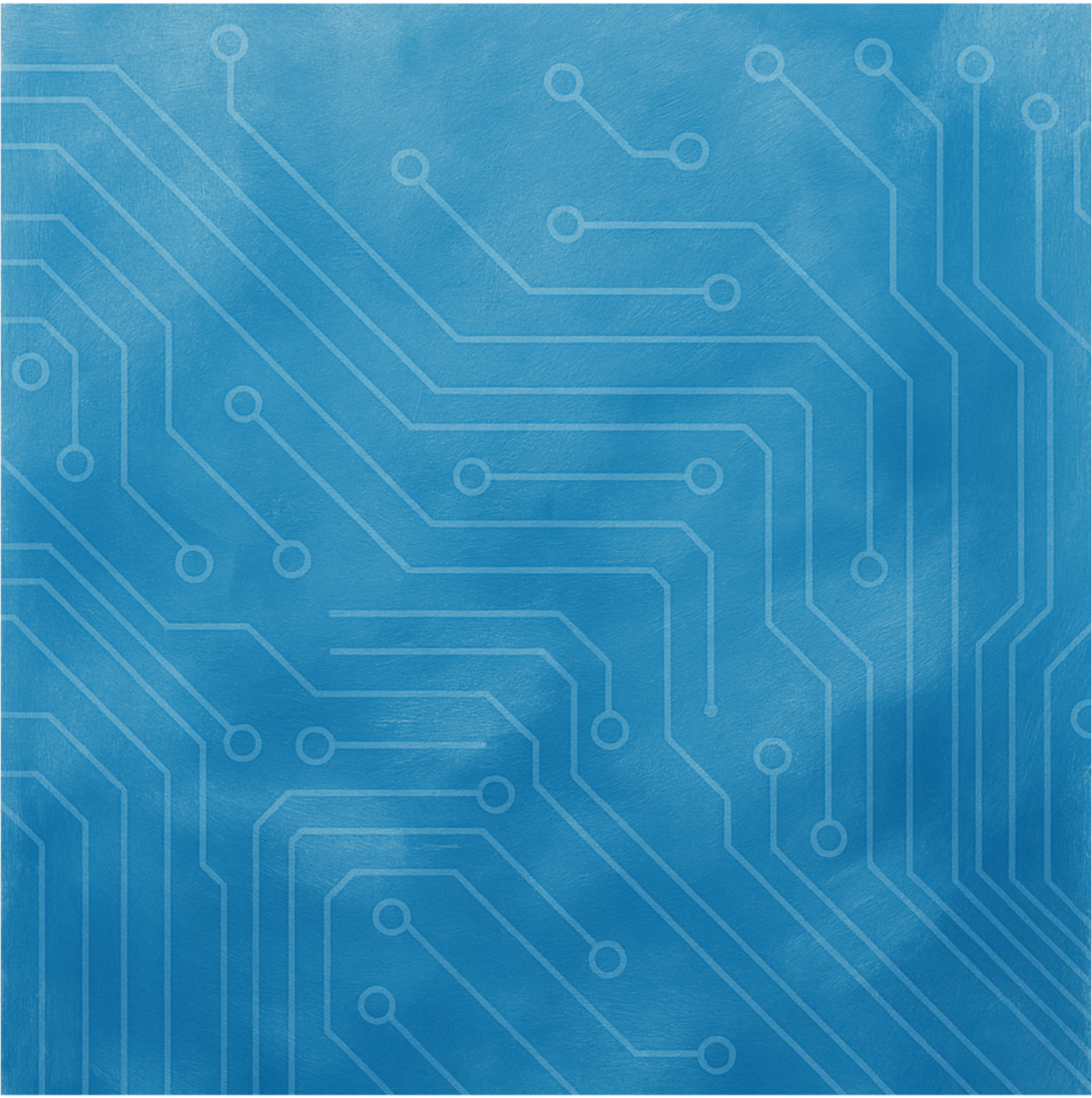
KaiRA Journal 2025 vol.9



Kyoto univ. AI Research Association

はじめに

本誌を手に取ってくださり、ありがとうございます。京都大学人工知能研究会 KaiRA 会長の岡本和優です。

人工知能という言葉は、いまや特別な響きをもたなくなりました。つい数年前—— GPT-2 がその安全性の懸念から一般公開すらされていなかった頃は、AI は研究室や巨大企業だけが扱う「特別な道具」でした。それが今では、インターネットさえつながっていれば、誰もが AI を使ったサービスを利用でき、日常の中に自然に溶け込むようになっています。

AI を「使う」ことについては確かに民主化が進みましたが、AI を「作る」「研究する」側の世界は、まだ十分に開かれているとは言えません。巨大企業だけが膨大な計算資源を用いてモデルを訓練し、研究の主導権はますます閉じた領域に集中しています。かつてオープンであると信じられた AI 研究の世界は、気づけばマネーゲームの世界へと収束しつつあります。

だからこそ、私たちのような小さなコミュニティが、自分たちの手で考え、試し、つくってみることに意味があります。シンギュラリティが来るかどうかといった大きな議論よりも、まずは自分たちの身近な問いから技術を生み出し、ほんの少しでも未来をよくする試みを続けていくこと——それこそが、AI 時代の創造性だと私は思っています。

大規模モデルが世界を塗り替えようとしている今でも、「自分たちで未来を形にしよう」とする学生たちの活動は決して色あせません。むしろ、この時代だからこそ、その小さな挑戦には大きな価値があるのではないでしょうか。

本誌には、学生が主体となって取り組んだ研究や開発の記録が収められています。そこに書かれているのは、華やかな成功ばかりではありません。しかし、そこで生まれた「試行錯誤の跡」こそが、この会誌の魅力です。

どんな問いを持ち、どう試して、どこでつまずき、何を得たのか。そのひとつひとつが、これからの技術や研究につながる小さな種だと思っています。

読者のみなさんには、ぜひこの 「挑戦のプロセス」 を楽しんでいただければ幸いです。巨大資本や巨大モデルが主導する時代だからこそ、学生が自分の興味や疑問から始めた小さな試みには、ほかにはない輝きがあると信じています。

京都大学人工知能研究会 KaiRA 会長
岡本和優
2025 年 11 月吉日

目次

第 1 章

ニューラルネットワーク入門

1.1　はじめに

近年の人工知能（AI）の目覚ましい発展は、まさに「ニューラルネットワーク」という技術によって牽引されてきました。スマートフォンが顔を認識したり、音声アシスタントが言葉の意味を理解したり、あるいは AI が美しい絵画や自然な文章を生成したり――私たちが日常で「すごい」と感じる AI の多くが、このニューラルネットワークを基盤として動いています。

とはいえ、「ニューラルネットワーク」と聞くと、難解な数式や複雑な理論を思い浮かべてしまう人も少なくないでしょう。

でも、安心してください。本稿の目的は、数学的な細部に立ち入ることではありません。私たちが共有したいのは、この技術の「どこが面白いのか」、そして「なぜここまで強力なのか」という「直感」です。

本稿は、「**AI という言葉は知っているが、体系的に学んだことはない**」という方に向けた、やさしい入門ガイドです。専門的な知識がなくても読み進められるよう、基本的な概念から丁寧に解説していきます。

1.2　生物の神経細胞から着想を得たモデル

ニューラルネットワークの最初のアイデアは、非常にシンプルで、私たちの身近にあるものから着想を得ています。それは、生物の脳に存在する**ニューロン（神経細胞）**です。

私たちの脳内では、ニューロンが複雑なネットワークを形成しています。個々のニューロンは、他のたくさんのニューロンから電気的な信号を受け取ります。そして、受け取った信号の合計が、ある一定の「しきい値（閾値）」を超えると、そのニューロンは「発火（興奮）」し、次のニューロンへと今度は自らが信号を送るのです。

この仕組みを、ごくごくシンプルに数学的なモデルとして置き換えてみましょう。

1. 複数の入力（信号）を受け取る。
2. それらを合計する。
3. 合計が「しきい値」を超えたら「1」（発火）を、超えなければ「0」（発火しない）を出力する。

たったこれだけです。これが、人工ニューラルネットワークの原点であり、出発点です。

ここで重要な直感が生まれます。私たちの「知能」や「記憶」は、決して一つの万能なニューロンが担っているわけではありません。むしろ、比較的単純な働きをする「多数の素朴な処理ユニット」

が、お互いに結びつき、チームとして連携することで、全体として非常に高度な知性が生まれるのではないか、という考えです。

この「チームワーク」こそが知性の本質である、という考えは、生物学的な事実とも一致します。例えば、あるサッカーチームが 1 年間練習して強力なチームワークを築いたとします。もし急にメンバーの半分がルールも知らない素人と入れ替わったら、そのチームはガタガタになってしまいます。脳のニューロンが頻繁に入れ替わらないのは、この「チームワーク」、すなわちニューロン同士の安定した接続パターンこそが「記憶」や「知能」そのものだからです。

ニューラルネットワークも同じです。個々の「処理ユニット（人工ニューロン）」は単純でも、その「接続の仕方」や「連携の強さ（チームワーク）」を学習させることで、大きな知能が生まれるのです。

1.3　パーセプトロン：最初のニューラルネットワーク

この「脳の神経細胞（ニューロン）」の仕組みを、世界で初めて工学的にモデル化したものが、1950 年代に考案された「パーセプトロン」です。これは、まさに AI の「ご先祖様」と呼べる存在です。

パーセプトロンの仕組みは、先ほどのニューロンのモデルとほぼ同じです。複数の入力に対し、それぞれ「重み」をかけて合計し、その値が「しきい値（あるいはバイアス）」を超えたら 1、そうでなければ 0 を出力します。

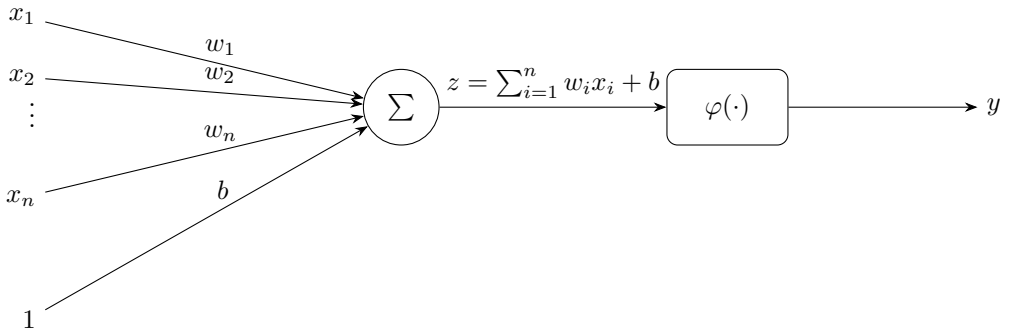


図 1.1: 単一パーセプトロンの模式図（重み w_i，バイアス b，活性化 φ）

ここで新しい言葉が 2 つ出てきました。「重み」と「しきい値（バイアス）」です。これらを簡単な例で考えてみましょう。

あなたが「今日は傘を持っていくべきか？」を判断する、シンプルなパーセプトロン（“傘判断” AI）になったと想像してください。

- **入力**：
  入力 1 x_1：「雨が降っているか？」（Yes=1, No=0）
  入力 2 x_2：「天気予報が雨か？」（Yes=1, No=0）
- **重み**：
  「どの入力を、どれだけ重視するか」をコントロールする値です。
  重み 1 w_1：「今、実際に降っている」（入力 1）は非常に重要なので、重み 5 に設定。
  重み 2 w_2：「予報」（入力 2）は、まぁ重要だが外れることもあるので、重み 2 に設定。
- **しきい値（バイアス）**：
  「どれくらいで”発火”（傘を持つ）と決めるか」のボーダーラインです。

あなたがとても慎重な人なら、しきい値を 1 に設定するかもしれません（予報が雨なだけで傘を持つ）。

あなたが楽観的な人なら、しきい値を 4 に設定するかもしれません（予報だけでは持たず、実際に降ってきたら持つ）。

この「重み」や「しきい値」をデータ（過去の経験）から自動で調整（学習）できるようにしたのが、パーセプトロンの画期的な点でした。

しかし、この初期のパーセプトロンには大きな限界がありました。それは、「線形的にしか境界を引けない」という問題です。

先ほどの例のように、入力の“足し算”で判断できるシンプルな問題（これは「線形分離可能」と呼ばれます）は解けます。しかし、世の中の問題はもっと複雑です。

例えば、「入力 1 と入力 2 の“どちらか一方だけ”が Yes の時に 1（発火）になる」という、有名な「XOR 問題」はこのパーセプトロンでは解けませんでした 。一本の直線では、この複雑な条件分けをどうやっても表現できなかったのです。

この“限界”の指摘は、当時のニューラルネット研究に大きな影響を与えました。もちろん、1960〜70 年代に訪れた「AI の冬の時代」は、パーセプトロンの限界だけが原因ではありません。むしろ、ルールベースの記号処理 AI が直面していた根本的な性能不足や計算資源の制約が、研究全体を停滞させた側面の方が大きかったと言われます。とはいえ、パーセプトロンでは XOR すら解けないという事実は、「より柔軟で表現力の高いモデルが必要だ」という強い問題意識を生み、のちの多層化——すなわち深層学習（ディープラーニング）へとつながる重要な伏線となりました。

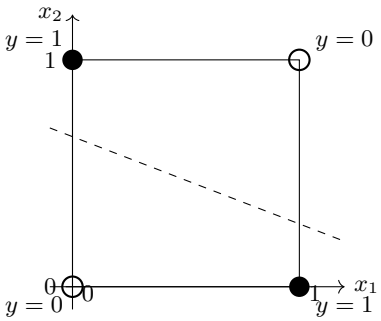


図 1.2: XOR の例：対角に異なるクラスが配置され、どの直線でも完全に分離できない（線形分離不可能）

1.4　多層パーセプトロン（MLP）と深層学習の基本

パーセプトロンの「XOR 問題」という壁。この限界を乗り越えるために生まれたのが、**多層パーセプトロン（MLP: Multi-Layer Perceptron）**です。

アイデアは、構造を「多層化」することでした。

パーセプトロンが「入力層」と「出力層」だけだったのに対し、MLP はその間に「**隠れ層（Hidden Layer）**」と呼ばれる新しい層を挿入しました。

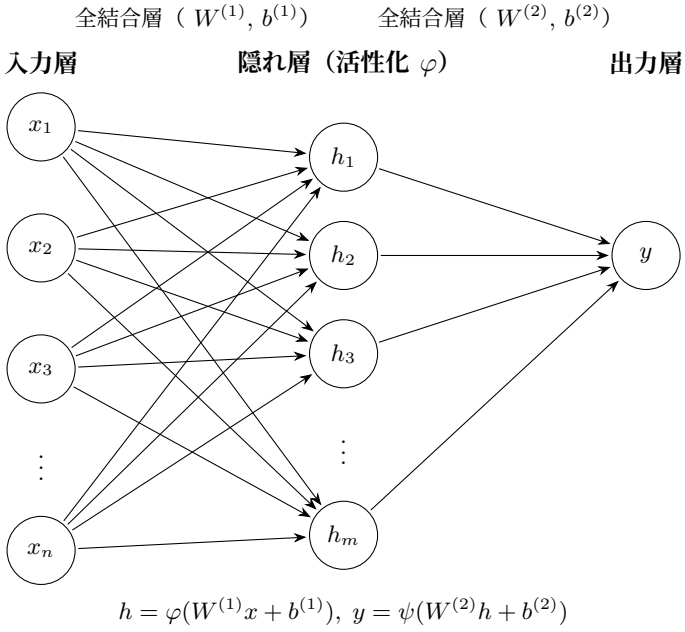


図 1.3: 多層パーセプトロン（MLP）の模式図。入力層→隠れ層（活性化 φ）→出力層。各層は全結合され、重みとバイアスが学習で更新される

この「隠れ層」が何を解決したのか。再び、例え話で考えてみましょう。

あなたは、画像に写っているのが「猫」か「猫でない」かを判定する AI（の出力層、つまり最終決定者）だとします。

まず、入力層から送られてくるのは何万個ものピクセルの色情報です。しかし、生のピクセル値だけを眺めても、「右上の点が茶色で、中央が黒で……」といった断片的な情報の寄せ集めにすぎず、そこから“猫”という概念を直接読み取るのは困難です。これは、パーセプトロン（隠れ層なし）が抱える「複雑なパターンを捉えられない」という限界そのものです。

