Predição do tráfego de rede de computadores usando redes neurais tradicionais e de aprendizagem profunda

Tiago P. Oliveira [1]
Jamil S. Barbar [1]
Alexsandro S. Soares [1]

Data submissão: 22.07.2014
Data aceitação: 20.03.2015

Resumo: Neste artigo são comparados quatro abordagens diferentes para a previsão do tráfego de redes de computadores, usando o tráfego de dispositivos de redes de computadores que se conectam a Internet e usando Redes Neurais Artificiais (RNA) para a predição, sendo elas: (1) *Multilayer Perceptron* (MLP) com *Backpropagation* para o treinamento; (2) MLP com *Resilient Backpropagation* (Rprop); (3) Rede Neural Recorrente (RNN); (4) *Stacked Autoencoder* (SAE) com aprendizagem profunda (*deep learning*). Também é apresentado que um modelo de rede neural mais simples, tais como a RNN e MLP, podem ser mais eficientes do que modelos mais complexos, como o SAE. A predição do tráfego de Internet é uma tarefa importante para muitas aplicações, tais como aplicações adaptativas, controle de congestionamento, controle de admissão, detecção de anomalias e alocação de largura de banda. Além disso, métodos eficientes de gerenciamento de recursos, como a largura de banda, podem ser usados para melhorar o desempenho e reduzir custos, aprimorando a Qualidade de Serviço (QoS). A popularidade das novas redes neurais profundas vêm aumentado em muitas áreas, porém há uma falta de estudos em relação a predição de séries temporais, como o tráfego de Internet.

[1]Universidade Federal de Uberlândia, UFU, Faculdade de Computação, FACOM
{tiago_prado@comp.ufu.br; jamil@facom.ufu.br; alex@facom.ufu.br}

Abstract: This paper compares four different artificial neural network approaches for computer network traffic forecast, such as: (1) Multilayer Perceptron (MLP) using the Backpropagation as training algorithm; (2) MLP with Resilient Backpropagation (Rprop); (3) Recurrent Neural Network (RNN); (4) deep learning Stacked Autoencoder (SAE). The computer network traffic is sampled from the traffic of the network devices that are connected to the Internet. It is shown herein how a simpler neural network model, such as the RNN and MLP, can work even better than a more complex model, such as the SAE. Internet traffic prediction is an important task for many applications, such as adaptive applications, congestion control, admission control, anomaly detection and bandwidth allocation. In addition, efficient methods of resource management, such as the bandwidth, can be used to gain performance and reduce costs, improving the Quality of Service (QoS). The popularity of the newest deep learning methods have been increasing in several areas, but there is a lack of studies concerning time series prediction, such as Internet traffic.

1 Introdução

Usar observações passadas para prever o tráfego de rede futuro é um passo importante para entender e controlar uma rede de computadores. A predição do tráfego das redes de computadores pode ser crucial para o seu gerenciamento, tais como aplicações adaptativas, controle de congestionamento, controle de admissão e alocação de banda.

Existem muitas pesquisas que focam em aplicações adaptativas e dinâmicas. Nelas são apresentados algoritmos que usam o tráfego para alterar a largura de banda de algum componente de rede [1][2][3], melhorando a Qualidade de Serviço (QoS) [4]. Vários trabalhos foram feitos usando Redes Neurais Artificiais (RNA) e eles mostram que elas são um modelo competitivo, superando métodos clássicos de regressão como o *Autoregressive Integrated Moving Average* (ARIMA) [5][6][7][8]. Além disso, existem trabalhos que combinam esses dois fatores, criando um sistema de predição baseado em rede neural para alocar dinamicamente a largura de banda em fluxos de vídeo em tempo-real [3].

O tráfego de rede é uma série temporal, ou seja, ele é uma sequencia de dados medidos regularmente em intervalos de tempo uniformes. Para o tráfego de rede, estes dados sequenciais são os bits transmitidos em algum dispositivo de rede em um certo período no tempo. Uma série temporal pode ser um processo estocástico ou um processo determinístico. Para prever uma série temporal são necessários modelos matemáticos que realmente representam as características estatísticas do tráfego amostrado. A escolha do método de predição para aplicações adaptativas que necessitam processamento em tempo-real deve levar em consideração o horizonte de predição, custo computacional, erro de predição e o tempo de resposta.

Este artigo analisa quatro métodos de predição que são baseados em RNAs. Avaliações foram feitas comparando *Multilayer Perceptron* (MLP) ou Perceptron de Múltiplas Camadas, Redes Neurais Recorrentes (RNN) e *Stacked Autoencoder* (SAE). MLP é uma rede neural direta (*feed-forward neural network*) com muitas camadas, que usa um treinamento supervisionado. SAE é uma rede neural profunda (*deep learning neural network*) que usa um algoritmo guloso como treinamento não supervisionado. Para a MLP foram comparados dois algoritmos de treinamento diferentes, o *Backpropagation* padrão e o *Resilient Backpropagation* (Rprop).

Esses métodos de predição foram selecionados com o objetivo de confirmar o quão competitivo as abordagens mais simples (RNN e MLP) são, quando comparadas com as mais complexas (SAE e MLP mais profundas). A análise foca em predições de curto prazo e em tempo-real e os testes foram feitos usando amostras de séries temporais de tráfego de Internet, que foram obtidas na base de dados *DataMarket* [9].

2 Redes Neurais Artificiais

Redes neurais artificiais são estruturas de processamento simples, que são separadas em unidades fortemente conectadas chamadas de neurônios. Os neurônios são organizados em camadas, uma camada pode ter vários neurônios e uma rede neural pode ter uma ou mais camadas [10]. Os neurônios são capazes de trabalhar em paralelo para processar os dados, armazenar conhecimento e usar esse aprendizado para inferir novos dados. Cada neurônio tem um peso sináptico, que é responsável para armazenar o conhecimento adquirido. O conhecimento da rede é adquirido através do processo de aprendizagem, também conhecido como algoritmo de aprendizagem ou algoritmo de treinamento [10].

Durante o processo de aprendizagem da RNA os pesos sinápticos são modificados de forma ordenada até que ela reconheça e diferencie os dados de um conjunto finito. Após o aprendizado, a RNA está pronta para reconhecer os padrões da série temporal, por exemplo. As redes neurais oferecem as mesmas funcionalidades dos neurônios do cérebro humano para resolver problemas complexos, tais como a não linearidade, alto paralelismo, robustez, tolerância a erros e ruídos, adaptabilidade, aprendizado e generalização [5][10].

