F-038

テンソルネットワークを用いた電力需要予測モデルの軽量化に関する研究
Tensor Network-Based Model Compression for Electrical Load Forecasting

渡 大地 [1)]　谷本 寛樹 [1)]　大久保 毅 [2)]　藤堂 眞治 [2)]　西山 薫 [1)]
Daichi Watari　Hiroki Tanimoto　Tsuyoshi Okubo　Synge Todo　Kaoru Nishiyama

1　はじめに

カーボンニュートラル社会の実現に向けて，次世代電力システムへの転換が期待されている．特に近年では，VPP (Virtual Power Plant) と呼ばれるシステムが有望視されている [1]．VPP とは，需要家群 (工場や家庭など) に設置された多数の太陽光発電や蓄電池といったエネルギーリソースを一括管理して大規模な発電所のように機能させることで，利益の創出や電力需要バランスの効率的な管理を実現するものである．VPP の運用では，まず各需要家毎の電力需要予測を行い，その予測情報をもとに蓄電池などの制御対象機器の計画立案を行う．したがって，電力需要予測の精度やコストが，VPP の精度や品質，コストにもたらす影響は非常に大きい．

近年の電力需要予測研究では，ニューラルネットワーク (NN; Neural Network) に基づく手法が注目されている．代表的な手法には MLP (Multi-Layer Perceptron) や LSTM (Long Short Term Memory)，Transformer に基づくものがあり，従来の機械学習手法より一般に精度が高いことが示されている [2, 3]．また，電力需要の多峰性 (例: 在宅勤務・出社のような観測不能な状態変数による変化) を考慮するために，単一の予測値の代わりに，混合型の確率分布を出力する混合密度ネットワーク (MDN: Mixture Density Network) に基づく手法も提案されている [4]．このような研究によって電力需要予測の精度は向上する一方，実運用の観点では NN モデルの学習コストが大きな課題となっている．VPP では需要家が追加されるたびに新たな予測モデルの構築が必要になる．また，時間の経過によって需要家の特性が変化することで予測モデルの劣化が発生することが知られており，定期的な再学習も必要である [5]．大量のパラメータを持つ NN モデルは学習に時間を要するため，モデル学習や再学習のたびに学習コストが発生し，運用コストを大きく増加させてしまう．

NN モデルの運用コストを下げるために，モデル軽量化技術の必要性が高まっている [6]．モデル軽量化では，モデルの精度をなるべく保ったまま，処理時間やメモリ使用量を削減することを目的とする．代表的な手法としては，モデルの重みパラメータやニューロンを削減してサイズを縮小する枝刈り [7, 8] や，特異値分解によって重みパラメータを分解および削減する手法 [9] がある．また，近年ではテンソルネットワークと呼ばれる，ある大きなテンソル (ベクトルや行列を一般化した概念) を小さなテンソルの縮約ネットワークで表現する手法を用いてパラメータ削減を行う方法についても研究されている．テンソルネットワークによる軽量化は画像処理や言語処理といった分野でその優れた軽量化性能が示されたが [10, 11]，時系列予測分野への適用例は少ない．

1)　京セラ株式会社 みなとみらいリサーチセンター
〒 220-0012　神奈川県横浜市西区みなとみらい 3-7-1
2)　東京大学 大学院理学系研究科
〒 113-0033　東京都文京区本郷 7-3-1

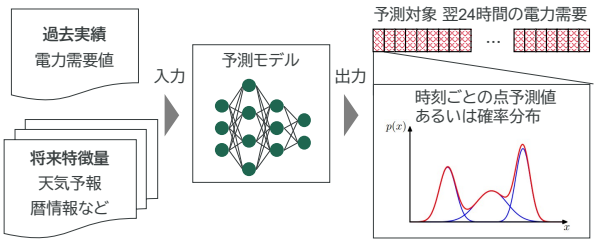


図 1: 電力需要予測タスクの概要図

そこで，本稿では電力需要予測に向けたニューラルネットワークモデルの学習効率化を目的として，テンソルネットワークに基づくモデル軽量化手法である TT (Tensor-Train) 分解 [12] を適用する．具体的には，確率的な電力需要予測に向けて，以前の研究で開発された埋め込み型 MDN モデル [4] に対して TT 分解を適用することで，モデルの軽量化を行う．軽量化したモデルをスクラッチから学習することで，学習の効率化を目指す．さらに，代表的なモデル軽量化手法である枝刈りや特異値分解と比較を行い，TT 分解の有効性を評価する．

