Autoencoders Assimétricos para a Compressão de Dados IoT

Mateus da Silva Gilbert[1], **Marcello Luiz R. de Campos**[2], **Miguel Elias M. Campista**[1]

[1,2]Universidade Federal do Rio de Janeiro (UFRJ)
[1]GTA/DEL-Poli/PEE-COPPE
[2]SMT/DEL-Poli/PEE-COPPE

{gilbert,miguel}@gta.ufrj.br, campos@smt.ufrj.br

Abstract. *IoT devices typically have severe limitations regarding energy consumption and the number of local computations. Thus, finding solutions that reduce these two issues are always welcome. The generated data may have intrinsic redundancies that allow its compression without loss of information, reducing the amount of data transmitted over the network, one of the most energy-consuming tasks for IoT devices. Consequently, many solutions using neural networks have emerged to reduce data transmission in IoT networks. This paper follows this trend to propose Asymmetric Autoencoders (AAEs), which have fewer neural network layers at the encoder than at the decoder. The proposed structure modifies typical autoencoders with the same number of layers at both the encoder and the decoder. The key idea of the asymmetrical design is to minimize the number of parameters stored and computations performed in IoT devices. Our experiments show improvements compared with symmetrical autoencoders, achieving lower reconstruction errors using temporal samples from a single sensor.*

Resumo. *Os dispositivos IoT possuem severas limitações em consumo de energia e número de computações locais. Assim, encontrar soluções que diminuam esses dois problemas é sempre bem-vindo. Os dados gerados podem apresentar redundâncias intrínsecas que permitam a sua compressão sem perdas de informação, reduzindo a quantidade de dados transmitidos pela rede, uma das tarefas com maior consumo de energia para dispositivos IoT. Consequentemente, muitas soluções que recorrem a redes neurais têm aparecido para reduzir a transmissão de dados em redes IoT. Este artigo segue essa tendência para propor os Autoencoders Assimetricos (AAEs), que possuem menos camadas de redes neurais no codificador que no decodificador. A estrutura proposta modifica autoencoders típicos com o mesmo número de camadas em ambos o codificador e o decodificador. A ideia chave do projeto assimétrico é minimizar o número de parâmetros armazenados e computações realizadas nos dispositivos IoT. Os experimentos mostraram melhorias em comparação aos autoencoders simétricos, atingindo menores erros de reconstrução usando amostras temporais de um único sensor.*

1. Introdução

A sociedade em geral deseja há muito tempo implementar ambientes e ferramentas inteligentes que proporcionem maior conforto ou que sejam capazes de se adaptar a

condições imprevistas de forma autônoma. Nesse sentido, a Internet das Coisas (*Internet of Things* – IoT) torna-se fundamental para realizar tais ambições. A IoT permite que diversos dispositivos sejam incorporados à Internet a fim de explorar sua infraestrutura e obterem a capacidade de se comunicar entre si [Gubbi et al. 2013]. Uma rede IoT característica pode ser composta por uma diversidade de dispositivos, implicando em um cenário de geração heterogênea de dados a taxas distintas, tornando o trabalho de combinar e extrair informações relevantes ainda mais desafiador [Mohammadi et al. 2018].

Na última década, muitos trabalhos vem recorrendo ao aprendizado profundo para implementar soluções em diversas áreas. Esse sucesso é consequência da capacidade de adaptação aos dados das redes neurais, que dispensa pré-tratamento rigoroso e mesmo assim consegue aprender funções não-lineares complexas [LeCun et al. 2015]. Como consequência, observa-se hoje o uso de autoencoders para a compressão e fusão de dados IoT [Bochie et al. 2021]. Os autoencoders são redes neurais particularmente atraentes a problemas que exijam algum tipo de redução de dimensionalidade ou extração de características que consigam ser usadas para reproduzir o dado original [Charte et al. 2018]. Essas funções são graças à sua estrutura formada por entrada e saída com a mesma dimensão, que permite verificar se o dado obtido, após passar pela rede, é o mesmo que foi inserido. Em especial, no cenário de transmissão de dados, os autoencoders se apresentam como ferramentas úteis para obter compressões profundas e agregar dados de fontes distintas. O objetivo é reduzir a frequência e a quantidade de dados transmitidos pelos nós da rede, levando à economia de energia [Bochie et al. 2021]. A redução de transmissões é particularmente importante para a reduzir o custo de cada dispositivo, dado que a transmissão de dados é uma tarefa de alto gasto energético [Akyildiz et al. 2002]. Minimizar o custo individual não só diminui o custo de implementação da rede, como também permite o emprego de mais dispositivos. Em aplicações de sensoriamento, por exemplo, o uso de mais dispositivos pode proporcionar o aumento da área de cobertura e a confiabilidade das amostras coletadas.

Uma tendência observada nas principais soluções que recorrem às redes neurais é o aumento da profundidade, isso é, o aumento da quantidade de camadas que compõem a rede. Redes com mais camadas de neurônios tendem a apresentar uma maior capacidade de se adaptar a problemas mais complexos. Porém, para muitas aplicações IoT, o aumento do número de camadas da rede pode dificultar o uso dessas ferramentas de maneira local, já que implementar e utilizar a rede no próprio dispositivo pode incorrer em altos gastos de memória e recursos computacionais. Desenvolvedores que desejam melhorar o desempenho de suas aplicações devem ponderar o investimento em equipamentos com mais recursos ou a busca por soluções que levem a computação para longe dos nós da rede (p.ex. computação em núvem). No primeiro caso, investir em equipamentos melhores pode aumentar o custo individual dos nós da rede. Já o segundo pode levar a maiores tempos de resposta, maior custo para a transmissão de dados ou a outros problemas que surgem com o envio de dados não processados. Esse último é particularmente problemático para a compressão de dados, uma vez que vai ao encontro do propósito dessa aplicação.

