RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

utbm
université de technologie Belfort-Montbéliard

UNIVERSITÉ DE TECHNOLOGIE DE BELFORT-MONTBÉLIARD

Développement d'algorithme quantique pour la création musicale et la génération d'images

Rapport de stage ST40 - A2024

GACHELIN Estouan
Nom de la spécialité d'Ingénieur
Spécialité Informatique

Université de Belfort-Montbéliard
90010 BELFORT cedex
www.utbm.fr

HOLWECK Frédéric
NOM Prénom

LAURI Fabrice
NOM Prénom

Chapitre 1

Introduction

1.1 Remerciements

Avant tout, je tiens à exprimer ma gratitude envers toutes les personnes qui m'ont accueilli, accompagné et aidé durant la réalisation de ce stage.

Je souhaite tout d'abord remercier Monsieur Holweck, qui m'a proposé ce stage et m'a apporté toutes les informations nécessaires, que ce soit pendant le stage ou lors du semestre précédent, où j'ai eu l'opportunité de suivre son cours d'introduction à l'algorithmie quantique. Ce cours m'a profondément inspiré et donné l'envie d'explorer davantage ce domaine.

Je tiens également à remercier chaleureusement Djim et Koffi, mes collègues de bureau. Forts de leur expérience et de leurs connaissances, ils n'ont pas hésité à me conseiller et à m'accompagner tout au long de ce stage.

Un remerciement spécial va aussi à l'UTBM et à son équipe, qui m'ont accueilli et ont su me fournir un environnement de travail à la fois agréable et stimulant.

Enfin, je remercie le programme QuantEdu porté par l'UBFC pour la mise en place du projet, qui m'a permis de réaliser ce stage et d'enrichir mes compétences.

1.2 Le sujet

1.2.1 Présentation du sujet

Dans le cadre de mon stage de ST40 pour mon diplôme d'ingénieur, j'ai participé à un projet financé par la Région Bourgogne-Franche-Comté, dont l'objectif principal était de sensibiliser le public aux technologies quantiques. L'initiative visait à démocratiser la compréhension du quantique et à susciter l'intérêt pour ce domaine en pleine émergence.

Pour rendre le sujet accessible et attractif, il a été décidé d'utiliser des applications visuelles et artistiques, telles que la génération d'images et de musique. Ces approches permettent non seulement de démontrer concrètement ce que les technologies quantiques rendent possibles, mais également d'éveiller la curiosité par des résultats tangibles et étonnants.

En parallèle de la mise en œuvre technique, une des missions essentielles du stage consistait à rédiger un rapport explicatif à destination des néophytes. Ce document a pour objet de vulgariser les concepts utilisés et de présenter clairement le fonctionnement des travaux réalisés.

Ce rapport détaille les différentes étapes du projet, les outils et méthodes utilisés, ainsi que les résultats obtenus au cours de cette expérience.

1.2.2 Enjeux du sujet

Les technologies quantiques sont appelées à jouer un rôle majeur dans de nombreux domaines, tels que l'informatique, la cryptographie, l'optimisation ou encore la modélisation de systèmes complexes. Elles offrent des capacités de calcul bien supérieures à celles des ordinateurs classiques pour certains types de problèmes,

ouvrant la voie à des avancées scientifiques et industrielles significatives.

Avec l'essor de l'intelligence artificielle et le besoin croissant de puissance de calcul, les technologies quantiques sont de plus en plus perçues comme une solution prometteuse. Cette perspective entraîne un développement rapide des investissements dans le domaine, que ce soit au niveau de la recherche fondamentale, des infrastructures ou des applications commerciales.

Cependant, leur complexité intrinsèque les rend souvent difficiles à comprendre pour un public non spécialisé. Cette barrière à la compréhension limite l'engagement et l'intérêt d'un plus large public, ainsi que le développement des compétences nécessaires pour répondre à la demande croissante dans ce secteur.

Le projet auquel j'ai participé s'inscrit dans une démarche de vulgarisation et de démocratisation. En présentant des applications concrètes et accessibles, comme la génération d'images et de musique, il vise à illustrer les potentialités des technologies quantiques de manière intuitive et engageante. Ce type d'approche permet non seulement de sensibiliser un large public, mais aussi de poser les bases pour encourager de futures vocations dans un secteur porteur d'innovation.

1.3 L'UTBM

L'Université de Technologie de Belfort-Montbéliard (UTBM) est une grande école d'ingénieurs française, créée en 1999. Elle s'inscrit dans le réseau des universités de technologie, aux côtés de l'UTC (Université de Technologie de Compiègne) et de l'UTT (Université de Technologie de Troyes). Sa mission principale est de former des ingénieurs polyvalents, aptes à relever les défis technologiques et industriels contemporains dans les domaines suivants :

- **Transition énergétique** : Étude et développement de solutions pour optimiser les systèmes énergétiques et intégrer les énergies renouvelables dans des infrastructures complexes ;
- **Technologies numériques** : Recherche sur l'intelligence artificielle, les systèmes embarqués et les technologies de communication pour accompagner la transformation numérique ;
- **Systèmes mécaniques et électroniques** : Conception, optimisation et automatisation de machines et dispositifs complexes pour l'industrie moderne ;
- **Mobilité et transports** : Innovation dans les moyens de transport durables, autonomes et connectés ;
- **Hydrogène et énergies nouvelles** : Développement de technologies autour de l'hydrogène, telles que la production, le stockage et l'utilisation dans des systèmes énergétiques avancés.

1.3.1 Recherche et innovation

L'UTBM joue un rôle clé dans la recherche scientifique et technologique, en menant des projets innovants dans des domaines émergents tels que :

- Les nouvelles technologies quantiques, et notamment leur application à l'informatique ;
- L'intelligence artificielle, utilisée pour résoudre des problématiques complexes dans les secteurs industriels, environnementaux et sociaux ;
- L'industrie 4.0, avec le développement de procédés de fabrication intelligents et interconnectés ;
- Les énergies alternatives et le développement durable, avec des recherches visant à réduire l'impact écologique des systèmes énergétiques ;
- L'hydrogène comme solution énergétique d'avenir, avec un focus sur les procédés de production écologiques et les applications dans les transports et l'industrie.

Les collaborations fréquentes avec des partenaires industriels et académiques permettent de lier recherche fondamentale et applications pratiques, tout en répondant aux besoins des entreprises et de la société.

Chapitre 2

Notions de base en informatique quantique

Afin de mieux comprendre les futurs concepts qui vont être abordés dans ce rapport, nous allons tout d'abord effectuer une introduction au concept d'algorithmie quantique. Cependant, il est essentiel de souligner que l'intérêt des ordinateurs quantiques ne réside pas uniquement dans leur rapidité d'exécution, mais dans leur capacité à exploiter de nouveaux algorithmes, souvent bien plus efficaces que ceux utilisés sur les ordinateurs classiques.

2.1 Qubits et portes quantiques

2.1.1 Les qubits

: La principale différence entre l'informatique classique et quantique réside dans la manière dont les données sont encodées et manipulées. En algorithmie classique, nous utilisons ce que nous appelons les bits qui peuvent être représentés comme des variables qui prennent soit la valeur de 1 soit la valeur 0. Ensuite, nous utilisons des portes logiques qui nous permettent de faire des opérations avec ces bits. En enchaînant un nombre suffisant de ces portes logiques, nous pouvons mettre en place un grand nombre d'opérations complexes.

En algorithmie quantique, nous utilisons ce que l'on appelle des qubits pour bits quantiques. Leurs différences est que, contrairement aux bits classiques, ils sont à la fois 0 et 1. Nous les écrivons donc comme une somme de ces états avec une amplitude pour chaque état qui est directement reliée à la probabilité pour eux d'être 0 ou 1. Nous pouvons alors utiliser une opération de mesure qui projettera l'état du qubit soit sur 0, soit sur 1 selon leur probabilité respective. Mais mesurer des qubits en projetant la valeur d'un qubit sur 0 ou 1 ne le rendra pas très différent d'un bit classique et donc empêchera souvent les avantages liés au quantique. C'est pour cela que la mesure est habituellement faite en fin d'algorithme. Mathématiquement, un qbits est écrit de la manière suivante :

$$|qubit\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{2.1}$$

avec α et β qui sont deux nombres complexes tels que le carré du module de α (ou β), soit la probabilité que, lors de la lecture, le qubit prenne la valeur 0 (ou 1).

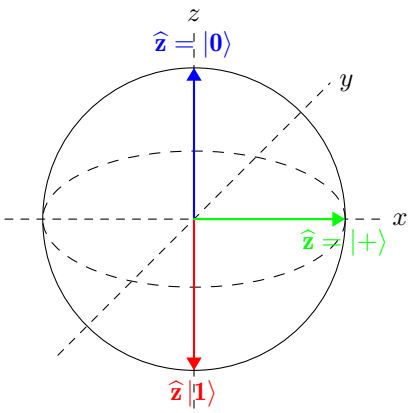
Par exemple, pour α, un nombre complexe comme ceci $\alpha = \frac{\sqrt{3}}{4} + \frac{1}{4}i$, la probabilité que le qubit soit mesuré à 0 est alors égale à $\frac{\sqrt{3}}{4}^2 + \frac{1}{4}^2 = \frac{3}{16} + \frac{1}{16} = \frac{1}{4}$ Puisque les qubits prend nécessairement les valeurs 0 ou 1, la somme des probabilités que les valeurs soient à 0 ou 1 est égale à 1.

On en déduit que la somme des carrés des modules de α et β vaut 1 car représente l'ensemble des sorties possibles.

Pour revenir à notre exemple, la probabilité que le qubit vaille 1 ce qui équivaut au carré du module de β est donc de $\frac{3}{4}$.

nous constatons donc que, si nous connaissons α, il est possible de trouver le carré du module de β, il existe alors une infinité de solutions que peut prendre β. Au niveau mathématique, un qubit est un vecteur appartenant à $\mathbb{C}^2$(L'ensemble des couples complexes) et de norme 1.

Une méthode fréquemment utilisée pour représenter les qubits est la sphère de Bloch.



Ici sont représentés trois qubits, un bleu, un rouge et un vert. Le qubit bleu est le qubit tel que $|q_1\rangle = |0\rangle$. C'est-à-dire, quand on va mesurer ce qubit, il prendra forcément la valeur 0. En rouge, la même chose, mais avec 1. Pour finir en vert, le vecteur $|q_3\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ qui est donc un vecteur qui, a la lecture, à 50 % de chance de valoir 1 ou de valoir 0. La sphère de Bloch fonctionne ainsi : "la hauteur" représente l'amplitude donc directement la probabilité de mesurer 0 ou 1. Quand à l'angle autour de l'axe z, il représente la phase du qubit, qui n'a pas d'impact direct sur les probabilités lors de la mesure, mais dont on verra l'importance plus tard dans ce document.

2.1.2 Les portes quantiques

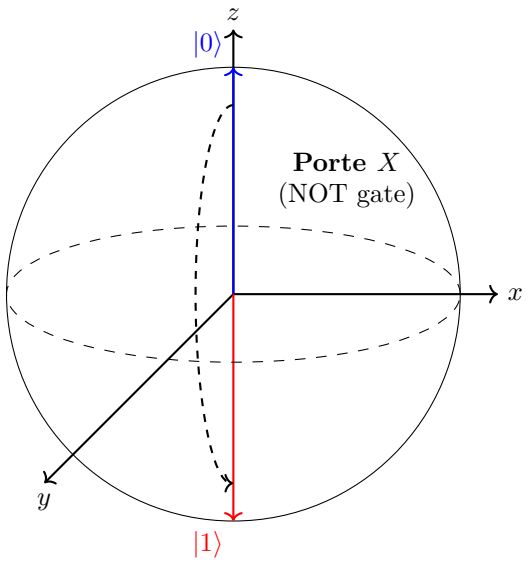
Maintenant, étudions les portes quantiques. En algorithmie classique ou quantique, les portes sont les plus petites opérations réalisables par l'ordinateur. En effet, tout algorithme n'est en fait qu'une suite de ces portes. Les portes prennent donc en entrée des informations, les traitent et sortent de nouvelles informations qui pourront être utilisées de nouveau. Par exemple, la porte classique AND prend en entrée deux bits d'information et sortira un bit d'information qui sera égale à 0 sauf si les deux bits en entrée sont égaux à 1.

Entrée A	**Entrée B**	**Sortie (A AND B)**
0	0	0
0	1	0
1	0	0
1	1	1

Table de vérité de la porte AND

En classique, nous utilisons 3 portes (or, and et not). Ces portes composent l'entièreté des algorithmes utilisés de nos jours.

Pour ce qui est des portes quantiques, les opérations consistent à faire une rotation du vecteur par rapport à un axe sur la sphère de Bloch. Par exemple, on considère la porte quantique X, une porte qui effectue la rotation d'angle π autour de l'axe X. Cette porte transformera l'état $|0\rangle$(vecteur bleu) en l'état $|1\rangle$(vecteur rouge) et inversement. Cette porte est aussi appelée NOT GATE.



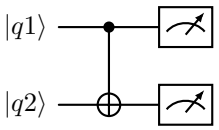
Il existe aussi des portes de rotation autour des autres axes z et y. De plus, nous avons pris un exemple avec une rotation de π mais nous pouvons choisir n'importe quelle valeur réelle. On considère les portes X Y et Z comme étant des rotations de π et les portes $R_X(\alpha)$,$R_Y(\alpha)$,$R_Z(\alpha)$ comme étant des portes de rotation α.

On peut aussi considérer des portes qui sont une succession de portes.
Par exemple la porte d'Hadamard qui transforme $|0\rangle$ en $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ et $|1\rangle$ en $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ est la succession des portes X et $R_Y(\frac{\pi}{2})$.

Il existe ensuite des portes ayant des impacts sur plusieurs qubits. Par exemple, les portes contrôlées qui fonctionnent de la manière suivante : elles prennent en entrée un qubit dit "target", un qubit de "contrôle", ainsi qu'un effet qui est appliqué sur le qubit "target" si le qubit de "contrôle" est à 1.
Exemple de la plus classique de ces portes : la porte CNOT pour control NOT. Dans cette porte si q1(le qubit de contrôle) vaut 1 on applique un NOT sur q2 (le qubit de target) on obtient donc la table de vérité suivante pour le circuit suivant :

$q1$	$q2$	$q1'$	$q2'$
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

que l'on peut représenter par le circuit suivant :



De cette façon, nous pouvons rajouter un contrôle sur toutes les portes vues précédemment. Il est aussi possible d'augmenter le nombre de qubits contrôlés. Par exemple, on peut implémenter une porte CCNOT qui contrôle deux qubits et applique NOT sur un troisième si les deux premiers valent tous les deux 1.