Historicamente, o uso das RNAs era limitado em relação ao número de camadas ocultas (ou escondidas), devido a dificuldade de treinar redes neurais com muitas camadas e neurônios [11]. Algoritmos de treinamento convencionais, como o *Backpropagation*, não tem um desempenho bom em RNAs profundas, com mais de três camadas ocultas [12]. Entretanto, surgiram as *Deep Belief Networks* (DBN), que são redes neurais profundas nas quais possuem um método de treinamento baseado em um algoritmo guloso, no qual treina uma camada por vez [13]. Desde então pesquisas encontraram vários resultados bons em relação ao uso de redes neurais profundas para dados não rotulados e treinamento não supervisio-

nado; com tempo de treinamento mais rápidos e erros menores do que as redes neurais mais comuns, como a MLP.

Redes de aprendizagem profunda referem-se a um método de aprendizado de máquina baseado em modelos de redes neurais com vários níveis de representação de dados, permitindo aprender dados mais complexos. Os níveis hierárquicos de representação de dados são organizados por abstrações, características e conceitos. Níveis mais altos são definidos pelos níveis mais baixos, onde a representação dos níveis mais baixos podem definir várias características diferentes dos níveis mais altos. Quanto mais alto o nível, mais abstrato e não linear será a representação dos dados [11][13]. Esses níveis hierárquicos são representados pelas camadas da RNA. Desse modo, a profundidade da rede neural se diz respeito ao número de níveis de composição das operações não lineares perante os dados treinados, i.e., quanto mais camadas, mais não linear e profunda será a RNA.

3 Revisão da Literatura

Vários tipos de RNAs foram estudadas para a predição de tráfego. Existem várias pesquisas com redes neurais diretas, tais como a MLP [5][7]. Porém, muitos estudos visam as RNNs [6], pois elas possuem um ciclo interno de memória que facilita o aprendizado de comportamentos dinâmicos, temporais e sequenciais. Consequentemente, a RNN é um bom modelo para o aprendizado de séries temporais.

Uma vantagem de qualquer tipo de RNA é no tempo de resposta, isto é, o quão rápido é feito a predição dos valores futuros. O processo de aprendizagem é a etapa mais lenta no uso das RNAs. Contudo, após a aprendizagem a rede neural está pronta para ser usada, calculando os resultados de modo muito rápido, comparado a outros modelos de predição, como o ARIMA [5][8]. Portanto, as RNAs são boas para predição em tempo-real, conseguindo um resultado satisfatório em relação a precisão da predição e em um pequeno tempo de resposta [5].

3.1 *Multilayer Perceptron* e *Backpropagation*

Uma das arquiteturas mais comuns para as RNAs é a MLP. Esse tipo de rede neural possui uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída. Boas práticas de uso sugerem somente uma ou duas camadas ocultas [14]. Isso se dá pelo fato de que os mesmos resultados podem ser obtidos aumentando o número de neurônios na camada escondida, ao invés de aumentar o número de camadas escondidas [15].

As redes MLPs são redes neurais diretas, nas quais todos os neurônios de uma camada são conectados com todos os neurônios da camada seguinte, mas não há conexão entre os neurônios da mesma camada. Esse tipo de rede neural é chamada de direta (*feed-forward*)

por causa do fluxo de informação que vai da camada de entrada até a camada de saída. O algoritmo de treinamento mais comum da MLP é o *Backpropagation*, que é um algoritmo de treinamento supervisionado, no qual a MLP aprende a saída desejada através de vários dados de entrada.

Geralmente, quando se usa o algoritmo de treinamento *Backpropagation*, ocorre um problema na magnitude da derivativa parcial, tornando-se muito alta ou muito pequena. Isso é um problema onde o processo de aprendizagem sofre muitas flutuações, deixando a convergência mais lenta ou inexistente, fazendo com que a rede fique presa em um mínimo local. Para ajudar a evitar esse problema, foi criado o algoritmo *Resilient Backpropagation*, que tem uma taxa de aprendizagem dinâmica. Nele, a taxa de aprendizagem é atualizada para cada conexão dos neurônios, para diminuir o erro de forma independente para cada neurônio.

3.2 Redes Neurais Recorrentes

Redes neurais recorrentes são RNAs que possuem uma ou mais conexões entre neurônios que formam um ciclo. Os ciclos são responsáveis de armazenar e transmitir as experiências de um neurônio para outro, criando uma memória interna que facilita o aprendizado de dados sequenciais [6][10]. Os ciclos podem ser usados em qualquer lugar da rede neural e em qualquer direção, e.g., ligando a camada de saída com a de entrada, ligando uma camada oculta com outra camada qualquer, ou qualquer combinação de ligações cíclicas [10].

Um dos objetivos deste trabalho é comparar modelos de redes neurais mais simples com os mais complexos. Para esse propósito foi escolhido a Rede Neural de Jordan (JNN), já que é uma rede neural recorrente simples. A JNN tem uma camada de contexto que guarda a saída anterior da camada de saída, esta camada de contexto é responsável por receber as saídas (*feedback*) das iterações anteriores e passá-las para a camada escondida; permitindo uma memória de curto prazo simplificada [6].

3.3 *Stacked Autoencoder* para aprendizagem profunda

O *Stacked Autoencoder* é um tipo de rede neural profunda que é construída a partir de camadas de *Autoencoders* esparsos, no qual a saída de cada camada é conectada com a entrada da camada seguinte [16]. O SAE usa todos os benefícios de uma rede neural profunda e possui um alto poder de classificação. Por causa disso, ele pode aprender dados úteis sobre agrupamento hierárquico e decomposição da entrada em relação parte-todo [17]. A principal ideia das redes neurais profundas está no fato de ter múltiplas camadas representando vários níveis de abstração dos dados. Consequentemente, redes neurais mais profundas possuem um poder de aprendizagem superior em comparação com redes mais rasas [11], ou seja, quanto maior a profundidade da rede neural maior será a capacidade de aprendizagem de dados mais complexos e abstratos.

A força do aprendizado em profundidade está na aprendizagem das representações dos dados através do algoritmo guloso de treinamento não supervisionado. Assim, após ser encontrado uma boa representação dos dados em cada camada, o SAE obtido pode ser usado para inicializar uma nova RNA com novos pesos sinápticos. Esta nova rede neural inicializada pode ser uma MLP, por exemplo, dando início a um treinamento supervisionado, caso necessário [11]. Muitos artigos dão ênfase nos benefícios do treinamento não supervisionado e guloso usado para a inicialização de uma nova rede neural [11][13][17][18][19]. Então, um dos objetivos deste artigo é verificar se esse treinamento não supervisionado, que pré-treina a rede neural profunda, realmente traz vantagens sobre modelos de RNA mais simples.

