DEIM2022 K24-4

Neural Architecture Search を取り入れた VMCPU 使用率予測モデル運用手法の検討

高橋佑里子† 田宮 豊†† 小口 正人†

† お茶の水女子大学 〒 112–8610 東京都文京区大塚 2 丁目 1 − 1
†† 富士通株式会社 〒 211–8588 神奈川県川崎市中原区上小田中 4 丁目 1 − 1
E-mail: †{yuriko-t,oguchi}@ogl.is.ocha.ac.jp, ††tamiya.yutaka@fujitsu.com

あらまし 仮想環境を用いたクラウドサービスにおいて，CPU 資源のオーバーコミットに由来する仮想マシン (Virtual Machine: VM) の性能低下を防ぐことを目的として，VM の CPU 使用率を予測し制御を行う技術が知られている．VM とそこで実行されるアプリケーションは変化していくため，変化に合わせて予測モデルを継続的に学習・更新することで予測精度を担保することが望ましい．しかし，従来は学習させるデータを変えるのみで，予測モデルの構造を変えることはなかった．そこで本研究は，Neural Architecture Search(NAS) を取り入れることで，環境の変化に応じて予測モデルの構造を変化させながら予測モデルを運用する手法の検討を行う．

キーワード 時系列データ，機械学習，深層学習，Neural Architecture Search，AutoML，仮想環境

1 はじめに

近年のクラウドサービスでは，物理サーバ (Physical Machine: PM) の低 CPU 使用率が課題となっており [1]，これを改善すべく，事業者では，サーバを仮想化することで CPU 使用率を向上させ，PM 数を削減する取り組みが行われている．この取り組みでは，PM が自身の CPU 資源を超えた仮想 CPU を割り当てられるオーバーコミット状態に陥ることで，仮想マシン (Virtual Machine: VM) の性能が低下する可能性がある [2]．

これを防ぐことを目的として，VM の CPU 使用率を予測モデルによって予測し，その結果に基づいて制御を行う技術が知られている [3]．それは，図 1 のように運用エリア (PM1) とテンポラリエリア (PM2) が存在し，PM1 のみに VM1,2,3 が割り当てられている状態で，10 分後に VM3 の CPU 使用率が大幅に上昇し PM1 の CPU 資源を超えるという予測がされた場合，VM3 をあらかじめ PM2 に移行しておき，10 分後にオーバーコミット状態が発生しないようにする，といった制御である．

VM 上で多く実行されるアプリケーションは時間が経つにつれ変化していくため，それに伴って実行される VM のワークロードも変化していく．図 2 のように，VM のワークロードが大きく 3 種類に分類できるとしたとき，日によって 3 種類の含まれる割合が変わっていく，といったイメージである．そのため，環境の変化に合わせて予測モデルを継続的に学習し，モデルを更新することで予測精度を担保することが望ましいが，従来は学習させるデータを変えるのみで，予測モデルのネットワーク構造を変えることはなかった．しかし，学習させるデータによって最適な予測モデルのネットワーク構造は異なる．

そこで本研究では，Neural Architecture Search(NAS) を取り入れることで，環境の変化に応じて予測モデルの構造を変化させながら予測モデルを運用する手法の検討を行う．NAS とは，ニューラルネットワークの構造自体を自動的に最適化することである．この NAS を，予測モデルの継続的な学習の中で適切なタイミングで行うことで，より高精度な VM の CPU 使用率予測が可能になると考えた．本研究の成果は以下の通りである．

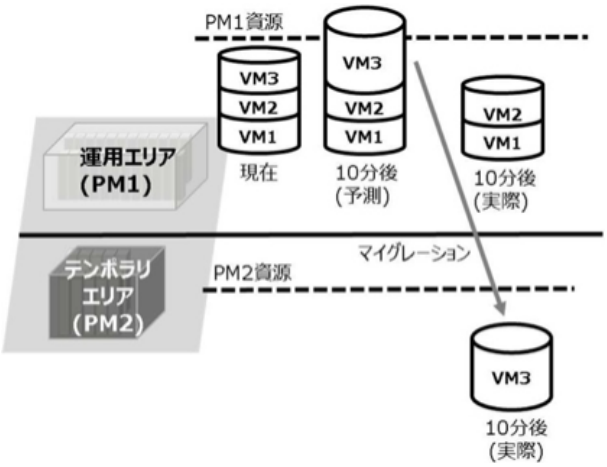

図 1 VM 制御のイメージ

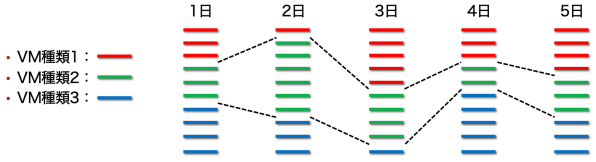

図 2 実行される VM の種類の割合が変化していくイメージ

- 予測モデルの精度を向上させるために，NAS が有効であることを確認した．
- モデルのネットワーク構造と学習時間，及びモデルのネットワーク構造と予測精度の関係を調査し，予測モデルの運用における最適な NAS の探索範囲を定めた．

