UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
PROJETO DE GRADUAÇÃO

ENRICO ZARDO FERREIRA

PREVISÃO DE CARGA EM MEDIDORES INTELIGENTES APOIADA EM TÉCNICAS DE *DEEP LEARNING* E SÉRIES TEMPORAIS

Vitória-ES

Outubro/2021

ENRICO ZARDO FERREIRA

PREVISÃO DE CARGA EM MEDIDORES INTELIGENTES APOIADA EM TÉCNICAS DE *DEEP LEARNING* E SÉRIES TEMPORAIS

Parte manuscrita do Projeto de Graduação do aluno Enrico Zardo Ferreira, apresentado ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Engenheiro Eletricista.

Vitória-ES

Outubro/2021

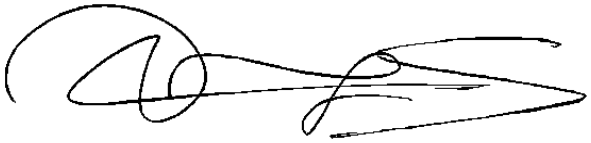
ENRICO ZARDO FERREIRA

PREVISÃO DE CARGA EM MEDIDORES INTELIGENTES APOIADA EM TÉCNICAS DE *DEEP LEARNING* E SÉRIES TEMPORAIS

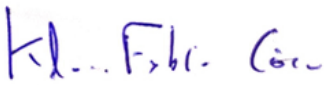
Parte manuscrita do Projeto de Graduação do aluno Enrico Zardo Ferreira, apresentado ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Engenheiro Eletricista.

Aprovado em 6 de Outubro de 2021.

COMISSÃO EXAMINADORA:



Prof. Dr. Jorge Leonid Aching Samatelo
Universidade Federal do Espírito Santo
Orientador



Prof. Dr. Klaus Fabian Côco
Universidade Federal do Espírito Santo
Examinador

Dr. Clebeson Canuto dos Santos
Examinador

Vitória-ES

Outubro/2021

Aos meus amigos e familiares por todo apoio e principalmente pelo amor.

AGRADECIMENTOS

Este projeto consolida o encerramento de uma importante etapa. Não é exagero afirmar que isso só foi possível graças às pessoas incríveis com que cruzei e aprendi na minha vida. Este agradecimento é para vocês.

Agradeço imensamente aos meus pais e família, pelo cuidado, apoio e educação dados com muito amor desde que nasci.

À minha namorada por todo o carinho, compreensão e companhia ao longo dessa árdua jornada.

À todos aqueles que eu tenho o prazer de chamar de amigos, sejam eles da escola, da universidade ou do trabalho. Obrigado por deixar tudo mais leve e alegre.

A meu orientador Jorge Leonid Aching Samatelo por todo zelo, tutela e orientação, não apenas neste trabalho, mas em vários momentos de minha vida acadêmica.

À banca examinadora pela aceitação do convite e pelo tempo investido para leitura e avaliação desse trabalho.

Agradeço à Universidade Federal do Espirito Santo pela minha formação.

RESUMO

O nível de desenvolvimento tecnológico alcançado pela humanidade é sem precedentes. Seja no transporte intercontinental, na produção alimentícia, na construção de arranha-céus ou para saber que horas são, a tecnologia está presente. Para que tudo isso possa funcionar corretamente, uma notável parte dessas invenções precisam ser abastecidas com energia elétrica. Esta ampla demanda resulta num consumo de eletricidade igualmente vultoso. Para que este fornecimento possa garantir uma utilização agradável e natural aos usuários finais, um bom gerenciamento da rede de distribuição e transmissão de eletricidade é alicerçador. Como a energia gerada é consumida instantaneamente, uma previsão precisa da quantidade de energia que será exigida se torna uma arma valiosa contra falhas e perdas, evitando que distribuidoras e operadores sofram com sistemas que não são confiáveis e onerosos. O objetivo deste projeto é realizar a previsão de carga residencial com técnicas de aprendizado de máquina (ML, do inglês *machine learning*) e aprendizado profundo (DL, do inglês *deep learning*), tendo em vista a enorme capacidade desses métodos em identificar padrões e ser resiliente a ruído. Para fazer isso, serão usadas informações provenientes de medições de consumo elétrico de medidores inteligentes (do inglês *smart meters*). Estes dados serão tratados como séries temporais e analisados por camadas e redes específicos desta área. Neste trabalho serão testadas e comparadas abordagens usando arquiteturas baseadas em redes neurais convolucionais (CNN, do inglês *convolutional neural networks*), Memória de Prazo-Curto longa (LSTM, do inglês *long short-term memory*) e Unidade Recorrente Seletiva (GRU, do inglês *gated recurrent unit*) que possuem viés induzido (do inglês *inductive bias*) em sua concepção, junto com estruturas que não dispõem disso, como blocos MLP-Mixer. Busca-se, por meio deste estudo, validar a previsão do consumo elétrico residencial de forma acertada e comparar o desempenho desses diferentes modelos de *machine learning*.

Palavras-chave: *Machine Learning*. *Deep-learning*. Séries temporais. *Forecast*. Regressão. *Smart meters*.

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

CNN	*Convolutional Neural Network*
DL	*Deep Learning*
GELU	*Gaussian Error Linear Unit*
GPU	*Graphics Processing Unit*
ML	*Machine Learning*
MLP	*Multilayer Perceptron*
MSE	*Mean Squared Error*
SGD	*Stochastic Gradient Descent*
SWL	*Stationary Wavelet Transform*
UFES	Universidade Federal do Espírito Santo

SUMÁRIO

1 INTRODUÇÃO

1.1 Apresentação

A eletricidade é uma das fontes energéticas mais significativas e uma necessidade básica em nossas vidas (ABERA; KHEDKAR, 2020). O uso deste insumo se tornou tão natural e intrínseco ao nosso cotidiano, que é praticamente impossível imaginar a sociedade sem ele.

Segundo a *United States Energy Information Administration* (2018), o consumo mundial deste insumo alcançou 23,398 PWh ao final de 2018, apresentando um crescimento de quase 10% em comparação com 2015. Este avanço, contudo, traz consigo inerentes desafios, como por exemplo garantir qualidade no abastecimento dos consumidores finais. No mesmo cenário de 2018, com cerca de 2,01 PWh perdidos com transporte em redes de distribuição (UNITED STATES ENERGY INFORMATION ADMINISTRATION, 2018), têm-se claro que ainda é possível e necessário melhorar a forma com que energia elétrica é administrada.

Com uma representatividade de aproximadamente 27%, de acordo com a *International Energy Agency* (2018), o setor residencial assume um importante papel no consumo energético global. Com uma parcela de contribuição tão evidente, torna-se natural pensar em ações e projetos que estimulem um gerenciamento mais consciente tanto por parte dos consumidores, quanto por parte das companhias de distribuição.

Um exemplo de política voltada a esta questão pode ser atribuído ao 12º maior consumidor de energia elétrica no mundo, o Reino Unido (UNITED STATES ENERGY INFORMATION ADMINISTRATION, 2018). Essa região cujas representatividades dos setores doméstico e industrial no consumo energético total eram iguais à $30,7\%$ e $17,0\%$ respectivamente em 2013 (DEPARTMENT OF ENERGY & CLIMATE CHANGE, 2014), desenvolveu um programa para instalação de *smart meters* para medirem eletricidade e gás, auxiliando na implantação deste dispositivo em indústrias até 2014 e em residências até 2020 (maiores esclarecimentos acerca do funcionamento deste equipamento são expostos no Anexo A). Segundo Zheng, Gao e Lin (2013), foi estimado pelo governo britânico um custo de aproximadamente 8,6 bilhões de libras na introdução de 26 milhões de medidores cotados a 340 libras cada até o final de 2020. Como retorno deste investimento, em contrapartida, eram esperados 14,6 bilhões de libras em compensações nas despesas operacionais de companhias de energia, além da redução tanto da emissão de poluentes oriundos da geração, quanto do preço das faturas energéticas para consumidores finais mais informados.

As compensações esperadas pelo governo inglês são o reflexo de uma boa administração da rede. Devido às suas características físicas, no caso de falta de equipamentos para

correto armazenamento, a eletricidade deve ser consumida e gerada de forma simultânea (IBRAHIM; ILINCA; PERRON, 2008). Segundo Shahzadeh, Khosravi e Nahavandi (2015), operadores de sistemas de transmissão e distribuição precisam saber quanto de energia será necessário a qualquer horário para que possam se planejar e providenciá-la. Eles ainda concluem afirmando que qualquer erro na precisão desta quantidade acarreta o aumento de gastos para estas companhias, quedas no fornecimento do consumidor final e problemas de confiança.

Monitoramento e aprimoramento da qualidade, maior confiança e eficiência, análise de carga e de falhas, integração com sistemas de cogeração, manutenção preventiva, conectar e desconectar de linhas de forma remota, são apenas alguns dos muitos outros benefícios que podem ser alcançados com uso de *smart meters* (KOPONEN et al., 2008).

Dong et al. (2018) afirmam que é crítico o desenvolvimento de um modelo preciso e eficaz de previsão para consumo energético no esforço para suavizar a demanda do lado do consumidor, todavia, esta mesma conclusão pode ser estendida para todos os demais benefícios destacados anteriormente em junção com os próprios *smart meters*, assim, integrar uma inteligência capaz de retirar percepções e *insights* a partir dos dados e medições obtidos com estes aparelhos se torna uma necessidade no intuito de extrair o máximo potencial deles.

Diversas técnicas já foram usadas na tarefa de previsão de carga, como regressão múltipla, séries estocásticas temporais e modelos de média móvel autoregressiva (ARMA, do inglês *autoregressive mean average*), entretanto, redes neurais artificiais, ou em inglês *Artificial Neural Networks* (ANN), graças às suas camadas internas e habilidade de aprendizado se mostram capazes de resolver este problema (GAJOWNICZEK; ZABKOWSKI, 2014).

Segundo Yan et al. (2019), os métodos atuais de aprendizado profundo (DL, do inglês *Deep Learning*) permitem realizar a previsão de consumo de energia a curto prazo, apresentando predições com uma acurácia notável. Dentro estas técnicas presentes na literatura, existem aquelas que são específicas para determinados tipos de entrada ou dados. Por exemplo, quando o problema está relacionado ao tratamento de séries temporais, as redes de maior destaque são memória de prazo-curto longa (LSTM, do inglês *long short-term memory*) e unidade recorrente seletiva (GRU, do inglês *gated recurrent unit*). Elas são usadas para armazenar informações de natureza sequencial, atualizando-as ao longo do tempo (GREFF et al., 2017).

Outro tipo de camada emprega em DL é a convolucional, cuja aplicação é feita normalmente em problemas com imagens, permite obter características que representam relações

especiais, como, por exemplo, contornos e bordas. No estudo guiado por Kim e Cho (2019), para um determinado conjunto de dados obtidos com *smart meters*, um modelo baseado em redes neurais convolucionais (CNN, do inglês *convolutional neural network*) e LSTM obteve uma métrica de erro quadrático médio, ou traduzido para o inglês como *mean squared error* (MSE), igual a 0,3738.

Apesar de desafiador, a tarefa de previsão de carga demandada é um problema cada vez mais atual e que urge uma solução, tendo em vista a crescente solicitação de energia por todo o mundo (UNITED STATES ENERGY INFORMATION ADMINISTRATION, 2018). Contudo, o aparecimento e consolidação de técnicas de ML que lidam com séries temporais também é notório, deixando claro que existem várias oportunidades a serem desenvolvidas com o objetivo de oferecer opções para um controle de redes de transmissão e distribuição de forma mais assertiva.

Este projeto tem como objetivo comparar o desempenho de diferentes modelos de DL, em uma tarefa que exige processar séries temporais. Tais modelos são selecionados considerando como características: que possam tratar dados de natureza sequencial e se possuem ou não um viés induzido (*inductive bias*). Esta característica faz referência à suposição que um modelo traz consigo, fazendo com que ele priorize funções e características baseado nesta hipótese, em detrimento a outras. Ao final, pretende-se elucidar qual arquitetura prevê com maior exatidão o consumo residencial de eletricidade.

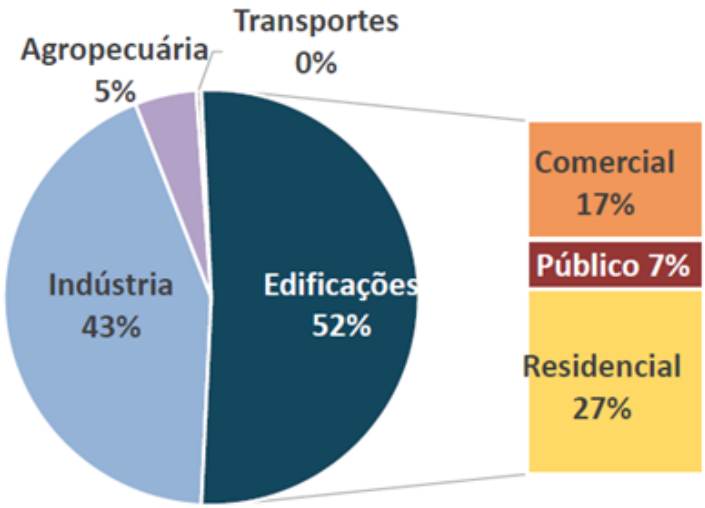
1.2 Justificativa

O Brasil hoje é um dos países que mais consomem energia elétrica em âmbito mundial, classificando-se em oitava posição no *ranking* criado pela *United States Energy Information Administration* (2018). Baseado no Plano Decenal de Expansão de Energia 2030 gerado pelo Ministério de Minas e Energia (MME) e a Empresa de Pesquisa Energética (EPE) (2020), a nação brasileira possuía em 2019 um consumo de 546 TWh e tem a previsão de alcançar um montante de 795 TWh até o final de 2030. Em outras palavras, é esperado um incremento de quase 46% para os próximos 10 anos.

Em comparação com 2018, as participações dos setores no gasto de eletricidade previstas para 2030 são pouco alteradas, destacando ligeiros incrementos no industrial e residencial em contraste ao comercial (BRASIL, 2019; BRASIL, 2020). Os valores correspondentes às parcelas são mostrados a seguir no Gráfico 1.

É visível que estas representações setoriais seguem os moldes observados por todo o mundo (INTERNATIONAL ENERGY AGENCY, 2018) e permitem entender melhor

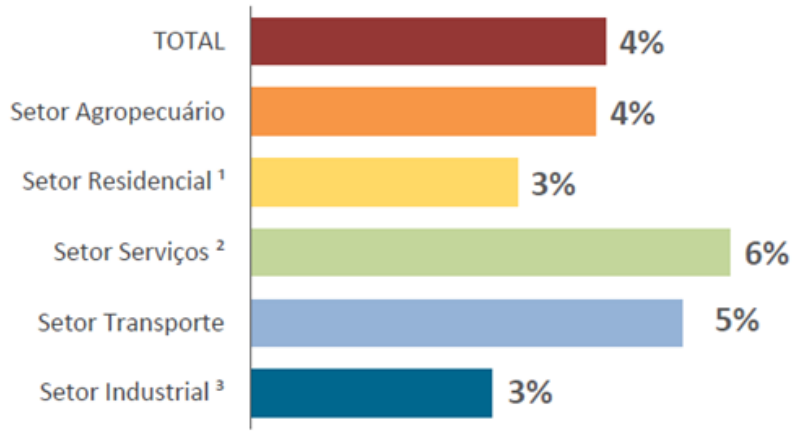
Gráfico 1 – Distribuição do consumo de eletricidade em 2030



Fonte: Ministério de Minas e Energia; Empresa de Pesquisa e Energia (2020).
Nota: Inclui consumo relativo a iluminação pública e saneamento.

as responsabilidades de cada um no consumo de eletricidade. Levando em consideração estas parcelas, também são evidenciadas no Plano Decenal de Expansão de Energia 2030 (Ministério de Minas e Energia; Empresa de Pesquisa e Energia, 2020) as suas contribuições para os ganhos em eficiência elétrica em 2030, comparadas com dados obtidos em 2019. Estes resultados são apresentados no Gráfico 2.

Gráfico 2 – Contribuição setorial para os ganhos de eficiência elétrica em 2030



Fonte: Plano Decenal de Expansão de Energia 2030, segundo Ministério de Minas e Energia; Empresa de Pesquisa e Energia (2020).
Nota: Inclui consumo relativo a iluminação pública e saneamento.
[1] Inclui consumo de energia nos domicílios urbanos e rurais.
[2] Compreende comércio, serviços, público, iluminação pública e saneamento.
[3] Inclui o setor energético.

Dentro deste cenário baseado em previsões, estas contribuições são responsáveis por reduzir o consumo elétrico de 795 TWh para 762 TWh, resultando num abatimento de

aproximadamente 32 TWh ou 4% em 2030 (Ministério de Minas e Energia; Empresa de Pesquisa e Energia, 2020). Este valor é próximo daquele necessário para suprir um trimestre do estado de São Paulo (GOVERNO DE SÃO PAULO, 2021), um dos principais polos econômicos brasileiros.

A fim de que estas previsões de contribuições consigam ser atingidas e até mesmo superadas, é essencial que medidas para tornar mais engenhoso e preciso o gerenciamento de sistemas de distribuição sejam tomadas. Segundo Alahakoon e Yu (2016), a administração de redes inteligentes para entregar energia de forma otimizada necessita de procedimentos avançados de análise de dados para obter informações acuradas e tomar decisões automáticas auxiliando e amparando os operadores em exercício. É mostrado, por meio da literatura, que técnicas de ML são perfeitamente capazes de cumprir com estas exigências.

1.3 Trabalhos Relacionados

O desafio da previsão de carga se trata de um problema de predição de séries temporais, apresentando uma maior complexidade devido à presença de componentes de tendências irregulares em conjunto com um padrão sazonal regular (KIM; CHO, 2019). Além da própria sequência anterior de medições de consumo, outras variáveis podem ser levadas em consideração a fim de encontrar resultados melhores para a solução.

No estudo guiado por Gajowniczek e Zabkowski (2014), foram utilizadas medições de carga nas últimas 24 horas e suas médias, máximas, mínimas e limites do espaço amostral em conjunto com o dia da semana e temperatura momentânea para prever o consumo de cada uma das próximas 24 horas. O modelo utilizado consistia em um *Multilayer Perceptron* (MLP) e conseguiu uma acurácia de 60% e MSE de 0,10 sobre o conjunto de testes, todavia, os autores se queixaram da falta de dados sobre condições climáticas no conjunto utilizado, podendo influenciar no gasto energético dos moradores com televisores ligados em dias chuvosos ou aquecedores em dias frios, por exemplo.

Usando uma solução híbrida composta pela aplicação da transformação de onduleta estacionária (SWT, do inglês *Stationary Wavelet Transform*) sobre um conjunto de dados contendo exclusivamente medições de consumo anteriores alimentando um modelo baseado em LSTM, Yan et al. (2019) conseguiram expressivos resultados para previsões de até 30 minutos. Em seu estudo foi possível a comparação deste método de ML com outros usados para esta saída de curto-prazo, onde a técnica apresentada obteve o melhor desempenho, com MSE de 0,0033 para o maior prazo analisado. Apesar das preciosas conclusões, um ponto negativo deste estudo é justamente a distância temporal da entrada para a saída

não ser superior a meia hora.

Uma abordagem muito interessante para a resolução deste desafio foi apresentada por Kim e Cho (2019), cuja métrica foi anteriormente informada. Aqui, buscou-se prever os próximos 60 valores de medições feitas minuto-a-minuto de um *smart meter* de uma residência, usando 60 minutos anteriores de entrada, contendo valores temporais, médias da potência ativa, reativa, tensão e corrente da casa e energia consumida por 3 diferentes aparelhos domiciliares referentes ao minuto correspondente da amostra. O modelo em si é baseado em CNN-LSTM. Segundo os autores, um ponto de melhoria seria a inserção de característica das residências no conjunto de dados para alimentar a rede e verificar o impacto destas variáveis. Eles complementam ainda dizendo que a quantidade de ocupantes é um fator chave na definição do consumo residencial.

