|X_T)$). Transductive transfer learning is used when the source and target tasks are the same, but their respective domains are different (when the source and target feature spaces are different, $\mathcal{X}_S \neq \mathcal{X}_T$, or when the source and target marginal probability distributions are different, $P(X_S) \neq P(X_T)$). Finally, unsupervised transfer learning is applied to problems where the target task is different and focuses on unsupervised learning problems such as clustering or dimensionality reduction. We will focus on and provide examples for a particular inductive transfer learning technique.

Inductive transfer learning techniques allow us to be very flexible when selecting the information that we want to transfer from one model to another. One cutting-edge approach is the implementation of quantum computing techniques, creating a hybrid classical-to-quantum (CQ) neural network. CQ transfer allows to pre-train a model with a classical deep neural network and successively manipulate the extracted features after removing the last layers with a variational quantum circuit [6]. Despite the early stages of quantum computing, CQ transfer learning approaches can compete with classical methods, which is very promising for the future of this area. We will now explore existing literature in the field of CQ transfer learning.

Contemporary research in CQ transfer learning uses a type of classifier known as a Variational Quantum Classifier (VQC). There are three main steps in a VQC circuit – encoding, classification and measurement [9]. Encoding involves preparing a superposed state of all the qubits q_n in the $|0\rangle$ state using Hadamard gates. The classification step involves rotating the qubits by $(\theta_1, \theta_2, \ldots, \theta_n)$ radians using R_y gates. The angle of rotation in radians corresponds to the weights associated with each qubit which are constantly updated via a process known as backpropagation. Adjacent qubits are then entangled using the Controlled-NOT or CNOT gate. Entanglement allows the exchange of information between qubits. Finally, the measurement step involves destroying the superposed state and measuring the resulting value of the qubit. Because our goal was to incorporate a VQC with a classical deep learning network we had to add classical components to our VQC. This required the addition a classical pre-processing layer and a classical post-processing layer. The pre layer performed data embedding into the bare VQC and the post layer facilitated data readouts. This combination of classical layers with the VQC is known as a dressed quantum circuit.

It has to be mentioned here that unlike quantum algorithms such as Shor's factorization algorithm to factorise large composite numbers into their prime factors [10] or the Bernstein-Vazirani algorithm [11] to identify a secret string inputted by the user – CQ in transfer learning offers no exponential quantum speedup. We simply aim to develop a quantum classifier to produce a comparable performance to that of a classical classifier. This is a mere proof of concept. We do encourage future researchers in the field, however, to evaluate feature efficiency – or the ability of a CQ hybrid transfer learning method to produce just as good a result as a purely classical transfer learning method. This is because there is evidence to suggest that quantum clustering methods provided reasonable results with just 2 free parameters while classical algorithms required 23 free parameters to produce comparable results[9].

As a final note, we would like to share with a reader that we used a quantum simulator and not a real quantum computer. This is because we wanted ideal results which are difficult to obtain in this era of error-prone, fault-intolerant quantum computers. Literature estimates suggest that using real quantum computers can result in up to a ~20% drop in accuracy scores[9].

3. Methods

Image classification is an instance where the application of inductive transfer learning methods (when the source and target marginal probability distributions differ) is particularly useful. A neural network for image classification consists of an input layer where each neuron corresponds to one of the features x_j (where $j = 1, \ldots, p$), a varying number of hidden layers, and an output layer where each neuron corresponds to one of the labels of Y. On a neural network, each layer after the input layer capture slightly more complex features in the images. The first layers' neurons activate when there are specific edges in certain regions of the images, and subsequent layers start detecting more complex patterns and shapes. The final layer receives as an input more easily identifiable and complex features, so that it is easier to correctly detect which features are found on each label of Y.

A common approach to applying transfer learning for computer vision is to utilize a previously trained model on data with the same feature space and transfer its trained parameters (**parameter transfer**). The final layers of the pre-trained neural network are removed, and the remaining layers and their weights are fixed and used as feature extraction models for the new data in the target domain and task. The final layers of the model are then retrained again on the target data, since they are intuitively more representative of the features found in the data they were originally trained for [3]. This process is shown in Figure 3.1 below. This approach is known as **feature extraction** or **embedding**. This method improves the training time and model performance. In the following section we will apply the feature extraction technique as an example.

