ジャンプ拡散過程モデルに対する深層学習に基づくパラメータ推定手法

東京都市大学・知識工学部 田村 慶信（Yoshinobu Tamura）†
†Faculty of Knowledge Engineering, Tokyo City University
東京都市大学・総合理工学研究科 曽根 寛喜（Hironobu Sone）††
††Graduate School of Integrative Science and Engineering, Tokyo City University
東京都市大学・総合理工学研究科 杉崎 航大（Kodai Sugisaki）†††
†††Graduate School of Integrative Science and Engineering, Tokyo City University
鳥取大学・名誉教授 山田 茂（Shigeru Yamada）††††
††††Emeritus Professor, Tottori University

1　はじめに

オープンソースソフトウェア（Open Source Software，以下 OSS を略す）は，低コストかつ短納期により開発できることから，クラウドサービス，組込みシステム，サーバ，およびアプリケーションソフトウェアなど，様々な場面で利活用されている．しかしながら，ソースコードが公開されているため，そのセキュリティや品質上の問題に多くの企業が悩まされているという現状もある．こうした OSS を利用したソフトウェアシステムの品質を定量的に評価する手法は未だ提案されておらず，職人的・試行錯誤的に運用が行われているのが現状である．また，近年の OSS の運用環境について考えた場合，OSS 単体ではなく様々な連携ソフトウェアとのネットワーク経由での通信状況などを考慮することは非常に重要となる．特に，OSS は多くのバージョンアップを繰り返しながら成長する場合が多い．バージョンアップにも，メジャーバージョンアップ，マイナーバージョンアップ，バグフィックスバージョンアップのように，機能の追加やセキュリティホールの除去など，その目的も様々である．このように，OSS はバージョンアップに伴い，その内部構造やアルゴリズムが変化するため，それに応じて OSS の特性や利用環境も変わってくる．ビッグデータを考慮した OSS に対する最近の研究動向としては，サービス形態，性能評価などを対象とした文献はいくつか提案されている [1]．ビッグデータを有する OSS 開発をプロジェクトマネジメントの観点から開発工数に基づき定量的に評価する手法は未だ提案されていない．特に，ビッグデータを考慮することを考えた場合，開発保守のための工数が大きくなるだけでなく，ログインするユーザの突発的な増減なども考慮する必要がある．ビッグデータを運用する OSS プロジェクトの場合，そこに潜む規則性や秩序を見い出し，複数のパラメータをもつ数理モデルを適用することは可能であっても，パラメータ推定などの問題から，実利用上適用することは非常に難しい．このような，ビッグデータを有するオープンンソースプロジェクト特有の状況に対して，定常的なノイズと突発的なノイズの 2 種類の雑音を適用する．これにより，ビッグデータからの外的要因を考慮しつつ，大規模 OSS システムを運用するプロジェクトを定量的に評価することが可能となる．

上記のような背景から，過去に，こうした OSS の環境変化を想定したジャンプ拡散過程モデルを提案してきた．本論文では，ビッグデータを扱う OSS の運用上の特徴を包括するためのジャンプ拡散過程モデルについて議論する．その際，最尤推定法と深層学習を組み合わせたジャンプ拡散過程モデルのパラメータ推定法を提案する．さらに，実際の開発工数データを利用した提案手法の数値例を示す．

2　ジャンプ拡散過程モデル

ソフトウェアの信頼性を評価するための数理モデルとして，これまでに数百におよぶソフトウェア信頼性モデルが提案されてきた [2,3]．ソフトウェア信頼度成長モデルはフォールトデータに基づいてソフトウェア信頼性が評価されるが，フォールト発生の原因となるプロジェクト状態を定量的に評価することができれば，ソフトウェア開発と運用において QCD（Quality，Cost，Delivery）の観点から最適化を図ることが可能となるものと考える．本論文では，ソフトウェア開発工数に基づくジャンプ拡散過程モデルについて議論する．