そこで登場するのが、隠れ層という“専門家チーム”です。隠れ層の各ニューロンは、生のピクセルをそのまま扱うのではなく、自分の得意分野に応じて情報をまとめ上げる役割を担います。

- ニューロン A：「私は尖った耳の特徴を検出します」
- ニューロン B：「私はヒゲのパターンを見るのが得意です」
- ニューロン C：「私は丸い目に反応します」

こうして隠れ層は、生のピクセルからより抽象的で意味のある“特徴”を抽出し、「尖った耳：90 %」「ヒゲ：70 %」「丸い目：85 %」といった形に変換してくれます。あなた（出力層）は、もはや膨大なピクセル値を直接扱う必要はなく、こうした中間的な“猫らしさ”の指標を組み合わせて、最終的な判断ができるようになります。

これなら、あなた（出力層）の仕事は簡単です。「耳とヒゲと目のスコアが高いから、これは“猫”だ」と、はるかに賢い判断ができます。

これが、多層パーセプトロン（MLP）の核心です。

1. 層を深くする（隠れ層を増やす）ことで、より複雑なパターン（ピクセル→耳→猫）を表現できるようになりました。
2. 各層が異なるレベルの抽象化を担うことで、全体として強力な知能が生まれます。人間が「耳」

や「ヒゲ」といった”良い特徴量”を必死に設計しなくても、モデル（隠れ層）自身がデータから自動で”良い特徴”を学んでくれるようになりました。

3. そして、隠れ層のニューロン（専門家）が「発火」するかどうかを決める際、**活性化関数（Activation Function）**と呼ばれる仕組みが使われます。

特に重要なのが 3 番目の「活性化関数」です。もし隠れ層が、入力された情報をただ足し合わせて次の層に渡すだけ（線形）なら、層をどれだけ重ねても、結局は 1 枚のパーセプトロンと同じことしかできません。

そこで、各ニューロンは受け取った信号に対し、「非線形（カクンと曲がる）」な処理を加えます。現在、最も標準的に使われる **ReLU（レル）**という活性化関数は、驚くほどシンプルです。「入力がマイナスなら 0（発火しない）、プラスならそのまま通す」たったこれだけです。

この単純な「カクン」という非線形性が、ニューラルネットワークに複雑な境界線を引く能力を与え、XOR 問題をはじめとする現実の複雑な問題を解く力を与えたのです。

そして、この「隠れ層」を何層にも何層にも深く重ねて、より高度な特徴を学習させよう、というアイデアが、現在の**深層学習（ディープラーニング）**の基本構造となっています。

1.5　学習の仕組み：誤差逆伝播法

MLP という強力な「構造」を手に入れました。では、この構造（隠れ層の専門家チーム）は、どうやって「学習」するのでしょうか？　どうすれば、ピクセル情報から「耳」を見つけ出すような“優秀な専門家”に育ってくれるのでしょうか？

その答えが、ニューラルネットワークの学習を可能にする、最も中心的で美しいアイデア、「**誤差逆伝播法（Backpropagation）**」です。

ニューラルネットワークは、文字通り「間違いから学習する」仕組みを持っています。

先ほどの「猫」の例で、学習のプロセスを直感的に見てみましょう。

1. **予測（順伝播）**：
   まず、ネットワークに「猫」の画像を入力します。ネットワークは、現在の（まだ学習途中の）重みに基づいて、「入力層」→「隠れ層」→「出力層」へと信号を順に伝えながら予測を計算します。たとえば、AI が「犬 70%・猫 30%」と予測したとします。
2. **誤差の計算**：
   人間側は「正解は猫 100% だよ」と教えてあげます。AI は、自分の予測（猫 30%）と正解（猫 100%）を比較し、どれくらい間違っていたか＝**誤差**を計算します。「うわ、70% も外していた！」というイメージです。
3. **逆伝播（Backpropagation）**：
   計算された誤差は、今度は「出力層」から「隠れ層」へ、さらに「入力層」へと逆向きに伝わっていきます。このとき誤差は、「どのニューロンがどれだけ間違いに貢献したか」という“責任”の大きさに応じて分配されます。
   出力層（あなた）:「犬と判断したのは大間違いだった！」
   隠れ層（専門家）:「私の“垂れた耳”レポートがミスに大きく関与してしまった…」「私の“ヒゲ”レポートは、あまり関係なかったかも…」
4. **重みの調整（更新）**：
   間違いに強く関わった接続（重み）は、**次は少しでも誤差が減る方向**へと微調整されます。

“垂れた耳”と“犬”を結びつけていた重みは少し弱くなり、“尖った耳”と“猫”を結びつけていた重みは少し強くなります。

この 1〜4 のプロセスを、何万、何億という膨大なデータ（たくさんの猫や犬の画像）で繰り返します。この「間違い」と「修正」のサイクルこそが、ニューラルネットワークの「学習」です。

では、具体的に「どちらの方向に」「どれくらい」重みを調整すれば、間違い（誤差）が減るのでしょうか？

その調整方法を教えてくれるのが、**勾配降下法（Gradient Descent）**です。

これはよく、**“霧の中の山下り”**に例えられます。

- あなたは今、深い霧に包まれた広大な山の、どこかの斜面に立っています。
- あなたの「**現在の位置**」**は、ネットワークの「現在の重みの組み合わせ**」です。
- あなたの「**標高**」**は、ネットワークの「現在の誤差の大きさ**」です。
- あなたの「**目的**」**は、山の一番低い場所、「谷底（誤差が最小になる地点）**」にたどり着くことです。

霧が深くて谷底は見えません。さあ、どうしますか？

最も賢明な方法は、自分の足元を調べることです。地面の「**傾き（勾配）**」**を調べ、今いる場所で「最も急な下り坂」になっている方向を見つけます。そして、その方向に「ほんの一歩だけ」進みます。**

一歩進んだら、またその場で最も急な下り坂を探し、また一歩進む。これを何千回、何万回と繰り返すのです。そうすれば、いつかは谷底（最適な重み）にたどり着けるはずです。

こうして、勾配（傾き）を手がかりに誤差の情報を逆向きに伝えながら、重みという“現在地”を少しずつ更新していく——これが、誤差逆伝播と勾配降下法によってニューラルネットワークが学習する仕組みの核心です。

実際には、データ全体を使って一歩（バッチ勾配降下法）、あるいは少量のデータで素早く一歩（ミニバッチ勾配降下法、確率的勾配降下法）など、山の下り方にも様々な工夫があります。特に確率的勾配降下法（SGD）は、ノイズが多くて不安定な（ジグザグに進む）反面、巨大なデータセットでも高速に学習でき、局所的な谷（局所最適解）から脱出しやすいというメリットがあります。

1.6　おわりに

以上が、ニューラルネットワークの基本的な仕組みと学習方法の概要です。見てきたように、ニューラルネットワークは、単純な計算を行う多数のユニットが層をなし、そのつながりの強さ（重み）を調整しながら、少しずつ正解に近づいていく仕組みです。隠れ層は生のデータから意味のある特徴を抽出し、誤差逆伝播は「どの部分がどれだけ間違いに関わったのか」をネットワーク全体に伝え、勾配降下法は「次にどちらへ進めばよいか」を教えてくれます。

こうした素朴なステップの積み重ねが、やがて“猫”や“犬”といった複雑な概念さえ自動で学び取る力へとつながります。私たちが日常で目にする高度な AI の裏側には、この静かで着実な試行錯誤の連続が息づいているのです。

ただ、ここまで扱ってきたモデルには、ひとつ意外な前提があります。それは、**ニューラルネットワークは本来「決定論的」に動作する**という点です。同じ入力を与えれば、いつも同じ答えを返す——これは数学的には自然ですが、実世界のあり方とは少し違います。

というのも、私たちが相手にしている自然現象は、たいていの場合、**揺らぎ**や**不確実性**を内包して

います。観測値はノイズを含み、物理現象は確率的にばらつき、生命現象に至っては再現性そのものが確率論的です。世界は“ひとつの答え”を返すのではなく、むしろ“ゆらいだ分布”として姿を見せます。

その意味で、決定論的なニューラルネットワークだけでは、自然のふるまいを本質的に捉えきれないのではないか――そんな疑問も生まれてきます。

こうした問題意識から生まれたのが、**ベイズ的ニューラルネットワーク（Bayesian Neural Network; BNN）**という考え方です。BNN では重みを固定の値としてではなく「確率分布」として扱い、モデル自身が「どれくらい確信しているか」まで推論します。予測の幅や不確実性そのものを扱える点で、自然界の“ゆらぐ”構造により近い表現とも言えます。

本稿では詳細に立ち入りませんが、もしニューラルネットワークと自然現象の関係に興味があるなら、この「確率的な知性」のアプローチはきっと刺激的に映るはずです。決定論的なモデルの上に、どこまで“自然のゆらぎ”を重ねられるのか――その先には、AI が世界をどう捉えうるのかという、静かな問いが立ち上がります。

第 2 章

画像生成入門

2.1　はじめに

本章では、画像生成の仕組みについて解説します。高校数学の知識があれば理解できる範囲にとどめており、その上で AI を学ぶ楽しさを感じていただくことを目的としています。

AI を学ぶ醍醐味は大きく 2 つあります。1 つ目が、広く利用されているものの、仕組みが不透明な AI について学び、仕組みの不透明さを解消することで、AI が得体の知れないものから身近で理解可能な技術として捉えられるようになる点です。2 つ目が、古典的なモデルから最新のモデルを順に学ぶことで、研究者たちの工夫やそれによる性能向上の歴史を追体験するような楽しみがある点です。

本章では、まず第 2 節でニューラルネットワークを画像データに適用できるように拡張した畳み込みニューラルネットワーク（CNN）について解説します。続いて、第 3 節では最も基本的な画像生成モデルであるオートエンコーダーを取り上げ、第 4 節および第 5 節では現在主流となっている拡散モデルおよび潜在拡散モデルについて解説します。

2.2　畳み込みニューラルネットワーク（CNN）

2.2.1　畳み込みニューラルネットワークの概要

前のページで紹介したニューラルネットワークでは、入力データを 1 次元配列として扱っていました。このような構造では、縦と横の空間的なつながりを持つ 2 次元画像データをうまく処理できません。そのため、画像の局所的特徴を捉えることができる**畳み込みニューラルネットワーク**（CNN）が使用されます。代表的なタスクとしては、0 から 9 までの手書き数字画像を識別するもの（MNIST **データセット**）があります。図 2.1 は、MNIST データセットの例です。

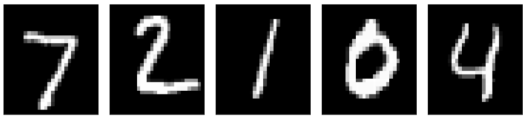


図 2.1: MNIST データセット 出典：https://arxiv.org/pdf/2201.03898

畳み込みニューラルネットワークの起源は、1979 年に計算機科学者の福島邦彦によって提唱されたネオコグニトロンにさかのぼります。これは、生物の脳の視覚野における情報処理の仕組みをモデル化したもので、現在の CNN の原型となりました。

2.2.2 画像データの扱い方

画像データは、各画素（ピクセル）の色を 1 つの要素とする 2 次元配列として扱われます。1 画素の色は、光の三原色である赤（Red）、緑（Green）、青（Blue）の 3 成分で表現され、それぞれの成分の強さを 0 から 1 の範囲の実数値で表します。したがって、画像の解像度が縦 n ピクセル、横 m ピクセルの場合、画像データは $n \times m \times 3$ の配列として表され、各要素は 0 から 1 の値をとるデータとして扱われます。画像データや、それに畳み込み処理を施して得られる 2 次元配列は、**特徴マップ**と呼ばれます。

2.2.3 畳み込みニューラルネットワーク

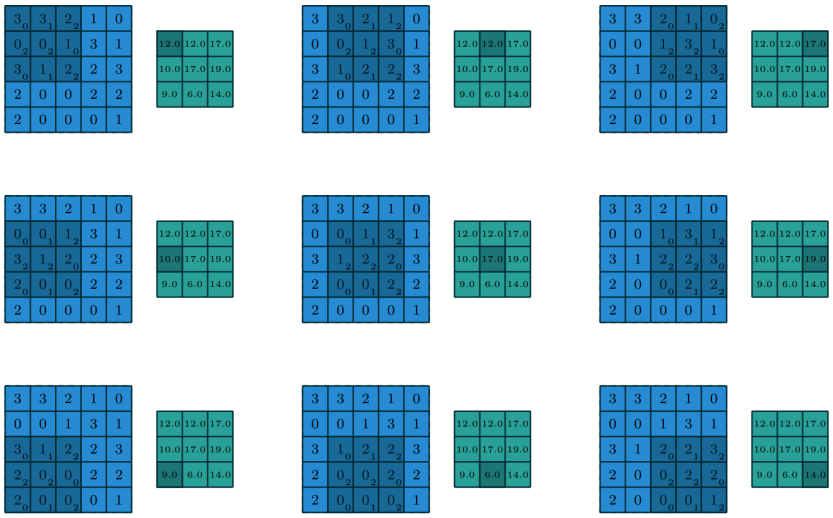


図 2.2: CNN の演算の様子 出典：https://arxiv.org/pdf/1603.07285

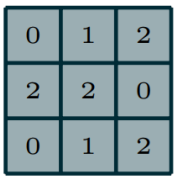


図 2.3: 図 2.1 に適用したフィルタ 出典：https://arxiv.org/pdf/1603.07285

図 2.2 は、特徴マップに対して図 2.3 のフィルタを適用する様子を示します。具体的には、特徴マップの一部（図 2.2 では 3×3 の領域）の左上から順にフィルタを重ね、フィルタの各要素と特徴マップの対応する要素を乗算し、それらの積の総和を求めるという処理を行います。図 2.2 の左上における計算は、次のように表されます。

$$3 \times 0 + 3 \times 1 + 2 \times 2 + 0 \times 2 + 0 \times 2 + 1 \times 0 + 3 \times 0 + 1 \times 1 + 2 \times 2 = 12$$

それぞれの特徴マップの一部にフィルタを適用するとき、フィルタを移動させる際の間隔（ピクセル数）を**ストライド** (Stride) と呼びます。図 2.2 の処理ではストライドは 1 となり、ストライドを 2

に設定すると、フィルタが 2 ピクセルずつ移動します。その結果、出力される特徴マップの縦横のサイズは約 1/2 になります。

このようにして、フィルタを画像全体に順に適用し、各位置で得られた値を並べることで、新しい特徴マップが得られます。

また、実際の画像データは、光の三原色である赤（Red）、緑（Green）、青（Blue）の各成分を表す 2 次元配列が 3 枚重なった構造をしています。これら 3 枚の画像データそれぞれに対して、複数種類のフィルタを適用することで、より多様な特徴を抽出することが可能です。図 2.4 は、3 種類のフィルタを適用する様子を示しています。

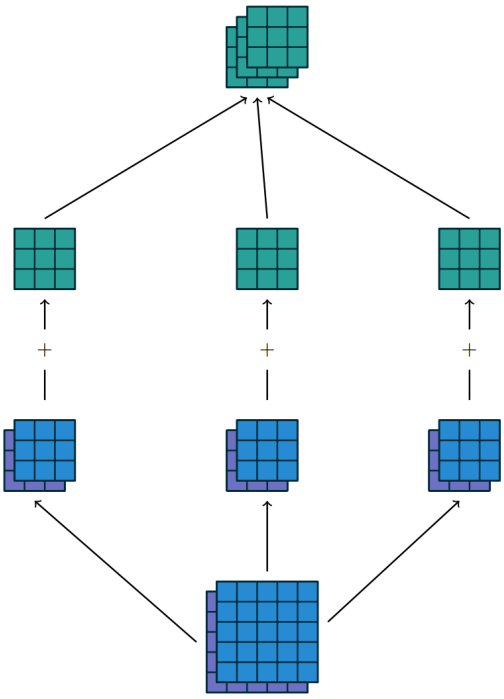


図 2.4: 複数種類のフィルタを適用 出典：https://arxiv.org/pdf/1603.07285

最後に、ReLU などの活性化関数を通すことで、ニューラルネットワークの 1 層が構成されます。このような層を繰り返し重ねることで、空間的に縮小されたより抽象的な特徴マップ（出力画像）が得られます。

