دانشگاه صنعتی امیرکبیر

(پلی‌تکنیک تهران)

دانشکده مهندسی کامپیوتر

پروژه کارشناسی

تولید موسیقی با محاسبات کوانتومی

نگارش:

عرفان عابدی

استاد راهنما:

دکتر مرتضی صاحب‌الزمانی

مهر ۱۴۰۰


بسم الله الرحمن الرحيم

صفحه فرم ارزیابی و تصویب پایان نامه_ فرم تأیید اعضاء کمیته دفاع

به نام خدا

دانشگاه صنعتی امیرکبیر
(پلی‌تکنیک تهران)

تعهدنامه اصالت اثر

تاریخ: مهر ۱۴۰۰

این‌جانب عرفان عابدی متعهد می‌شوم که مطالب مندرج در این پایان‌نامه حاصل کار پژوهشی این‌جانب، تحت نظارت و راهنمایی اساتید دانشگاه صنعتی امیرکبیر بوده و به دست‌آوردهای دیگران که در این پژوهش از آن‌ها استفاده شده است، مطابق مقررات و روال متعارف ارجاع و در فهرست منابع و مآخذ ذکر گردیده است. این پایان‌نامه قبلاً برای احراز هیچ مدرک هم‌سطح یا بالاتر ارائه نگردیده است.

در صورت اثبات تخلف در هر زمان، مدرک تحصیلی صادر شده توسط دانشگاه از درجه اعتبار ساقط بوده و دانشگاه حق پی‌گیری قانونی خواهد داشت.

کلیه نتایج و حقوق حاصل از این پایان‌نامه متعلق به دانشگاه صنعتی امیرکبیر می‌باشد. هرگونه استفاده از نتایج علمی و عملی، واگذاری اطلاعات به دیگران یا چاپ و تکثیر، نسخه‌برداری، ترجمه و اقتباس از این پایان نامه بدون موافقت کتبی دانشگاه صنعتی امیرکبیر ممنوع است. نقل مطالب با ذکر منبع بلامانع است.

عرفان عابدی

امضا

سپاس‌گزاری

این‌جانب، عرفان عابدی، از استاد راهنمای خود، جناب آقای دکتر مرتضی صاحب‌الزمانی به خاطر تمامی کمک‌ها و راهنمایی‌های سازنده‌شان در مسیر انجام این پروژه و نوشتن این گزارش سپاس‌گزارم.

عرفان عابدی
مهر ۱۴۰۰

چکیده

امروزه، با نزدیک‌شدن تکنولوژی‌های ساخت سخت‌افزار به محدودیت‌های فیزیکی قانون مور و پیش‌رفت روزافزون تکنولوژی‌های فوق سرد توجه بسیاری به دانشمندان به حوزه‌ی محاسبات کوانتومی معطوف شده است. کامپیوترهایی که در حال حاضر به صورت روزمره در حال استفاده هستند، در انجام محاسبات خود از قوانین فیزیک کلاسیک پیروی می‌کنند و به کامپیوترهای کلاسیک معروف هستند. محاسباتی که توسط کامپیوترهای کوانتومی انجام می‌شوند، بر خلاف کامپیوترهای کلاسیک، تابع قوانین فیزیک کوانتومی هستند؛ به همین علت، بسیاری بر این باور هستند که این گونه کامپیوترها در آینده‌ای نزدیک، قادر به انجام محاسباتی خواهند بود که به سادگی توسط کامپیوترهای کلاسیک ممکن نیست. یادگیری ماشین کوانتومی به گروهی از الگوریتم‌های کوانتومی اطلاق می‌شود که همانند یادگیری ماشین کلاسیک، تعدادی پارامتر قابل تنظیم دارند که بهینه‌سازی این پارامترها برای رسیدن به خروجی مطلوب، توسط یک کامپیوتر کلاسیک انجام می‌گیرد. در عین حال، از همان روزهای اولیه‌ی پیدایش کامپیوترها، بسیاری به دنبال تولید ملودی‌های موسیقی با استفاده از قدرت پردازشی کامپیوترها بوده‌اند و تاکنون، الگوریتم‌های متعددی برای حصول این امر پیشنهاد شده‌اند. در این پروژه، امکان تولید موسیقی با استفاده از الگوریتم‌های یادگیری ماشین کوانتومی بررسی شده و برنامه‌ی نرم‌افزاری‌ای برای انجام این کار توسعه داده شده است.

واژه‌های کلیدی:

محاسبات کوانتومی، یادگیری ماشین، تولید موسیقی کامپیوتری

فهرست مطالب

عنوان صفحه

فهرست معادلات

فهرست اشکال

شکل صفحه

فهرست الگوریتم‌ها

فصل اول

مقدمه

با گذشت چهل سال از طرح ایده‌ی محاسبات کوانتومی توسط ریچارد فاینمن [۸]، توجه جهان به کامپیوترهای کوانتومی در بالاترین سطح خود قرار دارد. عمده‌ی این توجه پس از ابداع الگوریتم فاکتورگیری شور[۱] و الگوریتم جست‌وجوی گروور[۲] در دهه‌ی نود میلادی آغاز شده است. از آن روز تاکنون، تلاش‌های بسیار زیادی برای طراحی الگوریتم‌های جدید برای کاربردهای متنوع و دسترسی به برتری محاسباتی کوانتومی در حال انجام هستند. در زمان تحریر این گزارش، شرکت‌های بسیاری از جمله آی‌بی‌ام، مایکروسافت و گوگل در حال سرمایه‌گذاری‌های جدی بر روی ساخت کامپیوترهای کوانتومی هستند. به عنوان مثال، در سال ۲۰۱۹، شرکت گوگل آزمایشی [۳] انجام داد که نشان‌گر برتری محاسباتی کوانتومی بود، به این معنا که پیچیدگی زمانی انجام این آزمایش در حالت کلاسیک بسیار بزرگ‌تر از حالت کوانتومی است.

با این وجود، به علت محدودیت تعداد کیوبیت[۳]های کامپیوترهای کوانتومی فعلی، تلاش‌های اندکی در راستای تولید قطعات موسیقی با استفاده از الگوریتم‌های کوانتومی صورت گرفته است. دو الگوریتمی که تاکنون در این راستا تحت بررسی قرار گرفته‌اند، تولید موسیقی با استفاده از تولیدکننده‌ی اعداد تصادفی[۴] و گشت کوانتومی روی گراف[۵] [۱۹] بوده‌اند، اما به نظر می‌رسد به علت وجود معادلات موجی در فیزیک کوانتومی و این حقیقت که موسیقی در اصل تشکیل شده از موج‌های صوتی است، در آینده می‌توان ارتباطات خیلی بیش‌تری بین تولید موسیقی و محاسبات کوانتومی کشف کرد.

۱ـ۱ هدف پروژه

هدف این پروژه، طراحی نرم‌افزاری است که بتواند با استفاده از الگوریتم‌های موجود یادگیری ماشین کوانتومی و یک مجموعه داده از برخی موسیقی‌های موجود، اقدام به تولید قطعه‌ی موسیقی[۶] جدیدی کند. مجموعه داده‌ی این پروژه به صورت سری گسسته‌ای از نت‌ها و آکوردهای موسیقی است که به صورت یک فایل midi ذخیره شده‌اند. عمده‌ی نرم‌افزارهای پخش محتوای چندرسانه‌ای، قابلیت پخش موسیقی با استفاده از فایل‌های midi به عنوان ورودی را دارند، اما شایان ذکر است که این نوع ذخیره‌سازی، با ذخیره‌سازی‌های شناخته‌شده‌تر محتوای صوتی مانند فایل‌های mp3 متفاوت است؛ چراکه فایل‌های mp3 با استفاده از پردازش سیگنال و تبدیل فوریه اقدام به ذخیره‌سازی می‌کنند.

[1] Shor's factorization algorithm

[2] Grover's quantum search algorithm

[3] معادل کوانتومی یک بیت کلاسیک

[4] Random number generator

[5] Quantum walk on graph

[6] Musical piece

۱ـ۲ اجزاء پروژه

پیاده‌سازی این پروژه با زبان برنامه‌نویسی پایتون[۷] انجام شده و به‌صورت کلی شامل سه بخش می‌شود:

۱. ماژول Midi که مسئولیت پردازش داده‌های موسیقی را بر عهده دارد. این ماژول تعدادی فایل که هرکدام از آن‌ها متشکل از چندین نت و آکورد است را به صورتی که به عنوان ورودی برای الگوریتم کوانتومی به راحتی قابل استفاده باشد در می‌آورد. هم‌چنین، این ماژول پس از دریافت خروجی تولید شده به وسیله‌ی اجرای الگوریتم یادگیری ماشین کوانتومی بر روی ورودی‌ها، فایل قطعه‌ی موسیقی جدیدی را تولید می‌کند.

۲. ماژول QLSTM که با استفاده از الگوریتم حافظه‌ی طولانی کوتاه مدت کوانتومی[۸] [۵]، اقدام به تولید قطعات موسیقی می‌کند.

۳. ماژول QuGAN که با استفاده از الگوریتم شبکه‌ی زایای دشمن‌گونه کوانتومی[۹] [۱۵] [۲۵]، اقدام به تولید قطعات موسیقی می‌کند.

۱ـ۳ ابزارهای مورد استفاده برای پیاده‌سازی

برای پیاده‌سازی اجزای مختلف این پروژه، از چند کتابخانه‌ی نرم‌افزاری استفاده شده که در فصول بعدی نقش آن‌ها به صورت کامل شرح داده خواهد شد، اما در این بخش صرفا به عنوان مقدمه از آن‌ها نام برده می‌شود:

- Music21
- PyTorch
- PennyLane

۱ـ۴ ساختار پایان‌نامه

این پایان‌نامه در پنج فصل به بررسی موضوع مطرح‌شده می‌پردازد. در فصل دوم مفاهیم اولیه‌ی یادگیری ماشین، محاسبات کوانتومی و تئوری موسیقی تشریح می‌شود. در فصل سوم مروری بر تلاش‌هایی که تا پیش از این در راستای حصول این امر صورت گرفته شده انجام می‌شود. در فصل چهارم شیوه‌های نوین پیشنهاد شده توسط این پایان‌نامه مطرح می‌شود و در نهایت، در فصل پنجم و آخر به نتیجه‌گیری و مسیرهای پیشنهادی برای پژوهش‌های آتی پرداخته می‌شود.

[7] Python
[8] Quantum long short-term memory
[9] Quantum generative adversarial network

فصل دوم

مفاهیم پایه

در این فصل به معرفی مفاهیم پایه موردنیاز برای درک قسمت‌های مختلف پروژه و جزییات پیاده‌سازی آن پرداخته می‌شود.

۲ـ۱ یادگیری ماشین

در این بخش تنها به مفاهیمی از یادگیری ماشین که برای فهم فصول بعدی لازم هستند اشاره می‌شود.
در حالت کلی، برای تعریف الگوریتم یا مدل‌های یادگیری ماشین، فرض می‌شود یک مجموعه داده موجود است که داده‌های آن به صورت دسته‌هایی دوتایی به شکل زیر هستند:

$$(x_i,\ y_i)\ ;\ y_i = g(x_i) \quad \forall i : 1 \leq i \leq k \tag{۱.۲}$$

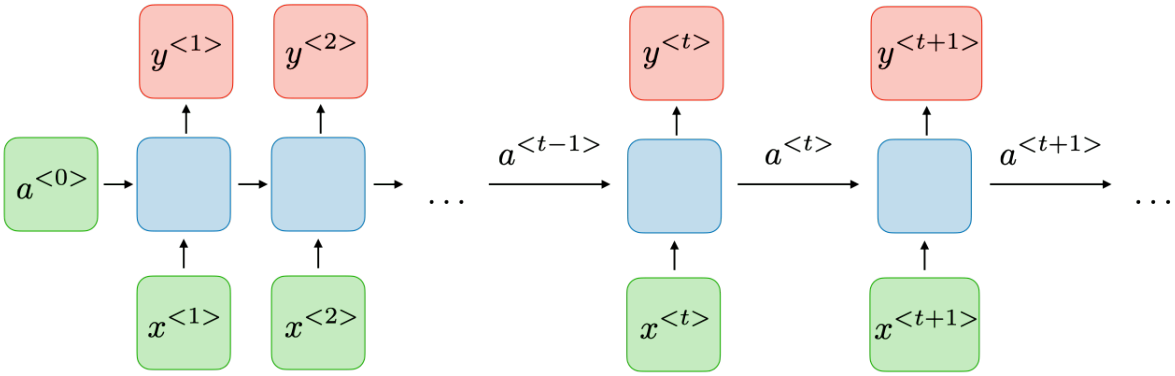
که k نشان‌گر تعداد داده‌های موجود در مجموعه داده است. این رابطه به این معنا است که فرض می‌شود بین xها و yها رابطه‌ی ریاضی‌ای وجود دارد و هدف از طراحی مدل، کشف همین رابطه است.
می‌توان الگوریتم یا مدل‌های یادگیری ماشین را به شکل تابعی به صورت زیر نمایش داد:

$$\hat{y}_i = f(x_i,\ \theta)\ ;\ x_i \in \mathbb{R}^m,\ \theta \in \mathbb{R}^n \tag{۲.۲}$$

تابع f دو نوع ورودی دریافت می‌کند، یکی x که معادل داده‌هایی‌ست که از یک مجموعه داده‌ی معین استخراج می‌شود و دیگری θ که مجموعه‌ای از پارامترهایی تنظیم‌پذیر است. در مرحله‌ی آموزش[1] این مدل، سعی می‌شود با هدف کمینه کردن یک تابع هزینه[2]، بهینه‌سازی این پارامترها به نحوی صورت گیرد که خروجی الگوریتم به جواب دلخواه نزدیک‌تر شود. تابع هزینه نیز عمدتا با این نیت تعریف می‌گردد که خروجی آن، ملاک خوبی از رضایت‌بخشی خروجی الگوریتم باشد. به عنوان مثال، اگر در یک مجموعه داده، x معادل مجموعه‌ای از اعداد در یک سری عددی و y معادل عدد بعدی ظاهرشده در این سری عددی باشد، در هنگام تعریف یک مدل یادگیری ماشین برای پیدا کردن عدد بعدی یک مجموعه با گرفتن اعداد قبلی آن، می‌توان تابع هزینه در هر مرحله از آموزش را به این صورت تعریف کرد:

$$\mathcal{L}_f = \frac{1}{k}\sum_{i=1}^{k}(\hat{y}_i - y)^2 \tag{۳.۲}$$

[1] Training phase
[2] Cost function



شکل ۲ ـ ۱: ساختار شبکه‌های عصبی بازگشتی [۲]

به این نوع تابع هزینه، تابع خطای میانگین مربعات[۳] گفته می‌شود.

بهینه‌سازی پارامترها در الگوریتم‌های یادگیری ماشین، به طور معمول توسط الگوریتم کاهش گرادیانی[۴] انجام می‌شود؛ به این معنا که میزان تغییرات پارامترها در زمان، طبق معادلات زیر به تابع هزینه وابسته می‌شود.