4 Experimentos e Resultados

As séries temporais utilizadas foram obtidas no *DataMarket* e coletados por R. J. Hyndman [9]. Os experimentos foram realizados a partir de dados coletados diariamente, a cada hora e a cada intervalo de cinco minutos. Ao todo seis séries temporais foram usadas, sendo elas: A-1d; A-1h; A-5m; B-1d; B-1h; B-5m.

As séries temporais utilizadas são compostas pelo tráfego de Internet (em bits) de um Provedor de Serviços de Internet (ISP) com centros em 11 cidades Europeias. Os dados correspondem a um enlace transatlântico e foram coletados a partir das 06:57 horas no dia 7 de Junho de 2005 até as 11:17 horas em 31 de Julho de 2005. Esta série foi coletada em intervalos diferentes, resultando em três séries temporais diferentes: A-1d é uma série temporal com dados diários; A-1h são dados coletados a cada hora; e A-5m contém dados coletados a cada cinco minutos. As séries temporais restantes são compostas pelo tráfego de Internet de um ISP, coletado em uma rede acadêmica no Reino Unido. Os dados foram coletados entre 19 de Novembro de 2004, as 09:30 horas, até dia 27 de Janeiro de 2005, as 11:11 horas. Da mesma maneira, a série foi dividida em três outras diferentes: B-1d são dados diários; B-1h são dados coletados de hora em hora; e B-5m, com dados coletados de cinco em cinco minutos.

Os experimentos foram conduzidos usando o *DeepLearn Toolbox* [20] e *Encog Machine Learning Framework* [21]. Ambos são de código aberto e possuem bibliotecas que abrangem várias técnicas de aprendizado de máquina e inteligência artificial. Eles foram escolhidos pois, são muito usados na comunidade cientifica, são de código aberto, possuem uma boa usabilidade e suprem nossas necessidades, tais como MLP, RNN e SAE.

DeepLearn Toolbox é um conjunto de bibliotecas escritas em MATLAB que cobrem várias técnicas tais como RNA, DBN, SAE, *Convolutional Autoencoders* (CAE) e *Convolutional Neural Networks* (CNN). Encog é um *framework* que suporta vários algoritmos de aprendizado de máquina, como redes Bayesianas, modelo oculto de Markov, RNA e algoritmos evolutivos, como programação genética e algoritmos genéticos. Para o Encog, foi usado

a versão Java 3.2.0, mas ele também está disponível para .Net, C e C++.

Para a predição e treinamento da *Multilayer Perceptron* com *Backpropagation* (MLP-BP), da MLP com *Resilient Backpropagation* (MLP-RP) e da JNN foi usado o Encog; já para o SAE foi usado o *DeepLearn Toolbox* para MATLAB. As configurações do computador onde os experimentos foram realizados são: Notebook Dell, Vostro 3550; Processador Intel Core i5-2430M, 2.40GHz de velocidade do *clock* e 3Mb de memória Cache); 6GB de memória RAM DDR3; Sistema Operacional Windows 7 Home Basic 64-Bit. A versão da Plataforma Java instalada é a *Enterprise Edition* com o JDK 7 e a versão do MATLAB é a R2013b.

4.1 Normalização dos Dados

Antes de treinar a rede neural é importante normalizar os dados [8], neste caso, a série temporal. Para diminuir a escala dos dados foi usado a normalização entre mínimo e máximo, que limita os dados no intervalo $[0.1, 1]$. Este intervalo foi escolhido para que os valores treinados não saiam do intervalo $[0, 1]$, que é o intervalo obtido na saída da função sigmoide usada na rede neural. O valor mínimo escolhido foi 0.1, ao invés de 0, para evitar a divisão por zero em algum cálculo com os valores normalizados.

A série temporal original é então normalizada, gerando uma nova série temporal que será usada para o treinamento. De todas as 6 séries temporais usadas, seus tamanhos variam entre 51 valores (para a série menor, com dados diários) e 19888 valores (para a série temporal maior, com dados coletados a cada cinco minutos). Para os experimentos, o intervalo para o conjunto de treinamento variou bastante, entre 25 valores (para a série temporal menor) e 9944 valores (para a série temporal maior). O tamanho do conjunto de entrada foi escolhido como a primeira metade da série temporal completa, a outra metade da série temporal é usada para o conjunto de teste, que será usada para avaliar a precisão da predição. O tamanho de cada série pode ser visto na Tabela 1.

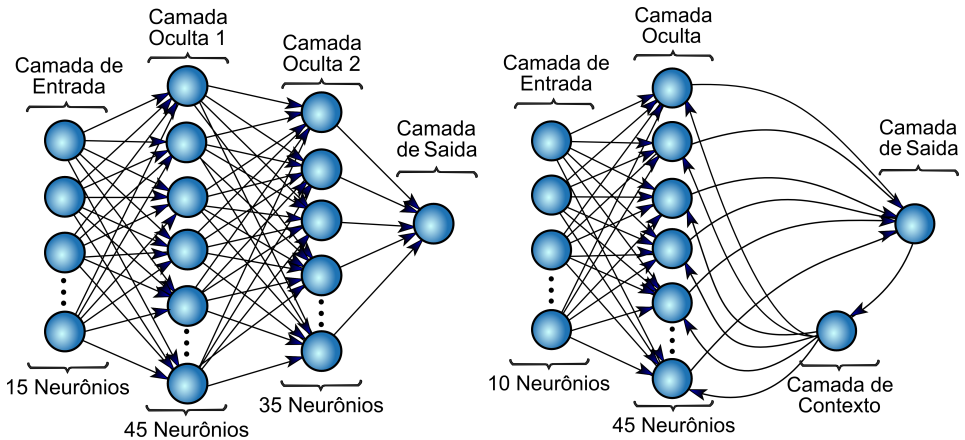
Tabela 1: O intervalo de tempo e o tamanho de cada série temporal

Série Temporal	Intervalo de tempo	Tamanho total da série temporal	Tamanho do conjunto de treinamento
A-1d	1 dia	51	25
A-1h	1 h	1231	615
A-5m	5 min	14772	7386
B-1d	1 dia	69	34
B-1h	1 h	1657	828
B-5m	5 min	19888	9944

4.2 Arquitetura e Topologia das RNAs Usadas

Para a MLP-BP foi usado a função de ativação sigmoide, taxa de aprendizagem igual a 0.01 e o *Backpropagation* como algoritmo de treinamento. Taxas de aprendizagem mais altas aceleram o treinamento, entretanto podem gerar muitas oscilações nele, dificultando a obtenção de um erro mais baixo. Por outro lado, uma taxa de aprendizagem mais baixa leva a um treinamento mais estável, porém é muito mais lento. Também foram testados valores diferentes para a taxa de aprendizagem, como 0.5, 0.25 e 0.1; porém, como já era esperado, os erros mais baixos foram obtidos usando 0.01 na taxa de aprendizagem.