Este trabalho propõe uma alternativa ao dilema da escolha entre construir uma rede de melhor desempenho ou limitar o tamanho do modelo, apresentando as redes chamadas Autoencoders Assimétricos (AAEs). A ideia central dessa nova arquitetura é aproveitar o uso comum da IoT em que a codificação dos dados ocorre nos sensores e a decodificação

no servidor central. Para tal, evita-se a implementação normalmente utilizada de espelhar os blocos de codificação e decodificação, construindo um modelo em que o número de camadas, neurônios e outros recursos sejam menores no codificador. Os AAEs, como consequência, afastam a maioria dos parâmetros e computações dos nós da rede IoT, seguindo assim uma abordagem já encorajada no desenvolvimento de sistemas de compressão de redes de sensores [Razzaque et al. 2013]. Ao utilizar os AAEs, um sistema de compressão ou de extração das principais características de dados IoT com poucas camadas torna-se possível, viabilizando o uso do aprendizado local nos próprios dispositivos da rede IoT. Os resultados experimentais são promissores. Este artigo constrói um sistema de compressão de amostras de temperatura, permitindo observar que os AAEs conseguem reduzir o número de operações realizadas no codificador, quando comparados aos autoencoders simétricos (AEs). Mesmo assim, os AAEs propostos apresentam um desempenho similar às suas concorrentes simétricas em termos de erro de reconstrução do sinal. Dependendo da aplicação, é possível até mesmo superar os AEs, se algum conhecimento prévio sobre o fenômeno de interesse for utilizado na construção do modelo. Nos experimentos, ao utilizar modelos com camadas convolucionais, foi possível explorar as relações temporais de curto e médio prazo para diminuir o erro de reconstrução na saída do AAE e, assim, superar os modelos simétricos.

Este trabalho está organizado como se segue. A Seção 2 revisa a literatura acerca do uso de autoencoders para solucionar problemas em IoT, apontando os artigos que dialogam com o trabalho atual. A Seção 3 faz uma breve revisão sobre redes neurais e suas características mais relevantes para o trabalho. Depois, na Seção 4, os AAEs são propostas. A Seção 5 discute a metodologia empregada, seguida pelos resultados experimentais na Seção 6. Por fim, a Seção 7 conclui o trabalho e apresenta possibilidades futuras para o emprego dos AAEs propostos.

2. Trabalhos Relacionados

No contexto de redução e agregação de dados, os autoencoders são os modelos mais comuns dentre as demais redes neurais. Isso ocorre dada a capacidade dessas redes de, na ausência de dados classificados, aprenderem a extrair as informações mais relevantes de maneira autônoma [Charte et al. 2018]. Não por acaso, essas são as principais redes neurais em uso para compressão, agregação e fusão de dados IoT.

Alsheikh et al. apresentam a compatibilidade dos AEs para a tarefa de compressão de dados em redes de sensores [Alsheikh et al. 2016]. Os autores demonstram que os AEs com poucas camadas são capazes de se adaptar melhor a fenômenos complexos, que necessitam de funções não-lineares para comprimir os dados de maneira satisfatória. Esse é um caso comum de dados IoT, que apresentam problemas como alta heterogeneidade, presença intermitente de ruído e taxas distintas de amostragem entre sensores. Adicionalmente, os autores demonstram que os AEs oferecem grandes taxas de compressão em detrimento de pequenos erros de reconstrução, quando comparados a outras técnicas já em uso para a redução de dados em redes de sensores (p. ex., *Lightweight Temporal Compression* [Schoellhammer et al. 2004]). Alsheik et al. demonstram ainda que os recursos e os gastos computacionais necessários para implementar o algoritmo de compressão no sensor são baixos. Entretanto, um problema da abordagem simétrica, a ser explorado neste artigo, é a necessidade de aumentar a profundidade da rede quando se deseja uma melhora de desempenho. Quando isso é feito, o número de parâmetros armazenados e

computações executadas nos dispositivos da rede pode aumentar consideravelmente. Os AAEs propostos mantém as vantagens descritas por Alsheik et al., oferecendo uma alternativa onde o custo para o nó sensor é mantido, mesmo com o aumento da profundidade do autoencoder.

Em outro trabalho, Ghosh e Grolinger empregam os AEs para extração de características relevantes e reduzir a dimensionalidade do dado a ser enviado para processamento em uma nuvem [Ghosh and Grolinger 2019]. Diferente do caso anterior, os autores analisam os AEs com diferentes profundidades, verificando como o processo de redução de dimensionalidade afeta uma aplicação hospedada na nuvem. Os autores constatam que os AEs não afetam a aplicação significantemente. Nesse trabalho, para produzir vetores menores para transmissão, os autores recorrem ao aumento de camadas, utilizando mapeamentos intermediários até o tamanho desejado. Usar os AAEs, nesse caso, diminui ou elimina a necessidade de realizar esses mapeamentos intermediários, permitindo reduzir o número de camadas e, consequentemente, parâmetros e computações locais. Já em outro trabalho, Yu et al. utilizam AEs em conjunto com veículos aéreos não tripulados (VANT) para coletar os dados de sensores próximos, enviando os dados comprimidos para uma nuvem [Yu et al. 2018]. Diferente dos casos anteriores, a operação da compressão é feita em um nó intermediário (a VANT), que carrega um codificador (do AE) para cada conjunto de sensores. Nessa situação, os AEs podem ajudar na escalabilidade do problema, já que diminuem o número de parâmetros para cada AE, sendo vantajoso ao nó intermediário. Outros trabalhos que recorrem aos autoencoders para redução e agregação de dados podem ser encontrados em Bochie et al. [Bochie et al. 2021].

