情報理工学特別研究報告書

題目

深層学習を用いた
スライドパズルの最短経路探索に関する研究

学生証番号　153326

氏名　勝山幸祐

提出日　令和 6 年 1 月 31 日

指導教員　蚊野　浩

京都産業大学
情報理工学部

要約

本研究では，深層学習を用いてスライドパズルの最短経路を求める手法を提案した．従来の A*探索アルゴリズムを用いた手法は最短経路を求めることができるが，計算量が大きく，大きなスライドパズルに適用する際の実用性に課題がある．そこで，本研究では，学習済みニューラルネットワークを利用して，スライドパズルの最短経路を求めるアプローチを採用した．

本研究では，学習データの生成に A*探索を用い，多様な初期盤面から最短経路を求めるデータセットを作成した．さらに，学習データの偏りを抑えるため，全ての可能な盤面を網羅し，解決可能な状態のみを対象とした学習データを生成した．ニューラルネットワークモデルには畳み込み層を含む構造を採用し，盤面情報をワンホットエンコーディングにより適切に数値化し，最適な移動方向を高精度で予測することを目指した．

実験の結果，最終的なモデルは約 99%の正答率を達成し，最短手数が長い盤面（最長 31 手）に対しても高い精度で解を求めることができた．また， A*探索と比較した際，短い手数の盤面では A*探索が高速であったが，長い手数の盤面では学習済みモデルの方が高速に解を求められることが確認された．

本研究の成果により，スライドパズルの解法において深層学習の有効性が示された．今後の課題として，モデルの精度向上や他の探索アルゴリズムとの比較を進めることで，より汎用的な解法の開発が期待される．

目次

1章 序論

スライドパズルは，空所を使って駒をスライドさせ，目的の配置に揃えるシンプルで奥深い知育玩具として知られている．このパズルは，一見単純に見えるが，最短ルートを見つけることはかなり困難である．

スライドパズルの解法として，A*探索アルゴリズムを用いる手法が知られている．この手法はヒューリスティック値とコスト値を使った評価値を基準に最短経路を計算する手法である．しかし，このアプローチは計算負荷が高く，大規模な問題や最短経路が長い盤面において実用性が限られるという課題がある．

そこで本研究では，従来手法の枠を超えた新しいアプローチとして，学習済みニューラルネットワークを用いてスライドパズルの最短経路を直接予測する方法を提案する．このアプローチは，複雑な計算を不要とし，モデルが瞬時に解法を提案できることを目指している．さらに，スライドパズルの全ての盤面に対して最適な解を予測する精度を評価し，従来手法との性能比較を行う．

2 章　スライドパズルとその解法

2.1　スライドパズル

スライドパズルの典型的な例として，3×3 の盤面を用いた「8 パズル」が挙げられる．8 パズルでは，3×3 の盤面上に配置された数字タイルを空白のマスを利用して上下左右に移動させ，特定のゴール配置を目指す．例えば，図 2.1 のように初期状態が「[5，6，4，8，7，3，1，0，2]」の場合，最終的に図 2.2 のような「[1，2，3，4，5，6，7，8，0]」となるのがゴールである（「0」は空白マスを表す）．空白マスを上下左右に動かすことでタイルの位置を変更し，ゴール状態に到達することがこのパズルの目的である．初期状態からゴール状態に最短の手数で達する経路を本パズルの最短経路とよぶ．

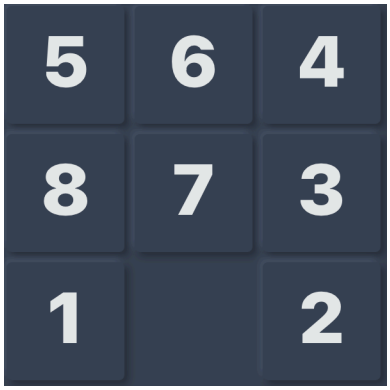

図 2.1 初期状態

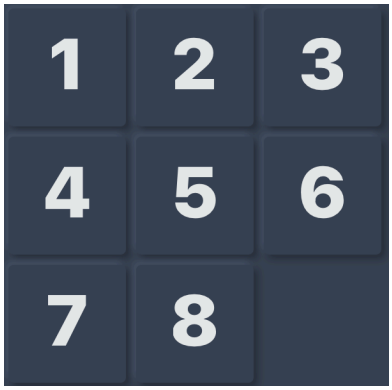

図 2.2 ゴール状態

スライドパズルの特性として，盤面の初期状態やサイズに応じて膨大な数の配置が存在する．８パズルの場合は，362，880 通り（9!）の配置が存在し，その中で，181，440 通りの配置（9!/2）に解が存在する．