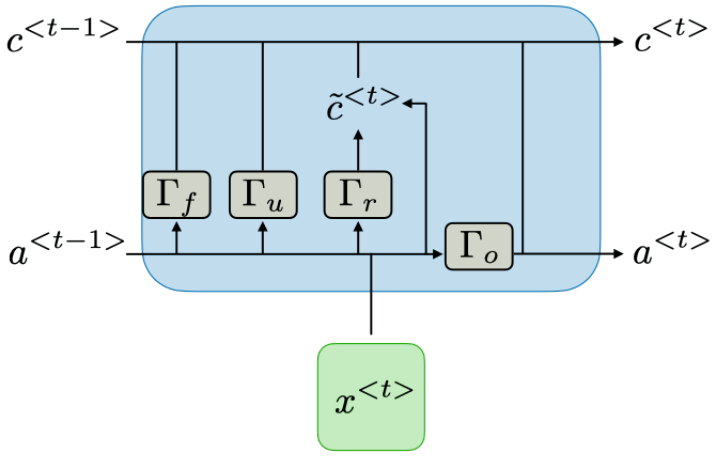
$$\frac{\partial \theta_j(t)}{\partial t} = -\eta \frac{\partial \mathcal{L}_f}{\partial \theta_j} = -\eta \sum_i \frac{\partial f\left(x_i, \theta(t)\right)}{\partial \theta_j} \frac{\partial \mathcal{L}_f}{\partial f\left(x_i, \theta(t)\right)} \tag{۴.۲}$$

در معادله‌ی بالا، متغیر η به نرخ یادگیری[۵] معروف است و میزان تغییرات پارامترها با توجه به گرادیان تابع هزینه را تنظیم می‌کند. پارامترهایی همچون نرخ یادگیری که برای کنترل پروسه‌ی آموزش مدل‌های یادگیری ماشین استفاده می‌شوند و در طی پروسه‌ی یادگیری تقریبا ثابت هستند، به نام ابرپارامتر[۶] معروف هستند.

۲ ـ ۱ ـ ۲ شبکه‌های عصبی بازگشتی

شبکه‌های عصبی بازگشتی[۷] گونه‌ی خاصی از الگوریتم‌های یادگیری ماشین هستند که ساختار کلی آن‌ها در شکل ۲ ـ ۱ آمده است. در این‌گونه شبکه‌های عصبی نیز xها نشانگر داده‌های ورودی از مجموعه داده‌ها و yها نشانگر خروجی‌های الگوریتم هستند. پارامترهای تنظیم‌پذیر در واحدهای آبی‌رنگ که به واحدهای بازگشتی[۸] معروف هستند قرار می‌گیرند. نکته‌ی اصلی شبکه‌های عصبی بازگشتی این است که هر واحد بازگشتی، در هنگام انجام محاسبات، از نتایج محاسبات واحدهای محاسباتی قبل از خود استفاده می‌کند، چراکه در این صورت می‌تواند با کسب آگاهی از خروجی‌های گذشته و ترتیب آن‌ها، خروجی معنی‌داری تولید کند. این‌گونه الگوریتم‌ها اغلب در

[3] Mean squared error
[4] Gradient descent
[5] Learning rate
[6] Hyperparameter
[7] Recurrent neural networks
[8] Recurrent block



شکل ۲ـ۲: واحد بازگشتی حافظه‌ی طولانی کوتاه‌ـمدت [۲]

یادگیری ویژگی‌های داده‌های ترتیبی[9] کاربرد دارند؛ به این معنا که نه تنها خود خروجی‌های الگوریتم، بلکه ترتیب آن‌ها هم از اهمیت بالایی برخوردار است. نت‌های موسیقی را نیز می‌توان به صورت مجموعه‌ای از داده‌های ترتیبی در نظر گرفت، چراکه هر مجموعه‌ای از نت‌های موسیقی، صدایی آهنگین تولید نمی‌کند و ترتیب نت‌های قرار گرفته در یک قطعه‌ی موسیقی نیز برای این‌که توسط گوش انسان به عنوان یک موسیقی حقیقی در نظر گرفته شوند حائز اهمیت است.

۲ـ۱ـ۱ـ۱ حافظه‌ی طولانی کوتاه‌ـمدت

حافظه‌ی طولانی کوتاه‌ـمدت[10] نوع خاصی از شبکه‌های عصبی بازگشتی است که ساختار کلی واحد بازگشتی آن در شکل ۲ـ۳ آمده است. در این شکل، المان $a^{(t)}$ بردار وضعیت نهان[11] و المان $c^{(t)}$ بردار وضعیت واحد[12] نام دارد. یکی از ویژگی‌های مهمی که حافظه‌های طولانی کوتاه‌ـمدت را در مقایسه با باقی شبکه‌های عصبی بازگشتی متمایز می‌کند، این است که به جای انتقال یک واحد اطلاعات از هر مرحله به مرحله‌ی بعدی، دو واحد اطلاعات را منتقل می‌کند [۱۲].

گیت‌های بازگشتی، واحدهای پردازشی‌ای هستند که به طور معمول در شبکه‌های عصبی بازگشتی حضور دارند و با علامت Γ نشان داده می‌شوند. حالت کلی عملیات این گیت‌ها به صورت زیر است:

$$\Gamma = \sigma(Wx^{(t)} + Ua^{(t-1)} + b) \tag{۵.۲}$$

که در معادله‌ی بالا، پارامترهای W, U, b همان پارامترهای تنظیم‌پذیر الگوریتم هستند و σ یک تابع غیرخطی

[9] Sequential data
[10] Long short-term memory
[11] Hidden state vector
[12] Cell state vector

است که از آن برای تعمیم توانایی مدل‌سازی شبکه‌های عصبی استفاده می‌شود و معمولا تابع فعال‌سازی[۱۳] نامیده می‌شود. در حافظه‌های طولانی کوتاه مدت، معمولا از تابع سیگموید[۱۴] به عنوان تابع فعال‌سازی استفاده می‌شود که تعریف این تابع به صورت زیر است:

$$S(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1} \tag{۶.۲}$$

همان‌طور که در شکل ۲ـ۳ مشاهده می‌شود، هر واحد بازگشتی حافظه‌های طولانی کوتاه‌ـ مدت شامل چهار گیت بازگشتی است که هر کدام به منظور خاصی تعبیه شده‌اند و سعی در پیاده‌سازی رفتار خاصی دارند:

- گیت به‌روزرسانی[۱۵] یا Γ_u میزان حفظ اطلاعات مراحل گذشته در محاسبات فعلی را تعیین می‌کند.
- گیت ارتباط[۱۶] یا Γ_r میزان پاک‌شدن اطلاعات مراحل گذشته در محاسبات فعلی را تعیین می‌کند.
- گیت خروجی[۱۷] یا Γ_o که میزان حفظ شدن اطلاعات محاسبات فعلی برای انتقال به مرحله‌ی بعدی را تعیین می‌کند.
- گیت فراموشی[۱۸] یا Γ_f میزان پاک‌شدن اطلاعات محاسبات فعلی برای انتقال به مرحله‌ی بعدی را تعیین می‌کند.

شایان ذکر است که پارامترهای این گیت‌های بازگشتی، معمولا پارامترهای مستقلی هستند و لذا به صورت جداگانه نیز بهینه‌سازی می‌شوند.

در نهایت، خروجی‌های واحد t ـام یک حافظه‌ی طولانی کوتاه مدت که با $c^{(t)}$ و $a^{(t)}$ نشان داده می‌شوند، به صورت زیر محاسبه می‌شوند:

$$\begin{aligned} \hat{c}^{(t)} &= tanh(W_c[\Gamma_r * a^{(t-1)}, x^{(t)}] + b_c) \\ c^{(t)} &= \Gamma_u * \hat{c}^{(t)} + \Gamma_f * c^{(t-1)} \\ a^{(t)} &= \Gamma_o * c^{(t)} \end{aligned} \tag{۷.۲}$$

معمولا در هنگام ساخت حافظه‌های طولانی کوتاه مدت توسط کتابخانه‌های یادگیری ماشین، پارامتری با نام حذف[۱۹] که عددی بین صفر و یک است نیز به واحدهای بازگشتی داده می‌شود. عملکرد این پارامتر به این صورت

[13] Activation function
[14] Sigmoid function
[15] Update gate
[16] Relevance gate
[17] Output gate
[18] Forget gate
[19] Dropout

شکل ۲ـ۳: نمونه عکس پرتره‌ی تولید شده توسط شبکه‌ی زایای دشمن‌گونه [۲۴]

است که در هر بار اجرای عملیات‌های درون یک واحد بازگشتی، کسری به اندازه‌ی پارامتر حذف از خروجی‌های آن واحد به صورت تصادفی صفر می‌شوند. این کار به این علت انجام می‌گیرد که خروجی هر لایه از یک حافظه‌ی طولانی کوتاه مدت، وابستگی بیش از حدی بر روی یک واحد خاص از واحدهای قبلی خود نداشته باشد.

۲ـ۱ـ۲ شبکه‌های زایای دشمن‌گونه

هر شبکه‌ی زایای دشمن گونه[۲۰]، متشکل از دو مدل یادگیری ماشین است [۹]. در هنگام مراحل یادگیری، این دو مدل با یک‌دیگر رقابت می‌کنند و سعی می‌کنند دیگری را در یک بازی مجموع‌ـصفر[۲۱] شکست دهند. یکی از این مدل‌ها، مدل زایا[۲۲] و مدل دیگر، مدل متمایزکننده[۲۳] نام دارد. با فرض این‌که مجموعه داده‌ای با توزیع مشخصی موجود باشد، مدل زایا سعی می‌کند داده‌های ساختگی‌ای تولید کند که شباهت زیادی به داده‌های این توزیع واقعی داشته باشند. در عین حال، مدل متمایزکننده سعی می‌کند پس از گرفتن یک داده‌ی ورودی، تشخیص دهد که این داده متعلق به آن توزیع است یا خیر.

برای واضح‌تر شدن چگونگی کارکرد شبکه‌های زایای دشمن‌گونه، می‌توان به مساله‌ی تولید پرتره اشاره کرد؛ به این معنا که مجموعه داده‌ای از عکس‌های پرتره‌ی صورت انسان‌های متفاوتی موجود است. در این حالت، مدل زایا تلاش می‌کند تا عکس پرتره‌ی جدیدی تولید کند و مدل متمایزکننده با گرفتن ورودی‌ای، سعی می‌کند تشخیص دهد که این ورودی توسط مدل زایا تولید شده یا از مجموعه داده‌ی اصلی نمونه‌برداری شده است. در نهایت، در صورت موفق بودن آموزش این دو مدل، مدل زایا می‌تواند عکس‌های پرتره‌ی جدیدی تولید کند که ساختگی بودن آن‌ها حتی توسط خود انسان‌ها هم ممکن نباشد و مدل متمایزکننده می‌تواند عکس‌های ساختگی را از عکس‌های واقعی به خوبی تشخیص دهد.

[20] Generative adversarial network

[21] Zero-sum game

[22] Generative model

[23] Discriminative model

۲ـ۲ محاسبات کوانتومی

۲ـ۲ـ۱ سیستم‌های تک‌کیوبیتی

طبق قوانین فیزیک کوانتومی، حالت یک سیستم می‌تواند به صورت ترکیب خطی‌ای از چند حالت پایه باشد. این حالت‌های پایه، حالت‌هایی هستند که در قوانین فیزیک کلاسیک نیز حالات صحیحی برای توصیف سیستم هستند. در محاسبات کوانتومی از بردارهای عمودی برای نشان‌دادن وضعیت یک سیستم استفاده می‌شود و وضعیت سیستم‌های چندکیوبیتی را نیز می‌توان از روی بردارهای یک سیستم تک‌کیوبیتی نیز ساخت.
به طور معمول، در محاسبات کوانتومی، صرفا سیستم‌هایی که دارای تنها دو حالت پایه هستند بررسی می‌شوند. به همین علت، بردارهای فضای حالات سیستم‌های تک‌کیوبیتی دو بعدی هستند. لذا دو بردار مستقل یکه به عنوان بردارهای پایه‌ی این فضا تعیین می‌شوند که رابطه‌ی یک به یکی با حالات پایه‌ی بیت‌های کلاسیک دارند. این حالات پایه به صورت زیر تعیین می‌شوند:

$$|0\rangle = \begin{bmatrix}1\\0\end{bmatrix} \quad |1\rangle = \begin{bmatrix}0\\1\end{bmatrix} \tag{۸.۲}$$

نشان دادن بردارهای حالت به صورت $|0\rangle$ و $|1\rangle$ به نمادگذاری برا–کت دیراک[۲۴] معروف است و به ازای هر کت به صورت زیر:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{bmatrix}\alpha\\\beta\end{bmatrix} \tag{۹.۲}$$

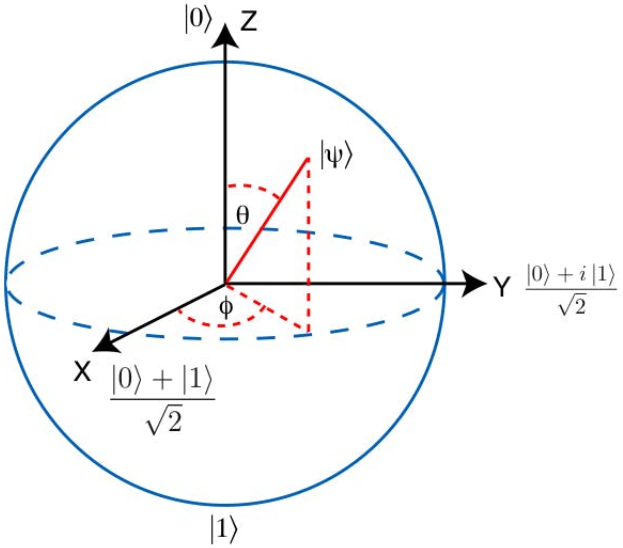
یک برا به این صورت تعریف می‌شود:

$$\langle\psi| = \alpha^*\langle 0| + \beta^*\langle 1| = \begin{bmatrix}\alpha^* & \beta^*\end{bmatrix} \tag{۱۰.۲}$$

که در این‌جا نماد α^* به معنای مزدوج مختلط[۲۵] عدد α است.

[24] Dirac's bra-ket notation

[25] Complex conjugate



شکل ۲ـ۴: کره‌ی بلاخ

حالت‌های سیستم در فیزیک کوانتومی در اکثر اوقات به صورت بردارهای مختلط بهنجار[۲۶] نشان داده می‌شوند.

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{bmatrix}\alpha\\ \beta\end{bmatrix}; \quad \alpha, \beta \in \mathbb{C} \tag{۱۱.۲}$$

که به امکان ایجاد یک بردار وضعیت از ترکیب خطی دو بردار وضعیت دیگر، اصل برهم‌نهی کوانتومی[۲۷] گفته می‌شود.