Maintenant que cette première approche a été effectuée, intéressons-nous à une description plus mathématique. Les portes quantiques peuvent être représentées par des matrices carrées de taille 2^n, où n est le nombre de qubits concerné. Pour qu'une matrice soit une porte quantique valide, elle doit être unitaire, c'est-à-dire qu'il existe une matrice $A^\dagger$, égale à la transposée conjuguée de A, telle que $A * A^\dagger = I$, où I est la matrice identité. Ces propriétés des portes font que ces portes sont, notamment, des opérations linéaires qui gardent la normalisation des qubits, mais sont aussi des opérations réversibles.
Au niveau des opérations, appliquer une porte X à un qubit puis une porte Z, revient à appliquer une porte $Z * X$. Appliquer une porte X à un premier qubit et une porte Y à un deuxième qubit revient à appliquer

une porte définit par la matrice $\mathbf{X} \otimes \mathbf{Y}$(appelé produit tensoriel) qui est donc bien une matrice de taille 4, car impactant 2 qubits.

Exemple :Soit la matrice de Pauli X (porte NOT quantique) donnée par :

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Et la matrice Y (une porte Pauli-Y) donnée par :

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$$

Le produit tensoriel $X \otimes Y$ est donné par :

$$X \otimes Y = \begin{pmatrix} X_{00} \cdot Y & X_{01} \cdot Y \\ X_{10} \cdot Y & X_{11} \cdot Y \end{pmatrix} = \begin{pmatrix} 0 \cdot Y & 1 \cdot Y \\ 1 \cdot Y & 0 \cdot Y \end{pmatrix}$$

En développant :

$$X \otimes Y = \begin{pmatrix} 0 & 0 & 0 & -i \\ 0 & 0 & i & 0 \\ 0 & -i & 0 & 0 \\ i & 0 & 0 & 0 \end{pmatrix}$$

Ainsi, $X \otimes Y$ est une matrice de taille 4×4, correspondant à une porte qui agit sur deux qubits.

En appliquant de manière différente ces diverses portes, il est possible de programmer et créer toute sorte d'algorithme répondant à toute sorte de problème.

2.1.3 Particularités de l'algorithmie quantique

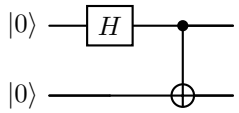
Les états quantiques étant des vecteurs, ils peuvent être en superposition. Par exemple, l'état $|\psi_1\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ est la superposition des deux états de base $|0\rangle$ et $|1\rangle$. Si on considère maintenant un système à deux qubits, on peut générer un état correspondant à la superposition de tous les états de base :

$$|\psi_2\rangle = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle).$$

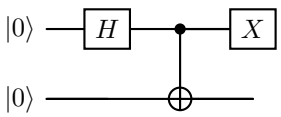
Du point de vue des portes quantiques, cet état $|\psi_2\rangle$ est obtenu en appliquant la porte de Hadamard à $|00\rangle$: $|\psi_2\rangle = H \otimes H \, |00\rangle$.

D'une manière générale, on obtient un état complétement parallélisé en appliquant n portes de Hadamard à l'état initial $|0 \dots 0\rangle$. Sur cet état superposé, on pourra, à travers le circuit quantique, appliquer simultanément des opérations sur tous les états de base.

Un autre phénomène important de l'algorithmie quantique est l'intrication. L'intrication est un phénomène dans lequel deux qubits partagent un état commun, indépendamment de la distance qui les sépare. Par exemple, dans un circuit à deux qubits, si nous appliquons une porte H sur le premier qubit et une porte CNOT qui contrôle le premier et s'applique sur le deuxième, nous obtenons l'état $|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. Dans cet état, si nous décidons de faire une mesure sur un des qubits, cela reviendra à aussi projeter le deuxième, bien que nous n'ayons pas directement fait de mesure sur celui-ci. En effet, le deuxième qubit ne pourra pas prendre une autre valeur que celui du premier.



Il existe toutes sortes d'états intriqués où les deux valeurs ne sont pas nécessairement les mêmes. Par exemple, si on prend le même circuit, mais on rajoute une porte X à la fin sur l'un des deux qubits, on obtient alors l'état $|\psi\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$. Cet état est aussi intriqué, car la mesure d'un des qubits changera aussi l'état du deuxième.



2.2 Algorithme de Grover

L'algorithme de Grover est l'un des premiers algorithmes quantiques ayant démontré un gain significatif par rapport à l'algorithmie classique. Cet algorithme a de nombreuses applications et sera utilisé dans la création de nos futurs algorithmes. Sa principale utilité réside dans la recherche d'un élément dans un tableau non trié, mais il peut également servir dans d'autres contextes, comme la génération de sorties aléatoires avec un nombre indéfini de résultats.

Pour illustrer son fonctionnement, nous allons présenter l'algorithme dans le cas de la recherche dans un tableau.

2.2.1 Exemple : Recherche dans un tableau

Imaginons que nous avons une liste de valeurs codées sur 3 bits, accompagnées de leurs index respectifs (également codés sur 3 bits). Le tableau est le suivant :

Valeur (3 bits)	**Index (3 bits)**
101	000
011	001
110	010
000	011
111	100
001	101
010	110
100	111

Supposons maintenant que nous cherchons l'index de la valeur 4 (représentée en binaire par 100). Avec un algorithme classique, puisque le tableau n'est pas trié, nous devrions effectuer en moyenne $\frac{N}{2}$ vérifications (avec N le nombre d'éléments dans le tableau, ici $N = 8$) pour trouver l'index de cet élément. En revanche, l'algorithme de Grover permet de trouver cet index en $\mathcal{O}(\sqrt{N})$ opérations. Bien que pour des valeurs de n relativement petites, le gain puisse sembler négligeable, pour des valeurs élevées, la réduction du nombre de calculs devient significative.

2.2.2 Fonctionnement de l'algorithme

L'algorithme de Grover commence par initialiser un état quantique où toutes les valeurs du tableau sont en superposition. C'est-à-dire dans notre exemple $\Psi = \frac{1}{\sqrt{8}}(|101000\rangle + |011001\rangle + ... + |100111\rangle)$. Cela implique que les trois premiers qubits représentent les valeurs et les trois suivants les indices correspondants. L'algorithme agit principalement sur les qubits représentant les indices, bien qu'il prenne en compte l'ensemble des qubits pour effectuer ses calculs.

Il est composé de deux composants clés :

— **L'oracle** : son rôle est de "marquer" les états correspondant à la valeur recherchée en leur appliquant une phase négative.

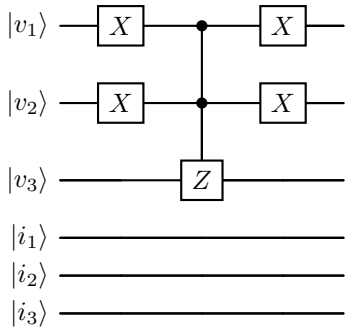
— **Le diffuseur** : il amplifie la probabilité des états marqués pour faciliter leur mesure.

Ces deux étapes sont répétées $\mathcal{O}(\sqrt{N})$ fois.

L'Oracle

Un oracle est un ensemble de portes quantiques caractéristiques, conçu pour accomplir une tâche spécifique. Dans le contexte de l'algorithme de Grover, l'oracle est une combinaison de portes qui identifie et marque un état cible.

Prenons l'exemple de la recherche de la valeur 4 (100 en binaire). L'oracle marque cet état en inversant sa phase, rendant ainsi cet état distinct des autres lors des étapes ultérieures de l'algorithme. Voici un exemple de circuit quantique correspondant à cet oracle :



Dans ce circuit, les portes X sont utilisées pour préparer l'état de contrôle, et les portes Z appliquent une inversion de phase à l'état cible lorsque les conditions sont remplies.
Après l'application de l'oracle, l'état correspondant à la valeur 4 (100 en binaire) aura une phase négative, tandis que les autres états resteront inchangés. Cette caractéristique permet de différencier cet état au cours des étapes d'amplification d'amplitude de l'algorithme de Grover.

Le Diffuseur

Le rôle du diffuseur est d'augmenter la probabilité des états marqués. Si un seul état est marqué (comme dans notre exemple), le diffuseur amplifie uniquement sa probabilité. Si plusieurs états sont marqués (grâce à plusieurs oracles), le diffuseur agit de manière similaire sur chacun d'eux. Le diffuseur remet aussi la phase de l'état ciblé à sa valeur d'origine. Il est donc important de refaire passer notre système dans l'oracle avant le diffuseur si nous voulons refaire l'algorithme.

Après $\sqrt{N}$ itération(N le nombre d'éléments) de l'oracle puis le diffuseur, nous obtenons une situation où nous avons presque la certitude d'obtenir la sortie que nous voulons : ce sera donc l'état constitué à la fois de l'index que nous voulons et la valeur dont nous voulions trouver l'index.

2.2.3 Analyse mathématique : Pourquoi cela fonctionne ?

L'algorithme de Grover repose sur la mécanique quantique et utilise deux concepts essentiels : la superposition et l'interférence constructive/destructive.
L'état initial du système est une superposition uniforme de tous les états possibles :

$$|\psi_0\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle ,$$

où N est le nombre total de configurations possibles.
L'oracle applique une phase négative à l'état recherché $|t\rangle$, ce qui donne :

$$U_f |x\rangle = \begin{cases} -|x\rangle , & \text{si } x = t, \\ |x\rangle , & \text{sinon.} \end{cases}$$

L'amplification des probabilités repose sur un processus itératif qui augmente la probabilité d'observer l'état recherché $|t\rangle$. Cette évolution est gouvernée par l'angle de rotation θ, défini comme :

$$\theta = \arcsin\left(\frac{1}{\sqrt{N}}\right),$$

où N est le nombre total d'états ($N = 2^n$ pour un registre de n qubits).
Après k itérations de l'algorithme, la probabilité $P_k(|t\rangle)$ de mesurer l'état recherché $|t\rangle$ est donnée par :

$$P_k(|t\rangle) = \sin^2\left((2k+1)\cdot\theta\right).$$

Pour un registre de 4 qubits ($N = 16$), on a :

$$\theta = \arcsin\left(\frac{1}{\sqrt{16}}\right) = \arcsin(0.25).$$

Numériquement, cela donne :

$$\theta \approx 0.2527 \text{ radians.}$$

En utilisant la formule précédente, nous obtenons les probabilités suivantes :

$$P_k(|t\rangle) = \sin^2\left((2k+1)\cdot 0.2527\right).$$

— $k = 0$ **(état initial)** :

$$P_0(|t\rangle) = \sin^2(1 \cdot 0.2527) = \sin^2(0.2527) \approx 0.0625 \quad (6.25\%).$$

— $k = 1$ **(1ère itération)** :

$$P_1(|t\rangle) = \sin^2(3 \cdot 0.2527) = \sin^2(0.7581) \approx 0.472715 \quad (47.27\%).$$

— $k = 2$ **(2ème itération)** :

$$P_2(|t\rangle) = \sin^2(5 \cdot 0.2527) = \sin^2(1.2635) \approx 0.908504 \quad (90.85\%).$$

— $k = 3$ **(3ème itération)** :

$$P_3(|t\rangle) = \sin^2(7 \cdot 0.2527) = \sin^2(1.7689) \approx 0.961266 \quad (96.13\%).$$

— $k = 4$ **(4ème itération)** :

$$P_4(|t\rangle) = \sin^2(7 \cdot 0.2527) = \sin^2(2.2743) \approx 0.581529 \quad (58.15\%).$$

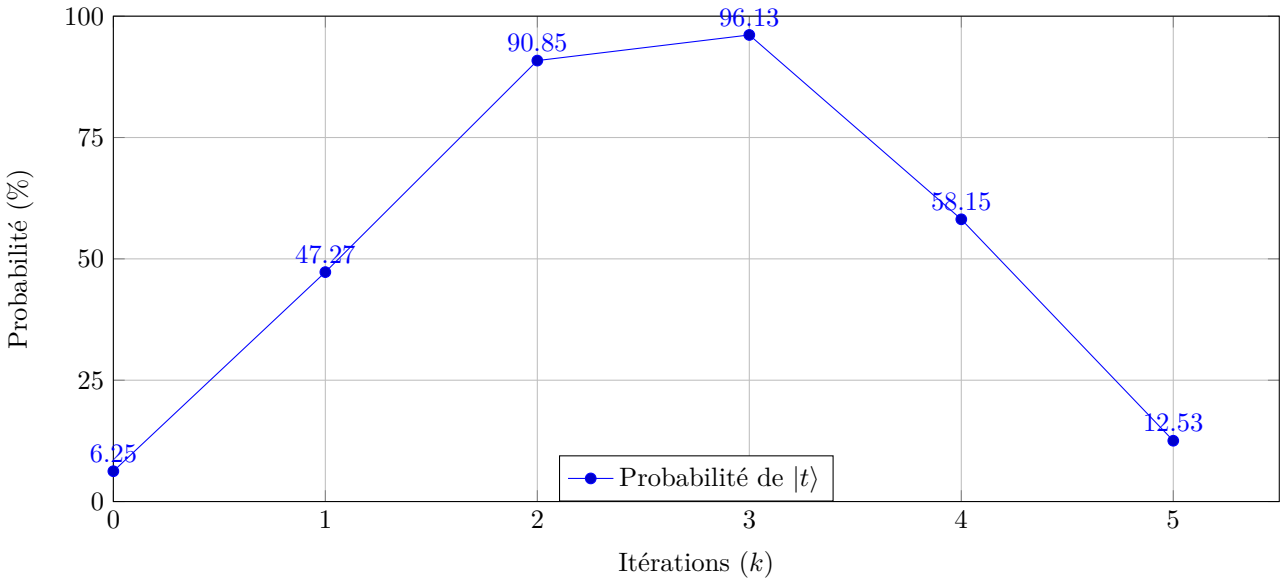
— $k = 5$ **(5ème itération)** :

$$P_5(|t\rangle) = \sin^2(7 \cdot 0.2527) = \sin^2(2.7797) \approx 0.125348 \quad (12.53\%).$$

Représentation tabulaire
Pour résumer ces résultats :

Itération	$P_k(\|t\rangle)$ (probabilité de l'état recherché)
0	6.25%
1	47.27%
2	90.85%
3	96.13%
4	58.15%
5	12.53%

Visualisation graphique



Évolution de la probabilité de mesurer $|t\rangle$ en fonction des itérations.