- VM の CPU 使用率予測モデルの運用において，NAS を取り入れる手法を提案し，フローチャートを作成した．
- 提案手法のフローチャートに沿った実験を行い，NAS を取り入れることでより高精度な予測モデルの運用が可能になることを確認した．

2 関 連 研 究

クラウドサービスにおける VM の CPU 使用率の予測や制御に関する研究は，以前から多く行われている．

[4] では，ラックユニットの消費電力を分析し，API 連携を用いて VM の管理ソフトウェアと連携したシステムの提案を行っている．仮想データセンターを想定して計算した結果，エアコンとサーバーを合わせた消費電力を 6.8%削減可能ということを確認している．

[5] では，実際のデータセンターから取得したデータトレースを使用し，VM のワークロードの特徴付けを行い，隠れマルコフモデルをベースとした手法でワークロードパターンの変動を予測する機能を開発している．

[6] では，サービスレベル契約 (SLA) 違反と電力コストの削減を目的として，線形回帰手法に基づいて PM の CPU 使用率の短期的な予測を行っている．PM が過負荷になった場合，一部の VM を他の PM に移行することで，エネルギー消費量と SLA 違反率の大幅な削減に成功している．

[7] では，VM の CPU 使用率などのバースト性があり，明確な周期性を持たない時系列データに対しても，将来の負荷やピーク負荷のタイミングを正確に予測できるニューラルネットワーククベースのフレームワークの開発を行っている．このフレームワークにより，ARIMA モデルや基本的なニューラルネットワークモデルと比較して予測誤差は最大 3 倍，予測タイミングは 2〜9 倍の改善に成功している．

[8] では，エネルギー消費量に基づいて，リアルタイムに VM を統合するフレームワークの提案を行っている．提案されたフレームワークを使用することで，フレームワークを使用していないデータセンターと比較して最大 80%の改善が示され，PM の高使用率と省エネルギーの実現に成功している．

3 関 連 技 術

3.1 Recurrent Neural Network (RNN)

RNN とは，再帰型構造を持つニューラルネットワークであり，過去の計算情報を保持することができるため，時系列データの学習に用いられている．しかし，系列が長くなり深いネットワークになると，誤差逆伝播のアルゴリズムでは勾配の消失，発散などの問題が生じ，長期間の情報を保持できないという欠点がある．

3.2 Long Short-Term Memory (LSTM)

そこで，RNN を長期依存を扱うことができるように改良したものが Long Short-Term Memory (LSTM) である．LSTM は RNN に入力ゲート，出力ゲート，忘却ゲートを導入することにより，RNN では扱うことができなかった長期依存を扱うことができる．LSTM の構造を図 3 に示す．

本研究で扱う VM の CPU 使用率は時系列データであり，比較的長いパターンを学習する必要があるため，長期の時系列データの学習を行うのに有効である LSTM を用いる．

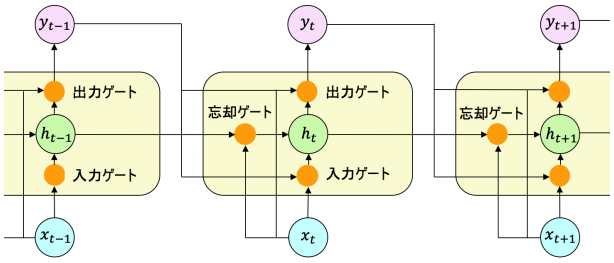

図 3 LSTM の構造

3.3 Neural Architecture Search (NAS)

Neural Architecture Search (NAS) とは，ニューラルネットワークの構造自体を自動的に最適化することである [9]．NAS によって生み出されたニューラルネットワークは，画像分類，物体検出などの分野で，人間が設計したニューラルネットワークの性能を上回ることが報告されている [10] [11]．

NAS は，探索空間，探索戦略，パフォーマンス推定戦略の大きく 3 つのポイントに分けられる．図 4 のように，探索戦略によって選ばれた探索空間に含まれるアーキテクチャでの学習を行い，作成されたモデルのパフォーマンスを推定した結果を元に別のアーキテクチャでの学習を試す，という流れを繰り返すことで，より良いアーキテクチャを模索する．

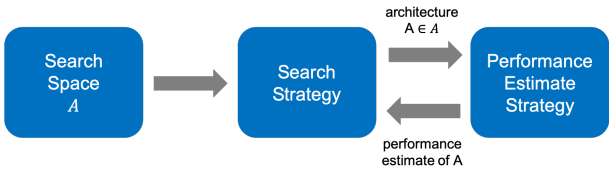

図 4 NAS の仕組み

4 使用するデータセット

本研究では，Microsoft 社が提供している Azure の VM トレースデータセット [12] [13] の一部の，”avg cpu”(平均 CPU 使用率) の項目を使用した．このデータセットは，1 点が 5 分間隔となっている．異なる傾向を持つデータセットを作成するために，以下の手順でデータセットを作成し，20 種類のうち 10 種類を過去のデータ，10 種類を運用段階のデータとした．1 種類のデータの長さは 360 点であり，これは 30 時間分に相当する．