3. Visão Geral de Redes Neurais

Redes neurais são algoritmos de aprendizado de máquinas poderosos capazes de aprender funções complexas, quando recursos suficientes são fornecidos, tendo em vista suas propriedades de aproximador universal [Cybenko 1989]. Adicionalmente, elas diferem dos demais algoritmos de aprendizado de máquinas mais primitivos graças à sua capacidade de se adaptar a problemas utilizando dados não processados [LeCun et al. 2015].

A unidade fundamental de uma rede neural é o chamado neurônio. Essa estrutura é composta por várias entradas, uma maneira de combinar essas entradas e uma transformação não-linear. Geralmente, um único neurônio aplica a transformação não-linear (referida como função de ativação) a uma soma ponderada dessas entradas. Uma camada é formada pelo conjunto desses neurônios que estão em uma mesma posição na hierarquia da rede, geralmente compartilhando o mesmo conjunto de entradas. A rede neural é construída encadeando essas camadas, permitindo que as informações extraídas e geradas sejam recombinadas conforme a arquitetura da rede.

3.1. Autoencoders

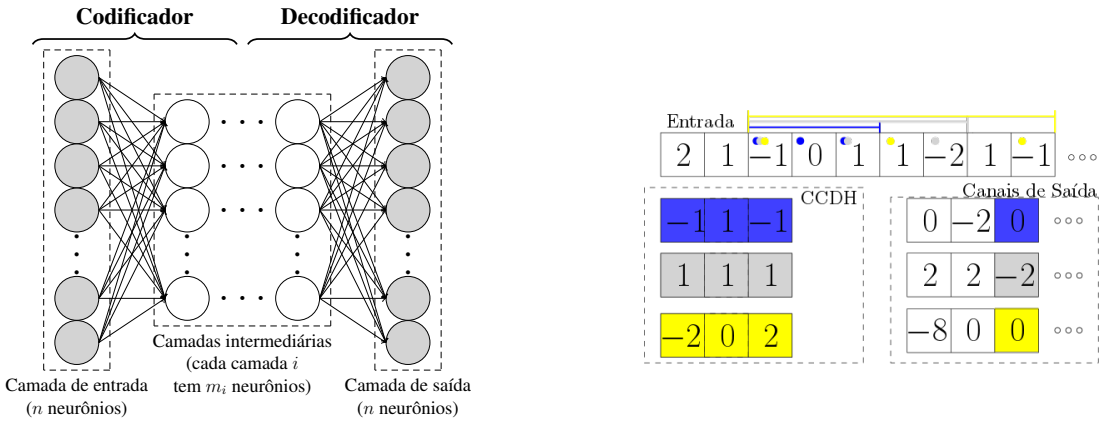
Dentro do conjunto de redes neurais, a principal escolha para problemas de compressão e redução de dimensionalidade, em geral, é o autoencoder. Nessas situações é comum construir a rede de maneira que as camadas internas tenham menos neurônios do que as camadas de entrada (e saída), para que a própria estrutura da rede force as camadas a extrair as características relevantes que "resumam" o dado inserido na entrada [Charte et al. 2018]. Durante o treinamento, a entrada e a saída da rede são comparadas, pois o objetivo é que a informação que flui pelas camadas da rede seja suficiente

para reconstruir o dado original. A implementação mais comum é dispor as camadas de tal forma que o codificador (porção que faz a redução de dimensionalidade) e o decodificador (que reconstrói o dado) são espelhados, isso é, possuem a mesma quantidade e tipo de camadas. Não por acaso, essa tendência é observada nas propostas para problemas em IoT [Alsheikh et al. 2016, Ghosh and Grolinger 2019, Bochie et al. 2021]. A Figura 1(a) traz uma ilustração de um AE típico.

3.2. Camadas Convolucionais

Uma camada muito popular em aprendizado profundo são as chamadas camadas convolucionais. Seu principal efeito é encorajar computações locais, substituindo a camada densa em que todos os neurônios de camadas adjacentes estão conectados (todos têm as mesmas entradas) por conexões localizadas que compartilham os próprios pesos [LeCun et al. 1995]. Na prática, isso faz com que a camada seja composta por um pequeno conjunto de parâmetros (chamado filtro ou *kernel* convolucional) que desliza sobre os dados de entrada. Dadas essas características, um filtro extrai um mesmo tipo de informações localizadas, por exemplo, bordas, no caso de imagens. Para aumentar a variedade de informações extraídas, é comum aumentar o número de filtros por camada convolucional, de tal modo que vários canais de informação são gerados. Essas informações podem ser recombinadas ou reprocessadas, no decorrer da rede. No cenário estudado, faz-se a opção por utilizar as camadas convolucionais com múltiplos filtros para explorar as relações locais inerentes ao dado que será comprimido.

Uma variação das camadas convolucionais, central para uma das implementações estudadas, é a Camada Convolucional Dilatada [Yu and Koltun 2015]. Uma desvantagem das camadas convolucionais tradicionais é que é necessário o encadeamento de camadas convolucionais, que promovem redução de dimensionalidade, para que características de baixa resolução (que se expandem por uma região maior do dado) sejam extraídas. A convolução dilatada é justamente uma alternativa para esse problema, permitindo uma expansão da área de cobertura do filtro. Para encorajar a extração desses dados de baixa resolução, zeros são acrescentados entre os parâmetros da rede, utilizando entradas mais distantes uma das outras. Uma implementação especial, adotada nos modelos propostos, é construir uma camada com diferentes taxas de dilatação, chamada Camada Con-



(a) Autoencoder simétrico (AE) típico.