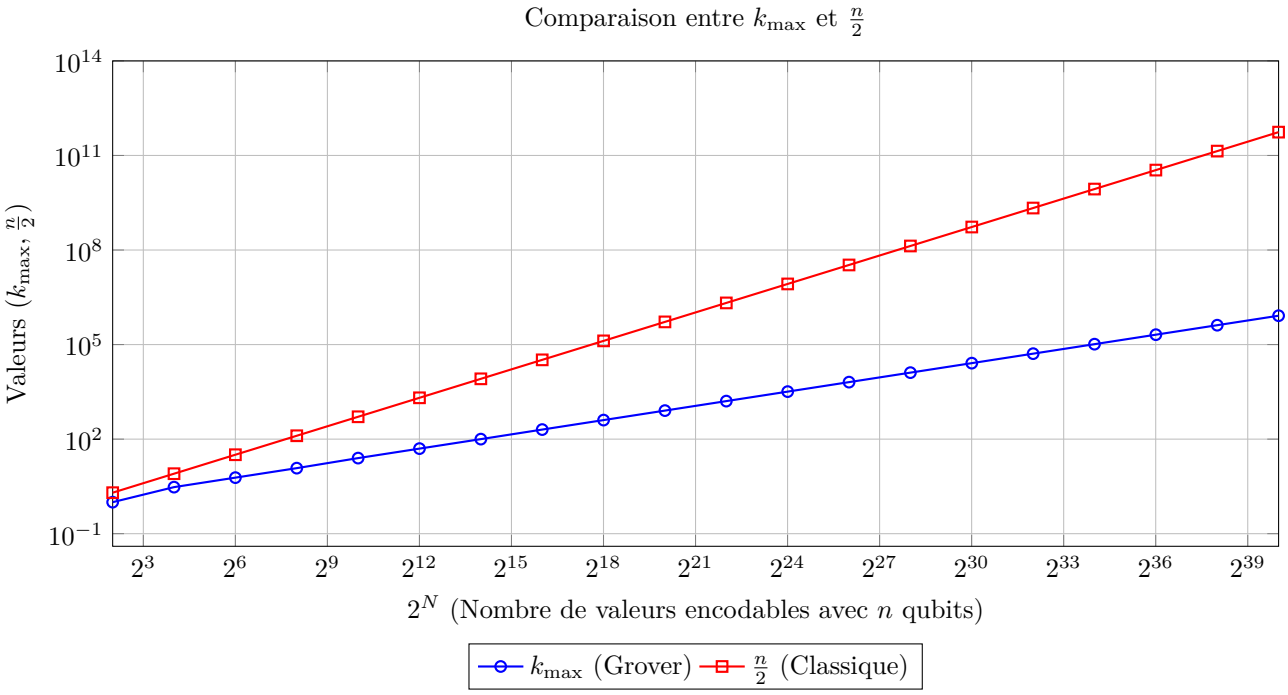
On observe que la probabilité de l'état recherché $|t\rangle$ approche 100% après $k = 3$ itérations, qui correspond au nombre optimal pour maximiser cette probabilité dans ce cas ($N = 16$). Au-delà de ce nombre d'itérations, la probabilité commence à diminuer en raison de la nature oscillatoire de la fonction sinus.

Maintenant si on veut visualiser l'évolution pour d'autres valeurs de N quelles sont les valeurs optimums de k et quelles sont leurs probabilités associées.

Nb qubits(n)	**Nb valeur encodée (N)**	$k_{\max}$	$p_{k_{\max}}$
2	4	1	1.000000
3	8	2	0.945313
4	16	3	0.961319
5	32	4	0.999182
6	64	6	0.996586
7	128	8	0.995620
8	256	12	0.999947
9	512	17	0.999448
10	1024	25	0.999461

Résultats de l'algorithme de Grover pour différentes valeurs de N.

Une formule permettant d'approcher le nombre d'itérations est $k = \frac{\pi}{4}\sqrt{N}$ avec N le nombre valeur dans le tableau. Pour comparer les performances de l'algorithme de Grover avec une version classique de recherche, il est important de rappeler que la recherche classique, dans le cas d'une base de données non triée, présente une complexité de $O(\frac{n}{2})$. Le graphique ci-dessous illustre cette différence de performance entre les deux approches



On retrouve dans ce tableau le nombre d'opérations nécessaires pour obtenir l'index d'une valeur en fonction du nombre de bits/qubits utilisés pour encoder les valeurs. On rappelle que le nombre de valeurs totales possibles à encoder est égal à 2^n avec n le nombre de bits/qubits.

2.2.4 Conclusion

En combinant l'effet de l'oracle et du diffuseur, l'algorithme de Grover exploite l'interférence pour amplifier les probabilités des états marqués. Cela permet une recherche quadratiquement plus rapide que les méthodes classiques, rendant l'algorithme particulièrement puissant pour des ensembles de données de grande taille.

2.3 Présentation des objectifs

Après cette introduction consistante au domaine du calcul quantique, nous allons explorer les travaux réalisés pour répondre au besoin de concevoir des méthodes permettant de vulgariser et de faire découvrir au plus grand nombre les principes de l'informatique quantique.

Dans un premier temps, nous présenterons le travail réalisé pour développer une application tactile de création musicale. Cette application vise à offrir une expérience immersive à l'utilisateur tout en lui permettant de comprendre certains phénomènes quantiques à travers une interaction ludique et intuitive.

Dans un second temps, nous examinerons les efforts menés pour générer des œuvres artistiques à l'aide de l'intelligence artificielle et de l'informatique quantique, mettant en lumière la synergie entre ces technologies pour produire des créations innovantes.

Chapitre 3

Musique Quantique

La création de musique de façon numérique est déjà une grande source de recherche en algorithmie classique. Au début, les ordinateurs ont été principalement utilisés afin de générer de la musique aléatoire et c'est ce que nous allons essayer de refaire. Pour cela, nous allons nous inspirer d'algorithmes présents dans un livre nommé "Quantum computer music foundation, methodes and advanced concept"[Ed22].

Avant de commencer, nous précisons que tous les algorithmes qui vont être présentés ont donc été implémentés et testés sur des simulateurs d'ordinateur quantique. Nous avons décidé d'utiliser pour cela la bibliothèque Python Qiskit[Qis20] créée par IBM. Ce choix a été fait, car IBM propose des services gratuits permettant, si on le souhaite, de faire tourner nos algorithmes sur de réels ordinateurs quantiques qu'ils mettent à disposition de tout le monde. Due aux longues files d'attente et les erreurs faites par ces ordinateurs, la majorité des algorithmes ont été testés sur simulateur uniquement.

3.1 Random Walk

Dans cette première approche[Edu21], nous utiliserons la capacité des ordinateurs quantiques à générer un véritable aléatoire pour concevoir un algorithme générant un suite de nombres aléatoires. En effet, sur les ordinateurs classiques, produire un aléatoire parfait est un défi important et complexe. Cela s'explique par le fait qu'il est impossible de générer un aléatoire complet : dans une situation où nous avons une connaissance totale des conditions initiales et des processus, il est toujours possible de prédire le résultat. Pour cette raison, l'aléatoire sur ordinateur classique est simulé. Bien que ces simulations soient souvent très performantes, elles ne parviennent jamais à reproduire complètement l'imprévisibilité d'un véritable aléatoire.

En revanche, en informatique quantique, le véritable aléatoire existe. Lorsque l'on mesure l'état d'un qubit, on peut connaître les probabilités qu'il prenne une certaine valeur (par exemple, 0 ou 1), mais le résultat final de la mesure est totalement imprévisible. Deux mesures réalisées dans des conditions strictement identiques peuvent aboutir à des résultats différents. Ceci est fondamentalement impossible à obtenir avec un algorithme classique. Cette propriété unique des systèmes quantiques constitue un atout majeur pour les applications nécessitant un aléatoire véritable.

Dans notre première approche, nous mettons en œuvre une chaîne de Markov dans laquelle chaque position correspond à une note. Les transitions entre ces notes sont effectuées par un circuit quantique, qui, à partir d'un état donné, évolue vers un autre état.

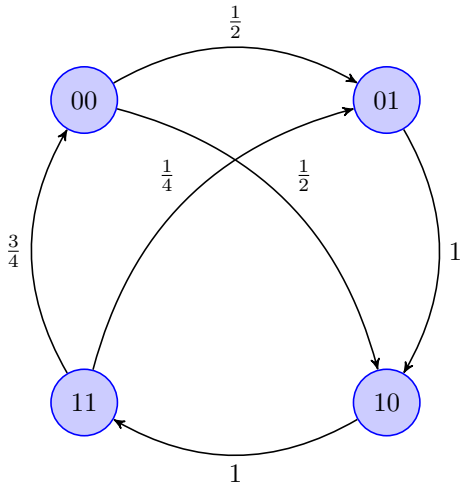
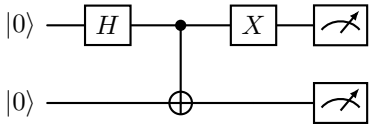


Figure 1 : Exemple de chaîne de Markov avec quatre états.

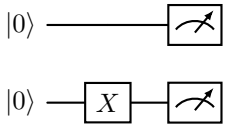
Penchons-nous plus sur le circuit. Tout d'abord, celui-ci n'est pas fixe, mais est défini en fonction de l'état sur lequel vous êtes. C'est-à-dire que nous n'avons pas affaire à un circuit unique, mais à un circuit qui change selon le dernier état. Cette méthode oblige l'utilisateur à utiliser un ordinateur classique et un ordinateur quantique. Par exemple, si vous êtes sur l'état 00, alors le circuit ressemblera à ça selon la chaîne de la figure 1 :



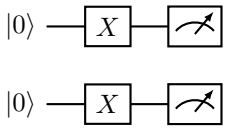
Ce circuit donnera donc la superposition des deux états possibles de sortie de probabilités équiprobables. Une mesure viendra décider sur quel état nous nous déplaçons.

$$|\Psi\rangle = \frac{1}{\sqrt{2}}(|10\rangle + |01\rangle) \tag{3.1}$$

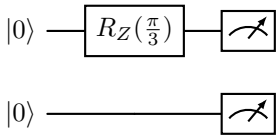
Ensuite, un nouveau circuit sera créé selon le nouvel état atteint. Les 4 circuits possibles pour notre exemple, sont donc déjà sauvegardés dans le programme et ne peuvent donc pas facilement être modifiés par l'utilisateur. En répétant suffisamment de fois l'algorithme, nous pouvons à terme obtenir une musique composée de notes choisies aléatoirement. On remarque tout de même que dans notre exemple, il n'y a pas de principe de durée des notes qui seront donc toutes de la même durée.



Circuit pour l'état 01



Circuit pour l'état 10



Circuit pour l'état 10

On peut imaginer ensuite la mise en place d'un système plus interactif où l'utilisateur pourrait entrer directement la probabilité de sortie depuis chaque état et ensuite créer le circuit automatiquement depuis cette chaine personnalisée grâce à des portes de rotation plus complexe.

3.2 Grover walk

Cette deuxième approche[Eua23] est assez semblable à la première car elle utilise aussi une chaîne de Markov, mais avec un objectif de la rendre plus interactive pour l'utilisateur en s'appuyant sur l'algorithme de Grover. En effet, ici, l'utilisateur pourra choisir les différentes notes qui peuvent s'enchaîner. Le désavantage de cette méthode est que l'on considère que l'utilisateur voudra systématiquement une équiprobabilité de chaque sortie possible. C'est-à-dire, si on considère que dans l'état 0 on peut aller à l'état 1 ou 2, il y aura 50% de chance d'aller sur chaque état.

Pour ce qui est du circuit, il sera donc encore une fois construit à chaque nouvel état. Il sera donc composé de n oracles qui viendront marquer les n états considérés comme étant possibles d'être joués après le dernier état atteint. Ensuite, le diffuseur viendra augmenter ces états de façon équiprobable

Après suffisamment de répétitions de l'algorithme (oracle et diffuseur), nous obtiendrons une grande probabilité que seules les sorties sélectionnées par l'utilisateur puissent être mesurées à la fin. Maintenant que nous avons mesuré la valeur, nous obtenons un nouvel état. Il suffit donc de récréer le nouveau circuit avec les nouveaux oracles et diffuseurs.

Note	**re**	**mi**	**fa**	**sol**	**la**	**si**	**do**	**do'**
re	1	0	0	0	0	0	0	0
mi	1	1	0	0	0	0	0	0
fa	0	1	1	0	0	0	0	0
sol	1	0	1	1	0	0	0	0
la	0	0	0	1	1	0	0	0
si	0	0	1	0	1	1	0	0
do	0	0	0	0	0	1	1	0
do'	0	0	0	0	0	0	0	1

exemple de tableau d'entrée

Note	**re**	**mi**	**fa**	**sol**	**la**	**si**	**do**	**do'**
re	1	0	0	0	0	0	0	0
mi	$\frac{1}{2}$	$\frac{1}{2}$	0	0	0	0	0	0
fa	0	$\frac{1}{2}$	$\frac{1}{2}$	0	0	0	0	0
sol	$\frac{1}{3}$	0	$\frac{1}{3}$	$\frac{1}{3}$	0	0	0	0
la	0	0	0	$\frac{1}{2}$	$\frac{1}{2}$	0	0	0
si	0	0	$\frac{1}{3}$	0	$\frac{1}{3}$	$\frac{1}{3}$	0	0
do	0	0	0	0	0	$\frac{1}{2}$	$\frac{1}{2}$	0
do'	0	0	0	0	0	0	0	1

tableau des probabilités associées de sortie de chaque note

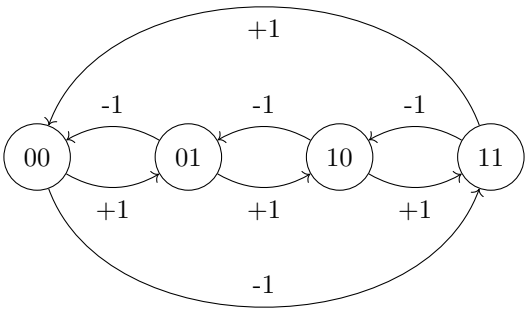
Chaque note est associée à une probabilité d'apparaître suite à une autre note, comme dans l'exemple précédent. Nous avons ainsi un algorithme facile d'utilisation où l'utilisateur choisit quelle note peut être jouée après une note donnée. Il faut tout de même rappeler que le tableau de probabilités n'est pas exact car l'algorithme de Grover ne retourne pas toujours la bonne valeur. Mais si l'on décide de faire tourner plusieurs fois le circuit et que l'on regarde l'état le plus souvent sorti de l'algorithme, on finit par atteindre une probabilité de sortie d'un état valide proche de 1. On remarque tout de même que si nous avons la moitié ou plus d'états possibles en sortie, l'algorithme de Grover ne nous permettra pas de les sélectionner, ce qui pourra apporter des erreurs.