時刻 $t = 0$ で OSS の運用が開始され，任意の時刻 $t(t \geq 0)$ までの投入開発工数 $\Lambda(t)$ は以下の常微分方程式によって記述されるものと仮定する．

$$\frac{d\Lambda(t)}{dt} = \beta(t)\{\alpha - \Lambda(t)\}. \tag{1}$$

ここで，$\beta(t)$ は時刻 t における開発工数の変化率を，α は OSS の特定バージョンのリリースに必要とされる開発工数を表す．OSS プロジェクトでは，開発者の習熟度のばらつきやネットワーク環境の不安定な状態に影響され，実際の開発工程は不規則な性質を示すと考えられる．よって，リリースまでに必要とされる開発工数の変化率は開発期間を通じて必ずしも一定であるとは限らない．そこで，開発工程のもつそのような確率的性質を，開発工数の変化率 $\beta(t)$ の時間的な不規則変動でモデル化する．このとき，式 (1) は，

$$\frac{d\Lambda(t)}{dt} = \{\beta(t) + \sigma\nu(t)\}\{\alpha - \Lambda(t)\}, \tag{2}$$

となる [4]．σ は定数パラメータである．$\nu(t)$ は解過程の Markov 性を保証するための標準化された Gauss 型白色雑音である．さらに，ログインするユーザの突発的な増減やマイナーバージョンアップのような急激な開発工数の変化などの影響を考慮し，ジャンプ項を導入する [5]．式 (2) を，以下の Itô 型の確率微分方程式 [4,6] に拡張して考える．

$$d\Lambda(t) = \left\{\beta(t) - \frac{1}{2}\sigma^2\right\}\{\alpha - \Lambda(t)\}dt + \sigma\{\alpha - \Lambda(t)\}d\omega(t) + d\left(\sum_{i=1}^{M_t(\tau)}(V_i - 1)\right). \tag{3}$$

ここで，$M_t(\tau)$ は，$\omega(t)$ とは独立な強度パラメータ τ をもつポアソン過程であり，時刻 t までにジャンプが発生した回数を表す．τ はジャンプ事象が生じる確率的な頻度である．また，V_i は i 回目のジャンプ幅を表す独立な確率変数である．式 (3) の確率微分方程式を Itô の公式を用いて変換すると，

$$\Lambda(t) = \alpha\left[1 - \exp\left\{-\int_0^t \beta(s)ds - \sigma\omega(t) - \sum_{i=1}^{M_t(\tau)}\log V_i\right\}\right], \tag{4}$$

となる [7]．本論文では，開発工数の変化率 $\beta(t)$ は，簡単のために既存のソフトウェア信頼度成長モデルにおける平均値関数を流用することにより，次式を満たすものとする．

$$\begin{aligned}
\beta(t) &\doteq \frac{\frac{dF_*(t)}{dt}}{\alpha - F_*(t)}, \\
F_e(t) &= \alpha(1 - e^{-\beta t}), &(5)\\
F_s(t) &= \alpha\left\{1 - (1 + \beta t)e^{-\beta t}\right\}. &(6)
\end{aligned}$$

ここで，β は開発工数の変化率を表す．

したがって，投入開発工数のサンプルパスは，

$$\begin{aligned}
\Lambda_e(t) &= \alpha\left[1 - \exp\left\{-\beta t - \sigma\omega(t) - \sum_{i=1}^{M_t(\tau)}\log V_i\right\}\right], &(7)\\
\Lambda_s(t) &= \alpha\left[1 - (1 + \beta t)\exp\left\{-\beta t - \sigma\omega(t) - \sum_{i=1}^{M_t(\tau)}\log V_i\right\}\right], &(8)
\end{aligned}$$

となる [8]．さらに，複数のジャンプ拡散過程を考慮するために，以下のような一般化ジャンプ拡散過程モデルについて考える．

$$\Lambda_{fe}(t) = \alpha\left[1-\exp\left\{-\beta t-\sigma\omega(t)-\sum_{k=1}^{K}\sum_{i=1}^{M_{t_k}(\tau_k)}\log V_i^k\right\}\right], \tag{9}$$

$$\Lambda_{fs}(t) = \alpha\left[1-(1+\beta t)\exp\left\{-\beta t-\sigma\omega(t)-\sum_{k=1}^{K}\sum_{i=1}^{M_{t_k}(\tau_k)}\log V_i^k\right\}\right]. \tag{10}$$