2.2.4　転置畳み込み

ストライドが 2 の畳み込み処理を適用する場合は、特徴マップの縦横のサイズは約 1/2 になるのに対して、図 2.5 のように転置畳み込みという処理を適用すると、特徴マップは結果的に縦横のサイズは 2 倍になります。

具体的には、図 2.5 のように、3×3 の特徴マップの各ピクセルの間に値が 0 のピクセルを追加します。このように、特定の値（通常は 0）のピクセルを挿入する処理を**パディング** (Padding) と呼びます。パディングを施した特徴マップに対して、ストライドを 1 に設定した畳み込み処理を行うと、最終的に 6×6 の特徴マップが得られます。

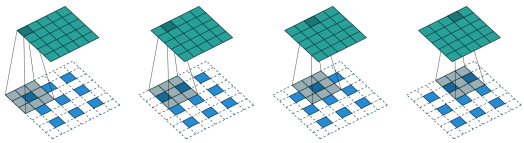


図 2.5: 転置畳み込み 出典：https://arxiv.org/pdf/1603.07285

2.3　オートエンコーダー

2.3.1　オートエンコーダーの概要

この節から、実際に画像を生成するモデルの解説に入ります。第 3 節、第 4 節、第 5 節で解説する 3 つのモデルに共通する点は、画像生成とは乱数から一定の処理を加えて画像を得る過程であるということです。この考え方を最も単純な形で実現している基本モデルが、**オートエンコーダー** (Autoencoder) です。

オートエンコーダーは、入力画像から圧縮された特徴量を抽出する**エンコーダー** (Encoder) と、その特徴量から画像を復元する**デコーダー** (Decoder) によって構成されます。エンコーダーには畳み込み処理が、デコーダーには転置畳み込み処理が用いられています。

オートエンコーダーは、2006 年にコンピュータ科学および認知心理学の研究者であるジェフリー・ヒントンによって提唱されました。ニューラルネットワークを用いた次元圧縮のための手法として開発されたものです。

2.3.2　オートエンコーダーの構造・学習

エンコーダーとデコーダーから構成されるオートエンコーダーの構造は、図 2.6 のようになります。まず、エンコーダーで畳み込み処理を行い、入力画像の次元を圧縮します。このエンコーダーによって圧縮された特徴マップは**潜在空間** (Latent Space) と呼ばれます。その後、デコーダーで転置畳み込み処理を行い、画像を復元します。学習は、元の画像と復元画像との差（誤差）が小さくなるように行われます。

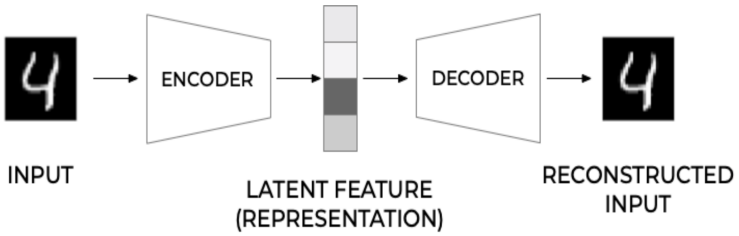


図 2.6: オートエンコーダーの構造 出典：https://arxiv.org/pdf/2201.03898

MNIST データセット（手書き文字画像）のうち、0・1・2 のデータのみを対象として、潜在空間を 2 次元に設定したオートエンコーダーを学習させたときの結果を示します。図 2.7 は、各データが潜在空間上でどのように分布しているかを表しています。紫色の点の集まりは「0」に対応するデータ、緑色の点の集まりは「1」に対応するデータ、黄色の点の集まりは「2」に対応するデータを示し

ています。

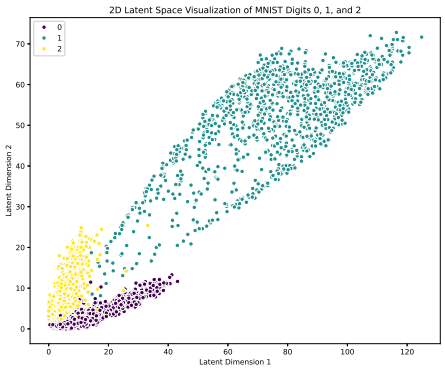


図 2.7: MNIST データセットの 0、1、2 の潜在空間における分布

2.3.3　オートエンコーダーの生成

一般に、画像生成では乱数を入力として処理を行い、画像を生成します。オートエンコーダーにおいては、潜在空間からランダムな値をサンプリングし、それをデコーダーに通すことで画像を生成します。

ここでは、図 2.7 の結果を基に、意図的に特定の手書き数字を生成できるかを検証します。ここで生成したい数字を「2」とすると、潜在空間内で「2」に対応するデータは、次元 1 が 0 から 10、次元 2 が 0 から 25 の範囲内に分布しています。仮に、次元 1 の値を 5、次元 2 の値を 10 とする潜在空間上の点を設定し、この点をデコーダーに通して画像を生成します。その結果、図 2.8 に示すように、手書き文字の「2」が生成されていることが確認できました。

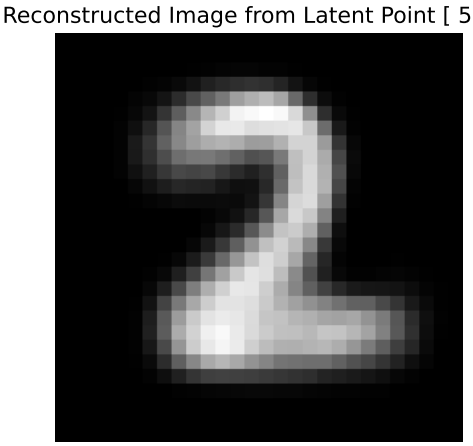


図 2.8: 潜在空間 (5,10) の点から生成した画像

2.4　ノイズ除去拡散モデル

2.4.1　ノイズ除去拡散モデルの概要

この節からは、現在の画像生成技術の核となる拡散モデルについて解説します。拡散モデルは、2015 年に機械学習および人工知能の研究者であるジャシャ・ソール＝ディックスタインによって考案されたもので、非平衡熱力学の概念から着想を得ています。本節では、その中でも特に**ノイズ除去拡散モデル** (Denoising Diffusion Probabilistic Models) に焦点を当てて紹介します。

ノイズ除去拡散モデルの特徴は、オートエンコーダーのように一度の処理で画像を生成するのではなく、図 2.9 のように各ピクセルがランダムな値で構成された砂嵐（ノイズ画像）のような状態から、繰り返しノイズを除去する処理を行い、最終的に高品質な画像を生成する点にあります。

図 2.9: 拡散モデルの画像生成過程例出典：https://arxiv.org/pdf/2006.11239

2.4.2　ノイズ除去拡散モデルの学習・画像生成

まず、ノイズ除去拡散モデルを学習させるためのデータを作成します。ある画像に対して、段階的にわずかなノイズを加えていきます。この操作を繰り返すことで、最終的には各ピクセルが完全にランダムな値で構成された画像に近づいていきます。これは、図 2.10 における右から左への処理に対応します。これを**順方向の拡散過程**と呼びます。

この過程で得られる中間段階の画像と、それぞれの画像に加えられたノイズの情報を学習に利用します。学習の目的は、前述のとおり画像からノイズを除去することです。したがって、ノイズのある画像から加えられたノイズを予測し、その予測値を差し引くことで、ノイズを除去した画像を得ることができます。この処理は、図 2.10 における左から右への処理に対応し、画像にノイズを加える過程の逆操作を再現しているといえます。これを**逆方向の拡散過程**と呼びます。画像生成は、各ピクセルがランダムな値で構成された画像から始まり、前述のノイズ除去処理を繰り返すことで、最終的に高品質な画像が生成されます。

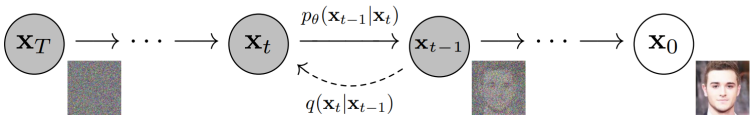


Figure 2: The directed graphical model considered in this work.

図 2.10: ノイズの付加と除去の過程 出典：https://arxiv.org/pdf/2006.11239

2.4.3　ノイズ除去拡散モデルの構造

それでは、画像に加えられたノイズをどのように予測するのかについて解説します。この問題をより一般的な観点から見ると、入力画像（ノイズを含む画像）と、予測すべき出力（ノイズ）が与えられていることになります。したがって、このタスクの構造は、オートエンコーダーにおける画像の圧縮と復元の過程に類似しています。

異なる点は、予測の対象がオートエンコーダーでは元の画像であるのに対し、ノイズ除去拡散モデルではノイズそのものであるという点です。したがって、基本的な構造はオートエンコーダーと同様に設計することが可能ですが、ノイズ除去拡散モデルでは、より高い精度を実現するために U-Net と呼ばれる構造が用いられます。

U-Net の構造は、オートエンコーダーと同様に、畳み込み処理によって特徴マップを圧縮するエンコーダーと、転置畳み込み処理によって特徴マップを拡大するデコーダーから構成されます。U-Net の特徴的な構造として、**残差接続**があります。

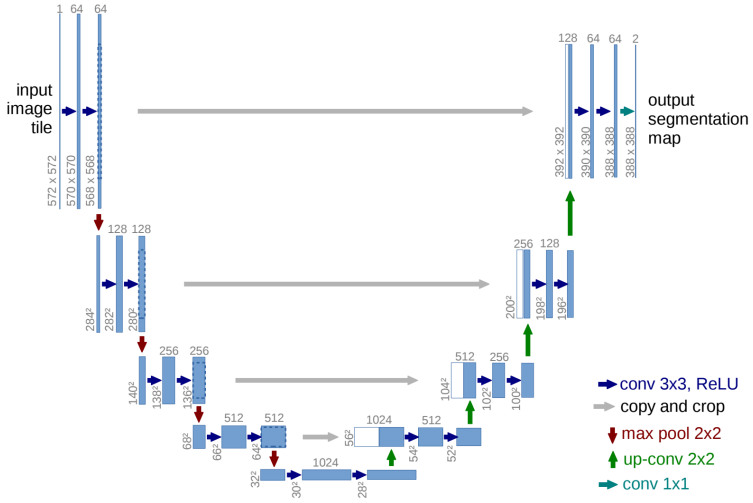


図 2.11: U-Net の構造 出典：https://arxiv.org/pdf/1505.04597

残差接続とは、図 2.11 に示すように、エンコーダー側の各段階で得られる特徴マップを、同じサイズのデコーダー側の特徴マップと積み重ね、その後、図 2.4 のように複数種類のフィルタを適用して畳み込み処理を行う構造です。この構造により、エンコーダーの途中で得られた空間的特徴情報がデコーダーに橋渡しされ、画像の位置関係などの情報を保持したまま、より高精度な予測が可能となります。

2.5　潜在拡散モデル

2.5.1　潜在拡散モデルの概要

この節では、イギリスの企業 Stability AI によって一般公開された画像生成モデル Stable Diffusion の中核となる**潜在拡散モデル**（Latent Diffusion Model）について解説します。Stable Diffusion は 2022 年 10 月に公開され、高価な専用機器を必要とせず一般的なパソコンでも実行できたことから、AI による画像生成が広く普及する契機となりました。

このモデルは、テキストによる指示に従って画像を生成する text-to-image **モデル**であり、現在一般的に想起される画像生成の代表的な形態です。Stable Diffusion は、ノイズ除去拡散モデルを基盤としつつ、さまざまな改良を加えることで、テキストの指示に忠実で高品質な画像生成を実現しています。

ここでは、その改良点のうち最も精度向上に寄与している**潜在拡散**（Latent Diffusion）について解説します。潜在拡散モデルでは、オートエンコーダーのように一度画像を圧縮して潜在空間を得た後、その潜在空間上でノイズ除去拡散モデルと同様のノイズ除去処理を行うことで、画像を生成します。

2.5.2　潜在拡散モデルの構造・学習・生成

以前のノイズ除去拡散モデルでは、画像の画素空間（ピクセル空間）上でノイズ除去を行っていたため、高解像度の画像を処理するには膨大な**計算量**が必要となります。ここでいう計算量とは、画像生成の際に必要となる加算や乗算などの演算回数を指します。計算量を削減しつつ、画像生成の表現力を両立している点が、潜在拡散モデルの大きな特徴です。

具体的には、まずオートエンコーダーと同様に、画像を圧縮および復元するように学習を行います。第 3 節で紹介したオートエンコーダーでは 2 次元に圧縮しましたが、潜在拡散モデルでは、より高次元である 32×32 のような低解像度の特徴マップとして圧縮します。

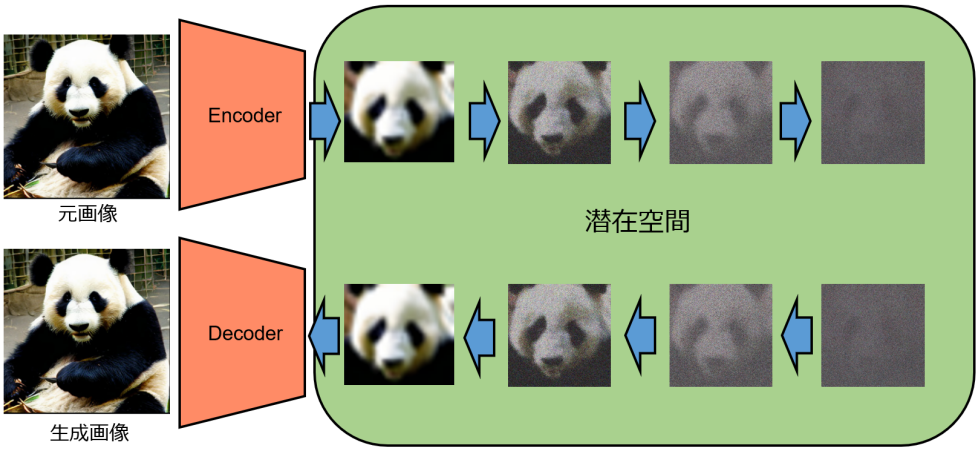


図 2.12: 潜在拡散モデル 画像：https://arxiv.org/pdf/2112.10752

図 2.12 に示すように、学習済みのオートエンコーダーに実際の画像を入力し、潜在空間上の特徴マップを得ます。この特徴マップに対して、ノイズ除去拡散モデルと同様に、段階的にノイズを加える処理（順方向の拡散過程）を行います。その後、ノイズが加えられた特徴マップからノイズを予測し、除去するように学習します（逆方向の拡散過程）。

最後に、画像を生成する際には、各ピクセルがランダムな値で構成された特徴マップから始め、前述のノイズ除去処理を繰り返した後、デコーダーを用いて最終的な画像を復元します。

第 3 章

事前学習済み Transformer を用いた化学反応収率のベイズ最適化

3.1　はじめに

ある日、あなたは研究室のボスから次のように言われました。「このリストの中から、最も収率のよい反応条件を探してくれ」と。

リストに載っている反応の数が 10 件程度であれば、すべてを試してみることも現実的かもしれません。しかし、それが数百件、数千件にも及ぶとなると話は別です。

化学反応には、温度・溶媒・触媒など、数多くの条件が関わります。それらの組み合わせをすべて網羅的に試すことは現実的ではありません。そのため研究者は、限られた実験回数の中で、見込みのありそうな条件や未知の可能性を秘めた条件を優先的に試す必要があります。

こうした「次にどの実験を行うべきか」を効率的に決定するための手法のひとつが、**ベイズ最適化（Bayesian Optimization）**です。

3.2　ベイズ最適化とは

機械学習に親しみのある方であれば、ベイズ最適化をハイパーパラメータチューニングの手法として耳にしたことがあるかもしれません。ベイズ最適化とは、**評価にコストがかかるブラックボックス関数を、できるだけ少ない試行で最適化する手法**です。