Uma outra análise que foge do comum e que apresenta resultados bastante pertinentes foi realizada por Shahzadeh, Khosravi e Nahavandi (2015). Neste estudo, métodos convencionais de ML foram comparados com testes utilizando diferentes modelos para cada *cluster* (tradução livre, agrupamento). A conclusão foi que, agrupando os dados em conjuntos e desenvolvendo redes específicas para cada um, as métricas gerais eram superiores, podendo chegar a 35%.

Diferentemente, em função da liberdade permitida pelo conjunto de dados, este trabalho pretende avaliar como parâmetros de entrada, como, por exemplo, condições climáticas e financeiras da região de coleta de dados podem influenciar no desempenho do modelo. Paralelamente, serão trabalhadas arquiteturas de ML próprias para tratamento de séries temporais, em especial aquelas formadas por redes convolucionais e recorrentes, avaliando suas performances em predições de até um dia de antecipação.

1.4 Objetivos

1.4.1 Objetivo Geral

Comparar desempenhos entre modelos de inteligência artificial os quais possuem diferenças intrínsecas em suas arquitetura. O contexto em que estes modelos serão confrontados será a predição de consumo elétrico em dispositivos *smart meter*, sendo obrigatório, portanto, que estes sistemas atuem sobre séries temporais.

1.4.2 Objetivos Específicos

No intuito de alcançar o objetivo geral, deve-se atender os seguintes objetivos específicos:

- Definir metodologia de remoção de *outliers*;
- Obter uma lista com possíveis variáveis principais que influenciam o processo;
- Ajustar os hiper parâmetros dos modelos a serem comparados;
- Analisar e comparar os resultados obtidos pelos diferentes modelos.

1.5 Estrutura do Texto

Este projeto está dividido em cinco capítulos, as quais serão elucidadas a seguir.

O Capitulo 1 apresenta um panorama sobre consumo e perdas de energia elétrica em âmbito global e nacional, amparando a importância do estudo que será guiado neste projeto. Além disso, também há a necessidade de expor trabalhos similares desenvolvidos e como eles impactam sobre este.

Aspectos necessários para melhor entendimento do que será desenvolvido nesta pesquisa são esclarecidos no Capítulo 2, como princípios e elementos de ML por exemplo.

No Capítulo 3 são mostrados os métodos e etapas empregadas para o desenvolvendo do projeto. Descrever-se-á o conjunto de dados utilizado, as análises realizadas e os problemas encontrados sobre o mesmo e as ações tomadas para alimentar essas informações ao modelo.

Os resultados obtidos são relatados no Capítulo 4, comparando as diferentes abordagens apontadas na seção anterior.

Por fim, no Capítulo 5, é discutido aquilo que for obtido como saída dos modelos. São confrontados desempenhos de cada um, apontando quais apresentaram resposta superior para a tarefa em questão. Outrossim, são comentadas as dificuldades encontradas durante o trabalho e possíveis pontos a serem melhorados ou acrescidos em estudos posteriores.

2 REFERENCIAL TEÓRICO

Nesta seção são apresentados conceitos teóricos necessários para o entendimento e execução deste trabalho, principalmente os assuntos referentes a DL. DL é explicado na seção 2.1. Já em sequência, as arquiteturas tradicionais são expostas e descritas na seção 2.2. Na seção 2.3 é explicado o conceito de *inductive bias*. Por fim, a seção 2.4 introduz as arquiteturas modernas que serão utilizadas neste projeto.

2.1 *Deep Learning*

Segundo Zhang, Han e Deng (2018), DL é um subconjunto de ML, que foi obtido a partir da integração de múltiplas ANN. Uma ANN tem a capacidade de aprender padrões importantes intrínsecos nos dados e que não precisam de complexas regras matemáticas. Isso acontece devido principalmente à habilidade das redes em gerar resultados não-lineares e à interação entre os seus neurônios. À medida que são empilhadas mais camadas de *perceptrons*, a interconexão entre eles se torna ainda mais forte e, consequentemente, a não linearidade do modelo se torna ainda mais presente. Desta forma, o campo de DL permitiu que resultados antes inalcançáveis pudessem ser obtidos.

O significado de profundo em DL é um pouco diferente do conhecimento popular. Neste caso, este termo faz referência à quantidade de camadas empilhadas no modelo. Quanto mais estruturas forem acopladas, mais "profundo" ele é.

2.2 Tipos de Camadas em DL

Dentro DL, existem vários tipos de camadas, cada uma com qualidades e aplicações específicas. Esta seção irá percorrer algumas delas.

2.2.1 Camadas densas

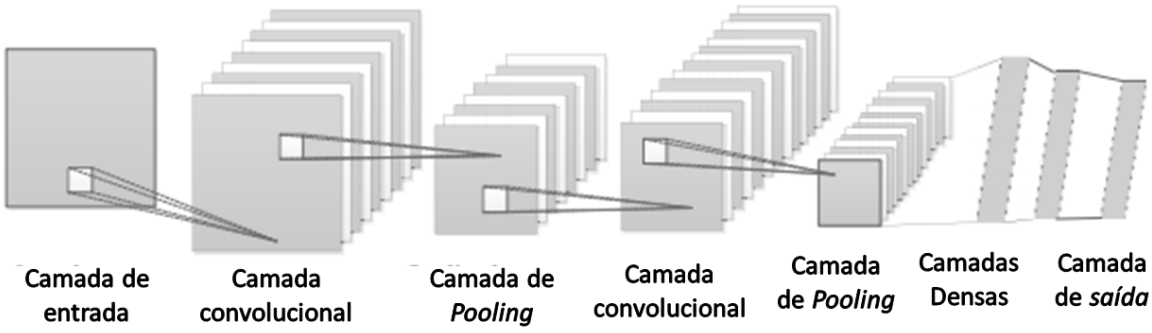
Também denominadas como *fully-connected* ou totalmente conectadas, estas camadas são ANN cujas unidades se ligam a todos as unidades de nível acima delas. Isso permite que todas as características obtidas anteriormente por outras estruturas consigam ser compreendidas e analisadas nas camadas mais baixas da rede, onde o processo de decisão está mais próximo.

2.2.2 Camadas convolucionais

Camadas convolucionais são especializadas na aquisição de peculiaridades e padrões espaciais aos dados. Para fazer isso, ela dispõe da utilização de filtros de tamanho fixo que realizam operações de convolução sobre toda a sua entrada gerando como saída um mapa de características por cada filtro. Diferentemente daqueles apresentados em áreas de estudo de processamento de imagem, os filtros empregados aqui não possuem pesos predeterminados. Ao contrário, estes valores são validados e aprimorados à medida que o modelo é treinado. Como são criados vários filtros, cada um acaba se aperfeiçoando na detecção de um determinado padrão presente na entrada, de modo que quando atuam em união, vários desses atributos são encontrados e a rede consegue entender o objeto ou informação que os originou.

Adaptada pelo autor, a Figura 1 foi apresentada por Zhiqiang e Jun (2017) e ilustra uma estrutura CNN e o funcionamento de camadas convolucionais, em conjunto com outras camadas, como de agrupamento (do inglês, *pooling*) e densas.

Figura 1 – Estrutura CNN típica



Fonte: Zhiqiang e Jun (2017).
Nota: Adaptado pelo autor.

Apesar de serem amplamente utilizadas em problemas envolvendo imagens por se mostrarem aptas a detectar pontos, linhas e faces, além de preservar as relações entre *pixels* da imagem aprendida (KIM; CHO, 2019), estas camadas vêm ganhando cada vez mais destaque em aplicações de séries temporais. Isso acontece devido à sua facilidade em encontrar características, que é usada para obter tendências temporais nos dados, eliminando delas o ruído e permitindo que os atributos principais possam ser entregues e memorizados pelas

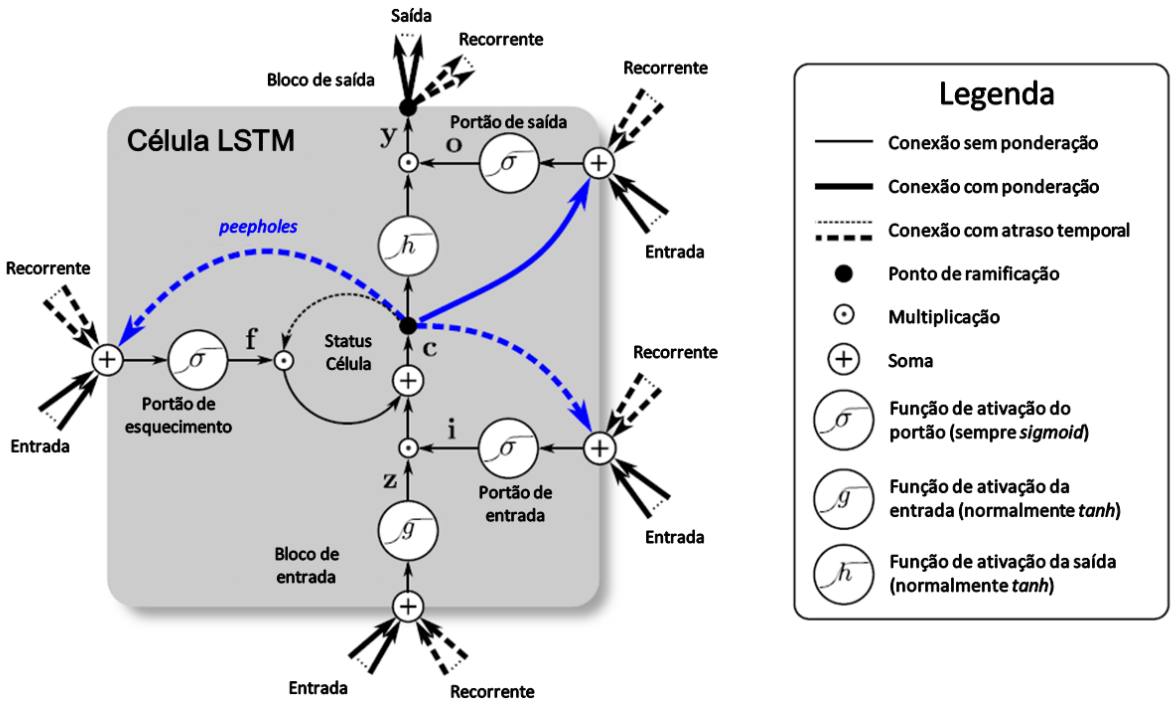
camadas recorrentes seguintes da rede.

2.2.3 Camadas LSTM

As camadas LSTM herdam a propriedade especial das redes neurais recorrentes (RNN, do inglês *recurrent neural networks*) de considerar as entradas como séries temporais interconectadas (YAN et al., 2019), tornando-as particularmente eficazes em problemas de dados sequenciais. Esta habilidade está atrelada à estrutura única utilizada por elas.

Apesar das diversas variações, a estrutura principal do modelo comumente usada como referência é aquela apresentada pela *Vanilla* LSTM. Segundo Greff et al. (2017) está LSTM apresenta: três portões (de entrada, de esquecimento e de saída), um bloqueio de entrada, um conjunto de estados da célula e uma função de ativação na saída. Posteriormente, a essa composição foram adicionadas as conexões *peephole* (tradução livre, olho mágico), geralmente aplicadas em sinais de caráter ritmado ou compassado. Toda essa estrutura permite com que padrões vistos durante do treinamento sejam "guardados", como uma forma de memória primitiva, pelos estados internos da célula que influenciam diretamente no resultado do conjunto. Como as entradas são consideradas séries temporais, informações referentes tanto aos estados internos quanto às saídas anteriores são ponderadas no cálculo atuais. Um esquema representativo da LSTM é mostrado na Figura 2.

Figura 2 – Esquema descrevendo uma célula LSTM



Fonte: Greff et al. (2017).
Nota: Adaptado pelo autor. A etiqueta recorrente se refere a saídas anteriores. A multiplicação dos vetores é feita elemento a elemento.

Para explicar o funcionamento destas células, é necessário enfatizar primeiramente que a entrada do modelo é a soma das entradas no instante analisado com as suas saídas anteriores. Conforme mostrado na Figura 2, a primeira ação a ser realizada é a atualização dos estados internos da célula, representados por c, portanto, as entradas são transformadas através do portão e bloco de entrada e do portão de esquecimento, mostrados respectivamente por i, z e f. Para os portões, usa-se a função de ativação sigmoide enquanto para o bloco de entrada utiliza-se outras como a tangente hiperbólica, ilustradas na Figura 2 como σ e g. Isso é feito desta forma, porque o alcance de valores possíveis para o gradiente para entradas normalizadas da sigmoide é menor que aquele existente para a tangente hiperbólica. Como a atualização de pesos internos de um modelo é feita consoante ao cálculo do gradiente, funções com um gradiente menor possuem uma atualização de pesos mais lenta, permitindo assim que estes valores sejam "esquecidos" mais vagarosamente. Desta forma, os parâmetros dos portões são mais resistentes a mudanças, enquanto aqueles relacionados aos blocos mudam mais facilmente. Os valores assumidos pelos portões e blocos podem ser representados matematicamente pelas Equações (2.1), (2.2) e (2.3) e os estados internos pela Equação (2.4), usando-se as entradas e as saídas recorrentes respectivamente como x_t e h_t, onde o subscrito t é o instante analisado no cálculo (x_t ocorre no momento atual e x_{t-1} um instante antes). Vale ressaltar que w e b representam pesos e bias, correspondentes ao subscrito de cada portão ou bloco.

$$i_t = \sigma(w_i[h_{t-1}, x_t] + b_i) \tag{2.1}$$

$$f_t = \sigma(w_f[h_{t-1}, x_t] + b_f) \tag{2.2}$$

$$z_t = g(w_z[h_{t-1}, x_t] + b_z) \tag{2.3}$$

$$c_t = f_t \odot + z_t \odot i_t \tag{2.4}$$

O bloco introdutório é então moldado pelo portão de entrada através de uma multiplicação elemento a elemento e, posteriormente, somado ao portão de esquecimento estabelecido pelos estados internos anteriores da célula. Este último componente é essencial para que memórias precedentes sejam ponderadas, quando novos parâmetros estiverem sendo criados. Ao final deste somatório, os estados internos são atualizados, transformados pela função de ativação de saída (descrita por h) e prontamente multiplicados elemento a elemento com os dados de entrada da célula transformados por uma função de ativação, que originam o portão de saída descrito na Figura 2 pela letra o. O resultado dessas operações são as saídas, descritas por y, que serão realimentadas à esta estrutura de forma recorrente na próxima iteração. Matematicamente, pode-se descrever estes procedimentos conforme as Equações (2.5) e (2.6).

$$o_t = \sigma(w_o[h_{t-1}, x_t] + b_o) \tag{2.5}$$

$$y_t = o_t \odot h(c) \tag{2.6}$$

Outro ponto presente na Figura 2 que demanda explicação são os *peepholes*. A sua necessidade surgiu a partir da carência do controle dos portões, apontado por Gers e Schmidhuber (2000) para que a célula aprendesse a cronometrar medições temporais de forma mais precisa. Para isso, conexões dos estados internos para os portões foram adicionadas à estrutura, e a elas foi dado o nome de *peepholes* (GREFF et al., 2017).

Aliado a tudo que foi elucidado sobre redes LSTM, ainda há comentários a serem feitos com relação aos estados internos dessas redes. Enquanto o modelo receber sequências de dados referentes ao mesmo *batch* (traduzido do inglês como lote, correspondendo a um agregado de dados), os estados internos são atualizados e contribuem para a saída da camada. A partir do momento em que o *batch* acaba, um novo é preparado para ser alimentado ao modelo e, antes disso acontecer, o modelo reinicia seus estados internos, retornando-os a um valor nulo. Isso é crucial durante o treinamento, pois a quantidade de sequências que afetarão as saídas graças aos estados internos é ditada pelo tamanho do *batch*, ou seja, quantas sequências existem dentro dele. Este parâmetro deve ser meticulosamente estudado, ao verificar o desempenho de uma rede.

2.3 *Inductive Bias*

Viés induzido, do inglês *inductive bias*, como o próprio nome já diz, é uma tendência forçada que o modelo busca ou segue, devido a uma imposição de pressupostos. Um objetivo de qualquer sistema de inteligência artificial é se tornar generalista e ser capaz de prever cenários baseados em dados que não tenham sido previamente analisados. Segundo Goyal e Bengio (2020), não existe um algoritmo de aprendizado para propósitos gerais e portanto, para obter essa tão necessária generalização, é necessário que o modelo tenha algumas preferências. Estas opções são exatamente o *inductive bias*.

Um exemplo deste conceito é uma regressão linear. Nela, uma variável é representada supondo-se uma relação descrita por uma equação do primeiro grau entre ela e uma segunda característica.

Outros casos em ML onde este conceito pode ser facilmente identificado, são nas redes convolucionais e recorrentes. Os filtros usados em CNNs supõem que podem encontrar padrões contidos dentro dos pontos contemplados por cada multiplicação dentro da

convolução, ou seja, que estão suficientemente próximos para serem reconhecidos pelo filtro. Já nas redes recorrentes se presume que possam extrair padrões de sequencias temporais já que tem mecanismos específicos para decifrá-las.

Em contrapartida, existem também modelos que não assumem este tipo de pressuposição, como por exemplo as camadas Densas. A única medida desta arquitetura é se conectar com todos os nós da camada anterior e configurar os pesos associados a estas conexões, permitindo a melhor descrição do fenômeno a ser aprendido.

O exposto é resumido no Quadro 1.

Quadro 1 – Explanação de arquiteturas com e sem *inductive bias*

Arquitetura	*Inductive bias* correspondente
Camadas Densas	Não possui
Camadas Convolucionais	Padrões identificados dentro de uma região ou espaço
Camadas Recorrentes	Padrões identificados em uma sequencia temporal

Fonte: Produção do próprio autor.

2.4 Arquiteturas Modernas de DL

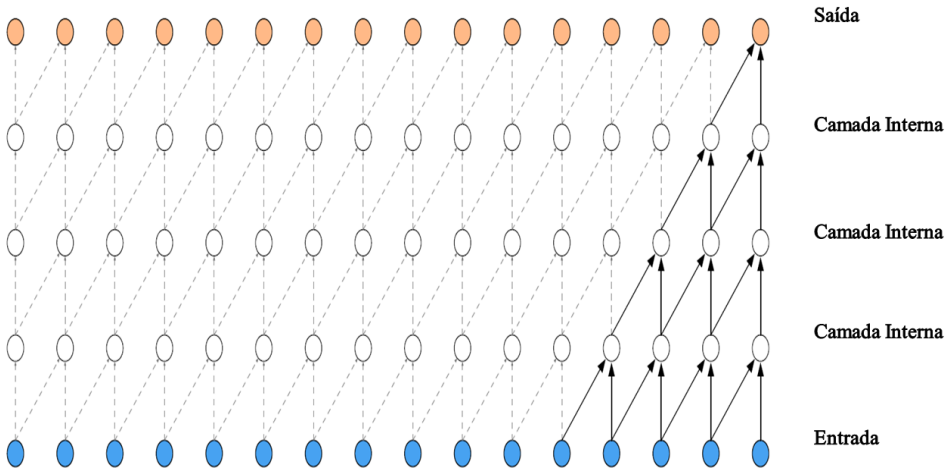
Outras duas arquiteturas menos tradicionais serão descritas nesta seção. É interessante ponderar que ambas são aplicadas no projeto, tendo-se em vista que uma não apresenta um viés induzido por ser formada basicamente de estruturas densas, enquanto a outra sim. Apesar disso, ambas possuem aplicação em dados sequenciais, sendo a primeira aplicada no processamento de áudio, e a segunda sendo projetada para lidar com sequencias temporais de longa duração. Estas redes são, respectivamente, a WaveNet e a MLP-Mixer.

2.4.1 WaveNet

A WaveNet é uma rede que usa camadas convolucionais empilhadas como base para alcançar seus objetivos. Esta rede introduzida por Van den Oord et al. (2016) está constituída de camadas causais e dilatadas. A primeira garante que amostras futuras não sejam usadas na realização de predições, o que acarretaria um vazamento de dados, como pode-se observar na Figura 3.