2.2　探索アルゴリズム

スライドパズルを解くための従来手法として，A*探索アルゴリズムが広く用いられている．A*探索は，最短経路問題を解くための効率的な手法であり，探索空間内で最適な解を見つけることを保証する特性を持つ．このアルゴリズムでは，各状態（盤面）に対して評価値 f(n)を計算し，評価値が最小となる状態を探索の優先候補とすることで，効率的な探索を実現する．

A*探索アルゴリズムにおける評価値 f(n)は，以下のように定義される．

$$f(n) = g(n) + h(n)$$

g(n)：現在の状態 n に到達するまでにかかったコスト（移動手数），h(n)：現在の状態からゴール状態までの推定コスト（ヒューリスティック値）を表している．この評価値は，探索の効率性と精度を左右する重要な要素であり，特にヒューリスティック値 h(n)の設計がアルゴリズムの性能に大きな影響を与える．

スライドパズルにおけるヒューリスティック値として，2 種類の方法が一般的に使用される．１つは現在の盤面でゴールの盤面と異なる位置に配置されているタイルの数をカウントする方法である．今回使用した方法は次に説明する方法であるため，この方法の具体的な説明は省略する．そして今回使用した方法はマンハッタン距離と呼ばれるものである．この方法は各タイルが現在の位置からゴール位置に到達するために必要な水平および垂直方向の移動距離の合計の値である．例えば，図 2．3 のように現在の盤面でタイル「3」が位置(2，2)にあり，ゴール位置が(1，3)の場合，そのマンハッタン距離は|2-1| + |2-3| = 2 となる．

	3	

図 2.3 現在の盤面

1	2	3
4	5	6
7	8	0

図 2.4 ゴールの盤面

A*探索アルゴリズムは，この評価値 f(n)を基に，最も効率的な経路を探索しながらゴール状態を目指す．その結果，初期状態からゴール状態までの最短経路を求めることが可能である．

3 章　ニューラルネットの構造と学習データの準備

3.1　ニューラルネットワークモデルの構造

スライドパズルの次の一手を予測するためにニューラルネットワークモデルを構築した．このモデルは，スライドパズルの盤面情報を入力として受け取り，次に取るべき最適なアクション（上，下，左，右）を確率分布として出力する仕組みとなっている．図 3.1 は本研究で使用したニューラルネットワークの簡易図である．

