計測自動制御学会東北支部 第 344 回研究集会 (2023.10.7)
資料番号 344-2

スパイク形式による画像の潜在表現を用いたモデルベース強化学習の性能評価

Performance Evaluation of Model-Based Reinforcement Learning using Latent Representation of Images in Spike Encoding

○平野貴也，沓澤京，大脇大，林部充宏

○ Takaya Hirano, Kutsuzawa Kyo, Dai Owaki, Mitsuhiro Hayashibe

東北大学

Thoku University

キーワード： モデルベース強化学習 (Model-based Reinforcement Learning)，
ニューラルネットワーク (Neural Network)，スパイキングエンコーディング (Spinking Encoding)，

連絡先： 〒 980-8579 宮城県仙台市青葉区荒巻字青葉 6-6-01
東北大学 工学研究科 ロボティクス専攻 林部・沓澤/大脇研究室
平野貴也，Tel.: (070)2624-2223，E-mail: takaya.hirano.p5@dc.tohoku.ac.jp

1. 緒言

近年, 画像認識と強化学習によるロボット制御の実用化が期待されている. 画像認識を用いることで, 対象物にセンサを取り付けることなく, 位置や形状等の特徴を検出でき, ロボットアームの把持姿勢等の推定が可能となる [1].

また, 強化学習は従来の手法では制御が困難な非線形システムに対して直接適用できる [2] ため, 期待される活躍の幅が広い. 強化学習手法のひとつであるモデルベース強化学習は, 現在の状態と行動から次の状態を予測する予測モデルを学習する手法である. 制御の際には, 学習済みのモデルを用いたモデル予測制御 (model predictive control, MPC) により, より良い状態になると予測される行動を選択することで制御を行う. モデルベース強化学習は, 訓練データを直接ニューラルネットワークの教師として予測モデルを学習させるため, サンプル効率が良く学習時間が短い特徴がある [3]. また, 学習された予測モデルは特定のタスクに限定されないため, 複数のタスクへの移行が容易である [4]. しかしながら, モデルベース強化学習では訓練データへの依存度が高くなってしまうため, 入力や外部の環境が訓練データに含まれない値になると, モデルバイアスと呼ばれる予測誤差が増大する [3]. 大きなモデルバイアスに影響された行動選択は, ロボットが予期しない制御に移行する可能性を生むため, 制御性能を劣化させる恐れがある [5].

画像認識や強化学習には通常, 人間の脳神経回路を模して数理モデル化された人工ニューラルネットワーク (artificial neural network, ANN) が用いられる. ANN は入力層, 中間層, 出力層の 3 つの層から構成され, 層間の入出力はアナログ値で行われる. ANN は層間の情報の伝達をアナログ値で行う一方で, 実際の神経回路は 0 か 1 のスパイク発火の有無によって情報の伝達を行う. このことから, ANN は単純化された数理モデルであり, 生物的妥当性は低い. そこで, 実際の神経回路をより尤もらしく模したモデルとしてスパイキングニューラルネットワーク (spiking neural network, SNN) がある. SNN では, 入力スパイクをシナプスモデルにより重みづけ、統合しシナプス電流に変換する. シナプス電流は細胞体モデルに入力され, 細胞体の電圧を変化させる. 細胞体は電圧がある閾値以下のときに 0, 閾値を超えると 1 を出力するようにモデル化されるため, SNN では 0 か 1 のスパイク発火の有無を入出力として扱うことが可能となる. SNN の特徴のひとつとしてノイズに対するロバスト性が挙げられる [6]. SNN はアナログ値を 0 か 1 のスパイクに変換して処理を行うため, 学習データの情報が制限され, モデルの過学習

を抑制できると考えられる．汎用的なモデルが生成されることが，SNN のロバスト性のひとつの要因となる．

Zhao 氏らによる先行研究では，ニューラルネットワークに SNN を利用して画像認識を行うことで，画像にガウシアンノイズやインパルスノイズが付加された場合においても，認識精度が低下しづらく，ノイズに対する強い頑健性を示していた [7]．また，Patel 氏らは，ゲーム画面を入力とするブロック崩しゲームにおいて，SNN を用いた Deep Q-learning を行っていた．この先行研究では，一旦通常の ANN でタスクを学習させた後に，学習後のネットワークの重みを SNN の重みに変換することで，ゲーム画面を一部隠したりする等の画面ノイズに対する頑健性を示していた [8]．Thalmeier 氏らは単振り子の予測モデルに SNN を適用していた．SNN を用いた予測モデルは，入力される振子の角度と角速度にノイズが加わった場合でも，振子の状態予測および制御が可能であった [9]．この結果は，予測モデルに SNN を導入したことで汎化性の高いモデルが生成され，ノイズが加わった際のモデルバイアスが抑制されたためと考えられる．Zhao 氏らや Patel 氏らの先行研究で示された通り，SNN は画像入力に対するノイズロバスト性が高い．そのため，SNN を画像処理に利用することで，画像入力にノイズが加わった際の予測モデルのモデルバイアスを低減できると考えられる．