（1） 元のデータセットを 0-1 に正規化する

（2） 開始点を 1 点ずつずらしながら 360 点ごとに時系列データを切り出す

（3） 切り出されたデータから 100 種類の波形を選択する

（4） 100 種類の波形を異なる割合で合計 100 個含むデータセットを 20 種類作成する

3 では，膨大な波形の中からなるべく似ていない 100 種類を選択するために，段階的に k-means 法クラスタリングを行った．具体的には，データを 10 万個ずつに分け，10 万個を 100 種類のクラスタに分類し，各クラスタから 1 種類のデータを残すという作業を繰り返し，残ったデータを再度 100 種類のクラスタに分類し，各クラスタから 1 種類のデータを残す，という作業を行った．

3 で最終的に選択された 100 種類の波形の一部 (5 種類) を図 5 に示す．縦軸が正規化後の平均 CPU 使用率，横軸が時間 (分) である．

5 予 備 実 験

5.1 予備実験 1

予備実験 1 では，NAS の有効性を確認するために 4 で作成した準備段階で使用する過去のデータ 10 種類に対して NAS を行い，出力されたモデルのネットワーク構造を比較した．NAS は表 1 のような探索範囲で行い，その他の条件は，エポック数 100，バッチサイズ 100，ウィンドウ幅 200，最大トライアル数 100 とした．

この実験では，ライブラリとして AutoKeras [14] [15] を使用した．AutoKeras とは，AutoML 対応の Keras モジュールであり，決められた探索範囲の中で様々なネットワーク構造やハイパーパラメータで学習を行い，複数回の試行 (トライアル) の後，最も精度が良くなったモデルを出力する機能を持つ．また，これ以上のトライアルは不要と判断した場合に自動で打ち切る機能もある．

表 1 予備実験 1 における NAS の探索範囲

パラメータ	範囲
ラーニングレート	0.01 または 0.001
LSTM の層数	1-3
LSTM のユニット数	1-30

実験により出力されたモデル 10 種類のネットワーク構造一覧を表 2 に示す．

表 2 予備実験 1 で出力されたモデルのネットワーク構造一覧

層数	ユニット数
3	29
3	30
3	22
3	21
3	28
3	30
3	27
3	28
3	20
3	24

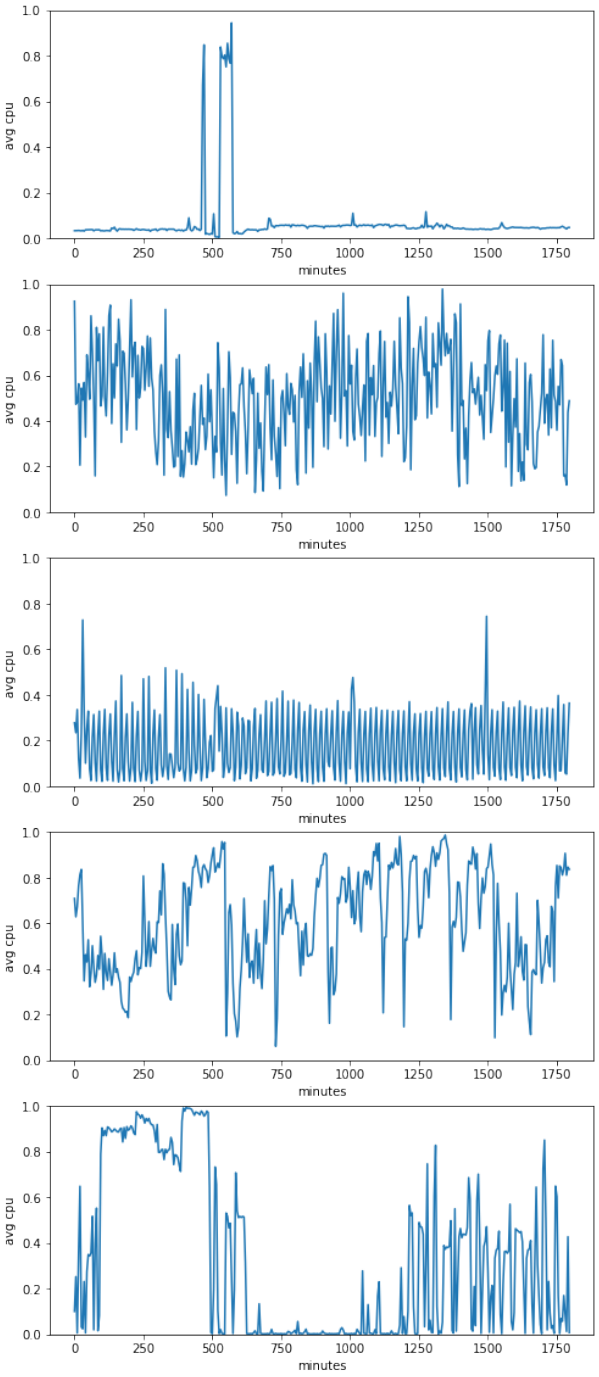

図 5 選択した 100 種類の波形の例 (5 種類)

この結果から，学習データによって最適なモデルのネットワーク構造は異なることが分かり，NAS を行うことは予測モデルの精度を向上させるために有効であるということが確認できた．

また，この実験では各 NAS のトライアル回数が，少ないもので 10 程度，多いもので 40 程度，平均で 22.8 となった．NAS にかかる時間はトライアル回数に比例するので，トライアル回数が多くなることで NAS にかかる時間が長くなる．そのため，運用で NAS を取り入れるには，事前に探索範囲を絞る必要があるということが分かった．