Para a rede neural profunda foi usado o SAE, com função de ativação sigmoide e taxa de aprendizagem igual a 0.01. Para o MLP-RP e a JNN, também foi usado a função de ativação sigmoide e o *Resilient Backpropagation* como algoritmo de treinamento.



(a) Arquitetura da *Multilayer Perceptron* (MLP). (b) Arquitetura da Rede Neural de Jordan (JNN).

Figura 1: Arquitetura da MLP e da JNN mostrando as camadas e o número de neurônios para cada camada. A MLP possui um fluxo direto da informação transmitida e a JNN possui a camada de contexto com um ciclo de *feedback* da camada de saída para a camada oculta.

O algoritmo de treinamento do SAE é baseado em um treinamento não supervisionado, camada por camada, através de um algoritmo guloso. Cada camada é treinada de forma independente. Desse modo, os *Autoencoders* treinados são empilhados, e consequentemente, produzindo uma rede neural profunda pré-treinada (inicializada) [16][17]. O algoritmo guloso do treinamento é suficiente para treinar dados não rotulados, i.e., ele não treina os dados considerando a saída esperada. Por outro lado, para dados rotulados, tais como as séries temporais, o treinamento guloso não é suficiente e é usado como um pré-treinamento

não supervisionado (ao invés da inicialização aleatória dos neurônios). Depois disso, a etapa de refinamento (*fine-tuning*) é iniciada e foi usado o *Backpropagation* como algoritmo de treinamento supervisionado [11][12][17].

Vários testes foram feitos variando a topologia da RNA, foram feitas variações tanto em número de neurônios por camada quanto em número de camadas. Para a MLP-BP e a MLP-RP, os melhores desempenhos foram obtidos com 4 camadas, por volta de 15 neurônios de entrada, um neurônio na camada de saída, 45 e 35 neurônios nas camadas ocultas, respectivamente, como mostra a Figura 1a. Verificou-se que aumentar o número de neurônios ou o número de camadas não resulta em um desempenho melhor, até certo ponto a média do *Normalized Root Mean Square Error* (NRMSE) foi parecido para as mesmas séries temporais.

A MLP-BP, MLP-RP e JNN obtiveram as médias do NRMSE mais parecidas, mas a JNN obteve uma média de erros mais baixa, mesmo usando uma quantidade bem menor de neurônios. Os melhores desempenhos da JNN foram obtidos com três camadas (não contando com a camada de contexto da JNN), por volta de 10 neurônios na entrada, um neurônio na saída e 45 neurônios na camada oculta, como mostra a Figura 1b. Da mesma forma, aumentar o número de neurônios não resultou em um desempenho melhor, na verdade, aumentar muito o número de neurônios foi prejudicial ao desempenho. Os resultados começaram a piorar a partir de 40 neurônios na camada de entrada e 120 neurônios no total.

Para o SAE, os melhores resultados foram encontrados com 6 camadas, com 20 neurônios na camada de entrada, 1 neurônio na camada de saída, 80, 60, 60 e 40 neurônios em cada uma das camadas ocultas, respectivamente. Aumentar o número de neurônios do SAE não produziu resultados melhores; na média o NRMSE foi semelhante. Resultados similares também foram encontrados com 4 camadas, assim como a MLP; entretanto, SAEs mais profundas atingiram resultados um pouco melhores. Uma comparação dos NRMSEs de cada modelo de predição é apresentada na Figura 7.

A influência do número de neurônios no erro é mais clara na Figura 2. Nota-se que o Erro Quadrático Médio (MSE) aumentou significativamente para redes neurais com mais de 120 neurônios. Para um número menor de neurônios, menor que 120, o erro não mudou muito. Além disso, quanto mais neurônios na RNA, mais difícil e demorado será o treinamento. Para o SAE, o MSE foi mais estável, mesmo para arquiteturas mais profundas.

4.3 Treinamento das Redes Neurais

Os treinamentos para cada rede neural foram feitos em um único lote. Assim, todos os dados de entrada do conjunto de treinamento são treinados em um único período de treinamento, que ajusta os pesos da rede neural de acordo com o lote inteiro. Também foram conduzidos testes com mais lotes (menos dados de entrada para cada período de treinamento),

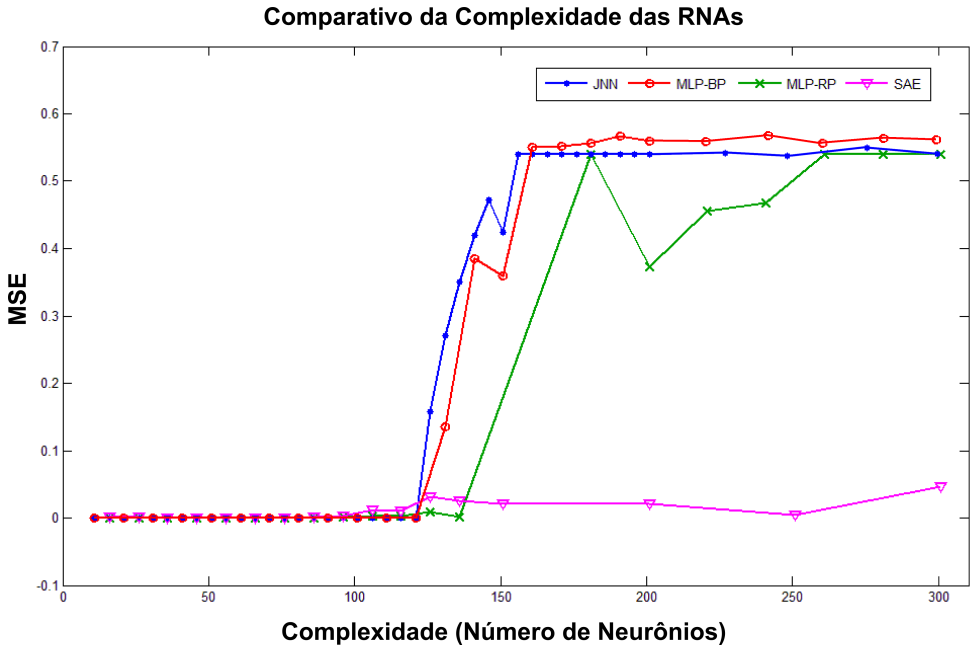


Figura 2: Proporção entre a complexidade, medida em número total de neurônios, da rede neural juntamente com os respectivos erros quadráticos médios (MSE) para a predição da série temporal B-5m.

porém foram obtidas taxas de erros similares. De qualquer modo, para lotes menores, o treinamento dura mais tempo para convergir, pois produz uma quantidade maior de lotes, fazendo com que uma quantidade menor de dados sejam treinados em cada período de treinamento.