したがって，本研究の目的は，モデルベース強化学習において画像処理に SNN を用いることで，予測モデルのモデルバイアスを低減し，画像ノイズに対して頑健な制御を獲得する事である．この目的のために，SNN で表現されたオートエンコーダによって入力画像から潜在変数を抽出し，それを用いたモデルベース強化学習を行った．その結果，制御の際に入力画像へガウシアンノイズを付加した場合において，SNN を使用した方が従来の ANN を使用するよりも，タスクの評価値の低下を抑えることが可能であった．また，ノイズが加わった際の潜在変数の変化量を比較し，SNN を用いた方がノイズによる影響を低減していることを確認した．

2. 提案手法

2.1 本手法で用いるネットワーク構造

本研究で用いるネットワーク構造は以下の 3 つの構造 (図 1) に分けられる [4]．

1) 入力画像 $\boldsymbol{img_t}$ を潜在状態 $\boldsymbol{h_t}$ に変換するオートエンコーダ E_ϕ, D_ϕ
2) オートエンコーダにより抽出された潜在状態 $\boldsymbol{h_t}$ と行動 $\boldsymbol{a_t}$ から次の潜在状態 $\hat{\boldsymbol{h}}_{\boldsymbol{t+1}}$ を予測する予測モデル f_θ
3) 予測した潜在状態 $\hat{\boldsymbol{h}}_{\boldsymbol{t+1}}$ の価値 $\hat{\boldsymbol{r}}_{\boldsymbol{t+1}}$ を評価する評価モデル R_ψ

制御の際は，今の画像と行動候補をネットワークに入力し，次の潜在状態と評価値を予測する．その後，予測した評価値が高い行動候補を選択し制御を行う．

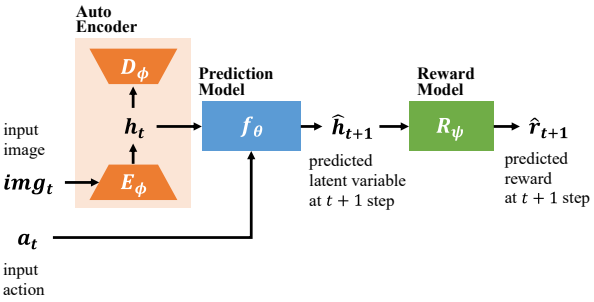

Fig. 1 Network structure

2.2 畳み込みオートエンコーダ

オートエンコーダでは，多層ニューラルネットワークを学習させることで，高次元の入力を圧縮する [10]．また，ニューラルネットワークは圧縮した情報を元の情報に復元できるように学習される．このため，オートエンコーダによる圧縮を用いると，高次元の入力から低次元の特徴量を抽出することが可能となる．

オートエンコーダにおいて，高次元な入力 $\boldsymbol{x}$ を圧縮し，低次元の潜在状態 $\boldsymbol{h}$ に変換するネットワークをエンコーダと呼ぶ．また，圧縮された潜在状態 $\boldsymbol{h}$ からもとの入力 $\boldsymbol{x}$ を復元するネットワークをデコーダと呼ぶ．誤差逆伝播法を用いて，元の入力とデコーダによって復元された値の誤差を最小化することによってネットワークの学習を行う．こうして，学習後のオートエンコーダのエンコーダを用いることで，入力の特徴を捉え，かつ低次元化された潜在変数を扱うことが可能となる．

本研究では，オートエンコーダのニューラルネットワークに畳み込みニューラルネットワーク (convolutional neural network, CNN) を用いる．ニューラルネットワークを CNN としたオートエンコーダは畳み込みオートエンコーダ (convolutional auto encoder, CAE) と呼ばれる．CAE は入力層と出力層の間に，特徴を抽出する畳み込み層とネットワークに移動不変性を付与するためのプーリング層から構成される．

2.3 スパイキングニューラルネットワーク (SNN)

2.3.1 ネットワーク構造

本研究では，CAE のニューラルネットワークに SNN を用いる．SNN ではスパイクによって情報を処理するために，ネットワークの各層が神経細胞のうちのシナプスモデルと細胞体モデルから構成される．ネットワークの層は，以下の処理によってスパイク信号を伝達する [11]．

ある時間 t において, $n-1$ 層目の細胞体から出力されたスパイク $\boldsymbol{o}^{t,n-1}$ は, n 層目のシナプスによって重みづけされ統合された後, シナプス電流 $\boldsymbol{x}^{t,n}$ に変換される (1).

$$x_i^{t,n} = \sum_{j=1}^{l(n-1)} w_{ij}^n o_j^{t,n-1} \tag{1}$$

ここで, $l(n)$ は第 n 層におけるニューロンの数である.

次にシナプス電流 $\boldsymbol{x}^{t,n}$ は, n 層目の細胞体に入力され, 細胞体の膜電位 $\boldsymbol{u}^{t,n}$ を上昇させる (2).

$$u_i^{t,n} = u_i^{t-1,n} f(o_i^{t-1,n}) + x_i^{t,n} + b_i^n \tag{2}$$

$$where$$

$$f(x) = \tau e^{-\frac{x}{\tau}} \tag{3}$$

細胞体の膜電位 $\boldsymbol{u}^{t,n}$ は, 漏れ時間電位 $\boldsymbol{u}^{t-1,n}$ と入力シナプス電流 $\boldsymbol{x}^{t,n}$, およびバイアス $\boldsymbol{b}^n$ によって決定される. また, 漏れ時間電位は時定数が τ の指数関数 $f(x)$ により減衰される (3).