To put this technique to practice we will use the ResNet-18 convolutional neural network architecture by training it from scratch (Model 1) using the *hymenoptera_data* dataset [4]. The ResNet-18 architecture consists of 18 convolutional layers using 3×3 kernels and shortcut connections with residual

mappings [1]. Each layer has a number of weights that connect the node's activations to the following layer's nodes. We have provided a more in-depth description of the ResNet architecture in our Kaggle report. We will apply a classical transfer learning approach (Model 2) and a CQ hybrid transfer learning approach using a 4-qubit dressed quantum circuit (Model 3) on the last layer to compare the performances of all three models.

We will use a model pre-trained on the 1000-class ImageNet dataset to transfer the weights for the first 17 layers. The model's final layer will consist of 512 input features and 2 final outputs (ants or bees). Figure 3.1 shows this transfer learning process, where A is the pre-trained model on the ImageNet dataset, A' is this same trained network minus its final layer, B is a second layer or group of layers that will be retrained on the target domain and for the target task. Our examples consist of the following settings:

- $\mathcal{D}_S$: 1000-class ImageNet dataset [7].
- $\mathcal{T}_S$: 1000 labels classification.
- $\mathcal{D}_T$: 246 labeled images of ants (124 images) and bees (122 images) [4].
- $\mathcal{T}_T$: 2 labels classification (ants or bees)

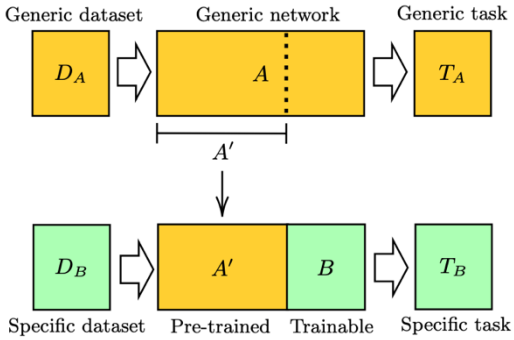

Figure 3.1:
Inductive Transfer Learning with a feature extraction or embedding approach [6].

4. Examples of the Technique in Practice

For Model 1, we trained the ResNet-18 network on domain $\mathcal{D}_T$ and task $\mathcal{T}_T$ from scratch. See Figure 4.1 for some samples of the images in the dataset. We did not expect the model to perform very well because the training data is way below 1000 images [4]. The *hymenoptera_data* dataset also includes a validation set of 153 labeled images (70 of ants and 83 of bees). We obtained the model's loss on both the training and validation sets during each epoch, and we used the validation set to measure the model's performance on unseen data (See Figure 4.5 below for a comparison with Models 2 and 3's loss per epoch).

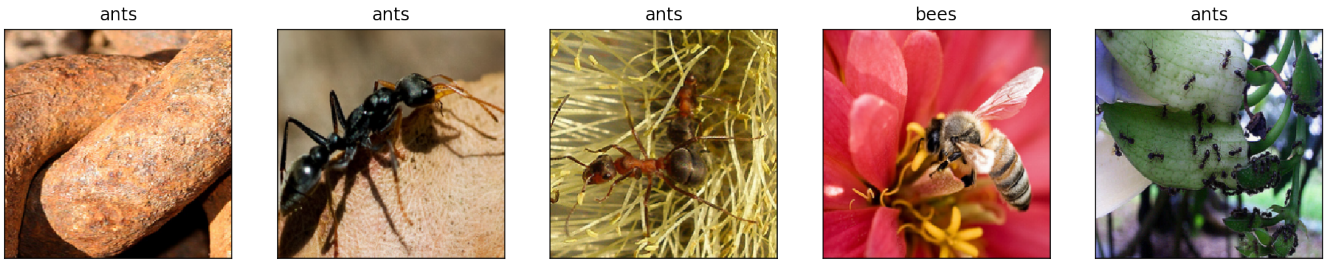

Figure 4.1:
Sample images from the hymenoptera_data *dataset.*

We set the hyperparameters of Model 1 to 30 epochs, a starting learning rate of 0.001 and a learning rate decay factor of 0.1 every 7 epochs. We also used cross entropy loss as the loss function during backpropagation and the model was trained in batches of 4 images. Model 1 achieved a best accuracy of 0.7255 on the validation set during epoch 11 and an AUC value of 0.7458. The confusion matrix for the

model's performance on the validation set with a threshold value of 0.5 is shown in Figure 4.2 along with a sample of five images and their predicted labels.

Model 1		Predicted Class	
		Ants	Bees
Actual Class	Ants	54	16
	Bees	26	57

Figure 4.2:
Confusion matrix for Model 1.

