Dr. Alexandre Barbosa de Lima

Desenvolvimento de Bancada Virtual Representativa das Células e do Sistema de Monitoramento de um "Electrical Energy Storage" com Baterias de Íons de Lítio

São Paulo – Brasil

2021

Agradecimentos

Ao Pai Eterno, por ter me sustentado ao longo deste projeto, realizado em meio à pandemia de SARS-CoV-2.

A minha esposa Shirlei, que sempre me apoiou de forma incondicional em todos os meus projetos acadêmicos e profissionais.

Ao meu supervisor, Prof. Dr. José Roberto Cardoso, pela orientação e suporte recebidos.

Ao Prof. Dr. Maurício B. C. Salles, pela sua valiosa colaboração.

Resumo

A literatura recente sugere que a abordagem de aprendizado de máquina, baseada em algoritmos de redes neurais profundas (aprendizado profundo ou *deep learning*), é o estado da arte em estimação do estado da carga (State-Of-Charge (SOC)) de baterias de íons de lítio. Os resultados obtidos indicam que as redes neurais profundas alimentadas para frente (*feed-forward*) conseguem estimar o SOC de uma célula de íons de lítio com grande acurácia.

Palavras-chaves: *Deep Learning*, *Electrical Energy Storage*, *Machine Learning*, *State-of-Charge*.

Lista de ilustrações

Lista de tabelas

LISTA DE ABREVIATURAS

API	*Application Programming Interface*
AUKF	*Adaptive Unscented Kalman Filter*
BMS	*Battery Management System*
CNTK	Microsoft Cognitive Toolkit
CPS	*Cyber-Physical System*
CPU	*Central Processing Unit*
DL	*Deep Learning*
DFNN	*Deep Forward Neural Networks*
DNN	*Deep Neural Networks*
ECM	*Equivalent Circuit Model*
EES	*Electrical Energy Storage*
EKF	*Extended Kalman Filter*
ENIAC	*Electronic Numerical Integrator and Computer*
EV	*Electric Vehicles*
GPU	*Graphics Processing Unit*
GTU	*Gated Recurrent Unit*
IA	Inteligência Artificial
IDE	*Integrated Development Environment*
IoT	*Internet of Things*
LSTM	*Long Short-Term Memory*
MAE	*Mean Absolute Error*
MCP	Modelo de neurônio artificial McCulloch e Pitts
ML	*Machine Learning*
MSE	*Mean Square Error*

NN	*Neural Networks*
PHS	*Pumped Hydroelectric Storage*
PLN	Processamento de Linguagem Natural
RLS	Regressão Linear Simples
RLM	Regressão Linear Múltipla
RNA	Redes Neurais Artificiais
RNN	*Recurrent Neural Networks*
RUL	*Remaining Useful Life*
SCF	Sistema Ciber-Físico
SGD	*Stochastic Gradient Descent*
SOC	*State-Of -Charge*
SVM	*Support Vector Machine*
TDF	Transformada Discreta de Fourier
TDIF	Transformada Discreta Inversa de Fourier

Sumário

1 Introdução

1.1 O Setor Elétrico Brasileiro

No Brasil, a energia hidroelétrica é a principal fonte de energia elétrica. Não obstante o grande potencial hídrico do país, é frequente o acionamento de usinas termoelétricas durante períodos de estiagem. Conforme apontado por Bolzon (1), é importante diversificar as fontes de energia, pois a dependência das hidroelétricas torna o sistema energético brasileiro vulnerável. A participação de fontes renováveis de energia na matriz energética do país é bem vinda, pois podem compensar e complementar o parque hidroelétrico a um custo menor do que as termoelétricas.

Tal desafio impõe a utilização de novas tecnologias nos processos de geração, transmissão e distribuição de energia. A integração das fontes renováveis, o aumento da eficiência do sistema, a melhoria dos sistemas de controle e a possibilidade de inserção do usuário final na operação do sistema dá origem ao conceito de rede elétrica inteligente ou *smart grid*.