(b) Operação simplificada de uma CCDH.

Figura 1. Estruturas básicas para a realização dos AAEs.

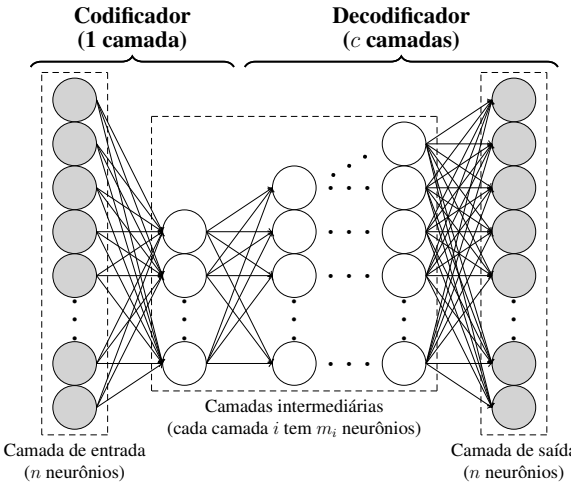


Figura 2. Autoencoder assimétrico proposto (AAE) com $c > 1$.

volucional Dilatada Hibrida (CCDH) [Wang et al. 2018]. Assim, é possível extrair numa mesma camada, informações de diferentes resoluções. Essa abordagem vem ganhando relevância, útil neste projeto para a extração de informações distintas que levem ao sucesso na reconstrução dos sinais comprimidos, como será visto nos experimentos. A Figura 1(b) traz um esboço da operação da CCDH. Em azul, cinza e amarelo os filtros com dilatação 1 (filtro convolucional convencional), 2 e 3. Na iteração ilustrada, a primeira amostra dos três filtros é a posição com valor -1, com os dois últimos filtros "pulado" uma e duas amostras, respectivamente, para completar a computação. Nos canais de saída para cada filtro, pode-se observar as saídas durante três iterações.

4. Autoencoders Assimétricos (AAEs)

Os AAEs propostos são construídos de tal forma que o número de camadas, neurônios e outras estruturas que compõem o bloco de codificação é menor que o do bloco de decodificação. Dessa forma, é possível limitar o número de camadas que devem ser armazenadas nos nós da rede IoT, buscando melhorar a reconstrução do dado no receptor (no caso de compressão) ou para melhor preservar a informação nos dados na saída do codificador. Como será visto nos resultados, essa abordagem permite obter as vantagens oriundas do aumento da profundidade da rede, sem que os custos de implementação do codificador aumentem. O deslocamento dos recursos para o decodificador exige que o dado da saída do codificador seja suficientemente rico para garantir a reconstrução do sinal.

A implementação mais simples limita o número de camadas do bloco de codificação, permitindo que o decodificador possua quantas camadas forem necessárias. Especialmente nos experimentos deste trabalho, todos os AAEs foram construídos com uma única camada de codificação, como visto na Figura 2.

AAEs com Camadas Convolucionais: Dependendo do tipo de amostras coletadas, torna-se possível explorar as relações inerentes ao dado para melhorar o procedimento de reconstrução. Em especial, os dados com relações temporais, como os utilizados nos experimentos deste trabalho, permitem o uso de camadas convolucionais em substituição

à última camada do decodificador usado pelo AAE. A ideia do uso de camadas convolucionais é reforçar as influências locais. No caso dos dados utilizados, essas relações são temporais de curto e médio prazo. Para diferenciar esses modelos dos anteriores, estes são referidos como CAAEs (AAEs com camadas Convolucionais). Adicionalmente, utilizando a implementação com CCDH, é possível extrair as informações temporais em diferentes resoluções, melhorando a reconstrução do sinal significativamente.

5. Metodologia

Os AAEs são avaliados em um cenário de compressão de amostras temporais de temperatura. A base de dados utilizada contém medidas de um sensor fornecida pelo *American River Hydrologic Observatory* (ARHO) [Bales et al. 2020]. Os dados são provenientes de um sensor que possui um período de amostragem de 15 minutos, localizado em uma região próxima a Caples Lake, na Califórnia. As medidas foram coletadas de junho de 2014 até outubro de 2017.

A fim de simular uma situação em que os dados são enviados após o acúmulo de uma sequência de amostras, cada entrada (e saída) do autoencoder é composta por 100 amostras. Seguindo a estrutura básica dessa rede, após o treinamento, as camadas são divididas de tal forma que o codificador e decodificador resultantes correspondem aos blocos de codificação e decodificação do autoencoder, respectivamente. Optou-se por simular uma compressão para 25 amostras. Assim, é simulado um cenário em que as amostras seriam acumuladas em um nó sensor por um período que supera um dia, enviadas após compressão utilizando o bloco de codificação do autoencoder no próprio sensor, sendo posteriormente recuperados por um dispositivo remoto que contém o decodificador do referido autoencoder.

5.1. Configuração dos Autoencoders

A ideia é avaliar a rede proposta em um cenário típico de sensoriamento relativamente simples, proporcionando a utilização de modelos com poucas camadas, para avaliar a abordagem proposta. Vale mencionar que a efetividade do sistema é proporcional à quantidade de dados transmitidos. Primeiramente, AEs são avaliados, verificando a melhora em reconstrução do sinal à medida que o número de camadas aumenta. A seguir, os AAEs são avaliados. Nessa configuração é possível demonstrar que os AAEs apresentam a mesma tendência de melhora em reconstrução, não ficando atrás dos seus equivalentes simétricos (com o mesmo número de camadas de decodificação). A Tabela 1 traz a configuração dos decodificadores, assim como o identificador que será utilizado nos experimentos para distinguir cada modelo. No caso dos AEs, a configuração do bloco de codificação espelha o de decodificação, enquanto nos AAEs, o bloco de codificação é o mesmo do AE-0, isso é, é composto por uma única camada com 100 entradas e 25 saídas, denotada por $100 \rightarrow 25$.