O treinamento das redes MLP-BP, MLP-RP e JNN duraram 1000 períodos (ou épocas). O treinamento do SAE é separado em duas etapas. A primeira delas é o pré-treinamento não supervisionado, que durou 900 épocas de treinamento. A segunda etapa é o refinamento, que usa um treinamento supervisionado e que durou 100 períodos de treinamento.

O tempo de treinamento é afetado principalmente pelo tamanho do conjunto de dados que serão treinados e pelo número de neurônios da RNA. Quanto maior o tamanho do conjunto de treinamento e maior o número de neurônios, maior será o tempo necessário para treinar a rede neural. A Tabela 2 mostra o tempo de treinamento médio obtido para cada série temporal e o modelo de predição.

A única diferença entre o MLP-BP e MLP-RP é o algoritmo de treinamento. A taxa

Tabela 2: Comparação da média do tempo de treinamento

Série Temporal	Tempo de treinamento (milisegundos)			
	MLP-BP	MLP-RP	JNN	SAE
A-1d	268	79	67	8874
A-1h	2373	1482	1473	585761
A-5m	86296	56078	33981	6724641
B-1d	558	326	108	17280
B-1h	7322	7181	2838	837567
B-5m	117652	78078	45968	8691876

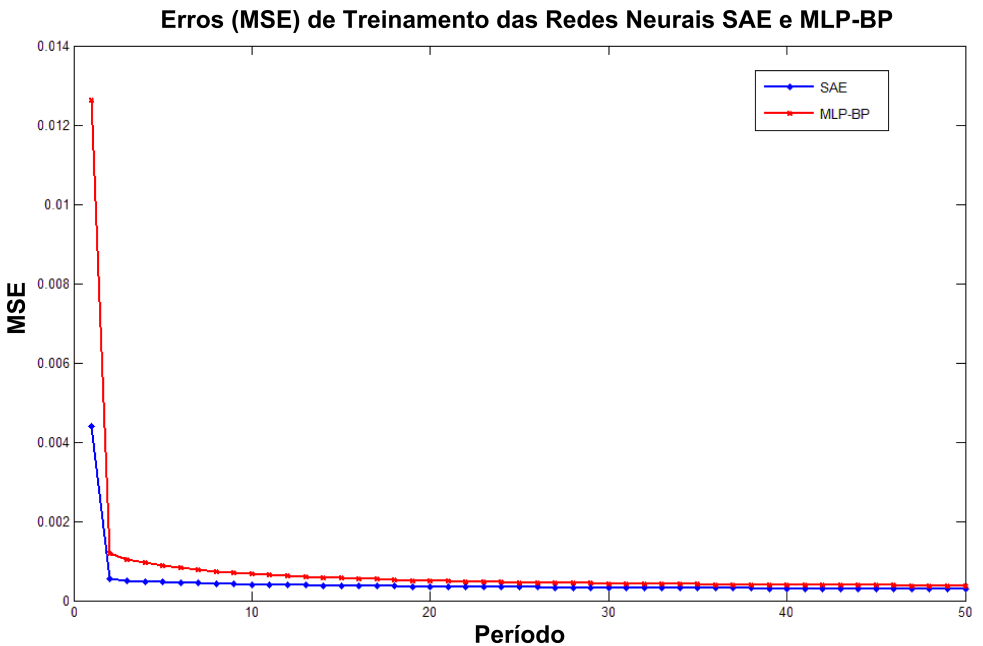


Figura 3: Comparação do MSE obtido pelo SAE e pelo MLP-BP, durante os 50 primeiros períodos de treinamento, para a série temporal B-5m.

de aprendizagem dinâmica do Rprop faz com que ele seja um dos algoritmos de treinamento mais rápidos para uma RNA e isso ficou claro nos resultados, o MLP-RP foi mais rápido do que o MLP-BP. A JNN com Rprop foi a rede neural que usou menos neurônios e camadas,

que foram 3 no total, ao invés de 4 camadas como os outros métodos. Desse modo, de todas as RNA utilizadas, o treinamento mais rápido foi da JNN, que usou menos neurônios e usa um algoritmo de treinamento mais rápido.

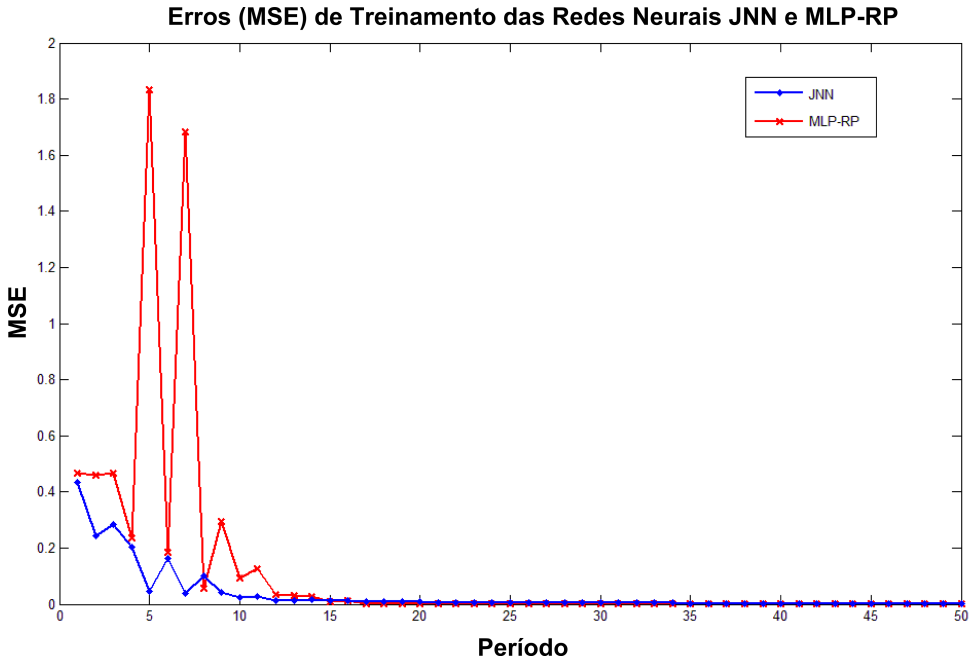


Figura 4: Comparação do MSE obtido pelo MLP-RP e pela JNN, durante os 50 primeiros períodos de treinamento, para a série temporal B-5m.