Nas redes tradicionais, a geração de energia segue a demanda e é centralizada. O paradigma do *smart grid* é diferente: as redes tenderão a ser distribuídas, inteligentes e flexíveis, com fluxos de energia e comunicação bidirecionais. Essa mudança exigirá a pesquisa e o desenvolvimento de novos métodos de controle, automação e operação do sistema elétrico que levem em conta a característica de Sistema Ciber-Físico (SCF)[1] (*Cyber-Physical System* (CPS)) do *smart grid*[2].

1.2 O Problema do Armazenamento de Energia

O armazenamento de energia atua como um mediador entre cargas variáveis e fontes variáveis. O armazenamento de eletricidade não é algo novo. Volta inventou a bateria moderna em 1799. Baterias foram implementadas em redes de telegrafia em 1836 (4). A Usina Hidrelétrica de Rocky River, em New Milford, Connecticut, foi o primeiro grande projeto de armazenamento de energia elétrica (*Electrical Energy Storage* (EES)) dos Estados Unidos. A usina utilizava a tecnologia de armazenamento hidrelétrico por

[1] Segundo Rajeev (2), "Um sistema ciber-físico consiste de dispositivos computacionais que se comunicam pela rede (*Internet of Things* (IoT)) e que interagem com o mundo físico por meio de sensores e atuadores."

[2] A Internet das Coisas (IoT) pode ser definida como uma rede altamente interligada de entidades heterogêneas tais como sensores, dispositivos embarcados, dispositivos portáteis e servidores, os quais oferecem novos serviços e aplicações para as áreas de transporte, redes elétricas, tratamento médico e automação predial (3).

meio de bombeamento (*Pumped Hydroelectric Storage* (PHS)).

Hannan et al. (5) apresentam uma taxonomia detalhada dos tipos de sistemas de armazenamento de energia levando em conta a forma de armazenamento de energia e materiais construtivos: mecânico, eletroquímico (baterias recarregáveis e de fluxo), químico, elétrico (ultracapacitor ou bobina magnética supercondutora), térmico e híbrido.

O avanço das tecnologias de EES viabilizou o surgimento do iPod, *smartphones* e *tablets* com baterias de íons de lítio. Se as fontes renováveis, tais como a solar e a eólica, tornarem-se prevalentes, o EES será um dos componentes críticos das rede elétricas, haja vista o caráter intermitente dessas fontes de energia (6), (7). Sistemas de EES são necessários mesmo quando fontes renováveis estão conectadas à rede (*grid*), porque é necessário suavizar o fornecimento de energia. Por exemplo, o EES de um prédio ou fábrica pode ser carregado durante as horas de demanda reduzida e suprir/complementar a demanda de energia no horário de pico.

Recentemente, a indústria e a academia têm dado grande importância ao tema da eletrificação do sistema de transporte, dada a necessidade de se reduzir a emissão dos gases de efeito estufa. Veículos elétricos híbridos, como o Toyota Prius, ou totalmente elétricos, como os vários modelos da Tesla, o Nissan Leaf e o GM Volt, são *cases* de sucesso nos Estados Unidos (6). Sistemas de EES também são relevantes para estas aplicações.

A tecnologia de EES consiste no processo de conversão de uma forma de energia (quase sempre elétrica) para uma forma de energia armazenável, que possa ser reconvertida em energia elétrica quando necessário. A EES possui as seguintes funções: auxiliar o atendimento das demandas máximas de carga elétrica, prover gerenciamento de energia variante no tempo, aliviar a intermitência da geração de energia renovável, melhorar a qualidade/confiabilidade da energia, atender cargas remotas e veículos, dar suporte à realização das redes inteligentes, melhorar a gestão da geração de energia distribuída/em espera e reduzir a importação de energia elétrica durante os períodos de pico de demanda (7), (8).

Um EES (que pode se conectar à rede ou operar em modo isolado ou *stand-alone*) é composto por dois sub-sistemas principais: i) armazenamento e ii) eletrônica de potência. Tais sub-sistemas são complementados por outros componentes que incluem sistemas de monitoramento e controle (4).

A tecnologia das baterias de íons de lítio tem atraído a atenção da indústria e da academia na última década. Isso se deve principalmente ao fato de as baterias de íons de lítio oferecerem mais energia, maior densidade de potência, maior eficiência e menor taxa de auto-descarga do que outras tecnologias de bateria como NiCd, NiMH, etc. (9).