For Model 2, A' is a ResNet-18 model pre-trained on a subset of the ImageNet dataset $\mathcal{D}_S$ minus its 18th layer. When the weights of the pre-trained model are frozen, they can be used to pre-process any image as a feature extraction method and output activations each of the 512 general features in the last hidden layer, in order to best describe which ones are contained in the image. We train the last layer B with our *hymenoptera_data* dataset $\mathcal{D}_T$ for our two classes $\mathcal{T}_T$. Model 2 was trained using the same hyperparameters as Model 1. It achieved a best accuracy of 0.9542 on the validation set during epoch 19 and an AUC value of 0.9793. The confusion matrix for the model's performance on the validation set with a threshold value of 0.5 is shown in Figure 4.3. Model 2 represents a reduction of 56.99% in training time, an improvement of 31.52% in accuracy, and a 31.31% increase in AUC over Model 1.

Model 2		Predicted Class	
		Ants	Bees
Actual Class	Ants	67	3
	Bees	4	79

Figure 4.3:
Confusion matrix for Model 2.

For Model 3, A' is the same as Model 2 and is still used as a feature extraction method. B now is the 4-qubit dressed quantum circuit with a depth of 6 layers trained with our *hymenoptera_data* dataset $\mathcal{D}_T$ for our two classes $\mathcal{T}_T$. We chose to use 4-qubits as this seemed to be reasonable sweet-spot for simulating a quantum computer on classical hardware. Doubling the number of qubits to 8 exponentially increased computational time and was therefore not very realistic[12]. Model 3 was trained using the same hyperparameters as Model 1 but with cross entropy for loss function which was minimized using an Adam optimizer. We chose the Adam optimizer over Stochastic Gradient Descent as there was evidence to suggest that the former provided better results quicker. Model 3 achieved a best accuracy of 0.9608 on the validation set during epoch 25 and an AUC value of 0.9757. The confusion matrix for the model's performance on the validation set with a threshold value of 0.5 is shown in Figure 4.4. Model 3 represents an improvement of 32.43% in accuracy, and a 30.83% increase in AUC over Model 1.

Model 3		Predicted Class	
		Ants	Bees
Actual Class	Ants	65	5
	Bees	1	82

Figure 4.4:
Confusion matrix for Model 3.

See Figure 4.5 for a comparison of the ROC and Precision-Recall curves for Models 1, 2, and 3. It is now evident that the use of transfer learning technique can improve training time and performance very significantly. The loss functions decrease with both the classical and CQ approaches, while both AUC and AP improve by a considerable amount over the model built from scratch. The one disadvantage there exists about transfer learning is that there has to be an existing pretrained model on a related domain available for you to be able to use. For computer vision and image classification tasks this is not an issue as there is a wide range of available pre-trained models publicly available. However, if one were to undergo the training process from scratch on a new domain, the collection of enough labeled data and training of a new model would be expensive and time-consuming.