そこで，最適な NAS の探索範囲を定めるために，以下の 2

つの実験を行った．

5.2　予備実験 2

予備実験 2 では，モデルのネットワーク構造によって学習時間がどのように変化するかを確認するために，4 で作成した準備段階で使用する過去のデータの 1 つに対し，表 3 から得られるすべての組み合わせ (90 通り) のネットワーク構造でそれぞれ 3 回学習を行い，かかった時間の平均値を取得し，比較した．その他の条件は，エポック数 100，バッチサイズ 100，ウィンドウ幅 200 とし，ライブラリとして Keras を使用して行った．

Keras [16] とは，Tensorflow [17] の上に構築された透過的かつ互換性のある深層学習ライブラリである．

表 3　予備実験 2 でのネットワーク構造探索範囲

パラメータ	範囲
LSTM の層数	1-3
LSTM のユニット数	1-30

結果は，図 6 のようになった．縦軸が 100 エポックにかかった学習時間の平均値，横軸がユニット数である．

この結果から，まず，同じユニット数同士の結果を比較すると，学習時間が層数に比例していることが読み取れる．また，ユニット数が 8 を超えたところから，ユニット数が大きくなるにつれて学習時間が少しずつ長くなることや，ユニット数が 4 の倍数の時に学習時間が短いことが読み取れる．

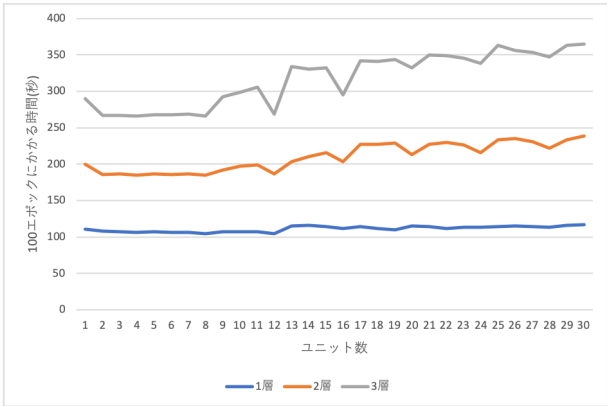

図 6　予備実験 2 の結果

5.3　予備実験 3

予備実験 3 では，モデルのネットワーク構造によって予測精度がどのように変化するかを確認するために，4 で作成した準備段階で使用する過去のデータ 10 種類のうちランダムに 5 種類を選択し，表 4 から得られるすべての組み合わせ (90 通り) のネットワーク構造で学習を行い，正解値と学習を行ったモデルの予測値の RMSE を比較した．その他の条件は，エポック数 100，バッチサイズ 100，ウィンドウ幅 200 とし，ライブラリとして Keras を使用して行った．

RMSE とは，二乗平均平方根誤差と呼ばれる回帰モデルの誤差を評価する指標の 1 つである．この値が小さく 0 に近いほど精度が良いということになる．長さ n の時系列データの正解値を y_t，予測値を $\tilde{y}_t$ とすると，以下のような式で計算される．

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(y_t - \tilde{y}_t)^2}$$

表 4　予備実験 3 でのネットワーク構造探索範囲

パラメータ	範囲
LSTM の層数	1-3
LSTM のユニット数	1-30

結果は，図 7，8，9，10，11 のようになった．縦軸が RMSE の平均値，横軸がユニット数である．

これらの結果から，層数が同じ場合はユニット数が大きいほど精度が良くなること，層数による精度の差は，1 層は 2 層,3 層と比較して悪いが，2 層と 3 層の間では差が少ないことが読み取れる．

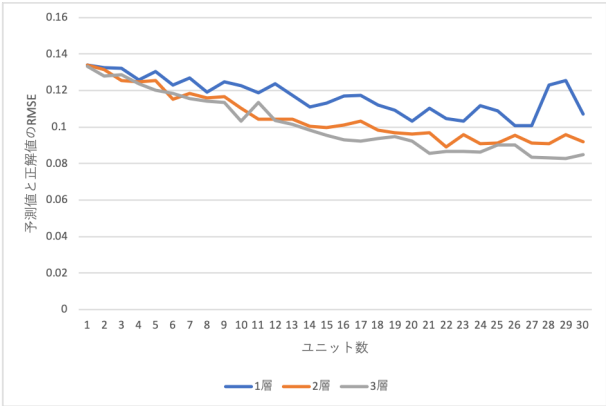

図 7　予備実験 3 の結果 (学習データ 1)

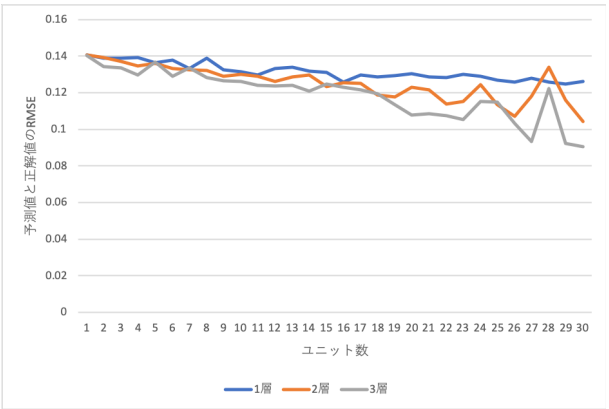

図 8　予備実験 3 の結果 (学習データ 2)