ここで，t_k $(k=1,2,\cdots,K)$ は k 番目のメジャーバージョンにおける時刻を表す．

3　モデルパラメータの推定

提案モデルに含まれているパラメータ α，β，σ，およびジャンプ項に含まれるパラメータ集合 $\boldsymbol{\theta}$ は一般には既知ではないので，実測データなどの利用可能なデータを使って値を推定しなければならない．本論文における提案モデルは，Wiener 過程とジャンプ拡散過程の 2 種類をもつ混合確率過程である．したがって，ジャンプ拡散過程モデルは混合確率過程のため複雑であり，決定的なパラメータ推定手法が存在せず，モーメント法のような従来手法では観測期間全体からの大域的近似解しか求めることができないという問題があった．本論文では，異なる確率過程とデータ構造の統計的独立性という観点から，最尤推定法と深層学習の 2 種類を併用したパラメータ推定法について議論する．

3.1　Wiener 過程に対するパラメータ推定

まず，確率過程 $\Lambda_*(t)$ の K 次の同時確率分布を

$$P(t_1,y_1;t_2,y_2;\cdots;t_K,y_K)=\Pr[\Lambda_*(t)\le y_1,\Lambda_*(t)\le y_2,\cdots,\Lambda_*(t)\le y_K|\Lambda_*(t)=0], \tag{11}$$

とし，その同時確率密度を

$$p(t_1,y_1;t_2,y_2;\cdots;t_K,y_K)=\frac{\partial^K P(t_1,y_1;t_2,y_2;\cdots;t_K,y_K)}{\partial y_1\partial y_2\cdots\partial y_K}, \tag{12}$$

とする．$\Lambda_*(t)$ は連続値をとるので，データ $(t_k\ ,\ y_k)$ に対し，尤度関数を

$$l=p(t_1,y_1;t_2,y_2;\cdots;t_K,y_K), \tag{13}$$

と表す．さらに，対数尤度関数を L とすると，

$$L=\log l, \tag{14}$$

となり，提案モデルでは，未知パラメータ α，β，および σ を同時尤度方程式

$$\frac{\partial L}{\partial\alpha}=\frac{\partial L}{\partial\beta}=\frac{\partial L}{\partial\sigma}=0, \tag{15}$$

の解として得ることができる．

3.2　ジャンプ拡散過程に対するパラメータ推定

例えば，入力データとしてクラウド稼働状況を示す総合データセットを適用したネットワーク構造を構築する．このとき，採取されたデータに基づく適切なデータ構造制約を与えつつ最適解を模索することで，複数のノイズパラメータのもつ揺らぎ特性の背後にある特徴量を自動で抽出し，不規則な拡散状態の特性を解明することも可能となる．その結果として，ジャンプ拡散項が外的要因として OSS の開発工数へ及ぼす影響や，OSS 開発工数の特性変化を数理的・データ的観点から精細かつ定量的に評価できると考えられる．

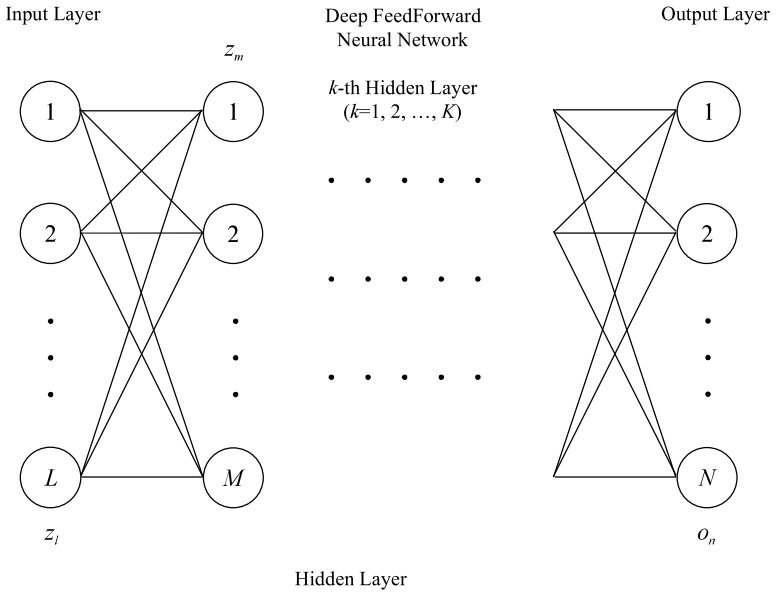

図 1： ディープラーニングの構造.