3.3 Addition Walk

Dans cette méthode[Eua23], nous allons encore utiliser une chaine de Markov. Contrairement au deux autres solutions, celle-ci ne pourra pas être directement utilisée sur un ordinateur quantique car demande d'avoir accès

à l'amplitude des chacun des états à un moment donné. Comme les autres fois, nous avons donc une chaine de Markov où chaque état représente une note. Mais cette fois chaque état possède uniquement deux sorties qui sont les deux états adjacents.

L'objectif de cette méthode sera donc de se mouvoir sur cette chaine de Markov en se déplaçant simultanément sur les deux états adjacents à notre situation actuelle. La situation pourra être une situation simple au début, mais se complexifiera au fur et à mesure. Au début, nous commencerons avec une superposition d'un nombre donné d'états, mais petit à petit, nous serons avec une superposition de nombreux états. Ensuite, après chaque itération du circuit, nous allons faire un tirage de notes selon la probabilité de chaque état d'être atteint, ce qui donnera notre note.



Exemple de graphique a quatre état

Cette fois l'objectif est de ne jamais casser la superposition des états. Pour récupérer les informations, nous allons utiliser le Statevector qui est un vecteur décrivant l'amplitude et la phase de chacune des sorties possibles, mais ce vecteur n'est pas récupérable avec un ordinateur quantique. C'est pourquoi cet algorithme ne marchera que sur simulateur. Cette méthode a l'avantage de permettre une différente répartition des valeurs. En effet, là où un algorithme classique ne rendrait possible que les états adjacents à l'état suivant, ici, nous avons la possibilité qu'un bien plus grand ensemble de notes puisse être sélectionné. Le circuit utilise 5 qubits pour fonctionner. Les trois qubits de fin prennent les valeurs de note. Le deuxième sert à construire la superposition et le premier est un qubit utilisé par le circuit comme une mémoire. Le qubit de superposition vaudra 0 dans le cas d'une valeur qui vient d'être augmentée et 1 si diminuée.

$|0\rangle$ ——————— Mémoire

$|0\rangle$ ——————— Superposition

$|n_1\rangle$ ——————— Note 1

$|n_2\rangle$ ——————— Note 2

$|n_3\rangle$ ——————— Note 3

Le principe du circuit est que l'on vient depuis un état, créer une superposition des deux états adjacents. Pour cela, nous appliquons une porte H à un qubit qui permettra d'appliquer des portes de contrôle qui ajouteront ou diminueront de 1 selon l'état de ce qubit. Par exemple, si on avait entré $|101\rangle(5)$ qui s'initialise dans notre circuit par $|00101\rangle$, on obtient alors l'état :

$$|\Psi_1\rangle = \frac{1}{\sqrt{2}}|01100> + \frac{1}{\sqrt{2}}|00110> \tag{3.2}$$

Ensuite, on sauvegarde l'état du vecteur en utilisant le fait que nous soyons sur simulateur, ce qui ne détruit pas la superposition, mais nous permet de garder les informations concernant les états. Ensuite si nous souhaitons recréer une nouvelle note, nous repartons du même vecteur auquel nous réappliquons le circuit ce qui nous donne un nouveau vecteur d'état

$$|\Psi_2\rangle = \frac{1}{\sqrt{4}}\,|01011\rangle - \frac{1}{\sqrt{4}}|00101> + \frac{1}{\sqrt{4}}|01101> + \frac{1}{\sqrt{4}}|00111> \tag{3.3}$$

Un phénomène qui peut être observé est que, comme il y a un nombre pair de notes et qu'à chaque fois, nous obtenons les valeurs plus 1 et moins 1, cela alternera les notes ayant un indice pair et un indice impair. On

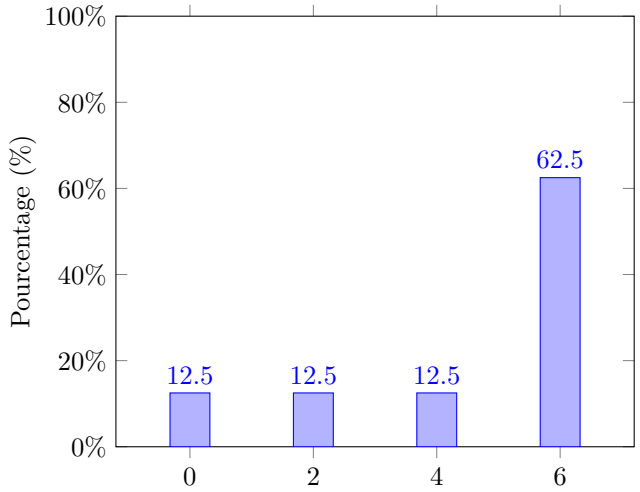
remarque aussi l'apparition de phases négatives, ce qui a un impact sur la distribution des notes dès la 3ème note. En effet, on obtient donc

$$\begin{aligned}\Psi_3 =& \frac{1}{\sqrt{8}}|01010> -\frac{1}{\sqrt{8}}|00100> -\frac{1}{\sqrt{8}}|01100> +\frac{1}{\sqrt{8}}|01100> \\ &-\frac{1}{\sqrt{8}}|00110> -\frac{1}{\sqrt{8}}|00110> +\frac{1}{\sqrt{8}}|01110> +\frac{1}{\sqrt{8}}|00000>\end{aligned} \tag{3.4}$$

ce qui se réécrit.

$$\Psi_3 >= \frac{1}{\sqrt{8}}|01010> -\frac{1}{\sqrt{8}}|00100> -\frac{1}{\sqrt{2}}|00110> +\frac{1}{\sqrt{8}}|01110> +\frac{1}{\sqrt{8}}|00000> \tag{3.5}$$

Ce qui donne la répartition de probabilité suivante :



Probabilité de chaque valeur d'être sélectionnée à la 3ème itération du circuit avec comme donnée d'entrée $|00101\rangle$

Pour analyser plus loin, avec plus d'itérations de l'algorithme, on remarque que le pic qui représente la probabilité de la note la plus susceptible d'être sélectionnée est cyclique et avance d'une position par itération.

Une des particularités de cet algorithme est aussi que nous n'initialisons pas les qubits à 0, mais on reprend l'état d'après l'initialisation précédente. Cela permet donc de commencer le programme avec des états plus compliqués qu'un état à une sortie, mais avec toutes sortes d'états notamment des états intriqués. Dans le cas d'un état intriqué, le résultat sera juste un fonctionnement normal sur chacun des états. Par exemple si on rentre $|\Psi_0\rangle = \frac{1}{\sqrt{2}}(|00000\rangle + |00111\rangle)$ on obtiendra $|\Psi_1\rangle = \frac{1}{\sqrt{4}}(|00000\rangle + |00001\rangle + |01110\rangle + |01111\rangle)$ on aura donc bien le +1 -1 fait sur chacune des valeurs. Bien sûr, les différents états d'entrée donneront donc des réparations de probabilité de sortie des différentes notes.

Une deuxième version du programme, qui cette fois peut être utilisée sur un véritable ordinateur quantique, a été réalisée. L'idée, cette fois, n'est pas de générer un arbre tout entier, mais de le réduire de temps en temps en mesurant. C'est-à-dire, pour chaque note, nous allons choisir aléatoirement un nombre de fois que nous voulons itérer le circuit avant de mesurer. Nous obtiendrons une note aléatoire parmi les notes possibles après la nème itération du circuit. Elle sera utilisée comme première note de la prochaine recherche de note.

Cette méthode permet de garder l'utilité du fait que ce soit quantique avec la superposition et rend le circuit utilisable sur un véritable ordinateur quantique car nous n'utilisons plus de statevector. Pour ce qui est du hasard, pour le moment nous utilisons un tirage classique, mais on peut imaginer l'utilisation d'une version quantique qui peut facilement être approchée par l'utilisation de l'algorithme de Grover.

3.4 One qubit note encoding

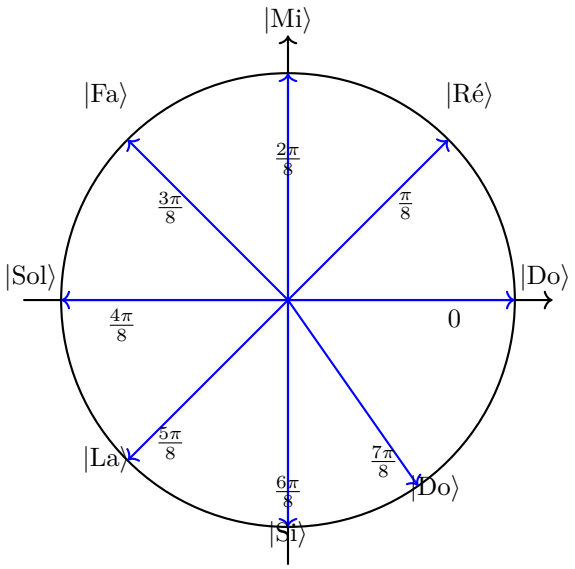
Dans cette partie, nous nous intéressons à un nouveau moyen d'encoder les notes[Sou22]. Jusqu'à maintenant, nous utilisions un système où on attribuait une valeur à chaque note. Par exemple la valeur 1 était un fa, 2 un sol... Ce qui impliquait que pour encoder 8 notes, il nous fallait obligatoirement 3 qubits minimum. Avec le nombre réduit de qubits utilisables actuellement, cela peut rapidement devenir une limitation importante.

Pour palier à cela, nous allons essayer de ne plus encoder les notes sur des valeurs mais sur la phase d'un seul qubits. Nous allons donc découper la phase totale d'un qubit en plusieurs angles auxquels nous attribuerons à

chacun une note. Par exemple, nous pouvons, pour encoder 8 notes attribuer une note à tous les angles de $\frac{\pi}{8}$. Cela pourrait donner do pour un angle de 0, ré pour un angle de $\frac{\pi}{8}$,mi pour $\frac{2\pi}{8}$...

Cette méthode permet de gagner un grand nombre de qubits nécessaires pour le stockage de nos notes. Cependant, cette méthode est compliquée à l'utilisation car nous n'avons pas de moyen, avec un véritable ordinateur quantique, de récupérer directement les valeurs des phases sans les réécrire avec une mesure. Cela implique encore une fois l'utilisation de nombreux qubits. Pour cette raison et les difficultés à l'utilisation, Cette méthode d'encodage ne sera pas utilisée telle quelle dans les autres algorithmes.

Nous ferons une exception dans l'algorithme du live musique que nous verrons après, car nous utilisons encore une fois le Statevector qui est une méthode pour récupérer les phases des qubits. Mais cette méthode ne peut pas être utilisée sur véritable ordinateur quantique.



Encodage des notes via la phase d'un qubit
Chaque note correspond à un angle spécifique de la phase

3.5 Modulation

L'objectif de cette section est d'ajouter une modulation à des notes, permettant à l'utilisateur de fournir une série de notes que l'algorithme modifie par l'application d'une modulation quantique. Plus précisément, nous utiliserons une méthode d'encodage de phase modifiée, comme vue précédemment [Sou22].

Le fonctionnement du programme est le suivant : l'utilisateur entre une séquence de notes qu'il souhaite faire moduler, ainsi que l'intensité de la modulation à appliquer. Ensuite, le circuit quantique est construit à partir de 6 qubits, répartis en 2 groupes de 3 qubits chacun, représentant chaque note de la séquence. Les notes sont jouées successivement dans l'ordre.

Tout d'abord, les qubits sont mis en superposition à l'aide de portes de Hadamard. Ensuite, chaque note est encodée sur les 3 premiers qubits en utilisant la phase. Pour ce faire, des portes de phase sont appliquées sur ces qubits, avec un angle défini par la formule :

$$\theta = \frac{\pi \cdot k}{2^i} \tag{3.6}$$

où k est la valeur à encoder et i représente l'indice du qubit sur lequel la porte est appliquée dans le groupe de trois qubits.

Une fois la note encodée, pour la décoder, on applique la transformation de Fourier inverse, QFT^{-1}, ce qui permet d'extraire la valeur encodée sur la phase. La note à jouer est ensuite récupérée par une lecture des qubits. Une fois la note mesurée, les qubits sont réinitialisés à zéro, permettant ainsi de préparer la note suivante.

Pour les notes suivantes, le processus est répété en appliquant la même logique aux 3 qubits restants. Cependant, à cette étape, nous introduisons la modulation, qui modifie les probabilités des notes qui peuvent être mesurées à la fin de l'application du QFT^{-1}. Cette modulation permet de "modifier" la note jouée, en augmentant la probabilité qu'une note modifiée apparaisse plutôt que la note d'entrée initiale.

3.5.1 Modulation quantique

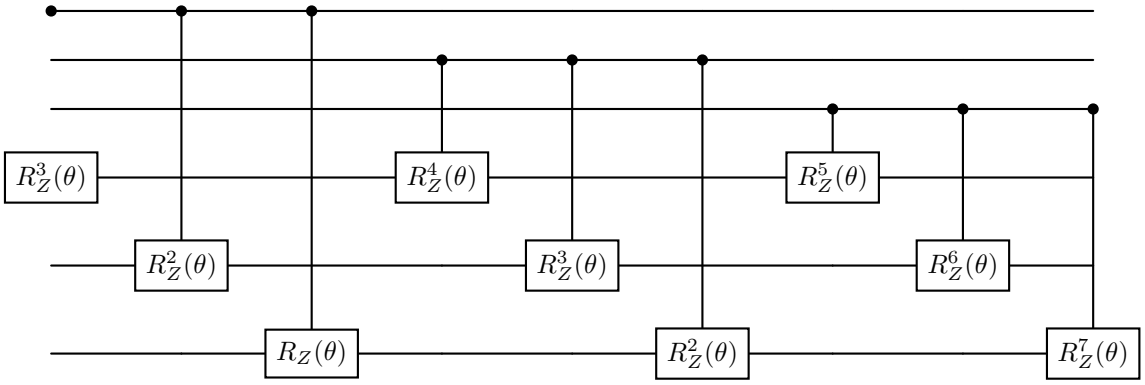
La modulation se fait de manière suivante : avant la mesure de la note précédente, et juste avant l'application du QFT^{-1} pour la note actuelle, un angle de modulation θ est appliqué. Cet angle est calculé différemment de

l'angle de phase utilisé pour l'encodage :

$$\theta = \frac{q \cdot \pi}{4} \tag{3.7}$$

où q est la puissance de modulation. Cette modulation modifie les amplitudes de probabilité des différentes issues, permettant ainsi d'influencer la sortie de la note quantique en augmentant la chance d'obtenir une note modifiée.