As 50 primeiras épocas de treinamento e seus respectivos erros são mostrados na Figura 3. Nela é comparado o treinamento supervisionado do SAE com o treinamento da MLP-BP. Observa-se que devido ao pré-treinamento do SAE, o seu refinamento converge mais rapidamente do que o treinamento da MLP. Entretanto, mais períodos de treinamento são o suficiente para que elas alcancem taxas de erros mais similares.

A Figura 4 mostra as 50 primeiras épocas de treinamento e seus respectivos erros para a MLP-RP e JNN, como ambas usam o Rprop como algoritmo de treinamento, os erros mudam drasticamente no início do treinamento, uma vez que não há uma taxa de aprendizagem fixa. Por isso, os erros continuam variando até que um valor melhor para a atualização dos pesos seja encontrado.

As Figura 5 e Figura 6 mostram os resultados das predições da MLP-RP e SAE, res-

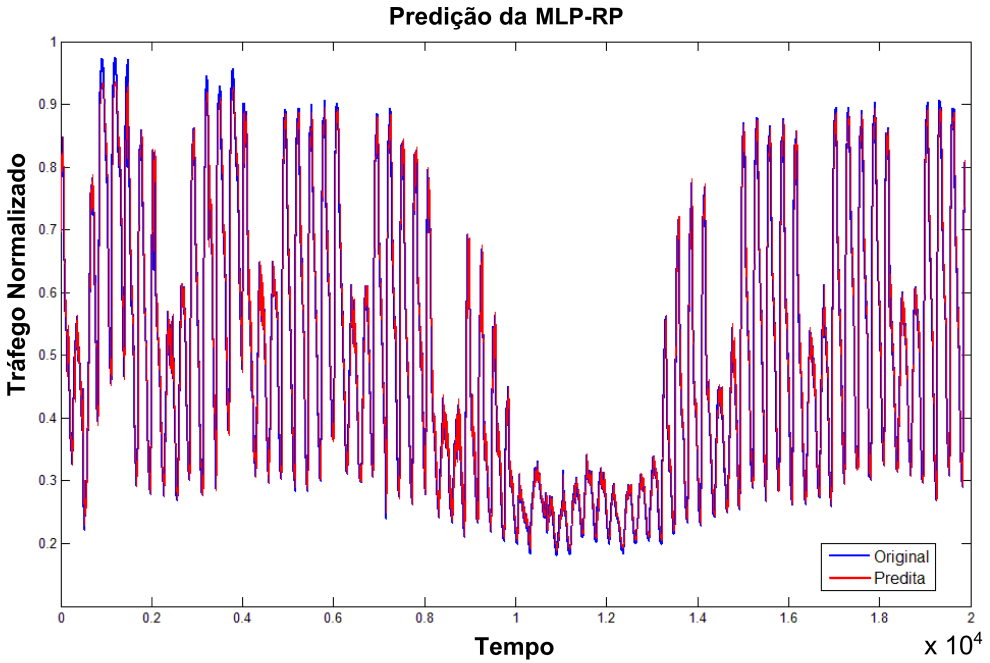


Figura 5: Comparação da série temporal Original (representada na cor cinza) e a série temporal Predita (representada na cor preta) usando o MLP-RP com duas camadas ocultas, para a série B-5m. Como o gráfico da série original e predita são muito similares e a escala do gráfico é muito baixa, torna-se difícil observar graficamente as diferenças de cada um. Contudo, nota-se que os valores preditos se encaixam bem com os valores originais.

pectivamente, para a série temporal B-5m. O gráfico para a MLP-BP e JNN foram muito similares com a MLP-RP mostrado na Figura 5, por isso, eles não serão apresentados. Devido a baixa escala da imagem é difícil de ver uma diferença entre elas, por isso somente é mostrado a MLP-RP. Nota-se que as MLP-RP, MLP-BP e JNN (veja Figura 5) se ajustou melhor aos dados reais, apesar disso, o SAE também teve bons resultados na generalização dos dados, i.e., na aprendizagem de dados que não foram treinados.

Todos os modelos de predição utilizados aprenderam as características das séries temporais e usaram essas características para prever os dados que, a priori, não são conhecidos. Um detalhe importante é que a JNN usou menos neurônios e somente uma camada oculta. Portanto, a fase de treinamento da JNN é mais rápida do que da MLP-BP, MLP-RP e SAE.

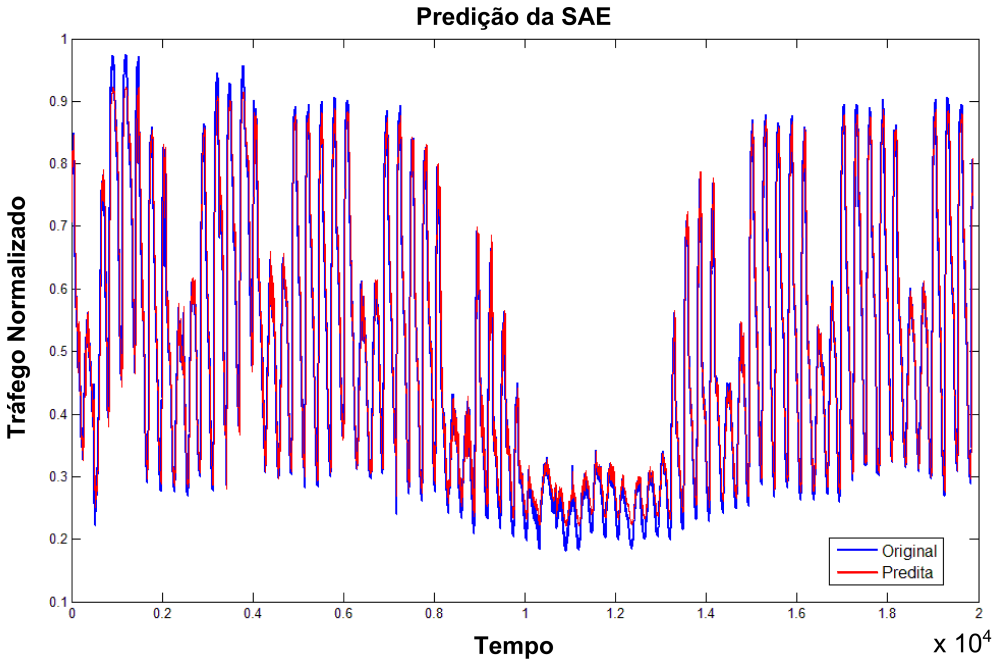


Figura 6: Comparação da predição do SAE com 4 camadas ocultas, para a série temporal B-5m, apresentando a série temporal Original (na cor cinza) e a série Predita (na cor preta). Nota-se que os valores preditos não se encaixam a partir de 1×10^4 até 1.4×10^4 na linha do tempo. Porém, para o resto da série, os valores preditos se encaixam bem nos dados originais.