Todos os autoencoders da Tabela 1 foram construídos com camadas densas. A função `SeLU` [Klambauer et al. 2017] é utilizada como função de ativação dos neurônios das camadas intermediárias, seguindo resultados obtidos em experimentos iniciais [Gilbert 2021]. Por fim, as camadas de saída utilizam a função `sigmoid` como função de ativação. Por conta disso, as amostras de temperatura devem ser mapeadas para um subintervalo em $(0, 1)$ para que a rede consiga reproduzir a sequência de valo-

Tabela 1. Configuração dos decodificadores dos autoencoders simétricos (AE) e assimétricos (AAE) avaliados. As setas indicam as transformações aplicadas nos decodificadores (ex. $x \rightarrow y$ indica que está havendo uma transformação de um vetor de dimensão x para y)

Id. do AE e do AAE	Nº de camadas	Configuração do decodificador
AE-0/AAE-0	1	$25 \rightarrow 100$
AE-1/AAE-1	2	$25 \rightarrow 50 \rightarrow 100$
AE-2/AAE-2	3	$25 \rightarrow 50 \rightarrow 75 \rightarrow 100$
AE-3/AAE-3	4	$25 \rightarrow 45 \rightarrow 65 \rightarrow 85 \rightarrow 100$
AE-4/AAE-4	5	$25 \rightarrow 40 \rightarrow 55 \rightarrow 70 \rightarrow 85 \rightarrow 100$

res em sua saída. Nos experimentos, optou-se por mapear as amostras para o intervalo $[0.001, 0.999]$.

Os CAAEs utilizam o AAE-3 como base. Três modelos são estudados: (1) substituição das camadas densas que promovem as transformações $50 \rightarrow 75 \rightarrow 100$ por uma única camada de convolução transposta [Dumoulin and Visin 2016] com quatro filtros de tamanho 3 e taxa de deslize (número de amostras que o filtro se desloca a cada iteração [Dumoulin and Visin 2016]) $s = 2$; (2) acréscimo de uma camada convolucional com quatro filtros de tamanho 3 e taxa de deslize unitário entre a camada que promove a transformação de $25 \rightarrow 50$ e a camada convolucional acrescentada no modelo anterior; (3) substituição dos filtros da camada acrescentada no modelo anterior por filtros com dilatação 1, 2, 3 e 4, gerando uma CCDH; (4) modificação da camada de convolução transposta para que cada filtro da CCDH tenha quatro filtros convolucionais. Todas as modificações visam incentivar a extração de mais características distintas, que auxiliem o AAE em sua reconstrução. Aumentando o número de filtros, é possível permitir a camada convolucional extrair características complementares, reforçado com o acréscimo do CCDH, dadas suas características. Em todos os casos, as saídas dos múltiplos filtros são combinados utilizando uma camada convolucional as mapeia para um único canal de saída. O resultado dessa fusão de informações é saída dos CAAEs.

5.2. Configuração dos Experimentos

Após experimentos iniciais [Gilbert 2021], decidiu-se por adotar a seguinte configuração. As amostras de temperatura são divididas de tal forma que as amostras até junho de 2016 foram separadas para treino (e validação), enquanto as demais corresponderam ao conjunto de testes. Optou-se por essa abordagem para verificar o desempenho dos modelos treinados a um período de trabalho que superava um ano. Além disso, buscou-se emular uma situação em que poucas amostras estivessem disponíveis para treino. Esse último ficou evidente com os resultados encontrados com o AE mais profundo, que obteve um desempenho geral pior que o AE com uma camada a menos.

Uma vez separados os conjuntos de treino e teste, os conjuntos de 100 amostras (para facilitar a compreensão do restante do texto, cada conjunto é referido como um vetor de amostras) foram agrupados utilizando o método de janelas deslizantes. Dessa forma, produz-se mais vetores do que se um simples particionamento do conjunto de treino fosse adotado. Além disso, ao fazer a opção por esse método de criação de vetores de treino, é possível permitir que situações distintas ao longo do dia possam ocupar todas as posições dos vetores resultantes. Garantir essa variedade de vetores é particularmente útil para per-

mitir que a rede treinada consiga generalizar bem. Nos experimentos, optou-se por adotar taxas de deslize iguais a 10, 17 e 23 para maximizar a ocorrência de vetores únicos (dado que são números primos entre si). Após esse processo de criação de vetores de amostras, esses vetores foram divididos aleatoriamente nos conjuntos de treino e validação, obedecendo à proporção $80:20$.