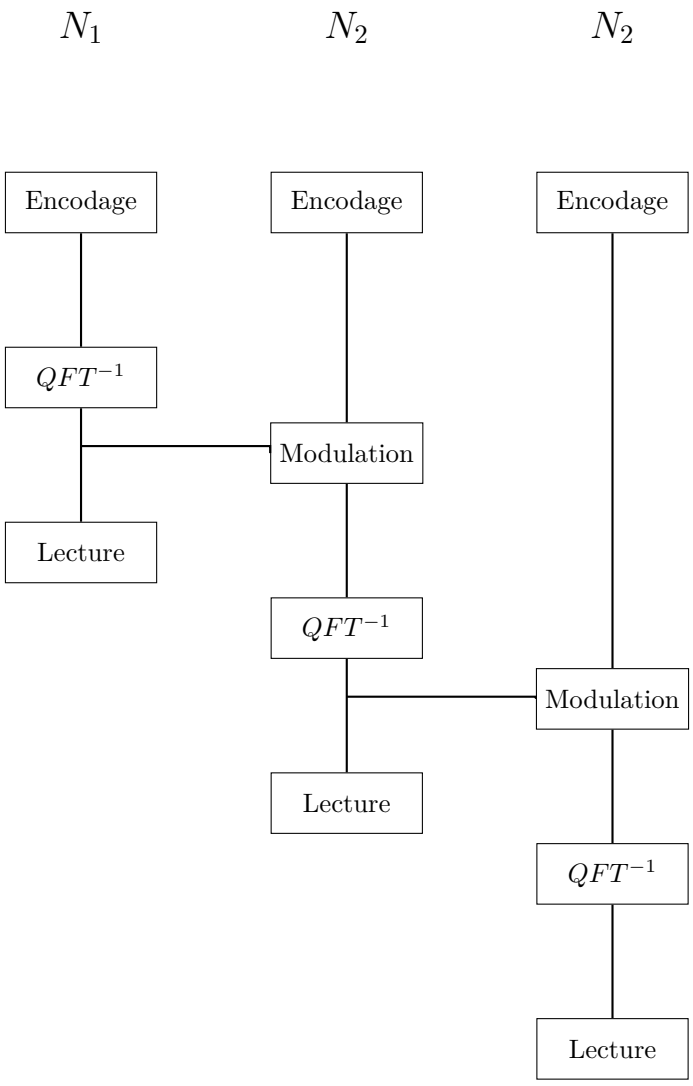
Le diagramme suivant illustre le circuit quantique mis en place. Les trois premiers qubits représentent la note précédente avant la lecture et la remise à zéro :



Le rôle des portes $R_Z(\theta)$ est d'introduire la modulation en ajustant la phase des qubits concernés. Cette modulation joue un rôle clé dans la modification de la probabilité des notes qui peuvent être lues à la fin du processus, et contribue à la variation de l'issue de la mesure.

3.5.2 Schéma du circuit

Voici un diagramme illustrant l'enchaînement des différentes étapes du processus de modulation et d'encodage :



3.6 Live musique

Dans cette partie[Sou22], l'objectif est de créer une application afin de répondre directement à l'objectif du sujet. C'est-à-dire de créer une application pour apprendre aux utilisateurs le fonctionnement du quantique et ses utilisations de façon ludique et visuelle.

Nous avons décidé de construire une application qui permettra aux utilisateurs de créer de la musique en direct en plaçant des portes quantiques sur un circuit. Dans cette modélisation, les notes sont considérées comme étant la phase d'un qubit. Cette méthode permet alors de jouer plusieurs notes simultanément si on utilise plusieurs qubits de note. Ensuite, l'utilisateur rajoute des portes quantiques, CNOT et Controle Phase afin de créer et de modifier différents états et alors de modifier les phases des différents qubits. Ensuite, une partie classique viendra calculer la phase des différents qubits grâce au Statevector qui est récupéré en profitant du fait que nous soyons sur un simulateur.

Sur ce circuit, nous avons aussi des qubits que nous appelons qubits de contrôle. Leur utilité est de permettre la création d'états plus compliqués et plus aléatoires, utiles pour des portes de contrôle. Pour pouvoir calculer véritablement leur impact sur le circuit s'il était vraiment fait et non juste simulé, nous créons un circuit parallèle qui sert à mesurer la valeur de ces qubits. Celui-ci est juste un circuit où nous appliquons une porte de Hadamard pour permettre la sortie de toutes les valeurs possibles. Une fois la phase des différents qubits calculée, il ne reste plus qu'à jouer les notes qui ont été sélectionnées et laisser l'utilisateur ajouter à nouveau des portes. La plus grande difficulté de cette méthode est de retrouver la phase des qubits depuis le Statevector ce qui peut avoir une certaine difficulté. C'est pour cela que nous avons décidé que l'utilisateur ne pourrait utiliser que les portes de phase. De cette façon, par exemple, si nous avons 2 qubits de notes et 2 qubits de contrôle, après tirage au sort des valeurs des qubits de contrôle (par exemple 11), nous nous retrouvons avec cette situation :

$$|\Psi\rangle = A\,|1100\rangle + B\,|1101\rangle + C\,|1110\rangle + D\,|1111\rangle \tag{3.8}$$

Par le STATEVECTOR, nous pouvons récupérer les valeurs des phases de cette situation. Mais comme nous

avons seulement utilisé des portes de phase et des portes CONTROLE NOT, nous pouvons écrire α et β les phases des qubits de notes de cette façon :

$$\begin{aligned} \lambda &= \frac{1}{A} \\ \frac{B}{\lambda} = \exp^{i\alpha} &\iff \alpha = \frac{\log \frac{B}{\lambda}}{i} \\ \frac{C}{\lambda} = \exp^{i\beta} &\iff \beta = \frac{\log \frac{C}{\lambda}}{i} \end{aligned} \tag{3.9}$$

Cette méthode permet de récupérer les phases des différents qubits sans trop de difficulté. Il nous suffit alors de faire la liaison entre la phase et une fréquence pour que l'utilisateur puisse changer les notes jouées en ajoutant des portes.

Maintenant que nous avons fini la partie logique de notre algorithme, nous avons décidé d'en faire une application qu'un utilisateur pourrait essayer. Afin de faciliter l'utilisation du code Python qui gère la simulation de la partie quantique, nous avons décidé d'utiliser un framework nommé FLASK qui permet de gérer le côté serveur de notre application avec du Python, ce qui permet ensuite une utilisation très simple du programme Python.

Chapitre 4

IA Quantique

De nos jours, quand on pense à la génération d'images ou de musiques, on pense rapidement à l'intelligence artificielle. En effet, actuellement, de nombreuses images sont générées par des IA et sont de très bonne qualité.

Dans l'objectif d'introduire l'intelligence artificielle, nous allons tous d'abord créer une IA capable de classifier des images, puis seulement nous utiliserons l'IA pour générer des images grâce à deux modèles différents. Bien sûr, comme nous avons l'objectif de montrer l'utilisation du quantique, tous les modèles vont d'abord être faits de manière classique. C'est-à-dire sans l'utilisation des principes de l'informatique quantique afin de comprendre leurs fonctionnements et tester leurs capacités. Ensuite, nous allons essayer de rendre au moins en partie ces IA quantiques.

4.1 IA de classification

4.1.1 Discrimination classique

Avant de générer des images, nous allons d'abord travailler sur un modèle de reconnaissance d'images qui aura pour but de reconnaître des éléments sur des images. Le modèle que nous allons créer, sera formé pour reconnaître les images de la base de données cifar10. Cette base de donnée est un ensemble d'images représentant divers objets ou animaux.

Sur les images, il y a un total de 10 différents éléments qui peuvent être représentés.Aucune image ne peut comporter une combinaison d'éléments. C'est-à-dire, qu'il n'y aura jamais un avion et un camion sur une même image comme ces objets sont tous deux des classes. De plus, ce sont des images d'assez basse résolution. Elles sont toutes de la taille de 32 pixels sur 32 pixels. Cette faible quantité de pixel rend donc les images plus difficiles à comprendre. Mais comme il y a moins d'informations à traiter, cela réduit la taille du modèle.

À l'heure actuelle, il n'existe pas de méthode permettant d'être sûr qu'un modèle réussira à accomplir sa tâche dans un temps d'entraînement donné et avec un nombre de neurones raisonnable. C'est pour cela qu'une grande partie de la mise en place de l'IA, a été de la recherche de modèles ayant déjà été testés et qui ont montré qu'ils pouvaient résoudre ce genre de problème.

Le design de modèle que nous avons repris est un design trouvé par une personne[KSH12] voulant résoudre, lui-aussi, le problème de la classification d'images.

Le modèle est constitué de plusieurs types de couches différentes. Tout d'abord, les couches de convolution[Vid] : ces couches sont des couches très importantes présententes dans de nombreux modèles de reconnaissance d'images. En effet ce sont des couches permettant de donner du contexte aux informations, par exemple, si un pixel est isolé ou pas. Ce contexte permet notamment de faire apparaître de manière chiffrée les contours des objets et forme qui sont présents sur l'image.

Ensuite les couches de pooling : ont pour objectif de réduire la taille du vecteur portant les informations mais en gardant le plus d'informations utiles possible. En effet, plus un modèle est petit, plus il est rapide à entraîner. Or les images sont de grandes entrées d'informations dont beaucoup qui ne seront pas utiles pour reconnaître ce qui est représenté.

Enfin les couches relues sont les plus classiques. Elles servent juste à faire la partie de raisonnement de notre modèle. Ce sont juste des couches où les neurones sont simplement reliés à tous ceux de la couche précédente. Les résultats que nous avons eus, sont plutôt bons avec une réussite d'environ 80 % sur la classification avec 10 classes et 95 % pour 2 classes. Bien sûr il est possible de trouver d'autres modèles avec une meilleure réussite. Mais l'objectif étant de comprendre le fonctionnement de l'intelligence artificielle, ces résultat nous conviendrons pour le moment.

Voici un exemple à trois classes de classification où Label représente la classe réelle de l'image et Pred la classe que notre IA pense voir représentée.

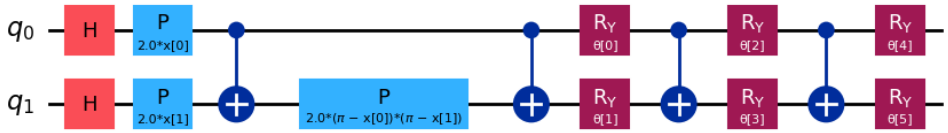
4.1.2 Discrimination quantique

Maintenant que nous avons réussi à créer un modelé sachant trier des images selon ce qui y est représenté de manière classique, nous pouvons essayer de le faire de manière quantique. La principale difficulté de le créer pour des IA quantiques, est le fait que les ordinateurs quantiques sont simulés. De ce fait, l'entraînement prendra beaucoup plus de temps. On peut considérer que, pour entraîner un modèle où nous utilisons 4 qubits, il faudra environ une journée. Ce temps est multiplié par deux pour chaque qubit supplémentaire.

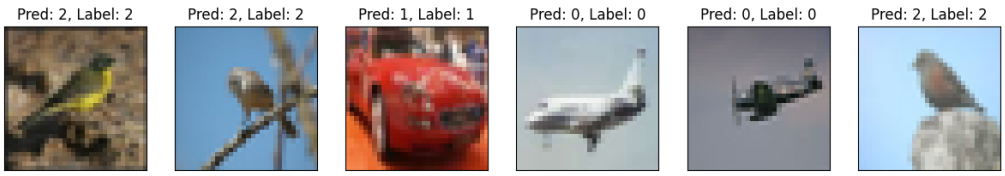
C'est pour cela que nous avons décidé de nous limiter à un nombre réduit de 4 qubits pour pouvoir tester plus facilement notre modèle. Cependant, avec les modèles quantiques, il faut comprendre que le nombre de qubits représente le nombre maximum de neurones que nous pouvons mettre sur une couche. De plus, il est difficile et assez inefficace de rentrer plus d'informations que nous avons de qubits. C'est pour cela que nous avons décidé de ne pas faire un modèle complètement quantique mais un modèle hybride avec des parties classiques et une partie quantique.

Ainsi, toutes les couches de convolution et de pooling, ainsi qu'une première couche de relu, sont classiques. Elles réduisent la taille du vecteur d'information à la taille 4. Ensuite, le modèle quantique vient appliquer une dernière couche relu, ce qui réduit à un vecteur de taille 2 pour le choix final.

Quant à leur fonctionnement, les couches quantiques sont utilisé de la même manière que celles classiques, à la seule différence près, que les poids des neurones et de leur liaison sont des angles sur des portes quantiques. Les résultats obtenus sont relativement similaires avec ceux du modèle classique.



Voici le circuit quantique représentant la partie quantique de l'IA. X est le tableau des valeurs d'entrée et θ le tableau des poids des paramètres.



Voici un exemple à trois classes de classification où Label représente la classe réelle de l'image et Pred la classe que notre IA pense y voir représentée.