細胞体の出力 $o_i^{t,n}$ は式 (4)(5) より, 膜電位がある閾値 V_{th} 以下のときに 0 を出力し, 閾値 V_{th} 以上のときに 1 を出力し膜電位がリセットされる [12].

$$o_i^{t,n} = g(u_i^{t,n}) \tag{4}$$

$$where$$

$$g(x) = \begin{cases} 1\ (x \geqq V_{th}) \\ 0\ (x < V_{th}) \end{cases} \tag{5}$$

SNN はこのような処理を各層で行うことで, 0 か 1 のスパイクのみを入出力として情報の伝達をする.

2.3.2 近似的な誤差逆伝播法による SNN の学習

通常の NN は, NN の出力と教師信号の誤差を伝播することで, ネットワークの重みとバイアスを学習させる. 一方, SNN は, 情報が 0 か 1 の不連続なスパイク列で表現されるため, 微分を行うと勾配消失や勾配爆発が生じ, 誤差逆伝播による学習が進行しない. そこで, スパイクの導関数を式 (6) で近似することで, 近似的に誤差逆伝播法を行う.

$$\frac{\partial g}{\partial u} = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(u - V_{th})^2}{2}\right) \tag{6}$$

2.3.3 スパイク式エンコーダ

SNN は入力を 0 か 1 のスパイクとする一方で, 入力画像はアナログ値であるため, 画像をスパイクへ変換するスパイク式のエンコーダが必要となる. 本研究では, スパイク式エンコーダとして poisson encoder を用いる [13]. poisson encoder ではスパイクの発火確率をポアソン過程でモデル化する. これより, 画像の画素値 λ が与えられたとき, 時刻 t までに n 回発火する確率 $P[n]$ はポアソン分布に従うと仮定され, 式 (7) と表される.

$$P[n] = \frac{(\lambda t)^n}{n!} e^{-\lambda t} \tag{7}$$

また, 微小時間 Δt の内に 1 回発火する確率は, 1 次までのマクローリン展開より式 (8) と計算される.

$$P[1] = \frac{(\lambda \Delta t)^1}{1!} e^{-\lambda \Delta t} \approx \lambda \Delta t \tag{8}$$

poisson encoder では式 (8) で表される確率 P をもとに, アナログ画像を時間方向に T 次元拡張したスパイク画像に変換する.

発火率デコーダ

SNN では出力においても、SNN からのスパイク出力を連続値へ変換するためのデコーダが必要となる. デコーダは SNN からの出力スパイク列 $\boldsymbol{o_i}$ を発火率 fr_i に変換し, 発火率 fr_i に重み W_i とバイアス b_i 加えることでアナログ値 s_i への変換を行う.

$$fr_i = \frac{1}{T} \sum_{t=0}^{T} o_i^t \tag{9}$$

$$s_i = W_i \cdot fr_i + b_i \tag{10}$$

2.4 モデルベース強化学習

モデルベース強化学習とは, 現在の状態と行動から次の状態を予測する予測モデルを構築し, 予測した結果をもとに制御を行う強化学習手法のひとつである.

2.4.1 予測モデル

モデルベース強化学習では, 予測モデルによって今の状態 $\boldsymbol{s}_t$ と行動 $\boldsymbol{a}_t$ から未来の状態 $\boldsymbol{s}_{t+1}$ を予測する. 本研究では, 画像を状態 $\boldsymbol{s}_t$ として扱うが, 画像は制御に関係ない情報が多く冗長である. そのため, 画像 $\boldsymbol{s}_t$ を CAE によって潜在変数 $\boldsymbol{h}_t$ へと圧縮し, 予測モデルへの入力とする. また, 予測モデルの出力も, 未来の予測画像状態 $\boldsymbol{s}_{t+1}$ そのものではなく, 未来の画像の予測潜在変数 $\boldsymbol{h}_{t+1}$ とする.

2.4.2 評価モデル

モデルベース強化学習では, 予測した状態を評価するために評価関数を用いる. 本研究では予測値が潜在状態であり, 手動で評価関数を設計することは困難である. そのため, 評価モデルにはニューラルネットワークを用いる. 評価モデルは予測された潜在状態 $\boldsymbol{h}_{t+1}$ を引数として受け取り, 予測評価値 r_{t+1} を出力するモデルである.

2.4.3 モデル予測制御

モデル予測制御では, 予測モデルが予測した次の状態と次の評価値をもとにより良い評価になるであろう行動を随時選択することで制御を行う. 本研究では, モデル予測制御のうちランダムな行動候補から行動を選択するランダムシューティング法を用いる [3].

ランダムシューティング法ではまず, K 個の行動候補 $\boldsymbol{a}_t$ をランダムに用意する. また, 現在の画像状態 $\boldsymbol{s}_t$ は重み ϕ のネットワークで表現されるオートエンコーダ E_ϕ によって潜在状態 $\boldsymbol{h}_t$ に変換される. その後, 行動候補 $\boldsymbol{a}_t$ と潜在状態 $\boldsymbol{h}_t$ を重み θ のネットワークで表現される予測モデル f_θ に入力し K 通りの次の潜在状態 $\boldsymbol{h}_{t+1}$ を予測する.

$$\boldsymbol{h}_t = E_\phi(\boldsymbol{s}_t) \tag{11}$$
$$\boldsymbol{h}_{t+1} = f_\theta(\boldsymbol{h}_t, \boldsymbol{a}_t) \tag{12}$$