A segunda característica permite que está arquitetura consiga processar sequências extensas, possibilitando que toda a entrada possa influenciar na saída, este efeito é chamado de aumento do campo receptivo da saída. Para alcançar este objetivo, a rede Wavenet usa

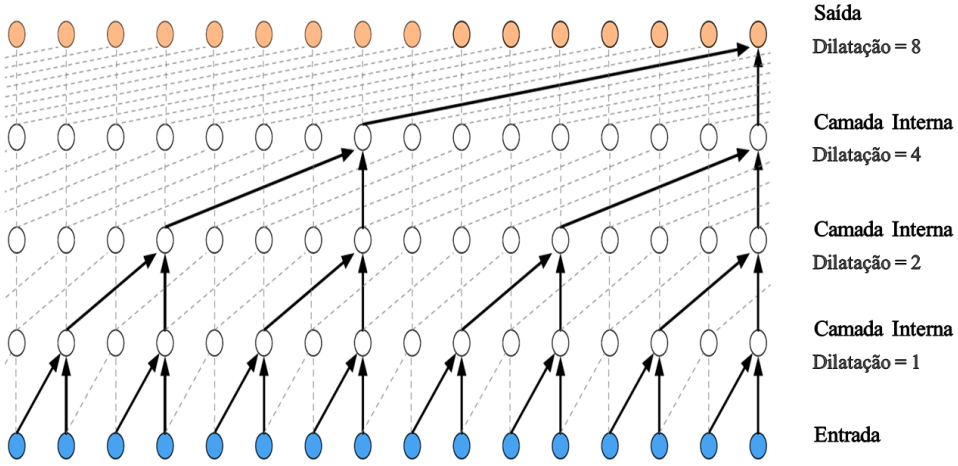
Figura 3 – Ilustração do efeito de camadas causais



Fonte: Van den Oord et al. (2016).
Nota: Traduzido pelo autor.

camadas dilatadas, cuja função é aplicar os filtros de maneira não sequencial, mas pulando um determinado número de elementos de entrada, tal valor é denotado como a taxa de dilatação. Como este número aumenta exponencialmente com o número de camadas, rapidamente todas as entradas conseguem ser analisadas e refletidas para a saída do modelo. Para o exemplo da Figura 3, com 5 camadas apenas 5 entradas eram observadas, mas agora com a mesma quantidade, todas são contempladas usando-se a dilatação, como fica evidente pela Figura 4.

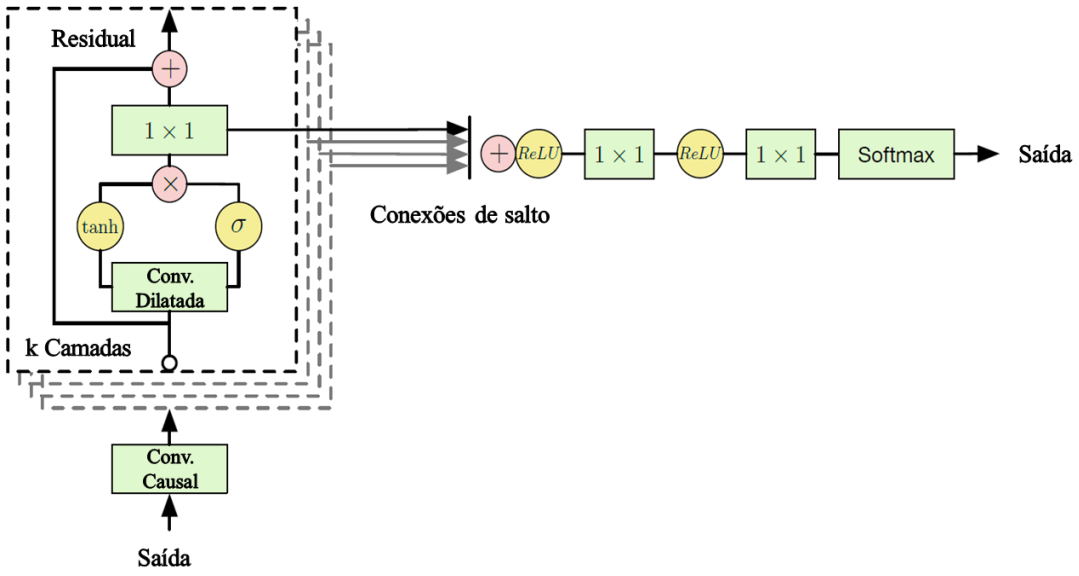
Figura 4 – Ilustração do efeito de camadas de dilatação



Fonte: Van den Oord et al. (2016).
Nota: Traduzido pelo autor.

Além desses pontos, a rede WaveNet ainda conta com outras três técnicas para melhorar seu rendimento, mostradas na Figura 5 e explicadas a seguir.

Figura 5 – Ilustração da estrutura da rede WaveNet



Fonte: Van den Oord et al. (2016).
Nota: Traduzido pelo autor.

A primeira técnica é a função de ativação de bloqueio (do inglês, *gated activations*), que da mesma forma que nas redes recorrentes, usa-se funções de ativação com um gradiente menos agressivo, fazendo com que se tenha uma certa forma de memória na camada, criando uma resistência à atualizações dos pesos de forma abrupta dentro da camada WaveNet. Esta operação está representada na Figura 5 logo após a camada convolucional dilatada pela letra σ. É notório que o sinal que é processado pela função sigmoide é operado tanto pela função sigmoide quanto pela *tanh*. Isso acontece porque a primeira desempenha este papel de lembrança, enquanto a segunda percebe modificações que ocorrem instantaneamente.

A segunda técnica são as conexões de salto, ou *skip-connections* do inglês, este tipo de conexão permite que as saídas de todas as camadas WaveNet possam ser recuperadas e consideradas ao final da rede. Isso é feito baseado na ideia de que resultados intermediários produto de processamentos iniciais também tem valor e devem ser analisados e, por isso, não podem ser meramente descartados. Vale ressaltar que as saídas de todas as camadas são consideradas ao final da rede, não apenas ao final de seu respectivo bloco.

Por fim, a ultima técnica são as conexões residuais, ou *residual connections* do inglês. Elas funcionam como uma espécie de conexões de salto curto, agem adicionando a entrada de uma camada à sua própria saída. Isto faz com que a os dados inseridos não sejam demasiadamente alterados por cada camada e auxilia um fluxo mais direto dos gradientes na retropropagação do erro.

Com os componentes de camadas explicados, é necessário salientar que as mesmas são

empilhadas dentro de um bloco e, cada vez que ocorre esta sobreposição, há um aumento exponencial da taxa de dilatação. Por exemplo, se a primeira camada possui uma taxa de dilatação igual a 1, a segunda terá 2, a terceira e a quarta 4 e 8, respectivamente, e assim por diante. Cada bloco formado por estes empilhamentos, deve-se começar sempre com uma taxa de dilatação 1.

Além de amontoar camadas para formar blocos, também se mostra uma pratica comum empilhar estes conjuntos, reiniciando-se as taxas entre estes agrupamentos para formar por completo a rede WaveNet. Esta ação acaba gerando um processamento ainda maior e mais meticuloso da sequência de entrada.

2.4.2 MLP-Mixer

A rede MLP-Mixer foi apresentada no artigo desenvolvido por Tolstikhin et al. (2021) da equipe *Google Research Brain Team*. O fato interessante desta arquitetura é que ela é baseada apenas em camadas densas e é capaz de produzir resultados comparáveis com os modelos da atualidade. No estudo, a rede MLP-Mixer é comparada com outras arquiteturas como CNN e arquiteturas baseadas em mecanismos de atenção, que apresentavam os melhores desempenhos no momento em que a pesquisa foi realizada. Para o conjunto de dados ImageNet-21k, o desfecho do estudo é mostrado na Tabela 1, onde "HaloNet" é um modelo baseado em mecanismos de atenção, "ViT-L/16" é um *Vision Transformer*, "BiT-R152x4" são CNNs e "Mixer-L/16" é uma instancia de uma rede MLP-Mixer. É importante indicar que o "Mixer-L/16" obtive estes resultados até três vezes mais rápida que seus concorrentes.

Tabela 1 – Comparação desempenho da rede MLP-Mixer com demais redes competitivas

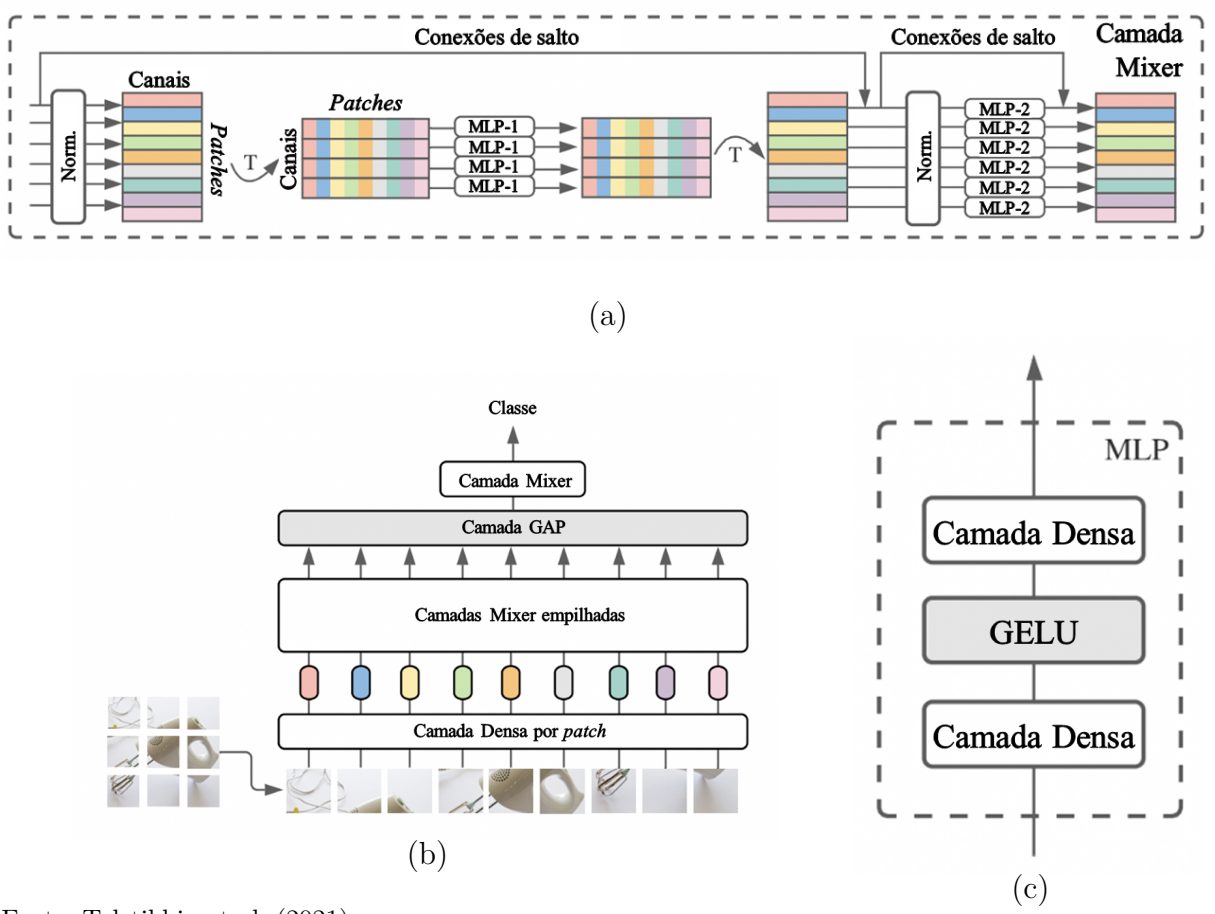
Modelos	Acurácia para validação do conjunto ImageNet
HaloNet	85,8%
Mixer-L/16	**84,15%**
ViT-L/16	85,30%
BiT-R152×4	85,39%

Fonte: Produção do próprio autor.

A estrutura MLP-Mixer que será descrita nesta sessão é mostrada na Figura 6.

O pipeline da rede MLP-mixer, mostrado na Figura 6(b), inicia com a divisão da entrada em diversos fragmentos não sobrepostos, denominados *patches*. Cada *patch* da entrada é alimentado a uma camada densa especifica, e as saídas destas camadas densas são alimentandas a um bloco Mixer, que é constituído por varias camadas Mixer. A saída do

Figura 6 – Ilustração da estrutura da rede MLP-Mixer



Fonte: Tolstikhin et al. (2021).
Nota: Traduzido pelo autor.

bloco Mixer tem a dimensão reduzida por meio de uma camada GAP (*Global Average Pooling*, traduzindo para agrupamento médio global), seguida de uma camada Mixer, que retorna a saída final da rede MLP-Mixer.

A estrutura interna de uma camada Mixer, mostrada na Figura 6(a), inicia com a normalização, vetorização e empilhamento dos *patches*, formando uma matriz, onde as linhas correspondem aos *patches* vetorizados e as colunas são os canais. Esta matriz é então transposta e as linhas resultantes são alimentadas à camada de pesos compartilhados, MLP-1. O resultado da operação anterior é então novamente transposto, voltando à dimensão original, e somado à entrada através de conexões de salto. Em seguida, a matriz resultante da soma é normalizada novamente e suas linhas são alimentadas à camada de pesos compartilhados, MLP-2. O produto desta interação é novamente somado ao valor da matriz antes da segunda normalização e, assim, obtém-se a saída para uma camada Mixer. Cabe indicar que cada camada de pesos compartilhados (MLP-1 ou MLP-2), está constituída por uma camada densa de entrada com função de ativação GELU (Unidade linear de erro gaussiano, do inglês *Gaussian Error Linear Unit*) e outra camada densa de

saída, como mostrado na Figura 6(c).

3 PROPOSTA

3.1 Visão Geral

O trabalho em questão possui o intuito de comparar diferentes arquiteturas de DL, confrontando principalmente as que possuem *inductive bias* com aquelas que não, para um problema envolvendo séries temporais. Ao fazer isso, busca-se validar a ideia de que as suposições derivadas da escolha de um modelo têm a capacidade de generalizar e, consequentemente, tornar mais assertiva uma solução quando apresentada a novos dados. Um exemplo deste fenômeno é o uso de redes neurais recorrentes, o qual parte do pressuposto que existem relações temporais entre os registros que serão aprendidas através do treinamento.

Fato é que a literatura mostra cada vez mais que modelos sem *inductive bias* podem apresentar desempenhos competitivos com outras arquiteturas específicas. Segundo Goyal e Bengio (2020), para conjuntos de dados enormes, a vantagem das soluções que apresentam este viés pode ser menor quando comparadas com aquelas que não tem. Isso acontece porque, ao priorizar determinadas características, o sistema rejeita outras. Um exemplo de modelo que utiliza majoritariamente redes densas e tem apresentado resultados expressivos é a rede MLP-Mixer. O estudo feito por Tolstikhin et al. (2021) e seus resultados apresentados na Tabela 1 comprovam que arquiteturas sem *inductive bias* podem ser concorrentes daquelas que possuem. Este feito será da mesma forma verificado no presente projeto.

O problema que validará os resultados distintos de cada abordagem será a predição do consumo energético residencial. Para esta tarefa, serão utilizados mais de dois anos de dados de medidores residenciais de energia inteligentes em conjunto com informações climáticas da cidade em que estavam operando. Também será requerido que os modelos prevejam todas as medições realizadas pelo dispositivo até uma pré-determinada quantidade de horas a frente, usando uma quantidade de amostras passadas também previamente fixada e apresentada na Seção 4.3.3. Mesmo assim, o modelo a ser usado se classifica como muitos-para-um (do inglês, *many-to-one*), pois a saída da sua última camada possui apenas uma amostra temporal com um número de características igual ao número de medições posteriores que serão previstas.

As redes confrontadas serão compostas por camadas densas, LSTM, Conv1d e blocos MLP-Mixer e WaveNet. Cada uma terá seus hiperparâmetros otimizados através de uma biblioteca em Python específica para otimização de funções, que buscará a melhor configuração para elas. Esta busca é feita procurando o conjunto de parâmetros que retornará a menor métrica do modelo, que será representada pelo erro médio quadrado (MSE, do inglês *Mean Squared Error*). Por fim, estas estruturas serão combinadas em

busca do melhor resultado possível.

3.2 Pré-processamento

Para que os pesos do modelo possam surtir maior efeito sobre os dados que a ele são apresentados, é uma prática comum transformar os valores de entrada, de forma que estes fiquem próximos de 0. É ilustrado em Ayyadevara (2019) que esta operação tem a capacidade de reduzir consideravelmente o tempo de aprendizado exigido, tendo em vista que ajustes finos nos parâmetros internos do modelo surtem um efeito mais perceptível.

Existem diversas transformações na literatura e uma das mais utilizadas envolve aplicar uma transformação que escala os elementos de acordo com a média e desvio padrão observados. Para esta operação ser utilizada, é feita a suposição que os dados sobre os quais ela será aplicada estão distribuídos de forma normal, porém, isso não é sempre uma verdade, visto que as mais variadas distribuições podem ser encontradas quando lidamos com informações reais.

Para resolver esta problemática, existem técnicas cuja função é aproximar a distribuição do conjunto o máximo possível para uma normal. Uma delas foi desenvolvida pelos estatísticos George Box e David Roxbee Cox, que a apelidaram de transformação BoxCox, onde as operações realizadas são apresentadas na Equação 3.1.

$$y(\lambda) = \begin{cases} \frac{y^\lambda - 1}{\lambda}, \ se\ \lambda \neq 0 \\ \log y, \ se\ \lambda = 0 \end{cases} \tag{3.1}$$

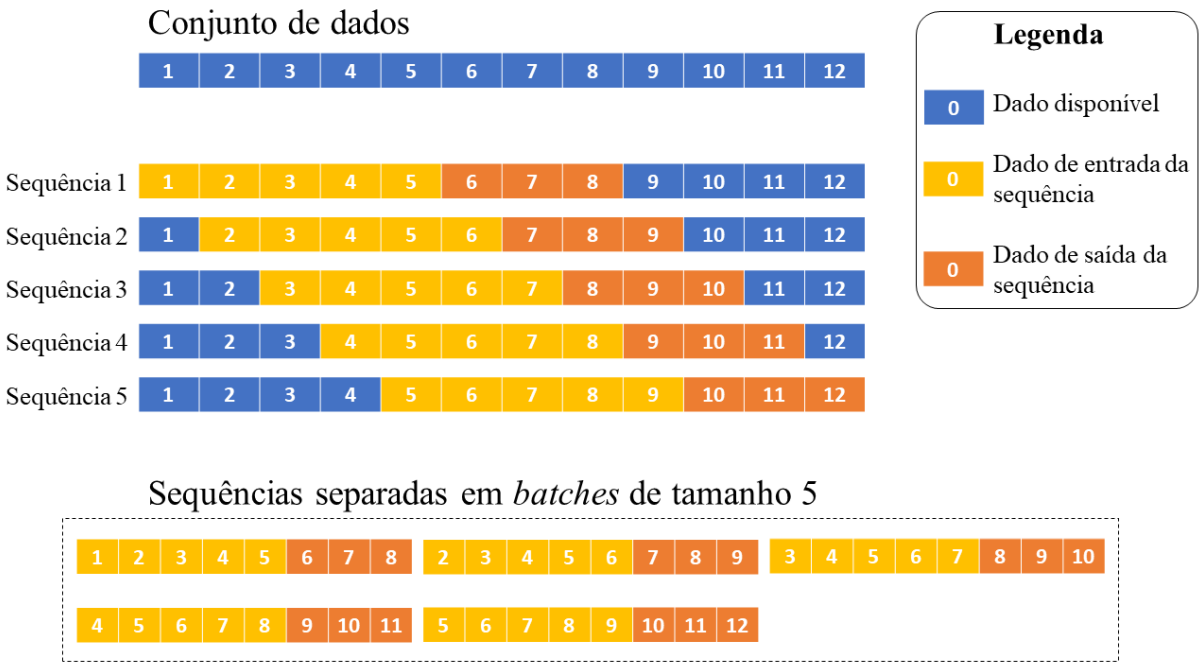
Por meio de algoritmos computacionais, é possível inferir o parâmetro *lambda* que gera a distribuição mais próxima da normal e, com ele em mãos, realiza-se a transformação. Esta busca é feita somente sobre os dados de treino, para que os de validação ou teste não influenciem no resultado, entretanto, todos estes conjuntos são transformados pela mesma função.

Assim que o método de BoxCox for aplicado sobre todos os dados, estes podem passar também por uma etapa de escalamento (do inglês, *scaling*), fazendo com que os dados estejam distribuídos segundo uma normal padrão.