A utilização eficiente da bateria de íons de lítio requer a supervisão de um sistema de gerenciamento de bateria (*Battery Management System* (BMS)), pois é necessário que a bateria opere em condições adequadas de temperatura e carga (SOC) (10). A temperatura da célula produz efeitos deletérios na tensão de circuito aberto, resistência interna e capacidade disponível e também pode levar a uma rápida degradação da bateria se ela operar acima de um dado limiar de temperatura. Portanto, a modelagem da bateria é de suma importância, uma vez que será usada pelo BMS para gerenciar o funcionamento da bateria (9).

1.3 Modelagem de Baterias

Há dois métodos de modelagem de baterias: i) *model-driven* (orientado por modelo) e ii) *data-driven* (orientado por dados[3]) (11).

Os modelos eletrotérmicos, que pertencem à categoria dos métodos *model-driven*, são comumente classificados como: i) eletroquímicos ou ii) baseados em modelos de circuitos equivalentes (*Equivalent Circuit Model* (ECM)) (9), (10).

Modelos eletroquímicos são baseados em equações diferenciais parciais (12) e conseguem representar os efeitos térmicos com maior precisão que os ECM (13). Contudo, a primeira classe de modelos requer que se tenha um conhecimento detalhado de parâmetros proprietários do fabricante da bateria: área da célula, porosidade do eletrodo, densidade do material, características do eletrólito, condutividade térmica, etc. Esta dificuldade pode ser eliminada por meio da caracterização da bateria utilizando-se câmara térmica e termopares. Mas esta solução é cara, demanda tempo e introduz outros desafios como a implementação de sistemas de expurgo de ar seco, ventilação, segurança, suprimento de ar e água, etc. Modelos eletroquímicos demandam o uso de sistemas de computação intensiva (10).

Por outro lado, a abordagem baseada em ECM tem sido utilizada para análise computacional/numérica de baterias (10). Neste caso, o objetivo é desenvolver um modelo elétrico que represente o fenômeno eletroquímico existente na célula. O nível de complexidade do modelo é o resultado de uma solução de compromisso entre precisão e esforço computacional. Note-se que um ECM extremamente complexo e preciso pode ser inadequado para aplicação em sistemas embarcados.

A literatura recente sugere que a abordagem de *machine learning* (aprendizado de máquina), baseada em algoritmos de redes neurais profundas (aprendizado profundo ou DL) é o estado da arte na área (11), (14), (15),(16). O aprendizado de máquina é um ramo da IA (17).

Chemali et al (14) mostram que as redes neurais profundas alimentadas para frente ou DFNN permitem que o comportamento de uma célula de íons de lítio em diferentes

[3] Os dados são coletados do dispositivo.

temperaturas seja modelado pelos pesos de uma única rede neural, a qual consegue estimar com acurácia e robustez o SOC em uma dada faixa de temperatura (ex.: 0^o C a 40^oC). Note-se que a estratégia de ECM não possui esta flexibilidade. O ECM de uma célula a 20^o C é diferente do ECM da mesma célula a 40^o C.

Lipu et al (18) apontam as seguintes **vantagens para os métodos de modelagem *data-driven***:

- não requerem um conhecimento abrangente do domínio;
- não requerem um modelo matemático;
- o tempo de desenvolvimento do algoritmo é menor; e
- possuem excelente acurácia desde que se disponha de um conjunto de dados para treinamento de boa qualidade.

1.4 Escopo

Esta pesquisa teve como motivação o estudo da aplicação de sistemas de EES na área de fontes de energia renováveis. A princípio, conforme consta do projeto de pós-doutorado aprovado, doravante designado simplesmente por projeto, o trabalho seria desdobrado em duas fases: a) desenvolvimento de uma bancada virtual representativa das células (*battery pack*) e do sistema de monitoramento de um EES com bateria de íons de lítio e b) construção, teste e validação da bancada real.

Contudo, a pesquisa realizada ao longo dos doze meses de duração do projeto mostrou que o objetivo principal do trabalho deveria ser redirecionado para **o desenvolvimento de novos algoritmos *data-driven* do tipo DL para estimação do SOC de células de íons de lítio**, o que efetivamente foi feito, uma vez que o SOC é o parâmetro de estado mais crítico de uma bateria (19).