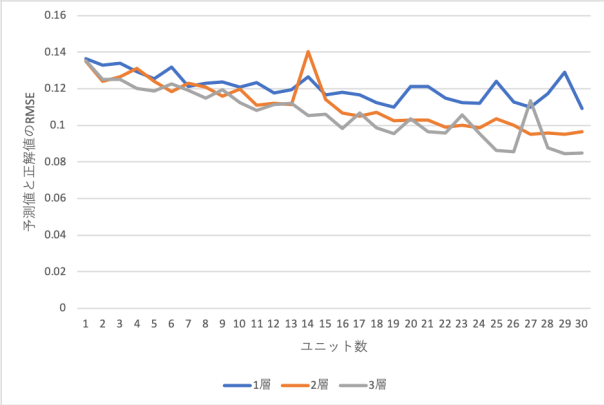

図 9　予備実験 3 の結果 (学習データ 3)

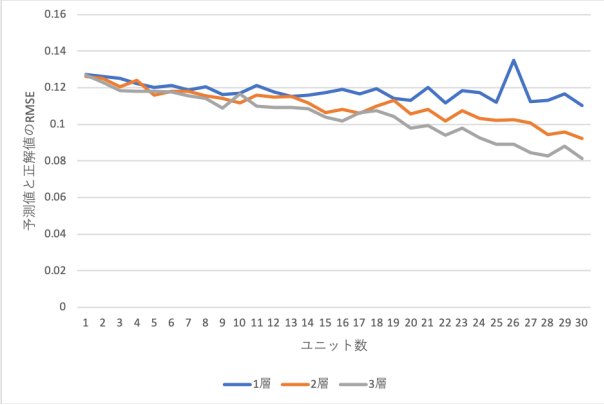

図 10　予備実験 3 の結果 (学習データ 4)

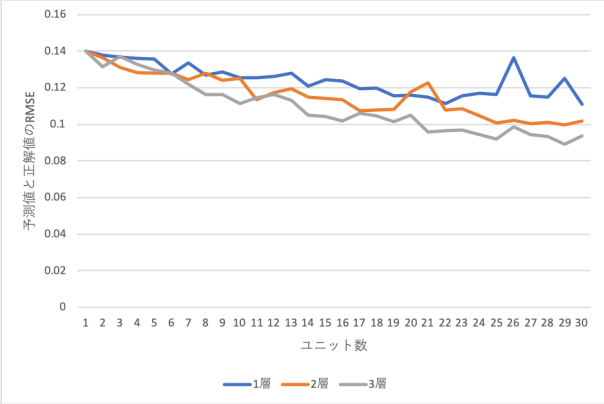

図 11　予備実験 3 の結果 (学習データ 5)

5.4　最適な NAS の探索範囲

予備実験 2，予備実験 3 の結果を踏まえ，NAS の探索範囲を以下の表 5 ように設定するのが良いと考えた．層数を 2 に固定する理由は，層数を 2 層から 3 層に増やすと学習時間が 1.5 倍になるのに対し，精度の差が少ないためである．また，ユニット数を 20-30 とする理由は，同じ層数の中でもユニット数大きいほど精度が良くなる傾向にあるが，学習時間の増加は比較的少ないためである．

上記の条件での NAS により出力されるモデルの例を図 12 に示す．これは，LSTM のユニット数が 29 の場合のモデルである．

表 5　最適な NAS の探索範囲

パラメータ	範囲
ラーニングレート	0.01 または 0.001
LSTM の層数	2(固定)
LSTM のユニット数	20-30

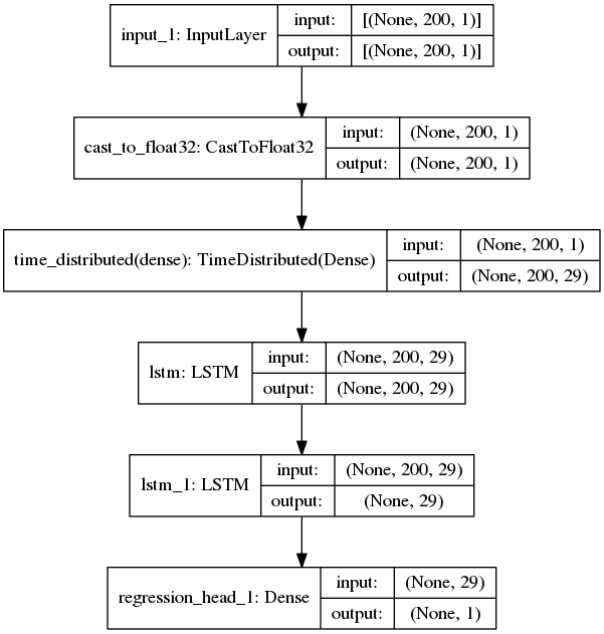

図 12　NAS により出力されたベストモデルの例

6　提 案 手 法

VMCPU 使用率予測モデル運用における提案手法のフローチャートを図 13 に示す．黄色部分が準備段階，オレンジ色部分が運用段階であり，パラメータはクラスタリングの基準作成時のクラスタ数を表す *n_clusters*，分岐における類似度の閾値を表す *max_similarity* の 2 種類である．