Uma consideração extra que deve ser levada sempre em conta ao se trabalhar com redes neurais recorrentes é a organização dos dados de forma sequencial. Isso deve ser feito porque estes sistemas possuem estados internos que "memorizam" características não apenas entre

registros temporais de uma sequência, como também entre os *batches* (traduzido do inglês como lote), sendo reiniciados apenas após percorrer todas suas amostras (BROWNLEE, 2017). Por conseguinte, as entradas do modelo devem ser agrupadas em pacotes de sequências, onde estas estão ordenadas serialmente. Um exemplo para esta técnica é ilustrado na Figura 8, para o caso em que 5 amostras temporais são usadas como entrada e 4 tentam ser previstas como saída do sistema. Além disso, o *batch* possui um tamanho igual a 5, ou seja, engloba 5 sequências.

Figura 7 – Exemplo de criação de *batches* de sequências



Fonte: Produção do próprio autor.

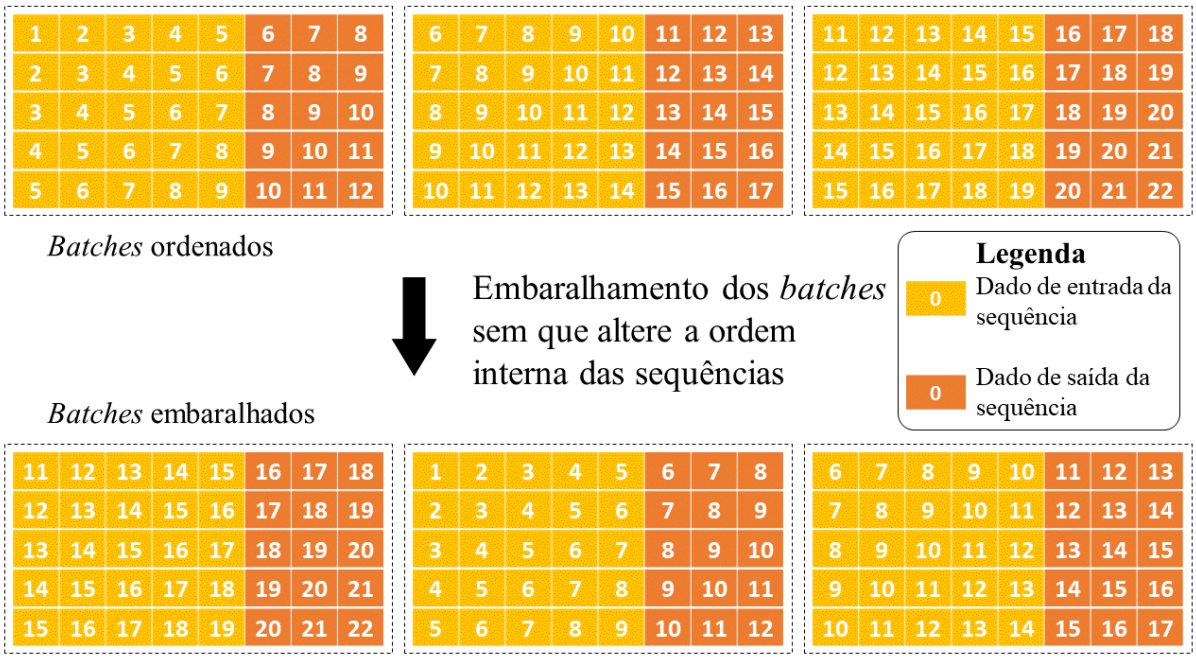
Como os estados internos são reiniciados entre diferentes *batches*, existe a possibilidade de se embaralhar estes diferentes pacotes de sequências sem que prejuízo seja gerado para o treinamento do modelo, desde que seu conteúdo interno não seja misturado. Um exemplo do que é feito, é mostrado na Figura 8.

3.3 Construção dos modelos

A pesquisa será realizada levantando os desempenhos de diferentes modelos integrados a duas classificações: aqueles que possuem *inductive bias* e aqueles que não o possuem.

Para o primeiro agrupamento, pode-se notar uma quantidade menor de tipos de camada pois a ele são incluídos a *fully-connected*, envelopada de forma a tratar registros temporais diferentes separadamente, e a camada MLP-Mixer. É interessante destacar que a segunda

Figura 8 – Exemplo de embaralhamento de *batches*



Fonte: Produção do próprio autor.

componente deste grupo é composta a partir da primeira, ou seja, basicamente apenas estruturas densas.

Já a coleção composta por modelos que carregam viés por natureza, validará a efetividade de camadas LSTM, Conv1D e WaveNet para solução de problemas com séries temporais. Estes modelos já são tidos como apropriados para este tipo de desafio.

Após a realização dos testes, a fim de obter melhores resultados, serão combinadas as camadas previamente citadas, em conjunto com inserção de camadas que auxiliem no combate ao *overfitting* e, para o caso de redes recorrentes, que ajudem a mitigar o efeito de gradientes explosivos ou minguantes (do inglês, *exploding* ou *vanishing gradients*).

3.4 Otimização de hiperparâmetros

Para que cada rede possa alcançar o melhor desempenho possível, existe a necessidade de validar diferentes possibilidades de características e verificar quais estão apresentando resultados mais consistentes. Esta tarefa será realizada através de um otimizador de hiperparâmetros que dá a liberdade de, por meio de várias tentativas, especificar quais escolhas feitas pelo construtor da arquitetura fazem mais sentido para os dados processados.

4 RESULTADOS

4.1 Recursos Computacionais

4.1.1 Recursos de *Software*

As etapas de treinamento e teste da arquitetura proposta foram realizadas utilizando Keras com *backend* baseado em Tensorflow, *software* de código aberto desenvolvido pela *Google brain Team*, destinado à programação numérica utilizando programação baseada em fluxo de dados em grafos (ABADI et al., 2016). O fato de ser um *framework* gratuito e possuir suporte a utilização de GPUs contribuem para que esta ferramenta seja amplamente empregada em problemas de ML e DL, acelerando o treinamento de redes.

Outro *framework* cuja aplicação é necessária neste trabalho é o Optuna, amplamente aplicado em otimização de hiperparâmetros, implementando estratégias eficientes de busca e descarte de características através de um campo amostral de opções configurado pelo usuário (AKIBA et al., 2019). Toda a programação da pesquisa foi realizada através da linguagem Python.

4.1.2 Recursos de *Hardware*

Para o desenvolvimento dos experimentos desta pesquisa foram utilizados dois recursos de *hardware* diferentes. O primeiro é o computador pessoal do autor, cujas configurações são as seguintes: (*i*) sistema operacional Windows, versão Windows 10; (*ii*) processador Intel Core i7-9750 CPU, 2,60 GHz com 6 núcleos físicos; (*iii*) memória RAM de 16 GB; (*iv*) unidade de armazenamento de 500 GB (SSD); (*v*) placa de vídeo NVIDIA GeForce GTX 1660 Ti, com 6 GB de memória dedicada. O outro recurso utilizado é a plataforma *Google Colaboratory*. Esta ferramenta permite ao usuário acesso às máquinas virtuais da empresa por um tempo de uso limitado. Existem as versões gratuitas e pagas, onde as duas se distinguem quanto à tempo de sessão e ao acesso de maior memória RAM e GPUs mais rápidas. Quanto a estes dois últimos recursos, independente de qual assinatura for feita, o usufruidor não os tem disponíveis conforme desejar, mas sim quando a companhia liberar o seu uso. Este controle é feito através de uma fila de prioridades, onde os melhores recursos (melhores GPUs e RAM) são mais difíceis de serem liberados por terem um acesso mais restrito.

As opções de GPUs, por ordem de desempenho e restrição, são K80, T4 e P100, onde apenas a primeira está disponível de forma livre. Para parte deste estudo foi realizada a aquisição do pacote pago, cujas especificações pode ser descritas à seguir, levando em consideração que a utilização da GPU dependia da prioridade que o autor possuía segundo o algoritmo da ferramenta: (*i*) sistema operacional Linux, versão Ubunto 18.04.5 LTS;

(*ii*) processador Intel(R) Xeon(R) CPU @2.20 GHz; (*iii*) memória RAM de 24.4 GB; (*iv*) unidade de armazenamento de 300 GB (HD); (*v*) placa de vídeo K80, com 12 GB de memória dedicada, ou T40 ou P100, com 16 GB de memória dedicada.

4.2 Conjunto de dados

As informações que são utilizadas para desenvolvimento deste projeto se originam no conjunto de dados *Smart Meter Energy Consumption Data in London Households*, composto pelas medições de consumo energético de 5567 residências londrinas, entre novembro de 2011 e fevereiro de 2014, durante o projeto *Low Carbon London*, guiado pela *UK Power Networks* (2014). A intenção deste estudo era validar se os hábitos de consumo dos clientes da rede de distribuição da cidade britânica mudariam caso a forma com que fossem cobrados mudasse. Para isso, o governo instalou estes dispositivos em residências de diferentes classes econômicas e, para algumas dessas unidades, alterou a tarifa sobre a qual era cobrada.

O conjunto de dados possui especificação sobre a segmentação geodemográfica da unidade em que o *smart meter* é instalado, segundo a instituição *Consolidated Analysis Center, Incorporated* (CACI) *Internacional* (UK Government, 2020). Esta classificação abrange 18 categorias, que podem ser agregados em 3 grupos principais.

As medições registradas pelos dispositivos são quantidades de energia consumida a cada meia hora, armazenadas e disponibilizadas pelo conjunto de dados, totalizando 1,55 GB. Estas informações são dispostas por medidor e ao longo do seu tempo de ativação. Não há no conjunto de dados, entretanto, uma informação de quantas pessoas habitam a residência, o que é uma falta importante.

Devido ao fato dos dados terem sido coletados através da plataforma Kaggle (MICHEL, 2016), o desafio correspondente também disponibilizava dados referentes às condições climáticas da cidade em que foram instalados estes dispositivos, obtidos através da interface programável de aplicações (API, do inglês *application programming interface*) da *darksky* (APPLE, 2021). As informações meteorológicas estão disponíveis em intervalos horários e diários.

Para validar o modelo com diferentes conjuntos de dados, tanto para treinamento, quanto validação e testes, os *smart meters* registrados foram divididos usando a técnica de *K-fold* estratificado com $K = 4$. Além disso, ficou definido que aproximadamente 5% dos dispositivos seriam destinados para o conjunto de teste, 20% para validação e o restante (75%) para o treinamento do modelo.

4.3 Particularidades da Implementação

Nesta sessão serão apresentadas algumas técnicas adotadas para este conjunto de dados em particular, de forma que permitisse a realização deste projeto e que ele obtivesse os melhores resultados.

4.3.1 Quanto aos dados meteorológicos

Incluído no conjunto de dados dos medidores inteligentes, encontra-se relatórios horários e diários das condições meteorológicas da cidade britânica, notificando informações como temperatura, umidade, pressão, fase da lua e ventos, por exemplo.

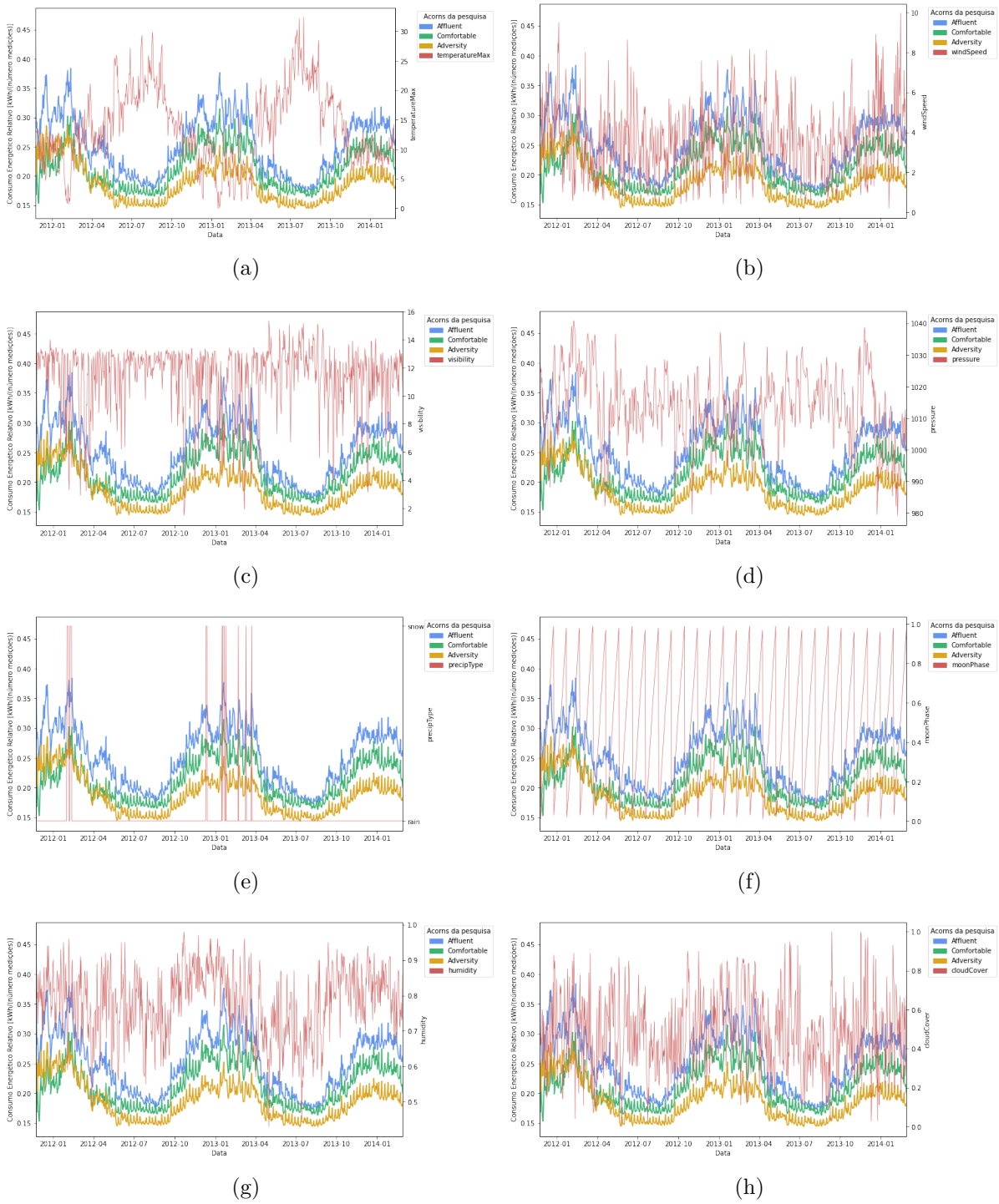
A partir destas características, foi necessária a validação das mesmas, afim de informar se alguma se relacionava com os dados de consumo. Para tanto, foram comparados o consumo diário relativo dos 3 agrupamento principais com cada uma dessas propriedades climáticas. É importante pontuar que este consumo diário relativo é calculado dividindo-se a soma da energia medida num dia por todos os *smart meters* pela soma da quantidade de medições por eles realizada. Os principais resultados são mostrados em sequência na Figura 9.

Como se pode observar na Figura 9, a variável que apresenta uma correlação mais clara com o consumo de energia é justamente a temperatura. A relação entre estas duas características se revela ainda mais quando apresenta-se um gráfico disperso entre as duas, conforme a Figura 10. Outras relações, como por exemplo o indicativo de precipitação de neve, que se relacionavam de certa forma com a curva de consumo pareciam também ser descritas pela variável da temperatura. Por este motivo, apenas esta tempérie foi escolhida para ser levada em consideração e também ser alimentada ao modelo.

Também pela Figura 9(a), pode-se notar que tanto a temperatura como o consumo possuem uma sazonalidade. Foi julgado importante, portanto, repassar como entrada o momento em que estes dados estavam sendo inseridos, através da sazonalidade da temperatura. Para fazer isso, foi necessário descrever este sinal por meio das suas componentes principais, de modo que a sua sazonalidade ficasse evidente, lançando mão da Transformada de Fourier. Como esta operação é sensível aos sinais ruidosos, foi feita uma média móvel com tamanho de uma semana e *stride* de mesmo tamanho, cujo resultado é mostrado na Figura 11(a) na curva em azul escuro.

Este sinal amortizado pela janela deslizante foi então reconstruído descartando-se as frequências que estivessem abaixo do limiar testado. Alguns valores de corte são mostrados na Figura 11(b) juntamente com os sinais resultantes no domínio do tempo. Para o andamento do estudo, foi escolhido o limiar de valor 80 representado pela curva azul.

Figura 9 – Curvas das características climáticas junto com consumo relativo: (a) Temperatura máxima, (b) Velocidade do vento, (c) Visibilidade, (d) Pressão, (e) Tipo de Precipitação, (f) Fase da lua, (g) Umidade e (h) Cobertura de nuvens

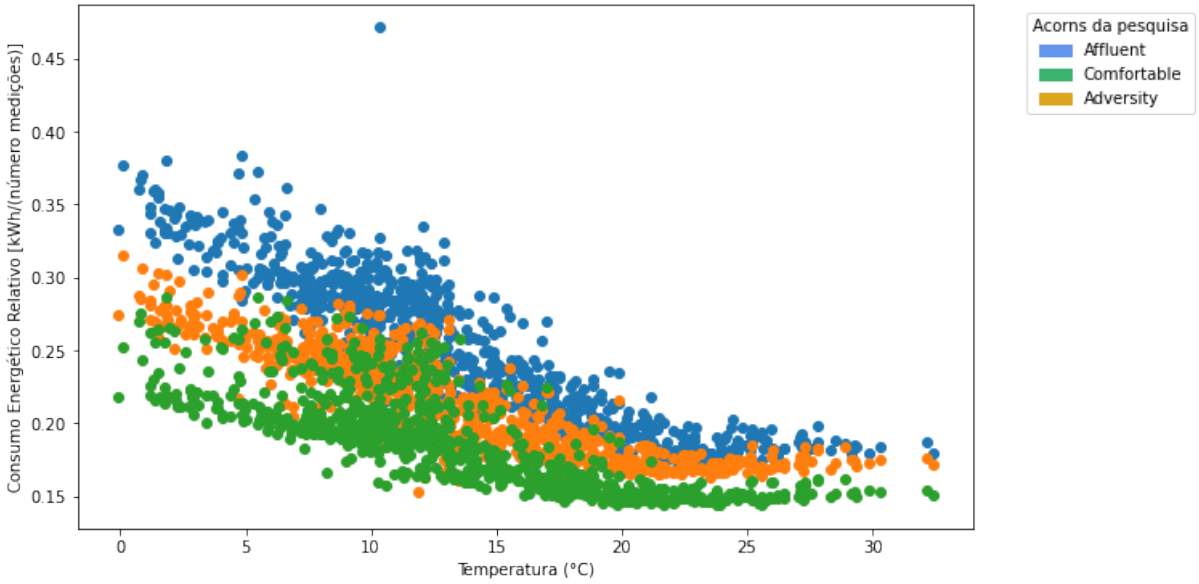


(a) (b) (c) (d) (e) (f) (g) (h)

Fonte: Produção do próprio autor.

Em conjunto com a temperatura semanal reconstruída com as frequências principais, ao invés da inserção dos dados originais, o que poderia levar a uma duplicação de informação nessas duas colunas, foi inserida a diferença entre a temperatura natural da semanal, ou

Figura 10 – Gráfico de dispersão entre Consumo relativo e Temperatura



Fonte: Produção do próprio autor.

seja, a oscilação da temperatura em torno da média semanal. Por fim, estes valores foram normalizados antes de serem alimentados ao modelo.