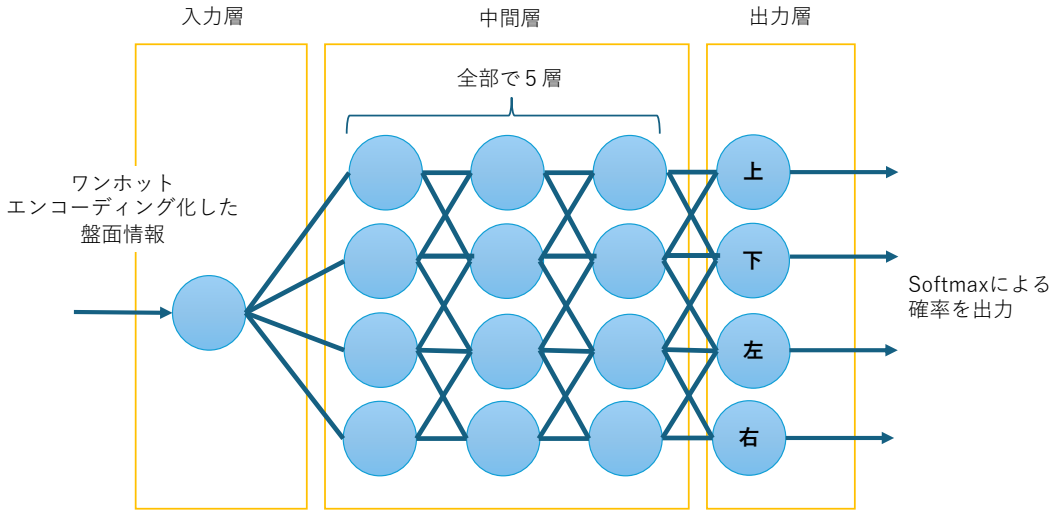

図 3.1 ニューラルネットワーク

スライドパズルのタイル番号をそのまま入力すると、ネットワークが「3 は 4 より小さい」「0 は特別」といった数値的の関係を学習してしまう可能性がある。しかし、スライドパズルではタイルの数値の大小ではなく、位置関係が重要である。例えば、タイル「1」の右に「2」があることが問題なのであり、「1」と「2」の大小関係は本質ではない。そこで、本研究ではワンホットエンコーディングを用いて各タイルを独立したカテゴリとして表現し、数値的な関係を排除した。ワンホットエンコーディングとは，タイル番号をベクトル形式に変換する方法であり，各タイル番号をそれぞれ固有の位置に「1」を立て，他の位置には「0」を割り当てる．例えば，3×3 の盤面において，タイル番号「1」から「8」と空白「0」を以下のようにエンコードする．

タイル「1」　→　[1, 0, 0, 0, 0, 0, 0, 0, 0]
タイル「2」　→　[0, 1, 0, 0, 0, 0, 0, 0, 0]
タイル「0」（空白）→　[0, 0, 0, 0, 0, 0, 0, 0, 1]

この変換により，各タイル番号が独立したカテゴリとして認識され，数値間の大小関係をモデルが誤って学習することを防ぐことができる．

具体例として，初期盤面が図 3.2 の配置だった場合を考える

1	2	3
4	5	6
7	8	0

図 3.2 初期盤面

この盤面をワンホットエンコーディングに変換すると，各マスが 9 次元のベクトルで表現されるため，入力データ全体は図 3.3 のような形になる

[1, 0, 0 ,0, 0, 0, 0, 0, 0]	[0, 1, 0 ,0, 0, 0, 0, 0, 0]	[0, 0, 1 ,0, 0, 0, 0, 0, 0]
[0, 0, 0 ,1, 0, 0, 0, 0, 0]	[0, 0, 0 ,0, 1, 0, 0, 0, 0]	[0, 0, 0 ,0, 0, 1, 0, 0, 0]
[0, 0, 0 ,0, 0, 0, 1, 0, 0]	[0, 0, 0 ,0, 0, 0, 0, 1, 0]	[0, 0, 0 ,0, 0, 0, 0, 0, 1]

図 3.3 ワンホットエンコーディング後

この変換後のデータは，モデルに対してサイズ(3, 3, 9)のテンソルとして入力される．この形式により，モデルは各タイルの配置や空白マスの位置を正確に認識し，それらの関係を基に最適なアクションを予測できるようになる．

中間層は，入力層から渡されたデータを処理し，高次元の特徴を抽出する役割を担っている．本研究では，中間層には畳み込み層を 3 層，全結合層を 2 層の計 5 層を配置した．それぞれの層には 63 個のノードを設定し，ReLU（Rectified Linear Unit）活性化関数を適用している．ReLU は，入力値が 0 未満の場合に 0 を出力し，それ以外ではそのままの値を出力する関数であり，データに非線形性を加える役割を果たす．この非線形性により，モデルは単純な線形分離では対応できない複雑なパターンを学習できるようになる．各全結合層では，前の層からの出力が次の層の入力として利用され，盤面情報の特徴が徐々に抽象化される．このプロセスを通じて，モデルは盤面における駒の配置や空白マスの位置関係を効果的に捉え，次の一手を予測するために必要な重要な特徴を抽出する．

出力層では，スライドパズルの次に取るべきアクション（上，下，左，右）をそれぞれの確率として出力する．この層には 4 つのノードを配置し，それぞれのノードが特定のアクションに対応する．また，出力層には Softmax 関数を使用しており，各ノードの

出力を確率分布として正規化している．この仕組みにより，モデルは各方向に対応する選択確率を算出し，その中で最も高い確率を持つアクションを選択する．例えば，ある盤面情報を入力した際に Softmax 関数が「上：0．7，下：0．1，左：0．1，右：0．1」といった出力を生成した場合，モデルは最適なアクションとして「上」を選択する．この仕組みを通じて，提案モデルはスライドパズルの次の一手を効率的かつ正確に予測する．

3.2　学習データの生成方法 1

深層学習では質の良い学習データを準備することが重要である．本研究における第一段階として，次に述べる方法で初期盤面を生成し，その盤面に対して A*探索アルゴリズムを適用して学習データを作成した．この学習データは，各盤面からゴール状態に至る最短経路およびその盤面遷移を含むデータセットである．