Ressalta-se que a estimação do SOC das células de íons de lítio de um BMS por meio de redes neurais artificiais profundas oferece pelo menos três vantagens significativas sobre a que é orientado ao modelo (*model driven*), quais sejam:

1. as redes neurais conseguem estimar com grande precisão a dependência funcional não linear que existe entre tensão, corrente e temperatura (quantidades observáveis) e quantidades não observáveis, como o SOC;
2. o treinamento de uma única rede neural com múltiplas camadas consegue modelar o comportamento de uma célula de íons de lítio em uma dada faixa de temperaturas; e
3. evita-se o problema da identificação dos parâmetros do ECM.

Os algoritmos de aprendizado supervisionado de máquina resultantes do projeto foram treinados utilizando-se os dados da bateria de íons de lítio Panasonic 18650PF como conjunto de treinamento (*dataset*) (20). Este *dataset* contém uma série de dez ciclos de condução de veículo: Ciclo 1, Ciclo 2, Ciclo 3, Ciclo 4, US06, HWFTa, HWFTb, UDDS, LA92 e Rede Neural (NN). Os ciclos 1-4 consistem em uma combinação aleatória de US06, HWFET, UDDS, LA92 e ciclos de condução NN. O perfil de potência do ciclo de condução é calculado a partir da medição de um caminhão elétrico Ford F150 com uma bateria de 35 kWh dimensionada para uma única célula 18650PF.

Os artigos científicos e "preprints" eletrônicos listados abaixo foram publicados e/ou submetidos para periódicos/conferências no decorrer do projeto:

- "State-of-Charge Estimation of a Li-Ion Battery using Deep Forward Neural Networks" (set/2020), vide apêndice A (21);
- "State-of-charge Estimation of a Li-ion Battery using Deep Learning and Stochastic Optimization" (nov/2020), vide apêndice B (22);
- "State-of-Charge Estimation of the Panasonic 18650PF Li-ion Cell using Deep Learning Models and Algorithms with Adaptive Learning Rates" (dez/2020), vide apêndice C (23);
- "Data-driven state-of-charge estimation of the Panasonic 18650PF Li-ion cell using deep forward neural networks" (2021[4]), vide apêndice D (24); e
- "The relevance of the optimization algorithm on the data-driven estimation of the state-of-charge of the Panasonic 18650PF lithium-ion cell using deep feedforward neural networks" (abr/2021)[5], vide apêndice E.

1.5 Método do Trabalho

O cerne do projeto é a estimação do SOC de células de íons de lítio por meio de algoritmos de DL.

A revisão de literatura efetuada mostrou que a linguagem de programação Python é a mais indicada para esta pesquisa, uma vez que possui vários *frameworks* gratuitos e de código aberto (*opensource*) para DL e que essa linguagem tem sido muito usada pela comunidade de ML (25), (26), (27). Não obstante, outras linguagens e ambientes também possuem bibliotecas de DL como R (28) e MATLAB (29).

[4] Aceito para a conferência virtual INDUSCON 2021, que será realizada na EPUSP em agosto de 2021.
[5] Submetido para a conferência virtual "FTC 2021 - Future Technologies Conference 2021", Vancouver, Canadá, 28–29 de outubro de 2021. Coautores: Prof. Mauricio Barbosa Camargo Sales e Prof. José Roberto Cardoso.

Frameworks open source para ML como TensorFlow (30), PyTorch (31), MXNet (32) e Microsoft Cognitive Toolkit (CNTK) (33)[6] têm se destacado nos últimos anos. Contudo, o TensorFlow e a *Application Programming Interface* (API) Keras[7] oferecem alguns diferenciais, tais como a possibilidade de execução dos códigos no Google Colaboratory, ou "Colab" (35), usando o navegador (*browser*) e com acesso gratuito a *Graphics Processing Unit* (GPU)[8]. Além disso, os códigos Python armazenados na plataforma GitHub (36) com extensão `.ipynb` podem ser executados no Colab, o que sem dúvida facilita um eventual compartilhamento dos resultados do trabalho. A extensão `.ipynb` é usada pelo Colab e pelo *Integrated Development Environment* (IDE) Jupyter Notebook (37), pois ambos utilizam o interpretador interativo IPython como *kernel* (38). Note-se que todo código Python (extensão `.py`) é executável no *shell* IPython.