さらに, 予測された次の潜在状態 $\boldsymbol{h}_{t+1}$ を重み ψ のネットワークで表わされる評価モデル R_ψ に入力し, 評価値 $\boldsymbol{r}_{t+1}$ を予測する.

$$\boldsymbol{r}_{t+1} = R_\psi(\boldsymbol{h}_{t+1}) \tag{13}$$

次に, 再度 K 個の行動候補 $\boldsymbol{a}_{t+1}$ をランダムに用意する. その後, 行動候補 $\boldsymbol{a}_{t+1}$ と先ほど予測した潜在状態 $\boldsymbol{h}_{t+1}$ を予測モデルに入力し, 2 つ先の潜在状態 $\boldsymbol{h}_{t+2}$ を予測する. さらに, $\boldsymbol{h}_{t+2}$ を評価モデル R_ψ に入力して, 2 つ先の評価値 $\boldsymbol{r}_{t+2}$ を予測する.

$$\boldsymbol{h}_{t+2} = f_\theta(\boldsymbol{h}_{t+1}, \boldsymbol{a}_{t+1}) \tag{14}$$
$$\boldsymbol{r}_{t+2} = R_\psi(\boldsymbol{h}_{t+2}) \tag{15}$$

この操作を H 回繰り返し, H ステップ先までの未来の潜在状態 $\boldsymbol{h}_{t+1}$〜$\boldsymbol{h}_{t+H}$ と評価値 $\boldsymbol{r}_{t+1}$〜$\boldsymbol{r}_{t+H}$ を K 通り求める. 最後に, 各行動候補における評価値を総和し, 最も合計評価値が高い行動 $\boldsymbol{a}_t$ を行動候補から選んで制御入力 $\boldsymbol{a}_{\mathbf{control}}$ として制御を行う.

$$\boldsymbol{a}_{\mathbf{control}} = \max_{\boldsymbol{a}_t} \sum_{i=1}^{H} \boldsymbol{r}_{t+i} \tag{16}$$

一般に, ランダムシューティングにおける K, H をそれぞれ population, horizon と呼ぶ.

3. 実験方法

3.1 実験環境

本実験では OpenAI gym によって提供されるシミュレーション環境を用いた. 実験は, 2 次元平面上において, 中心を固定された 1 リンクのアームの先端を円形のターゲットに到達させるタスクとした. ターゲットはアーム先端の軌道円周上にランダムに生成される. シミュレーションの 1 ステップの時間間隔は 0.05 秒とし, 200 ステップを 1 エピソードとした. 環境から取得できる状態はシステム全体が写る画像のみとし, 色はグレースケール, サイズは 64 × 64 とした.

本環境における評価値 r は, アームの先端 $\boldsymbol{x}_{\mathrm{arm}}$ とターゲットの座標 $\boldsymbol{x}_{\mathrm{target}}$ の距離に-1 を乗じたものとした (式 (17)).

$$r = -|\boldsymbol{x}_{\mathrm{arm}} - \boldsymbol{x}_{\mathrm{target}}|^2 \tag{17}$$

3.2 評価するネットワーク構造

本研究では SNN を CAE のネットワークに用いた場合と, そうでない場合を比較し, ロバスト性を評価する. そのため, 評価するネットワークは CAE に従来の CNN を用いたものと, 畳み込みスパイキングニューラルネットワーク (convolutional spiking neural network, CSNN) を用いたものの 2 種類を用いる.

CAE におけるエンコーダ部のネットワーク構造を表 1 に示す.

Table 1 Network structure of encoder in CAE

layer	output size	kernel	stride	padding
Input	4x64x64	4x4	2	1
Conv1	16x32x32	4x4	2	1
Conv2	32x16x16	4x4	2	1
Conv3	64x8x8	4x4	2	1
Conv4	128x4x4	4x4	2	1
Conv5	256x2x2	4x4	2	1

CAE のデコーダ部のネットワーク構造は表 1 の構造を反転したものを用いた.

予測モデルと評価モデルは通常の ANN を用いて構成した. それぞれのネットワーク構造を表 2 に示す.

Table 2 Network structure of evaluation model and prediction model

network type	layers
prediction model	1025x1024x1024x1024
evaluation model	1024x1024x1024x1

3.3 ロバスト性評価

SNN を CAE のネットワークに用いた場合のロバスト性を評価するために, CAE に CSNN を用いたものと CNN を用いたものに対して, 以下の 3 つの検証を行った.

1) 入力画像にノイズを加えた際の再構成画像の比較

2) 入力画像にノイズを加えた際の潜在状態の特徴マップの比較

3) 入力画像にノイズを加えた際のタスク評価値の比較

加えたノイズ ϵ はガウシアンノイズとし, 確率密度関数 $p(\epsilon)$ は式 (18) で表される.

$$p(\epsilon) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\epsilon^2}{2\sigma^2}\right) \tag{18}$$

ガウス分布における平均は 0 とし, 標準偏差 σ は 5 から 255 を 25 刻みで与えた. ネットワークに入力する際には, 画像を 255 で除し, 0 から 1 の範囲へ正規化を行った.

4. 実験結果

4.1 入力画像にノイズ付加した際の再構成画像比較

入力画像にガウシアンノイズ (σ=105, 205) を付加した際の, 再構成画像を図 2 に示す. CAE のネットワークに CNN を用いたとき, 標準偏差 σ が 105 以上になると再構成画像のアーム長さが不安定になり, 元の画像の特徴を捉えた再構成が不可能であった. 一方, CAE のネットワークに CSNN を用いると, 標準偏差 σ が 205 より小さなガウシアンノイズにおいて, ノイズを除去し元の画像を復元することが可能であった.