本稿の構成について述べる．まず第 2 章で電力需要予測の定式化を行う．次に第 3 章で対象とする電力需要予測モデルとテンソルネットワークによるモデル軽量化手法の詳細について述べる．そして，第 4 章で評価実験の結果を示す．最後に第 5 章で結論を述べる．

2　電力需要予測の概要

電力需要予測の目的は，与えられた入力をもとに，将来の電力需要の変化を予測することである．図 1 に本研究で対象とする電力需要予測の概要を示す．VPP では各種電力市場に入札するためにも前日時点で機器運転・入札に関する計画立案を完了させる必要がある．したがって，需要家ごとに翌日 24 時間分の電力需要の時系列を予測するモデルを構築する．予測時刻において，過去の電力需要実績と予測対象時刻の天気予報情報 (外気温など) や暦情報 (曜日，休日情報など) を入力として予測モデルに与え、予測モデルの出力として翌 24 時間の電力需要時系列を得る．このとき，出力は時刻毎の点予測値，あるいは確率分布とする．予測出力の時間粒度は電力市場の要件に合わせて 30 分粒度とする．

次に，電力需要予測の定式化を行う．対象とする需要家のインデックスを $p\,(p \in \{1, \ldots, P\})$，時刻を t とし，T ステップ先まで予測を行うこととする．需要家 p, 時刻 t の電力需要を $y_{p,t}$ とおくと，求める目的変数の系列は $y_{p,t+1:t+T}$ となる．また，電力需要を予測するために使用する特徴量である共変量ベクトル (天気予報や暦情報など) を $\boldsymbol{c}_{p,t}$ とおき，予測に利用可能な過去のデータ長を T' とすると，電力需要予測では以下の関数 $f(\cdot)$ を学習することとなる．

$$\boldsymbol{x}_{p,t} \overset{\text{def}}{=} [y_{p,t-T':t}, \boldsymbol{c}_{p,t+1:t+T}] \xrightarrow{f(\cdot)} y_{p,t+1:t+T} \qquad (1)$$

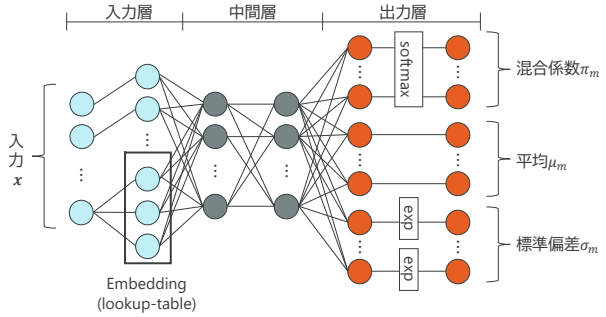


図 2: MDN モデルのアーキテクチャ [4]

ここで，$\boldsymbol{x}_{p,t}$ は予測の入力に用いる特徴量をまとめた説明変数ベクトルである．本研究では，関数 $f(\cdot)$ にニューラルネットワークを用いて学習を行う．

3　電力需要予測モデルの軽量化

本節では，まずベースとなる MDN モデルの説明を行い，その後，適用する TT 分解の詳細について述べる．

3.1　混合密度ネットワークモデル

VPP では，予測誤差が計画立案の品質にもたらす影響が非常に大きい．リスク低減のためにも予報値の出力だけでなく確率分布などの出力も望まれる．また，需要家 (特に住宅など) における電力需要は，一般に入手が難しい人の行動や在室・在宅状況にも依存する．これらの説明変数が入手できない場合，電力需要は説明変数に対して多峰的に分布する可能性がある．

本研究では上記の事象を考慮するために，先行研究で提案された MDN に基づく電力需要予測モデルを用いる [4]．MDN では，目的変数 y が M 個のガウス分布を重ね合わせた確率分布に従うと仮定する．具体的には次式の確率密度関数 $P(\boldsymbol{x})$ を用いる．

$$P(\boldsymbol{x}) \overset{\text{def}}{=} \sum_{m=1}^{M} \pi_m(\boldsymbol{x}) \mathcal{N}(y_t|\mu_m(\boldsymbol{x}), \sigma_m^2(\boldsymbol{x})) \quad \forall t \tag{2}$$