4.1.3 XOR : évaluation de la contribution quantique au modèle

Dans le cadre de notre projet, nous avons cherché à comprendre en quoi la partie quantique du modèle apporte une valeur ajoutée, notamment en terme de réduction du nombre de paramètres nécessaires pour résoudre un

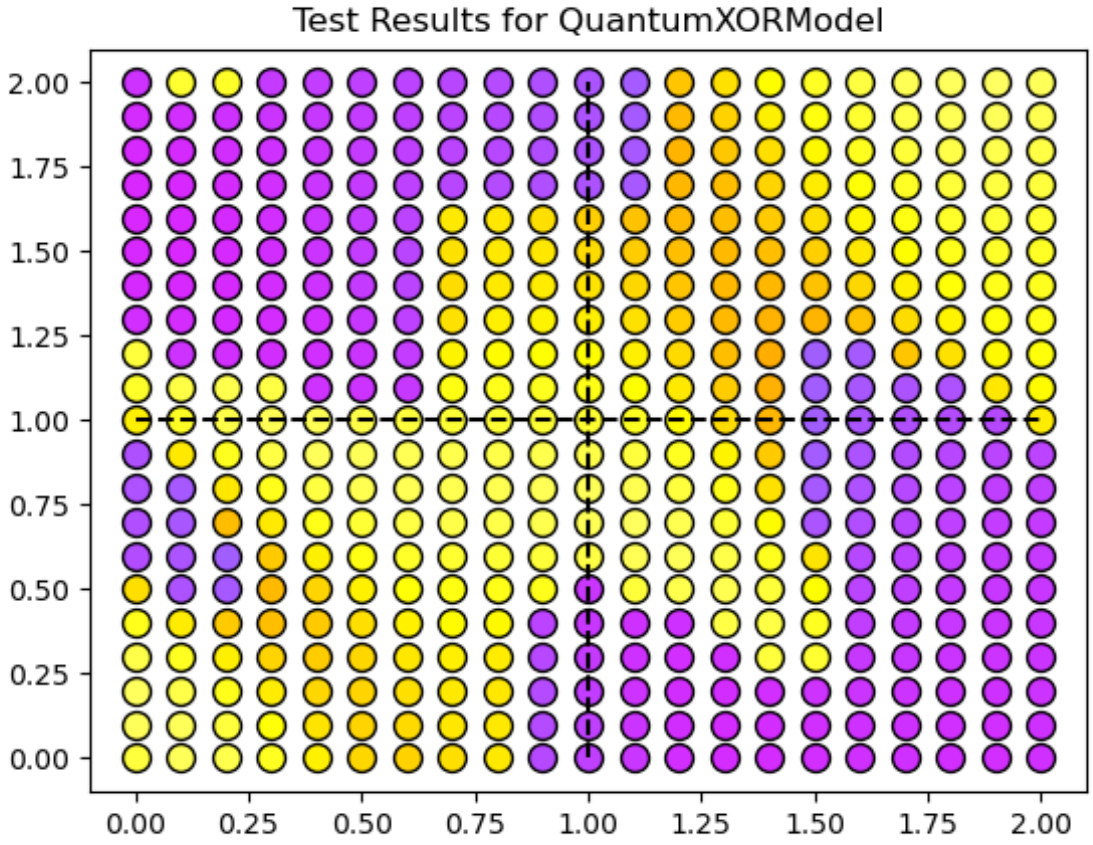
problème donné. Pour cela, nous avons choisi d'étudier un problème simple qui puisse être résolu par une IA complètement quantique : la simulation d'une porte XOR. C'est un bon cas d'étude pour évaluer la capacité d'un modèle à apprendre une fonction non linéaire.

Le problème XOR étudié consiste à définir une règle sur deux nombres réels x et y dans l'intervalle $[0, 2]$. Les règles sont les suivantes :

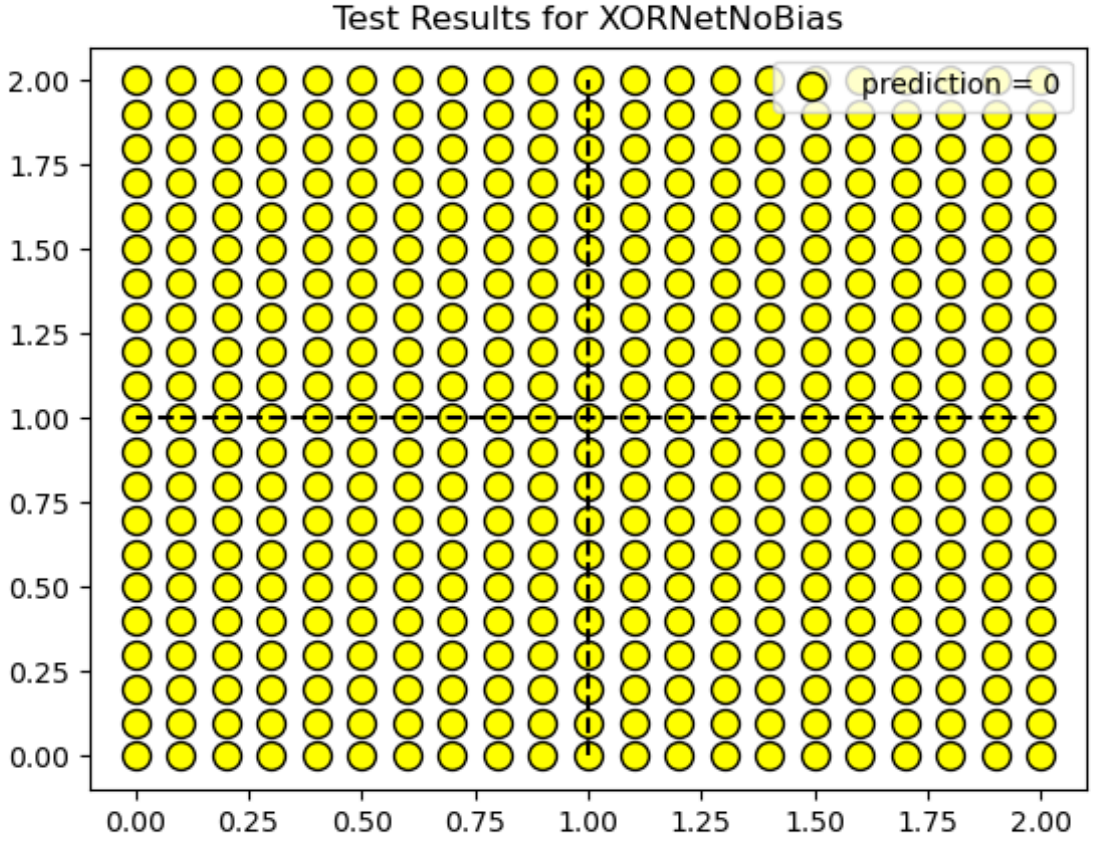
— Si $x < 1$ et $y < 1$, retourner 1 ;
— Si $x > 1$ et $y > 1$, retourner 1 ;
— Sinon, retourner 0.

Pour résoudre ce problème, j'ai conçu deux modèles d'intelligence artificielle, l'un classique et l'autre quantique. Chacun comporte seulement 6 paramètres. L'objectif était d'évaluer si la version quantique permettait de résoudre le problème avec moins de paramètres que le modèle classique.

Les résultats obtenus montrent que le modèle quantique atteint une précision modérée, suffisante pour résoudre en partie le problème, tandis que le modèle classique échoue complètement. Cela illustre la capacité des modèles quantiques à capturer des relations complexes avec un faible nombre de paramètres. Ceci constitue un avantage significatif dans ce contexte.



On voit ici le résultat pour l'IA quantique avec 6 paramètres qui atteint environ 75 % de réussite.



Voici les résultats pour l'IA classique. On remarque que l'IA n'arrive pas à différencier les deux situations. Elle considère que toutes les situations sont l'une des possibilités.

4.1.4 Feature maps

Dans les IA quantiques, on utilise au début de chaque modèle d'IA un petit algorithme qui s'appelle la feature map. Il permet d'encoder les données dans l'espace Hilbertien (qui est l'espace de représentation des qubits) pour qu'un algorithme quantique puisse les utiliser. Donc la features maps sert à encoder la donnée. Mais il existe plusieurs feature maps différentes pour différents encodages selon le besoin du problème.

Jusqu'à maintenant, nous utilisions la zzfeaturesmaps qui est l'une de celles utilisées le plus souvent. Cette features maps encode les valeurs sur les phases des qubits. Il faut donc autant de qubits qu'il y a de valeurs d'entrée. Dans le cas précédent du XOR, nous avions une valeur X et une valeur Y, il fallait donc au minimum deux qubits. Cette feature maps a pour avantage d'encoder donc de façon unique chaque entrée X et Y sur la phase de sortie. Pour un cas avec deux entrées X et Y encodées sur 2 qubits, cela donne :

$$|\Psi >= \frac{1}{\sqrt{4}}(|00 > +e^{2i(\pi-X)(\pi-Y)+2iY}\,|01\rangle + e^{2i(\pi-X)(\pi-Y)+2iY}\,|10\rangle + e^{2i(X+Y)} \tag{4.1}$$

Mais il existe toutes sortes de feature maps qui encodent différemment les données et qui peuvent avoir une plus ou moins grande importance selon les problèmes que notre IA doit résoudre.

4.1.5 Étude de la non-linéarité

Nous nous sommes posés la question sur ce qu'apportait la zzfeatures map plutôt qu'une autre features maps. Un des points important est la non-linéarité. En effet, beaucoup des problèmes qu'une IA doit apprendre à résoudre sont non linéaires. C'est-à-dire, par exemple, un problème avec deux entrées ne peut pas être approché par une droite.

En quantique, l'ensemble des opérations sont appliquées avec des portes quantiques qui sont, par définition, des opérations linéaires. Or il est démontrable qu'une suite d'opérations linéaires est aussi une opération linéaire. C'est-à-dire, si f et g des fonctions linéaires alors $f(g)$ est aussi une fonction linéaire.

Il est donc possible dans un circuit quantique de faire apparaître de la non-linéarité à seulement deux moments, l'encodage et le décodage. À l'encodage, la non-linéarité est faite par la features maps en encodant sur la phase. En effet, l'encodage pour la zzfeatures map se fait sur la phase, ce qui revient à appliquer une matrice $\begin{bmatrix} 1 & 0 \\ 0 & e^{iX} \end{bmatrix}$
Or la fonction exponentielle n'est pas linéaire. Cette méthode a donc bien permis d'encoder l'information mais de façon non-linéaire.

Cette recherche donne du sens à plusieurs résultats. En effet lors de la recherche sur la solution du problème XOR, nous avions remarqué qu'il y avait une solution classique à 6 paramètres. Si l'on considère X et Y nos entrées, A et B deux neurones reliés à X et Y et pour finir C un neurone relié à A et B, nous avions bien un total de 6 paramètres. Ensuite, on les définit

$$A = \frac{1}{3}X + \frac{2}{3}Y \tag{4.2}$$

$$B = \frac{2}{3}X + \frac{1}{3}Y \tag{4.3}$$

$$C = 3A - 3B \tag{4.4}$$

Table 4.1 – Valeurs de A, B et C pour différentes valeurs de X et Y

X	Y	A	B	C
0	0	0	0	0
0	1	$\frac{2}{3}$	$\frac{1}{3}$	1
1	0	$\frac{1}{3}$	$\frac{2}{3}$	-1
1	1	1	1	0

On voit alors que nous nous trouvons dans un problème de linéarité où il faudrait juste rajouter une fonction valeur absolue pour trouver le bon résultat mais qui est non linéaire. En effet, le problème XOR n'est pas linéaire. Il est donc impossible pour une fonction linéaire d'approximer totalement ce problème. On peut donc s'attendre à ce que toute IA avec une features map n'apportant pas de la non-linéarité soit donc incapable de résoudre le problème. En algorithmie classique, cette non-linéarité est apportée par les fonctions d'activation, mais qui ne peuvent malheureusement pas être répliquées identiquement simplement de façon quantique.

4.2 Génération

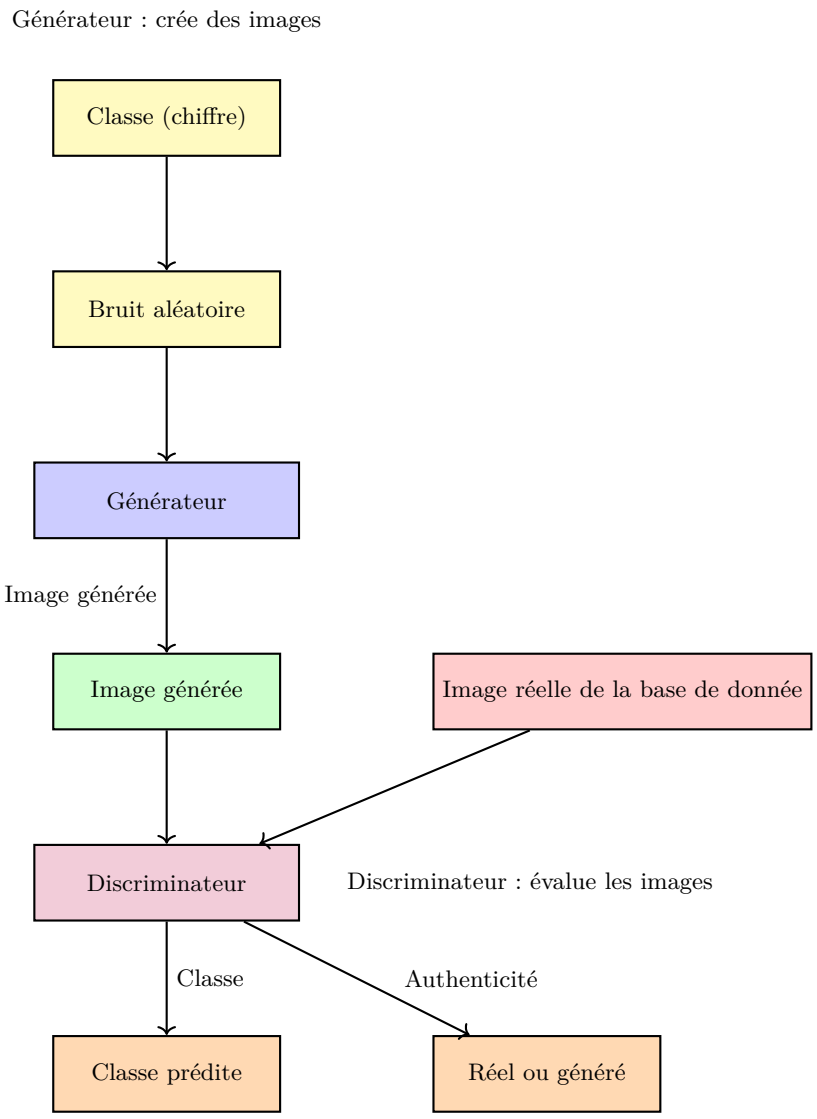
4.2.1 Génération classique

Maintenant que nous avons une IA de classification qui marche, il est temps de créer l'IA de génération. Pour cela, nous allons utiliser un CGAN (conditional generative adversarial networks) afin de pouvoir réutiliser le discriminateur créé précédemment. L'objectif de cette IA sera de générer des images ressemblantes à des images d'une base de données.

Le GAN est un modèle composé de deux IA indépendantes. La première IA prend comme entrée un vecteur de bruit. Elle va l'utiliser afin de créer une image. Comme nous voulons faire un CGAN, nous rajoutons aussi en entrée une valeur représentant ce que nous voulons voir représenté sur l'image. Cela constitue la part conditionnelle du modèle. Le bruit est là afin de créer de l'aléatoire afin que l'image générée ne soit pas toujours identique.