ブラックボックス関数とは、入力に対して出力を得ることはできるものの、その内部構造や関数形が不明な関数を指します。例えば化学反応の収率は、温度・溶媒・触媒など多様な条件に依存しますが、その関係は極めて複雑で、明示的な数式で表すことは困難です。

ベイズ最適化は、このようなブラックボックス関数の最適化において、少ない試行回数で最適解を探索するために用いられます。その流れは主に次の 2 つのステップから構成されます。

1. **サロゲートモデルの構築**
   まず、既存のデータをもとにブラックボックス関数を近似する「サロゲートモデル」を構築します。一般的には、**ガウス過程回帰（Gaussian Process Regression, GPR）**が用いられます。GPR は入力空間における関数の振る舞いを確率的にモデル化し、予測値とその不確実性を同時に推定することが可能です。予測値に加えて不確実性も考慮することで、未知領域を探索する指針を得ることができます。
2. **獲得関数の最適化**

次に、サロゲートモデルの予測結果をもとに「次にどの点を評価すべきか」を決定します。その判断に用いられるのが**獲得関数（Acquisition Function）**です。獲得関数は、予測値と不確実性の両方を考慮しながら、探索（exploration）と活用（exploitation）のバランスをとって次の候補点を選びます。代表的な獲得関数には、**EI（Expected Improvement）**、**PI（Probability of Improvement）**、**UCB（Upper Confidence Bound）**などがあります。

この 2 つのステップを繰り返すことで、ベイズ最適化は徐々に最適解に近づいていきます。特に、評価コストが高い場合や探索空間が広い場合において、その効率性が際立ちます。実験コストが高く探索空間が膨大な**化学反応条件の最適化**は、まさにベイズ最適化が最も有効に機能する領域の一つです。

3.3 従来のベイズ最適化の課題

ベイズ最適化は少ない試行で最適解を見つける強力な手法ですが、その性能はサロゲートモデルの精度に大きく依存します。一般的なベイズ最適化では、探索開始時にはデータが少なく、サロゲートモデルは初期のわずかなデータから構築されます。

そのため、探索初期においては予測精度が低く、効率的な探索が難しいという問題があります。また、サロゲートモデルは通常、各タスクごとに一から学習を行うため、過去の実験データや他の反応系で得られた知識を活用できません。

一方、化学反応の分野には既に膨大な反応データや論文情報が存在します。にもかかわらず、従来のベイズ最適化ではそれらを活用せず、「毎回ゼロから探索を始める」という非効率な枠組みにとどまっています。

3.4 事前学習済みモデルをサロゲートとして活用する

従来のベイズ最適化では、サロゲートモデルは都度学習されるのが一般的です。しかし、近年の機械学習分野——特に自然言語処理や画像認識——では、**事前学習済みモデル（pretrained model）**の活用が標準的なアプローチとなっています。大規模データから汎用的な表現を学習し、それを個別タスクに転移することで、データが少ない段階から高い性能を発揮できることが知られています。

一方、ベイズ最適化の分野では、こうした「事前知識をもつサロゲートモデル」を導入した研究はまだ限られています。多くの手法は、対象ごとにデータを収集し、その場で一からモデルを構築するという前提に立っており、初期段階では予測性能が十分でないのが現状です。

化学反応の最適化の文脈では、すでに多くの反応データベースや文献情報が整備されています。これらを活用しないのは大きな機会損失といえます。そこで本研究では、**事前学習済みの反応予測モデル**をベイズ最適化のサロゲートとして活用する手法を検討しました。

その一例として、今回は Transformer アーキテクチャに基づく収率予測モデルである **ReactionT5**[1] を用いました。ReactionT5 は約 18 万件の反応データを学習しており、化学反応の収率を高精度に予測する能力を持ちます。また、ファインチューニングにより特定の反応系に適応させることも可能です。これをサロゲートモデルとして利用することで、未知の反応条件に対しても初期段階から一定の予測精度を確保し、より効率的な条件探索が可能になります。

3.5　ReactionT5 を用いたベイズ最適化の枠組み

全体の流れは図 3.1 に示すように、従来のベイズ最適化と同様の構造を持ちます。ただし、ReactionT5 のような Transformer ベースの事前学習済みモデルを含め、一般的なニューラルネットワークは推論時に確率的な振る舞いを持たないため、そのままでは不確実性の推定を行うことができません。したがって、これを可能にするためにはいくつかの工夫が必要となります。その中で最も簡単な手法の一つが、**MC Dropout**[2] による不確実性推定です。

MC Dropout では、推論時にも Dropout を有効化し、複数回の推論を行うことで予測の分布を得ます。この分布の分散を不確実性の尺度として扱うことで、近似的にベイズ推定を実現します。Dropout さえ含まれていれば、既存のニューラルネットワークに容易に導入できる点も利点です。本研究では、ReactionT5 に MC Dropout を適用して不確実性を推定し、ベイズ最適化のサロゲートモデルとして活用しています。

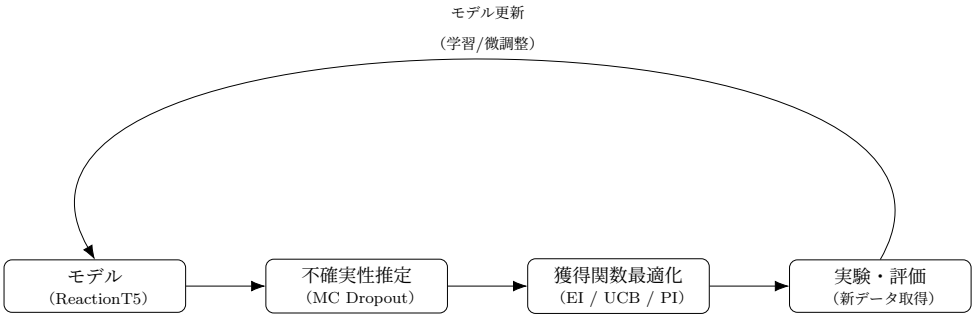


図 3.1: ReactionT5 を用いたベイズ最適化の全体フロー

3.6　実験設計

ベイズ最適化の有効性を評価するためには、未知の実験空間において、限られた試行回数で最適条件を効率的に探索できるかどうかを確認する必要があります。そのため本研究では、ReactionT5 の学習に使用されていない反応データセットを用いて検証を行いました。具体的には、表 3.1 に示す 3 つのデータセットを使用しました。

Suzuki-Miyaura および Buchwald-Hartwig のデータセットについては、元のデータから「完全格子」と呼ばれる、すべての条件組み合わせが網羅された部分のみを抽出して利用しました。これにより、探索空間が明確に定義された状態で、ベイズ最適化がどの程度効率的に高収率条件を見つけられるかを評価できるようにしました。

表 3.1: 使用データセットと条件の組み合わせ

データセット名	組み合わせ
NiB	33 substrates × 23 ligands × 2 solvents = 1518 entries
Suzuki-Miyaura	4 reactant_1 × 3 reactant_2 × 1 catalyst × 11 ligands × 7 reagents × 4 solvents = 3696 entries
Buchwald-Hartwig	2 ligands × 22 additives × 3 bases × 15 aryl halides = 1980 entries

事前学習済みモデルとして、ReactionT5 の収率予測モデルを使用しました。このモデルに対して MC Dropout を適用することで、予測値に加えてその不確実性も推定できるようにしました。MC Dropout の設定としては、ドロップアウト率をデフォルトの 0.1 に設定し、推論時に 10 回のサンプリングを行いました。

ファインチューニングは、探索を 10 回行うごとに実施し、それまでに得られた実験データを用いてモデルを更新しました。得られたデータが 25 件以上となった場合には、学習データとテストデータを 8:2 の割合で分割し、テストデータにおける予測性能を確認しました。ファインチューニングの設定は以下の通りです。

- エポック数: 2
- 訓練バッチサイズ: 8
- 評価バッチサイズ: 16
- 学習率: 1×10^{-4}
- 最適化手法: AdamW
- Weight Decay: 1×10^{-2}

ベイズ最適化の獲得関数としては、Expected Improvement（EI）を採用しました。通常のベイズ最適化では初期点をランダムに選択しますが、本研究では事前学習済みモデルを利用しているため、初期点も ReactionT5 の予測結果に基づいて選択しました。

また、比較対象として、Optuna の TPE（Tree-structured Parzen Estimator）Sampler を用いた手法と、RDKit から取得した Morgan Fingerprint を特徴量に用いたガウス過程回帰（GPR）ベースの手法を実装しました。

ベイズ最適化は確率的な手法であるため、それぞれの手法とデータセットに対して、シード値を変えて 5 回ずつ実験を行いました。

3.7 結果

NiB、Suzuki-Miyaura、および Buchwald-Hartwig の各データセットに対するベイズ最適化の結果を図 3.2 に示します。各グラフでは、横軸に試行回数、縦軸にこれまでに得られた最高収率をプロットしています。

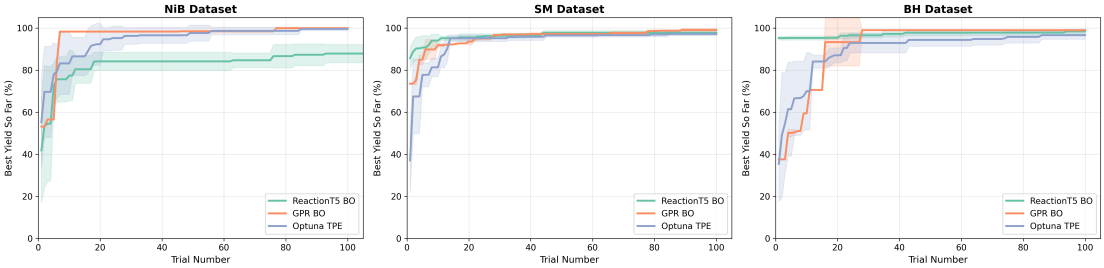


図 3.2: 各データセットにおけるベイズ最適化の結果

図 3.2 から、ReactionT5 を用いたベイズ最適化が必ずしもすべてのデータセットで最良の性能を示すわけではないことがわかります。NiB データセットでは、ReactionT5 ベースの手法は他の手法と比較して収束が遅く、最終的な最高収率も低い傾向を示しました。一方、Suzuki-Miyaura および Buchwald-Hartwig データセットにおいては、ReactionT5 ベースの手法が他の手法を上回る性能を示し、より効率的に高収率条件を探索できていることが確認されます。

3.8　考察

図 3.2 の結果から、ReactionT5 をサロゲートモデルとして用いたベイズ最適化は、従来手法と比較してデータセットによって挙動が異なることが確認されました。

まず、NiB データセットにおいては、ReactionT5 ベースの手法は他の手法に比べて収束が遅く、最終的な最高収率も低い傾向を示しました。この要因として、NiB データセットが他のデータセットに比べて化学構造や反応条件の多様性が高く、ReactionT5 の事前学習時に十分に類似する反応が含まれていなかった可能性が考えられます。すなわち、事前学習済みモデルに内在する「化学的事前知識」が、必ずしも未知の反応空間に対して有効に転移しない場合があることを示唆しています。

一方、Suzuki-Miyaura および Buchwald-Hartwig データセットでは、ReactionT5 ベースの手法が GPR ベースおよび Optuna TPE を上回る性能を示しました。これらのデータセットは、比較的均質な反応系から構成されており、事前学習済みモデルが学習している反応パターンと親和性が高かったと考えられます。その結果、初期段階から高い予測精度を発揮し、少ない試行回数で高収率条件を見出すことができたと解釈できます。

総じて、本手法は「事前知識を有するサロゲートモデル」を導入することで、特定の反応系における探索効率を改善できる可能性を示しました。一方で、NiB のように事前知識と対象系の分布が大きく異なる場合には、事前学習のバイアスが探索効率を損なうリスクもあるため、モデルの適用範囲や、事前学習に用いるデータの選定方針が今後の課題として挙げられます。

3.9　おわりに

最後に、より大きな展望として「人間と道具」の関係について考えてみたいと思います。この問題を考える上で外せない哲学者の一人に、ハイデガーがいます。ハイデガーは、道具を含むあらゆる存在を「目的のために作られ、使われるもの」としての道具的連関から切り離し、存在そのもののあり方として捉え直そうとしました。この試みは、道具や自然を単なる客体的な利用の対象から解放し、人間と世界との関わりそのものを見つめ直そうとするものでした。しかしその過程で、人間と道具が協働し、「目的のために共に何かをなす」という、本来あったはずの目的的な関係までもが、見えに

くくなってしまったように思えます。

たとえば、職人にとっての道具は、単に同じ機能を備えていればよいというものではありません。長年使い込まれた道具は、職人の身体の一部のように馴染み、手の感覚や動きと一体化して初めて、最高の成果を発揮します。そこには、単なる主体と客体の関係を超えた、相互に影響を及ぼし合う協働的な関係があります。このように、人間と道具、さらには自然を含むあらゆる存在が互いに関わり合いながら、共に目的を実現していく関係性を再構築することが、これからの科学技術にとって重要ではないでしょうか。

「事実から当為は導けない」というのは、広く共有された命題の一つです。しかし、時として事実は、いかなる言葉よりも雄弁に当為を語り得ます。もし、これからの科学技術研究に意義があるとするならば、それは実生活上の利益を超えて、私たちが当然のものとして受け入れてきた主体と客体、人間と自然といった二項対立的な関係を問い直し、新たな協働のあり方を模索することにあるのではないでしょうか。

本稿で取り上げたベイズ最適化のような手法や、あるいは AI 技術全般が、そのような新たな関係性の構築に寄与することを願っています。

今回の研究は、京都大学医学研究科・小島諒介先生のご指導のもとに実施しました。本研究で使用した実験データおよびソースコードは、以下の GitHub リポジトリにて公開しています。

- `https://github.com/kazumasa-okamoto/ReactionT5-bo-yield`

第 4 章

Agentic な京大シラバス検索・時間割作成システム

4.1　はじめに

昨年度の 11 月祭では、我々は「完全一致のキーワード検索」にしか対応していなかった従来の公式シラバス検索システムに代わり、LLM による文章埋め込み（Embedding）技術を用いて、ユーザーの検索意図をより深く反映した抽象的な検索を可能にするシラバス RAG（Retrieval-Augmented Generation）システムの開発に取り組んだ [3, 4]。本年度は、2023 年以降に見られる Agent 型 AI の急速な発展を踏まえ、LLM を単なる情報検索の補助手段としてではなく、より能動的な「エージェント（Agent）」として活用することを目的とした。具体的には、ユーザーから提示される履修希望や興味・関心といった曖昧な要望を解釈し、時間割の空きコマや履修要件といった複雑な制約条件を考慮しながら、学期全体の時間割を自律的に構築・提案するシステムの開発を目指した。

4.2　背景技術：プロンプティング手法とエージェント構造

本プロジェクトで用いた LLM の制御手法として、基本となるプロンプティング手法である CoT[5] から、外部ツール連携の基礎となる ReAct[6]、そして Plan-and-Execute[7] のエージェント構造について順に解説する。

4.2.1　CoT (Chain of Thought)

Chain of Thought[5] は、大規模言語モデルに複雑な問題を解く際、最終的な答えに至るまでの推論過程を明示的に自然言語として生成させるプロンプティング手法である。単純な入出力のペアではなく、人間が問題を解く際の段階的な思考プロセスを模倣させることで、より正確で解釈可能な推論を実現する。単に「Let’s think step by step（段階的に考えよう）」というフレーズを指示に加える（Zero-Shot-CoT）だけでも推論性能の向上が見られる。さらに、具体的な推論過程を例示する Few-Shot-CoT により、Fine-tuning などの追加学習を行うことなく、プロンプトの工夫のみで実装できる利点がある。

Standard Prompting

■ **質問:** カフェテリアには 23 個のリンゴがありました。昼食に 20 個使い、さらに 6 個買いました。リンゴは何個ありますか？

回答: 答えは 27 個です。

Chain-of Thought Prompting

■ **質問:** カフェテリアには 23 個のリンゴがありました。昼食に 20 個使い、さらに 6 個買いました。リンゴは何個ありますか？ 回答は順序立ててステップごとに出力してください。