準備段階では，過去のデータすべてを細分化し，クラスタリングの基準の作成を行い，クラスタリングの基準に当てはめて得られたクラスタ番号の分布を保存する．その後，過去のデータをすべて使用して NAS を行い，ベストモデルを保存する．運用段階では，最初に直近のデータを取得し，細分化を行った後，準備段階で作成したクラスタリングの基準に当てはめ，得られたクラスタ番号の分布を過去の複数の分布と比較する．直近の分布と過去の分布のコサイン類似度の最大値が *max_similarity* 以上だった場合は，NAS を行う必要がないため，その時点で使用されたモデルを流用する．一方，直近の分布と過去の分布のコサイン類似度の最大値が *max_similarity* 未満だった場合は，過去のデータも含めて NAS を行い，ベストモデルを保存して使用する．この一連の流れを一定間隔で繰り返すというものである．

次に，フローチャート内で使用している言葉が本研究内で示している内容を，使用している技術と合わせて説明する．

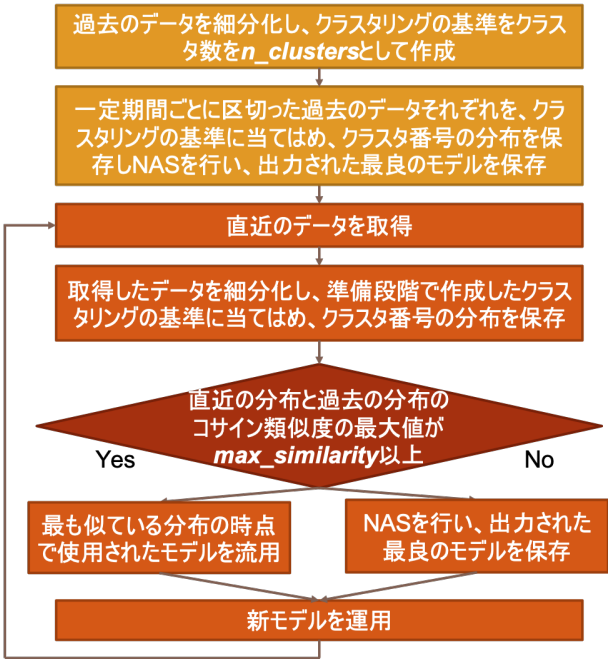

図 13　提案手法のフローチャート

6.1　細　分　化

本研究では，図 14 のように，データを 1 点ずつずらしながら学習元データ数と正解データ数の合計値ごとに区切る処理を細分化と呼ぶ．また，本研究では，学習元データ数を 200，正解データ数を 1 と設定しているため，201 点ごとに区切る処理を行っていることになる．

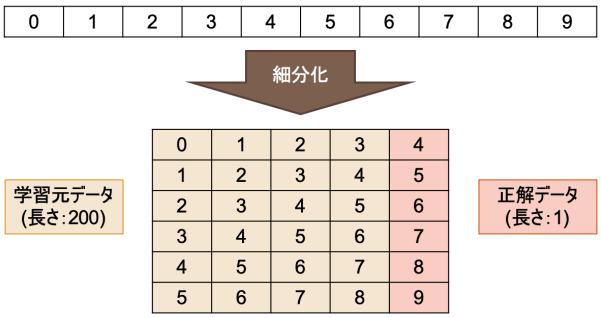

図 14　細分化のイメージ

このような処理を行う理由は，時系列データの機械学習で使用するデータの特徴にある．時系列データの学習を行う場合は通常，図 15 のように，学習元データ数と正解データ数を設定し，データを開始点を 1 点ずつずらしながら，長さ学習元データ数の学習元データとそれに対応する直後の長さ正解データ数の正解データのペアを作成し，学習させる．つまり，学習には長い波形をそのまま使うというわけではなく，細かい波形をいくつも使っているということになる．そこで本研究では，この特徴を踏まえ，長い時系列データを多数の細かい時系列データに変形することで，データの特徴を適切に判断できるのではないかと考えた．

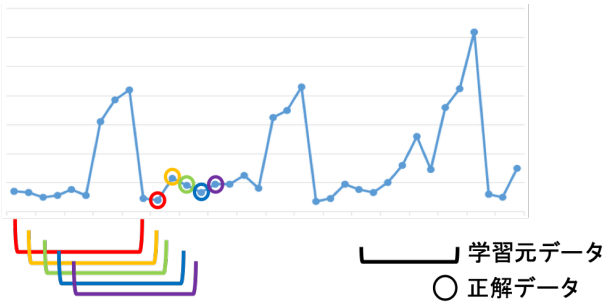

図 15　時系列データ学習の特徴

6.2　クラスタリングの基準

クラスタリングには，scikit-learn [18] の中にある sklearn.cluster.Kmeans クラスを使用した．このクラスの fit メソッドを使うことで，与えられたデータとクラスタ数を元にクラスタリングの基準を作成することができ，predict メソッドを使用することでその基準に基づいてクラスタ番号を割り振ることができる．似たデータは細分化後のクラスタリング結果も似ている [19] ため，基準に当てはめることで得られるクラスタ番号の分布を元にデータの類似度の判断を行う．