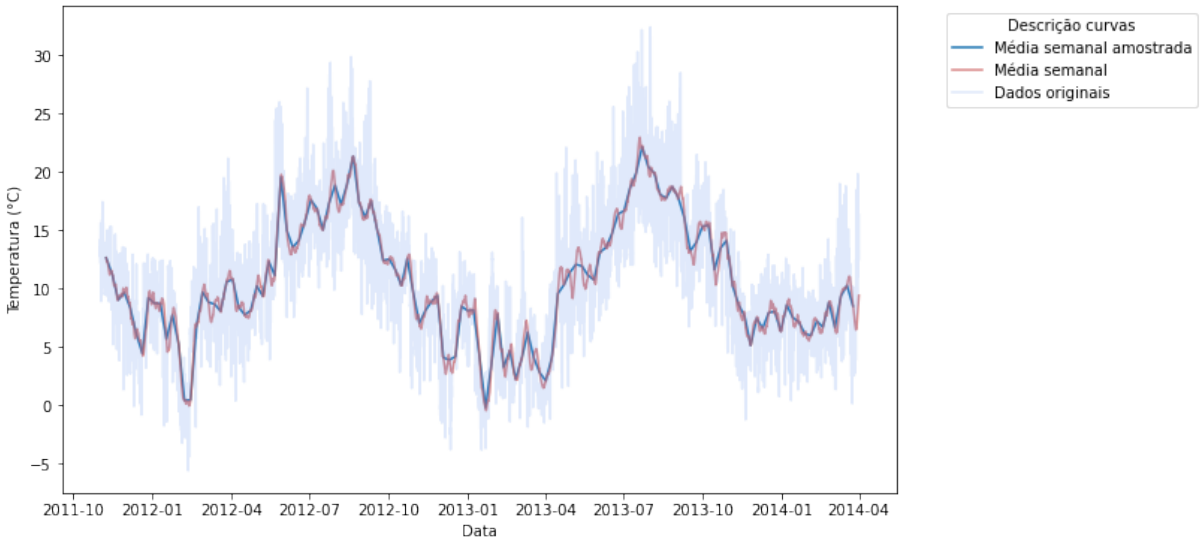
4.3.2 Quando aos dados de consumo energético

Conforme comentado na Seção 4.2, o programa para analisar o consumo de clientes da rede de distribuição londrina surgiu para avaliar se o comportamento destes usuários seria afetado quando houvesse mudança sobre a tarifa cobrada. Como não é interessante para o presente projeto obter variações induzidas no comportamento de consumo aferido pelos *smart meters*, foi analisado o impacto da remoção dos dispositivos localizados em residências cuja mudança de arrecadação aconteceu. A comparação entre moradias que mantiveram e que mudaram a forma que pagam energia elétrica é mostrada na Figura 12(a), onde aqueles classificados como 'Std' são os medidores com tarifa padrão, enquanto os categorizados com 'ToU' são aqueles que sofrem a alteração. Aliado a isso, também foi averiguado se haviam diferenças de consumo entre os dias da semana. Esta comparação também é mostrada na Figura 12(b).

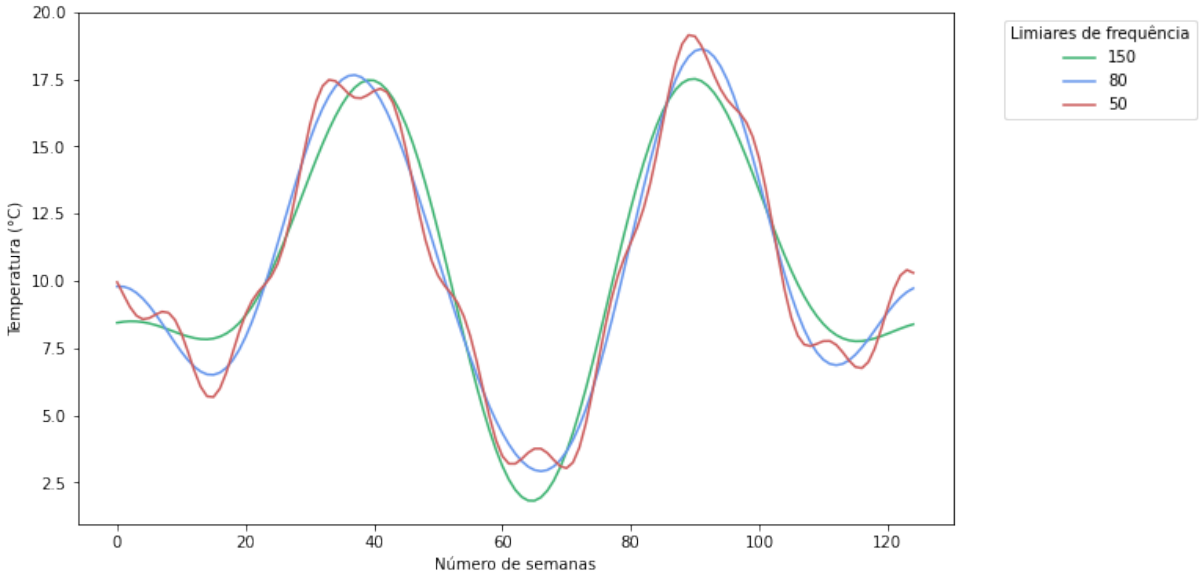
A partir das imagens da Figura 12, pode-se afirmar que o descarte dos medidores cuja tarifa é modificada não impacta fortemente sobre os experimentos desta pesquisa.

Outra conclusão tomada foi que aparentemente os dias de segunda-feira, sábado e domingo apresentam um consumo maior do que os demais dias da semana. Por isso, também será

Figura 11 – Curvas relacionadas à temperatura sobre o tempo: (a) Temperatura original e após operações com janela deslizante e (b) Reconstrução do sinal de temperatura semanal com frequências principais acima de um limiar



(a)



(b)

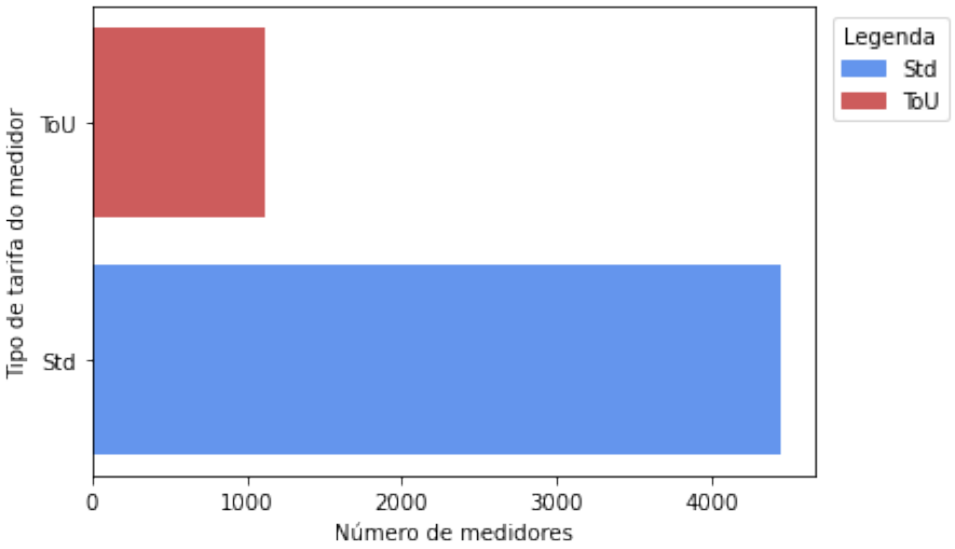
Fonte: Produção do próprio autor.

entrada do modelo se o registro a ser lido pertence ou não a um desses três dias.

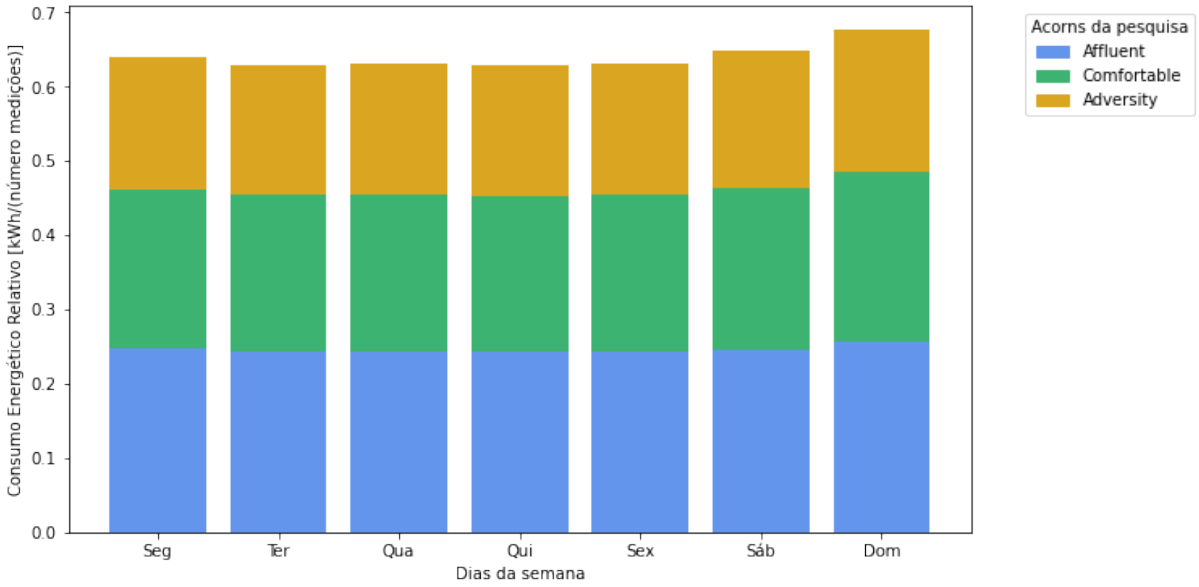
4.3.3 Quanto aos dados de entrada em geral

Em função da limitação de memória RAM e GPU do computador pessoal do autor, nem todos os dados puderam ser alimentados ao modelo. A intenção do projeto de verificar o desempenho do modelo para *smart meters* de diferentes agrupamentos geodemográficos

Figura 12 – Ponderações sobre os dados de consumo: (a) Comparação entre quantidade de medidores mudando para a tarifa não convencional e mantendo e (b) Consumo acumulado relativo por dia de semana



(a)



(b)

Fonte: Produção do próprio autor.

deve de ser cancelado e apenas uma categoria foi escolhida para os testes, que é a '*Affluent*'. Dessarte, o objetivo principal de validar as arquiteturas de ML entre redes com e sem *inductive bias* pôde ter sequência.

Outra consequência da falta de recursos de *hardware* foi a necessidade de reduzir o número de *smart meters* nos conjuntos de treino, validação e teste. Para o grupo '*Affluent*' haviam

1.702 medidores, onde cada *fold* teria 1.276 dispositivos para treino, 339 para validação e 87 para testes. Para que os testes pudessem ser desenvolvidos na máquina pessoal do autor, estes números mudaram para 200, 50 e 50, respectivamente. Isso não é o ideal, pois com mais dados o modelo teria mais exemplos para buscar, aprender e generalizar padrões.

Por fim, é necessário informar que todos os experimentos tentaram prever 5 horas de consumo com antecedência, ou seja, exigiu-se do modelo prever 10 registros. Para isso, usou-se como base 12 horas, ou 24 registros, passados como sequências de entrada. Cada registro temporal será composto pelas seguintes informações: consumo energético do medidor, se o dia da semana é segunda-feira, sábado ou domingo, a temperatura horária e a temperatura semanal correspondente à medição. As séries formadas a partir desses registros terão um tamanho igual a 24, correspondendo a 12 horas de informações, e serão alimentadas ao modelo que retornará 10 registros, que são 5 horas, como previsão. Estas sequências serão inicialmente agrupadas em *batches* de tamanho 256. A Figura 13(a) ilustra os tamanhos das entradas, das saídas esperadas e dos *batches* iniciais para este projeto, além de apresentar o tamanho das sequências previstas pelo modelo. Já na Figura 13(b), pode-se observar com mais detalhes quais são as variáveis usadas na tarefa de previsão e o que é por fim previsto.

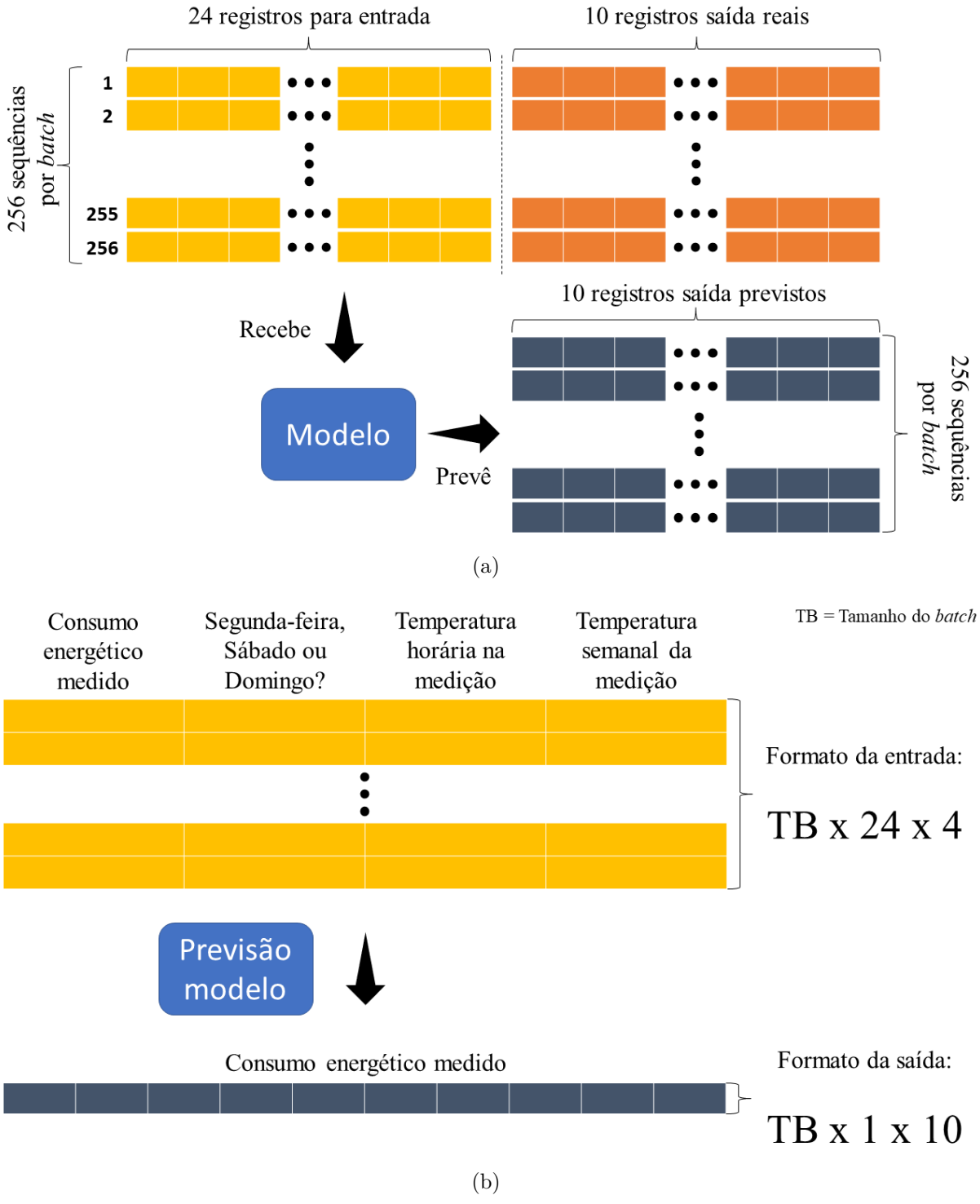
Vale observar ainda na Figura 13(b) que usam-se 24 registros temporais e 4 características como entradas, enquanto a saída apresenta 10 valores do consumo energético, que serão previstos pelo modelo. Entretanto, a última camada do modelo não é recorrente, mas sim uma camada densa. Isso fará com que estas 10 previsões não sejam distribuídas uma para cada instante temporal, mas 10 para o mesmo instante. Isso é evidenciado pelo formato do vetor resultante Por este motivo, o modelo não pode ser considerado como um sistema *many-to-many*, mas sim *many-to-one*.

4.4 Experimentos

4.4.1 Métricas

Para que as diferentes redes pudessem ser comparadas sobre uma mesma base, foi empregada a métrica erro quadrático médio (MSE, do inglês *Mean Squared Error*) sobre os dados transformados por BoxCox e padronizados do conjunto de teste do *fold* analisado e os valores de saída do modelo. Para validar os valores reais preditos pelo sistema, basta convertê-los através da despadronização da sua distribuição (usando média e desvio padrão observados nos dados originais) e a transformação inversa de BoxCox, todavia, o uso dos dados transformados, ou originais, não impacta em nada nas análises realizadas. Isso acontece porque qualquer que seja a base de dados usada, ambas devem ser minimizadas na busca do valor ótimo, a única diferença é que os dados estão em uma escala e distribuição

Figura 13 – Dados inseridos no modelo de ML: (a) Tamanho das séries de dados e *batches* e (b) Características consideradas para a previsão e resultado do modelo com seus formatos



Fonte: Produção do próprio autor.

diferentes.

Todas as redes implementadas nesta pesquisa foram submetidas a uma bateria de testes

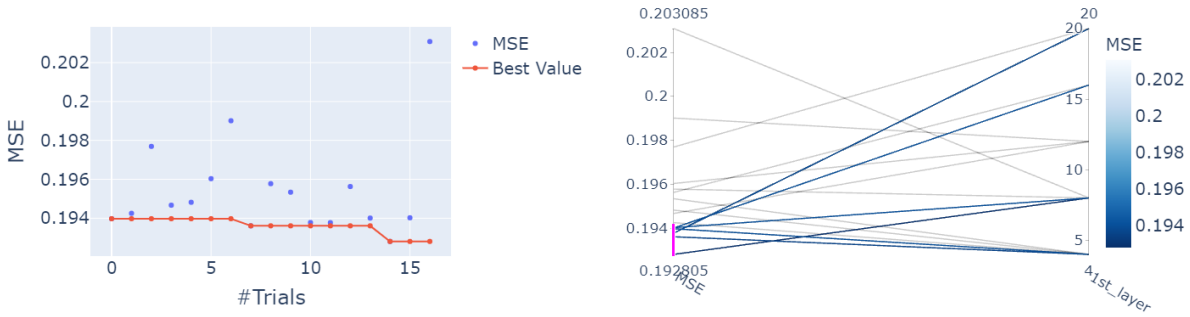
guiados pelo *framework Optuna*. Por meio desta aplicação, foi possível definir os melhores parâmetros para configurar cada um dos modelos e analisar quais foram as métricas observadas para cada tentativa. Note que estas especificações experimentadas por rodada deviam ter o seu espaço de opções determinado pelo usuário. Naturalmente, cada arquitetura possui seus fatores específicos, portanto, o que for configurado para o *framework* entre os modelos irá variar.

4.4.2 Modelos sem *inductive bias*

4.4.2.1 Modelo *Fully-Connected*

Para os modelos *Fully-Connected* acoplados a envelopadores *TimeDistributed*, foi testada com uma camada antes da final. Este modelo, apesar de simples, também servirá como *baseline* para os experimentos mais complexos. A estrutura desta rede é mostrada no Anexo B, na Tabela 4. No Gráfico 3, são destacadas por meio das linhas em azul as configurações que obtiveram as melhores pontuações para o algoritmo de otimização, em outras palavras, a menor métrica à esquerda e, à direita, o histórico de pontuações das tentativas. Estas linhas ligam os escores à configurações aplicadas na correspondente tentativa.

Gráfico 3 – Tentativas feitas para o modelo *Fully-Connected*: (a) Histórico de tentativas e (b) Pontuação por parâmetro escolhido



(a) (b)

Fonte: Produção do próprio autor.

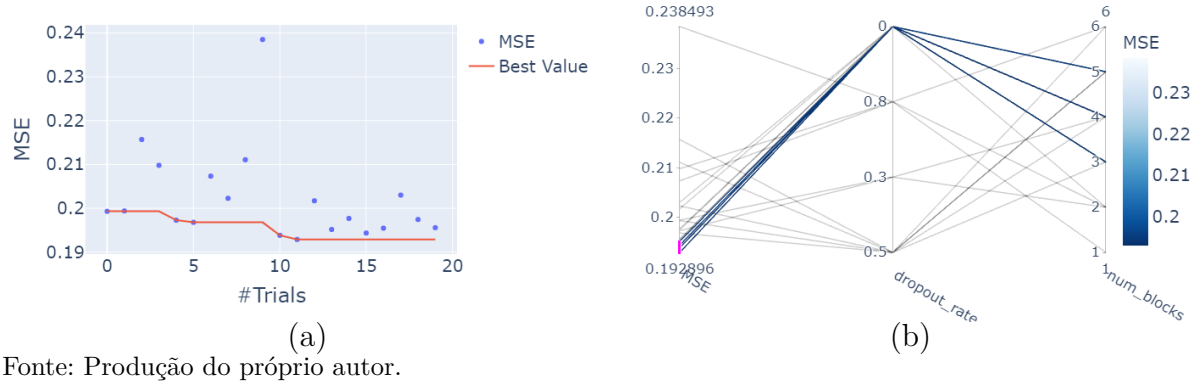
É perceptível que as melhores marcas foram obtidas por camadas com menos nós, por exemplo 5 e 10, que atingiu uma pontuação de $0,192805$. Isso pode indicar que a solução do problema não dependa de algo extremamente complicado, por não haver relações demasiadamente complexas.