本論文におけるディープラーニングの構造を図 1 に示す．図 1 において，$z_l(l = 1, 2, \cdots, L)$ および $z_m(m = 1, 2, \cdots, M)$ は特徴抽出のための結合ユニットを意味する．また，$o_n(n = 1, 2, \cdots, N)$ は，圧縮された特徴量を表す．ディープラーニングに関しては，いくつかのアルゴリズムが提案されているが，本論文では，オープンソースプロジェクトにおけるバグトラッキングシステム上におけるフォールトビッグデータを分析するために，簡単のため，典型的なディープフィードフォワードニューラルネットワークを採用する [9–14]．

特徴抽出のための結合ユニットにおけるパラメータに対する情報量として以下のようなデータを適用する．このとき，9 種類の説明変数に対して，ジャンプ項のパラメータ集合 $\boldsymbol{\theta}$ に含まれる各パラメータが目的変数となるように学習され，各入力データは出現頻度のような数値データに変換される．

- バグトラッキングシステム上へ登録された日付
- ソフトウェアプロダクト名
- ソフトウェアコンポーネント名
- ソフトウェアバージョン名およびバージョン番号
- フォールト報告者のニックネーム
- フォールト修正者のニックネーム
- フォールトの状態
- オペレーティングシステム名
- フォールト重要度

本論文におけるディープラーニングの目的変数としては，ジャンプ項のパラメータ集合 $\boldsymbol{\theta}$ に含まれる，それぞれのジャンプパラメータとして定義し，実測データから得られた開発工数データに基づき学習させることで，パラメータ $\boldsymbol{\theta}$ の推定値 $\widehat{\boldsymbol{\theta}}$ を得ることが可能となる．

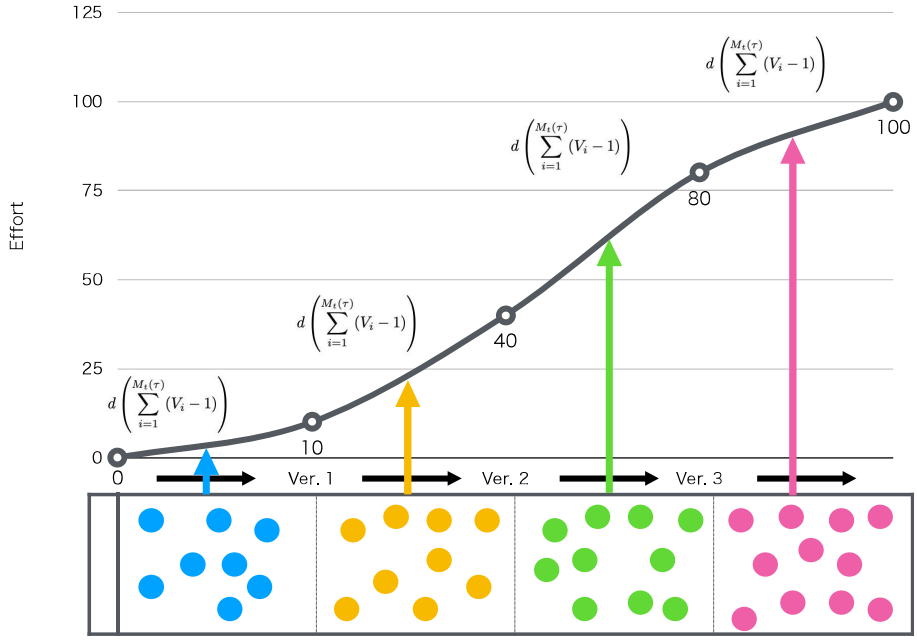

図 2： 深層学習に基づくジャンプ拡散項に含まれるパラメータ推定の概念図.