回答: 最初に 23 個ありました。20 個使ったので、23-20=3 個残りました。そして 6 個買ったので、3+6=9 個になります。

CoT により、算術推論・常識推論・論理推論など様々なタスクでの性能改善が見られた [8]。さらに、CoT の性能改善はモデルの規模（スケール）に強く依存する「創発的性質（Emergent Abilities）」を示すことが知られている。一説には 100B（1000 億）パラメータを超えるような大規模モデルでは劇的な性能向上が観察される一方、それ未満のモデルでは標準的なプロンプティングと性能がほぼ同等に留まると報告されている。CoT によって LLM 内部の思考プロセスは改善され、推論能力は向上した。しかし、推論に用いる知識が LLM 内部に事前学習されたものに限られるという課題（知識の陳腐化やハルシネーション）は依然として残る。

4.2.2 ReAct (Reason and Act)

前述の CoT が扱った「推論（Reasoning）」能力と、外部環境に対する「行動（Action）」を結びつけたものが ReAct フレームワーク [6] である。ReAct は、LLM に「Reason（推論）」と「Act（行動）」を交互に生成させ、外部環境と相互作用しながらタスクを遂行するエージェントを実現する。ReAct における「ツール（Tools）」は、LLM と外部環境を接続する重要なインターフェースとして機能し、「Act（行動）」によって引数付きで呼び出される。ReAct フレームワークは、以下の中核要素が循環的に相互作用することで動作する。

(1) Thought（思考）
- 現在の状況と利用可能な情報を分析
- 目標達成に向けて次に必要な行動を計画
- 過去の観察結果を統合して推論を更新

(2) Action（行動）
- 特定のツールを選択し、適切なパラメータで呼び出し
- 外部環境への具体的な操作を実行
- 情報取得や状態変更を実現

(3) Observation（観察）
- ツールの実行結果を受信
- 新たに得られた情報を文脈に統合
- 次の思考プロセスへのインプットとして機能

ReAct は CoT と同様に、追加の Fine-tuning を必要とせず、few-shot prompting で実現される。

ReAct 応答例

■ 利用可能なツール:

`Search[entity]`　Wikipedia API を通じてエンティティ情報を検索
`Lookup[keyword]`　現在参照中のページ内でキーワードを検索
`Finish[answer]`　最終回答を出力してタスクを終了

■ 質問: Apple Remote 以外で、Apple Remote が本来操作するように設計されていたプログラムを操作できる別のデバイスは何ですか？

■ 実行ログ (思考・行動・観察):
思考 1: Apple Remote を調べて、それがどのプログラムを操作するように設計されていたかを知る必要がある。
行動 1: `Search[Apple Remote]`
観察 1: Apple Remote は、2005 年に Apple から発売されたリモコンで、Front Row というメディアセンタープログラムを操作するために設計されていた。

思考 2: Apple Remote は Front Row を操作するために設計されていた。次に、他にどのデバイスが Front Row を操作できるかを調べる必要がある。
行動 2: `Search[Front Row controls]`
観察 2: Front Row は、Apple Remote または Mac のキーボードで操作できる。

思考 3: Front Row は Apple Remote と Mac のキーボードで操作できることが分かった。
答え: `Finish[Mac のキーボード]`

4.2.3　Plan-and-Execute

ReAct は強力なフレームワークである一方、いくつかの課題も存在する。ReAct は「思考」と「行動」を密接に繰り返す（Thought-Action-Observation のループ）ため、局所的な最適解に陥りやすい。つまり、タスク全体を見通した大局的な戦略の維持が困難となり、長期的な文脈理解が必要なタスクでは元の目標を見失ったり、非効率なステップを繰り返したりする場合があった。

Plan-and-Execute[7] は、これらの課題に対処するため、役割を二つの LLM（あるいは二つの役割）に分離するアプローチである。

Planner (計画者): Executor が使用可能なツール群を前提として、与えられたタスクを達成するための一連の行動計画（サブタスクのリスト）を生成する。

Executor (実行者): Planner が生成した計画に基づき、各ステップを順次実行する。この際、ステップ内部では ReAct のように思考とツールの使用を（局所的に）繰り返しながらサブタスクを遂行する。

効果と性能改善

Plan-and-Execute を採用する最大の利点は、認知負荷の分離にある。 複雑・長期的なタスクにおいて、Planner は全体を俯瞰して計画を立案することに集中する。これにより、長期的な文脈の理解

や冗長な行動の回避といった、大局的な最適化が可能になる。一方、Executor は計画全体を意識する必要がなく、目の前の局所的なサブタスク実行に集中できる。

もう一つの利点として、解釈可能性と介入可能性の向上が挙げられる。ReAct が逐次的な思考の連鎖であるのに対し、本手法では最初に行動計画全体が明示されるため、システムが何をしようとしているのかを人間が容易に解釈できる。さらに、計画が生成された段階で人間がその内容をレビューし、修正を加えるといった介入も可能になる。

4.3 データ設計と前処理

京都大学が公開しているシラバスデータ [9] から、科目名・教員・成績評価・授業概要等の各種データを抽出し、検索に利用可能な状態へ前処理する基盤は、昨年度の RAG システム開発の時点で確立していた。

本プロジェクトにおけるデータ処理は、この昨年度のデータを基盤としつつ、従来の「人間による検索」用から「LLM がツールとして利用」しやすい形式への変更、および「時間割全体の最適化」に必要な情報の追加を主眼とした。

データの前処理に関する詳細（基本的な抽出ロジック等）は昨年度会誌の記載 [4] を参照されたい。本章では、昨年度からの主要な変更点と、エージェント化のために新たに追加した項目についてのみ詳述する。

4.3.1 シラバスデータの変更点

メタデータ

既存のシラバスデータから抽出したメタデータに対し、LLM が計算やフィルタリングに利用できるよう、以下の正規化と形式変更を行った。

- 単位数
  従来「2 単位」のような文字列型 (str) であったデータを、エージェントが履修単位数を計算できるように数値型へ変更した。その際、「0.5 単位」といった例外が存在するため、整数 (int) 型ではなく浮動小数点数 (float) 型を採用した。
- 開講年度・開講期
  「2025・前期」のように単一の文字列であったものを、「年度 (2025)」と「開講期 (前期)」の二つに分離し、メタデータ検索時の絞り込みを容易にした。
- 配当学年
  「全回生」「2 回生以上」「主として 1・2 回生」など、表記揺れが多数存在したため、正規表現を用いてこれらを網羅的に解釈し、対象学年を示す数値のリスト（例: [1, 2]）に正規化した。（薬学部等、6 年制学部にも対応し 1〜6 の範囲とした。）

シラバス本文

シラバス本文は、そのまま LLM に渡すには冗長であり、トークン長（コンテキスト長）とコストを圧迫する。そこで、LLM の認知負荷低減と情報圧縮の観点から、用途に応じて二段階の要約レベルを生成した。

1. 検索結果・授業比較用の構造化要約

これは、検索ツールが LLM に提示するための要約である。LLM が時間割に未登録の授業を評価・比較するために参照するものであり、授業の主要な情報は保持しつつ冗長性を排除する必要がある。そこで、「授業のテーマ」「目的・背景」「主要トピック」といった項目をあらかじめ定義し、LLM に項目立てて構造化要約させた。これによりトークン数は平均で半分以下に抑えられた。また、副次的な効果として、情報の構造化による LLM の認知負荷軽減も期待される。

2. 現在時間割の確認用・短縮要約

これは、既に時間割に組み込まれている授業の情報を LLM に渡す際に用いる要約である。他の授業との比較などを行わないため、詳細な説明は不要であり、キーワードや単文レベルまで情報を圧縮した。

検索結果の要約例

この授業で学べること**中心テーマ:** 哲学史（古代から 18 世紀まで）* **目的・背景:** 哲学の基本的な問いを理解し、古典的テクストを通じて哲学史的知識を身につけること。* **主要トピック:** プラトン、アリストテレス、トマス・アクィナス、デカルト、ロック、カントの思想* **得られるスキル:** 哲学的な問いを自ら考える能力、哲学史に関する基礎知識 ## 学習の進め方* **授業スタイル:** 古典テクストの読解を通じた講義形式* **学習の流れ:** 導入から始まり、各哲学者の古典を時系列で読み解く ## 前提知識* **推奨される知識:** 特になし* **備考:** （授業内容からの推測）

現在時間割用の要約例

哲学史の基礎/古代から 18 世紀までの主要哲学者の思想/古典的テクストの読解/哲学の基本的な問いの理解

4.3.2　単位取得率データ

本プロジェクトでは、京都大学が公開している年度毎の「履修者数」と「単位取得者数」のデータ [10] を基に単位取得率を計算し、これを「単位取得の容易さ」を評価する指標の一つとして LLM に提供した。しかし、この単位取得率データには「科目名」と「開講部局」しか記載がなく、特に科目名は、一部が省略・変更されているため（例：「(演習)」やクラスコード「1S5」の欠落)、シラバスデータとの対応付けは困難であった。

今回は、特に楽単度合いの必要性が高いと想定される全学共通科目に絞り、以下の方法を用いてデータの紐付けを試みた。

1. 正規化: 双方のデータに対し、半角全角の統一、空白文字の削除を行う。
2. 前方一致検索: シラバスデータ側の授業名 S に対し、S.startswith(P) で完全に S に前方から含まれている単位取得率データ側の授業名 P を検索する。
3. 最長一致の採択: S に対して複数の P が前方一致した場合（例：S=”哲学 II”に対し P1=”哲学”と P2=”哲学 II”が一致)、最も一致文字列が長い P（この例では P2）を採択する。
4. 開講部局による絞り込み: 上記で絞り込んでも複数の候補が残る場合、全学共通科目の開講部局である「国際高等教育院」のデータを優先する。
5. データの合算: それでもなお候補が残る場合（例：学部生用と大学院用の同一名称科目)、デー

タは区別せず、履修者数と単位取得者数を合算して扱った。

この手法により、全学共通科目 3154 個の授業のうち、約 95.6% にあたる 3015 個のデータ紐付けに成功した。一方、残りの 139 個は対応するものが見つからなかった。 これら未対応データには、昨年度で廃講になった科目や、表記揺れの範疇を超えて名称変更された科目などが含まれると推測される。一部、名称が同一であるにも関わらず対応付けられなかったデータも存在し、これは今後の課題である。

4.3.3 データ処理における課題

現状のデータ前処理には、依然として多くの課題が残されている。

- シラバスデータの形式が学部・学科毎に異なっており、現状では特定の学部（例：医学部）に一切対応できていないほか、他学部でも一部データの抽出ミスが散見される。
- 履修要件、クラス指定科目、推奨科目といった重要な制約条件の自動抽出が実現できていない。
- 単位取得率データは、全学共通科目以外（専門科目）には未対応。専門科目は科目名の揺れがより大きく、対応付けが格段に困難になるためである。

4.4 時間割提案 Agent の基本構造

時間割提案 Agent の基本的な機能について説明する。

4.4.1 ユーザー情報の入力

ユーザーの情報や要望についての入力について、チェックボックスやスライダーを用いて確実に入力できる構造化データの項目と、時間割全体への要望や興味関心のある分野などテキストベースで入力する項目の 2 つを用意した。

確実に入力する構造化データ

- 学部・学科・学年
- 1 限を許すか
- 空コマを許すか
- 全休を作りたいかどうか
- 授業を入れたくないコマ
- 授業形式への希望
- 授業評価方法の希望
- 既に履修することを決めている科目

テキストによる入力

テキスト入力欄を用い、主にユーザーの興味・関心分野や、上記項目では伝えきれない詳細な要望・制約を入力できるようにした。初期実装では単一のテキスト入力のみであったが、後に LLM を用いて対話的に要求を引き出すチャット形式に変更した。ただし、この対話による要求抽出プロセスの高度化は、本プロジェクトでは十分に改良できず、今後の課題として残されている。

4.4.2　LLM の利用ツール

エージェントが環境（シラバスデータベースや内部の時間割状態）と相互作用し、タスクを遂行するために、以下の 3 つのツールを定義し、LLM が ReAct 形式で呼び出せるよう設計した。

- 検索ツール
  シラバスデータベースから授業を検索する。基本的には昨年度と同様、FAISS[11] を用いたシラバス本文のベクトル検索である。メタデータ検索では、複雑な and, or 検索まで可能ではあるが、初期実装では LLM のツール利用を簡素化するため、検索パラメータを「曜日・時限」と「検索クエリ」のみに限定した。Plan-and-Execute アーキテクチャへの移行後は、Planner の認知負荷が軽減されたため、任意パラメータとして「開講部局」「分野」「配当学年」等も指定可能とし、より高度な絞り込みを実現した。検索結果を ToolMessage の形式で、「授業 ID」「授業名」「曜日・時限」「授業形式」「評価方法」「単位取得率」「シラバス要約」を JSON 形式で返す。該当する検索結果が見つからなかった場合は、ToolMessage で検索結果が見つからなかったことを返す。
- 授業追加ツール
  指定した ID の授業を現在の時間割に追加する。授業名による指定は、LLM の出力（表記揺れ）によって不安定となり、状態管理を複雑にするため、一意な授業 ID を指定する形式を採用した。ToolMessage を用いて、授業の追加が成功したか否かを返す。失敗した場合は、その理由（「月曜 2 限が重複」等）も併せて返す。
- 終了ツール
  時間割提案完了・計画の各ステップのタスクの完了を判断した際に呼び出し、次のステップへと移行する。

4.4.3　LLM へのデータ入力形式

エージェントの思考ステップにおいて、プロンプトに含める現在の状態として、以下の情報を渡す。具体的には以下のような形式で渡す。

現在の状態
現在の時間割情報
- **現在の時間割**:
月曜 1 限: 1227
月曜 2 限: 610
月曜 3 限: 無し
〜〜〜
金曜 5 限: 無し
- **集中講義**: 無し
- **各授業 id の詳細**:
1227: ' 科目名': ' 英語リーディング ER30', ' 担当教員': ' 非常勤講師', ' 授業形式': [' 演習'], ' 評価': ' 平常点': 60, ' 課題': 0, ' 発表': 0, ' 討論': 0, ' 小レポート': 0,' 小テスト': 0, ' 期末レポート': 40, ' 期末試験': 0, ' シラバス本文': "アカデミックリーディングの基礎/学術入門書の精読/英語の学術的語彙/近代自然科学の歴史に関する知見"
- **総取得単位数**: 21.0
- **学部**: 理学部
- **学科**: 理学科
- **現在の学年**: 1
ユーザーからの要望
- 一時限目には授業を入れないでください
- 一日の内で授業と授業の間に空いている時間があっても良い
- 授業が一つも入っていない曜日が作りたい
- 授業形式は、講義, 演習が好ましく、実験は避けたいです。
- 授業の評価方法は、講義, 演習が好ましく、実験は避けたいです。
- **避けたい曜日・時限**: 木曜 5 限
興味のある内容: "1. **学びたい分野**: - 数学系（特に応用的な分野） - 金融や保険数学 2. **興味のある応用先**: - 金融 - 保険数学 - その他の数学の応用先（具体的には未記載） 3. **人文社会系の科目**: - 経済 - 法律 - 社会に出てからも活躍しそうな内容. **授業形式の希望**: - 授業にあまり出たくない - 期末レポートや期末テストの割合が高い形式を希望"

4.5 ReAct 型 Agent

本プロジェクトの初期実装として、LLM が環境と相互作用しながら自律的にタスクを遂行する ReAct アーキテクチャ [6] を採用した。将来的な一般公開時の運用コストを考慮し、LLM モデルには低コストな「gpt-4o-mini」を使用した。

4.5.1 初期実装と課題

初期の設計では、検索・授業追加・終了の 3 つのツールのみを用い、単一の LLM が「現在の状態（時間割情報）」と「ユーザー要望」に基づき、「思考」と「行動」を逐次的に繰り返す、シンプルな ReAct エージェントを構築した。