まず初期盤面を生成するためにスライドパズルの基本的なルールや動作を記述したプログラム(game．py)を作成した．ここでの基本的なルールや動作とは，空所を利用し駒は移動可能，駒は３×３の枠内でのみ移動可能，ゴール状態を目指し，駒を移動させるなどである．作成したプログラムを使用し初期盤面を生成する．初期盤面の生成方法は，図 3.4 で示すように，ゴール状態のスライドパズルから駒をランダムに上下左右に動かし，この動作を複数回にわたり行うことで初期盤面を生成する．

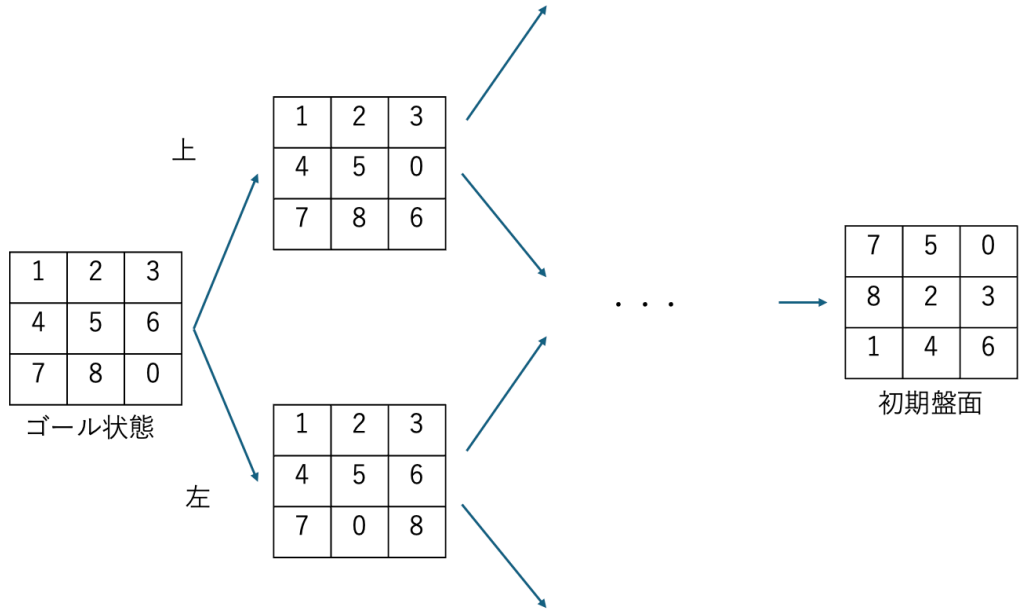

図 3.4 学習データの生成方法 1 において初期盤面を生成する方法

生成された初期盤面に対しては，A*探索アルゴリズムを適用し，ゴール状態への最短経路およびその過程における盤面遷移を記録した．このアルゴリズムにより，初期盤面からゴール状態までの最短経路を正確に算出できる．

さらに，探索の過程で得られた各盤面遷移（盤面状態とその際に取られたアクションのペア）を，学習データとして記録した．具体的には，次の一手として取るべきアクション（上，下，左，右）をラベルとして付与し，盤面情報とアクションのペアを構築した．このデータは，ニューラルネットワークモデルの学習において，入力データ（盤面情報）と出力データ（次のアクション）の関係を学習するために利用される．

学習データの総量として 100,000 個の初期盤面を用意し，その状態遷移データをモデルの訓練に活用した．しかし，この学習データには同じ盤面が複数回含まれる可能性がある。また初期のデータ生成過程では，特定の盤面が偏って含まれることが確認されたため，これを改善するための工夫を 3．3 にて後述する．

3.3　改善後の学習データの生成方法 2

スライドパズルの最短経路を予測するモデルの精度向上を目指し，学習データの生成方法やニューラルネットワーク（NN）モデルの構造を繰り返し改良した．

精度向上に向けて，まず学習データの生成方法を見直した．生成方法 1 では，ランダムに生成した初期盤面に対して A*探索アルゴリズムを適用し，ゴール状態への盤面遷移とその時のアクションをセットにしたデータを使用していた．しかし，この方法では特定の盤面や簡単な盤面にデータが偏るという課題があった．