提案手法により，複数のジャンプ項から構成された一般化ジャンプ拡散過程モデルのパラメータを推定することも可能となる．例えば，図 2 に示すように，短期に分割された期間ごとの局所的最適解を得ることができる．これにより，従来手法では観測期間全体からしか求めることができなかった大域的近似解の精度を高めることも可能となる．

4　数値例

Apache HTTP Server Project [15] におけるデータを利用し，ある特定バージョンにおけるジャンプ拡散項に含まれるパラメータの推定結果のみを使用した場合における数値例を示す．推定された投入開発工数として式 (9) および式 (10) におけるサンプルパスを図 3 および図 4 に示す．図 3 から，指数形ジャンプ拡散過程モデルの場合においては，Wiener 過程およびジャンプ拡散過程におけるノイズと頻度が大きい様子が分かる．このことから，指数形ジャンプ拡散過程モデルでは投入開発工数が悲観的に見積もられていることが確認できる．一方，図 4 においては，ノイズが小さいことから，安定した稼働が期待できる様子が確認できる．特に，Wiener 過程におけるノイズの大きさと，ジャンプ拡散項におけるジャンプの頻度と大きさから得られる情報は，将来的に投入すべき開発工数の予測に役立つと考えられる．

5　おわりに

従来から，数多くのソフトウェア信頼度成長モデルが利用されてきた．また，過去に提案されたソフトウェア信頼度成長モデルを OSS へ適用した研究もいくつか行われている．一般的に，ソフトウェア信頼性はフォールトデータに基づいて評価されるが，フォールト発生の原因となるプロジェクト状態を定量的に評価することができれば，ソフトウェア開発と運用において QCD の観点から最適化を図ることが可能となるものと考える．

本論文では，過去に提案された開発工数を予測するためのジャンプ拡散過程モデルについて，パラメータ推定の観点から議論した．特に，Wiener 過程のパラメータに対して最尤推定法を適用し，ジャンプ拡

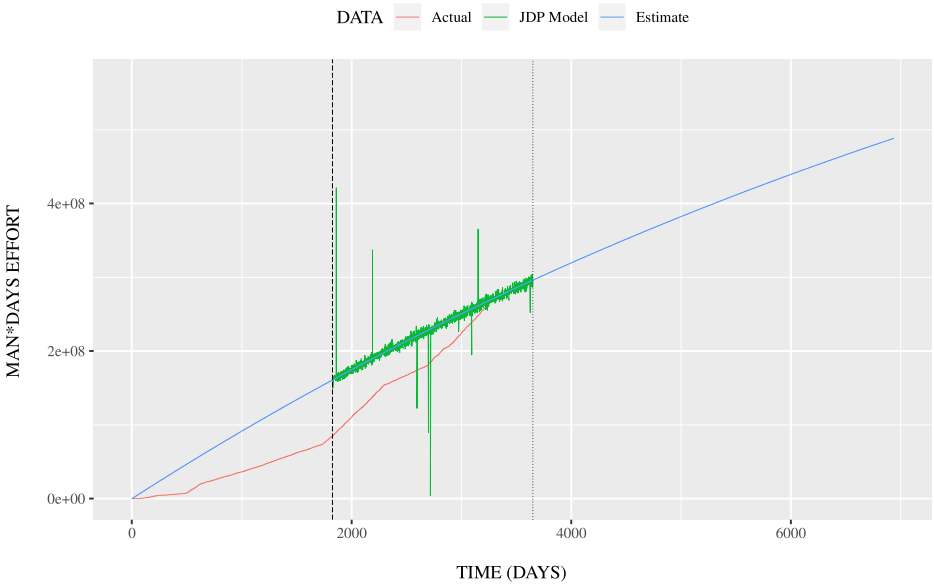

図 3： 指数形ジャンプ拡散過程モデルにおける推定された開発工数.

散過程のパラメータに対して深層学習を用いることで，複合確率過程における複雑性を考慮したパラメータ推定手法を提案した．また，バグトラッキングシステム上に登録されているデータを利用した提案手法に基づく数値例を示した．過去に，フォールトデータに基づく確率微分方程式モデルやジャンプ拡散過程モデルが提案されてきたが [8]，フォールトデータと比較して開発工数データの値の方が非常に大きくなることから，ノイズを考慮することを考えた場合においてモデルの振る舞いにおける具体性が高くなり，より現実的な評価を行うことが可能となる．また，ジャンプ項に含まれるパラメータに対して深層学習を適用することにより，複雑な要因から生じたジャンプ事象に対して，特徴量という観点からジャンプパラメータを推定することが可能となるものと考える．

参考文献

[1] I.M. Ibrahim et al., “A robust generic multi-authority attributes management system for cloud storage services,” *IEEE Trans. Cloud Computing*, 10.1109/TCC.2018.2867871, 30 Aug. 2018.