6.3　分布の類似度

分布の類似度の基準は，細分化後のデータをクラスタリングの基準に当てはめた際のクラスタ番号分布のコサイン類似度とする．コサイン類似度とは，ベクトルの内積を用いて類似度を計算する方法である．数値はベクトル同士の成す角度の近さを表現しており，-1 から 1 までの値をとるが，数値が大きくなるほど類似度が高いということになる．以下のような式で計算される．

$$cos(\vec{q},\vec{d}) = \frac{\vec{q}\cdot\vec{d}}{|\vec{q}|\cdot|\vec{d}|} = \frac{\sum_{i=1}^{|V|} q_i d_i}{\sqrt{\sum_{i=1}^{|V|} q_i^2}\cdot\sqrt{\sum_{i=1}^{|V|} d_i^2}}$$

6.4　NAS

NAS はライブラリとして AutoKeras を使用し，予備実験で決定した表 5 のような探索範囲で行った．その他の条件は，エポック数 100，バッチサイズ 100，ウィンドウ幅 200，最大トライアル数 100 とした．

7　実　　験

7.1　実 験 方 法

7.1.1　パラメータ

実験では，提案手法のパラメータを表 6 のように設定して行った．このように設定したところ，運用段階で NAS が行われる回数は 4 回となった．

表 6　実験での提案手法のパラメータ設定

パラメータ	値
n_clusters	50
max_similarity	0.94

7.1.2 実験環境

実験環境は表 7 の通りである．

表 7　実験環境

OS	CentOS Linux release 7.5.1804(Core)
サーバ	FUJITSU Primergy CX400 M1
CPU	Intel Xeon Haswell 2 ソケット 14 コア 2.6GHz E5-2697 145W
GPU	NVIDIA Tesla P100 16GB
Storage HDD	270GB read 0.21GB/s write 1.07GB/s
Memory	256GB DDR4 2133MHz
Python	3.6.8
CUDA	11.0

7.1.3 比較対象

実験では，提案手法以外に以下の値と正解値との RMSE を比較した．NAS を行わない場合のネットワーク構造は，LSTM2 層，ユニット数 30 とした．ユニット数 30 は，探索範囲の上限値である．

- 過去のデータで学習したモデルを使用し続ける場合の予測値
- 過去のデータで NAS 行い出力されたベストモデルを使用し続ける場合の予測値
- 各区間で毎回学習を行う場合の予測値
- 各区間で過去のデータを含めて毎回 NAS を行う場合の予測値
- 各区間で過去のデータを含めずに毎回 NAS を行う場合の予測値
- 1 つ前の値を繰り返しただけのデータ (repeat 値)

7.2 結　　果

上記のような条件で実験を行ったところ，結果は図 16 のようになった．縦軸は RMSE の平均値，横軸は区間番号である．また，表 8 は，各区間の RMSE の平均値の合計値と，合計値が小さい順に順位をつけたものである

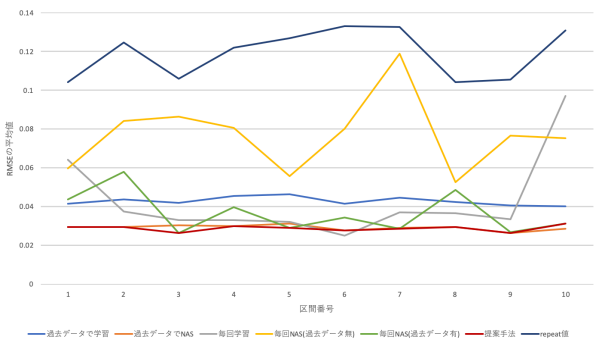

図 16　実験結果

この結果から，まず，毎回 NAS を行う 2 種類の結果を比較すると，NAS を行う場合は過去のデータを使用することが精度を向上させるために必要であることが読み取れる．また，毎回学習を行う場合と過去のデータを含めて毎回 NAS を行う場合の結果の比較から，NAS を行うことで通常の学習よりもモデルの精度を向上させることが可能であることが読み取れる．そして，過去データで NAS を行い出力されたベストモデルの精度が良いことも読み取れる．さらに，提案手法では適切なタイミングで NAS を行っており，NAS を取り入れることでより高精度な予測モデルの運用が可能であることも読み取れる．

表 8　各区間の RMSE の平均値の合計値と順位

種類	合計値	順位
過去データで学習	0.4292	4
過去データで NAS	0.2923	2
毎回学習	0.4292	5
毎回 NAS(過去データ無)	0.7711	6
毎回 NAS(過去データ有)	0.3670	3
提案手法	0.2881	1
repeat 値	1.1909	7

これらのことより，NAS を取り入れることで，より高精度な予測モデルの運用が可能になることが確認できた．ただし，過去のデータで NAS 行い出力されたベストモデルを使用し続ける場合の方が過去のデータを含めて毎回 NAS を行う場合よりも高精度という結果になったので，今後も実験を行う必要があると考えられる．