ここで，$\pi_m(\cdot)$ は m 番目のガウス分布の混合係数，$\mathcal{N}(y_t|\cdot,\cdot)$ は平均 $\mu_m(\cdot)$，標準偏差 $\sigma_m(\cdot)$ のガウス分布であり，$\sum_{m=1}^{M} \pi_m(\cdot) = 1$, $\sigma_m(\cdot) \geq 0$ $\forall m$ を満たす．なお，これらは説明変数ベクトル $\boldsymbol{x}$ の関数である．これにより，多峰性を考慮した確率密度関数を出力できる．

図 2 に MDN を用いたモデルのアーキテクチャ図を示す．本研究では需要家 p 毎にモデルを構築する．モデルのベースは多層パーセプトロン $f(\cdot)$ である．入力側に一部のデータをユークリッド空間に埋め込む Embedding 層を備えており，出力は以下の式で示すような混合ガウス分布のパラメータである．

$$f(\boldsymbol{x}_{p,t}) = (\pi_{p,1} \cdots \pi_{p,M}\, \mu_{p,1} \cdots \mu_{p,M}\, \sigma_{p,1} \cdots \sigma_{p,M})^{\mathsf{T}} \tag{3}$$

なお，入力は先行研究 [4] と同様に，過去の電力需要 (過去 7 日間の予測対象時刻と同時刻値)，予測対象時刻の外気温予報値，日付情報 (各日付を正弦値，余弦値に変換)，時刻情報，曜日情報，休祝日情報を用いる．時刻，曜日，休祝日情報は Embedding 層にてルックアップテーブルによって各々 V 次元のユークリッド空間に埋め込む．損失関数には学習用データセット $\mathcal{D}$ に対する負の対数尤度 $-\Sigma_{(\boldsymbol{x},y)\in\mathcal{D}} \log P(y|\boldsymbol{x})$ を設定することで，混合ガウス分布のパラメータを学習する．

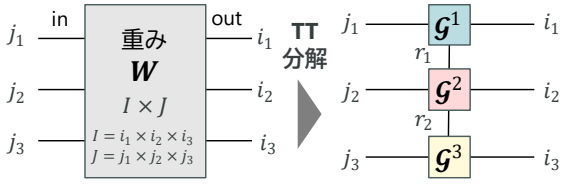


図 3: TT 分解の例 (3 ノードに分解した例)

3.2　テンソルネットワークによる重み行列の圧縮

本研究では，テンソルネットワークの一種である TT 分解 [12] を埋め込み型 MDN モデルの中間層である全結合層に対して適用することで，モデルパラメータの削減，ひいては学習の効率化を図る．以下では，全結合層への TT 分解の適用について説明を行う．なお以降では，通常のスカラーやベクトル，行列に対して区別を行うために，花文字を用いてテンソルを表記する (例: $\mathcal{A}$)．

まず，ある全結合層における線形変換は，出力ベクトル $\boldsymbol{u} \in \mathbb{R}^{I\times 1}$ と入力ベクトル $\boldsymbol{h} \in \mathbb{R}^{J\times 1}$ を用いて以下の関係式で表される．

$$\boldsymbol{u} = \boldsymbol{W}\boldsymbol{h} + \boldsymbol{b} \tag{4}$$

ここで，$\boldsymbol{W} \in \mathbb{R}^{I\times J}$ と $\boldsymbol{b} \in \mathbb{R}^{I\times 1}$ はそれぞれ全結合層のパラメータである重みとバイアスを意味する．重みはバイアスよりもはるかに大きなパラメータ数となるため，ここでは重み行列 $\boldsymbol{W}$ への TT 分解の適用を考える．