بهنجار بودن به معنای صدق شرایط زیر است:

$$\alpha\alpha^* + \beta\beta^* = |\alpha|^2 + |\beta|^2 = 1 \tag{۱۲.۲}$$

به علت شرط بهنجاری، می‌توان حالت کلی یک سیستم تک‌کیوبیتی را به صورت زیر نیز نوشت:

$$|\psi\rangle = \begin{bmatrix} e^{i\phi_1}\cos\frac{\theta}{2} \\ e^{i\phi_2}\sin\frac{\theta}{2}\end{bmatrix} \quad \theta, \phi_1, \phi_2 \in \mathbb{R} \tag{۱۳.۲}$$

[26] Normalizable

[27] Quantum Superposition

که آن را می‌توان به صورت زیر نیز نوشت:

$$|\psi\rangle = e^{i\phi_1}\begin{bmatrix}\cos\frac{\theta}{2}\\ e^{i\phi_2-\phi_1}\sin\frac{\theta}{2}\end{bmatrix} = e^{i\phi_1}\begin{bmatrix}\cos\frac{\theta}{2}\\ e^{i\phi}\sin\frac{\theta}{2}\end{bmatrix} \Rightarrow |\psi\rangle = \begin{bmatrix}\cos\frac{\theta}{2}\\ e^{i\phi}\sin\frac{\theta}{2}\end{bmatrix} \quad (۱۴.۲)$$

در این معادله، ϕ_1 فاز کلی سیستم نامیده می‌شود که طبق قوانین فیزیک کوانتومی، در رفتار سیستم فاقد اهمیت است و به همین علت در مرحله‌ی آخر از آن صرف نظر شده است.
در نهایت، حالت کلی یک کیوبیت را می‌توان با استفاده از ابزاری به نام کره‌ی بلاخ (شکل ۲ـ۴) نمایش داد.

۲ـ۲ـ۲ سیستم‌های چندکیوبیتی

دو سیستم تک‌کیوبیتی جداگانه را می‌توان به طور مجزا و به صورت زیر نمایش داد:

$$|a\rangle = \begin{bmatrix}a_0\\ a_1\end{bmatrix}, \quad |b\rangle = \begin{bmatrix}b_0\\ b_1\end{bmatrix} \quad (۱۵.۲)$$

در عین حال، می‌توانیم بردار وضعیت آن‌ها را به صورت هم‌زمان با استفاده از عملگری به نام ضرب تانسوری[۲۸] تعریف کنیم که به صورت زیر عمل می‌کند:

$$|b\rangle \otimes |a\rangle = \begin{bmatrix}b_0 \times \begin{bmatrix}a_0\\ a_1\end{bmatrix}\\ b_1 \times \begin{bmatrix}a_0\\ a_1\end{bmatrix}\end{bmatrix} = \begin{bmatrix}b_0a_0\\ b_0a_1\\ b_1a_0\\ b_1a_1\end{bmatrix} = |ba\rangle \quad (۱۶.۲)$$

۲ـ۲ـ۳ درهم‌تنیدگی

درهم‌تنیدگی کوانتومی[۲۹]، یکی از اصول فیزیک کوانتومی است و به این معناست که برخی بردار وضعیت‌های سیستم‌های چندکیوبیتی را نمی‌توان به صورت ضرب تانسوری دو بردار تک‌کیوبیتی مجزا تعریف کرد. این امر نشان‌گر این است که وضعیت این دو کیوبیت به هم وابسته هستند. به عنوان مثال، اگر بردارهای زیر که در

[28] Tensor product
[29] Quantum Entanglement

محاسبات کوانتومی به وضعیت‌های بل[۳۰] معروف هستند در نظر گرفته شوند:

$$|\Phi_{\pm}\rangle = \frac{1}{\sqrt{2}}\big(|0\rangle|0\rangle \pm |1\rangle|1\rangle\big), \qquad |\Psi_{\pm}\rangle = \frac{1}{\sqrt{2}}\big(|0\rangle|1\rangle \pm |1\rangle|0\rangle\big) \tag{۱۷.۲}$$

مشاهده می‌شود که هیچ‌کدام از این بردارها را نمی‌توان به صورت ضرب تانسوری‌ای از ترکیب خطی بردارهای $|0\rangle$ و $|1\rangle$ نوشت.

۲ـ۲ـ۴ گیت‌های کوانتومی

در کامپیوترهای کلاسیک، محاسبات با استفاده از گیت‌هایی همانند AND، OR و NOT انجام می‌شود. معادل این گیت‌ها در محاسبات کوانتومی، گیت‌های کوانتومی هستند. این گیت‌ها به فرم ماتریس‌های یکانی[۳۱] $2^n \times 2^n$ هستند که در این‌جا، عدد n نشان‌گر تعداد کیوبیت‌های سیستم است. اعمال شدن هر گیت کوانتومی بر روی یک سیستم، به معنای ضرب‌شدن ماتریس آن گیت بر روی بردار وضعیت کیوبیت‌های سیستم است. به دلیل استفاده از گیت‌های کوانتومی، الگوریتم‌های کوانتومی به نام مدارهای کوانتومی نیز مطرح هستند و عمق یک مدار کوانتومی، به معنای بیشینه‌ی تعداد گیت‌هایی است که بر روی هر کدام از کیوبیت‌ها اعمال می‌شود.

۲ـ۲ـ۴ـ۱ گیت‌های کوانتومی تک‌کیوبیتی

در این بخش، صرفا تعدادی از گیت‌های کوانتومی به صورت خلاصه معرفی می‌شوند و اثر آن‌ها بر روی پایه‌های برداری فضای سیستم‌های تک‌کیوبیتی نشان داده می‌شود؛ چراکه تاثیر این گیت‌ها بر بردار وضعیت کیوبیت‌های دلخواه، با استفاده از ترکیب خطی تاثیر این گیت‌ها بر پایه‌های برداری به دست می‌آید.

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \qquad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \qquad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \qquad H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{۱۸.۲}$$

گیت‌های X, Y, Z به گیت‌های پائولی[۳۲] و گیت H به گیت هادامارد[۳۳] معروف است. تمامی گیت‌های تک‌کیوبیتی، حالت خاصی از گیت پارامتردار U_3 هستند.

[30] Bell states
[31] Unitary matrix
[32] Pauli gates
[33] Hadamard gate

$$U_3(\theta,\phi,\lambda) = \begin{bmatrix} \cos(\frac{\theta}{2}) & -e^{i\lambda}\sin(\frac{\theta}{2}) \\ e^{i\phi}\sin(\frac{\theta}{2}) & e^{i(\phi+\lambda)}\cos(\frac{\theta}{2}) \end{bmatrix} \tag{۱۹.۲}$$

گیت‌های دوران پائولی[۳۴] ، گیت‌هایی هستند که با استفاده از رابطه‌های زیر به دست می‌آیند و به معنای چرخش وضعیت کیوبیت حول محورهای مختصات مختلف با زاویه‌ی ϕ هستند.

$$\begin{aligned} R_x(\phi) = e^{-i\phi X/2} &= \begin{bmatrix} \cos(\phi/2) & -i\sin(\phi/2) \\ -i\sin(\phi/2) & \cos(\phi/2) \end{bmatrix} \\ R_y(\phi) = e^{-i\phi Y/2} &= \begin{bmatrix} \cos(\phi/2) & -\sin(\phi/2) \\ \sin(\phi/2) & \cos(\phi/2) \end{bmatrix} \\ R_z(\phi) = e^{-i\phi Z/2} &= \begin{bmatrix} e^{-i\phi/2} & 0 \\ 0 & e^{i\phi/2} \end{bmatrix} \end{aligned} \tag{۲۰.۲}$$

همان‌طور که گفته شد، اعمال گیت بر روی یک سیستم به معنای ضرب ماتریس آن گیت بر روی بردار وضعیت کیوبیت‌های سیستم است؛ پس به عنوان مثال مشاهده می‌شود:

$$\begin{aligned} X|0\rangle &= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle \\ H|0\rangle &= \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 1 \end{bmatrix} = |+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\ H|1\rangle &= \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ -1 \end{bmatrix} = |-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \\ Z|+\rangle &= \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}\frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ -1 \end{bmatrix} = |-\rangle \end{aligned} \tag{۲۱.۲}$$

بردار وضعیت‌هایی که در معادله‌ی بالا با اسامی $|+\rangle$ و $|-\rangle$ مشخص شده‌اند، بردار وضعیت‌های مهمی هستند و در الگوریتم‌های بسیار زیادی ظاهر می‌شوند.

۲ـ۲ـ۴ـ۲ گیت‌های کوانتومی چندکیوبیتی

گیت‌های چندکیوبیتی نیز، همانند بردارهای وضعیت سیستم‌های چندکیوبیتی، دو نوع متفاوت دارند. در این بخش ـ برای سادگی محاسبات ـ تنها گیت‌های دوکیوبیتی بررسی می‌شود؛ اما همین روابط برای تعداد کیوبیت‌های

[34] Pauli rotation

بالاتر نیز صادق است.
نوع اول، گیت‌هایی هستند که می‌توان آن‌ها را به صورت ضرب تانسوری دو گیت تک‌کیوبیتی تجزیه کرد.
به عنوان مثال:

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \mathbb{I} \otimes X \tag{۲۲.۲}$$

این گیت معادل این است که هم‌زمان یک گیت همانی یا $\mathbb{I}$ بر روی کیوبیت اول و یک گیت X بر روی کیوبیت دوم اعمال شود.

نوع دوم، گیت‌هایی هستند که به ضرب تانسوری دو گیت تک‌کیوبیتی تجزیه‌پذیر نیستند و تنها همین نوع گیت‌ها هستند که در هنگام اعمال بر روی برخی از حالت‌های کیوبیتی برهم‌نهاده، منجر به ایجاد درهم‌تنیدگی می‌شوند، به عنوان مثال، گیت $CNOT$[۳۵] به این صورت تعریف می‌شود:

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{۲۳.۲}$$

به عنوان یک مثال از ایجاد درهم‌تنیدگی و برهم‌نهی، می‌توان دید:

$$CNOT(H \otimes I(|00\rangle)) = CNOT(\frac{1}{\sqrt{2}}\big(|0\rangle + |1\rangle\big) \otimes |0\rangle) = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \tag{۲۴.۲}$$

این گیت به این دلیل نام‌گذاری شده که تاثیر آن بر روی کیوبیت دوم، توسط وضعیت کیوبیت اول کنترل شده؛ به این معنا که تنها در صورتی که کیوبیت اول در وضعیت $|1\rangle$ باشد، گیت X که معادل کوانتومی گیت NOT کلاسیک است بر روی کیوبیت دوم اعمال خواهد شد.

۲ـ۲ـ۵ اندازه‌گیری

اندازه‌گیری در محاسبات کوانتومی را می‌توان به گونه‌های مختلفی تعریف کرد. در این متن، یکی از این شیوه‌ها به عنوان معیار در نظر گرفته شده و تنها به آن پرداخته می‌شود. عمل اندازه‌گیری در فیزیک کوانتومی، یک بردار

[35] Controlled NOT

وضعیت (که ممکن است برهم‌نهاده باشد) را به عنوان ورودی گرفته و یک عدد حقیقی بین ۱ـ و ۱ را به عنوان خروجی می‌دهد. این اندازه‌گیری‌ها با توجه به یک مشاهده‌پذیر [۳۶] انجام می‌گیرند. مشاهده‌پذیرها در فیزیک کوانتومی، ماتریس‌های هرمیتی[۳۷] هستند. در این متن، فرض می‌شود که همیشه اندازه‌گیری با توجه به مشاهده‌پذیر Z انجام می‌شود و به صورت زیر تعریف می‌شود:

$$\langle\psi|Z^{\otimes n}|\psi\rangle \tag{۲۵.۲}$$

که n تعداد کیوبیت‌های سیستم است. به عنوان مثال:

$$\langle 1|Z|1\rangle = \begin{bmatrix}0 & 1\end{bmatrix}\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}\begin{bmatrix}0\\ 1\end{bmatrix} = \begin{bmatrix}0 & 1\end{bmatrix}\begin{bmatrix}0\\ -1\end{bmatrix} = -1 \tag{۲۶.۲}$$

که معادل قرار گرفتن وضعیت کیوبیت بعد از اندازه‌گیری در حالت $|1\rangle$ خواهد بود. در صورتی که بردار موردنظر دچار برهم‌نهی باشد، امیدریاضی اندازه‌گیری‌های متعدد به عنوان خروجی اندازه‌گیری در نظر گرفته می‌شود.

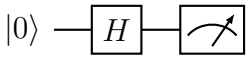
در صورتی که این اندازه‌گیری‌ها به صورت جداگانه بر روی کیوبیت‌های سیستم اعمال شوند و در سیستم درهم‌تنیدگی وجود داشته‌باشد؛ درایه‌های آرایه‌ای که از این اندازه‌گیری‌ها به وجود می‌آید به میزان درهم‌تنیدگی موجود در سیستم با هم مرتبط خواهند بود. به عنوان مثال، اگر در سیستم $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ کیوبیت اول اندازه‌گیری شود و بعد از اندازه‌گیری در حالت $|0\rangle$ قرار بگیرد، کیوبیت دوم نیز حتما در حالت $|0\rangle$ خواهد بود و بالعکس.

۲ـ۲ـ۶ نمایش دیداری مدارهای کوانتومی

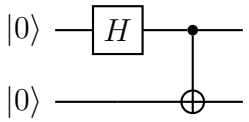
به طور معمول، مدارهای کوانتومی به صورتی مشابه با مدارهای کلاسیک نمایش داده می‌شوند؛ با این تفاوت که جای بیت‌ها و گیت‌های کلاسیک، کیوبیت‌ها و گیت‌های کوانتومی قرار گرفته‌اند و در انتهای مدار کوانتومی نیز معمولا نشانه‌هایی به معنای انجام اندازه‌گیری قرار می‌گیرد. در این بخش سه مثال از نمایش مدارهای کوانتومی نمایش داده شده است:

[36] Observable

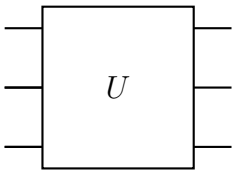
[37] Hermitian matrix



شکل ۲ـ۵: یک مدار ساده‌ی تک‌کیوبیتی



شکل ۲ـ۶: مدار دوکیوبیتی سازنده وضعیت بل



شکل ۲ـ۷: یک گیت کلی چندکیوبیتی

- مداری که در شکل ۲ـ۵ نشان داده شده، یک کیوبیت را از وضعیت $|0\rangle$ به وضعیت $|+\rangle$ برده و آن را اندازه‌گیری می‌کند.
- مداری که در شکل ۲ـ۶ نشان داده شده، بین دو کیوبیت درهم‌تنیدگی ایجاد می‌کند و یکی از وضعیت‌های بل را می‌سازد. گیت دوکیوبیتی اعمال شده در این مدار، همان گیت CNOT است.
- مداری که در شکل ۲ـ۷ نشان داده شده، یک گیت کلی است که بر روی چند کیوبیت اعمال می‌شود. این گونه علامت‌گذاری معمولا به این معناست که جزئیات درونی گیت اهمیت ندارند و صرفا نام و کارکرد گیت حائز اهمیت است.

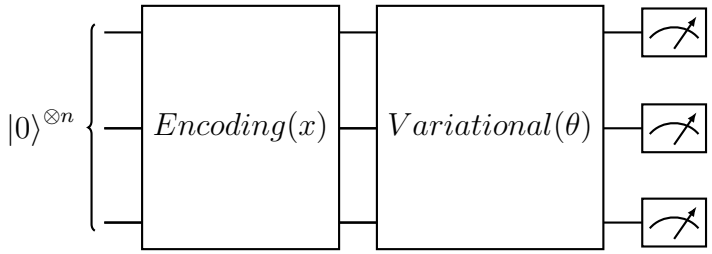
۲ـ۲ـ۷ یادگیری ماشین کوانتومی

یادگیری ماشین کوانتومی به مجموعه‌ای از الگوریتم‌ها اطلاق می‌شود که از ادغام الگوریتم‌های کوانتومی در مدل‌های یادگیری ماشین ایجاد شده‌اند. این الگوریتم‌ها به دو نوع هستند: در نوع اول، تمامی قسمت‌های الگوریتم یادگیری ماشین فرم کوانتومی به خود می‌گیرند؛ و در نوع دوم، برخی از زیر روال‌ها[۳۸] ی مدل یادگیری ماشین، با الگوریتم کوانتومی معادلی جایگزین می‌شوند. الگوریتم‌های نوع دوم، به الگوریتم‌های ترکیبی[۳۹] کوانتوم‌ـ کلاسیک معروف هستند و در حال حاضر که در عصر کامپیوترهای کوانتومی نویزدار مقیاس متوسط[۴۰] قرار داریم، به طور معمول کاربرد بیش‌تری دارند، چراکه زیرروال‌های کوانتومی استفاده شده در این الگوریتم‌ها عمدتا به تعداد کیوبیت‌های کم‌تری لازم دارند و مدار کم‌عمق‌تری دارند.