Segundo Chollet (25) [p. 61],

> "Keras possui os seguintes recursos principais:
>
> - Permite que o mesmo código seja executado perfeitamente na *Central Processing Unit* (CPU) ou GPU.
> - Possui uma API amigável que facilita a prototipagem rápida de modelos de aprendizado profundo.
> - Possui suporte embutido para redes convolucionais (para visão computacional), redes recorrentes (para processamento de sequências) e qualquer combinação de ambas.
> - Suporta arquiteturas de rede arbitrárias: modelos de múltiplas entradas ou saídas, compartilhamento de camadas, compartilhamento de modelos e assim por diante. Isso significa que o Keras é apropriado para criar essencialmente qualquer modelo de aprendizado profundo, de redes adversárias generativas (39) a máquinas neurais de Turing (40)." (tradução livre)

Além disso, o TensorFlow oferece uma ferramenta de visualização baseada em *browser* denominada TensorBoard, cujo principal objetivo é ajudar o usuário a monitorar visualmente tudo o que acontece dentro do seu modelo durante o treinamento (25). O TensorBoard automatiza algumas funcionalidades como a visualização da curva de aprendizado das RNA.

Assim, optou-se pela utilização da linguagem Python e do *framework* TensorFlow 2 em conjunto com a API Keras.

[6] A biblioteca Caffe2 (34) foi absorvida pelo PyTorch.
[7] Segundo (26), são os *frameworks* mais utilizados pela comunidade de DL. O Keras faz parte do TensorFlow 2.0.
[8] Por até doze horas.

Os códigos foram executados em um laptop Lenovo Legion 5i equipado com processador Intel Core i7, 16GB de memória RAM, placa dedicada (GPU) NVIDIA GeForce RTX 2060 6GB e armazenamento 1 TB + 128GB SSD PCIe.

Os modelos de DL foram desenvolvidos por meio da ferramenta computacional Anaconda[9] utilizando-se o IDE Spyder 4.0 e a linguagem Python 3.7.7 (41).

Os códigos-fonte Python (extensão `.py`) desenvolvidos serão entregues ao supervisor do projeto, tendo em vista os direitos de propriedade intelectual da Universidade.

1.6 Estrutura do Relatório

A fundamentação teórica do trabalho é apresentada no capítulo 2. Os resultados obtidos pelo projeto são apresentados pelas seções intituladas *Experimental Results* dos apêndices A, B, C, D e E (artigos científicos e "preprints" eletrônicos publicados/submetidos durante a pesquisa). O capítulo 3 sumariza os resultados obtidos e lista algumas sugestões para trabalhos futuros.

[9] O Anaconda é uma ferramente computacional gratuita e de fácil instalação que permite gerenciar distribuições de Python, ambientes de trabalho e módulos nos sistemas operacionais Windows, Mac OSX e Linux. Alguns dos gráficos aplicativos Python mais populares já estão instalados ou estão disponíveis para instalação no Anaconda, como os IDE Spyder e Jupyter Notebook (em que o IDE é montado no *browser*).

2 Modelagem Orientada por Dados

2.1 Aprendizado de Máquina

Augusta Ada Lovelace[1] (1815 – 1852), a primeira programadora de computadores[2] (17), especulou sobre a possibilidade de IA aproximadamente um século antes de o primeiro calculador eletrônico digital ser desenvolvido por John P. Eckert e John Mauchly, o *Electronic Numerical Integrator and Computer* (ENIAC)[3], em 1946, nos EUA, a pedido do Exército Norte-Americano (43).

Nos primórdios da IA (segunda metade da década de 1950), tarefas abstratas e que podem ser bem descritas pela Matemática, mas que não requerem conhecimento sobre o mundo real, como "jogar xadrez", impulsionaram o desenvolvimento na área.

Contudo, tarefas "fáceis" para seres humanos, tais como reconhecer emoções faciais e análise de sentimentos em sinais de voz, são "difíceis" para os computadores, pois demandam que eles tenham algum conhecimento do mundo físico (44).