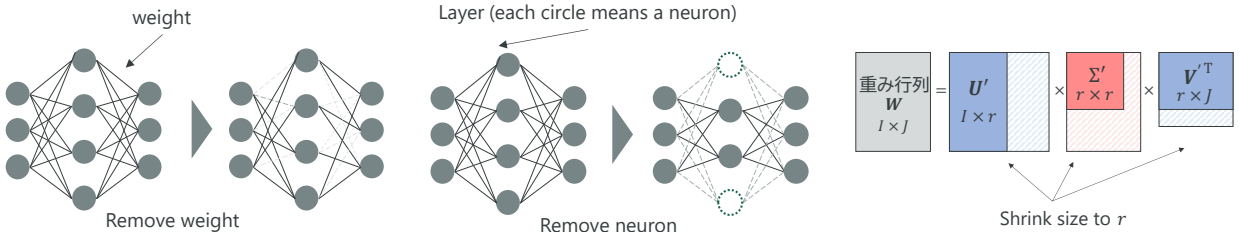
次に，重み行列 $\boldsymbol{W}$ をテンソル $\mathcal{W}$ に変形する．出力と入力の次元がそれぞれ $I = \prod_{k=1}^{d} i_k$, $J = \prod_{k=1}^{d} j_k$ のような要素積で表せる時，重みテンソル $\mathcal{W}$ には以下のような分解 (TT 分解) が存在する．

$$\mathcal{W}_{i_1,\ldots i_d,j_1,\ldots j_d} = \sum_{\alpha_0,\ldots,\alpha_d} \mathcal{G}^1_{i_1,j_1,\alpha_0,\alpha_1} \cdots \mathcal{G}^d_{i_d,j_d,\alpha_{d-1},\alpha_d} \tag{5}$$

ここで，$\mathcal{G}^k$ は分解後のテンソルを表し，ノードと呼ぶ．d は分解後のテンソル数，つまりノード数を意味する．また，ノードのインデックスを指す α_k は $\alpha_k = 1, \ldots, r_k$ の範囲をとり，各ノードの共通のインデックス α_k を足し上げる操作を縮約と呼ぶ．それらの要素数 $\{r_k\}_{k=0}^{d} \in \mathbb{Z}_+$ を TT 分解のランクという．なお，TT 分解では $r_0 = r_d = 1$ となり，その他の要素数は設定可能なハイパーパラメータである．例として，図 3 にノード数 $d = 3$ に設定した際の例を示す．3 つのノードの縮約計算によって元の重み $\boldsymbol{W}$ を表せる．最終的に TT 分解によって式 (4) は，出力と入力，バイアスをそれぞれテンソル $\mathcal{U}, \mathcal{H}, \mathcal{B}$ へ変換した上で，以下のように表せる．

$$\mathcal{U}_{i_i,\ldots,i_d} = \sum_{j_1,\ldots,j_d} \sum_{\alpha_0,\ldots,\alpha_d} \prod_k \mathcal{G}^k_{i_k,j_k,\alpha_{k-1},\alpha_k} \mathcal{H}_{j_1,\ldots,j_k} + \mathcal{B}_{i_1,\ldots,i_d} \tag{6}$$

TT 分解によるモデルパラメータの圧縮は，あるテンソル次元 k におけるテンソル $\mathcal{G}_k$ のサイズをランク r_k で制限することによって行う．もし，ランク r_k が十分に小さければ TT 分解によってモデルパラメータ数と計算量の削減が可能である [10]．ランクは重要なハイパーパラメータであり，小さくすればするほど高い圧縮率を得ることができるが，反対にモデルの表現力は犠牲にすることとなる．モデルの精度を落とさない範囲で，なるべく最小のランクに設定することが基本である．



(a) 非構造枝刈り SNIP　**(b)** 構造枝刈り SNAP　**(c)** 特異値分解 SVD

図 4: 比較するモデル軽量化手法

4　評価実験

TT 分解によるモデル軽量化が学習や精度・モデルサイズにどのような影響を及ぼすかを調べた．

4.1　評価条件

評価用データセットには国内 56 戸の住宅で測定された 30 分刻みの電力需要データを使用する．学習期間を 2018 年 12 月 30 日-2020 年 6 月 23 日，検証期間を 2020 年 6 月 24 日-同年 6 月 30 日としてモデルの学習を行う．その後, 評価期間 2020 年 7 月 1 日-同年 7 月 31 日に対して，毎日 $T = 48$ ステップずつ予測を実行して精度を評価する．なお，予測の評価指標には負の対数尤度 NLL (Negative-Log Likelihood) と平均絶対パーセンテージ誤差 MAPE (Mean Absolute Percentage Error) を用いる．前者については出力の混合ガウス分布と真値を用いて算出し，後者については混合ガウス分布から確率が最も高い最頻値へ変換を行った後，真値を用いて算出する．どちらも小さいほど良い値となる指標である．