しかし、この構成で検証を行った結果、実用上、以下の深刻な問題点が多発した。

- 探索の非効率性
  関連する授業が見つからない場合、LLM は同一または類似の検索クエリを際限なく発行し続け、検索の無限ループに陥る。
- 状態認識の欠如
  既に授業追加済みのコマを「空き」であると誤認し、重複して授業追加を試み、失敗する。エージェントはこの失敗から学習できず、同じコマへの追加を繰り返した。
- タスク完了判断不能
  時間割が十分に埋まった、あるいはこれ以上追加すべき授業がないという「タスク完了」の条件を LLM 自ら判断できず、設定した最大ステップ数に達して強制終了するケースが多発した。
- トークン数制限
  ReAct は設計上、過去の「思考・行動・観察」の全履歴をコンテキストに含める。これに加えて「現在の時間割詳細」という静的な情報も毎回のプロンプトに含めるため、思考が長くなると、API のトークン数上限に容易に到達した。

4.5.2　課題への対策

上記の問題点を解決するために、次の 2 点を変更した。

一つ目は、プロンプトによる「空コマ」の明示である。時間割の情報から空いているコマを判断させてしまうと LLM の負荷が上がってしまうため、事前に空コマを求め明示的に与えることとした。これにより、既に埋まっている時間に授業を追加してしまう事だけでなく、検索の段階で埋まっている時間を検索してしまうという問題まで、大幅に削減できた。

二つ目は、思考履歴のトークン数による削減である。全履歴を渡す代わりに、トークン数で履歴を打ち切り、最新の思考のみをコンテキストに含めるよう変更した。この変更後も、エージェントのタスク遂行性能に顕著な低下は見られず、コストと API 制限の問題を実用的なレベルで解決した。

4.5.3　ReAct 型の限界

前節で述べた対策では根本的な問題は解決せず、システムプロンプトによって検索・授業追加・完了判断の思考方法について指示をしても、検索の失敗、不毛な授業追加の繰り返し、そしてタスク完了の判断ミスといった問題を本質的には解決できなかった。また、ユーザーの興味関心のある授業を検索・追加することはできるが質が高いとは言えず、一般的な履修の決め方・履修要件などはほとんど考慮して考えることが出来なかった。検証のため、gpt-5-mini など少し上位のモデルを使用したところ、検索と授業追加に顕著な性能差が見られた。これによりタスクの成否が LLM の推論能力に強く依存することは確認できた。しかし、これにより我々は、本プロジェクトにおける最大のジレンマ、すなわち「性能とコストのトレードオフ」に直面することとなった。

低コストモデルでは、履修要件やユーザーの要望まで理解して検索・授業追加をすることが出来ず、高性能モデルでは要件を理解することはできるが ReAct 型の思考と行動の繰り返しにより、API 料金が嵩んでしまう。この「性能」と「コスト」のトレードオフは、ReAct 型アーキテクチャを本タスクで実用化する上での根本的な障壁であると結論づけた。この問題を解決するため、我々はアーキテクチャそのものを見直し、「複雑な思考」と「単純な実行」を分離する必要があると考えた。これが、

次章で述べる Plan-and-Execute アーキテクチャへの移行の直接的な動機である。

4.6 Plan-and-Execute 型 Agent

前章で述べた ReAct 型の根本的な課題、すなわち「性能」と「コスト」のトレードオフを克服するため、我々は Plan-and-Execute（計画・実行）アーキテクチャ [7] へと移行した。本アーキテクチャは、その構造的特性から、このジレンマに対する直接的な解決策となると判断したからである。

第一の利点は、LLM が担うべき認知的な負荷を分離することが出来る点にある。

- Planner は、履修要件、ユーザーの曖昧な要望、時間割全体のバランスといった、大局的で複雑なコンテキストの理解に集中する。
- Executor は、複雑な全体要件を考慮する必要がなく、Planner から指示された個別のサブタスクの実行という、単純化されたタスクに集中できる。

第二の利点は、この役割分離により、役割に応じた LLM モデルの使い分けが可能となる点である。これは、ReAct 型が直面した最大のジレンマに対する直接的な解決策となる。

- Planner は、複雑な要件を正確に解釈し、質の高い実行計画を立案するため、高性能・高コストな LLM を割り当てる。計画の立案はタスクの初回に一度だけ実行されるため、API コストへの影響は最小限に抑えられる。
- Executor は、Planner の計画に従ってツールを実行する（比較的単純な）役割であるため、低コストな LLM を割り当てる。Executor は「検索」や「追加」のたびに何度も呼び出されるが、安価なモデルを用いることで、全体の API コストを劇的に低減できる。

4.6.1 Plan-and-Executor 型の実装

Planner

Planner は、「ユーザー情報」「履修要件」「確定済みの授業」「推奨科目リスト」「履修戦略のヒント」といった情報をマークダウン形式で受け取り、実行計画を立案する。

特に、Planner が複雑な思考に集中できる利点を活かし、LLM が単独では知り得ない情報（ドメイン知識）を参照情報としてプロンプトに組み込んだ。これは Few-Shot プロンプティングのように機能し、より質の高い計画を生成するよう出力を誘導した。

出力は、Executor が局所的な判断に陥らないよう、全体の方針を示す「全体の指針」と、具体的なサブタスクのリストである「計画」を含む厳格な JSON 形式として定義した。

Executor

基本は今までの ReAct 型を原型としたが、役割に合わせて以下の点を変更した。

- システムプロンプトで、全体の時間割を自律的に考えるための指示を削除し、Planner の計画に従って検索・授業追加を行うように指示
- 従来の ReAct 型では、現在の時間割に加えユーザー情報などを毎回渡していた。しかし、Executor はユーザー情報を解釈する必要はないため、プロンプトから時間割以外の冗長な情報を削除した。各ステップの実行に必要な情報は、すべて Planner が生成した計画から渡される。

4.6.2　改善点

ReAct 型から Plan-and-Execute へ変更したことにより、生成される時間割の質が大幅に改善された。

第一に、本構成の主目的であった Executor の動作安定性が向上した。ReAct 型では Executor 自身が思考する必要があったため、無駄な検索や失敗が多発していた。一方、本構成では Executor の役割が「計画の忠実な実行」に単純化された。Planner から「検索クエリの方向性」や「授業の比較基準」が明確に与えられるため、よりユーザーの要望に沿った授業選択が可能となった。

第二に、時間割全体の質が向上した。Planner が高性能モデルの推論能力を活かし、質の高い計画を立案できた場合に限定されるものの、その計画を Executor が実行することで、人間が考案したものと遜色のないまともな時間割を提案できるケースも確認された。しかし、この改善は、システム全体のボトルネックが「Executor の実行能力」から「Planner の計画立案能力」へとシフトしたことを意味する。 LLM が本質的に知り得ない、各学部・学科固有の「ヒューリスティックな履修戦略」を Planner に理解させ、質の高い計画を安定して生成させることは極めて困難であり、本稿執筆時点でもプロンプトの試行錯誤が続いている。

4.7　今後の課題と展望

本システムは現段階では未完成であり、多くの課題が残されている。

- データ処理について、履修要件や推奨科目などまだまだ対応しきれていないものが多く残っている。
- 学部学科ごとに異なるようなヒューリスティックな履修戦略について、一部学部にしか対応できていない。
- LLM のコンテキスト理解とトークン数のバランスを取った時間割情報などの渡し方の試行錯誤。
- 計画の質が良ければ上手く時間割を考えられるが、Planner の質の良い計画立案の制御がまだ完全にはできていない。
- Executor がタスクを完了できなかった際に、対応しきれず次のステップに移ってしまう。Executor で問題が起こった場合に、Planner へ戻り全体を見通して計画の修正ができるようにしていく。
- 時間割を一度提案するだけでなく、提案した時間割をベースとしてユーザーとの対話しながら修正・改善できるようにする。

今回の 11 月祭までには、一般に公開できるレベルまでシステムの完成度を高めることはできなかった。 しかし、来春の新入生が入学する時期を目標とし、本章で挙げた課題の改善と試行錯誤を継続する。最終的には、より高精度かつ多機能な時間割提案エージェントとして公開できるよう、引き続き開発を行っていく。

第 5 章

音楽と自然言語の類似度測定システムの開発

5.1　はじめに

これまでの楽曲検索システムは曲名やアーティスト名、あるいはジャンルやテンポといったタグによる検索が主流でした。しかし、特定のタグやメタ情報だけでは「タグに存在しない感性的な要求」や楽曲の特徴からは直接推察できない要望に応えることが困難です。また、YouTube 等のプラットフォームでは話題性やランキングに基づいて検索結果が偏るため、純粋な類似性評価とは異なる順位付けが行われる場合があります。そこで本プロジェクトでは、自然言語クエリと音楽の類似度を直接測定するシステムを開発しました。

5.2　アプリの機能

本システムは以下の主要機能を備えています。

- **自然言語クエリ入力**：ユーザーは「激しい感情を歌った曲」や「リラックスできるメロディ」など、一般的な日本語表現で楽曲の特徴を指定できます。
- **CLAP モデルによる類似度測定**：CLAP モデルを用いて、テキストと音楽（音響特徴）間の類似度を計算します。
- **楽曲データベースとの照合**：事前に構築した楽曲データベース（J-POP 中心）とクエリの埋め込みを比較し、類似度の高い楽曲を特定します。
- **類似度の可視化**：類似度に基づき楽曲をランキングし、2 次元プロット等で可視化してユーザーに提示します。

これにより、ユーザーは従来のタグベース検索では見つけにくい、自身の意図に近い楽曲を効率的に探索できます。

図 5.1: クエリの入力例

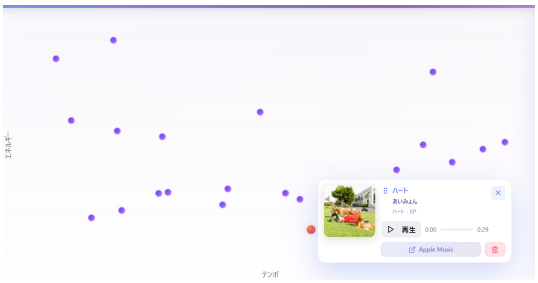


図 5.2: 類似度の可視化結果

5.3　推論・可視化の流れ

5.3.1　概要

ユーザーが自然言語で楽曲の特徴をクエリとして入力した際に、CLAP モデルを用いてテキストと音楽（音響特徴）の類似度を測定します。本節ではまず CLAP の基本構造と本研究での適用方針を説明し、その後で具体的な推論プロセスを示します。

5.3.2　推論プロセス

具体的な推論プロセスは以下の通りです。

1. **クエリ入力と前処理**：ユーザーが「激しい感情を歌った曲」のような自然言語 (日本語) クエリを入力します。
2. **日本語から英語への翻訳**：Google Translation API を用いて英語に翻訳します。
3. **テキスト・音楽埋め込みの生成**：翻訳したテキストとその他の多様なクエリ X_s を Text Encoder に、音源は前処理（メルスペクトログラム等）を行った上で Audio Encoder に入力し、埋め込みベクトル E_{query} を生成します。
4. **類似度の計算**：データベースに格納した楽曲の埋め込み $E_{\text{music},i}$ とクエリ埋め込み E_{query} のコサイン類似度 $\text{Sim}(E_{\text{music},i}, E_{\text{query}})$ を計算し、各楽曲において、類似度スコアの平均で割ります。
5. **検索結果の出力**：翻訳したテキストに関する類似度を基に楽曲をランキングし、類似度を座標軸として可視化してユーザーに提示します。

この推論により、ユーザーは自身の意図した自然言語のニュアンスに近い楽曲群を探しやすくなります。

5.3.3 CLAP モデルについて

CLAP[12] は、自然言語と音響を同一の埋め込み空間にマッピングすることを目的とした対照学習型モデルです。典型的には以下の構成を持ちます。

- テキストエンコーダ（Text Encoder）とオーディオエンコーダ（Audio Encoder）を備えています。
- 対応するテキスト・音声ペアの埋め込みを近づけ、非対応ペアを遠ざける対照損失で学習します。

そのため、CLAP は「テキストで表現された音の概念」と「実際の音響特徴」を意味的に結びつけることができます。ゼロショット検索やテキスト条件付きの音検出に有効であり、本研究ではこれを日本語クエリと楽曲の類似度測定に応用します。
既存の CLAP は環境音や一般音声を中心に学習されていることが多く、楽曲固有のメロディや楽器、ハーモニーといった特徴を扱うには、対象ドメイン（例：J-POP）でのファインチューニングが有効です。そこで本研究では CLAP の Text/Audio Encoder を J-POP データで微調整し、日本語クエリに対して高精度な類似度評価が行えるように設計します。

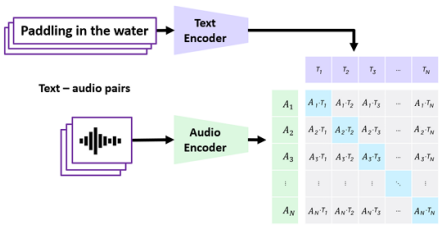


図 5.3: 可視化結果

5.4 学習の流れ

5.4.1 概要

既存の CLAP モデル（例：laion/clap-htsat-fused）はクラシックや環境音中心に学習されているため、日本の楽曲に対する類似度測定には最適化されていません。 本研究では J-POP を中心としたデータセットで CLAP をファインチューニングし、日本語クエリと楽曲埋め込みの対応を強化します。

1. **データセットの構築**：日本の楽曲を対象に、楽曲ファイル（オーディオ）と対応するテキストデータ（YouTube コメント、歌詞、タグ等）を収集します。
2. **ノイズ除去**：収集したテキストデータにはノイズが含まれるため、フィルタリングや前処理で不要な情報を除去します。
3. **対照学習**：正例ペア（対応する楽曲とテキスト）間の類似度を最大化するように対照損失で学

習します。

この手順により CLAP は日本語の表現と楽曲の音響特徴との対応関係を深め、実用的な類似度評価が可能になります。

5.4.2　データセットの構築

学習用に使うデータセットとして必要なのは、プレビュー音源（オーディオ)、歌詞、タグ、曲の感想を表すコメントやレビューの情報 (テキスト) です。
プレビュー音源 (30 秒) については、アーティスト名のリストを手動で作成し、それらを ITunes Search API によって検索をすることで取得します。各アーティストにつき上位 5 曲を検索します。曲の人気度や新しさなどに基づいて検索される曲が選ばれています。これにより、同時に曲のトラック情報やメタ情報を取得することができます。
歌詞については、Genius API を用いてアーティスト名、曲のタイトル名で検索し、取得できる部分は取得します。Genius API の検索で辿り着かない一部楽曲は music.jp という Web サイトを Web スクレイピングすることにより、取得します。以上の方法により楽曲の情報が取得できた楽曲を対象に、学習を行います。
楽曲の特徴や感想を表すコメントやレビューの情報の収集については、YouTube Data API を用います。該当する楽曲の MV をアーティスト名、曲のタイトル名で検索し、そのコメント欄から各曲 100 件ずつ収集します。コメントは新しい順、人気順の二つのソートの方法がありますが、良質なコメントを多く集めたいので、人気順を選択しました。Youtube のコメント欄は学習に有用な、曲の特徴を表したコメントばかりではありません。学習に不向きなコメントは次に解説する手順で除去することで、CLAP 側のファインチューニングの精度の向上を試みました。

5.4.3　ノイズとなるコメントの除去

日本語版 BERT[13][14] を用いてコメントを以下のラベルに分類し、ラベル 0 に該当するコメントのみ曲の特徴を表していると判断して残すことにしました。今回用いたモデルは RoBERTa をベースにしたものです。RoBERTa は NSP(次の文章の予測タスク) を削除したり、マスキングパターンを固定しないなど、BERT の学習手法をロバストに最適化されています。このモデルは今回のような分類タスクに優れています。

1. **ラベル 0**: 曲の感想や、曲にまつわる投稿者のエピソード (例: 儚いラブソングだと感じました。)
2. **ラベル 1**: 曲に関連する MV やアニメ、ドラマなど、関連コンテンツに関するコメント (例: 昨日のドラマはとてもよかった)
3. **ラベル 2**: 曲の再生数や人気度に関するコメント (例: 祝 1 億回再生!)
4. **ラベル 3**: アーティストのビジュアルや行動、エピソードに関するコメント (例: ドラマの主題歌書かせたらこの人がピカイチだね)
5. **ラベル 4**: 曲に関係ない、雑多なコメント (例: 毎日〇〇日記)