4.4 Resultados Principais

A ideia principal da aprendizagem profunda é que a profundidade da RNA permite aprender dados complexos e não lineares [11]. Contudo, o uso do SAE para a predição de séries temporais não foi benéfico, i.e., o pré-treinamento não trouxe benefícios significantes para a predição. Os resultados com os respectivos erros normalizados NRMSE são mostrados na Figura 7. O desempenho do SAE foi superado pelos outros métodos.

Na predição de séries temporais, que usa dados rotulados, o método SAE possui mais complexidade do que os outros usados, já que ele possui uma etapa extra com o treinamento não supervisionado, que inicializa os pesos da rede neural para a etapa de refinamento. Mesmo com a complexidade adicional, o SAE foi inferior. Por causa disso, a abordagem com o SAE não é recomendada para a predição de séries temporais.

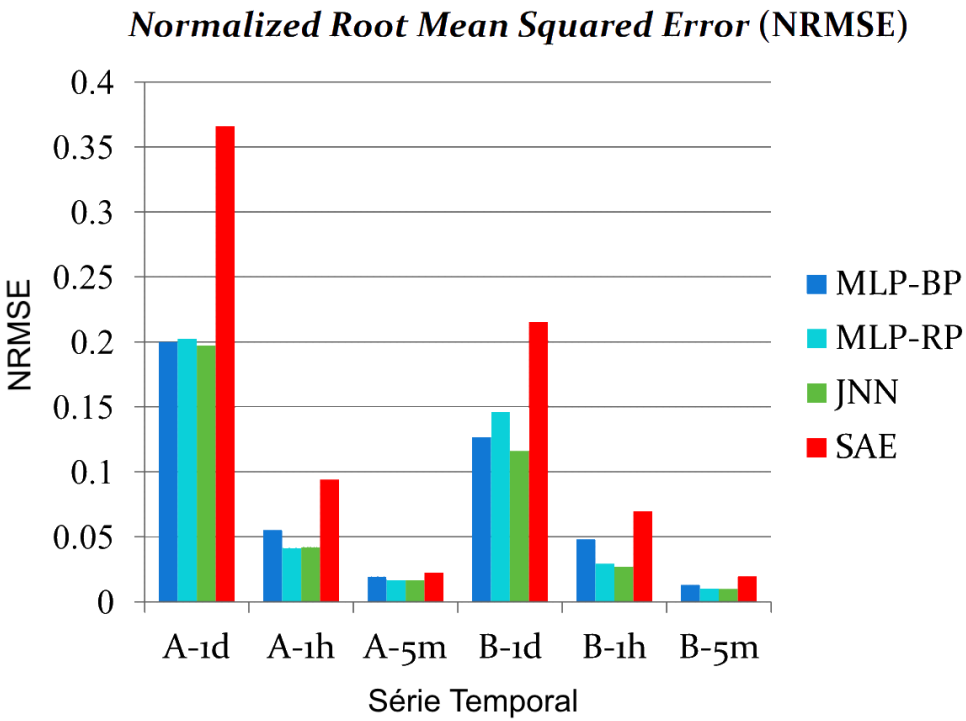


Figura 7: Comparação dos erros normalizados NRMSE obtidos de cada RNA para cada série temporal.

Entre a MLP-BP e a MLP-RP não houve diferenças significativas, mas no geral a MLP-RP obteve resultados melhores, tanto em precisão quanto em tempo de treinamento. Por fim, comparando todos os resultados mostrados na Figura 7, nota-se que a JNN teve os melhores resultados com os menores NRMSE. Além disso, a JNN obteve predições melhores mesmo utilizando menos neurônios. Assim, quanto menos neurônios na rede neural menor será sua complexidade, fazendo com que o treinamento seja muito mais rápido, como apresentado na Tabela 2.

Existem trabalhos, no campo de reconhecimento de padrões, que mostram que o uso de *Autoencoders* é vantajoso [19] porque são baseados em dados não rotulados. Por outro lado, existem trabalhos na previsão de uso de energia, mostrando que os *Autoencoders* são piores que RNAs mais clássicas [22], como a MLP e RNN. Assim, reforçando o fato de que problemas diferentes requerem soluções ou métodos diferentes para resolvê-los.

5 Conclusão

Todas as RNAs analisadas foram capazes de se ajustar e predizer o tráfego de rede com uma boa precisão. Entretanto, a inicialização dos pesos da rede neural através do pré-treinamento não supervisionado não refletiu em melhorias no caso de predição de séries temporais. Os resultados nos levam a crer que as MLPs e as RNNs são melhores que o SAE para a predição de tráfego de Internet. Além disso, a aprendizagem profunda do SAE traz mais complexidade computacional durante o treinamento, então a escolha da MLP ou RNN é mais vantajosa.

Teoricamente, de todas as RNA estudadas neste artigo, o melhor método de predição seria a RNN, uma vez que é a rede que usa observações passadas como *feedback* no aprendizado de dados temporais e sequenciais. Conformemente, os experimentos conduzidos mostraram que os melhores resultados, tanto em precisão quanto em tempo de treinamento, foram obtidos com a JNN, uma rede neural recorrente simples. Portanto, de todos os métodos usados, o melhor método de predição para previsão em tempo-real ou a curto prazo é a RNN com o Rprop como algoritmo de treinamento, visto que ela obteve os menores erros em um tempo significativamente menor, como apresentado na Figura 7 e Tabela 2.

O uso e a importância de redes neurais com aprendizagem profunda está aumentando e resultados cada vez melhores são obtidos no reconhecimento de padrões em imagens, áudio e vídeo [18][19][23][24]. No entanto, os principais algoritmos de treinamento para essas redes são algoritmos de treinamento não supervisionados, os quais usam dados não rotulados para o treinamento. Em contraste, séries temporais e tráfego de rede são dados rotulados, por isso é usado o treinamento não supervisionado como pré-treinamento e depois é aplicado o treinamento supervisionado como refinamento. Ainda assim, como mostra em [24], a DBN e as *Restricted Boltzmann Machines* (RBM), que são métodos de aprendizagem profunda, podem ser modificados para trabalharem melhor com dados rotulados.