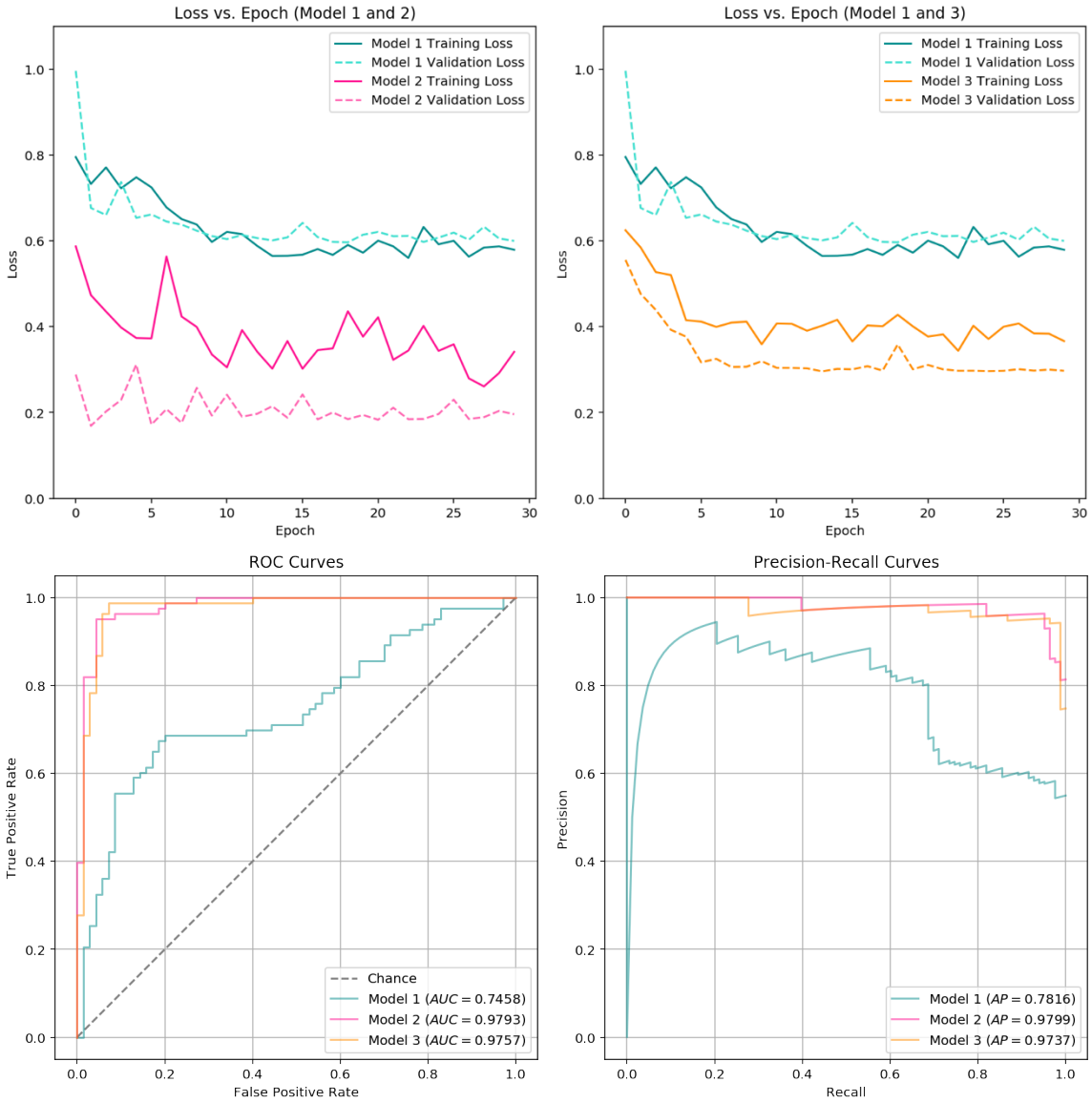

Figure 4.5:
Top left: Loss vs. Epoch for models 1 and 2. Top right: Loss vs. Epoch for models 1 and 3. Bottom left: ROC curves for models 1, 2, and 3. Bottom right: Precision-Recall curves for models 1, 2, and 3.

5. Summary

Transfer learning techniques can improve significantly on similar models built from scratch. Training time can be reduced by one half with classical parameter transfer methods, and performance in terms of AUC, AP and accuracy can also be improved significantly. These results can be attained with a minimum amount of training data as long as there is a pre-existing model trained in a similar domain. Both the classical and the CQ hybrid approach accomplished very similar performance in terms of accuracy and AUC. This both highlights the advantages of transfer learning and positive outlook of novel quantum methods to perform classifications.

6. References

1. He, Kaiming, et al. "Deep Residual Learning for Image Recognition." *ArXiv:1512.03385* [Cs], Dec. 2015. arXiv.org, http://arxiv.org/abs/1512.03385.
2. Pan, Sinno Jialin, and Qiang Yang. "A Survey on Transfer Learning." *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, Oct. 2010, pp. 1345–59. DOI.org (Crossref), doi:10.1109/TKDE.2009.191.
3. Razavian, Ali Sharif, et al. "CNN Features Off-the-Shelf: An Astounding Baseline for Recognition." *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, IEEE, 2014, pp. 512–19. DOI.org (Crossref), doi:10.1109/CVPRW.2014.131.
4. "Hymenoptera_data." *Kaggle*, https://www.kaggle.com/ajayrana/hymenoptera-data.
5. Zamir, Amir, et al. "Taskonomy: Disentangling Task Transfer Learning." *ArXiv:1804.08328 [Cs]*, Apr. 2018. arXiv.org, http://arxiv.org/abs/1804.08328.
6. Mari, Andrea, et al. "Transfer Learning in Hybrid Classical-Quantum Neural Networks." *ArXiv:1912.08278 [Quant-Ph, Stat]*, Dec. 2019. arXiv.org, http://arxiv.org/abs/1912.08278.
7. Sasank Chilamkurthy, *PyTorch transfer learning tutorial*. https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html. Accessed: 2020-04-13.
8. Arute, Frank; Arya, Kunal; Babbush, Ryan; Bacon, Dave; Bardin, Joseph C.; Barends, Rami; Biswas, Rupak; Boixo, Sergio; Brandao, Fernando G. S. L.; Buell, David A.; Burkett, Brian