MDN モデルのハイパーパラメータについて，埋め込み次元は $V = 3$ に設定する．最適化器には Adam，活性化関数には Leaky ReLU を用い，圧縮対象となる中間層は 1 層 50 ニューロンの全結合層とした．学習エポック数は最大 2000 とし，学習終了時点で最も検証損失の少ないパラメータを用いて予測を行う．

4.2　比較手法

本研究では，何もしていないモデルと TT 分解を用いる提案手法に加え，学習前に軽量化を行う代表手法として図 4 に示す 3 手法を比較する．以下に詳細を述べる．

- 元のモデル (Original) [4]
  4.1 節に記述した設定を用いて学習を行う．
- 提案手法 (TT)
  3.2 節に示された手法で重み行列を分解する．なお，本稿ではノード数 $d = 3$ とし，入出力のサイズについては $[i_1, i_2, i_3] = [j_1, j_2, j_3] = [5, 2, 5]$ に固定した．ランクについては一律の値 $\{r_k\}_{k=1}^{d-1} = R \in \mathbb{Z}_+$ を設定し，R のみを変化させることで圧縮率を制御することとした．
- 非構造的枝刈り手法 (SNIP: Single-shot Network Pruning) [7]
  学習前のモデルに対して枝刈りを行う代表的な手法．枝刈り対象はある層の重みパラメータ $\boldsymbol{W} = \{w_q\}_{q=1}^{Q}$ である．学習データ $\mathcal{D}$ に対する各重みの重要度 s_q を計算して，s_q が大きい順に K 個の重みのみ可変とし，それ以外を 0 埋めして不変とすることで最終的に疎行列 $\boldsymbol{W}'$ を得る．重要度 s_q には，各重みの勾配 $g_q(\boldsymbol{W}; \mathcal{D})$ から計算した，$s_q = |g_q(\boldsymbol{W}; \mathcal{D})| / \sum_q |g_q(\boldsymbol{W}; \mathcal{D})|$ を用いる．
- 構造的枝刈り手法 (SNAP: Single-shot Network Architecture Pruning) [8]
  SNIP の枝刈り対象を重みからニューロンに変更した手法．あるニューロン (活性化出力) h_q の重要度 s_q を計算して大きい順に K 個選ぶ．最終的に K 個のニューロンをもつ全結合層として再構成する．また，重要度 s_q は活性化出力 $h_q(\boldsymbol{W}; \mathcal{D})$ と損失関数の値 L を用いて $s_q = |h_q(\boldsymbol{W}; \mathcal{D})| / |L|$ と定義する．
- 特異値分解に基づく重み行列分解手法 (SVD: Singular Value Decomposition) [9]
  重み行列 $\boldsymbol{W}$ に対して特異値分解を適用し，$\boldsymbol{W} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathsf{T}}$ のように分解する．ここで，$\boldsymbol{U} \in \mathbb{R}^{I \times I}, \boldsymbol{V}^{\mathsf{T}} \in \mathbb{R}^{J \times J}$ は直交行列，$\boldsymbol{\Sigma} \in \mathbb{R}^{I \times J}$ は対角成分に特異値を並べた行列である．$\boldsymbol{\Sigma}$ の対角成分の個数を r 個に制限して図 4(c) のように再構成した $\boldsymbol{U}'\boldsymbol{\Sigma}'\boldsymbol{V}'^{\mathsf{T}}$ を元に $\boldsymbol{U}'\sqrt{\boldsymbol{\Sigma}'}$ と $\boldsymbol{V}'^{\mathsf{T}}\sqrt{\boldsymbol{\Sigma}'}$ の 2 つの行列を学習可能なパラメータとして、スクラッチ (初期状態) から学習する．

表 1: 軽量化時のモデルサイズと精度の関係 (精度において最良の値は太字，2 番目に良い値は斜体で表記)

手法	対象中間層の圧縮率 (%)	モデル全体のパラメータ数	MAPE (%)	NLL (-)
Original	100.00	4.79K	32.11	292.63
SNIP	4.90	4.79K	*31.96*	**290.92**
SNAP	6.00	1.41K	33.22	293.30
SVD	5.88	2.42K	35.01	297.12
TT	4.08	2.35K	**31.74**	*292.26*