[38] Subroutine

[39] Hybrid

[40] Noisy intermediate-scale quantum (NISQ) computers



شکل ۲ـ۸: نمایش دیداری الگوریتم‌های یادگیری ماشین کوانتومی

الگوریتم‌های یادگیری ماشین کوانتومی به طور کلی از دو لایه‌ی کدگذاری[۴۱] و لایه‌ی پارامتریک[۴۲] تشکیل شده‌اند که هر کدام از این لایه‌ها در اصل یک زیرمدار کوانتومی است. لایه‌ی کدگذاری با دریافت یک داده به عنوان ورودی، با استفاده از روش‌های کدگذاری داده‌ی کوانتومی[۴۳] وضعیت اولیه‌ی کیوبیت‌ها را به نسبت داده‌های ورودی تغییر می‌دهد. لایه‌ی پارامتریک حاوی پارامترهای تغییرپذیر است که با توجه به الگوریتم گرادیان کاهشی، برای کمینه کردن تابع هزینه بهینه‌سازی می‌شود. نمایش دیداری الگوریتم‌های یادگیری ماشین کوانتومی در شکل ۲ـ۸ آمده است. شایان ذکر است که در حالت کلی، هر کدام از این لایه‌ها ممکن است چندین بار تکرار شوند و شکل ۲ـ۸ ساده‌ترین حالت این الگوریتم‌ها را نشان می‌دهد.

در یادگیری ماشین کوانتومی، می‌توان اثبات کرد که اگر یک مدار کوانتومی پارامتردار $f(x;\theta)$ از گیت‌های دوران پائولی و $CNOT$ تشکیل شده باشد، گرادیان آن را می‌توان به صورت زیر نوشت [۲۱]:

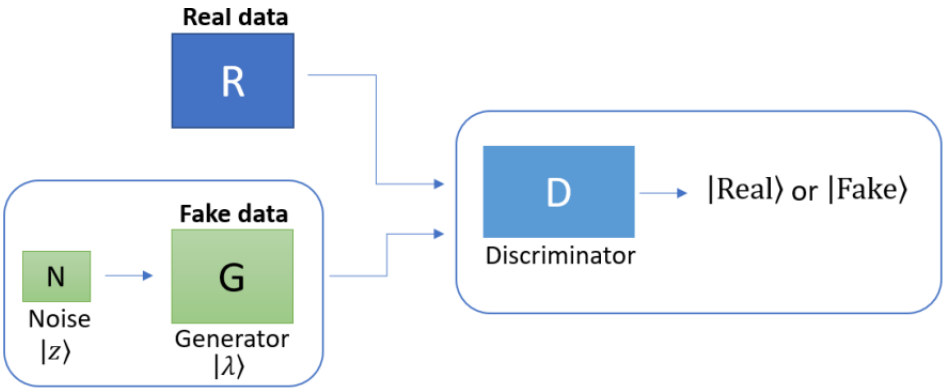
$$\nabla f(x;\theta) = \frac{1}{2}\left[f(x;\theta+\frac{\pi}{2}) - f(x;\theta-\frac{\pi}{2})\right] \qquad (۲۷.۲)$$

که این رابطه به قانون انتقال پارامتر[۴۴] معروف است.

۲ـ۲ـ۷ـ۱ حافظه‌ی طولانی کوتاه مدت کوانتومی

الگوریتم حافظه‌ی طولانی کوتاه مدت کوانتومی، از جای‌گذاری گیت‌های بازگشتی الگوریتم حافظه‌ی طولانی کوتاه مدت با معادل کوانتومی آن‌ها ایجاد شده‌است، به این معنا که خروجی گیت‌های بازگشتی، برابر با امیدریاضی اندازه‌گیری کیوبیت‌های یک مدار کوانتومی پارامتردار هستند. ساختار درونی این مدارهای کوانتومی پارامتردار متغیر هستند و بسته به مساله‌ی مورد بررسی، می‌توانند صورت‌های مختلفی به خود بگیرند.

[41] Encoding
[42] Variational
[43] Quantum data encoding
[44] Parameter-shift rule



شکل ۲ـ۹: شبکه‌ی زایای دشمن‌گونه‌ی کوانتومی

۲ـ۲ـ۷ـ۲ شبکه‌های زایای دشمن‌گونه‌ی کوانتومی

ساختار کلی شبکه‌های زایای دشمن‌گونه‌ی کوانتومی که در شکل ۲ـ۹ آمده است، به طور معمول متشکل از سه زیرروال کوانتومی به صورت زیر است:

- زیرروال اول همان مدار کدگذاری است که با گرفتن داده‌ی ورودی، آن را در وضعیت کیوبیت‌ها کدگذاری می‌کند.
- زیرروال دوم، زیرروال تشخیص است که با دریافت داده‌ی کدگذاری شده در وضعیت کیوبیت‌ها، سعی می‌کند تشخیص دهد که داده‌ی فعلی، یک داده‌ی واقعی از مجموعه داده‌ها است یا یک داده‌ی ساختگی که از جای دیگری تولید شده‌است.
- زیرروال سوم، زیرروال زایا است که با دریافت یک ورودی تصادفی به عنوان نویز، سعی می‌کند داده‌ی جدیدی که شبیه به داده‌های مجموعه داده باشد تولید کند. گرفتن نویز به این علت است که زیرروال زایا همیشه یک خروجی ثابت تولید نکند و خروجی آن در هر بار اجرا، خروجی‌ای بدیع و نوین باشد.

به خاطر ساختار ذاتی مدارهای کوانتومی، شبکه‌های زایای دشمن‌گونه‌ی کوانتومی متشکل از دو مدار هستند. مدار داده‌ی واقعی، ابتدا با استفاده از زیرروال کدگذاری، داده‌های واقعی را در وضعیت کیوبیت‌ها کدگذاری می‌کند؛ سپس با استفاده از زیرروال تشخیص، سعی می‌کند ساختار داده‌های واقعی را یاد بگیرد.

مدار داده‌ی ساختگی، ابتدا با استفاده از زیرروال کدگذاری، یک نویز را به عنوان ورودی می‌گیرد و این نویز را در وضعیت کیوبیت‌ها کدگذاری می‌کند؛ سپس با استفاده از زیرروال زایا، سعی می‌کند داده‌های جدیدی تولید کند و در نهایت با استفاده از زیرروال تشخیص، سعی می‌کند ساختگی یا واقعی بودن داده را تشخیص دهد.

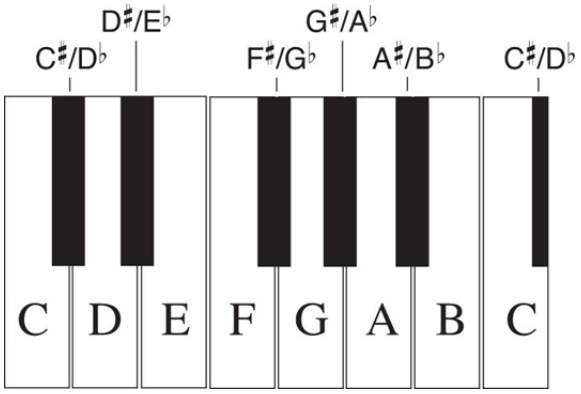
نکته‌ی مهم این است که زیرروال‌های تشخیص استفاده شده در این دو مدار، پارامترهای یکسانی دارند، به این معنا که با آپدیت شدن این پارامترها در یک مدار، پارامترهای مدار دیگر نیز تغییر خواهند کرد.

تابع هزینه‌های زیرروال تشخیص که با D و زیرروال زایا که با G نشان داده می‌شود، به صورت زیر تعریف

می‌شود:

$$Cost_D = Pr(real|fake) - Pr(real|real)$$
$$Cost_G = -Pr(real|fake) \quad (۲۸.۲)$$

به این معنا که زیرروال تشخیص تلاش می‌کند احتمال تشخیص داده‌ی واقعی به عنوان داده‌ی واقعی را افزایش دهد، در حالی که احتمال تشخیص داده‌ی ساختگی به عنوان داده‌ی واقعی را کم کند؛ در حالی که زیرروال زایا تلاش می‌کند تا احتمال تشخیص خروجی تولید شده‌اش توسط زیرروال تشخیص به عنوان داده‌ی واقعی را افزایش دهد.



شکل ۲ـ۱۰: نت‌های روی کلیدهای یک پیانو

۲ـ۳ تئوری موسیقی

تئوری موسیقی، زبان توصیف قطعات موسیقی است و همانند زبان طبیعی، قواعد و قوانین خاصی دارد.

۲ـ۳ـ۱ نت

پایه‌ای‌ترین جز یک قطعه موسیقی، نت[۴۵] نام دارد. اصوات، موج‌های مکانیکی‌ای هستند که از ارتعاش ذرات هوا تشکیل شده‌اند. هرگاه این ارتعاش فرکانس بالاتری داشته باشد، مغز انسان صدا را به عنوان صدایی زیرتر، و هروقت ارتعاش فرکانس کم‌تری باشد، صدا را به عنوان صدایی بم‌تر تشخیص می‌دهد. نت‌ها، ارتعاشاتی با فرکانس‌های معین هستند. اکثریت قریب به اتفاق موسیقی‌های تولید شده تا به امروز، از تنها ۱۲ نت تشکیل شده‌اند، چرا که طبق تجربه، سیستم شنوایی انسان این ۱۲ نت را آهنگین‌تر و منظم‌تر می‌داند. به طور خلاصه‌تر، هرگاه که نسبت فرکانس نت‌هایی که کنار هم پخش می‌شوند، به کسرهای ساده نزدیک‌تر باشند، نت‌های متفاوت در کنار هم آهنگین‌تر به نظر می‌رسند. منظور از کسرهای ساده، کسرهایی است که در هنگام نوشته‌شدن به صورت اعشاری، اعشار کم‌تری داشته‌باشند.

در یک پیانو، نت‌ها بر روی کلیدهای آن قرار گرفته‌اند و در شکل ۲ـ۱۰ قابل مشاهده هستند. نت کلیدهای سفید پیانو از حرف A تا G نام‌گذاری شده‌اند. نت کلیدهای سیاه، با توجه به نت کلیدهای سفید نام‌گذاری شده‌اند و نام‌های آن‌ها شامل علائم $\sharp$ و $\flat$ هستند. به این معنا که از چپ به راست، نت کلید سیاهی که بعد از یک کلید سفید آمده، اسم نت آن کلید سفید به علاوه‌ی یک $\sharp$ را می‌گیرد؛ و در عین حال، نام کلید سفید قبلی‌اش به علاوه‌ی یک $\flat$ را نیز می‌گیرد. پس به عنوان مثال، نت کلید سیاهی که بین کلیدهای سفید F و G قرار گرفته، $F\sharp$ یا $G\flat$ نام دارد.

[45] Note

۲ـ۳ـ۲ پرده و نیم‌پرده

جابه‌جایی از هر نت به نت بعدی آن، یک نیم‌پرده[۴۶] نام دارد. در اکثر جابه‌جایی‌ها، این نیم‌پرده به معنای رفتن از هر نت X به نت $X\sharp$ است؛ اما بین جفت نت‌های B, C و E, F نت میانی‌ای وجود ندارد، پس یک نیم‌پرده در این جفت نت‌ها، به معنای رفتن به نت بعدی است. با همین منطق، جابه‌جایی از یک نت به دومین نت بعد از آن، یک پرده[۴۷] نام دارد.

۲ـ۳ـ۳ اکتاو

همان‌طور که گفته شد، در موسیقی ۱۲ نت وجود دارد؛ اما تقریبا همه‌ی آلات موسیقی بیش‌تر از ۱۲ حالت برای نواختن نت‌ها دارند. به عنوان مثال در پیانو، این امر این‌گونه ممکن می‌شود که اگر فرکانس یک نت را دو برابر کنیم، به نتی می‌رسیم که نام همان نت را دارد، اما یک اکتاو[۴۸] بالاتر است.

طبق قضیه‌ی ریشه‌ی گویا[۴۹] در ریاضیات، به ازای هر دو عدد طبیعی a, b و عدد طبیعی n که بزرگ‌تر از ۱ است، رابطه‌ی زیر هیچ‌گاه برقرار نیست:

$$(\frac{a}{b})^n \neq 2 \qquad (۲۹.۲)$$

به همین علت، در پیانو نسبت فرکانس بین هر دو نت مجاور، $2^{\frac{1}{12}}$ است. برای دیدن چرایی این مساله، نسبت برخی از فرکانس‌های نت‌های مجاور در یک اکتاو و فواصل آن‌ها با کسرهای ساده را بررسی می‌کنیم:

[46] Half-step

[47] Whole step

[48] Octave

[49] Rational root theorem

$$
\begin{aligned}
2^{\frac{1}{12}} &\approx 1.05946309 \simeq \frac{16}{15} \;;\; error = 0.67\% \\
2^{\frac{2}{12}} &\approx 1.12246205 \simeq \frac{9}{8} \;;\; error = 0.67\% \\
2^{\frac{3}{12}} &\approx 1.18920712 \simeq \frac{6}{5} \;;\; error = 0.22\% \\
2^{\frac{4}{12}} &\approx 1.25992105 \simeq \frac{5}{4} \;;\; error = 0.89\% \\
2^{\frac{5}{12}} &\approx 1.33483985 \simeq \frac{4}{3} \;;\; error = 0.79\% \\
2^{\frac{7}{12}} &\approx 1.49830708 \simeq \frac{3}{2} \;;\; error = 0.11\% \\
2^{\frac{8}{12}} &\approx 1.58740105 \simeq \frac{8}{5} \;;\; error = 0.78\% \\
2^{\frac{9}{12}} &\approx 1.68179283 \simeq \frac{5}{3} \;;\; error = 0.90\% \\
2^{\frac{10}{12}} &\approx 1.78179744 \simeq \frac{16}{9} \;;\; error = 0.22\%
\end{aligned}
\tag{۳۰.۲}
$$

۲ـ۳ـ۴ آکورد

یک آکورد[۵۰]، مجموعه‌ای از نت‌هاست که به صورت تقریبا هم‌زمان نواخته می‌شوند و در کنار هم، صدایی متفاوت تولید می‌کنند. معمولا آکوردها متشکل از ۳ نت یا بیش‌تر هستند و با اسم اولین نت موجود در آکورد شناخته می‌شوند که به این نت، نت ریشه[۵۱] نیز گفته می‌شود. این ۳ نت موجود در یک آکورد، می‌توانند هر ۳ نتی باشند، اما لزوما هر ترکیبی از نت‌ها، صدای آهنگینی تولید نخواهد کرد. به مجموعه‌ی چند آکورد کنار هم، یک سلسله‌آکورد[۵۲] گفته می‌شود و برخی سلسله‌آکوردها به کرات در موسیقی‌های مختلف تکرار می‌شوند. آکوردها عمدتا به دو دسته‌ی مینور[۵۳] و ماژور[۵۴] تقسیم می‌شوند.