6. **ラベル 5**: 歌詞

BERT のファインチューニングには事前に教師データを用意しておく必要があります。教師データのラベリングについては手動で 600 件行いました。ラベル 0 や 4 が多くなり、ラベル 5 は少ないなど、ラベルの不均衡が起きていたため、少ないラベルに該当するデータをコピーして水増ししました。これによって、データが 1200 件まで増え、全てのラベルに属するデータを均等に学習することができます。
学習時では 91 %の正解率 (accuracy) でラベル分類が行われました。全部のデータに手動で正解ラベルを与えたわけではないため、実際のコメントのラベル分類の正解率は正確には分かりませんが、これと同等の精度で分類できていると期待できます。実際、歌詞コメントなど、曲の内容に無関係なコメントは除去できていました。

5.4.4 対照学習の方法

本システムでは、**正例ペアのみを収集したデータセット**を用い、**InfoNCE Loss** に基づく対照学習を実行することで、モデルにオーディオとテキストの対応関係を学習させます。

1. **日本語テキストの英訳と埋め込み**: CLAP モデルの事前学習を考慮し、収集した**日本語テキスト**は、学習前に **Google Translation API** を用いて英語に機械翻訳します。これにより、翻訳された英語テキスト T と対応する楽曲データ A のペア (A, T) が学習の正例として用いられます。これらはそれぞれ Audio Encoder と Text Encoder に入力され、埋め込みベクトル $\boldsymbol{v}_A$ と $\boldsymbol{v}_T$ に変換されます。
2. **損失関数 InfoNCE Loss の基盤**: 損失関数には、バッチ内ネガティブサンプリング（In-batch Negative Sampling）の特性を持つ **InfoNCE Loss**（Information Noise-Contrastive Estimation Loss）を用います。この手法は、データセットから負例を明示的に用意しなくても、学習に利用されたミニバッチ内の正例でない組み合わせを自動的に負例として扱う仕組みです。
   具体的には、ミニバッチ内に N 個の正例ペア $\{(A_1, T_1), \ldots, (A_N, T_N)\}$ があるとき、オーディオ A_i に対するテキスト T_i は**正例**ですが、それ以外のテキスト T_j $(j \neq i)$ は全て負例として機能します。これにより、N 個の正例と $N(N-1)$ 個の負例が損失計算の過程で暗黙的に機能します。
3. **対照損失の計算とモデルの学習**: InfoNCE Loss $\mathcal{L}_{A \to T}$ は、オーディオ埋め込み $\boldsymbol{v}_{A,i}$ が、バッチ内の N 個のテキスト埋め込みの中から、正解テキスト $\boldsymbol{v}_{T,i}$ を正しく識別できる確率を最大化するように定義されます。コサイン類似度 $\mathrm{sim}(\boldsymbol{v}_{A,i}, \boldsymbol{v}_{T,j})$ を用いた、オーディオ A_i に基づく損失 $\mathcal{L}_{A \to T,i}$ は以下の通りです。

$$\mathcal{L}_{A \to T,i} = -\log \frac{\exp(\mathrm{sim}(\boldsymbol{v}_{A,i}, \boldsymbol{v}_{T,i}))}{\sum_{j=1}^{N} \exp(\mathrm{sim}(\boldsymbol{v}_{A,i}, \boldsymbol{v}_{T,j}))} \tag{5.4.1}$$

   最終的な対照損失 $\mathcal{L}$ は、オーディオからテキストへの損失 $\mathcal{L}_{A \to T}$ と、その逆 $\mathcal{L}_{T \to A}$ の双方向の和として計算され、これを最小化することで学習を行います。
   $\mathcal{L} = 0.5 \times \frac{1}{N} \sum_{i=1}^{N} (\mathcal{L}_{A \to T,i} + \mathcal{L}_{T \to A,i})$
   学習の目的は、正例ペア間の類似度を最大化し、負例として機能するバッチ内の非対応ペア間の類似度を最小化することで、エンコーダが楽曲の音響特徴と英訳されたテキスト表現の間の

対応関係を学習することです。

この手法で CLAP モデルをファインチューニングすることで、日本語クエリと楽曲の類似度測定に特化した性能向上がされました。
実際に学習後のモデルを用いて、自然言語クエリと楽曲の類似度測定を行った結果、ユーザーの意図に近い楽曲が学習前の時より上位にランクインするなど、実用的な性能が確認されました。

表 5.1: 「力強く歌うことで、葛藤から克服しようとしている」というクエリに対する類似度評価の比較

モデル / 楽曲	変更前	変更後
残響賛歌	−0.9784	0.9528
天体観測	1.1011	0.9412
群青	−0.9766	0.9510

5.5　おわりに

本研究では、自然言語クエリと楽曲オーディオを同一埋め込み空間で比較するシステムを提案しました。ユーザーは感性的な日本語表現を用いるだけで、自身の意図に近い楽曲を効率的に探索・可視化できます。しかし、「米津玄師のような曲」など、特定の固有名詞を含むクエリに対する類似度評価については、CLAP の事前学習データへの依存度が高く、十分な精度を確保することが難しいという課題が残っています。今後はこういった面も含めて、CLAP のさらなる最適化、多ジャンル対応を行うことでより広範なユーザー体験の向上を目指します。

第 6 章

顔認識で操作するゲーム

6.1　はじめに

「目は口ほどに物を言う」という言葉が示すように、人間同士のコミュニケーションにおいて表情はきわめて大きな役割を担っています。しかし現在のヒューマン–コンピュータ・インタラクションにおいては、そのモダリティの多様性にもかかわらず、依然としてキーボード入力や音声指示といった言語的・運動的な情報に強く依存したままです。

もしこの制約を超え、表情そのものを操作インタフェースとして利用できれば、より自然で直感的な応答が可能となります。そこで本プロジェクトでは、顔表情認識を用いて操作するゲームを開発し、表情入力のユーザビリティと可能性を検証することを目的とします。

6.2　使用ライブラリ・論文解説

本プロジェクトでは、顔認識の基盤として Google が提供する MediaPipe Face Landmarker[15] を採用しています。MediaPipe Face Landmarker は、顔の目や口といった主要ランドマークを CPU 環境でもリアルタイムに推定できる軽量・高精度なライブラリであり、表情入力を扱う本プロジェクトに非常に適しています。

このライブラリを利用することで、ユーザーの細かな表情変化を高い精度・低遅延で取得でき、自然かつ滑らかな操作体験を実現できる点が大きな利点です。

MediaPipe Face Landmarker は、Google が開発した軽量顔検出器 BlazeFace[16] を基盤として構築されています。本節は、その論文をもとに、その主要なアーキテクチャ上の特徴を概説します。

BlazeFace は、モバイル GPU 上での推論に特化して設計された軽量かつ高速な顔検出モデルです。スマートフォン上で 200 – 1000 FPS という極めて高い速度で動作し、AR アプリケーションなどで求められる 2D/3D ランドマーク推定や表情分類へほぼ遅延なく顔領域を提供できる点が特長です。

この革新的な性能は、以下の三つの主要なアーキテクチャ上の工夫によって実現されています。

6.2.1　軽量な特徴抽出ネットワーク

MobileNetV1/V2[17][18] に着想を得ながらも、モバイルでの物体検出に最適化された独自のコンパクトなネットワークを採用しています。このネットワークでは、図 6.1 に示すように、一般的な 3x3 カーネルの代わりに 5x5 カーネルを用いた深度方向分離可能畳み込み（depthwise separable convolution）をモデルのボトルネック部分で活用します。これにより、演算量を抑えつつも、より

少ない層で広い受容野（receptive field）を確保し、効率的な特徴抽出を可能にしています。

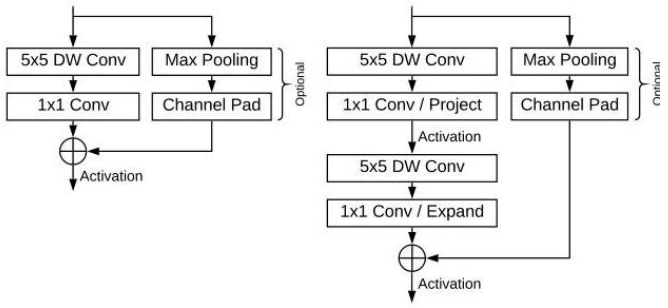


図 6.1: BBlazeBlock（左）とダブル BlazeBlock（右）のアーキテクチャ

6.2.2　GPU に最適化されたアンカー方式

BlazeFace では、GPU の処理特性に合わせた独自のアンカー生成方式が採用されています。一般的なモデルが特徴マップを 1 × 1 まで段階的に縮小していくのに対し、BlazeFace は図 6.2 に示すように解像度を 8 × 8 で止め、それ以上のダウンサンプリングを行いません。

これは、低解像度の特徴マップを扱う際に発生する GPU ディスパッチの固定オーバーヘッドを削減し、推論速度の大幅な向上につながる設計です。

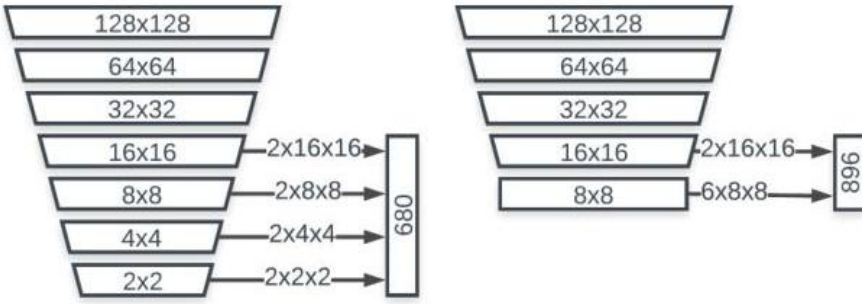


図 6.2: BlazeFace のアンカー方式

6.2.3　後処理での tie resolution 戦略

物体検出では多くの場合、複数のバウンディングボックスの候補が同一の物体を指す場合に NMS（Non-Maximum Suppression）を用いて 1 つのバウンディングボックスに絞り込みますが、BlazeFace は NMS に代えて 独自のブレンディング戦略を採用しています。

複数のバウンディングボックスの重み付き平均を取ることで最終出力を決定し、フレーム間での検出結果のちらつき（時間的ジッター）を大幅に低減します。その結果、より滑らかで安定した検出が実現されます。

6.2.4 まとめ

これらの工夫により BlazeFace は、モバイル環境における顔検出の速度と精度の基準を大きく引き上げました。そしてその成果は MediaPipe Face Landmarker を通じて、現在の多くの AR・表情認識アプリケーションの基盤技術となっています。

6.3 ゲーム入力の実装

本節では、MediaPipe ライブラリを用いた表情認識によるゲーム入力の実装について、主に工夫した箇所に絞って解説していきます。

6.3.1 入力の詳細

今回はリアルタイム推論を行うので、比較的検出しやすい「まばたきをする」「口を開ける」「笑顔になる」の 3 つの表情入力を対象とします。各表情は立ち上がり時（しきい値を下回る状態から上回った瞬間）に一度だけ入力として扱います。長押しのような操作は存在しません。

6.3.2 軽量化・遅延削減

今回の実装において、肝となるのはやはり軽量化・遅延削減です。ゲーム入力にはリアルタイム性が求められるため、表情認識をいかに素早く処理してゲームに反映させるかが重要になってきます。最適化を行わなかった初期実装では、ゲーム画面がカクついて到底快適ではありませんでした。画面にカクつきが生じるのは、1 フレーム当たりの推論時間が描画間隔に対して間に合っていなかったのが原因であると考えられます。そこで実装したのが非同期推論です。

非同期推論とは、ゲームの描画とは別の時間軸で表情認識の推論を行う機能です。初期実装では、1 フレームの推論が完了してから次のフレームの描画を始めていました。対して非同期推論では、1 フレームの推論の進捗に関わらず、次のフレーム描画を始めます。安定したフレームレートが重視されるゲーム描画には、この非同期推論は有効でしょう。表情認識処理とは無関係に描画が行われるので、画面のカクつきはこれで解消することができます

では実際、どのように非同期推論を実装しているのかについても少し触れておきます。

MediaPipe Face Landmarker には、標準機能で非同期推論を行う LIVE STREAM モードが搭載されています。今回の場合では、メインの描画処理とは別スレッドで、この LIVE STREAM モードで表情認識処理を行い、その処理結果を入力として採用しています。

さて、非同期推論を導入したことで画面のカクつきは解消されました。しかし、このままではもうひとつ問題があります。表情認識処理が非同期になった分、入力の遅延が発生してしまったのです。以前までの同期推論では、表情認識処理が完全に終了してから次のフレームが描画されるため、遅延の問題は発生しませんでした。そこで、次は推論自体を軽量化することで、遅延の解消に取り組みます。

具体的には、解像度の縮小とフレームレートの削減を行っています。表情認識処理に入力する画像の解像度を下げることで、処理は軽量になりますが、そのぶん精度の低下にもつながります。しかし、今回は「口を開ける」「まばたきをする」「笑顔になる」という比較的検出しやすい 3 つの特徴を区別できればよく、解像度を 80 × 60 ピクセル（幅×高さ）まで下げてもプレイにはあまり問題は生

じませんでした。同様にフレームレートに関しても、60fps から 15fps まで下げることで軽量化と精度の維持を両立しています。

加えて、カメラのバッファサイズの削減、MJPEG の使用、CPU/GPU デリゲートの切り替えにも対応しました。これらにより、入力タイミングの精度が求められる runner ゲームにおいても、遅延を気にせず快適にプレイできる水準を実現しました。

6.3.3　検知精度の向上

先ほどまでは軽量化・遅延削減の工夫を解説してきましたが、次に表情入力検知について紹介します。「まばたきをする」「口を開ける」「笑顔になる」の 3 入力それぞれに関して、検知条件などを解説していきます。

まず、「まばたき」の検出に関しては、右目と左目における「まばたきの程度（eyeBlink）」の平均値に加えて、補助的に「目の細め具合（eyeSquint）」の平均値も用いています。これらの平均値のうち最大値を使用することで比較的高精度で「まばたき」を検出することができました。

次に、「口を開ける」の検出には「あごの開き具合（jawOpen）」と「口の開き具合（1 - mouthClose）」の二つを用いています。「あごの開き具合（jawOpen）」は数値が出ないことがあったので、予備の条件として「口の開き具合（1 - mouthClose）」を使用しています。

最後に、「笑顔」の検出には「左右の口角の上がり具合（mouthSmileLeft/Right）」を用いたうえで、予備の条件として、「左右の口角の引き上げ量（mouthCornerPullLeft/Right）」も用いています。残念ながら口角を動かさずに笑う人の笑顔は検出できなさそうですね。

これらに加え、検出の ON/OFF に異なるしきい値を設けるヒステリシスにより、誤検知やチャタリングを抑制しています。

6.4　制作ゲームの紹介

表情認識を用いて制作したゲームのうち、代表的な 2 作を紹介します。

6.4.1　Runner

当研究会のマスコットキャラクター「KaiRA 君」を操作し、迫り来る障害物を避ける横スクロールアクション。「口を開ける」とジャンプし、タイミングよく障害物を回避します。どこか某ブラウザの恐竜ゲームを想起させる内容です。

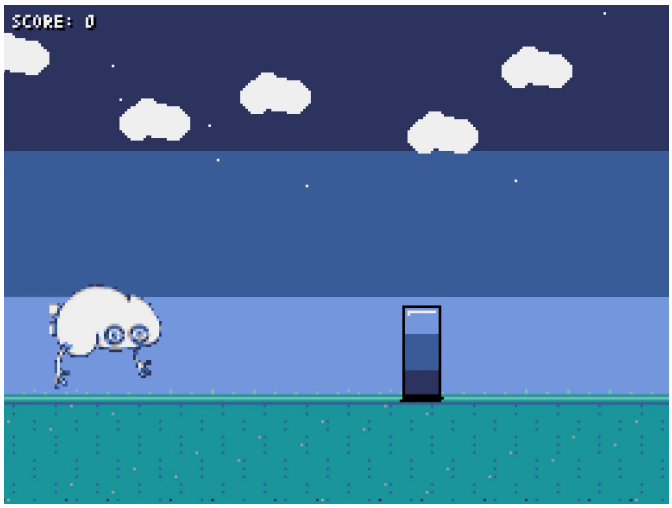


図 6.3: runner ゲームのプレイ画面

6.4.2　Speed React

次々と提示される状況に対して、素早く正確なリアクションを求められる独創的なゲーム（バカゲー）。「笑顔」「驚いた顔」「真顔」のうち状況に合ったリアクションを行います。2 人対戦が可能な「VS React」も用意しています。