4.3　実験結果

まず，表 1 に対象中間層を軽量化した際の各手法の結果を示す．圧縮率は，もともとの重み数 Q に対する軽量化後の重み数 Q' の比率 Q'/Q として定義する．一般的に学習前のモデル軽量化では，モデルを軽量化すればするほど精度が低下すると予想されており [7]，公正な比較のためにも TT の圧縮率を下回らない範囲でその他の軽量化手法を適用する．そこでまず，提案手法は $R = 1$ に設定して最大限圧縮を行った．次に，SVD は $r = 1$，SNIP，SNAP ではそれぞれ残す重み数/ニューロン数が $K = 125, K = 3$ 個となるように設定した．結果より，まず同程度まで圧縮した際のパラメータ数は SNAP が最も小さくなった．これは，ニューロンを削除することで前層からの接続だけでなく後層への接続も削除され，結果的に重みの数が大きく削減されるためである．一方，TT は 2 番目に小さいモデルを構築することができ，全体で見た時に元のモデルの約 49%まで削減できた．なお，SNIP は削減対象となった重みを 0 埋めして固定するため，更新する重みの数は削減されるが，全

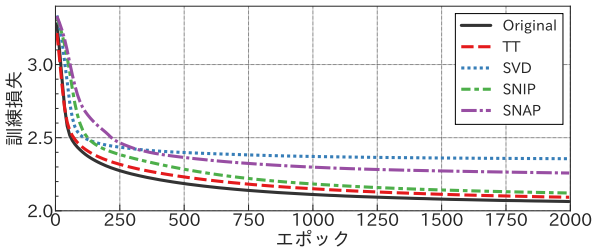


図 5: 訓練損失の変化

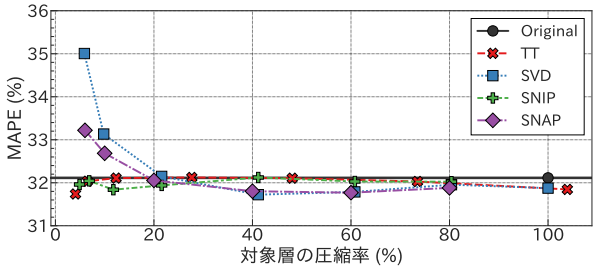


図 6: 圧縮率を変化させた時の MAPE の変化

体のパラメータ数は削減されない．理論的にはパラメータ数だけ NN モデルの計算量は増加するため，SNAP や SVD，TT では計算量を削減可能である．次に，精度指標の MAPE や NLL については TT は対象中間層の圧縮率が最小にも関わらず，他手法と比べても良い結果となった．TT は元のモデルよりも MAPE，NLL の両方を改善し，他手法と比較しても MAPE は最良，NLL は次善の値となった．

図 5 に訓練損失の変化を示す．損失は 56 需要家分の全モデルで 3 試行行った際の平均値である．結果より，SVD は高止まりしてしまい，学習がうまく進んでいないことがわかる．SNIP や SNAP では損失の減少がやや緩やかである．そして，TT は元のモデルの訓練損失とほぼ同じ減衰曲線を示しており，学習特性が元のモデルからそこまで変化していないことがわかる．以上，表 1 と図 5 の結果から，TT は Original より訓練誤差は大きいにも関わらず精度を改善しており，またその他の軽量化手法と比べても精度が最良あるいは次善となっている．したがって TT は過学習を避けやすく，汎化性能が良い手法であると考えられる．

最後に，設定圧縮率を変化させた実験について述べる．上述の設定から各種ハイパーパラメータを変更して圧縮率を変化させ，それぞれ実験を行った．図 6 に MAPE の変化，図 7 に検証損失が最小となるエポック数の結果を示す．Original については圧縮率は変化しないため，水平線で示している．まず図 6 より，精度については SVD や SNAP が圧縮するにつれて悪化しているのに対し，SNIP と TT はほとんど変化しておらず優位性を示すことがわかった．次に図 7 より，検証損失が最小のエポックについて，他手法のほとんどが軽量化する (図の左領域) につれてエポック数が大きくなっているのに対し，TT は最も軽量化した場合を除いてほとんど変化しておらず，元のモデルと同等レベルである．検証損失が早く最小になるほど，学習エポック数を削減し，ひいては学習時間を削減することができる．以上より，TT は他の軽量化手法よりも圧縮率に対して，精度だけでなく学習特性も頑健であることがわかった．