(October 2019). "Quantum supremacy using a programmable superconducting processor". *Nature*. **574** (7779): 505–510

9. E. Gil Fuster, J. Lattore "Variational Quantum Classifier." Universitat de Barcelona
10. Shor, Peter W. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer." SIAM Journal on Computing 26.5 (1997): 1484–1509. Crossref. Web.
11. Michael A. Nielsen and Isaac L. Chuang. 2011. Quantum Computation and Quantum Information: 10th Anniversary Edition (10th. ed.). Cambridge University Press, USA.
12. Ville Bergholm, Josh Izaac, Maria Schuld, Christian Gogolin, Carsten Blank, Keri McKiernan and Nathan Killoran. PennyLane. arXiv, 2018. arXiv:1811.04968

7. Appendix

A.1. A Brief Introduction to the Fundamentals of Quantum Computing

The fundamental unit of quantum information is the quantum bit or the qubit. This is analogous to the bit in a classical computer. A qubit can exist in either the zero state $|0\rangle$, the one state $|1\rangle$ or in a linear combination of the above. The notation used above is known as a bra-ket notation first popularized by the English physicist Paul Dirac. A vector in the 'ket' notation is represented as $|v\rangle$ and is simply a column vector. In contrast, a vector in the 'bra' notation is represented as $\langle v|$ which is simply a row vector. The following equation represents the $|0\rangle$ and $|1\rangle$ in their equivalent column vector representations (Equation 1).

$$|0\rangle = \begin{bmatrix}1\\0\end{bmatrix} \; and \; |1\rangle = \begin{bmatrix}0\\1\end{bmatrix} \tag{1}$$

These qubits can also exist as a linear combination of $|0\rangle$ and $|1\rangle$ which is depicted as follows (Equation 2). This is a unique property of qubits and is known as the superposition principle.

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{2}$$

Here, α and β are known as the probability amplitudes associated with the qubit existing in the $|0\rangle$ and $|1\rangle$ state respectively. Without going into the mathematical intricacies associated, it will also be stated here that the actual probability of finding the qubit in the $|0\rangle$ state is α^2 and $|1\rangle$ state is β^2. In other words, the following equation (Equation 3) will always hold true.

$$\alpha^2 + \beta^2 = 1 \tag{3}$$

The probabilistic nature coupled with the superposition principle of qubits is in stark contrast to the deterministic nature of bits. However, it has to be stated here that the probabilistic nature of qubits only holds until the time the qubits are measured. In other words, the moment an external entity wishes to know the value of the qubit, the probabilistic nature quickly disappears, and the qubit assumes the value of either $|0\rangle$ or $|1\rangle$ depending on the values of α^2 and β^2.

Next, it is important to understand why quantum computers may have added computational powers and may be able to offer a certain quantum-speedup in certain types of problems. We shall be able to understand this by comparing how bits and qubits operate. For example, consider the scenario where we have access to 50 bits and 50 qubits. Because each bit can either take the value of 0 or 1, the total number of different combinations in which the system can exist in is 2^{50}. However, at any given point of time, only 1 combination out of 2^{50} can exist. In the case of qubits, however, all 2^{50} simultaneously exist in superposition as a linear combination. This parallel processing ability is what gives quantum computers the advantage. It has to be mentioned here that while the above explanation is lucid, it is also vastly simplistic and even mildly erroneous. If parallel processing was indeed the reason why quantum computers have promising applications, then all classical algorithms must have a comparable quantum algorithm with a verifiable quantum speedup. This, however, is not the case and most classical problems cannot be solved faster by a quantum computer. There are a handful of problems where quantum computers offer an exponential speedup relative to their classical counterparts. The hope is that more such problems will be identified in the future with advancements in quantum information science (Figure 1).