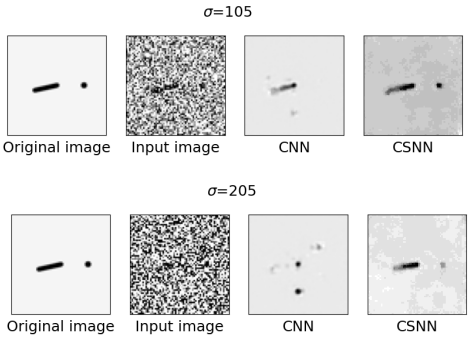

Fig. 2 Reconstruct images by CNN and CSNN

4.2 入力画像にノイズを付加した際の潜在変数の特徴マップ比較

潜在変数の特徴マップの変化を比較するために, 以下の 2 つの状態から得た画像にノイズを付加し, 特徴量を抽出した.

1) ターゲットの位置を 0 ラジアンで固定し, アームの角度を $-3/4\pi, -1/4\pi, 1/4\pi, 3/4\pi \pm \pi/30$ で変化

2) アームの位置を 0 ラジアンで固定し, ターゲットの角度を $-3/4\pi, -1/4\pi, 1/4\pi, 3/4\pi \pm \pi/30$ で変化

特徴量は学習済みの CAE によって 1024 次元の潜在変数として抽出を行った. その後, 次元削減アルゴリズムである t-Distributed Stochastic Neighbor Embedding (t-SNE) [14] によって 2 次元の潜在変数 h_1, h_2 に変換したのち標準化し, 特徴マップを得た. さらに, 2 次元の特徴量が角度ごとに分類されていることを確かめるために, k-means 法を用いた特徴量のクラスタリングを行った. それぞれの状態の代表的な特徴マップを図 3, 図 4 に示す. プロットの色は各アームの角度を表す. また, 図上の長方形は k-means 法によって分類されたグループである. 同じ色のプロットが同一グループ内に存在していれば, アームの角度変化を特徴量へ変換できていることが示唆される.

4.2.1 アームの角度を変化させた際の特徴マップの変化

図 3 にノイズ標準偏差 σ が 155, 230 の際のアーム角度を変化させた際の特徴マップを示す. 図 3 より, CNN を用いた場合, 標準偏差 σ が 155 以上のノイズが付加されると同一グループ内に異なるアーム角度が混在する結果となった. 一方 CSNN を用いた場合は, 標準偏差 σ が 230 未満のとき特徴量を分類することが可能であった.

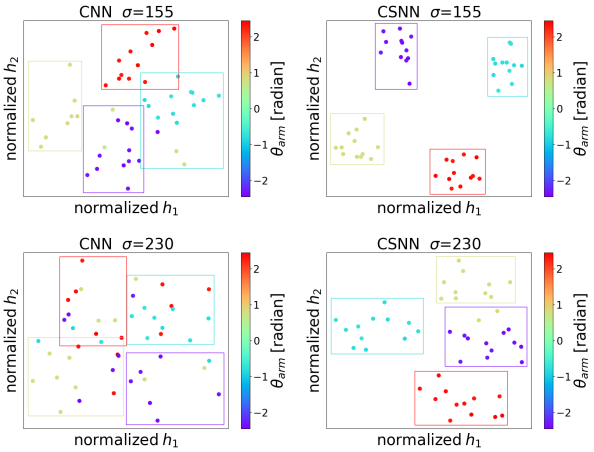

Fig. 3 Feature maps of arm angle by CNN and CSNN

4.2.2 ターゲットの角度を変化させた際の特徴マップの変化

図 4 にノイズ標準偏差 σ が 80, 130 の際のターゲット位置の特徴マップを示す. 図 4 より, CNN を用いた場合, 標準偏差 σ が 80 以上のノイズが付加されると同一グループ内に異なるアーム角度が混在する結

果となった．一方 CSNN を用いた場合は，標準偏差 σ が 130 未満のとき特徴量を分類することが可能であった．

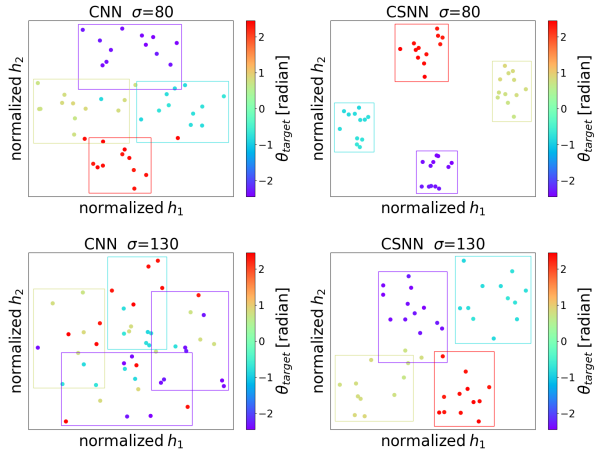

Fig. 4 Feature maps of target position by CNN and CSNN

4.3 ノイズを加えた際のタスク評価値の比較

入力画像にガウシアンノイズを付加し，タスク性能の評価を行った．各標準偏差 σ につき 50 エピソードづつシミュレーションを行い，1 エピソードにおける評価値の合計の中央値を比較した．図 5 に比較結果を示す．赤，青，灰色のグラフはそれぞれ CSNN を用いたネットワーク，CNN を用いたネットワーク，ランダムなトルク入力で制御を行った際の評価値である．