Em trabalhos futuros utilizaremos outras técnicas de aprendizagem profunda, como DBNs e *Continuous Restricted Boltzmann Machine* (CRBM). Além disso pretendemos criar uma ferramenta de gerenciamento de largura de banda. Esta ferramenta de gerenciamento poderá ser útil no controle de congestionamento através da alocação dinâmica da largura de banda, baseando-se no tráfego de rede previsto. O objetivo é garantir uma melhor QoS e uma alocação mais justa da largura de banda nos dispositivos de rede, tudo de forma dinâmica e adaptativa.

Referências

[1] M.-S. Han, "Dynamic bandwidth allocation with high utilization for xg-pon," em *Advanced Communication Technology (ICACT), 2014 16th International Conference on*,

Feb 2014, pp. 994–997.

[2] H. Zhao, W. Niu, Y. Qin, S. Ci, H. Tang, e T. Lin, "Traffic load-based dynamic bandwidth allocation for balancing the packet loss in diffserv network," em *Computer and Information Science (ICIS), 2012 IEEE/ACIS 11th International Conference on*, May 2012, pp. 99–104.

[3] Y. Liang e M. Han, "Dynamic bandwidth allocation based on online traffic prediction for real-time mpeg-4 video streams," *EURASIP J. Appl. Signal Process.*, vol. 2007, no. 1, pp. 51–51, Jan. 2007. [Online]. Disponível em: http://dx.doi.org/10.1155/2007/87136

[4] T. Nguyen, T. Eido, e T. Atmaca, "An enhanced qos-enabled dynamic bandwidth allocation mechanism for ethernet pon," em *Emerging Network Intelligence, 2009 First International Conference on*, Oct 2009, pp. 135–140.

[5] P. Cortez, M. Rio, M. Rocha, e P. Sousa, "Multi-scale internet traffic forecasting using neural networks and time series methods," *Expert Systems*, vol. 29, no. 2, pp. 143–155, 2012. [Online]. Disponível em: http://dx.doi.org/10.1111/j.1468-0394.2010.00568.x

[6] M. Hallas e G. Dorffner, "A comparative study on feedforward and recurrent neural networks in time series prediction using gradient descent learning," em *Proc. of 14th European Meeting on Cybernetics and Systems Research*, 1998, pp. 644–647.

[7] X. Ding, S. Canu, T. Denoeux, T. Rue, e F. Pernant, "Neural network based models for forecasting," em *Proceedings of ADT'95*. Wiley and Sons, 1995, pp. 243–252.

[8] H. Feng e Y. Shu, "Study on network traffic prediction techniques," em *Wireless Communications, Networking and Mobile Computing, 2005. Proceedings. 2005 International Conference on*, vol. 2, Sept 2005, pp. 1041–1044.

[9] R. Hyndman, "Time series data library." [Online]. Disponível em: http://data.is/TSDLdemo

[10] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1998.

[11] Y. Bengio, "Learning deep architectures for AI," *Found. Trends Mach. Learn.*, vol. 2, no. 1, pp. 1–127, Jan. 2009. [Online]. Disponível em: http://dx.doi.org/10.1561/2200000006

[12] D. Erhan, P.-A. Manzagol, Y. Bengio, S. Bengio, e P. Vincent, "The difficulty of training deep architectures and the effect of unsupervised pre-training," em *Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2009, pp. 153–160. [Online]. Disponível em: http://jmlr.csail.mit.edu/proceedings/papers/v5/erhan09a/erhan09a.pdf

[13] G. E. Hinton, S. Osindero, e Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, Jul. 2006. [Online]. Disponível em: http://dx.doi.org/10.1162/neco.2006.18.7.1527

[14] J. de Villiers e E. Barnard, "Backpropagation neural nets with one and two hidden layers," *Neural Networks, IEEE Transactions on*, vol. 4, no. 1, pp. 136–141, Jan 1993.

[15] K. Hornik, M. Stinchcombe, e H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, Jul. 1989. [Online]. Disponível em: http://dx.doi.org/10.1016/0893-6080(89)90020-8

[16] P. Vincent, H. Larochelle, Y. Bengio, e P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," em *Proceedings of the 25th International Conference on Machine Learning*, ser. ICML '08. New York, NY, USA: ACM, 2008, pp. 1096–1103. [Online]. Disponível em: http://doi.acm.org/10.1145/1390156.1390294

[17] Y. Bengio, P. Lamblin, D. Popovici, e H. Larochelle, "Greedy layer-wise training of deep networks," em *Advances in Neural Information Processing Systems 19 (NIPS'06)*, B. Schölkopf, J. Platt, e T. Hoffman, Eds. MIT Press, 2007, pp. 153–160.

[18] H. Larochelle, D. Erhan, e P. Vincent, "Deep learning using robust interdependent codes," em *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS-09)*, D. V. Dyk e M. Welling, Eds., vol. 5. Journal of Machine Learning Research - Proceedings Track, 2009, pp. 312–319. [Online]. Disponível em: http://jmlr.csail.mit.edu/proceedings/papers/v5/larochelle09a/larochelle09a.pdf

[19] M. Ranzato, Y. lan Boureau, e Y. L. Cun, "Sparse feature learning for deep belief networks," em *Advances in Neural Information Processing Systems 20*, J. Platt, D. Koller, Y. Singer, e S. Roweis, Eds. Cambridge, MA: MIT Press, 2007, pp. 1185–1192. [Online]. Disponível em: http://books.nips.cc/papers/files/nips20/NIPS2007_1118.pdf

[20] R. Palm, "Deeplearntoolbox, a matlab toolbox for deep learning." [Online]. Disponível em: https://github.com/rasmusbergpalm/DeepLearnToolbox

[21] Encog, "Encog artificial intelligence framework for java and dotnet." [Online]. Disponível em: http://www.heatonresearch.com/encog

[22] E. Busseti, I. Osband, e S. Wong, "Deep learning for time series modeling. stanford, cs 229: Machine learning," 2012. [Online]. Disponível em: http://cs229.stanford.edu/projects2012.html

[23] I. Arel, D. Rose, e T. Karnowski, "Deep machine learning - a new frontier in artificial intelligence research [research frontier]," *Computational Intelligence Magazine, IEEE*, vol. 5, no. 4, pp. 13–18, Nov 2010.

[24] J. Chao, F. Shen, e J. Zhao, "Forecasting exchange rate with deep belief networks," em *Neural Networks (IJCNN), The 2011 International Joint Conference on*, July 2011, pp. 1259–1266.