図 6.4: Speed React ゲームのプレイ画面

6.5　おわりに

本プロジェクトでは、表情認識を用いたゲーム操作を通じて、人間とコンピュータの新しい関わり方の可能性を検討しました。実装を進めるなかで強く感じたのは、表情を入力として扱うことは想像以上に難しいという点です。その理由のひとつは、表情がしばしば意図的な操作ではなく、無意識の反応として現れる情報だからです。

キーボードやマウスのような明確な「命令」とは異なり、表情は感情や注意の変化、緊張や驚きといった非言語的で曖昧な状態をそのまま映し出します。したがって、表情をインタフェースとして用いることは、単なる操作系の設計を超えて、人間の身体性や無意識との向き合い方を考える営みでもあると感じました。

しかし同時に、もし将来「無意識のコンピューティング」とでも呼べるような新しいインタラクションの枠組みが成立するとすれば、そのもっとも素朴で原初的な入口は、表情によるインプットなのかもしれません。意図と言語を介さず、身体が自然に発する信号をそのまま拾い上げることができる点で、表情は人間の奥深いレイヤーに最も近い情報だからです。

今回の試みは小さな実装にすぎませんが、ヒューマン–コンピュータ・インタラクションがより「人間そのもの」に寄り添う方向へ進む可能性を垣間見ることができました。今後は、表情だけでなく姿勢や視線、呼吸などのマルチモーダルな信号を統合することで、より自然で直感的なインタラクションが実現できると考えています。本プロジェクトがその一歩となれば幸いです。

第 7 章

歩く KaiRA 君：強化学習を用いた二足歩行ロボットの歩行制御

7.1　はじめに

近年、強化学習によってロボットの自律的な動作が可能になりつつあります。
特にロボット分野では、学習したモデルで外部環境を認識し、凹凸路や滑り面での歩行・バランス回復を組み立てるといった事例が増えています。そして、それを応用して災害時の救助用ロボットが開発されたりしております。
今回は、歩く KaiRA 君：強化学習を用いた二足歩行ロボットの歩行制御について解説します。

7.2　学習目標

本プロジェクトでは、Unity ML-Agents を用いて、二足歩行ロボット（KaiRA くん）をシミュレーション環境で学習させる。
学習環境には目標オブジェクトが存在し、図 7.1 の矢印の方向に置かれている。
KaiRA くんは、4 つの関節（左右の上脚・下脚）を持ち、できるだけ早く目標に向かって歩行することを目的とする。
この目的を達成するために、観測情報を基に行動（関節の目標回転角と駆動強度）を決定し、報酬関数と PPO[19] を用いて学習を進める。

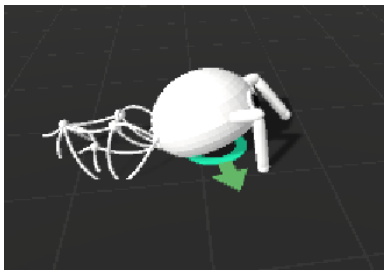


図 7.1: KaiRA くんの概観

7.3　学習フロー

本節では、二足歩行ロボット（KaiRA くん）をシミュレーション環境で学習させるための具体的な構成と、学習を導くための報酬設計について述べる。学習フローは以下の通りである。

1. **観測（Observations）**：エージェント（KaiRA くん）は、環境と自身の状態に関する情報を観測する。
2. **行動（Actions）**：観測情報に基づいて、行動方策（ニューラルネットワーク）によってエージェントの関節の目標回転角と駆動強度を決定する。
3. **環境への適用**：決定された行動が物理エンジンに適用され、KaiRA くんの動作がシミュレーションされる。
4. **報酬の取得（Reward）**：目標に向かって歩行する度合いに基づいて報酬を受け取る。
5. **学習の更新**：収集された観測、行動、報酬のデータを用いて、エージェントの行動方策が PPO によって更新される。

7.3.1　観測

エージェントは、環境と自身の状態に関する合計 24 次元の情報を観測し、これをニューラルネットワークへの入力とする。

- **目標との関係（11 次元）**:
  - 目標速度と現在の平均速度の差の絶対値（1 次元）
  - 胴体の安定化された座標系から見た現在の平均速度ベクトル（3 次元）
  - 安定化された座標系から見た目標速度ベクトル（3 次元）
  - 目標方向と胴体の現在の向きのズレを表すクォータニオン情報（4 次元）
- **目標位置**: 安定化された座標系からの静止した目標の相対位置（3 次元）
- **環境情報**: 胴体直下の地面までの正規化された距離（Raycast による、1 次元）
- **身体情報（9 次元）**：各ボディパーツ（胴体、左右の上脚・下脚の計 5 パーツ）の地面との接触状態（5 次元）および、左右の上脚・下脚（計 4 パーツ）の現在の関節の駆動強度（4 次元）

クォータニオンとは

クォータニオンは、3 次元空間での回転を「どの軸を元に」「どれだけ回転するか」を表現する数学的な方法です。
図 7.2 で表すと、ベクトル r' はベクトル r を軸 $\vec{n}$ の周りに θ だけ回転させたものであり、これを表すには軸の情報 (3 次元) と回転量 (1 次元) が必要です。
そして、この情報を $q = (q_0, q_1, q_2, q_3) = (n_x \sin\frac{\theta}{2}, n_y \sin\frac{\theta}{2}, n_z \sin\frac{\theta}{2}, \cos\frac{\theta}{2})$ の 4 次元ベクトルで表現します。

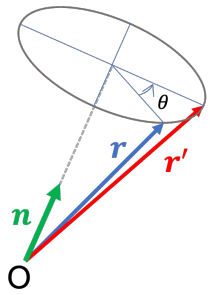


図 7.2: クォータニオンのイメージ

7.3.2 行動

エージェントは、二足ロボットの 4 つの主要関節（左右の上脚・下脚）に対して、合計 10 次元の連続的な行動信号を出力する。

- **関節の目標回転（6 次元）**：左右の上脚はそれぞれ 2 軸、下脚はそれぞれ 1 軸の目標回転角を決定する。これによりロボットの姿勢と歩幅が設定される。
- **関節の駆動強度（4 次元）**：各関節の可動域を保ちつつ、目標とするトルクを達成するための最大の力（トルク）の上限値を決定する。

駆動強度と駆動力の定義

強化学習における「駆動強度」と「駆動力」は、物理エンジン（Unity の Joint Drive）の制御における異なる要素を指す。

- **駆動強度 (Joint Strength / Max Force Limit)**: エージェントが出力する行動信号（4 次元）が直接設定する値であり、関節モーターが目標角度に到達するために出力できる**最大の力（トルク）の上限**を意味する。この値を動的に調整することで、関節を「柔らかく」（低強度）または「硬く」（高強度）制御し、着地時の衝撃吸収や地面を蹴る際の反発力を調整する戦略を学習する。
- **駆動力 (Joint Force / Torque)**: 実際にエージェントの関節に加える力やトルクである。設定された**駆動強度の制限内**で、物理エンジンによって動的に計算される。

この制御により、エージェントは、単に目標の姿勢をとるだけでなく、その姿勢を達成・維持するための**関節の「剛性」**をも同時に学習対象とすることができる。

7.3.3 ニューラルネットワーク構造

観測の正規化

観測データは、まず 2 つの正規化層を通過する。これにより、学習時に収集されたデータの平均と標準偏差を用いて、観測値が正規化（平均 0、分散 1）され、学習の安定性が高められる。

ネットワークの構成

正規化された観測ベクトルは、以下の 3 つの全結合層からなるボディエンコーダーに渡され、行動を決定するための潜在表現（特徴量）が抽出される。

1. **第 1 層**: 入力観測を 512 ユニットの中間特徴量に変換する。
2. **第 2 層**: 512 ユニットを維持し、さらに深い特徴表現を学習する。
3. **第 3 層**: 512 ユニットを維持し、最終的な潜在表現を生成する。

活性化関数

各全結合層の出力には、**Sigmoid** 活性化関数が適用されている。一般的に強化学習では ReLU や Tanh が用いられることが多いが、ここでは Sigmoid が採用されており、出力が 0 から 1 の範囲に制限されることで、行動の選択に対して異なる非線形性を導入している。

行動の出力

ボディエンコーダーから得られた最終的な特徴量は、行動モデルに入力され、連続的な行動（目標関節角度および駆動強度）が出力される。

- **行動の平均 (μ)**: 最終層の出力は、行動の平均値（‘mu‘）となる。
- **行動の分散 (σ)**: 同時に、行動の分散（‘log_sigma‘）が出力され、探索の度合い（ランダム性）を決定する。

これらの平均（μ）と分散（σ）を用いて、最終的な連続行動がサンプリングされる。

7.3.4　報酬関数

KaiRA くんの行動を評価し、目標とする歩行を学習させるために、複数の要素を組み合わせた複合的な報酬関数が用いられている。KaiRA くんの目標は「目標速度で、目標の方向を向いて歩くこと」である。

- **目標速度マッチング報酬 (Reward_speed)**:KaiRA くんの全パーツの平均速度ベクトルと、目標とする速度ベクトル（目標方向 × 目標速度）との差が小さいほど高報酬となる。以下の Sigmoid 形状のカーブを用いて計算される。

$$\text{Reward\_speed} = \left(1 - \left(\frac{|\vec{v}\_\text{actual} - \vec{v}\_\text{goal}|}{\text{TargetWalkingSpeed}}\right)^2\right)^2$$

  ここで、$\vec{v}$_actual は平均速度、$\vec{v}$_goal は目標速度ベクトルである。速度が完全に一致した場合に 1 に近づく。
- **目標方向への向き報酬 (Reward_direction)**: 胴体（‘body.forward‘）の向きが、目標とする進行方向（‘cubeForward‘）にどれだけ一致しているかを評価する。内積を用いて計算され、完全に一致した場合に 1 となる。

$$\text{Reward\_direction} = (\vec{d}\_\text{goal} \cdot \vec{d}\_\text{body} + 1) \times 0.5$$

- **総合報酬**: これらの報酬を乗算することで、「**目標速度を保ちつつ、目標方向へ向かう**」という

複合的な目標を同時に追求させる。

$$\text{Total Reward} = \text{Reward\_speed} \times \text{Reward\_direction}$$

- **接触報酬**: 目標オブジェクトに接触した場合は、ボーナス報酬 (+1.0) が加算される。

この報酬を時点 t で得られる即時報酬 r_t として、$G_t = \sum_{k=0}^{t-L} \gamma^k r_{k+L}$ で表される累積報酬を最大化するように、エージェントは PPO を用いて行動方策を学習していく。

7.4　学習結果

本プロジェクトでは、Unity ML-Agents を用いて KaiRA くんの二足歩行を学習させました。学習は 10^7 ステップにわたり実行され、きちんと目標に向かって歩行できるようになりました。
図 7.3 は学習の進行に伴う累積報酬の推移を示しています。学習初期には報酬が低く、ランダムな動作が多かったですが、学習が進むにつれて報酬が増加し、安定した歩行が実現されました。

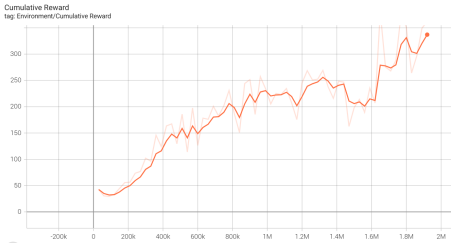


図 7.3: 累積報酬の推移

7.5　おわりに

本プロジェクトでは、Unity ML-Agents を用いて KaiRA くんの二足歩行を試していきました。平面での学習は成功したのですが、斜面や段差などの複雑な地形ではうまく歩行できませんでした。今後は、平面・障害物の状況を画像を用いて捉えることで、そういった地形にも対処できるような学習を目指していきたいと考えています。

参考文献

[1] Tatsuya Sagawa and Ryosuke Kojima. Reactiont5: a pre-trained transformer model for accurate chemical reaction prediction with limited data. *Journal of Cheminformatics*, Vol. 17, No. 1, p. 126, 2025.

[2] Y Gal and Z Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. arxiv [stat. ml]. *Retrieved fro http://arxiv. org/abs/1506.02142*, 2015.

[3] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks, 2021.

[4] KaiRA. 京大シラバス検索 rag システム, 2024.

[5] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023.

[6] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023.

[7] Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models, 2023.

[8] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners, 2023.

[9] 京都大学. Kulasis-シラバス一覧. `https://www.k.kyoto-u.ac.jp/external/open_syllabus/all`, 2025. Accessed: 2025-11-10.

[10] 京都大学. 各授業の平均履修登録者数. `https://www.kyoto-u.ac.jp/sites/default/files/inline-files/10-1-fd74b48ede8655ef7a87d79175a75c51.pdf`, 2024. Accessed: 2025-11-10.

[11] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus, 2017.

[12] Benjamin Elizalde, Soham Deshmukh, Mahmoud Al Ismail, and Huaming Wang. Clap learning audio concepts from natural language supervision. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5. IEEE, 2023.

[13] Kei Sawada, Tianyu Zhao, Makoto Shing, Kentaro Mitsui, Akio Kaga, Yukiya Hono, Toshiaki Wakatsuki, and Koh Mitsuda. Release of pre-trained models for the Japanese lan-

guage. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pp. 13898–13905, 5 2024. https://arxiv.org/abs/2404.01657.

[14] Tianyu Zhao and Kei Sawada. rinna/japanese-roberta-base.

[15] Google AI Edge – MediaPipe Team. 顔ランドマーク検出ガイド — mediapipe solutions. https://ai.google.dev/edge/mediapipe/solutions/vision/face_landmarker?hl=ja, 2024. 最終更新日: 2024-05-21 UTC.

[16] Valentin Bazarevsky, Yury Kartynnik, Andrey Vakunov, Karthik Raveendran, and Matthias Grundmann. Blazeface: Sub-millisecond neural face detection on mobile gpus. *arXiv preprint arXiv:1907.05047*, 2019.

[17] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[18] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.

[19] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[20] Umberto Michelucci. An introduction to autoencoders, 2022.

[21] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning, 2018.

[22] Ajay Jain Jonathan Ho and Pieter Abbeel. Denoising diffusion probabilistic models, 2020.

[23] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[24] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Bjorn Ommer. High-resolution image synthesis with latent diffusion models, 2021.

本会誌について

執筆者

第 1 章：ニューラルネットワーク入門　岡本和優
第 2 章：画像生成入門　宮前明生
第 3 章：事前学習済み Transformer を用いた化学反応収率のベイズ最適化　岡本和優
第 4 章：Agentic な京大シラバス検索・時間割作成システム　千葉一世、神原みちる
第 5 章：音楽と自然言語の類似度測定システムの開発　稲葉陽孔、佐藤海里
第 6 章：顔認識で操作するゲーム　岡本和優、坂本竜弥
第 7 章：歩く KaiRA 君：強化学習を用いた二足歩行ロボットの歩行制御　稲葉陽孔

京都大学人工知能研究会 KaiRA

代　　表　岡本和優
活動日時　毎週木曜日 18:40〜21:00、毎週土曜日 10:00〜12:00
活動場所　文学部教室・株式会社 Athena 京都オフィス（百万遍ビル 3F）など
入会資格　大学、学部、回生問わず、AI に興味がある方
会　　費　無料
活動内容　機械学習に関する本の輪読会、コード読み・実装会、kaggle への参加など

会誌 vol.9

発行日	2025 年 11 月吉日　初版発行
発行	京都大学人工知能研究会 KaiRA
公式サイト	https://kyoto-kaira.github.io/
メールアドレス	kyoto.kaira@gmail.com
X(旧 Twitter)	https://x.com/kyoto_kaira