آکوردهای ماژور متشکل از سه نت هستند، ابتدا نت ریشه، سپس نتی که در سه نیم‌گام جلوتر از آن قرار دارد و در آخر نتی که چهار نیم‌گام جلوتر از نت دوم قرار دارد. آکوردهای ماژور عمدتا اصوات شادابی تولید می‌کنند.

آکوردهای مینور نیز مشابه آکوردهای ماژور هستند، با این تفاوت که نت دوم آن‌ها، چهار نیم‌پرده با نت اول فاصله دارد و نت سوم، سه نیم‌پرده با نت دوم فاصله دارد. آکوردهای مینور عمدتا اصوات غمگینی تولید می‌کنند.

[50] Chord
[51] Root note
[52] Chord progression
[53] Minor chord
[54] Major chord

۲ـ۳ـ۵ گام

هر گام[۵۵] یک گروه از نت‌هاست که هنگام نواخته‌شدن به ترتیبی خاص، صدایی آهنگین تولید می‌کنند. برخلاف آکوردها، قرار دادن هر گروهی از نت‌ها کنار هم، لزوما تشکیل یک گام نمی‌دهد. گام‌ها، همانند آکوردها با استفاده از اولین نت نواخته‌شده‌شان شناخته می‌شوند و به دو نوع هستند. یکی گام‌های کروماتیک[۵۶] که هر گام کروماتیک متشکل از ۱۲ نت در یک گام است؛ و دیگری گام دیاتونیک[۵۷] که هر گام دیاتونیک، متشکل از ۷ نت در یک گام است. هستند که هر کدام از این انواع، زیرشاخه‌های خود را دارند. به عنوان مثال، گام‌های ماژور[۵۸] و گام‌های مینور[۵۹] از زیرشاخه‌های گام‌های دیاتونیک هستند که گام‌های ماژور، ترتیبی همانند [پرده ـ پرده ـ نیم‌پرده ـ پرده ـ پرده ـ پرده ـ نیم‌پرده] و گام‌های مینور ترتیبی همانند [پرده ـ نیم‌پرده ـ پرده ـ پرده ـ نیم‌پرده ـ پرده ـ پرده] دارند. به طور ساده، به گامی که قطعه‌ای را آغاز کند و در قطعه حضور پررنگی داشته‌باشد، کلید[۶۰] آن قطعه گفته می‌شود که این کلید، شکل کلی قطعه را تعیین می‌کند. به عنوان مثال، اگر قطعه‌ای در A مینور باشد، به این معناست که با نت‌های زیر آغاز شده و این نت‌ها حضور فعالی در طول قطعه دارند:

$$A, B, C, D, E, F, G, A \tag{۳۱.۲}$$

[55] Scale
[56] Chromatic scale
[57] Diatonic scale
[58] Major scale
[59] Minor scale
[60] Key

فصل سوم

کارهای پیشین

۳ـ۱ الگوریتم‌های کلاسیک تولید موسیقی

از همان ابتدای پیدایش کامپیوترهای امروزی، موسیقی‌دان‌ها در تلاش برای تولید ملودی‌های جدید به وسیله‌ی کامپیوترها بوده‌اند. مثال‌هایی از تلاش‌ها برای تولید موسیقی به وسیله‌ی محاسبات کلاسیک در زیر آمده‌اند:

- در دهه‌ی ۴۰ میلادی، پژوهشگران مجمع علمی‌ـصنعتی استرالیا[۱] بلندگویی را به یک کامپیوتر امکا۱[۲] متصل کردند تا صدایی که کامپیوتر در حین اجرای برنامه‌ها تولید می‌کرد را بشنوند. در همین راستا و در سال ۱۹۵۱، جف هیل[۳] که ریاضی‌دانی با پیش‌زمینه‌ای در موسیقی بود، برنامه‌ای بر روی این کامپیوتر اجرا کرد تا صدای تولید شده تا حد بسیار خوبی شبیه ملودی‌های موسیقی شود [۷].

- مدل‌های ترجمه‌ای[۴] سعی می‌کنند تا داده‌های غیرصوتی را به صوت تبدیل کنند. به عنوان مثال، اگر مدلی ساخته شود که سعی کند عکس‌های با فرمت jpg را به فایل‌هایی با فرمت mp3 تبدیل کند (چرا که هر دوی این فرمت‌ها از تبدیل فوریه برای کدگذاری داده‌ها استفاده می‌کنند)، ممکن است عکس یک خط صاف را به عنوان یک موسیقی با یک نت ثابت تعبیر کند.

- مدل‌های گرامری[۵] سعی می‌کنند با پردازش کردن داده‌های موسیقیایی به عنوان جملات زبانی که همانند زبان طبیعی، از قواعد گرامری خاصی تبعیت می‌کند، الگوریتم‌های پردازش زبان طبیعی[۶] را بر روی داده‌های موسیقیایی اعمال کنند.

- متدهای تکاملی، با استفاده از چهارچوب الگوریتم‌های ژنتیک[۷]، با یک سری از نت‌های موسیقی به عنوان ژن افراد یک جمعیت برخورد می‌کنند و بعد از ترکیب کردن ژن‌های این افراد و ایجاد جهش‌های تصادفی در این ژن‌ها، موسیقی‌های جدیدی تولید کنند. به عنوان مثال، یک الگوریتم ژنتیک ممکن است دو گام زیر را به این صورت ترکیب کند:

$$\begin{aligned} C_{major} &= [C, D, E, F, G, A, B, C] \\ A_{major} &= [A, B, C\sharp, D, E, F\sharp, G\sharp, A] \\ Genetic(A_{major}, C_{major}) &= [C, D, E, F, E, F\sharp, G\sharp, C] \end{aligned} \tag{۱.۳}$$

که نیمه‌ی اول گام C_{major} با نیمه‌ی دوم گام A_{major} ترکیب شده‌است و نت آخر از A به C جهش یافته‌است.

[1] Australian council for scientific and industrial research (CSIR)
[2] MK1
[3] Geoff Hill
[4] Translational models
[5] Grammatical models
[6] Natural Language Processing (NLP)
[7] Genetic algorithms

- با پیدایش و همه‌گیر شدن حوزه‌های هوش مصنوعی و یادگیری ماشین، مدل‌های یادگیری ماشین زیادی برای تولید موسیقی پیشنهاد شده است. کتابخانه‌ی Magenta که توسط بخشی از تیم کتاب‌خانه‌ی یادگیری ماشین TensorFlow توسعه یافته‌است، شامل پیاده‌سازی انواع مختلفی از این مدل‌ها، از جمله Melody RNN [۱۷] و GANSynth [۱۶] است. مدل اول با استفاده از نوع خاصی از حافظه‌های طولانی کوتاه‌ـمدت و نوع دوم با استفاده از نوع خاصی از شبکه‌های زایای دشمن‌گونه اقدام به تولید موسیقی‌های بدیع می‌کند.

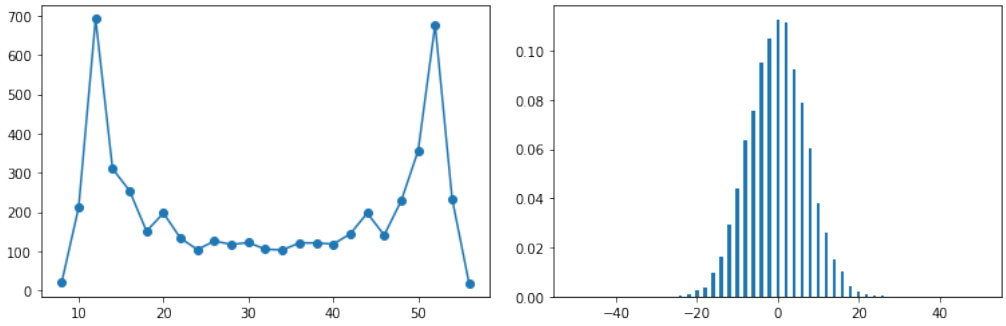
۳ـ۲ الگوریتم‌های کوانتومی تولید موسیقی

در سال‌های اخیر و با همه‌گیرتر شدن حوزه‌های کوانتومی، چندین پیشنهاد برای بررسی ارتباط بین این حوزه و موسیقی پیشنهاد شده‌است. منبع [۲۲] برای اولین بار، بررسی تولید موسیقی کوانتومی[۸] و به‌طور کلی‌تر، هنرهای کوانتومی[۹] را مطرح کرد؛ ایده‌ی اصلی این مقاله این است که در صورتی که بتوان در دنیای روزمره، موج‌های کوانتومی‌ای تولید کرد که بتوانند از نویزهای محیط در امان بمانند، می‌توان موج کوانتومی‌ای تولید کرد که در وضعیت برهم‌نهی از دو موسیقی متفاوت باشد. در صورت تحقق این امر، در هنگام شنیده‌شدن این موج توسط گوش انسان، عمل اندازه‌گیری انجام می‌گیرد و به همین خاطر، هر شنونده موسیقی متفاوتی می‌شنود. این مقاله، منجر به پروژه‌ی Quantum Music [۲۳] شد که مقالات منتشر شده در راستای این پروژه، ایده‌های الهام گرفتن از معادلات موج کوانتومی برای تولید موسیقی [۱۱] و استفاده از تصادفی‌بودن نتایج برخی اندازه‌گیری‌ها در محاسبات کوانتومی و بازپخت کوانتومی[۱۰] برای تولید موسیقی [۱۴] را به صورت ابتدایی بررسی می‌کنند.

در نهایت، منبع [۱۹] روش استفاده از گشت کوانتومی[۱۱] روی گراف را برای تولید موسیقی پیشنهاد می‌کند. مساله‌ی گشت کوانتومی روی گراف، معادل کوانتومی مساله‌ی گشت گراف کلاسیک است؛ به این معنا که یک موجود ریاضیاتی به نام گردشگر[۱۲] در یکی از گره‌های یک گراف قرار می‌گیرد، به صورتی که حرکت این موجود در گراف، تابع قوانین خاصی است. هدف این گونه مسائل، آنالیز مسیرهای طی شده توسط این موجود و توزیع احتمالاتی[۱۳] مکان نهایی آن در گراف بعد از گذشت زمان مشخصی است. اما به خاطر تفاوت‌های موجود بین این دو حوزه، نتایج آن‌ها نیز متفاوت است [۱۳]. به عنوان مثال، رابطه‌ی انحراف معیار[۱۴] (σ) توزیع احتمالاتی گشت با تعداد گام‌های طی‌شده (T) در حالات کلاسیک و کوانتوم به شکل زیر متفاوت است:

$$\sigma^2_{Classical} \sim T$$
$$\sigma^2_{Quantum} \sim T^2 \qquad (۲.۳)$$

[8] Quantum music
[9] Quantum arts
[10] Quantum annealing
[11] Quantum walk
[12] Walker
[13] Probability distribution
[14] Standard deviation



شکل ۳ـ۱: مقایسه‌ی توزیع احتمالی گشت گرافی کلاسیک (راست) و کوانتومی (چپ) [۱۰]

به طور خلاصه، الگوریتم گشت گراف کوانتومی، وضعیت گردشگر را در چند کیوبیت کدگذاری می‌کند و با فرض شروع مسیر گردشگر از یک گره خاص در گراف، در هر مرحله، با استفاده از یک کیوبیت سکه[۱۵] که در هر مرحله تحت تاثیر یک گیت هادامارد قرار می‌گیرد، وضعیت کیوبیت‌ها را به برهم‌نهی‌ای از وضعیت ندهای مجاور آن گراف می‌برد. نکته‌ای که باعث تفاوت حرکات گردشگر کلاسیک و کوانتومی می‌شود، اثر تداخل کوانتومی[۱۶] است که احتمالات حضور گردشگر در برخی نت‌ها را تقویت و در برخی دیگر، تضعیف می‌کند. مقایسه‌ی توزیع احتمال مکان نهایی گردشگر در اجرای الگوریتم گشت بر روی گراف حلقوی‌ای با شش گره در حالت کلاسیک و در حالت کوانتومی‌ای که کیوبیت سکه‌ی آن در زمان $t = 0$ از وضعیت اولیه‌ی زیر شروع شده:

$$|i\rangle = \frac{|0\rangle + i|1\rangle}{\sqrt{2}} \tag{۳.۳}$$

در شکل ۳ـ۱ آمده است. بیان ریاضی عملگرهای الگوریتم گشت کوانتومی به صورت زیر است:

$$\begin{gathered} |q_t\rangle = |state_t\rangle \otimes |coin_t\rangle \\ U = |0\rangle\langle 0| \otimes \sum_j |j+1\rangle\langle j| + |1\rangle\langle 1| \otimes \sum_j |j-1\rangle\langle j| \\ S = U(H \otimes \mathbb{I}) \\ |q_{t+1}\rangle = S|q_t\rangle \end{gathered} \tag{۴.۳}$$

که منظور از بردارهایی به شکل $|j\rangle$ و $|j+1\rangle$ نمایش وضعیت کیوبیت‌های بردارهای وضعیت به شکل عدد طبیعی است که عمل‌گرهای جمع و تفریق هم‌نهشت[۱۷] با پیمانه‌ی 2^n (که در این‌جا برابر با $2^3 = 8$ است) به

[15] Coin qubit
[16] Quantum interference
[17] Modular arithmetic

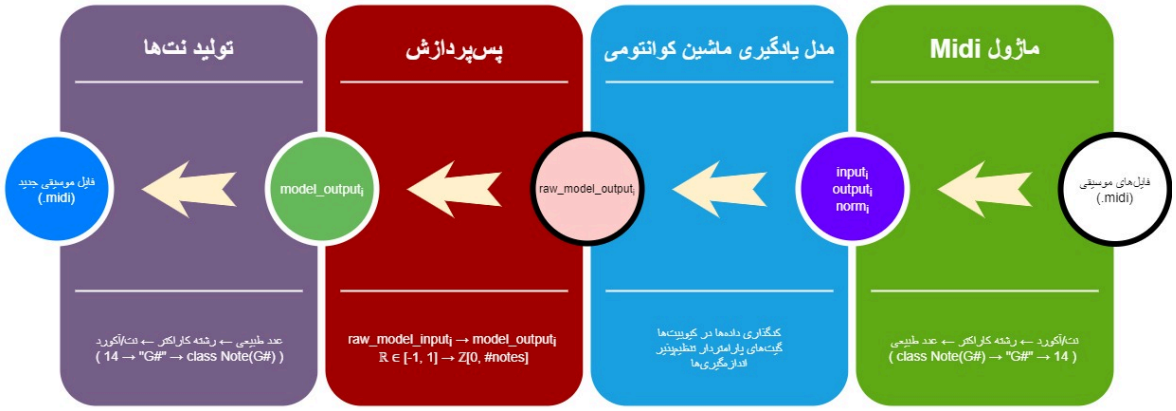
آن‌ها اعمال می‌شود. به عنوان مثال:

$$|7\rangle = |111\rangle$$
$$|7+1\rangle = |8 \; mod \; 8\rangle = |0\rangle = |000\rangle \qquad (۵.۳)$$
$$|0-1\rangle = |-1 \; mod \; 8\rangle = |7\rangle = |111\rangle$$

الگوریتم پیشنهاد شده در [۱۹] نمونه‌گیری از یک الگوریتم گشت کوانتومی روی یک گراف با هشت گره است که تنها به سه کیوبیت نیاز دارد. این الگوریتم پس از اندازه‌گیری وضعیت گردشگر گراف در انتهای چند مرحله گشت، وضعیت $|000\rangle$ را به عنوان سکوت و هفت وضعیت پایه‌ی دیگر را به عنوان هفت نت ماژور (معادل کلیدهای سفید پیانو) در نظر می‌گیرد و بر اساس چندین بار اجرای این الگوریتم، قطعه‌ای در یک کلید ماژور تولید می‌کند.