改良後の生成方法 2 では，すべての盤面（9! = 362，880 通り）を列挙し，それぞれの盤面がゴール状態に到達可能かどうかを判定した．スライドパズルでは，交換回数が偶数回である場合のみゴール状態に到達可能であるという数学的性質がある．このため，盤面をゴール状態へ変換する際に必要なタイルの交換回数を計算し，偶数回であれば完成可能，奇数回であれば完成不可能と判定した．例えば，初期状態が(3，2，7，4，1，6，5，8，9)である場合，次のような交換を行うことでゴール状態に到達できるかを判定する．

1. 3 と 1 を交換 →（1，2，7，4，3，6，5，8，9）
2. 7 と 3 を交換 →（1，2，3，4，7，6，5，8，9）
3. 7 と 5 を交換 →（1，2，3，4，5，6，7，8，9）（完成）

この場合，交換回数は 3 回（奇数）であるため，この初期盤面はゴール状態に到達不可能であると判断される．一方で，交換回数が偶数回の場合は完成可能と判定し，その盤面を学習データとして使用した．

この判定を全ての盤面に適用し，完成可能な盤面のみを対象として A*探索アルゴリズムを適用し，ゴール状態までの最短経路を取得した．その際，初期盤面と最初の 1 手目のアクション（上下左右のいずれか）をペアとして記録し，学習データに含めた．

精度向上に向けて，ニューラルネットワークモデルの構造も改良した．中間層のノード数を従来の 64 ノードから 128 ノードに増加させることで，モデルの表現力を向上させた．ノード数の増加により，モデルはより多くの特徴を学習可能となり，盤面の複雑なパターンを捉える能力が向上した．

生成方法２では、すべての盤面に対して次の最適手を学習データとして与えるので、学習データがはっきりとしている。生成方法１に関しては、学習データの作り方は書いてあるが、学習データの個数が書いていない。その中には同じ学習データもあるはずで、これらの個数も説明する必要がある。

4 章　実験と考察

4.1 実験条件と手法

本研究では，提案したニューラルネットワークモデルと学習データの性能を評価するために，スライドパズルの一種である 3×3 の 8 パズルを対象として実験を行った．

本実験では，テストデータとして，ランダムに生成した初期盤面を使用した．このテストデータは，最短手数が短い盤面から長い盤面まで幅広い難易度を含むように設計した．これにより，提案モデルが単純な問題と複雑な問題の両方に適切に対応できるかどうかを評価した．

モデルの性能評価には，いくつかの基準を用いた．１つは，完成していない初期盤面からゴール盤面まで，モデルがどのような動かし方をし，何手で完成したかターミナル上に出力した．例としては図 4.1 のようになる．

```
Initial State:
 1  5  2
 4  3  6
 7  8  0

After action U:
 1  5  2
 4  3  0
 7  8  6

After action L:
 1  5  2
 4  0  3
 7  8  6

After action U:
 1  0  2
 4  5  3
 7  8  6

After action R:
 1  2  0
 4  5  3
 7  8  6

After action D:
 1  2  3
 4  5  0
 7  8  6

After action D:
 1  2  3
 4  5  6
 7  8  0

Puzzle solved!
Action : U, L, U, R, D, D
Step : 6
Optimal Steps (A*): 6
```

図 4.1 モデルの性能確認

図 4.1 は初期盤面（initial State）に対して，次に動かす方向（After action）と動かした時の盤面を出力している．また After action には U（上），D（下），L（左），R（右）の 4 方向のうちからモデルがゴール状態に近づくと判断したアクションが選択される．そして Step には完成までに動かした回数が記録され，Optimal Steps(A*)には A*

探索が導き出した最短手数が出力される．これによりスライドパズルの状態遷移を確認しながら，モデルと A*探索の最短手数を比較することができる．

もう一つの評価方法は正答率を評価基準とし，モデルが正確に最短手数を予測できた割合を測定した．この指標により，モデルの予測精度全体を把握することができる．

これらの実験条件と評価基準を通じて，提案したニューラルネットワークモデルの性能を評価することを目指した．

4.2　実験結果

提案したニューラルネットワークモデルの性能を評価するために，学習データとニューラルネットの改良前後におけるモデルの正答率や，複雑な盤面（最短手数が 20 手以上）に対する精度を比較した．また，スライドパズルを完成させるまでの時間を A*探索のみを用いた時とモデルのみを用いた時それぞれ測定し，比較した．

まず改良前のモデルを評価した．図 4.2 は完成していないスライドパズル(Total Trials)200 個に対して，完成したスライドパズル(Match Count)の数が 142 個だったことを表している．