La deuxième IA est celle de discrimination créée plutôt. Elle prendra en entrée soit une image venant d'une base de données, soit une image générée par la première IA. Son rôle est toujours de comprendre ce qui est représenté sur l'image mais, cette fois aussi, de dire si l'image est générée ou venant de la base de données.

L'entraînement de ces IA est une mise en compétition. C'est-à-dire l'IA de génération créera des images avec pour objectif que ce qui y est représenté soit reconnu de la deuxième mais que l'image ne soit pas considérée comme étant artificielle. Celle de classification quant à elle sera entraînée à reconnaître ce qui est représenté sur l'image et aussi à reconnaître si l'image qu'elle a reçue en entrée a été générée ou vient de la base de donnée



Dans notre cas, on a décidé d'entraîner le CGAN à créer des images grâce à la base de données MNIST. C'est une base de données de chiffres écrits à la main en noir et blanc d'une taille de 28 pixels sur 28 pixels. La difficulté la plus fréquente des CGAN est le mode collapse. C'est une situation où le modèle ne fait plus preuve d'originalité et génère toujours les mêmes images car ne prend plus en compte le bruit en entrée.

Malheureusement nous n'avons pas réussi à sortir de ce problème. Malgré cela, nous avons réussi à faire générer des images différentes pour chaque classe. Mais celles pour chaque classe sont toujours identiques. Le problème semble venir du discriminateur qui est incapable de faire la différence entre les images générées et les images venant de la base de données.

4.2.2 Génération quantique

Ensuite afin de revenir un peu plus sur l'étude des algorithmes quantiques, nous avons décidé de rajouter des parties quantiques dans notre modèle malgré le problème du mode collapse encore présent. Si nous trouvions une méthode pour régler le problème en classique, il serait facile de l'appliquer sur le modèle quantique.

Tous d'abord, il est trop difficile avec les simulateurs actuels de simuler un modèle complètement quantique de grande taille comme on l'a vue pour l'IA de classification d'images que l'on a étudié plus tôt. C'est pour cela que nous avons décidé de viser des moments clés qui ne nécessitent que peu de qubits ou peu d'itérations.

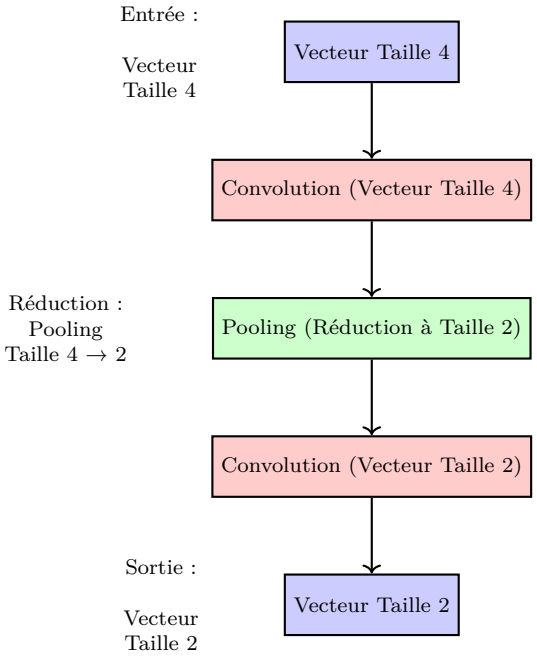
Tout d'abord nous avons commencé par la création des vecteurs de bruit nécessaires à la génération de nouvelles images. Nous avons, tout simplement, généré un vecteur de taille 100 constitué de 0 ou de 1. Pour cela nous avons mis en place un circuit qui applique une porte d'Hadamard sur 100 qubits, ce qui après mesure, nous donne donc des vecteurs de taille 100 générés de façon totalement aléatoire. Bien que la génération de vecteurs prenne beaucoup de qubits, il ne s'agit que d'un circuit identique calculé plusieurs fois. Cela ne demande pas de régénérer le circuit et fait gagner du temps.

En plus de cette génération de vecteurs, il est aussi possible de rajouter directement des couches quantiques à certains endroits du modèle. Mais ces couches sont extrêmement impactées par la lenteur de la simulation. C'est pour cela que nous pouvons juste rajouter quelques couches quantiques.

Pour cela nous avons décidé de rajouter ces couches à la fin du discriminateur. Mais contrairement à ce que l'on avait fait pour le premier discriminateur, cette fois, ces couches ne seront pas uniquement des couches connectées mais aussi des couches de réduction de taille de vecteurs. En effet, ces couches prendront en entrée un vecteur venant de la partie classique du modèle et de taille 4. Nous avons d'abord essayé avec un vecteur de taille 8. Mais le temps pris pour l'entraînement était trop long pour pouvoir être compatible avec nos moyens.

Ensuite, ce vecteur passe à travers une première couche qui sert de couche de convolution. Ces couches ont pour objectif de donner du contexte à une information. Cela permet, notamment dans la classification d'images, de permettre le repérage du contour des objets.

Ensuite le vecteur toujours de taille 4 passera dans une couche de pooling qui permet de réduire la taille du vecteur, dans notre cas, par 2 en essayant de perdre le moins d'informations possible. Après ces deux premières couches, nous nous retrouvons donc avec un vecteur de taille 2 que nous passons une dernière fois dans une couche de convolution.



Les résultats sont similaires aux résultats de la version classique à part le fait que le modèle est plus lent à entraîner à cause de la simulation de la partie quantique, chaque qubit rajouté multiplie environ par 2 le temps nécessaire.

4.3 VAE

Après plusieurs semaines à rester bloqué sur le problème de mode collapse du GAN, nous avons décidé d'essayer un autre type de modèle d'IA afin de générer des images : le VAE (en français auto-encodeur variationnel).

4.3.1 VAE classique

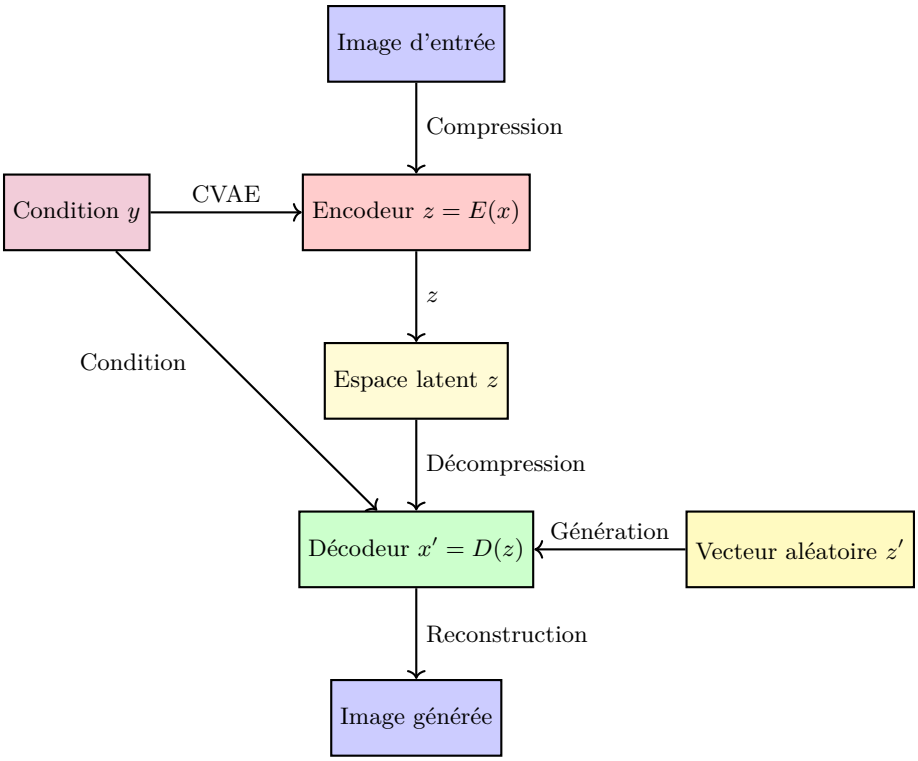
Le principe de ce modèle est d'apprendre à une IA à compresser une image en un vecteur de petite taille et le décompresser pour retrouver l'image d'origine. On peut donc séparer le modèle en un encodeur et un décodeur qui fonctionnent ensemble pour l'entraînement. Mais seul le décodeur est utilisé pour la génération de nouvelles images. Dans ce dernier cas, il suffit donc récréer de nouveaux vecteurs aléatoires puis les passer dans le décodeur pour créer de nouvelles images.

Les résultats sont bons, c'est-à-dire les images représentent bien des chiffres écrits à la main comme on le souhaitait et qui de plus ne sont pas toujours identiques. Au niveau plus complexe, il faut comprendre que le modèle prend les différentes images et les fait correspondre à des coordonnées dans un espace. Quand on prend un nouveau vecteur aléatoire, le décodeur nous retourne une image ressemblant aux images aux alentours.

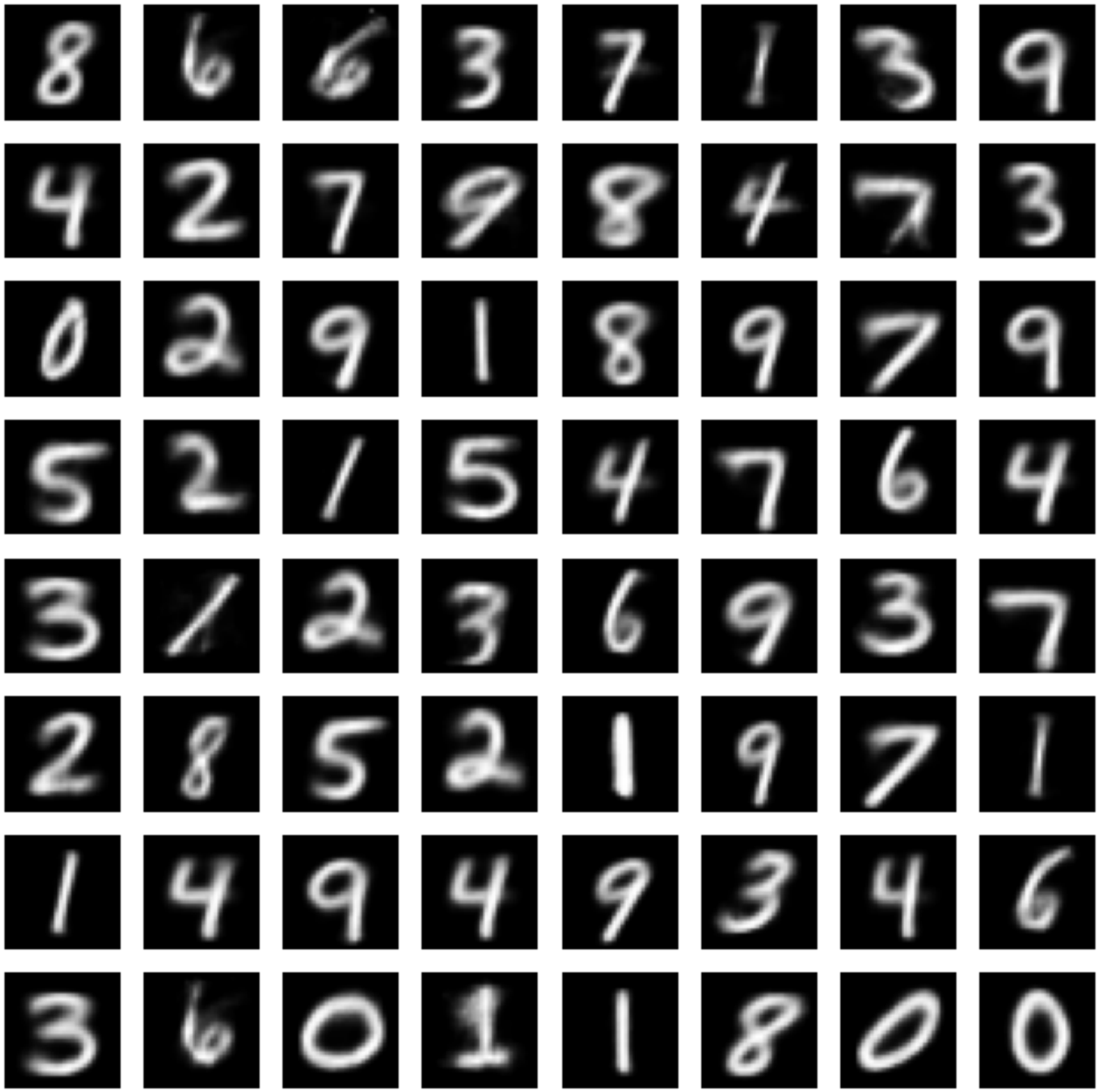
On remarque, tout de même, un problème. Les images générées ont souvent l'air d'être un peu floues et donc de moins bonne qualité que celles générées par le GAN.

Ensuite, nous pouvons rajouter une façon de choisir la valeur de l'image que nous voulons générer. Par exemple que des 8 ou autre. Cet ajout de conditions va donc faire de notre modèle un CVAE (pour conditionnel VAE).

VAE : Modèle de compression et reconstruction



CVAE : Génération conditionnelle d'images



Voici le résultat de plusieurs chiffres générés grâce à notre VAE.

4.3.2 VAE quantique

Maintenant, nous allons voir de quelle manière, nous allons rajouter du quantique dans le VAE que nous avons précédemment créé. Comme pour le GAN, nous avons eu, encore une fois, des difficultés liées à la simulation et le temps pris par l'entraînement. Pour pallier à cela, nous avons encore une fois créé un modèle hybride. L'avantage, c'est que cette fois, le modèle fonctionne sur la diminution de la taille du vecteur d'informations que nous réduisons à une taille de 2. Il est donc facile d'ajouter deux couches quantiques similaires à ce que nous avons fait dans le discriminateur du GAN à la fin de l'encodeur. Il est aussi possible d'en rajouter une au début du décodeur construit à l'envers de la partie de l'encodeur. Mais dans la situation d'un CVAE, comme la condition de génération (la valeur que l'on veut générer) est rajoutée en plus du vecteur compressé, nous obtenons un vecteur trop grand pour le nombre de qubits que nous pouvons simuler.

Chapitre 5

Projet présenté

Nous allons maintenant retourner sur divers sujets que nous avons étudiés au début afin de les complémenter avec l'expérience que nous avons accumulée sur les divers domaines explorés. Cela nous permettra de rendre les résultats présentables pour répondre au besoin du projet.