فصل چهارم

پیاده‌سازی و نتایج نو



شکل ۴ـ۱: دیاگرام مراحل کلی پیاده‌سازی شده در پروژه

در این پروژه، (تا جایی که نویسنده اطلاع دارد) برای اولین بار از الگوریتم‌های یادگیری ماشین کوانتومی برای تولید موسیقی استفاده شده است. همان‌طور که در بخش ۱ـ۲ اشاره شد، این پروژه شامل سه ماژول Midi، QLSTM و QuGAN است که در این فصل توضیحات کامل عملکرد آن‌ها شرح داده می‌شود. شایان ذکر است که موسیقی‌های تولید شده توسط این مدل، دارای تندای[1] ثابت هستند؛ به این معنا که فاصله‌ی بین نت‌های مختلف همیشه یکسان و برابر با پنجاه میلی‌ثانیه است. عملیات‌های مربوط به تئوری موسیقی، پردازش مجموعه داده به عنوان ورودی پروژه و تولید فایل‌های موسیقی به عنوان خروجی پروژه با استفاده از کتابخانه‌ی Music21 صورت می‌گیرد. مدارهای کوانتومی این پروژه، با استفاده از کتابخانه‌ی PennyLane که کتابخانه‌ای مختص یادگیری ماشین کوانتومی است طراحی شده‌اند و بهینه‌سازی پارامترهای این مدارها، با استفاده از رابط کاربری بین این کتابخانه و کتابخانه‌ی PyTorch انجام شده است. لازم به ذکر است که در این پروژه، به علت در دسترس نبودن کامپیوترهای کوانتومی موردنیاز، محاسبات کوانتومی توسط شبیه‌سازی کوانتومی موجود در کتابخانه‌ی PennyLane انجام می‌گیرد.

۴ـ۱ مراحل کلی پروژه

مراحل کلی پیاده‌سازی شده در این پروژه، در شکل ۴ـ۱ آمده است. توضیح این شکل به طور خلاصه به این صورت است: در مرحله‌ی اول، مجموعه داده‌ی این پروژه که به صورت تعدادی فایل موسیقی شامل نت‌ها و آکوردها با پسوند midi است، به عنوان ورودی وارد ماژول Midi می‌شود. این ماژول در ابتدا نگاشتی از نت‌ها و آکوردهای فایل‌های ورودی به اعداد طبیعی می‌سازد، سپس مجموعه‌ای از جفت ورودی و خروجی‌ها به صورت اعداد طبیعی را تولید می‌کند. در مرحله‌ی دوم، مجموعه‌ی تولید شده به عنوان ورودی وارد یک مدل یادگیری ماشین کوانتومی (که یکی از ماژول‌های QLSTM یا QuGAN است) می‌شود. در مرحله‌ی سوم، خروجی‌های تولید شده توسط مدل یادگیری ماشین کوانتومی که اعدادی حقیقی در بازه‌ی $[-1, 1]$ هستند، وارد مرحله‌ی پس‌پردازش می‌شوند تا این اعداد، به اعدادی طبیعی تبدیل شوند. در نهایت، خروجی‌های مرحله‌ی پس‌پردازش

[1] Tempo

الگوریتم ۱.۴ استخراج نت‌ها، آکوردها و فرکانس‌های آن‌ها از مجموعه داده

```
notes ← empty list
frequencies ← empty list
notes_file ← list of notes and chords read from file
for new_part ∈ notes_file do
  if new_part is a Note then
    notes.add(new_part.note_string)
    frequency = get_frequency_from_note(new_part)
    frequencies.add(frequency)
  else if new_part is a Chord then
    sum_frequencies = 0
    for note ∈ new_part.notes do
      frequency = get_frequency_from_note(note)
      sum_frequencies = sum_frequencies + frequency
    end for
    sum_frequencies = sum_frequencies / new_part.note_count
    notes.add(new_part.chord_string)
    frequencies.add(sum_frequencies)
  end if
end for
save notes to file: "notes.pk"
save frequencies to file: "frequencies.pk"
```

به تابعی به نام $generate\_notes$ داده می‌شود تا از روی این اعداد طبیعی، نت‌ها و آکوردهای جدیدی ساخته شوند و در یک فایل با پسوند midi ذخیره شوند. این فایل حاوی موسیقی جدید تولید شده توسط پروژه است.

۴ـ۲ ماژول Midi

ماژول Midi مسئولیت پیش‌پردازش داده‌ها برای استفاده از مدل‌های یادگیری ماشین کوانتومی را بر عهده دارد. مجموعه داده‌ی این پروژه، شامل ۹۲ قطعه‌ی موسیقی پیانو به صورت فایل‌های midi است، هرکدام از این فایل‌ها، مجموعه‌ای از نت‌ها، آکوردها و زمان پخش آن نت/آکورد از ابتدای قطعه به میلی‌ثانیه است.

۴ـ۲ـ۱ استخراج داده‌ها از مجموعه داده

این ماژول، در ابتدا تنها یک‌بار پوشه‌ی شامل مجموعه داده‌ها را بررسی می‌کند و فایل‌های midi موجود در در پوشه را به ترتیب خوانده و پردازش می‌کند. نتیجه‌ی این پردازش، دو لیست به نام notes و frequencies است که تناظر یک به یکی بین این دو لیست وجود دارد؛ به این معنا که اولین فرکانس موجود در لیست فرکانس‌ها، فرکانس صدای تولید شده توسط اولین نت موجود در لیست نت‌ها است. این ماژول پس از خواندن هر فایل

الگوریتم ۲.۴ ساخت نگاشت یک به یک از نت‌ها به اعداد طبیعی

```
notes_to_frequencies = map(keys=notes, values=frequencies)
mapping ← empty dictionary
notes_to_frequencies = sort_by_value(notes_to_frequencies)
sorted_notes = notes_to_frequencies.keys
for i := 0 to notes.length do
  mapping.update(key=i, value=sorted_notes[i])
end for
```

midi از روی دیسک کامپیوتر، موجودیت‌های حاضر در آن فایل را با استفاده از کتابخانه‌ی Music21 بررسی می‌کند. در صورتی که موجودیت بررسی شده چیزی غیر از نت یا آکورد باشد، به آن توجهی نمی‌کند و در صورتی که آن موجودیت یک نت یا آکورد باشد، آن را به صورت زیر پردازش می‌کند:
کتابخانه‌ی Music21 هر نت موجود در یک فایل midi را به صورت یک نمونه[۲] از کلاس music21.note.Note در نظر می‌گیرد که این کلاس شامل خواص مختلفی از جمله نام آن نت، اکتاو آن نت، فرکانس صدای تولید شده توسط آن نت و مدت زمان پخش آن نت است. این ماژول در هنگام پردازش یک نمونه از کلاس music21.note.Note نام آن نت را به صورت یک رشته[۳] کاراکتر و فرکانس آن نت را به صورت یک عدد حقیقی پردازش می‌کند که این رشته کاراکتر به لیست notes و این فرکانس به لیست frequencies اضافه می‌شوند. هم‌چنین، این کتابخانه هر آکورد موجود در یک فایل midi را به صورت یک نمونه از کلاس music21.chord.Chord در نظر می‌گیرد که این کلاس شامل خواص مختلفی از جمله نت‌های موجود در آن آکورد و حجم صدای تولید شده در هنگام پخش آن آکورد است. نت‌های موجود در یک کلاس music21.chord.Chord به صورت لیستی از کلاس‌های music21.note.Note هستند که همین امر، کار پردازش آکوردها را بسیار آسان‌تر می‌کند. ماژول Midi در هنگام پردازش یک نمونه از کلاس music21.chord.Chord، این آکورد را به عنوان یک نت جدید محسوب می‌کند و برای آن رشته کاراکتر و فرکانس خاصی در نظر می‌گیرد. رشته کاراکتر متناظر یک آکورد به این صورت ساخته می‌شود: نت‌های آن آکورد به صورتی پشت سر هم قرار می‌گیرند که کاراکتر نقطه (’.’) بین آن‌ها قرار گرفته باشد. فرکانس این آکورد نیز به صورت میانگین فرکانس نت‌های موجود در آن آکورد در نظر گرفته می‌شود که همانند مرحله‌ی پردازش نت‌ها، رشته کاراکتر تولید شده به لیست notes و فرکانس تولید شده به لیست frequencies اضافه می‌شوند. فرم شبه‌کدی الگوریتم استخراج نت‌ها و آکوردها از یک فایل به صورت کامل در الگوریتم ۱.۴ آمده است.

۴ـ۲ـ۲ نگاشت داده‌ها به اعداد طبیعی

ماژول Midi به جهت قابل‌فهم کردن نت‌ها برای مدل‌های یادگیری ماشین، یک نگاشت یک به یک از نت‌هایی که در لیست notes قرار گرفته‌اند به اعداد طبیعی می‌سازد. این نگاشت به صورت زیر تشکیل می‌شود:

[2] Instance
[3] String

ابتدا با توجه به تناظری که گفته شد بین لیست notes و لیست frequencies وجود دارد و با استفاده از یک جدول درهم‌سازی[۴]، یک نگاشت از نت‌ها به فرکانس‌ها با نام notes_to_frequencies ساخته می‌شود. در مرحله‌ی بعد، به دلیل این‌که جداول درهم‌سازی در زبان برنامه‌نویسی پایتون دارای ترتیب نیز هستند، می‌توان این نگاشت را با توجه به فرکانس‌ها به صورت صعودی مرتب کرد؛ به این معنا که جفت (نت، فرکانس)ی که مقدار فرکانس کم‌تری داشته‌باشد، در ابتدای این نگاشت قرار می‌گیرد. سپس نت‌های نگاشت notes_to_frequencies به صورت جداگانه استخراج شده و به عدد طبیعی متناظر مکان‌شان در ترتیب موجود نگاشت می‌شوند که این نگاشت، در متغیر mapping ذخیره می‌شود. فرم شبه‌کدی الگوریتم ساخت نگاشت یک به یک از نت‌ها به اعداد طبیعی در الگوریتم ۲.۴ آمده است.

این ماژول سپس در هر بار اجرای کد، با گرفتن پارامتری به نام SequenceLength تعداد زیادی جفت ورودی و خروجی برای مدل یادگیری ماشین کوانتومی فراهم می‌کند؛ به این صورت:

$$
\begin{aligned}
input_i &= [inotes_{(i)}, ..., inotes_{(SequenceLength+i)}] \\
norm_i &= \sqrt{\sum_{k=i}^{SequenceLength+i} (inotes_{(k)})^2} \\
output_i &= [inotes_{(SequenceLength+i+1)}] \\
where\ 0 \leq i \leq n - SequenceLength - 1\ ;\ n &= \#notes
\end{aligned}
\tag{۱.۴}
$$

که در معادله‌ی بالا، $inotes_{(i)}$ برابر با عدد طبیعی‌ای‌ست که معادل iـ امین نت در مجموعه نت‌هاست.

۳ـ۴ ماژول QLSTM

هدف این ماژول، تولید موسیقی با استفاده از حافظه‌های طولانی کوتاه مدت کوانتومی است. در این بخش، جزئیات پیاده‌سازی این ماژول شرح داده می‌شوند. در کتابخانه‌ی PyTorch، هر مدل آموزش‌پذیر یادگیری ماشین، زیرکلاسی از کلاس torch.nn.Module است. به همین دلیل، کلاس‌های این ماژول، همگی زیرکلاسی از torch.nn.Module هستند. یکی از توابع مهم کلاس torch.nn.Module، تابع forward است که در هر بار اجرا، مقداری داده به عنوان ورودی گرفته و با اعمال یک الگوریتم یادگیری ماشین به این ورودی‌ها، نتایج تولید شده توسط این الگوریتم را به عنوان خروجی می‌دهد. این کلاس‌ها در زیر شرح داده می‌شوند.

[4] Hash map

۴ـ۳ـ۱ طراحی اولیه

طراحی اولیه‌ی این ماژول، الهام گرفته از پیاده‌سازی‌های معمول تولید موسیقی با استفاده از حافظه‌های طولانی کوتاه مدت کلاسیک بود که در بخش زیر به این پیاده‌سازی‌های کلاسیک پرداخته می‌شود:

۴ـ۳ـ۱ـ۱ پیاده‌سازی کلاسیک

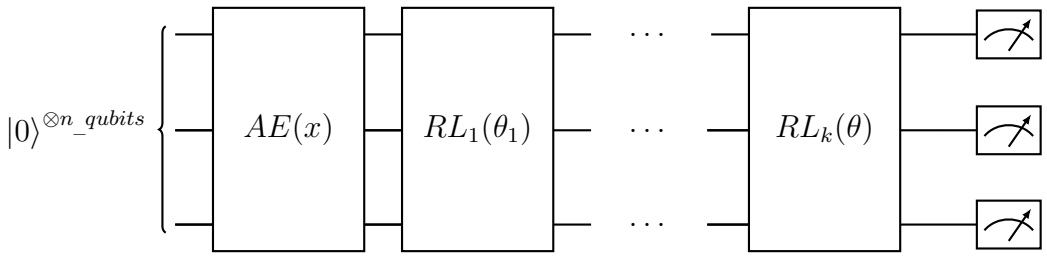
ایده‌ی کلی این پیاده‌سازی‌ها به این صورت است که در اولین مرحله، $output_i$ های تولید شده در ماژول Midi، با استفاده از کدگذاری یک بارز[۵] شکل جدیدی پیدا می‌کنند. این کدگذاری به این صورت انجام می‌شود که اگر n تعداد $output_i$ وجود داشته باشد، داده‌ی k ام این مجموعه که $output_k$ نام دارد، تبدیل به برداری n بعدی می‌شود که همه‌ی ورودی‌های آن صفر است و تنها k امین ورودی آن برابر با یک قرار می‌گیرد. در این مساله، عدد n برابر با تعداد نت‌های منحصر به فرد موجود در لیست notes تولید شده در ماژول Midi در نظر گرفته می‌شود. سپس، با گرفتن ابعاد بردارهای ورودی و خروجی گیت‌های بازگشتی به عنوان ورودی، ابعاد ماتریس‌های W و U که در معادله‌ی ۵.۲ تعریف شده بودند تعیین می‌شود و واحدهای بازگشتی با استفاده از این ابعاد و پارامتر حذفی که برابر با ۰.۳ است ساخته می‌شوند. پس از قرار دادن تعدادی لایه‌ی بازگشتی پشت سر هم، ابتدا خروجی‌های لایه‌های مختلف تحت تاثیر یک لایه‌ی خطی که در کتابخانه‌ی PyTorch به صورت کلاس torch.nn.Linear پیاده‌سازی شده است قرار می‌گیرند. کارکرد این لایه به این صورت است که با گرفتن سه عدد n, m و b برداری n بعدی را با استفاده از تبدیل خطی زیر، تبدیل به برداری m بعدی می‌کند:

$$\begin{aligned} y &= Ax + b \\ dim(A) &= m * n \end{aligned} \tag{۲.۴}$$

دلیل وجود این لایه‌ی خطی این است که بردارهایی که توسط خروجی‌های واحدهای بازگشتی ساخته شده، تبدیل به بردارهایی با طول n شوند. این کار باعث می‌شود خروجی‌های مدل، با خروجی‌هایی که تحت کدگذاری یک بارز قرار گرفته بودند، قابل مقایسه شوند. در انتها، بردار نهایی تولید شده به عنوان برداری از احتمالات خروجی داده شدن هر کدام از نت‌های موجود در لیست notes تعبیر می‌شود و تابع آنتروپی متقاطع [۶] به عنوان تابع هزینه‌ی این مدل در نظر گرفته می‌شود. تابع آنتروپی متقاطع به این صورت تعریف می‌شود که اگر N عدد دسته‌بندی وجود داشته باشد و برداری N بعدی به نام x به عنوان ورودی به این تابع داده شود، خروجی آن به صورت معادله‌ی زیر خواهد بود:

[5] One-hot encoding

[6] Cross Entropy



شکل ۴ـ۲: نمایش دیداری مدار کوانتومی استفاده شده در حافظه‌ی طولانی کوتاه مدت کوانتومی

$$loss(x, class) = -log\Big(\frac{exp(x[class])}{\sum_j exp(x[j])}\Big) = -x[class] + log\Big(\sum_j exp(x[j])\Big)$$
$$loss = \frac{\sum_{i=1}^{N} loss(i, class[i])}{N} \tag{۳.۴}$$

در مراحل اولیه‌ی پروژه، این پیاده‌سازی کلاسیک به کمک کتابخانه‌ی PyTorch به طور کامل انجام شد تا از انجام‌پذیر بودن کلیت مساله اطمینان حاصل شود.