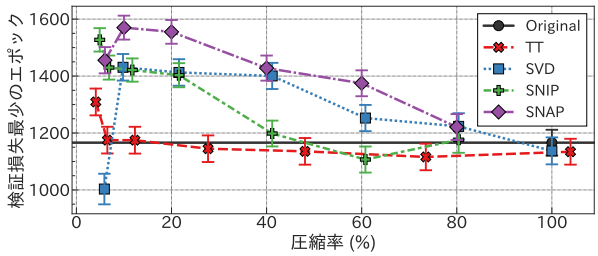


図 7: 圧縮率を変化させた時の検証損失最小エポック

5 結論

本稿では，テンソルネットワークに基づくモデル軽量化手法である TT 分解について評価を行った．枝刈りや特異値分解など代表的なモデル軽量化と比較して，TT 分解はモデルの圧縮性能と精度のバランスが優れていることが分かった．また，他の軽量化手法と異なり TT 分解は圧縮に伴う学習エポック数の増加がみられず，スクラッチから学習する場合の学習時間低減に効果があると考えられる．

今後は，より大きな時系列予測向け NN モデルに TT 分解を適用し，学習時間が短く高精度で軽量なモデルの実現を目指す．また，特異値などに基づくスペクトラム分析を行い，なぜ TT 分解が他の軽量化手法と性質が異なるか明らかにすることを考えている．

参考文献

[1] 泉谷聡史ら，“需要電力予測誤差を考慮した蓄電池充放電計画の検討と VPP 実証実験の報告，” エネルギー・資源学会研究発表会講演論文集, 2021, pp. 146–150.

[2] O. Rubasinghe et al., “Highly accurate peak and valley prediction short-term net load forecasting approach based on decomposition for power systems with high PV penetration,” *Appl. Energy*, vol. 333, p. 120641, 2023.

[3] E. Giacomazzi, F. Haag, and K. Hopf, “Short-Term Electricity Load Forecasting Using the Temporal Fusion Transformer: Effect of Grid Hierarchies and Data Sources,” in *Proc. e-Energy*, 2023, pp. 353–360.

[4] 岸本政徳ら，“混合密度ネットワークを用いた消費電力量予測手法の開発，” 令和 4 年電気学会電力・エネルギー部門大会, 2022, p. 312.

[5] I. Žliobaitė, M. Pechenizkiy, and J. Gama, “An Overview of Concept Drift Applications,” *Big Data Analysis: New Algorithms for a New Society*, Springer, 2016, pp. 91–114.

[6] L. Shen et al., “On Efficient Training of Large-Scale Deep Learning Models: A Literature Review,” in *arXiv*, 2023. arXiv:2304.0358 [cs.LG].

[7] N. Lee, T. Ajanthan, and P. H. S. Torr, “SNIP: Single-shot Network Pruning based on Connection Sensitivity,” in *Proc. ICLR*, 2019.

[8] S. Verdenius, M. Stol, and P. Forré, “Pruning via Iterative Ranking of Sensitivity Statistics,” in *arXiv*, 2020. arXiv:2006.00896[cs.LG].

[9] M. Khodak, N. Tenenholtz, L. Mackey, and N. Fusi, “Initialization and Regularization of Factorized Neural Layers,” in *Proc. ICLR*, 2021.

[10] A. Novikov, D. Podoprikhin, A. Osokin, and D. Vetrov, “Tensorizing Neural Networks,” in *arXiv*, 2015. arXiv:1509.06569[cs.LG].

[11] X. Liu and K. K. Parhi, “Tensor Decomposition for Model Reduction in Neural Networks: A Review [Feature],” *IEEE Circuits Syst. Mag.*, vol. 23, no. 2, pp. 8–28, 2023.

[12] I. V. Oseledets, “Tensor-Train Decomposition,” *SIAM J. Sci. Comput.*, vol. 33, no. 5, pp. 2295–2317, 2011.