Os vetores de treinos são agrupados em lotes de 20, que servem de entrada para o treino da rede a cada iteração. As redes são treinadas utilizando o método de *early stopping*, em que o treinamento é encerrado após não se verificar uma melhora no desempenho das redes com o conjunto de validação após 15 épocas (época equivale a utilizar todos os vetores do conjunto de treino em cada iteração de treinamento). Adicionalmente, optou-se por adotar uma taxa de aprendizado variável que oscila entre dois valores limites [Smith 2017]. Esses valores foram definidos para cada autoencoder previamente, observando o comportamento da rede para diferentes valores de taxa de aprendizado variando de 1×10^{-4} a 0.1. Por fim, optou-se por utilizar Nadam como função de otimização [Dozat 2016], uma variação do otimizador Adam [Kingma and Ba 2014] (popular no treinamento de redes neurais) que utiliza momento de Nesterov [Nesterov 1983] em sua implementação. Após o treinamento, o desempenho de cada rede é analisado tirando uma média do erro de reconstrução de cada vetor de amostras construídas do conjunto de treinamento. Optou-se por utilizar o Erro Médio Quadrático (MSE) para medir a diferença entre o sinal original e o reconstruído na saída da rede. Para cada modelo analisado, optou-se por repetir os experimentos 10 vezes com diferentes inicializações. Dessa forma, é possível obter uma ideia das dificuldades de treinar cada modelo e o desempenho correspondente quando os modelos são comparados entre si.

Os programas utilizados nos experimentos deste trabalho podem ser encontrados no `Github` de um dos autores [Gilbert 2021]. Em especial, os programas são escritos em `Python` utilizando as bibliotecas `Numpy` [Harris et al. 2020] e `Tensorflow` [Abadi et al. 2015]. Os experimentos foram executados em um Dell Precision 7920 Rack com placa de vídeo NVIDIA RTX A4000.

6. Resultados Experimentais

Os experimentos são divididos em três etapas. Na primeira, seguindo o modelo básico simétrico [Alsheikh et al. 2016], o desempenho dos AEs como compressores é avaliado à medida que a profundidade do modelo aumenta. A seguir, os AAEs são analisados, comparando-os com seus concorrentes simétricos. Por fim, conhecendo as particularidades do sinal utilizado nos experimentos, modificações ao AAE-3 são adicionadas, colocando camadas convolucionais a fim de explorar as relações temporais inerentes ao dado utilizado.

6.1. Mais Camadas, Melhor Reconstrução

A Figura 3 traz a distribuição de resultados para os autoencoders simétricos, em um diagrama de caixas (*boxplot*). É notável a melhora de desempenho à medida que o número de camadas aumenta, com a melhora interrompendo após a inserção de 5 camadas (AAE-4). Esse comportamento pode ser atribuído à possibilidade do AAE-3 já ser um modelo simétrico suficientemente complexo para o qual uma melhora de desempenho não seja mais possível. Alternativamente, há a possibilidade de a quantidade de dados

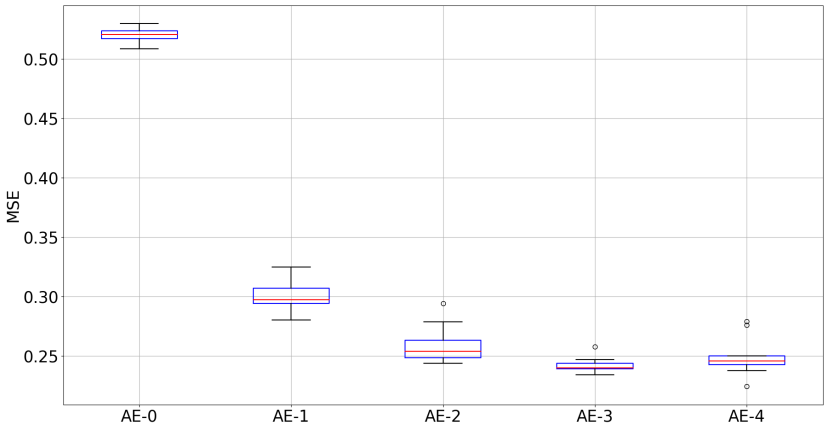


Figura 3. Notável melhora do erro de reconstrução à medida que o número de camadas aumenta.

do conjunto de treino não ser suficiente para ajustar todos os parâmetros do autoencoder mais complexo. Esse último é um problema comum no uso de aprendizado profundo com dados que são sequencias temporais [Iwana and Uchida 2021], o que pode ser outro indicativo da inviabilidade de utilizar autoencoders simétricos profundos com dados provenientes de redes IoT.

A Tabela 2 traz uma estimativa do número de parâmetros necessários para implementar o codificador (no caso, os parâmetros que compõem cada camada). É aparente o impacto do aumento do número de camadas no aumento do número de parâmetros e operações locais executadas no dispositivo sensor. Assim, se o dispositivo em questão for sensível a esse aumento, o tamanho do autoencoder utilizado tende a ser limitado. Como será visto a seguir, esse problema pode ser contornado utilizando os AAEs propostos, que tem como produto um codificador com os mesmos requisitos computacionais que a configuração AE-0, ou seja, com número de parâmetros igual a 2525.

6.2. AEs vs. AAEs

Como pode ser visto na Figura 4, não só os AAEs superam o modelo base facilmente (recordando a Figura 3, o melhor desempenho obtido com o AE-0 apresentou um MSE superior a 0.5) como conseguem um desempenho equiparável aos concorrentes simétricos. A figura mostra duas tendências. A primeira é que os AAEs com o mesmo decodificador (em termos de profundidade) não apresentam uma diferença muito grande em desempenho. Isso é mais evidente com os modelos AAEs mais profundos, em que distância entre as medianas é inferior a 0.02 em termos MSE. Adicionalmente, outra

Tabela 2. Número de parâmetros obtidos com o Tensorflow [Abadi et al. 2015] para implementar o codificador do autoencoder simétrico (AE).