4.4.2.2 Modelo baseado na estrutura MLP-Mixer

O segundo e último modelo sem *inductive bias* é o MLP-Mixer. Para seu desenvolvimento, foi fixada uma largura de *patch* de 4 e para as opções dos parâmetros para a *framework*,

foi dada a taxa de *dropout* (denominada 'dropout_rate') e a quantidade de blocos MLP-Mixer(chamada de 'num_blocks'). A estrutura desta rede é mostrada no Anexo B, na Tabela 5. Os resultados obtidos à partir desta configuração são mostrados no Gráfico 4.

Gráfico 4 – Tentativas feitas para o modelo MLP-Mixer: (a) Histórico de tentativas e (b) Pontuação por parâmetro escolhido



(a) (b)

Fonte: Produção do próprio autor.

Duas coisas chamam a atenção no estudo deste modelo. Primeiramente, todas as melhores tentativas usaram uma taxa de *dropout* nula e, quanto maior este valor, piores eram os resultados. Segundo, com um número de blocos igual a 4, o melhor ensaio teve uma pontuação de $0,192896$, que é menor que o obtido com o modelo totalmente conectado, apesar da maior simplicidade deste.

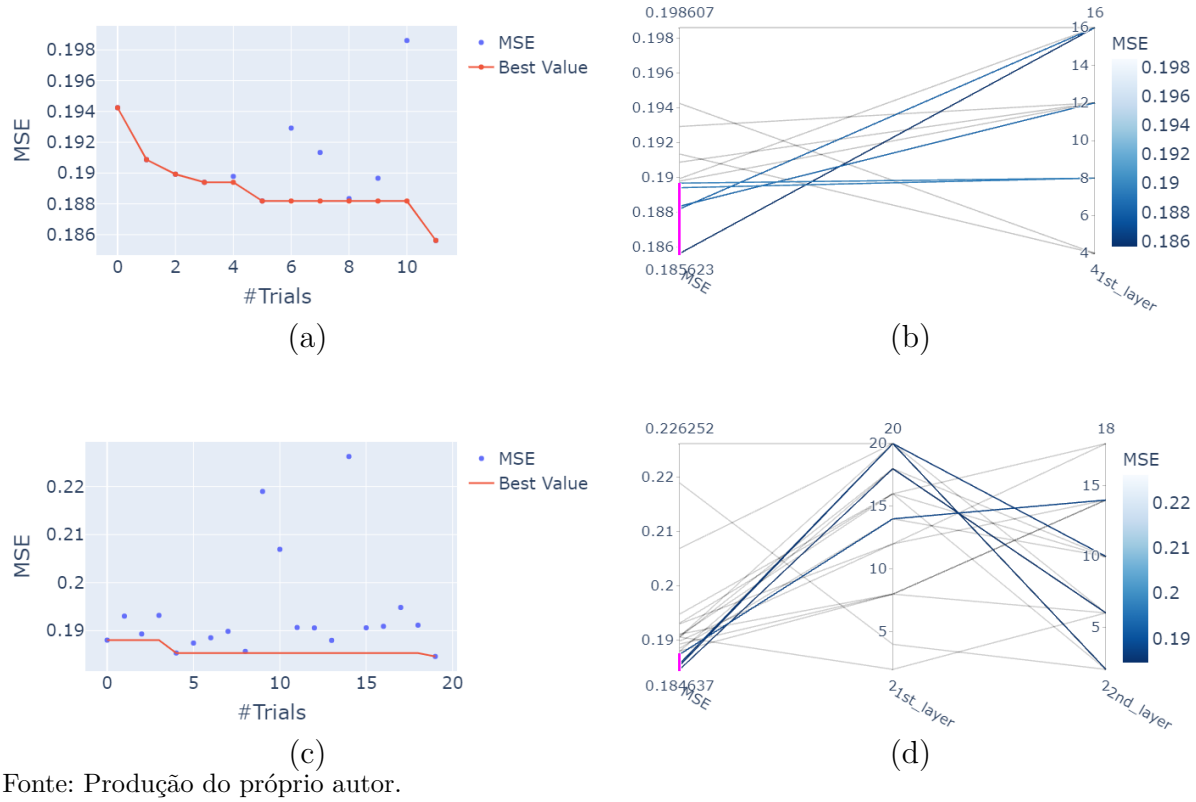
Outro fator que influencia negativamente esta arquitetura é que, conforme exposto por Tolstikhin et al. (2021), ela possui grande capacidade de evolução do aprendizado, quando apresentado à quantidade maiores de dados. Em virtude da limitação do recurso de *hardware*, não foi possível inserir os dados de todos os 1286 medidores separados para treinamento, mas apenas 200. Caso mais dados fossem introduzidos, haveria a possibilidade do modelo ficar mais genérico e, portanto, com melhores resultados.

4.4.3 Modelos com *inductive bias*

4.4.3.1 Modelo baseado em LSTM

Uma das primeiras escolhas, ao se pensar em desafios envolvendo séries temporais, é o uso de redes recorrentes, devido à sua capacidade de memorizar características através dos registros. Na experiência feita com esta camada, foi testada uma arquitetura com uma camada simples e outra com duas empilhadas para estudos diferentes. Como opções para o *framework*, foram dadas as unidades de cada uma dessas camadas. A estrutura desta rede é mostrada no Anexo B, na Tabela 6. Os resultados podem ser observados segundo o Gráfico 5.

Gráfico 5 – Tentativas feitas para o modelo LSTM: (a) Histórico de tentativas para versão com 1 camada LSTM, (b) Pontuação por parâmetro escolhido para versão com 1 camada LSTM, (c) Histórico de tentativas para versão com 2 camadas LSTM e (d) Pontuação por parâmetro escolhido para versão com 2 camadas LSTM



Fonte: Produção do próprio autor.

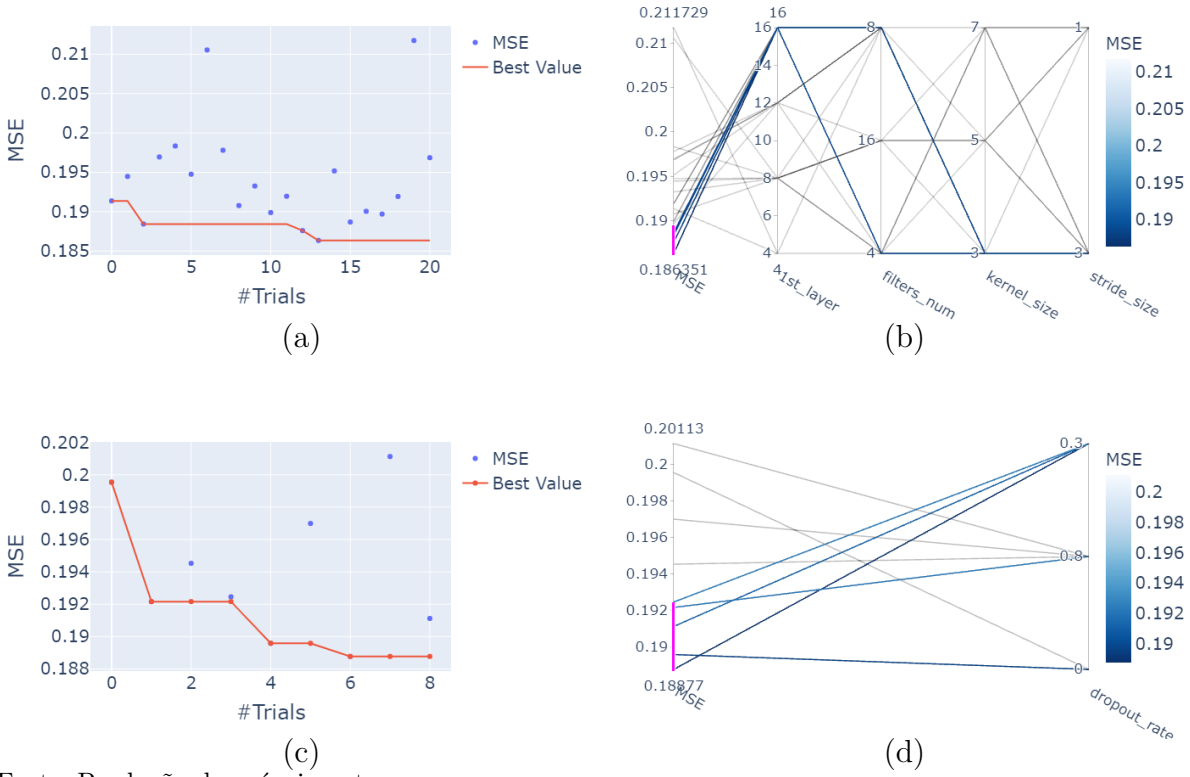
Pode-se notar claramente nos Gráficos 5(a) e 5(b) que camadas com mais unidades internas pontuam mais. Isso é ainda mais evidente para rede de apenas uma camada, cujo melhor resultado é obtido com o número máximo de nós 16, obtendo uma métrica de $0,185623$. É interessante comentar que uma rede recorrente profunda alcançou um resultado melhor que a arquitetura simples, atingindo $0,184637$ de MSE para 18 unidades na primeira camada e 6 na segunda.

4.4.3.2 Modelo baseado em Conv1d e LSTM

Baseado na premissa de realçar detalhes e bordas e, posteriormente, memorizá-las, foi testada uma arquitetura baseada em ambas as camadas Conv1d e LSTM, nesta ordem e em duas versões. Na primeira, apenas uma componente LSTM foi usada, deixando como opções para o *framework* a quantidade de unidades desta camada, e as opções de quantidade de filtros, tamanho do *kernel* e saltos destes filtros (*stride*) da Conv1d, ou seja, quatro tipos de parâmetros. Já a segunda versão tinha apenas um parâmetro de escolha opcional para o algoritmo do Optuna, que era a taxa de *dropout*. Isso foi feito porque nesta versão foram usadas duas componentes LSTM com a mesma configuração usada

pelo melhor modelo até então, descrito na Seção 4.4.3.1 e as mesmas especificações da melhor tentativa da versão anterior para a componente Conv1d. A estrutura desta rede é mostrada no Anexo B, na Tabela 7. Os resultados sobre as tentativas realizadas são mostrados no Gráfico 6.

Gráfico 6 – Tentativas feitas para o modelo baseado em Conv1d-LSTM: (a) Histórico de tentativas para versão com 1 camada LSTM, (b) Pontuação por parâmetro escolhido para versão com 1 camada LSTM, (c) Histórico de tentativas para versão com 2 camadas LSTM e (d) Pontuação por parâmetro escolhido para versão com 2 camadas LSTM



Fonte: Produção do próprio autor.

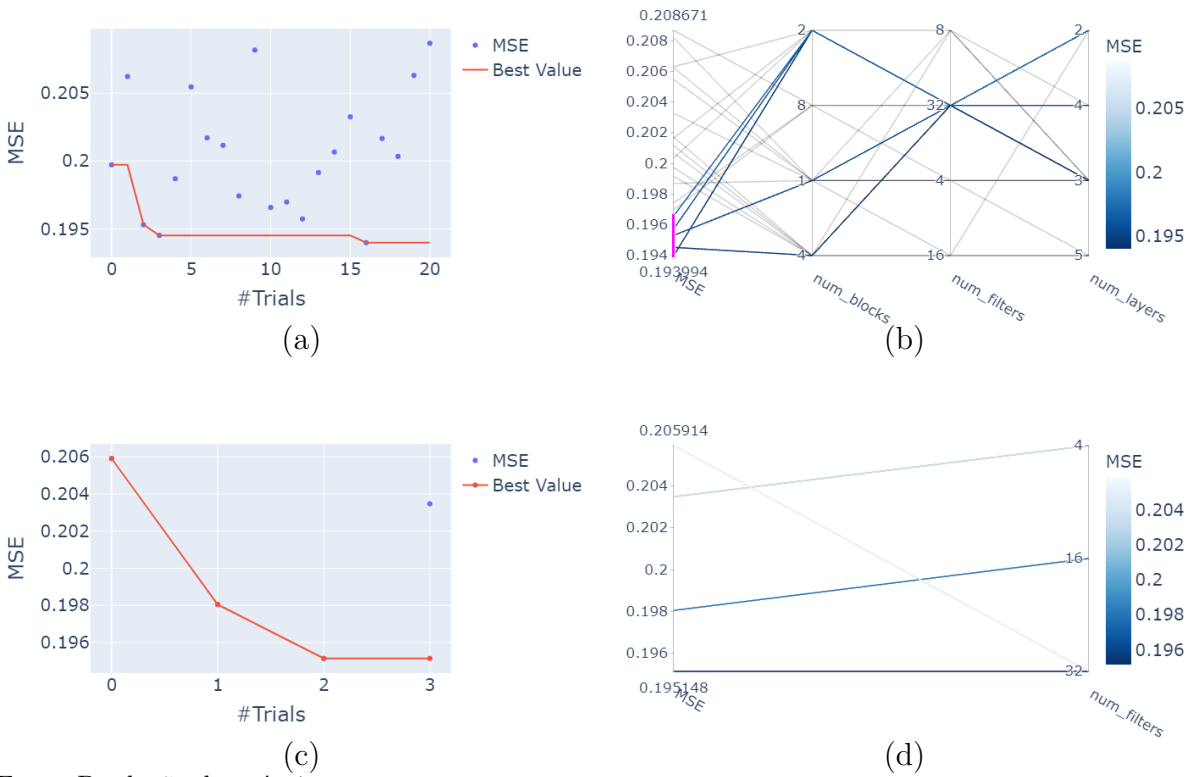
É interessante comentar que apesar de haver uma complexidade maior nesta arquitetura, os resultados encontrados não superam os obtidos por meio da segunda versão do modelo baseado apenas em LSTM. Pelo Gráfico 6(b), pode-se observar que todas as melhores pontuações escolheram 16 unidades na camada LSTM, *stride* e *kernel* de tamanho 3, divergindo apenas na quantidade de filtros, onde o melhor possuía valor 8 e teve um MSE de 0,186351. Já a segunda versão, olhando o Gráfico 6(b), observa-se que a pontuação aumentou, mostrando que o modelo não se adaptou bem ao uso de duas camadas LSTM em conjunto com a Conv1d. De qualquer forma, observa-se que seu melhor resultado, $0,188770$, foi obtido com taxa de *dropout* igual a 0,3.

4.4.3.3 Modelo baseado em WaveNet

O último modelo a ser validado é o baseado em WaveNet. Nos testes realizados são usadas, também, duas versões. Na primeira versão, ficou disponível para verificação do *framework*, as opções de três parâmetros: quantidade de filtros, quantidade de camadas por bloco e quantidade de blocos. As conexões de salto foram aplicadas, concatenando-se as saídas de todas as camadas de forma que os seus filtros fossem entregues ao restante da rede como características e as saídas das camadas como amostras temporais.

Já para a segunda versão, lançou-se mão da configuração de número de camadas e blocos resultante da versão inicial, deixando a quantidade de filtros para ser testada pela aplicação. Somado à isso, também inverteu-se a forma como as saídas das camadas e os filtros eram entendidos pelo restante da rede. Nesta versão os produtos eram as características e os filtros eram considerados como as amostras temporais. A estrutura desta rede é mostrada no Anexo B, na Tabela 8. Os resultados obtidos com estas configurações são mostrados no Gráfico 7.

Gráfico 7 – Tentativas feitas para o modelo baseado em WaveNet: (a) Histórico de tentativas para primeira versão, (b) Pontuação por parâmetro escolhido para primeira versão, (c) Histórico de tentativas para segunda versão e (d) Pontuação por parâmetro escolhido para segunda versão



Fonte: Produção do próprio autor.

De forma impressionante, com MSEs de $0,193994$ e $0,195148$ para a primeira e segunda versões, respectivamente, foram as melhores métricas obtidas para o modelo WaveNet.

Estas métricas foram inclusive piores que aquelas obtidas pelos modelos sem *inductive bias*. Estes resultados foram obtidos usando 32 filtros, 4 camadas por bloco e 2 blocos na configuração da rede.

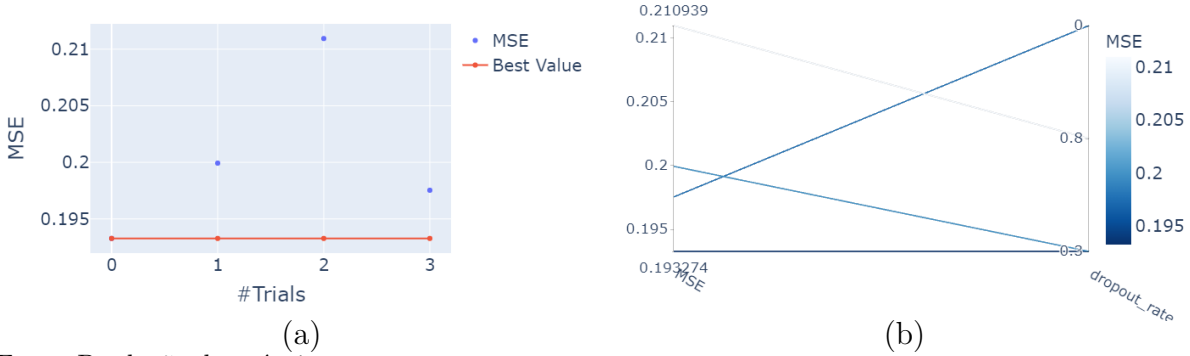
A arquitetura WaveNet, que também foi desenvolvida para trabalhar com séries temporais, possui uma vantagem com relação à camadas que possuem estados internos. Diferentemente destas, ela não possui o algoritmo de *Backpropagation Through Time* (tradução livre, retropropagação através do tempo), responsável por gerar o fenômeno de *vanishing* ou *exploding gradients* (BROWNLEE, 2017). Quanto maior a sequência de dados, maior é o efeito deste fenômeno. Isso permite com que a WaveNet possa ser usada para sequências de comprimento mais longo, diferentemente do que é sugerido para camadas como LSTM e GRU, todavia, pode ser que a sequência para a base de comparação escolhida seja pequena demais para que esta vantagem possa ser levada em consideração.

4.4.4 Modelos buscando melhor desempenho

4.4.4.1 Modelo baseado em WaveNet e MLP-Mixer

Como o uso da camada WaveNet criaria mais características por meio dos filtros, caso sua saída fosse acoplada com um bloco MLP-Mixer, este teria mais *patches* para trabalhar sobre. Esta união foi testada usando-se as melhores configurações dos modelos obtidos separadamente nas Seções 4.4.2.2 e 4.4.3.3, deixando como opção livre a taxa de *dropout* para o *framework*. A estrutura desta rede é mostrada no Anexo B, na Tabela 9. O resultado desta validação é mostrada no Gráfico 8.