図 4.2 学習済みモデルの精度

図 4.2 より最短手数で求めることができたものは 7 割程度しかなく，精度としてはあまり良いとは言えない．

また図 4.3 は最短手数が 20 手のものをモデルに与えた時の出力結果である．

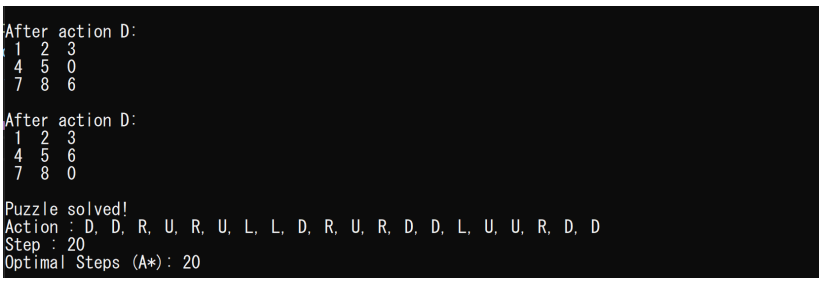

図 4.3 最短手数が 20 手の盤面にたいするモデルの予測結果

図 4.3 よりモデルが最短手数の 20 手でスライドパズルを完成させていることがわかる．

それに対して図4.4は最短手数が23手のものをモデルに与えた時の出力結果である.

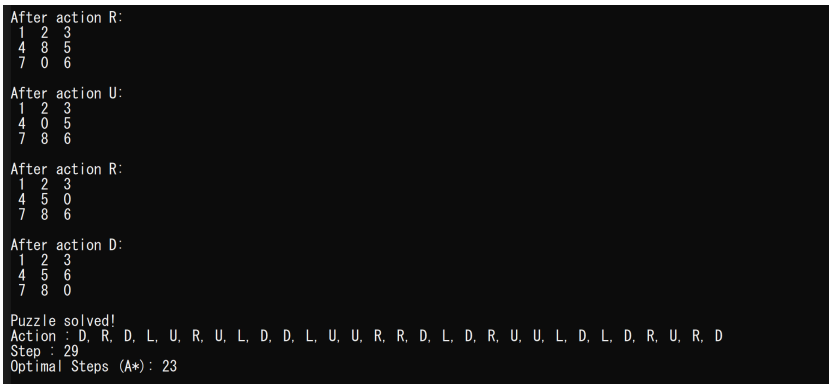

図 4.4 最短手数が 23 手の盤面にたいするモデルの予測結果

図 4.4 よりモデルが最短手数の 23 手より多い手数でスライドパズルを完成させていることがわかる. 同様の実験を行い, 21 手を超えると途端に精度が悪くなっていることが確認した.

最終モデルでは, 改良後のデータ生成方法２を用いた学習を繰り返し行い, 精度の確認をした. 図 4.5 は完成していないスライドパズル(Total Trials)1000 個に対して, 完成したスライドパズル(Match Count)の数が 992 個だったことを表している.

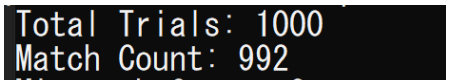

図 4.5 改良後のモデルの精度

図 4.5 よりテストデータ全体に対する正答率は 99.2%を達成し, 改良前と比較して非常に高い精度を示した.

特に, 最短手数が長い盤面における性能向上が顕著であり, 複雑な問題に対しても高い予測精度を実現した. 8 パズルにおいて最も長い最短手数は 31 手とされており, その盤面は以下の図 4.6 と図 4.7 の 2 種類である.

8	6	7
2	5	4
3	0	1

図 4.6 最短手数が 31 手の盤面 1

6	4	7
8	5	0
3	2	1

図 4.7 最短手数が 31 手の盤面 2

図 4.8,　図 4.9 は最短手数が 31 手の盤面に対しても学習済みモデルが 31 手で完成することができるのか検証した図である.