[2] S. Yamada, *Software Reliability Modeling: Fundamentals and Applications*, Springer–Verlag, Tokyo/Heidelberg, 2014.

[3] P.K. Kapur, H. Pham, A. Gupta, and P.C. Jha, *Software Reliability Assessment with OR Applications*, Springer–Verlag, London, 2011.

[4] L. Arnold, *Stochastic Differential Equations–Theory and Applications*. John Wiley & Sons: New York, 1974.

[5] R.C. Merton, “Option pricing when underlying stock returns are discontinuous,” *Journal of Financial Economics*, Vol. 3, Issues 1–2, pp. 125–144, 1976.

[6] E. Wong, *Stochastic Processes in Information and Systems*. McGraw–Hill: New York, 1971.

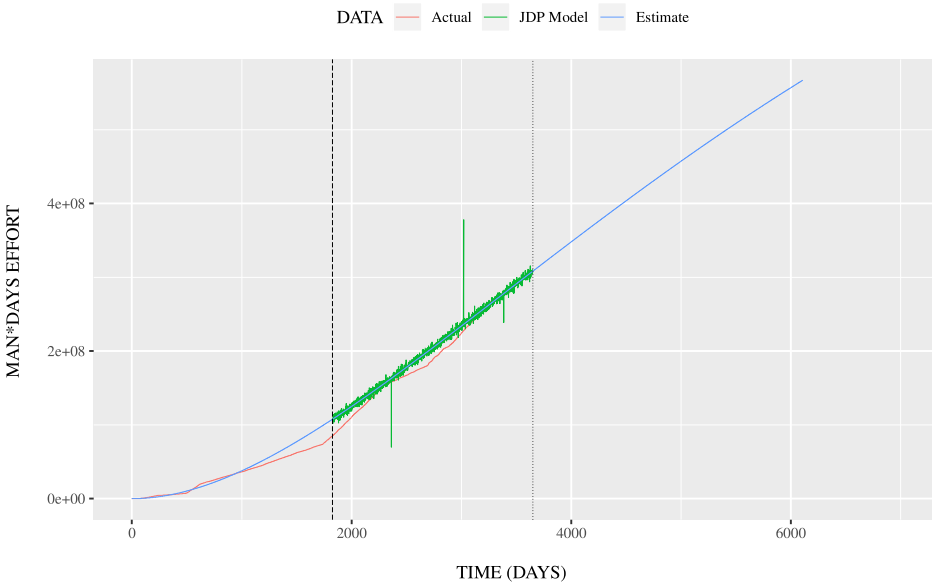

図 4： S 字形ジャンプ拡散過程モデルにおける推定された開発工数.

[7] S. Yamada, M. Kimura, H. Tanaka, and S. Osaki, "Software reliability measurement and assessment with stochastic differential equations," *IEICE Transactions on Fundamentals*, Vol. E77–A, No. 1, pp. 109–116, 1994.

[8] Y. Tamura, T. Takeuchi, and S. Yamada, "Software reliability and cost analysis considering service user for cloud with big data," International Journal of Reliability, Quality and Safety Engineering, Vol. 24, No. 1, World Scientific, pp. 1750009-1–1750009-14, 2017.

[9] D.P. Kingma, D.J. Rezende, S. Mohamed, and M. Welling, "Semi-supervised learning with deep generative models," *Proceedings of Neural Information Processing Systems*, 2014, pp. 3581–3589.

[10] A. Blum, J. Lafferty, M.R. Rwebangira, and R. Reddy, "Semi-supervised learning using randomized mincuts," *Proceedings of the International Conference on Machine Learning*, 2004, pp. 1–13.

[11] E.D. George, Y. Dong, D. Li, and A. Alex, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 20, No. 1, pp.30-- 42, 2012.

[12] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research*, Vol. 11, No. 2, pp. 3371–3408, 2010.

[13] H.P. Martinez, Y. Bengio, and G.N. Yannakakis, "Learning deep physiological models of affect," *IEEE Computational Intelligence Magazine*, Vol. 8, No. 2, pp. 20–33, 2013.

[14] B. Hutchinson, L. Deng, and D. Yu, "Tensor deep stacking networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 35, No. 8, pp. 1944–1957, 2013.

[15] The Apache Software Foundation, The Apache HTTP Server Project, http://httpd.apache.org/