7.3 考　　察

実験結果から，過去のデータを使用して NAS を行うだけでも十分効果があるが，より高精度な予測モデルの運用をしたい場合は提案手法を導入するのが良いと考えられる．

ただし，過去のデータを使用して NAS を行い出力されたベストモデルを使用し続ける場合の方が過去のデータを含めて毎回 NAS を行う場合よりも高精度という結果になったことについては原因は現段階では不明であるため，今後も調査を行う必要がある．

8　まとめと今後の課題

仮想環境における VM の CPU 使用率を予測モデルの運用において，Neural Architecture Search (NAS) を取り入れる手法の検討を行った．

まず，複数のデータセットに対して NAS を行い，出力されたベストモデルの構造の比較を行い，NAS の有効性を確認した．次に，モデルのネットワーク構造と学習時間/予測精度の関係を調べる実験を 90 通りのネットワーク構造で行い，運用時における NAS 探索範囲の方針を定めた．そして，VM の CPU 使用率予測モデルの運用において，NAS を取り入れる手法を提案し，フローチャートに沿って実験を行った．その結果，NAS を取り入れることで，より高精度な予測モデルの運用が可能になることが確認できた．

今後の課題としては，まず，過去のデータを使用して NAS を行い出力されたベストモデルを使用し続ける場合の方が過去のデータを含めて毎回 NAS を行う場合よりも高精度になる原因の調査が挙げられる．他にも，提案手法のフローチャートの

改善や，実際の環境で提案手法に沿った予測モデルの運用を行い，どの程度の改善が見込まれるかのシミュレーションも行いたいと考えている．

謝　　辞

本研究の一部はお茶の水女子大学と富士通株式会社との共同研究契約に基づくものであり，JST CREST JPMJCR1503 の支援を受けたものである．

文　　献

[1] Josh Whitney and Pierre Delforge. Data center efficiency assessment. *Issue paper on NRDC (The Natural Resource Defense Council)*, 2014.

[2] Rahul Ghosh and Vijay K Naik. Biting off safely more than you can chew: Predictive analytics for resource over-commit in iaas cloud. In *2012 IEEE Fifth International Conference on Cloud Computing*, pages 25–32. IEEE, 2012.

[3] 児玉宏喜, 鈴木成人, 福田裕幸, 吉田英司, et al. マイグレーションを利用したデータセンタの高効率運用手法の提案とオーバコミット時における vm の性能評価. **研究報告システムソフトウェアとオペレーティング・システム** *(OS)*, 2018(13):1–7, 2018.

[4] Hiroyoshi Kodama, Hiroshi Endo, Shigeto Suzuki, and Hiroyuki Fukuda. High efficiency cloud data center management system using live migration. In *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, pages 733–738. IEEE, 2017.

[5] Arijit Khan, Xifeng Yan, Shu Tao, and Nikos Anerousis. Workload characterization and prediction in the cloud: A multiple time series approach. In *2012 IEEE Network Operations and Management Symposium*, pages 1287–1294. IEEE, 2012.

[6] Fahimeh Farahnakian, Pasi Liljeberg, and Juha Plosila. Lircup: Linear regression based cpu usage prediction algorithm for live migration of virtual machines in data centers. In *2013 39th Euromicro Conference on Software Engineering and Advanced Applications*, pages 357–364. IEEE, 2013.

[7] Ji Xue, Feng Yan, Robert Birke, Lydia Y Chen, Thomas Scherer, and Evgenia Smirni. Practise: Robust prediction of data center time series. In *2015 11th International Conference on Network and Service Management (CNSM)*, pages 126–134. IEEE, 2015.

[8] Salam Ismaeel and Ali Miri. Real-time energy-conserving vm-provisioning framework for cloud-data centers. In *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0765–0771. IEEE, 2019.

[9] Thomas Elsken, Jan Hendrik Metzen, Frank Hutter, et al. Neural architecture search: A survey. *J. Mach. Learn. Res.*, 20(55):1–21, 2019.

[10] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 4780–4789, 2019.

[11] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018.

[12] Mohammad Shahrad, Rodrigo Fonseca, Íñigo Goiri, Gohar Chaudhry, Paul Batum, Jason Cooke, Eduardo Laureano, Colby Tresness, Mark Russinovich, and Ricardo Bianchini. Serverless in the wild: Characterizing and optimizing the serverless workload at a large cloud provider. *arXiv preprint arXiv:2003.03423*, 2020.

[13] Azure/azurepublicdataset: Microsoft azure traces. `https://github.com/Azure/AzurePublicDataset`.

[14] Haifeng Jin, Qingquan Song, and Xia Hu. Auto-keras: An efficient neural architecture search system. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1946–1956. ACM, 2019.

[15] Autokeras. `https://autokeras.com/`.

[16] Home - keras documentation. `https://keras.io/ja/`.

[17] Tensorflow. `https://www.tensorflow.org/`.

[18] scikit-learn — machine learning in python. `https://scikit-learn.org/`.

[19] Yuriko Takahashi, Shigeto Suzuki, Takuji Yamamoto, Hiroyuki Fukuda, and Masato Oguchi. Time-series data regression modeling method for efficient operation of virtual environments. In *2021 15th International Conference on Ubiquitous Information Management and Communication (IMCOM)*, pages 1–6. IEEE, 2021.