Id. do AE	Nº de camadas	Nº de parâmetros no codificador
AE-0	1	2525
AE-1	2	6325
AE-2	3	12650
AE-3	4	18295
AE-4	5	21775

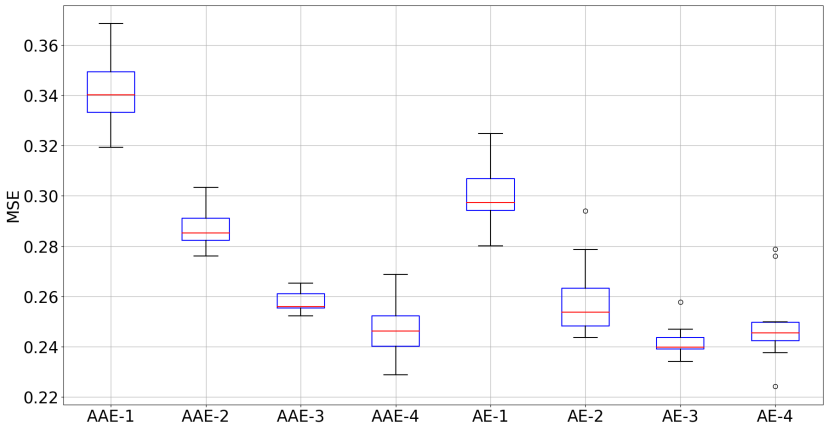


Figura 4. AAEs conseguem acompanhar o desempenho de seus pares simétricos, com AAE-4 podendo, inclusive em alguns casos, superar os AEs estudados.

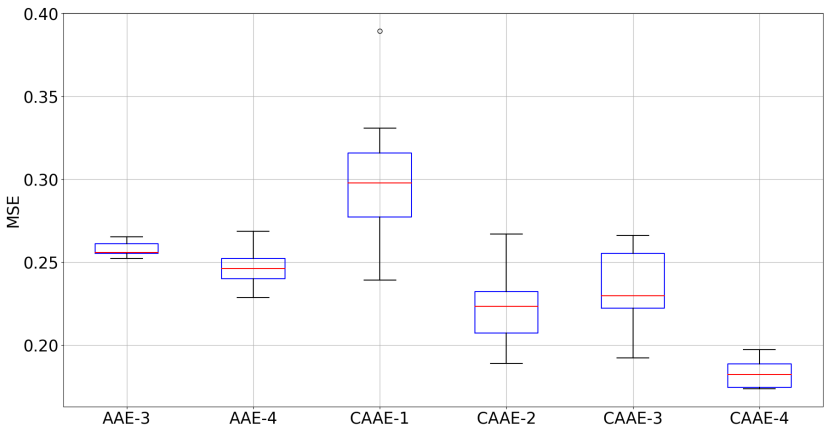


Figura 5. CAAEs comparados as melhores AAEs. Acrescentar estruturas ao decodificador da AE que explorem as relações temporais do sinal original permitem uma melhor em desempenho.

tendência observada é que em alguns casos os AAEs conseguem ter desempenho similar, ou melhor, que os modelos simétricos cujos decodificadores possuem uma camada a menos (p. ex. AAE-2 e AE-1). Na figura 4 pode-se observar que em todos os casos há superposição entre os intervalos interquartis.

Antes de seguir para o último experimento, é interessante observar que a tendência observada com os autoencoders simétricos de parada da melhora de desempenho com o aumento da profundidade do modelo não é observada com os AAEs. Esse resultado pode ser um indicativo da compatibilidade dos AAEs com problemas de sequências temporais, uma vez que esses modelos possuem menos parâmetros que os autoencoders simétricos.

6.3. Explorando Relações Locais: CAAEs

A Figura 5 traz o desempenho dos CAAEs, em que camadas convolucionais são acrescentadas, comparado-os aos dois AAEs de melhor desempenho. Recordando, esses novos AAEs são obtidos modificando a arquitetura do AAE-3, substituindo a última camada densa por um bloco de camadas convolucionais. Num primeiro momento, pode-se

observar que executar uma simples substituição da camada densa por uma convolucional (CAAE-1) não traz vantagens em desempenho. Essa situação muda com o acréscimo de mais uma camada, entre a camada densa e a convolucional que promove aumento dimensional. Essa nova camada permite extrair diferentes informações, sendo posteriormente agregadas pela rede para produzir o sinal reconstruído. Especialmente para os modelos com convoluções dilatadas híbridas, cada canal de saída extrai uma informação que é complementar às demais, sendo posteriormente agregadas para produzir a saída da rede. Isso também é possível com o modelo mais simples, mas a presença das diferentes dilatações reforçam esse fenômeno. Por fim, é vista a importância de, com camadas convolucionais, utilizar múltiplos canais de saída. Isso é algo já observado em outras áreas (por exemplo, visão computacional) justamente para a extração de informações distintas e complementares. Isso fica evidente ao comparar CAAE-3 e CAAE-4, onde as vantagens de se adotar a camada de convolução hibrida só é aproveitada quando mais filtros de convolução transposta são colocados na saída de cada canal.

De maneira mais abrangente, os resultados apresentados pela Figura 5 reforçam dois pontos. O primeiro é explorar as características do fenômeno que gera as amostras. No cenário estudado, as amostras de temperatura apresentam relações temporais. Sendo assim, acrescentar camadas convolucionais permite explorar essas relações. Com as camadas dilatadas, as relações temporais de diferentes escalas (curto e médio prazo) podem ser extraídas. O segundo ponto é notar que não é só possível conseguir resultados que se equiparam os AEs tradicionais flexibilizando o arranjo da rede, como é possível superá-los se o AAE for construído de maneira adequada.