۴ـ۳ـ۱ـ۲ پیاده‌سازی کوانتومی

در مرحله‌ی بعد، سعی شد تا با جایگزین کردن حافظه‌ی طولانی کوتاه مدت کلاسیک با یک حافظه‌ی طولانی کوتاه مدت کوانتومی، الگوریتمی کوانتومی برای تولید موسیقی طراحی شود. حافظه‌ی طولانی کوتاه مدت کوانتومی استفاده شده در این بخش، بر اساس مدلی که در بخش ۲ـ۲ـ۷ـ۱ معرفی شد طراحی شده است. بعد از پیاده‌سازی و انجام تست‌های مختلف بر روی این پیاده‌سازی با استفاده از ابرپارامترهای متفاوت، به این نتیجه رسیده شد که این‌گونه پیاده‌سازی، تنها می‌تواند با یادگیری بر روی هر مجموعه داده، یک نت ثابت تولید کند. این مساله به این دلیل است که اختلاف بسیار زیادی بین تعداد نت‌های منحصر به فرد موجود در مجموعه داده و تعداد کیوبیت‌های قابل شبیه‌سازی بر روی کامپیوترهای کلاسیک وجود دارد؛ این امر باعث می‌شود که پارامترهای موجود در لایه‌ی torch.nn.Linear مدل، نتوانند به تبدیل مناسبی از برداری با بعد پایین که از خروجی اندازه‌گیری‌های کیوبیت‌ها تولید شده است به برداری با بعدی به اندازه‌ی تعداد نت‌های منحصر به فرد موجود در مجموعه داده دست پیدا کنند. به همین خاطر، در نسخه‌ی کنونی پروژه معماری متفاوتی برای استفاده از حافظه‌های طولانی کوتاه مدت کوانتومی پیشنهاد شده که جزئیات آن به صورت کامل در بخش‌های بعدی بیان شده است.

الگوریتم ۳.۴ نحوه‌ی کارکرد یک واحد بازگشتی در حافظه‌ی طولانی کوتاه مدت کوانتومی

c_{t-1} ← Cell state vector from the previous cell
h_{t-1} ← Hidden state vector from the previous cell
x_t ← Current cell's input
drop ← Current cell's dropout value
VQC_forget ← pre-generated quantum recurrent gate
VQC_input ← pre-generated quantum recurrent gate
VQC_update ← pre-generated quantum recurrent gate
VQC_output ← pre-generated quantum recurrent gate
$y_t = [h_{t-1}\ ;\ x]$
$f_t = sigmoid(VQC\_forget(y_t))$
$i_t = sigmoid(VQC\_input(y_t))$
$g_t = sigmoid(VQC\_update(y_t))$
$o_t = sigmoid(VQC\_output(y_t))$
$c_t = (f_t * c_t) + (i_t * g_t)$
$h_t = o_t * tanh(c_t)$
$h_t = dropout(h_t, drop)$

۴ـ۳ـ۲ کلاس QLSTMCell

هر واحد بازگشتی کوانتومی این الگوریتم در کلاس QLSTMCell تعریف شده. ورودی‌های مهمی که برای ساختن نمونه‌ای از این کلاس لازم است به شرح زیر هستند:

- n_qubits تعداد کیوبیت‌های استفاده شده در مدارهای کوانتومی این واحد بازگشتی است.
- n_qlayers تعداد لایه‌های موجود در قسمت پارامتردار مدارهای کوانتومی را تعیین می‌کند.
- hidden_size ابعاد داده‌های ورودی و خروجی واحد بازگشتی را تعیین می‌کند.

این کلاس سپس با استفاده از پارامترهای دریافت شده، چهار مدار کوانتومی برای گیت‌های بازگشتی می‌سازد. هر کدام از این مدارهای کوانتومی، از ترکیب تابع کدگذاری AmplitudeEmbedding و تابع پارامتریک RandomLayers کتابخانه‌ی PennyLane ساخته شده است.

هر تابع AmplitudeEmbedding با دریافت ورودی‌ای با ابعاد 2^n، آن را در n کیوبیت هدف به صورت زیر کدگذاری می‌کند:

$$input = [x_0, x_1, \ldots, x_{2^n-1}]\ ;\ \ x_i \in \mathbb{R}$$
$$norm = \sqrt{\sum_{i=0}^{2^n} x_i^2} \qquad (۴.۴)$$
$$output = \frac{x_0}{norm}|00...0\rangle + \frac{x_1}{norm}|00...1\rangle + ... + \frac{x_{2^n-1}}{norm}|11...1\rangle$$

الگوریتم ۴.۴ نحوه‌ی کارکرد یک حافظه‌ی طولانی کوتاه مدت کوانتومی

n_layers ← Number of QLSTMCell's present in this QLSTM
QLSTMCells ← A list of initialized QLSTMCell's
inputs ← A list of inputs generated by the Midi module
outputs ← Empty list
$h_t = [0, \ldots, 0]$
$c_t = [0, \ldots, 0]$
for $i = 0 \rightarrow$ n_layers **do**
  $h_t, c_t = QLSTMCells[i](inputs[i], h_t, c_t)$
  outputs.add(c_t)
end for

و هر تابع RandomLayers با دریافت یک ورودی با ابعاد (L, k) و یک عدد طبیعی seed ،تعداد L لایه‌ی پارامتریک را با استفاده از seed به صورت تصادفی تولید می‌کند. هر لایه‌ی تولید شده توسط RandomLayers شامل ترکیبی از k گیت کوانتومی پارامتریک و تعدادی گیت $CNOT$ است.

نمایش دیداری این گیت‌های بازگشتی کوانتومی، در شکل ۴ـ۲ آمده است که این شکل، دقیقا حالت خاصی از شکل ۲ـ۸ است که در آن، از AmplitudeEmbedding برای لایه‌ی کدگذاری و از RandomLayers برای لایه‌ی پارامتریک استفاده شده است. لازم به ذکر است که در این شکل به جهت اختصار از عبارت AE به جای AmplitudeEmbedding و از عبارت RL به جای RandomLayers استفاده شده است.

پس هر واحد بازگشتی در پیاده‌سازی فعلی، یک QLSTMCell است که مدارهای کوانتومی آن، از پشت سر هم قرار گرفتن زیرمدارهای تولید شده توسط AmplitudeEmbedding و RandomLayers ساخته شده‌اند و در نهایت، آرایه‌ای متشکل از امیدریاضی اندازه‌گیری جداگانه‌ی تک‌تک کیوبیت‌های هر مدار به عنوان خروجی آن مدار در نظر گرفته می‌شود. فرم شبه‌کدی الگوریتم اجرا شده در هر بار اجرای تابع forward یک کلاس QLSTMCell به صورت کامل در الگوریتم ۳.۴ آمده است. در این الگوریتم، متغیرهای c_{t-1} و h_{t-1} که به ترتیب بردار وضعیت واحد و بردار وضعیت پنهان واحد بازگشتی قبلی هستند به صورت ورودی داده می‌شوند. متغیر x_t نیز که ورودی واحد بازگشتی کنونی است، از مجموعه داده‌ی پروژه به صورت ورودی به این الگوریتم داده می‌شود. هم‌چنین، توابع VQC_forget ،VQC_input ،VQC_update و VQC_output مدارهای کوانتومی‌ای هستند که از پیش با استفاده از پارامترهای n_qubits و n_qlayers ساخته شده‌اند و ساختارشان به صورتی که در شکل ۴ـ۲ ترسیم شده قرار دارد. drop که پارامتر حذف این واحد بازگشتی است نیز به عنوان ورودی به این الگوریتم داده می‌شود.

الگوریتم ۵.۴ پس‌پردازش ماژول QLSTM

$$raw\_model\_output_i = QLSTMCells[i](input_i)$$
$$model\_output_i = mean(raw\_model\_output_i)$$
$$model\_output_i = model\_output_i * norm_i$$

۴ـ۳ـ۳ کلاس QLSTM

این کلاس با دریافت ورودی‌ای با نام n_layers، تعداد n_layers لایه از QLSTMCell ها را در کنار هم قرار می‌دهد و تغییرات لازم برای رد کردن خروجی‌های هرکدام از این لایه‌ها به عنوان ورودی لایه‌ی بعدی را انجام می‌دهد. این کلاس در هر بار اجرا، آرایه‌ای به نام outputs که مجموعه‌ی خروجی‌های تولید شده توسط n لایه از واحدهای بازگشتی کوانتومی است را خروجی می‌دهد. فرم شبه‌کدی الگوریتم اجرا شده در هر بار اجرای تابع forward کلاس QLSTM به صورت کامل در الگوریتم ۴.۴ آمده است.

۴ـ۳ـ۴ کلاس QLSTMusic

این کلاس با گرفتن ورودی و خروجی‌های تولید شده از ماژول Midi و یک عدد طبیعی به نام n_epochs، به تعداد n_epochs بار ورودی و خروجی‌ها را به یک QLSTM رد می‌کند و بعد از انجام پس‌پردازش روی خروجی‌های تولید شده، پارامترهای آن را برای کمینه‌کردن یک تابع هزینه بهینه‌سازی می‌کند. چگونگی انجام این پس‌پردازش به طور کامل در بخش بعد شرح داده شده است.

۴ـ۳ـ۵ پس‌پردازش

هر لایه از QLSTM که شامل n کیوبیت باشد، n خروجی که هر کدام از آن‌ها عددی در بازه‌ی $[-1, 1]$ است تولید می‌کند. به همین دلیل، برای تبدیل کردن این اعداد به اعداد طبیعی‌ای که معادل نت‌های مجموعه داده باشند، باید روی خروجی‌ها پس‌پردازش انجام داد. پس‌پردازش پیشنهادی در این مرحله، در الگوریتم ۵.۴ آمده است. در این الگوریتم، متغیرهای $input_i$ و $norm_i$ طبق معادله‌ی ۴.۴ تعریف شده‌اند، $raw\_model\_output_i$ خروجی اولیه‌ای است که از مدار کوانتومی گرفته می‌شود و $model\_output_i$ خروجی نهایی بعد از پس‌پردازش است.

۴ـ۳ـ۶ تابع هزینه

به علت این‌که ممکن است عدد طبیعی معادل نت پیشنهادی، بزرگ‌تر یا کوچک‌تر از عدد طبیعی معادل نت خروجی مجموعه داده باشد، از تابع خطای میانگین مربعات که در معادله‌ی ۳.۲ معرفی شد، به عنوان تابع هزینه استفاده می‌شود. علت مناسب بودن این تابع هزینه، این است که در هنگام ساخت نگاشت یک به یک از نت‌ها به اعداد طبیعی، مرتب‌سازی به گونه‌ای انجام شد که هرچه‌قدر فرکانس نت‌ها به هم نزدیک‌تر باشد، اعداد طبیعی متناظر آن‌ها نیز به هم نزدیک‌تر باشند.

در نهایت، تابع generate_notes با گرفتن پارامتر n_notes با چندین‌بار اجرای الگوریتم‌های یادگیری ماشین و پس‌پردازش، به تعداد n_notes نت موسیقی جدید تولید کرده و با قرار دادن فاصله‌هایی به اندازه‌ی پنجاه میلی‌ثانیه بین آن‌ها، نتایج حاصل را در یک فایل با پسوند midi ذخیره می‌کند.

۴ـ۴ ماژول QuGAN

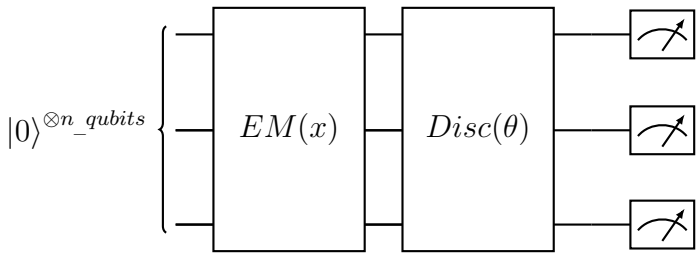
ماژول QuGAN بر اساس شبکه‌های زایای دشمن‌گونه‌ی کوانتومی که در بخش ۲ـ۲ـ۷ـ۲ معرفی شد طراحی شده است. این ماژول نیز همانند ماژول QLSTM از ماژول Midi برای پیش‌پردازش داده‌ها استفاده می‌کند.

این ماژول به طور کلی شامل سه زیرمدار کوانتومی است. از ترکیب‌های مختلف این سه زیرمدار، سه مدار کوانتومی تشکیل می‌شود که توضیح کارکرد آن‌ها در زیر آمده است:

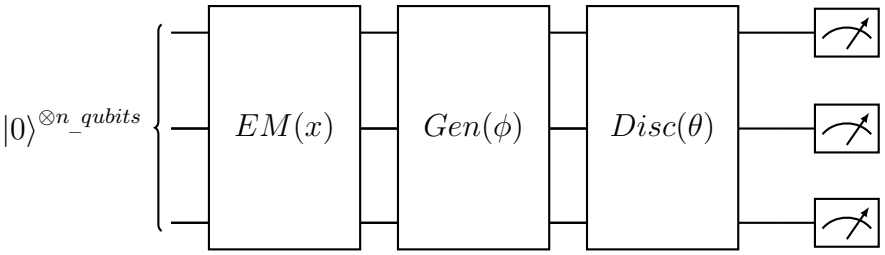
- زیرمدار اول که در کد پروژه توسط تابع encode_music ساخته می‌شود، با دریافت نت‌هایی به عنوان ورودی، آن‌ها را با استفاده از تابع AmplitudeEmbedding در کیوبیت‌های سیستم کدگذاری می‌کند.
- زیرمدار دوم که زیرمداری پارامتریک است و در کد پروژه توسط تابع discriminator ساخته می‌شود، با استفاده از تابع RandomLayers لایه‌هایی پارامتریک تولید می‌کند. این لایه‌ها سعی در تشخیص واقعی یا ساختگی بودن داده‌های موجود در کیوبیت‌ها دارند و عمل‌کرد متمایزکنندگی شبکه را پیاده‌سازی می‌کنند. خروجی این زیرمدار، آرایه‌ای متشکل از امیدریاضی اندازه‌گیری تک‌تک کیوبیت‌های سیستم به صورت مجزا است.
- زیرمدار سوم، همانند زیر مدار دوم مداری پارامتریک است، اما سعی در تولید داده‌هایی ساختگی از روی کیوبیت‌هایی که مقداری نویز به عنوان داده‌ی اولیه بر روی آن‌ها کدگذاری شده‌اند دارد. این زیرمدار در کد پروژه توسط تابع music_generator ساخته می‌شود و عملکرد زایایی شبکه را پیاده‌سازی می‌کند.