A solução para os problemas "difíceis" de IA consiste em permitir que os computadores aprendam com a experiência e que consigam formular algum tipo de modelo do mundo exterior a partir de uma hierarquia de conceitos. Esta abordagem, denominada aprendizado de máquina ou ML, tem como grande vantagem evitar que todo o conhecimento a ser adquirido pela máquina tenha que ser formalmente especificado pelo homem (17). Segundo Arthur Samuel, um dos pais da IA (45),

> "*Machine Learning* é a área do conhecimento que dá aos computadores a habilidade de aprender sem serem explicitamente programados."

Tom Mitchell oferece uma definição formal de ML (45):

> "Um computador APRENDE a partir da sua EXPERIÊNCIA (E) com relação a alguma TAREFA (T) e alguma MEDIDA DE DESEMPENHO (P), se o seu desempenho em T, medido por P, melhora com a sua experiência E".

[1] Seu nome de solteira era Augusta Ada Byron, pois era filha do ilustre poeta inglês Lord Byron.

[2] Para maiores detalhes da contribuição que Lady Lovelace deu para a Computação, recomenda-se a leitura do artigo (42), que aborda a história da descrição da Máquina Analítica de Charles Babbage de autoria de Lady Lovelace, o famoso "*Notes by A. A. L.*".

[3] No ENIAC, a programação era feita por meio da fiaação do painel. Até então não havia o conceito de programa armazenado. John von Neumann propôs uma arquitetura que transformou os calculadores eletrônicos da década de 1940 (como o ENIAC) nos computadores da era contemporânea. Tal arquitetura possui três componentes principais: uma unidade central de processamento (CPU) para processamento das instruções codificadas usando 0's e 1's (codificação binária), memória (para armazenamento das instruções e de dados) e sistemas de Entrada/Saída (*Input*/*Output*).

O jogo de damas é um problema que pode ser resolvido por meio de algoritmos de ML. De acordo com a definição, tem-se que:

- E = "treinamento" do programa, ou seja, o programa deve jogar um número suficientemente grande[4] de partidas de jogos de damas.
- T = jogar uma nova partida.
- P = probabilidade de ganhar a próxima partida contra um novo oponente.

Para a grande maioria dos algoritmos de ML, o homem fornece os dados de entrada (conjunto de treinamento) bem como as respostas esperadas a partir dos dados. A saída do processamento é um conjunto de regras, que serão aplicadas sobre novos dados de entrada, de forma a produzir respostas originais. Um sistema de ML é treinado ao invés de ser explicitamente programado para uma tarefa. A Fig. 1 ilustra os dois paradigmas de programação, o clássico (diagrama superior), usado pela IA simbólica, e o de ML (diagrama inferior).

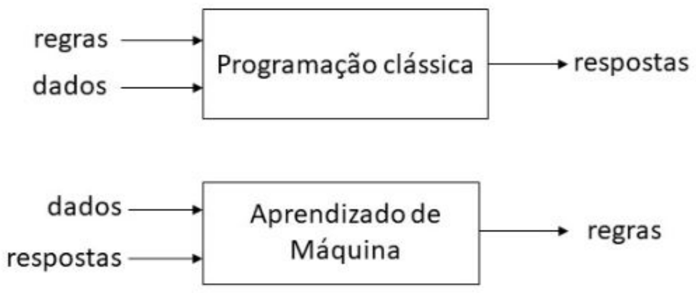


Figura 1 – Paradigmas de programação.

Segue-se uma lista com alguns algoritmos importantes de ML (46):

- *nearest neighbor* (vizinho mais próximo);
- *decision trees and rules* (árvores de decisão e regras);
- regressão;
- DL (usando RNA);
- *Support Vector Machine* (SVM);
- regras de associação; e
- *clustering with k-means.*

[4] Por exemplo, centenas de milhares de jogos.