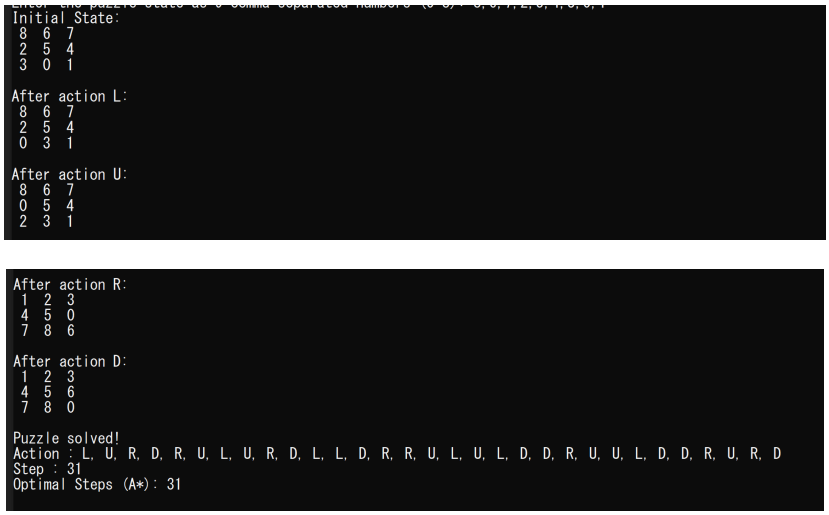

図 4.8 最短手数が 31 手の盤面にたいするモデルの予測結果 1

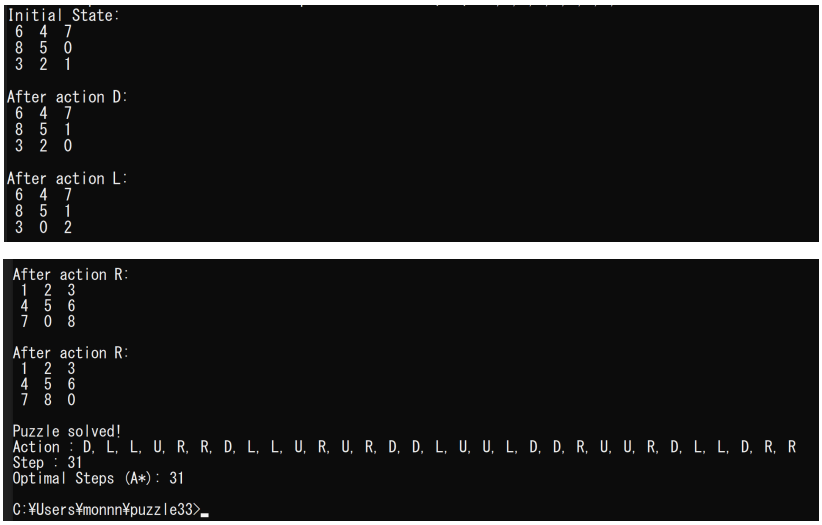

図 4.9 最短手数が 31 手の盤面にたいするモデルの予測結果 2

図 4.8,図 4.9 を見るとわかるよう両方の盤面共に 31 手で完成させることができた. これらの結果から, 学習データの偏りを改善することで, ニューラルネットワークモデルが多様な盤面に対応可能となり, 複雑なスライドパズルの解法にも有効であることが示された.

次にスライドパズルを完成させるまでの時間を従来の A*探索アルゴリズムとモデルそれぞれ測定し比較した.

表 4.1 は実際に A*探索とモデルの計算時間を比較した表である

表 4.1 A*探索とモデルの計算時間

最短手数(回)	A*探索(秒)	モデル(秒)
20	0.16	0.89
31	3.50	1.17

表 4.1 からわかるように 20 手の盤面では A 探索が速く解を求めることができたが，31 手の盤面では学習済みモデルの方が短時間で解を求められることが分かった.

4.3 考察

実験結果を総合すると,学習済みモデルはスライドパズルの解法として有効であることが示された．特に，学習データの改善により，精度が大幅に向上し，最短経路が長い盤面に対しても正確な予測が可能となった．しかし，モデルの精度を 100%まで上げることができなかった．この課題を解決するには，さらに多くのデータセットの学習させることやニューラルネットワークの改良などが考えられる.

また，A*探索との比較において，短い手数の盤面では A*探索が優位であるが，長い手数の盤面では学習済みモデルが高速に解を求められることが明らかとなった．これは，A*探索では盤面が複雑になるにつれて探索の分岐が指数的に増加し，処理時間が長くなるためである．一方，学習済みモデルは推論フェーズにおいて直接次の手を予測するため,長い手数の盤面では A*探索よりも効率的に解を求めることができるからであると考えられる.