CNN を用いた場合では，ノイズ標準偏差 σ が 80 以上になると，ランダムな制御と同程度まで評価値が低下した．一方，CSNN を用いた場合は，σ が 155 以下において，タスク評価値の低下を抑えることが可能であった．

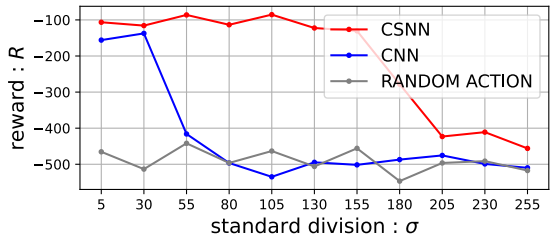

Fig. 5 Noise standard division and task reward

5. 考察

図 2 より，CSNN を用いるとノイズが大きくなった場合でも，元の画像を再構成が可能であった．また，図 3，図 4 より，ノイズによる潜在変数の特徴マップの特徴量の混在，変形も小さく抑えられていた．このことから，CSNN を用いると，通常の CNN よりもノイズによる潜在変数への影響が小さいことが示唆された．結果として，ノイズが加わった際のタスク評価値が下がりにくくなったと考えられる．

SNN を用いた際のノイズへのロバスト性向上の一因として，以下の 2 点が考えられる．

1) スパイクエンコーディングによる疑似的なノイズの付加および学習
2) SNN によるノイズが与える潜在変数への影響の低減

5.1 スパイクエンコーディングによる疑似的なノイズの付加および学習

本研究では，モデルの学習の際に，画像へノイズを付加していない．そのため，通常の CNN では，状態画像が決定的にモデルへ入力される．一方で，CSNN では，状態画像はスパイクエンコーディングによって，確率的にスパイクへ変換される．画素値 λ が与えられた際に，Δt 秒間に 1 回発火する確率 (画素値が 1 に変換される確率) $P[1]$ は，$\lambda\Delta t$ である (式 (8))．本研究では，$\Delta t = 1$ としたため，発火確率 $P[1]$ は式 (19) と表せる．

$$P[1] \approx \lambda \tag{19}$$

また，画素値 λ は正規化し，0 以上 1 以下の実数としたため，発火確率 $P[1]$ は，λ の確率で 1 を出力し，$1-\lambda$ の確率で 0 を出力するベルヌーイ分布とみなすことができる．ベルヌーイ分布の期待値 $E[X]$，分散 $V[X]$ はそれぞれ式 (20)，(21) と表される．

$$E[X] = \lambda \tag{20}$$
$$V[X] = \lambda(1-\lambda) \tag{21}$$

よって，入力画像 λ のスパイク画像 o^t への変換は，式 (22) で表すことができる．

$$o^t = \lambda \pm \sqrt{\lambda(1-\lambda)} \tag{22}$$

本研究では，ターゲットとアームの画素値を 0，背景の画素値を 180 とした．画素値は 255 で除し，正規化したため，λ はそれぞれ 0, 180/255 となる．よって，ターゲットとアームのスパイク $o^{t,\mathrm{target,arm}}$，背景のスパイク $o^{t,\mathrm{back}}$ は式 (23),(24) と表せる．

$$o^{t,\mathrm{target,arm}} = 0 \tag{23}$$
$$o^{t,\mathrm{back}} \approx 0.71 \pm 0.46 \tag{24}$$

式 (23) より，ターゲットとアームのスパイク値は常に 0 であり，疑似的なノイズ付加がなかったと考えられる．一方，背景については，式 (24) より，標準偏差 0.46 程度のばらつきがノイズとしてスパイク値に含まれていたと考えられる．また，標準偏差 0.46 を画素値に変換すると，117 程度の数値である．ノイズ付加時のタスク評価 (小節 4.3) では，SNN を使用したモデルは標準偏差が 155 以下ノイズに対して頑健

であった. スパイクへの変換時に生じるばらつきと, 実験の際に頑健性を示したノイズの標準偏差が同等な値であることから, スパイクエンコーディングによって, モデルへの入力に疑似的なノイズが付加され, ノイズに対して堅牢なモデルが学習されたと考えられる.

5.2 SNN によるノイズが与える潜在変数への影響の低減

小節 4.2 において, SNN を使用した方が, ノイズによる潜在変数への影響が小さい結果となった. この結果は, SNN のネットワーク構造がノイズによる影響を低減したからだと考えられる.

そこで, 通常の NN を用いた場合と, SNN を用いた場合について, 入力 $\boldsymbol{\lambda}$ に対する潜在変数 $\boldsymbol{h}$ の変化量を解析的に考える. 比較に用いるネットワーク構造について, 通常の NN は 1 層のネットワークとし, SNN を用いたネットワーク構造は図 6 で表される, スパイクエンコーダ, 1 層の SNN, 発火率デコーダとする.