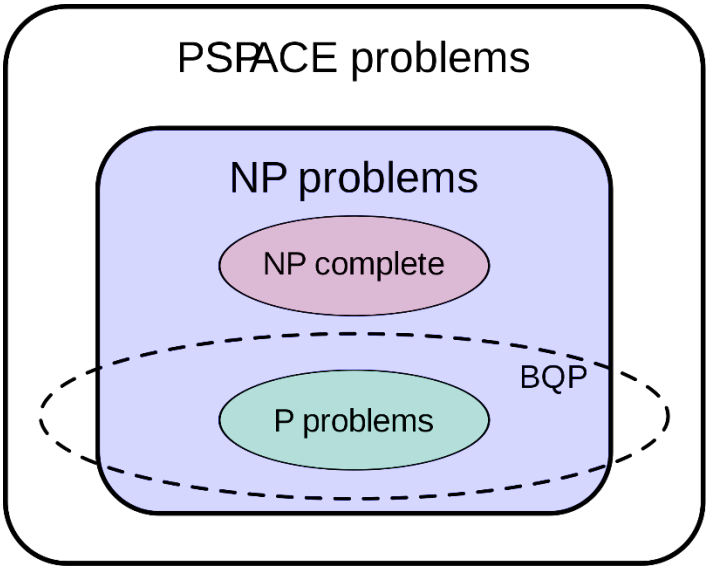

Figure 1. *P refers to the set of all problems that can solved by a classical computer in polynomial time. NP refers to the set of all such problems for which given a solution, it can be verified in polynomial time. Finally, BQP refers to all such problems that can be solved by a quantum computer in polynomial time. Note that* $P \subset BQP$. *The hope is that we find more problems in the set* $BQP \cap P'$. *Also, while still under debate, we have assumed* $P \neq NP$.

If a quantum computer does not entirely rely on parallel processing to produce speedups, how does a quantum computer work then. According to the fundamental of quantum physics, if all possible solutions to a given problem were super positioned as a linear combination of each other, then each solution would be just as likely as any other. In other words, this is not how a quantum computer works. In contrast, a sequence of carefully chosen quantum gates (similar to logic gates in classical computing) are applied on the qubits. These gates can be viewed as simple matrix multiplications on super positioned qubits. The result of the application of such gates is that most trivial, unimportant results will collapse (similar to waves destructively interfering or simply cancelling out). The remaining states that have non-zero amplitudes and hence non-zero probabilities (these are akin to waves constructively interfering and adding up) hopefully contain the solution to our problem. We will look at one example of a quantum gate to get an idea. Other quantum gates used in this literature have also been included later. This is the Hadamard gate as given in Equation (4).

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{4}$$

The Hadamard gate when applied to the $|0\rangle$ and $|1\rangle$ qubit yields the following (Equations 5 & 6). The results are also known as the Hadamard-mapped states.

$$H\,|0\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 1 \end{bmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \tag{5}$$

$$H\,|1\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ -1 \end{bmatrix} = \frac{|0\rangle - |1\rangle}{\sqrt{2}} \tag{6}$$

Note that squaring the coefficients of the resulting states into which the Hadamard gates maps $|0\rangle$ and $|1\rangle$ yields 0.5 in both cases (Squaring $\frac{1}{\sqrt{2}}$). In other words, the application of the Hadamard gate creates superposition of the qubits wherein both the $|0\rangle$ and the $|1\rangle$ have a 0.5 probability of existing. Simply put, if a user where to measure a Hadamard mapped qubit there is a 50% chance the result will be $|0\rangle$ and a 50% chance the qubit will be $|1\rangle$. Other quantum gates used in this literature have been included in the appendix.

Another gate that we used in the construction of the dressed quantum circuit was the R_y gate. This is given by the following matrix (Equation 7).

$$R_y(\theta) = \begin{bmatrix} cos\frac{\theta}{2} & -sin\frac{\theta}{2} \\ -sin\frac{\theta}{2} & cos\frac{\theta}{2} \end{bmatrix} \tag{7}$$

Finally, the following is the Controlled-Not gate or the CNOT gate. This operates on two qubits at a given time. Two qubits can potentially exist in a superposition of four different states. The four states are as follows (Equation 8).

$$|00\rangle = \begin{bmatrix}1\\0\\0\\0\end{bmatrix}, |01\rangle = \begin{bmatrix}0\\1\\0\\0\end{bmatrix}, |10\rangle = \begin{bmatrix}0\\0\\1\\0\end{bmatrix}, |11\rangle = \begin{bmatrix}0\\0\\0\\1\end{bmatrix} \quad (8)$$

The CNOT gate itself is given as follows (Equation 9)

$$CNOT = \begin{bmatrix}1 & 0 & 0 & 0\\0 & 1 & 0 & 0\\0 & 0 & 0 & 1\\0 & 0 & 1 & 0\end{bmatrix} \quad (9)$$