و مدارهای کوانتومی این ماژول، به این شکل ساخته شده‌اند:

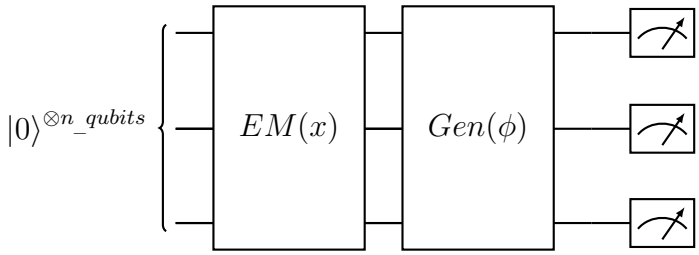
- مدار اول که real_music_discriminator نام دارد، ابتدا به کمک تابع encode_music تعدادی نت از مجموعه داده گرفته و آن‌ها را در کیوبیت‌های سیستم کدگذاری می‌کند، سپس زیرمدار discriminator سعی می‌کند با بهینه‌سازی پارامترهای خود تشخیص دهد آیا داده‌ها واقعی هستند یا خیر. نمایش دیداری این مدار در شکل ۴ـ۳ آمده است.
- مدار دوم که generated_music_discriminator، نام دارد، ابتدا با گرفتن مقداری نویز به عنوان ورودی، آن نویزها را توسط encode_music در سیستم کدگذاری می‌کند، سپس زیرمدار music_generator با بهینه‌سازی پارامترهای خود، اقدام به ساخت داده‌ی جدیدی از روی نویز کدگذاری شده می‌کند و در نهایت، زیرمدار discriminator سعی می‌کند با بهینه‌سازی پارامترهای خود، تشخیص دهد آیا داده یکی از داده‌های واقعی است یا خیر. نمایش دیداری این مدار در شکل ۴ـ۴ آمده است.



شکل ۴ـ۳: نمایش دیداری مدار real_music_discriminator



شکل ۴ـ۴: نمایش دیداری مدار generated_music_discriminator



شکل ۴ـ۵: نمایش دیداری مدار کوانتومی final_music_generator

- مدار سوم که final_music_generator نام دارد، با ترکیب زیرمدارهای encode_music و music_generator ابتدا مقداری نویز را در سیستم کدگذاری کرده و سپس آرایه‌ای متشکل از امیدریاضی اندازه‌گیری تک‌تک کیوبیت‌های سیستم را به عنوان خروجی تولید می‌کند که نتیجه‌ی نهایی الگوریتم، از این خروجی ساخته می‌شود. نمایش دیداری این مدار در شکل ۴ـ۵ آمده است.

در شکل‌های این مدارها، زیرمدار encode_music با نام $EM(x)$، زیرمدار discriminator با نام $Disc(\theta)$ و زیرمدار generator با نام $Gen(\phi)$ مشخص شده است.

نکته‌ی اصلی این است که بعد از هربار اجرای مدارهای اول و دوم، وزن‌های زیرمدار discriminator آن‌ها با هم همگام می‌شوند، چراکه در کل برنامه باید تنها یک discriminator وجود داشته‌باشد. الگوریتم کلی اجرا شده در هنگام آموزش این ماژول، دارای چند حلقه است. در هر مرحله از حلقه‌ی اول که discriminator_iteration نام دارد، ابتدا با استفاده از الگوریتم گرادیان کاهشی، یک گام در بهینه‌سازی پارامترهای زیرمدار discriminator برداشته می‌شود، سپس پارامترهای این زیرمدار در دو مدار real_music_discriminator و generated_music_discriminator همگام‌سازی می‌شوند. در حلقه‌ی دوم که generator_iteration نام دارد، بهینه‌سازی پارامترهای زیرمدار music_generator انجام می‌شود و در هر مرحله از اجرای حلقه‌ی

الگوریتم ۶.۴ نحوه کارکرد کلی ماژول QuGAN

```
real_music_discriminator ← pre-generated quantum circuit
generated_music_discriminator ← pre-generated quantum circuit
steps ← number of iterations for the outer loop
n_iterations ← number of iterations for each inner loop
discriminator = real_music_discriminator.discriminator
generator = generated_music_discriminator.generator
for i = 0 → steps do
  for j = 0 → n_iterations do
    real_music_discriminator.optimize(discriminator)
    sync_disc(real_music_discriminator, generated_music_discriminator)
  end for
  for j = 0 → n_iterations do
    generated_music_discriminator.optimize(generator)
  end for
end for
```

سوم، ابتدا حلقه‌ی discriminator_iteration و سپس حلقه‌ی generator_iteration اجرا می‌شوند. فرم شبه‌کدی این الگوریتم در الگوریتم ۶.۴ آمده است که در این الگوریتم، تابع sync_disc تابعی‌ست که پارامترهای موجود در زیرمدارهای discriminator موجود در مدارهای real_music_discriminator و generated_music_discriminator را همگام می‌سازد. هم‌چنین شایان ذکر است که بهینه‌سازی پارامترها در این الگوریتم، با استفاده از توابع هزینه‌ی معرفی شده در معادله‌ی ۲۸.۲ انجام می‌گیرد.

لازم به ذکر است که در این ماژول، نویزی که به مدارهای generated_music_discriminator و final_music_generator داده می‌شود، تعدادی نت تصادفی از مجموعه داده است. پس از بهینه‌سازی پارامترهای زیرمدارهای discriminator و music_generator، پارامترهای زیرمدار music_generator موجود در مدار final_music_generator با مدار generated_music_discriminator همگام‌سازی شده و پس از پس‌پردازش خروجی‌هایی که از مدار final_music_generator تولید می‌شود، به مجموعه‌ای از نت‌های موسیقی دست می‌یابیم.

۴ـ۴ـ۱ پس‌پردازش

به همان دلیل ارائه شده در بخش ۴ـ۳ـ۵، داده‌های این ماژول نیز نیاز به پس‌پردازش دارد، اما این ماژول برای تولید قطعات موسیقی آهنگین‌تر، نیاز به پس‌پردازش متفاوتی دارد. پس‌پردازش استفاده شده در این ماژول طبق الگوریتم ۷.۴ عمل می‌کند. در این الگوریتم، همانند پس‌پردازش ماژول QLSTM، متغیرهای $input_i$ و $norm_i$ طبق معادله‌ی ۴.۴ تعریف شده‌اند، $raw\_model\_output_i$ خروجی اولیه‌ای است که از مدار کوانتومی گرفته می‌شود و $model\_output_i$ خروجی نهایی بعد از پس‌پردازش است. منظور از mapping نیز همان نگاشت یک به یک از نت‌ها به اعداد طبیعی است.

الگوریتم ۷.۴ پس‌پردازش ماژول QuGAN

$raw\_model\_output_i = model(input_i)$
$model\_output_i = (raw\_model\_output_i + 1) * norm_i$
$counter = 1$
while $model\_output_i \notin mapping.keys()$ **do**
  $model\_output_i = model\_output_i * counter/(counter + 1)$
  $model\_output_i = model\_output_i.to\_int()$
  $counter = counter + 1$
end while

در نهایت، تابع generate_notes با گرفتن پارامتر n_notes با چندین‌بار اجرای الگوریتم‌های یادگیری ماشین و پس‌پردازش، به تعداد n_notes نت موسیقی جدید تولید کرده و با قرار دادن فاصله‌هایی به اندازه‌ی پنجاه میلی‌ثانیه بین آن‌ها، نتایج حاصل را در یک فایل با پسوند midi ذخیره می‌کند.

فصل پنجم

نتیجه‌گیری

در این پروژه‌ی کارشناسی، سعی شد تا به بررسی محاسبات کوانتومی، یادگیری ماشین، تئوری موسیقی و تلاقی این سه حوزه برای تولید موسیقی‌های بدیع پرداخته شود. به نظر می‌رسد که تا به حال تلاش‌های کمی در راستای بررسی این تلاقی صورت گرفته است؛ با این وجود که به خاطر خواص موجی فیزیک کوانتومی و خواص موجی نت‌های موسیقی، احتمال داده می‌شود که پتانسیل‌های بسیار زیادی در این زمینه وجود داشته باشد. ماحصل این پروژه، برنامه‌ی نرم‌افزاری‌ای است که با استفاده از یک مجموعه‌داده و چندین کتاب‌خانه‌ی پرکاربرد، موسیقی‌های بدیعی تولید می‌کند.

یک نکته‌ی حائز اهمیت این است که یادگیری این الگوریتم‌ها به نسبت سریع است و لزوما به تعداد کیوبیت‌های زیاد یا مدارهای عمیق برای تولید قطعات موسیقی آهنگین احتیاجی ندارد. نکته حائز اهمیت دیگر این است که الگوریتم‌های پیاده‌سازی شده در این کتاب‌خانه، محدود به همین مجموعه داده نیستند و می‌توان با تزریق مجموعه داده‌ی دیگری (به عنوان مثال، اصوات متعلق به سازهای دیگر، همچون گیتار و درام) به آن، نتایج متفاوتی دریافت کرد.

در نهایت، کد این پروژه و ۴ نمونه موسیقی تولید شده با استفاده از آن را می‌توان در مخزن گیت‌هاب Maqneta [۱] مشاهده کرد.

۵ـ۱ کارهای آینده

همان‌طور که اشاره شد، هنوز جنبه‌های بسیاری از این حوزه کشف‌نشده باقی مانده‌اند. به عنوان مثال، الگوریتم متفاوتی [۴] برای شبکه‌های عصبی بازگشتی کوانتومی نیز ارائه شده‌است که در این پروژه، امکان تولید موسیقی با استفاده از این الگوریتم بررسی نشده است. هم‌چنین، الگوریتم‌های کلاسیکی برای تولید موسیقی با استفاده از یادگیری تقویتی[1] و یادگیری انتقالی[2] نیز وجود دارند و با توجه به این‌که نسخه‌ای کوانتومی از هر دوی این الگوریتم‌ها موجود است [۱۸] [۶] احتمالا بتوان با بررسی آن‌ها نیز به تولید موسیقی پرداخت.

در حال حاضر، فعالیت‌هایی در این حوزه در جریان است. به عنوان مثال، کنفرانس ISQCMC [۲۰] که در اواخر ماه نوامبر سال میلادی جاری برگزار خواهد شد، به برخی جنبه‌های دیگر تولید موسیقی با استفاده از محاسبات کوانتومی، همانند تولید موسیقی با استفاده از اتوماتاهای سلولی کوانتومی[3] خواهد پرداخت.

[1] Reinforcement learning

[2] Transfer learning

[3] Quantum cellular automata

منابع و مراجع

[1] Abedi, Erfan. Maqenta. https://github.com/theerfan/Maqenta, 2021. [Online; accessed 07-Oct-2021].

[2] Amidi, Afshine and Amidi, Shervine. Recurrent neural networks cheatsheet. https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks, 2019. [Online; accessed 02-Oct-2021].

[3] Arute, Frank et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, Oct 2019.

[4] Bausch, Johannes. Recurrent quantum neural networks. https://arxiv.org/abs/2006.14619, Jun 2020. [arXiv preprint].

[5] Chen, Samuel Yen-Chi, Yoo, Shinjae, and Fang, Yao-Lung L. Quantum long short-term memory. https://arxiv.org/abs/2009.01783, Sep 2020. [arXiv preprint].

[6] Dong, Daoyi, Chen, Chunlin, Li, Hanxiong, and Tarn, Tzyh-Jong. Quantum reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38(5):1207–1220, Oct 2008.

[7] Doornbusch, Paul. Computer sound synthesis in 1951: The music of CSIRAC. *Computer Music Journal*, 28(1):10–25, Apr 2004.

[8] Feynman, Richard P. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6):467–488, Jun 1982.

[9] Goodfellow, Ian J., Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron, and Bengio, Yoshua. Generative adversarial networks, Jun 2014.

[10] Google Quantum AI Team. Quantum walk. https://quantumai.google/cirq/tutorials/quantum_walks, 2021. [Online; accessed 02-Oct-2021].

[11] Helweg, Kim. Composing with quantum information: Aspects of quantum music in theory and practice. *Muzikologija*, pages 61–77, 2018.

[12] Hochreiter, Sepp and Schmidhuber, Jürgen. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, Nov 1997.

[13] Kempe, Julia. Quantum random walks: An introductory overview. *Contemporary Physics*, 44(4):307–327, Jul 2003.

[14] Kirke, Alexis. Programming gate-based hardware quantum computers for music. *Muzikologija*, pages 21–37, 2018.

[15] Lloyd, Seth and Weedbrook, Christian. Quantum generative adversarial learning. *Physical Review Letters*, 121(4), Jul 2018.

[16] Magenta Team. Gansynth. https://github.com/magenta/magenta/tree/main/magenta/models/gansynth, 2020. [Online; accessed 02-Sept-2021].

[17] Magenta Team. Melody RNN. https://github.com/magenta/magenta/tree/main/magenta/models/melody_rnn, 2020. [Online; accessed 02-Oct-2021].

[18] Mari, Andrea, Bromley, Thomas R., Izaac, Josh, Schuld, Maria, and Killoran, Nathan. Transfer learning in hybrid classical-quantum neural networks. *Quantum*, 4:340, Oct 2020.

[19] Miranda, Eduardo R. Quantum computer: Hello, music! https://arxiv.org/abs/2006.13849, Jun 2020. [arXiv preprint].

[20] Miranda, Eduardo R. and Coecke, Bob. 1st international symposium on quantum computing and musical creativity. https://iccmr-quantum.github.io/1st_isqcmc/, 2021. [Online; accessed 07-Oct-2021].

[21] Mitarai, K., Negoro, M., Kitagawa, M., and Fujii, K. Quantum circuit learning. *Physical Review A*, 98(3), Sep 2018.

[22] Putz, Volkmar and Svozil, Karl. Quantum music. *Soft Computing*, 21(6):1467–1471, Aug 2015.

[23] Vedral, V. et al. Quantum music. http://quantummusic.org, 2015. [Online; accessed 02-Oct-2021].

[24] Wang, Phil. This person does not exist. https://thispersondoesnotexist.com/, 2019. [Online; accessed 02-Oct-2021].

[25] Zoufal, Christa, Lucchi, Aurélien, and Woerner, Stefan. Quantum generative adversarial networks for learning and loading random distributions. *npj Quantum Information*, 5(1), Nov 2019.