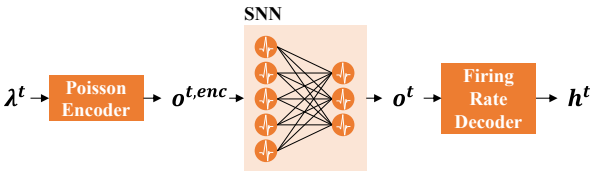

Fig. 6 SNN network structure

まず, 通常の NN における潜在変数 $\boldsymbol{h}$ の変化量は式 (25) と表せる.

$$\begin{aligned}\frac{\partial}{\partial \lambda_i} h_j^{\mathrm{NN}} &= \frac{\partial}{\partial \lambda_i} \sum_i w_{ji}\lambda \\ &= \sum_i w_{ji}\end{aligned} \tag{25}$$

ここで, w, b はそれぞれニューラルネットワークの重みとバイアスである.

次に, SNN における潜在変数の変化量について求める. まず, 潜在変数の変化量は発火率デコーダ (小節 2.3.3) より, 式 (26) となる.

$$\begin{aligned}\frac{\partial}{\partial \lambda_i} h_j^{\mathrm{SNN}} &= \frac{\partial}{\partial \lambda_i}(w_j^{\mathrm{dec}} fr_j + b_j^{\mathrm{dec}}) \\ &= w_j^{\mathrm{dec}} \frac{\partial}{\partial \lambda_i} fr_j \\ &= w_j^{\mathrm{dec}} \frac{1}{T} \sum_t^T \frac{\partial}{\partial \lambda_i^t} o_j^t\end{aligned} \tag{26}$$

ここで, $w^{\mathrm{dec}}, b^{\mathrm{dec}}$ はそれぞれ発火率デコーダの重み, バイアスである. また, デコーダの重み w^{dec} は, 出力の値を調整するためのものであり, SNN のネットワーク構造とは無関係なため, $w^{\mathrm{dec}} = 1$ としても一般性は失われない (式 (27)).

$$\frac{\partial}{\partial \lambda_i} h_j^{\mathrm{SNN}} = \frac{1}{T} \sum_t^T \frac{\partial}{\partial \lambda_i^t} o_j^t \tag{27}$$

発火率デコーダに入力されるスパイク $\boldsymbol{o}^t$ は SNN の出力である. よって, SNN のネットワーク構造 (小節 2.3) より, 式 (28) のように計算される.

$$\begin{aligned}\frac{\partial}{\partial \lambda_i} h_j^{\mathrm{SNN}} &= \frac{1}{T} \sum_t^T \frac{\partial}{\partial \lambda_i^t} g(u_j^t) \\ &= \frac{1}{T} \sum_t^T (\frac{\partial g}{\partial u_j^t} \frac{\partial}{\partial \lambda_i^t} u_j^t) \\ &= \frac{1}{T} \sum_t^T \frac{\partial g}{\partial u_j^t} (\frac{\partial}{\partial \lambda_i^t} (\tau u_j^{t-1} + x_j^t + b_j)) \\ &= \frac{1}{T} \sum_t^T \frac{\partial g}{\partial u_j^t} (\sum_i w_{ji} \frac{\partial}{\partial \lambda_i^t} o_i^{t,\mathrm{enc}})\end{aligned} \tag{28}$$

ここで, 時刻 $t-1$ における膜電位 u^{t-1} は, 時刻 t における入力 λ^t によらないため, 変化量を 0 とした. また, o^{enc} はスパイクエンコーダからの出力スパイクとする.

o^{enc} は, 確率 λ で 1 を出力するベルヌーイ分布によって決定される (小節 5.1). よって, 一様分布 U とステップ関数 H を用いて, 式 (29) のように表せる.

$$o_i^{t,\mathrm{enc}} = 1 - H(s_i^t - \lambda_i^t) \tag{29}$$

$$where$$

$$s_i^t \sim U(0,1) \tag{30}$$

ここで, s_i^t は一様分布 $U(0,1)$ からのサンプリング出力である.

これより, 式 (28) は, 式 (29) を用いて以下の式 (31) と計算される.

$$\begin{aligned}\frac{\partial}{\partial \lambda_i} h_j^{\mathrm{SNN}} &= \frac{1}{T} \sum_t^T \frac{\partial g}{\partial u_j^t} \sum_i w_{ji} \frac{\partial}{\partial \lambda_i^t} \left(1 - H(s_i^t - \lambda_i^t)\right) \\ &= \frac{1}{T} \sum_t^T \frac{\partial g}{\partial u_j^t} \sum_i w_{ji} \delta(s_i^t - \lambda_i^t)\end{aligned} \tag{31}$$

ここで, $\delta(x)$ は $x = 0$ で無限大の値をとるデルタ関数とする. よって, SNN における潜在変数の変化量は, 式 (32) のように計算される.

$$\frac{\partial}{\partial \lambda_i} h_j^{\mathrm{SNN}} = \underset{t}{\mathrm{mean}} \left(\sum_i \frac{\partial g}{\partial u_j^t} w_{ji} \delta(s_i^t - \lambda_i^t) \right) \tag{32}$$

いま, 入力 $\boldsymbol{\lambda}$ は 0 以上 1 以下の実数である. これより, 入力範囲における潜在変数の変化量の総和

$dh^{\mathrm{NN}}, dh^{\mathrm{SNN}}$ は式 (33), (34) と計算される.