Gráfico 8 – Tentativas feitas para o modelo baseado em WaveNet e MLP-Mixer: (a) Histórico de tentativas e (b) Pontuação por parâmetro escolhido



(a) (b)

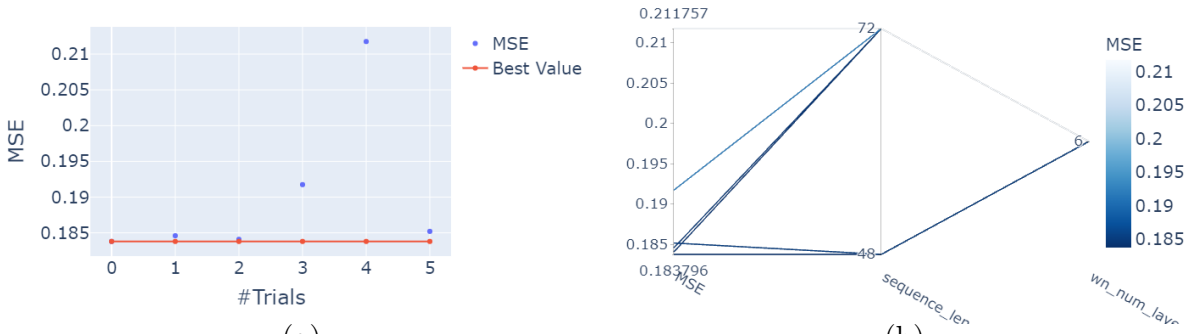
Fonte: Produção do próprio autor.

Com o melhor resultado apresentando um valor de $0,193274$ para uma taxa de *dropout* de 0,3, temos que a inclusão do bloco MLP-Mixer trouxe melhorias para o modelo, mesmo que elas sejam pequenas.

4.4.4.2 Modelo baseado em WaveNet e MLP-Mixer para sequências maiores

Conforme comentado na Seção 4.4.3.3, a maior vantagem do modelo WaveNet em comparação com redes recorrente poderia estar sendo desperdiçada devido ao tamanho da sequência não ser tão grande. Esta suposição é posta em prova neste experimento. Usando-se a mesma configuração utilizada na Seção 4.4.4.1, com exceção do número de camadas por bloco WaveNet que era dado a opção ao *framework* entre manter em 4 ou aumentar para 6, verificou-se o desempenho para sequências de tamanho 48 (correspondente a 24 horas, o dobro dos testes anteriores) e 72 (correspondente a 36 horas, o triplo do valor prévio). Vale ressaltar que valores maiores não foram usados devido à limitações dos recursos de *hardware* do computador pessoal do autor e do ambiente de execução do Colab. No segundo, houve a tentativa do treino com sequências de tamanho 96 (correspondente a 48 horas), mas a memória necessária foi maior que a disponível. A estrutura desta rede é mostrada no Anexo B, na Tabela 10. Os resultados deste ensaio são mostrados no Gráfico 9.

Gráfico 9 – Tentativas feitas para o modelo baseado em WaveNet e MLP-Mixer para sequências longas: (a) Histórico de tentativas e (b) Pontuação por parâmetro escolhido



(a) (b)

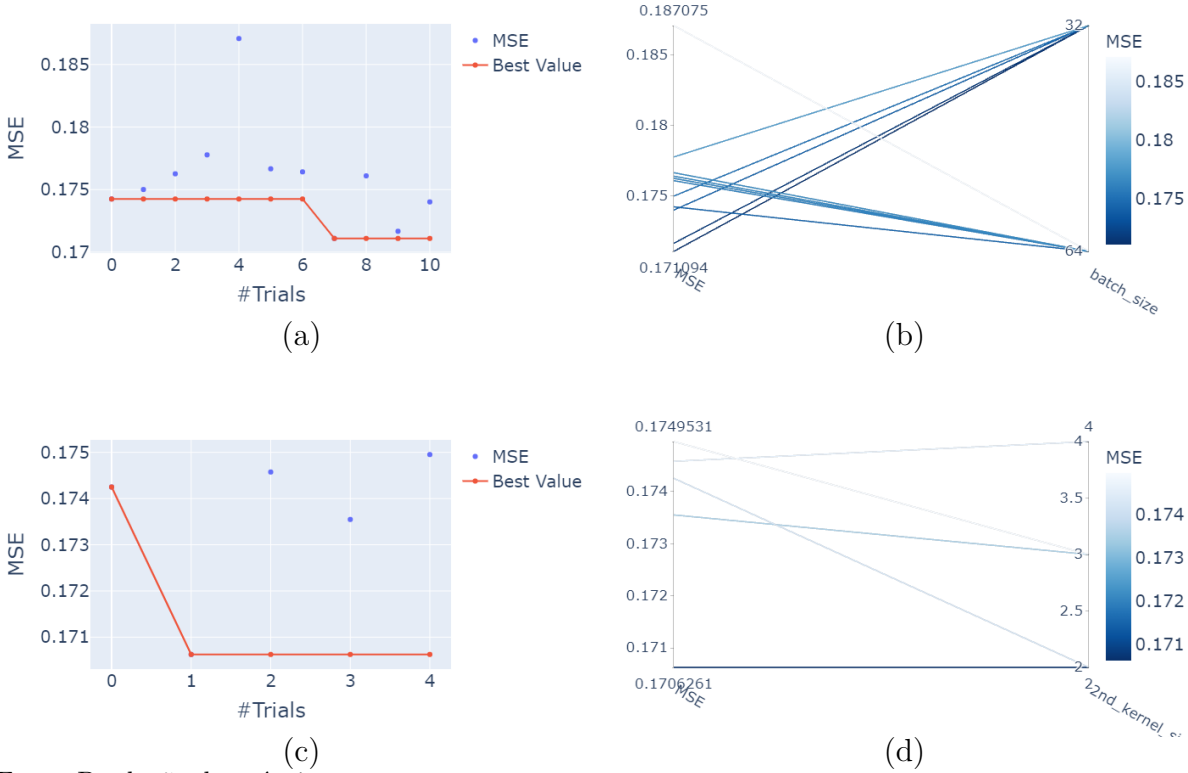
Fonte: Produção do próprio autor.

Com uma pontuação máxima de 0,183796, o modelo baseado em WaveNet e MLP-Mixer deixou de ter um dos piores desempenhos e passou a apresentar o melhor, usando um tamanho de sequência igual a 48 e um número de camadas por bloco igual a 6. Este resultado ilustra claramente a capacidade que esta rede possui para lidar com sequências mais longas. É possível dizer que, com mais recursos de *hardware*, haveria uma forma de buscar aumentar ainda mais um pouco o tamanho da sequência e, principalmente, inserir mais medidores e, consequentemente, mais dados. Com uma gama maior de informações, o bloco MLP-Mixer poderia generalizar ainda mais, conforme mostrado em seu *paper* (TOLSTIKHIN et al., 2021), gerando resultados ainda melhores.

4.4.4.3 Modelo baseado em LSTM para tamanhos menores de *batch*

Se por um lado o modelo WaveNet não é afetado pelo fenômeno de *vanishing* ou *exploding gradients*, o mesmo não pode ser dito das redes recorrentes. Como os estados internos de uma camada LSTM são reiniciados somente ao final de um *batch* de sequências, este fenômeno se torna mais presente para *batches* maiores. Foi necessário, portanto, validar se o desempenho das arquiteturas testadas na Seção 4.4.3.2 seriam melhores após diminuir o tamanho do *batch*. Foram feitas duas versões, onde a primeira é idêntica à segunda versão do modelo baseado em Conv1d e LSTM, já a segunda possui a adição de mais uma camada Conv1d, cuja opção escolhas para o *framework* era o tamanho do *kernel* desta segunda camada e usou-se o melhor tamanho de *batch* obtido na versão anterior. A estrutura desta rede é mostrada no Anexo B, na Tabela 11.Os resultados encontrados são apresentados no Gráfico 10.

Gráfico 10 – Tentativas feitas para o modelo baseado em Conv1d e LSTM para *batches* menores : (a) Histórico de tentativas para uma camada Conv1d, (b) Pontuação por parâmetro escolhido para uma camada Conv1d, (c) Histórico de tentativas para duas camadas Conv1d e (d) Pontuação por parâmetro escolhido para duas camadas Conv1d



Fonte: Produção do próprio autor.

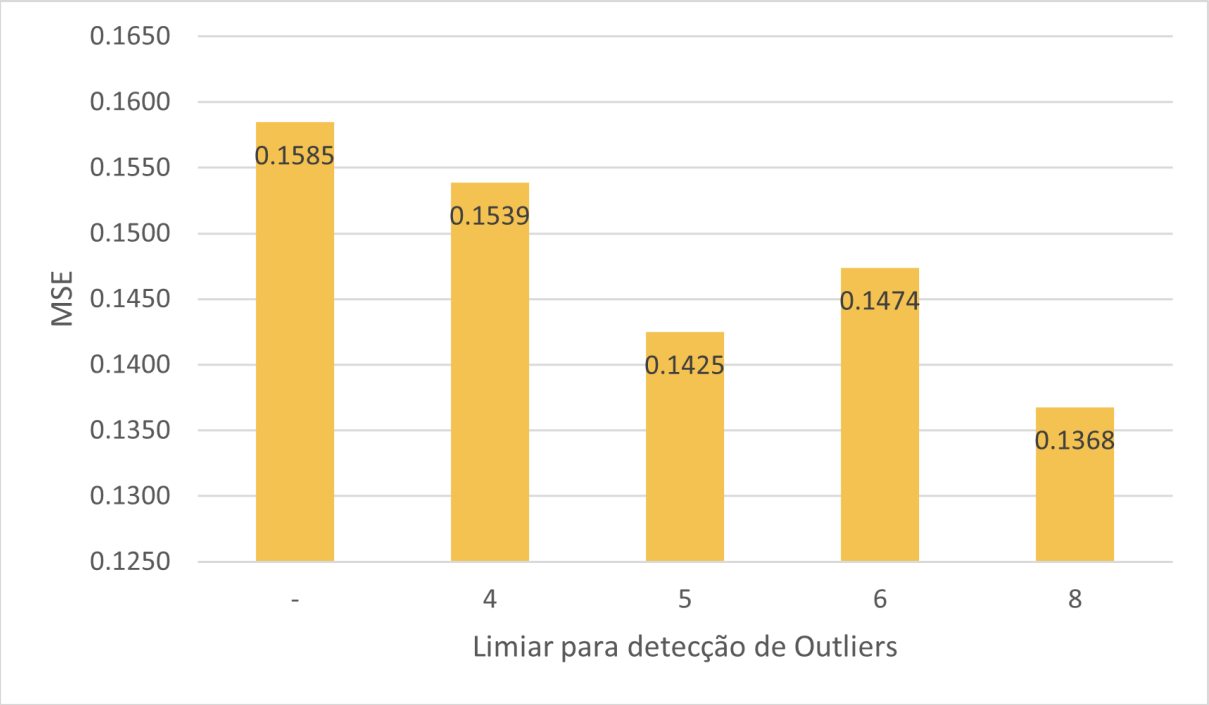
Com estes resultados de $0,171094$ e $0,170626$ para as duas versões, respectivamente, ficou provado o efeito negativo resultante dos *vanishing* e *exploding gradients*. Resolvendo este problema, o modelo conseguiu apresentar as métricas entre os demais experimentos.

4.4.5 Impacto do tratamento de *Outliers*

Além dos testes realizados com os modelos, também verificou-se o desempenho frente à remoção de dados que estivessem muito longe da média do conjunto. Esta distância medida em desvios padrões é chamada de limiar e qualquer valor que o ultrapasse é considerado como um *outlier*. Aqui, os pontos que extrapolassem o limite seriam restringidos e substituídos pelo valor do próprio limiar.

Para este experimento foi utilizado o modelo que obteve a métrica de $0,170626$ na Seção 4.4.4.3 e testados os limiares com valores 4, 5, 6 e 8. Vale ressaltar também que as métricas obtidas neste teste, diferentemente dos anteriores, não são sobre os dados após a transformação BoxCox, mas sim sobre os valores reais de consumo energético. Os resultados obtidos para cada limiar são mostrados no Gráfico 11.

Gráfico 11 – MSE obtido para experimentos com diferentes valores de limiar de *outlier*



Fonte: Produção do próprio autor.

Pode-se observar que usando-se um limiar de 8 desvios padrões, o modelo apresentou um resultado melhor para a sua previsão. Isso pode mostrar que o uso de limites menores pode restringir padrões naturais dos dados, enquanto o uso do maior limiar coibiu a interferência dos dados que estavam realmente fora da realidade e manteve a maior parte da natureza das informações.

4.5 Pesquisa reproduzível

Para que esta pesquisa possa ser posteriormente reproduzida, todos os códigos relevantes para realização dos experimentos estão disponíveis em https://github.com/enricozf/energy-consumption-forecast.

4.6 Resumo

Neste capítulo foi possível verificar o desempenho de diferentes arquiteturas de DL, experimentando para cada uma diferentes configurações por meio de tentativas de hiper parâmetros com o *framework* Optuna. Pôde-se através disso comparar como modelos com e sem *inductive bias* se comportam frente um desafio que envolve séries temporais, bem como a união dos mesmos pode auxiliar na solução desta problemática. Foi possível também buscar formas para que a métrica obtida pelos modelos fosse a menor possível. Ao final dos experimentos, as melhores métricas de cada modelo podem ser observadas na Tabela 2.

Tabela 2 – Comparação desempenho da rede MLP-Mixer com demais redes competitivas

Modelo	Descrição	MSE (%)
Fully-Connected	2 Camadas Dense	0,192805
MLP-Mixer	4 Blocos	0,192896
LSTM	1 Camada LSTM	0,185623
LSTM	2 Camadas LSTM	0,184637
Conv1d + LSTM	1 Camada LSTM	0,186351
Conv1d + LSTM	2 Camadas LSTM	0,188770
WaveNet	-	0,193994
WaveNet + MLP-Mixer	Melhores configurações anteriores	0,193274
WaveNet + MLP-Mixer	Melhores configurações e sequência longa	0,183796
Conv1d + LSTM	1 Camada Conv1d e *batch* menor	0,171094
Conv1d + LSTM	2 Camadas Conv1d e *batch* menor	0,170626

Fonte: Produção do próprio autor.

Com estes resultados da Tabela 2, é possível dividir a análise em duas partes. A primeira, relacionada às tentativas com quantidades iguais de tamanho de *batch* e sequência de entrada, mostra que, mesmo com uma pequena diferença, os modelos baseados apenas em camadas LSTM desempenham melhor que os demais. Isso é interessante, pois ele é mais simples que o modelo que também apresenta camadas Conv1d, e mesmo assim consegue apresentar resultados melhores. O modelo com duas camadas e mais complexo que o com apenas uma. todavia, se sai melhor no desafio, indicando que as relações entre os dados podem ser mais complicadas.

Já a segunda análise, com as demais redes, mostra que os modelos recorrentes são mais robustos ao problema, apresentando métricas melhores que o que usam WaveNet. Quando

analisa-se as melhores métricas dos dois melhores separadamente, observa-se que seus resultados estão próximos, tornando difícil a tarefa de determinar se um deles se sobressai. Quando a complexidade, observamos que o modelo com apenas uma camada Conv1d tem praticamente o mesmo desempenho, mas tem um nível menor de complicação. Isso pode pesar na escolha de qual modelo satisfez melhor a tarefa, porque além de ser mais simples, não obteve um resultado tão aquém do seu concorrente. Quando a análise é estendida, entretanto, à imagem da pontuação do modelo por parâmetro escolhido, presente no Gráfico 10, pode-se chegar a outra conclusão. Isso pode ser afirmado, pois observa-se que a variação das pontuações para o modelo de 2 camadas Conv1d é menor que a encontrada para o outro. Assim, a complexidade maior acaba agregando uma robustez também mais elevada. Desta forma, pode-se inferir que o melhor modelo apresenta a segunda configuração apresentada na Seção 4.4.4.3.

Usando o modelo com a melhor métrica e o melhor limiar de *outliers*, foi analisado novamente os resultados obtidos, mas desta vez convertendo-os de volta para uma escala de energia, em kWh. Juntamente com isso, afim de se ter uma base para comparação, foi obtido pelos dados qual era o consumo médio diário de todos os medidores da categoria *'Affluent'* e, com isso, foi feita uma média para qual seria o consumo médio para cinco horas, intervalo este previsto pelo modelo. Estes resultados são mostrados na Tabela 3.

Tabela 3 – Comparação predição do melhor modelo e consumos aproximados

Variável	Grandeza	Valor
Consumo diário de residências britânicas	kWh	11,81372
Consumo aproximado médio de 5 horas	kWh	2,46119
MSE para previsão com 5 horas	$(\text{kWh})^2$	0,13677
Raíz quadrado do MSE para previsão com 5 horas	kWh	0,36983
Razão entre raíz quadrada do MSE e consumo aproximado	-	0,15026

Fonte: Produção do próprio autor.

5 CONCLUSÕES E PROJETOS FUTUROS

5.1 Conclusões

O objetivo principal deste trabalho foi comparar desempenhos entre modelos de DL que possuem uma forma de *inductive bias* em sua arquitetura, sobre outros que não o tem. Quando se é analisado exclusivamente os resultados dos experimentos desenvolvidos neste projeto e resumidos na Tabela 2, fica evidente que modelos não enviesados possuem um desempenho inferior aos outros, todavia, algumas ponderações devem ser feitas após este estudo.

A primeira é que uma suposição inserida por uma arquitetura se beneficia de características dos dados e isso faz com que esta apresente resultados melhores que outras redes, todavia, se o viés escolhido não combinar com os dados, como foi o caso do modelo WaveNet apenas, o resultado pode ser pior do que se nenhuma tendência houvesse recebido preferência. Isso é importante pois mostra que a generalização e não escolha de uma predisposição podem apresentar resultados melhores do que uma rede com viés.

A segunda consideração que deve ser feita é que, por questões de limitação de recursos de *hardware*, o modelo MLP-Mixer não pode ser alimentado com uma quantidade gigantesca de dados, como se é instruído para que ele obtenha maior generalização. Talvez, com a liberdade para alimentar todos os dados de treino ao modelo, a sua capacidade de generalização poderia ser muito maior, e seus resultados consequentemente comparáveis com os modelos que se mostraram superiores nas experiências.

Ao fim do trabalho conclui-se que, a menos que seja possível introduzir uma quantidade muito grande de dados ao modelo ou que não se conheça os dados que está trabalhando, deve-se aplicar modelos de DL que insiram suposições coerentes em suas análises.

Analisando exclusivamente os objetivos específicos, pode-se afirmar que eles também puderam ser alcançados. Com relação à definição do método de remoção de *outliers*, foi usada a delimitação dos dados de consumo por limiar de desvio padrão, determinando-se claramente que valores que ultrapassassem um limite seriam reajustados para este mesmo limite. Sobre a lista de variáveis a serem alimentadas ao modelo, foi possível analisar as possibilidades e determinar que 4 seriam as mais importantes como entrada, sendo elas o consumo energético, se o dia era uma segunda-feira ou fim de semana e temperaturas semanal e divergência da temperatura horária para a semanal.

O terceiro objetivo específico dizia sobre o ajuste dos hiper parâmetros dos modelos. Isso foi feito utilizando-se o *framework* Optuna e as suas tentativas com opções escolhidas pelo

autor. Por último, a comparação entre os diferentes modelos foi realizada, evidenciando quais apresentavam um desempenho melhor para a tarefa de séries temporais apresentada neste trabalho.

Outro resultado interessante foi aquele obtido junto com a Tabela 3, onde pode-se observar, baseado numa comparação com os consumos médios diários para a categoria *Affluent*, que a previsão feita pelo modelo alcançou um desempenho razoável. Apesar do erro encontrado ser equivalente a aproximadamente 15% para o período previsto, este valor é reduzido para apenas $3,1031\%$. Este valor é relevante, principalmente quando considera-se que o erro tende a aumentar ao passo que o período para previsão se torna maior. Se foi realizada uma previsão com 5 horas, é suposto que a primeira hora prevista esteja com um erro menor que a última, devido à própria dificuldade da tarefa.

Isso conclui que o desafio da previsão do consumo para unidades domiciliares pôde ser efetuado com um desempenho consideravelmente positivo.

5.2 Temas a serem pesquisados

Como trabalhos futuros, principalmente pelo gargalo que se mostrou a falta de recursos de *hardware*, sugere-se estudar e implementar meios que permitam a inserção de todos os dados do conjunto de treino para serem analisados pelo modelo. Isto definitivamente melhoraria o resultado das arquiteturas sem *inductive bias*, podendo inclusive torná-las competitivas. Uma sugestão para se fazer isso é, diferentemente do que foi desenvolvido neste projeto, não fazer um embaralhamento perfeito dos *batches*. Este fator era o que exigia um uso extensivo de memória do computador, fazendo com que o mesmo não conseguisse executar este procedimento. Apesar do embaralhamento perfeito ser a medida correta a se tomar, não fazê-la permitirá com que mais dados de outros *smart meters* sejam estudados pelo modelo e podendo torná-lo mais genérico. Recomenda-se também estudar arquiteturas que não foram testadas nos experimentos realizados neste projeto.