7. Conclusões e Trabalhos Futuros

O trabalho apresentou os Autoencoders Assimétricos (AAEs), uma alternativa de baixo custo para o desenvolvimento de soluções de compressão e redução de dimensionalidade em cenários IoT. Os experimentos mostraram ser possível obter modelos de compressão que não exigem o aumento significativo do número de camadas que devem ser armazenadas nos dispositivos sensores. Isso aponta para a possibilidade, não só do emprego dos AAEs em cenários de compressão, mas também em situações em que se deseja extrair as características principais de um dado antes de sua transmissão. Em um cenário simples de compressão de amostras de temperatura, foi possível obter um desempenho equiparável entre os AAEs e os autoencoders simétricos (AEs), tendo inclusive modelos que superaram os AEs. Isso fica especialmente evidente quando as características do fenômeno que gera as amostras foram considerados na arquitetura do modelo proposto. Explorando as relações temporais inerentes às amostras de temperatura, foi possível construir um AAE que obteve um MSE inferior a $0,2$, enquanto o melhor AE analisado não superou $0,23$.

Como trabalhos futuros pretende-se avaliar os AAEs em problemas de compressão mais desafiadores, como na compressão conjunta de dados de diferentes sensores. Adicionalmente, vislumbra-se incorporar os AAEs às técnicas tradicionais de compressão, a fim de trazer as vantagens da abordagem por meio de redes neurais, como robustez e generalização. Por fim, outra área que deve ser explorada é a redução de dimensionalidade de modo geral, na qual os AAEs podem ser utilizados para desenvolver sistemas de extração de características compactos.

Agradecimentos

O presente trabalho foi realizado com apoio do CNPq, da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001, FAPERJ (E-26/211.144/2019), FAPESP (2015/24494-8, 2014/50937-1, 2015/24485-9) e RNP (Programa de Monitoramento (PMon) 2021).

Referências

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E. (2002). Wireless sensor networks: a survey. *Computer networks*, 38(4):393–422.

Alsheikh, M. A., Lin, S., Niyato, D., and Tan, H.-P. (2016). Rate-distortion balanced data compression for wireless sensor networks. *IEEE Sensors Journal*, 16(12):5072–5083.

Bales, R., Cui, G., Rice, R., Meng, X., Zhang, Z., Hartsough, P., Glaser, S., and Conklin, M. (2020). Snow depth, air temperature, humidity, soil moisture and temperature, and solar radiation data from the basin-scale wireless-sensor network in American River Hydrologic Observatory (ARHO). `https://doi.org/10.6071/M39Q2V`. Accessed in 08/06/2020.

Bochie, K., Gilbert, M. S., Gantert, L., Barbosa, M. S., Medeiros, D. S., and Campista, M. E. M. (2021). A survey on deep learning for challenged networks: Applications and trends. *Journal of Network and Computer Applications*, page 103213.

Charte, D., Charte, F., García, S., del Jesus, M. J., and Herrera, F. (2018). A practical tutorial on autoencoders for nonlinear feature fusion: Taxonomy, models, software and guidelines. *Information Fusion*, 44:78–96.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314.

Dozat, T. (2016). Incorporating nesterov momentum into adam. In *International Conference on Learning Representations (ICLR)*.

Dumoulin, V. and Visin, F. (2016). A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*.

Ghosh, A. M. and Grolinger, K. (2019). Deep learning: Edge-cloud data analytics for IoT. In *2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE)*, pages 1–7. IEEE.

Gilbert, M. d. S. (2021). Redução de dados em redes de sensores utilizando redes neurais. Projeto Final de Graduação. `https://www.gta.ufrj.br/ftp/gta/TechReports/Gilbert21/Gilbert21.pdf`.

Gubbi, J., Buyya, R., Marusic, S., and Palaniswami, M. (2013). Internet of things (IoT): A vision, architectural elements, and future directions. *Future generation computer systems*, 29(7):1645–1660.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, 585(7825):357–362.

Iwana, B. K. and Uchida, S. (2021). An empirical survey of data augmentation for time series classification with neural networks. *Plos one*, 16(7):e0254841.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Klambauer, G., Unterthiner, T., Mayr, A., and Hochreiter, S. (2017). Self-normalizing neural networks. *Advances in neural information processing systems*, 30:971–980.

LeCun, Y., Bengio, Y., et al. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

Mohammadi, M., Al-Fuqaha, A., Sorour, S., and Guizani, M. (2018). Deep learning for iot big data and streaming analytics: A survey. *IEEE Communications Surveys & Tutorials*, 20(4):2923–2960.

Nesterov, Y. E. (1983). A method for solving the convex programming problem with convergence rate o (1/kˆ 2). In *Dokl. akad. nauk Sssr*, volume 269, pages 543–547.

Razzaque, M. A., Bleakley, C., and Dobson, S. (2013). Compression in wireless sensor networks: A survey and comparative evaluation. *ACM Transactions on Sensor Networks (TOSN)*, 10(1):1–44.

Schoellhammer, T., Greenstein, B., Osterweil, E., Wimbrow, M., and Estrin, D. (2004). Lightweight temporal compression of microclimate datasets [wireless sensor networks]. In *29th Annual IEEE International Conference on Local Computer Networks*, pages 516–524.

Smith, L. N. (2017). Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 464–472. IEEE.

Wang, P., Chen, P., Yuan, Y., Liu, D., Huang, Z., Hou, X., and Cottrell, G. (2018). Understanding convolution for semantic segmentation. In *2018 IEEE winter conference on applications of computer vision (WACV)*, pages 1451–1460. Ieee.

Yu, F. and Koltun, V. (2015). Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*.

Yu, T., Wang, X., and Shami, A. (2018). Uav-enabled spatial data sampling in large-scale iot systems using denoising autoencoder neural network. *IEEE Internet of Things Journal*, 6(2):1856–1865.