5 章　結論

本研究では，スライドパズルの最短経路を予測するための学習済みモデルを提案し，従来の A*探索アルゴリズムと比較した．実験の結果，提案モデルは従来手法に匹敵する精度を達成し，特に長い手数の盤面では A*探索よりも高速に解を求められることが確認された．

また，学習データの偏りを修正することで，モデルの精度向上が可能であることを示した．特に最短手数が長い盤面への対応力が向上し，複雑なスライドパズルの解法としての実用性が高まった．

一方で，全ての盤面に対して必ず最短経路を導き出すことができない．この課題を克服するための改善が今後の研究課題として挙げられる．また，本研究のアプローチはスライドパズルに限らず，他の組合せ最適化問題やゲーム AI にも応用可能であると考えられる．

今後の研究としては，より多様な学習データの作成やモデルの構造の最適化，他の探索アルゴリズムとの比較を進めることで，さらに精度の高い解法の開発が期待される．

参考文献

[1] 8 パズルでグラフ探索(幅優先探索、A*アルゴリズム、IDDFS、IDA*)
https://qiita.com/persimmon-persimmon/items/48bf1b021c349d338f0f

[2] [活性化関数]ソフトマックス関数(softmax function)とは
https://atmarkit.itmedia.co.jp/ait/articles/2004/08/news016.html

[3] 8 パズル、15 パズルの不可解な配置と判定法
https://manabitimes.jp/math/979

[4] 8 パズル -Puzzle DE programming
http://www.nct9.ne.jp/m_hiroi/puzzle/eight.html

[5] ダミー変数(One-Hot エンコーディング)とは？実装コードを交えて徹底解説
https://www.codexa.net/get_dummies/

謝辞

本論文を作成にあたり，丁寧な御指導を賜りました蚊野浩教授に感謝いたします.

付録　開発したプログラムとその説明

game4.py

[内容]

スライドパズルの状態遷移やA*探索を用いた最短経路計算を行う.

[関数]

- one_hot_encode_state ： 状態をワンホットエンコーディングする.
- _create_actions ： 上下左右の移動アクションを定義する.
- get_rev_action ： 指定されたアクションの逆アクションを返す.
- get_random_state： ランダムなアクションを指定回数適用し，ランダムなパズル状態を生成する.
- get_next_state ： 指定したアクションに従って新しい状態を取得する.
- heuristic ： マンハッタン距離を用いたヒューリスティック値を計算する.
- get_valid_actions ： 現在の状態で可能な移動方向を取得する.
- a_star_solve ： A*探索を用いて最短経路を求める.

generata_data.py

[内容]

スライドパズルの学習データを生成し，JSONファイルに保存する.

[関数]

- is_solvable ： 指定したパズル状態が解けるかどうかを判定する.
- generate_training_data ： すべての可能なパズル状態について A*探索を実行し，学習データを作成して保存する.

train_nn.py

[内容]

ニューラルネットワークを学習させ,スライドパズルの最短経路を予測するモデルを構築する.

[関数]

- load_training_data ： JSONファイルから学習データを読み込む.
- residual_block ： 残差ブロック（ResNet風）を構築する.
- create_model ： CNNを用いたスライドパズル解決モデルを作成する.

Current.py

[内容]

ニューラルネットワークを用いてスライドパズルの最短経路を予測し，A*探索との比較を行う．

[関数]

- get_rev_action ： 指定されたアクションの逆アクションを返す．
- decode_action ： 予測されたアクションのスコアを元に，有効なアクションの中から最適なものを選択する．
- get_valid_actions ： 現在のパズル状態から可能な移動方向を取得する．
- render_string ： パズルの状態を文字列として整形して表示する．

solve.py

[内容]

ニューラルネットワークモデルを用いてスライドパズルを解く．

[関数]

- get_rev_action ： 指定されたアクションの逆アクションを返す．
- decode_action ： 予測されたアクションのスコアを元に，有効なアクションの中から最適なものを選択する．
- get_valid_actions ： 現在のパズル状態から可能な移動方向を取得する．
- render_string ： パズルの状態を文字列として整形して表示する．

measure_time.py

[内容]

A*探索とニューラルネットワークモデルの計算速度を比較する．

[関数]

- measure_time ： 指定した方法（A*またはモデル）を複数回実行し，処理時間を測定する．
- solve_with_astar ： A*探索を使用してパズルを解く．
- decode_action ： 予測されたアクションの中から有効なものを選択する．
- solve_with_model ： 学習済みモデルを使用してスライドパズルを解く．