5.1 Musique quantique GHZ et W

Maintenant que nous avons réussi à comprendre les bases du fonctionnement de l'intelligence artificielle, nous souhaitons retourner dans le domaine de la musique pour tenter de créer un nouveau type de musique, cette fois liée à un phénomène de la physique quantique.

5.1.1 Différentes intrications à 3 qubits

Dans l'introduction, nous avons abordé le phénomène de l'intrication quantique. Ce phénomène permet de lier des qubits de manière à ce que la lecture de l'état d'un qubit influence instantanément l'état de l'autre, indépendamment de la distance qui les sépare. Par exemple, dans l'état

$$\Psi = \frac{1}{\sqrt{2}} \left(|00\rangle + |11\rangle \right),$$

la mesure de l'un des qubits nous donne non seulement une information sur ce qubit, mais aussi sur l'état de l'autre qubit.

Nous avons également examiné un cas à trois qubits, où plusieurs types d'intrication peuvent exister. Nous allons nous concentrer sur deux d'entre eux :

— L'état GHZ, qui est donné par

$$\Psi = \frac{1}{\sqrt{2}} \left(|000\rangle + |111\rangle \right),$$

— L'état W, qui est représenté par

$$\Psi = \frac{1}{\sqrt{3}} \left(|100\rangle + |010\rangle + |001\rangle \right).$$

Ces deux états sont considérés comme des formes distinctes d'intrication quantique, bien qu'ils soient tous deux à trois qubits.

Enfin, si nous appliquons des portes U (des portes de rotation sur chaque axe) à chaque qubit, le type d'intrication du système reste inchangé. En effet, l'application de ces portes ne modifie pas la structure de l'intrication, c'est-à-dire que le système conservera toujours le même type d'intrication, qu'il soit GHZ ou W.

5.1.2 Génération de musique

Il est maintenant temps d'utiliser ces phénomènes pour générer de la musique. Pour cela, nous allons réutiliser l'algorithme de l'addition WALK, qui a la particularité de fonctionner avec des états d'entrée que nous pouvons choisir. Nous allons appliquer cet algorithme en utilisant comme états d'entrée soit un état GHZ, soit un état W. L'algorithme générera alors une séquence musicale.

Une fois qu'un grand nombre de morceaux auront été produits, nous pourrons analyser l'impact de l'état d'entrée sur la musique générée. Pour cela, nous avons décidé de créer une IA qui analysera les musiques produites et tentera de les classifier, afin de déterminer si elles proviennent de l'état W ou de l'état GHZ.

En ce qui concerne les résultats, il est important de noter que le modèle de couches utilisé pour l'IA n'est probablement pas optimal, mais nous avons néanmoins obtenu des résultats intéressants. Nous avons généré 10 000 morceaux de musique, répartis en deux groupes : un groupe de 20 notes et un autre de 100 notes.

Pour le groupe de 100 notes, l'IA réussissait à identifier à quel groupe appartenait la musique dans 75% des cas, tandis que pour le groupe de 20 notes, elle y parvenait seulement dans 63% des cas. Bien que ces résultats puissent être améliorés, ils montrent néanmoins qu'il existe une différence perceptible - même si légère - pour les musiques courtes entre les musiques générées, différence qui peut être détectée par l'IA.

5.2 Site web

Nous avons déjà évoqué un site créé pour permettre aux utilisateurs d'utiliser le programme de Live Music. Nous avons décidé d'y rajouter plusieurs créations afin de montrer les résultats de nos recherches. Tout d'abord, avec les images que nous avons générées grâce au VAE quantique que nous avons généré, nous avons créé une œuvre artistique. Cette œuvre représente une superposition des états 0 et 1 afin de représenter la superposition quantique. Pour finir, nous avons rajouté sur ce site web des exemples de musique de type GHZ et W que nous avons générés dans la section précédente. L'objectif est de mettre en avant les résultats obtenus par nos IA quantiques ainsi que par nos algorithmes afin de démontrer leur potentiel. Enfin, en regroupant l'ensemble de ces réalisations, nous prévoyons de publier cette page sur le site de QuantEdu afin de promouvoir l'informatique quantique.

Chapitre 6

Conclusion

6.1 Bilan du projet

Bien que l'on ne puisse jamais considérer une réelle fin dans un projet de recherche, de nombreuses pointes ont pu être mis en place durant cette première expérience dans ce domaine.

Tout d'abord, la recherche et l'état de l'art sur différents algorithmes de musique a été menée de façon constructive afin de comprendre ce qui est possible aujourd'hui avec nos moyens actuels. Cette approche a permis de créer une application fonctionnelle permettant à des néophytes dans les concepts de l'algorithmie quantique de comprendre son fonctionnement de façon visuelle.

De plus, la mise en place des IA permettant de générer des images a aussi été une réussite. Nous pourrons encore les améliorer afin d'avoir des meilleurs résultats notamment en ce qui concerne le GAN où nous n'avons pas réussi à supprimer les problèmes du mode collapse. Enfin, la plupart des algorithmes créés n'ont pas pu être testés sur véritable ordinateur quantique du fait des limitations à leur accès.

6.2 Bilan de l'expérience

6.2.1 Autonomie et gestion du projet

Une des dimensions marquantes de ce stage a été le degré d'autonomie dont j'ai bénéficié. Étant l'unique responsable de ce projet, j'ai eu l'opportunité d'assumer la gestion complète des différentes étapes. Dès le début, aucun plan détaillé n'était préétabli, ce qui m'a laissé une grande liberté pour définir la direction à suivre en fonction des besoins et des objectifs identifiés.

Cette autonomie m'a permis de renforcer mes compétences en gestion de projet, en structurant mon travail de manière indépendante et en trouvant des solutions face aux défis rencontrés. Bien sûr, Monsieur Holweck, a joué un rôle essentiel en m'apportant des éclairages précieux sur les concepts ainsi que des pistes afin de mieux explorer et comprendre le sujet. J'ai également pu bénéficier des conseils et des échanges avec mes collègues de bureau, ce qui a enrichi mon expérience.

6.2.2 Recherche et développement sur des sujets peu documentés

Une des grandes difficultés de ce stage résidait dans le fait que la plupart des sujets sur lesquels je travaillais étaient très peu documentés. Que ce soit dans le domaine de la génération quantique de musique ou de l'intelligence artificielle quantique, les informations disponibles étaient limitées. Cette situation a souvent rendu difficile la validation de mes approches avant de les mettre en œuvre. En effet, une grande partie de mon travail a été réalisée dans un climat d'incertitudes, où je n'étais jamais sûr des résultats finaux.

Cela m'a contraint à expérimenter de manière pragmatique, en m'appuyant sur mes capacités d'analyse et de raisonnement pour avancer malgré le manque de documentation. L'absence de références claires m'a poussé à développer une forte capacité d'adaptation, à affiner ma méthodologie et à ajuster mes hypothèses en cours de route. Ce fut une expérience d'apprentissage extrêmement précieuse, me permettant de devenir plus autonome dans ma réflexion et dans la gestion de projets de recherche.

6.2.3 Applications et implications professionnelles

L'une des étapes marquantes a été la mise en application concrète des connaissances acquises. J'ai été amené à comprendre et à implémenter des solutions adaptées aux enjeux pratiques du projet. Cette phase m'a permis de découvrir l'importance de l'application des théories dans un contexte réel et d'intégrer les défis techniques dans un cadre plus large. Elle m'a également offert un aperçu du rôle d'un ingénieur, qui doit non seulement maîtriser

des compétences techniques diverses, mais aussi jongler avec des tâches variées, en apprenant constamment de nouvelles choses pour résoudre des problèmes complexes.

6.3 Perspectives pour l'avenir

Ce stage m'a permis de découvrir le monde de la recherche, une expérience qui m'a beaucoup plu. J'ai apprécié le côté autonome, où il faut chercher des solutions par soi-même, tout en explorant un domaine complexe avec beaucoup de potentiel. J'ai aussi aimé voir ma progression au fil du temps, en partant avec peu de connaissances sur le sujet pour finir par mieux le maîtriser et mieux comprendre les enjeux liés à ce secteur.

L'un des aspects intéressants de ce stage a été de travailler de manière individuelle sur mon projet. Cela m'a permis de développer une autonomie importante et de renforcer ma capacité à résoudre des problèmes par moi-même. Bien que le travail en équipe ait été moins présent, cette expérience m'a permis de mieux comprendre l'importance de la collaboration dans un environnement de recherche. Je vois cela comme une opportunité de développer cette compétence dans mes futures expériences professionnelles.

Ce stage reste une expérience extrêmement enrichissante. J'ai appris beaucoup, tant sur le plan technique que personnel, et je repars avec un réel sentiment d'accomplissement. Cette expérience m'a aussi confirmé mon envie de poursuivre dans ce domaine, en explorant à l'avenir des rôles où les échanges et la collaboration avec d'autres chercheurs seraient plus fréquents.

En résumé, ce stage m'a permis de me tester et de mieux comprendre ce que je recherche dans mes futures expériences professionnelles : ayant, avant même de commencer ce stage, envie de faire une thèse dans les technologies innovantes, je suis content que ce stage m'ait permis de confirmer mon envie de partir dans cette voie.

Bibliographie

[Bos] H. J. M. Bosmans. ISQCMC 2021. https://hjmb.co.uk/media/ISQCMC.2021.pdf. Article de conférence.

[Cas] P. Castro. PyTorch GAN. https://github.com/prcastro/pytorch-gan. Dépôt GitHub.

[Don] Chris Donahue. LakhNES. https://github.com/chrisdonahue/LakhNES. Dépôt GitHub.

[DRA] DRA-chaos. Quantum-Classical Hybrid Neural Network. https://github.com/DRA-chaos/Quantum-Classical-Hyrid-Neural-Network-for-binary-image-classification-using-PyTorch-Qiskit-pipeline. Dépôt GitHub.

[Ed22] Eduardo Reck Miranda (Ed.). Quantum Computer Music : Foundations, Methods and Advanced Concepts. Chapitres 3, 4, 5. Springer, 2022.

[Edu21] Suchitra T. Basak Eduardo Reck Miranda. "Quantum Computer Music : Foundations and Initial Experiments". In : *Quantum* (2021).

[Eua23] Simon D. Small Euan J. Allen Jacob F. F. Bulmer. "Making Music Using Two Quantum Algorithms". In : *arXiv* (2023).

[IBMa] IBM. IBM Quantum. https://quantum.ibm.com/. Plateforme en ligne.

[IBMb] IBM. Qiskit PauliFeatureMap. https://docs.quantum.ibm.com/api/qiskit/qiskit.circuit.library.PauliFeatureMap. Documentation.

[inc] Auteur inconnu. Performance of Quantum Kernel on Initial Learning. Fichier local : file :///C :/Users/estou/Down
Document PDF.

[KSH12] Alex Krizhevsky, Ilya Sutskever et Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In : *Advances in Neural Information Processing Systems*. T. 25. Curran Associates, Inc., 2012.

[Lab] V7 Labs. Neural Networks Activation Functions. https://www.v7labs.com/blog/neural-networks-activation-functions. Article de blog.

[Lye] Lyeoni. PyTorch MNIST GAN. https://github.com/lyeoni/pytorch-mnist-GAN. Dépôt GitHub.

[Maf] Mafda. Generative Adversarial Networks 101. https://github.com/mafda/generative_adversarial_networks_101. Dépôt GitHub.

[Mir05] Eduardo Reck Miranda. "Quantum Computing and Music". In : *arXiv* (2005).

[Mir20] Eduardo Reck Miranda. "Quantum Computer Music : Foundations and Initial Experiments". In : *arXiv* (2020).

[Osha] Scott Oshiro. Profil GitHub. https://github.com/scottoshiro2. Profil GitHub.

[Oshb] Scott Oshiro. QuiKo v0.1. https://github.com/scottoshiro2/QuiKo_v0.1. Dépôt GitHub.

[Pen] PennyLane. Quantum Feature Map. https://pennylane.ai/qml/glossary/quantum_feature_map/. Glossaire en ligne.

[Qis20] Qiskit Development Team. Qiskit : An Open-source Framework for Quantum Computing. https://qiskit.org. Accessed : 10 February 2025. 2020.

[Res] IBM Research. Quantum Computing Explained. https://www.youtube.com/watch?v=lnr8vUBQd8w. Vidéo YouTube.

[Sha] M. Shahbazi. Transitional cGAN. https://github.com/mshahbazi72/transitional-cGAN. Dépôt GitHub.

[Sou22] Satofumi Souma. "Exploring the Application of Gate-Type Quantum Computational Algorithm for Music Creation and Performance". In : *Quantum Computer Music : Foundations, Methods and Advanced Concepts*. Sous la dir. d'Eduardo R. Miranda. Springer, 2022. isbn : 9783031139086.

[Vid] Analytics Vidhya. Everything You Need to Know About CNNs. https://medium.com/analytics-vidhya/everything-you-need-to-know-about-convolutional-neural-networks-cnns-3a82f7aa29c5. Article de blog.

Lien des différents codes https ://github.com/estouan/ST40

GACHELIN Estouan
Rapport de stage ST40 - A2024

Développement d'algorithme quantique pour la création musicale et la génération d'images

Ce rapport présente le développement d'algorithmes quantiques appliqués à la création musicale et à la génération d'images dans le cadre de mon stage de ST40 à l'UTBM. Il s'inscrit dans le cadre d'un projet visant à démocratiser les technologies quantiques à travers des approches artistiques et interactives.
Dans un premier temps, les bases de l'informatique quantique sont introduites, notamment le concept de qubits, de portes quantiques et les algorithmes fondamentaux comme celui de Grover. L'étude se poursuit avec la conception et la mise en œuvre de plusieurs méthodes de génération musicale exploitant les principes quantiques, tels que la Random Walk quantique, l'utilisation de l'algorithme de Grover et d'autres modèles de génération musicale fondés sur la superposition et l'intrication.
Une seconde partie du projet explore l'utilisation de l'intelligence artificielle quantique pour la génération et la classification d'images. Un modèle hybride combinant réseaux de neurones classiques et circuits quantiques a été testé pour des tâches de classification, illustrant les avantages des approches quantiques en termes de réduction de paramètres et d'efficacité pour certains problèmes spécifiques. L'étude se termine par une tentative d'application des méthodes quantiques aux réseaux antagonistes génératifs (GANs) afin de produire des images à partir de données d'entraînement.

Université de Belfort-Montbéliard
90010 BELFORT cedex
www.utbm.fr