Por fim, faz sentido também buscar um conjunto de dados semelhante a este que possua o número de habitantes que moram na residência onde o *smart meter* esta instalado, tendo em vista que este dado impacta fortemente sobre o consumo energético doméstico.

REFERÊNCIAS

ABADI, M.; BARHAM, P.; CHEN J AMD CHEN, Z.; DAVIS, A.; DEAN, J.; DEVIN, M.; GHEMAWAT, S.; IRVING, G.; ISARD, M.; KUDLUR, M.; LEVENBERG, J.; MONGA, R.; MOORE, S.; MURRAY, D. G.; STEINER, B.; TUCKER, P.; VASUDEVAN, V.; WARDEN, P.; WICKE, M.; YU, Y.; ZHENG, X. Tensorflow: A system for large-scale machine learning. 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), Geórgia, v. 12, p. 265–283, 2016. Disponível em: <https://www.usenix.org/sites/default/files/osdi16_full_proceedings.pdf>. Acesso em: 12 set. 2021. Citado na página 35.

ABERA, F. Z.; KHEDKAR, V. Machine learning approach electric appliance consumption and peak demand forecasting of residential customers using smart meter data. Wireless Personal Communications, Suiça, v. 111, p. 65–82, 2020. Disponível em: <https://doi.org/10.1007/s11277-019-06845-6>. Acesso em: 11 abr. 2021. Citado na página 13.

AKIBA, T.; SANO, S.; YANASE, T.; OHTA, T.; KOYAMA, M. Optuna: A next-generation hyperparameter optimization framework. In: Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. [S.l.: s.n.], 2019. Citado na página 35.

ALAHAKOON, D.; YU, X. Smart electricity meter data intelligence for future energy systems: A survey. IEEE Transactions on Industrial Informatics, Nova Jersey, v. 12, n. 1, p. 425–436, 2016. Disponível em: <https://ieeexplore.ieee.org/abstract/document/7063262>. Acesso em: 11 abr. 2021. Citado 2 vezes nas páginas 17 e 62.

APPLE. DARKSKY. 2021. Interface de programação para dados meteorológicos. Disponível em: <https://darksky.net/forecast/40.7127,-74.0059/us12/en>. Acesso em: 16 abr. 2021. Citado na página 36.

AYYADEVARA, V. K. Neural Networks with Keras Cookbook. [S.l.]: Packt Publishing, 2019. Citado na página 32.

BRASIL. Ministério de Minas e Energia. In: Consumo Anual de Energia Elétrica por classe (nacional). Brasília, DF: MME, EPE, 2019. Disponível em: <https://www.epe.gov.br/pt/publicacoes-dados-abertos/publicacoes/consumo-de-energia-eletrica/consumo-anual-de-energia-eletrica-por-classe-(nacional)>. Acesso em: 10 abr. 2021. Citado na página 15.

BRASIL. Ministério de Minas e Energia. In: Plano Nacional de Energia 2030. Brasília, DF: MME, EPE, 2020. Disponível em: <https://www.epe.gov.br/pt/publicacoes-dados-abertos/publicacoes/plano-decenal-de-expansao-de-energia-2030>. Acesso em: 10 abr. 2021. Citado 3 vezes nas páginas 15, 16 e 17.

BROWNLEE, J. Long Short-Term Memory Networks With Python. [S.l.]: Machine Learning Mastery, 2017. Citado 2 vezes nas páginas 33 e 49.

DEPARTMENT OF ENERGY & CLIMATE CHANGE. UK Energy in Brief 2014. Londres: [s.n.], 2014. Disponível em: <https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/350941/UK_Energy_in_Brief_2014_revised.pdf>. Acesso em: 16 abr. 2021. Citado 2 vezes nas páginas 13 e 36.

DONG, C.; DU, L.; JI, F.; SONG, Z.; ZHENG, Y.; HOWARD, A.; INTREVADO, P.; WOODBRIDGE, D. M.; HOWARD, A. J. Forecasting smart meter energy usage using distributed systems and machine learning. In: 2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS). Exeter, Reino Unido: [s.n.], 2018. Disponível em: <https://ieeexplore.ieee.org/abstract/document/8622954>. Acesso em: 11 abr. 2021. Citado na página 14.

GAJOWNICZEK, K.; ZABKOWSKI, T. Short term electricity forecasting using individual smart meter data. Procedia Computer Science, Amsterdã, v. 35, p. 589–597, 2014. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1877050914011053>. Acesso em: 11 abr. 2021. Citado 2 vezes nas páginas 14 e 17.

GERS, F. A.; SCHMIDHUBER, J. Recurrent nets that time and count. In: Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium. Como, Itália: [s.n.], 2000. Disponível em: <https://ieeexplore.ieee.org/document/861302>. Acesso em: 18 abr. 2021. Citado na página 24.

GOVERNO DE SÃO PAULO. Secretaria de Infraestrutura e Meio Ambiente. In: Resumo Executivo: Dados de produção e consumo de energia elétrica. São Paulo, Brasil: Secretaria de Infraestrutura e Meio Ambiente, 2021. Disponível em: <http://dadosenergeticos.energia.sp.gov.br/portalcev2/intranet/BiblioVirtual/eletrica/Resumo_Executivo_EE.pdf>. Acesso em: 17 abr. 2021. Citado na página 17.

GOYAL, A.; BENGIO, Y. Inductive biases for deep learning of higher-level cognition. CoRR, [Online], 2020. Disponível em: <https://arxiv.org/abs/2011.15091>. Acesso em: 12 set. 2021. Citado 2 vezes nas páginas 24 e 31.

GREFF, K.; SRIVASTAVA, R. K.; KOUTNÍK, J.; STEUENEBRINK, B. R.; SCHIMIDHUBER, J. LSTM: A search space odyssey. IEEE Transactions on Neural Networks and Learning Systems, Nova Jersey, v. 28, n. 10, p. 2222–2232, 2017. Disponível em: <https://ieeexplore.ieee.org/document/7508408>. Acesso em: 16 abr. 2021. Citado 3 vezes nas páginas 14, 22 e 24.

IBRAHIM, H.; ILINCA, A.; PERRON, J. Energy storage systems — characteristics and comparisons. Renewable and Sustainable Energy Reviews, Amsterdã, v. 12, n. 12, p. 1221–1250, 2008. Disponível em: <https://www.sciencedirect.com/science/article/abs/pii/S1364032107000238>. Acesso em: 15 abr. 2021. Citado na página 14.

INTERNATIONAL ENERGY AGENCY. Electricity final consumption by sector. Paris: [s.n.], 2018. Disponível em: <https://www.iea.org/data-and-statistics>. Acesso em: 14 abr. 2021. Citado 2 vezes nas páginas 13 e 15.

KIM, T. Y.; CHO, S. B. Predicting residential energy consumption using CNN-LSTM neural networks. Energy, Amsterdã, v. 182, p. 72–81, 2019. Disponível em:

<https://www.sciencedirect.com/science/article/abs/pii/S0360544219311223>. Acesso em: 12 abr. 2021. Citado 4 vezes nas páginas 15, 17, 18 e 21.

KOPONEN, P.; SACO, L. D.; ORCHARD, N.; VORISEK, T.; PARSONS, J.; ROCHAS, C.; MORCH, A. Z.; LOPES, V.; TOGEBY, M. Definition of Smart Metering and Applications and Identification of Benefits. Finlândia: [s.n.], 2008. Disponível em: <https://www.researchgate.net/publication/235709839_Definition_of_Smart_Metering_and_Applications_and_Identification_of_Benefits>. Acesso em: 11 abr. 2021. Citado na página 14.

MICHEL, J. D. Smart Meters in London. Canada: [s.n.], 2016. Página web para acesso aos dados. Disponível em: <https://www.kaggle.com/jeanmidev/smart-meters-in-london>. Acesso em: 9 mar. 2021. Citado na página 36.

SHAHZADEH, A.; KHOSRAVI, A.; NAHAVANDI, S. Improving load forecast accuracy by clustering consumers using smart meter data. In: 2015 International Joint Conference on Neural Networks. Killarney: [s.n.], 2015. Disponível em: <https://ieeexplore.ieee.org/document/7280393>. Acesso em: 11 abr. 2021. Citado 2 vezes nas páginas 14 e 18.

TOLSTIKHIN, I.; HOULSBY, N.; KOLESNIKOV, A.; BEYER, L.; ZHAI, X.; UNTERTHINER, T.; YUNG, J.; STEINER, A.; KEYSERS, D.; USZKOREIT, J.; LUCIC, M.; DOSOVITSKIY, A. Mlp-mixer: An all-mlp architecture for vision. CoRR, [Online], 2021. Disponível em: <https://arxiv.org/abs/2105.01601>. Acesso em: 29 jul. 2021. Citado 5 vezes nas páginas 28, 29, 31, 45 e 50.

UK Government. Acorn consumer classification (CACI). Londres: [s.n.], 2020. Disponível em: <https://www.gov.uk/government/statistics/quality-assurance-of-administrative-data-in-the-uk-house-price-index/acorn-consumer-classification-caci>. Acesso em: 10 abr. 2021. Citado na página 36.

UNITED STATES ENERGY INFORMATION ADMINISTRATION. Electricity net consumption (billion kWh). Washington: [s.n.], 2018. Disponível em: <https://www.eia.gov/international/data/world/electricity/electricity-consumption>. Acesso em: 11 abr. 2021. Citado 2 vezes nas páginas 13 e 15.

Van den Oord, A.; DIELEMAN, S.; ZEN, H.; SIMONYAN, K.; VINYALS, O.; GRAVES, A.; KALCHNRENNER, N.; SENIOR, A.; KAVUKCUOGLU, K. Mlp-mixer: An all-mlp architecture for vision. CoRR, [Online], 2016. Disponível em: <https://arxiv.org/abs/1609.03499>. Acesso em: 28 mai. 2021. Citado 3 vezes nas páginas 25, 26 e 27.

YAN, K.; LI, W.; JI, Z.; QI, M.; DU, Y. A hybrid LSTM neural network for energy consumption forecasting of individual households. IEEE Access, Nova Jersey, v. 7, p. 157633–157642, 2019. Disponível em: <https://ieeexplore.ieee.org/abstract/document/8880605>. Acesso em: 12 abr. 2021. Citado 3 vezes nas páginas 14, 17 e 22.

ZHANG, D.; HAN, X.; DENG, C. Review on the research and practice of deep learning and reinforcement learning in smart grids. CSEE Journal of Power and Energy Systems, Nova Jersey, v. 4, n. 3, p. 362–370, 2018. Disponível em: <https://ieeexplore.ieee.org/abstract/document/8468674>. Acesso em: 10 abr. 2021. Citado na página 20.

ZHENG, J.; GAO, D. W.; LIN, L. Smart meters in smart grid: An overview. In: 2013 IEEE GREEN TECHNOLOGIES CONFERENCE (GREENTECH). Denver: [s.n.], 2013. p. 57–64. Disponível em: <https://ieeexplore.ieee.org/abstract/document/6520030>. Acesso em: 10 abr. 2021. Citado 2 vezes nas páginas 13 e 62.

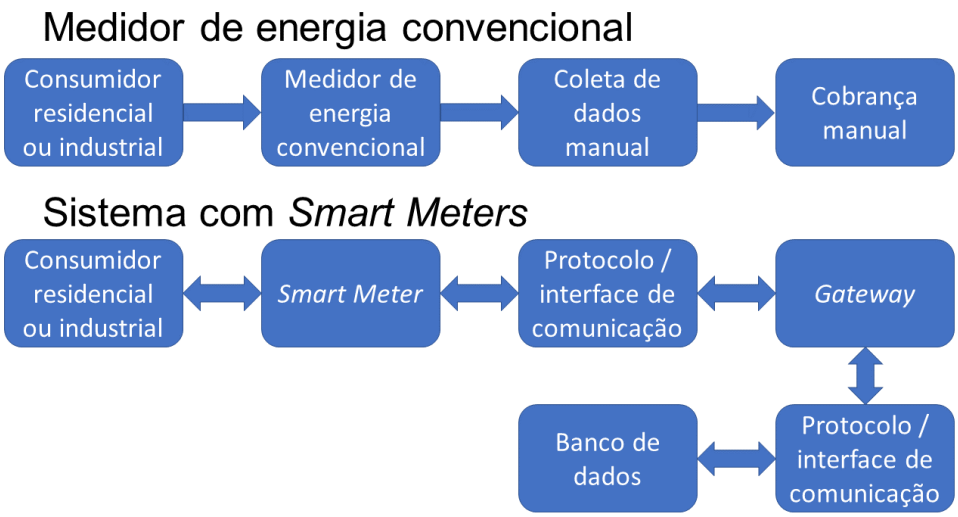
ZHIQIANG, W.; JUN, L. A review of object detection based on convolutional neural network. In: 2017 36th Chinese Control Conference (CCC). Dalian: [s.n.], 2017. Disponível em: <https://ieeexplore.ieee.org/abstract/document/8029130>. Acesso em: 17 abr. 2021. Citado na página 21.

Anexos

ANEXO A – *SMART METERS*

O *smart meter* é um engenhoso dispositivo feito para obter informação referentes ao uso de outros aparelhos que estejam conectados à rede elétrica e das medições de eletricidade da unidade consumidora em que estiver ligado. Para aprimorar o monitoramento das distribuidoras ou operadores do sistema, envia-se estes dados através de uma conexão segura graças ao *gateway* interno do medidor (ZHENG; GAO; LIN, 2013; ALAHAKOON; YU, 2016). Uma característica marcante para estes componentes é a possibilidade de comunicação bidirecional. Ou seja, ele pode não somente enviar, como também receber informações e comandos. A partir deste avanço, eles são amplamente utilizados para realizar ligação ou desconexão de pontos da rede remotamente e enviar ordens para outros dispositivos, caso também se comuniquem pelo protocolo *power line communications* (tradução livre, comunicação sobre linhas de potência). Além disso, como sugerido pelo próprio nome, eles servem para medição e monitoramento de parâmetros do sistema elétrico e envio desses dados. Estas aferições são demasiadamente relevantes para que o gerenciamento de redes de distribuição e transmissão seja feito de forma controlada e fluida, evitando surdos e falhas, deixando-as, por conseguinte, mais confiáveis. Na Figura 14 a seguir, apresenta-se um esquema que compara o medidor convencional com um *smart meter*.

Figura 14 – Arquiteturas de medição de um medidor de energia convencional e um smart meter



Fonte: Zheng, Gao e Lin (2013).
Nota: Traduzido pelo autor.

ANEXO B – ESTRUTURAS DOS MODELOS

Este anexo é responsável por apresentar as estruturas que foram usadas durantes os experimentos deste trabalho, apresentados na Seção 4.4. As redes de cada subseção serão apresentadas em uma tabela, onde diferentes versões testadas ocuparam outras colunas e as camadas serão povoadas pelas linhas. Para posições da tabela onde a descrição for Optuna, deve-se atentar para o fato de esta característica foi determinada segundo as opções de escolha apresentadas ao *framework* Optuna e explicadas na seção correspondente.

Tabela 4 – Estrutura da rede densa testada na Seção 4.4.2.1

Camadas	Descrição
Input	Formato de entrada (24,4)
TimeDistributed(*Dense*)	Optuna
Flatten	-
Dense	10 unidades

Fonte: Produção do próprio autor.

Tabela 5 – Estrutura da rede baseada em MLP-Mixer testada na Seção 4.4.2.2

Camadas	Descrição
Input	Formato de entrada (24,4)
Patches	6 *Patches* de tamanho 4
Blocos MLP-Mixer Empilhados	16 unidades internas
GlobalAveragePooling1D	-
Dense	10 unidades

Fonte: Produção do próprio autor.

Tabela 6 – Estrutura da rede baseada em LSTM testada na Seção 4.4.3.1

Camadas	Descrição versão 1	Descrição versão 2
Input	Formato de entrada (24,4)	Formato de entrada (24,4)
LSTM	Optuna	Optuna
LSTM	-	Optuna
Dense	10 unidades	10 unidades

Fonte: Produção do próprio autor.

Tabela 7 – Estrutura da rede baseada em Conv1d e LSTM testada na Seção 4.4.3.2

Camadas	Descrição versão 1	Descrição versão 2
Input	Formato de entrada (24,4)	Formato de entrada (24,4)
Conv1d	Optuna	NF[1] = 8, TK[2] = 3, TS[3] = 3
LSTM	Optuna	16
LSTM	-	8
Dense	-	20 unidades
Dropout	-	Optuna
Dense	10 unidades	10 unidades

Fonte: Produção do próprio autor;
[1] Número de Filtros;
[2] Tamanho do *kernel*;
[3] Tamanho do *Stride*.

Tabela 8 – Estrutura da rede baseada em WaveNet testada na Seção 4.4.3.3

Camadas	Descrição versão 1	Descrição versão 2
Input	Formato de entrada (24,4)	Formato de entrada (24,4)
Conv1d	Optuna	Optuna
Blocos WaveNet	Optuna	Optuna
concatenate	Formato (TB[1], CW[2], NF[3])	Formato (TB[1], NF[3], CW[2])
Flatten	-	-
Dense	10 unidades	10 unidades

Fonte: Produção do próprio autor;
[1] Tamanho do Batch;
[2] Camadas WaveNet;
[3] Número de Filtros.

Tabela 9 – Estrutura da rede baseada em WaveNet e MLP-Mixer testada na Seção 4.4.4.1

Camadas	Descrição
Input	Formato de entrada (24,4)
Conv1d	NF[3] = 32, TK[4] = 2, *padding* =′ *causal*′
Bloco WaveNet	NF[3] = 32, NC[5] = 4
Bloco WaveNet	NF[3] = 32, NC[5] = 4
concatenate	Formato (TB[1], CW[2], NF[3])
Patches	6 *Patches* de tamanho 4
Bloco MLP-Mixer	Optuna
Bloco MLP-Mixer	Optuna
Bloco MLP-Mixer	Optuna
Bloco MLP-Mixer	Optuna
GlobalAveragePooling1D	-
Dense	10 unidades

Fonte: Produção do próprio autor;
[1] Tamanho do Batch;
[2] Camadas WaveNet;
[3] Número de Filtros;
[4] Tamanho do *Kernel*;
[5] Número de Camadas WaveNet.

Tabela 10 – Estrutura da rede baseada em WaveNet e MLP-Mixer para sequências longas testada na Seção 4.4.4.2

Camadas	Descrição
Input	Formato de entrada (24,4)
Conv1d	NF[1] = 32, TK[2] = 2, *padding* =′ *causal*′
Bloco WaveNet	NF[1] = 32, NC[3] = 4
Bloco WaveNet	NF[1] = 32, NC[3] = 4
concatenate	Formato (TB[4], CW[5], NF[1])
Patches	6 *Patches* de tamanho 4
Bloco MLP-Mixer	16 estados internos, $dropout = 0,3$
Bloco MLP-Mixer	16 estados internos, $dropout = 0,3$
Bloco MLP-Mixer	16 estados internos, $dropout = 0,3$
Bloco MLP-Mixer	16 estados internos, $dropout = 0,3$
GlobalAveragePooling1D	-
Dense	10 unidades

Fonte: Produção do próprio autor;
[1] Número de Filtros;
[2] Tamanho do *Kernel*;
[3] Número de Camadas WaveNet.

[4] Tamanho do Batch;
[5] Camadas WaveNet;

Tabela 11 – Estrutura da rede baseada em Conv1d e LSTM para tamanhos de *batch* menores testada na Seção 4.4.4.3

Camadas	Descrição versão 1	Descrição versão 2
Input	Formato de entrada (24,4)	Formato de entrada (24,4)
Conv1d	NF[1] = 8, TK[2] = 3, TS[3] = 3	NF[1] = 8, TK[2] = 3, TS[3] = 3
Conv1d	-	Optuna
LSTM	16	16
LSTM	8	8
Dense	20 unidades	20 unidades
Dropout	0,3	0,3
Dense	10 unidades	10 unidades

Fonte: Produção do próprio autor;
[1] Número de Filtros;
[2] Tamanho do *kernel*;
[3] Tamanho do *Stride*.