$$\begin{aligned} dh_j^{\mathrm{NN}} &= \int_0^1 \left| \sum_i w_{ji} \right| d\lambda_i \\ &= \left| \sum_i w_{ji} \right| \end{aligned} \tag{33}$$

$$\begin{aligned} dh_j^{SNN} &= \mathop{\mathrm{mean}}_t \int_0^1 \left| \sum_i \frac{\partial g}{\partial u_j^t} w_{ji} \delta(s_i^t - \lambda_i^t) \right| d\lambda_i \\ &= \mathop{\mathrm{mean}}_t \left| \sum_i \frac{\partial g}{\partial u_j^t} w_{ji} \right| \end{aligned} \tag{34}$$

ここで, スパイクエンコーダは λ が 0 以上 1 以下の範囲内で 1 となる不連続点が存在するため, デルタ関数を 0 以上 1 以下の範囲で積分すると 1 となる. また, 式 (6) より, $\partial g/\partial u$ の最大値は式 (35) である.

$$\frac{\partial g}{\partial u_j^t} \leqq \frac{1}{\sqrt{2\pi}} \tag{35}$$

よって, dh^{SNN} の上限は式 (36) のように計算される.

$$dh_j^{\mathrm{SNN}} \leqq \frac{1}{\sqrt{2\pi}} \left| \sum_i w_{ji} \right| \tag{36}$$

以上より, 式 (33) および式 (36) を比較すると, SNN を使用した方が $1/\sqrt{2\pi}$ 倍以下の変化量に抑えられることが示唆される. また, この抑制項はスパイク発火の式 (5) から導出される. そのため, 入力を膜電位として蓄積し, 一定の閾値を超えたときに発火するという操作が潜在変数の変化量を抑え, 結果としてノイズに対するロバスト性を実現していると考えられる.

6. 結言

本研究では, モデルベース強化学習において, SNN をニューラルネットワークに用いた畳み込みオートエンコーダを画像処理に用いることで, 予測モデルにおけるモデルバイアスを低減し, 画像ノイズに対する高ロバストな制御の獲得を目的とした.

その結果, 入力画像にガウシアンノイズが付加されても, SNN を用いたネットワークではタスク評価の低下を CNN に比べて大幅に抑えることが可能であった. また, ノイズが加わった際の潜在変数を k-means 法によりクラスタリングし, 適切に分類できるノイズ強度を比較した. ノイズによる潜在変数の変化と混在は, SNN を用いた方が小さく抑えられる結果となった.

実環境では, 画像に対するノイズだけでなく, 背景変化等の環境ノイズが加わることが多い. 今後は, 環境ノイズに対してもロバストな制御を獲得する必要がある.

参考文献

1) S. Kumra and C. Kanan. Robotic grasp detection using deep convolutional neural networks. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 769–776, 2017.
2) R. S. Sutton, A. G. Barto, and R. J. Williams. Reinforcement learning is direct adaptive optimal control. *IEEE Control Systems Magazine*, Vol. 12, No. 2, pp. 19–22, 1992.
3) Tingwu Wang, Xuchan Bao, Ignasi Clavera3, Jerrick Hoang, Yeming Wen, Eric Langlois, Shunshi Zhang, Guodong Zhang, Pieter Abbeel3, and Jimmy Ba1. Benchmarking model-based reinforcement learning. *arXiv:1907.02057*, 2019.
4) Hafner, Danijar, Lillicrap, Timothy, Fischer, Ian, Villegas, Ruben, Ha, David, Lee, Honglak, Davidson, and James. Learning latent dynamics for planning from pixels. *Proceedings of Machine Learning Research*, Vol. 97, , 2018.
5) Tatsuya Matsushima, Hiroki Furuta, Shixiang Gu, and Yutaka Matsuo. Morel : Model-based offline reinforcement learning. *arXiv:2005.05951*, 2020.
6) K. Naya, K. Kutsuzawa, D. Owaki, and M. Hayashibe. Spiking neural network discovers energy-efficient hexapod motion in deep reinforcement learning. *IEEE Access*, Vol. 9, pp. 150345–150354, 2021.
7) Dongcheng Zhao, Yang Li, Yi Zeng, Jihang Wang, and Qian Zhang. Spiking capsnet: A spiking neural network with a biologically plausible routing rule between capsules. *Information Sciences*, Vol. 610, pp. 1–13, 2022.
8) Devdhar Patel, Hananel Hazan, Daniel J. Saunders, Hava T. Siegelmann, and Robert Kozma. Improved robustness of reinforcement learning policies upon conversion to spiking neuronal network platforms applied to atari breakout game. *Neural Networks*, Vol. 120, , 2019.
9) Dominik Thalmeier, Marvin Uhlmann, Hilbert J. Kappen, and Raoul-Martin Memmesheimer. Learning universal computations with spikes. *PLoS Computational Biology*, 2016.
10) G. E. HINTON and R. R. SALAKHUTDINOV. Reducing the dimensionality of data with neural networks. *Science*, Vol. 313, pp. 504–507, 2006.
11) Wu Yujie, Deng Lei, Li Guoqi, Zhu Jun, and Shi Luping. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in Neuroscience*, Vol. 12, , 2018.
12) Gerstner, W., Kistler, and W. M. Spiking neuron models: Single neurons, populations, plasticity. *Cambridge University Press*, 2002.
13) 島崎秀昭. スパイク統計モデル入門. 2015.
14) Van der Maaten, Laurens, , and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, pp. 2579–2605, 9 2008.