De acordo com (47),

> "Os algoritmos de aprendizagem profunda (DL) são o foco central dos sistemas modernos de aprendizado de máquina. À medida que os volumes de dados continuam crescendo, tornou-se habitual treinar grandes redes neurais com centenas de milhões de parâmetros com capacidade suficiente para memorizar esses volumes e obter precisão estado da arte." (tradução livre)

Os algoritmos de ML podem ser divididos em quatro grupos (25):

- **Aprendizado Supervisionado** (*Supervised Learning*), em que o homem fornece os dados de entrada (conjunto de treinamento), bem como as respostas esperadas a partir dos dados (associa-se um rótulo para cada dado). A saída do processamento é um conjunto de regras, que serão aplicadas sobre novos dados de entrada, de forma a produzir respostas originais (inferências do tipo classificação ou regressão). Esta foi a técnica empregada por este projeto.
- **Aprendizado Não Supervisionado** (*Unsupervised Learning*), para treinamento de um modelo descritivo capaz de reconhecer padrõoes. Não há um professor. Exemplos: i) descobrir padrões de compra em supermercados e ii) identificar grupos de pessoas que compartilham interesses comuns tais como esportes, religião ou música (tarefas de agrupamento ou *clustering* de dados).
- **Aprendizado Auto-Supervisionado** (*Self-supervised Learning*), é um tipo de aprendizagem supervisionada, mas os rótulos são gerados a partir dos dados de entrada por meio de um algoritmo.
- **Aprendizado por Reforço** (*Reinforcement Learning*), em que um programa é treinado a interagir em um ambiente por meio de ações para atingir um objetivo. O aprendizado usa um mecanismo de recompensas e/ou punições. O algoritmo AlphaZero[5], desenvolvido pelo Google DeepMind, utiliza esta técnica (48).

Por exemplo, considere o problema da previsão do preço de um imóvel em reais (variável dependente y) dada a área construída em m^2 (variável independente, explanatória ou regressor x). Neste caso, o conjunto de treinamento poderia consistir em uma base de dados com os preços de venda de milhares de imóveis e respectivas áreas construídas coletados na cidade de São Paulo nos últimos doze meses. Este problema pode ser resolvido utilizando-se uma Regressão Linear Simples (RLS). Note-se que a previsão (predição ou saída) é uma variável contínua, ou seja y é um número real.

[5] O AlphaZero derrotou o campeão mundial do jogo Go, Lee Sedol, em um desafio patrocinado pelo Google em 2016, na Coréia do Sul.

Há casos em que a saída (variável y) é uma variável discreta (ex.: 0 ou 1). Considere o problema de classificar um câncer de mama dado o tamanho do tumor. A resposta (saída) é “sim” ou “não”, que pode ser associada aos bits 1 ou 0, respectivamente. Em ML, diz-se que o tamanho do tumor (variável x) é uma *feature* (atributo ou característica).

Outros atributos do câncer de mama seriam: espessura dos grupos, uniformidade do tipo de célula, etc.

Em suma, a regressão prevê o valor de uma variável contínua, enquanto que a classificação prediz o valor de uma variável discreta.

Note-se que a modelagem *data-driven* de baterias de íons de lítio por meio de algoritmos de ML é um problema de regressão não linear, haja vista que tais baterias apresentam características de sistemas não lineares (14), (18). Os atributos de entrada podem ser tensão, corrente e temperatura da bateria, por exemplo.

Suponha uma Regressão Linear Múltipla (RLM) em que as variáveis de entrada são x_1 e x_2 e a variável de saída é y. Associe a variável y com o preço previsto de um imóvel em reais dadas a idade da construção (variável x_1) e a área construída em m^2 (variável x_2). Admita ainda que se possua um banco de dados com 50 preços em reais de imóveis em função da idade e da área construída. Em ML, adota-se a seguinte notação para este problema:

- $m = 50$ corresponde ao número de exemplos de treinamento;
- x_1 e x_2 denotam as variáveis de entrada ou *features*;
- y representa a variável de saída;
- (x_1, x_2, y) denota um exemplo de treinamento;
- $(x_1^{(i)}, x_2^{(i)}, y^{(i)})$ é o i-ésimo exemplo de treinamento; e
- $h_\Theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 = y(x)$ é a hipótese (ou modelo) e $\Theta = [\theta_0, \theta_1, \theta_2]^T$ é o vetor de parâmetros a serem estimados.

A RLM acima pode ser resolvida por um método numérico de otimização denominado algoritmo do gradiente, cujo objetivo é estimar os valores de θ_0, θ_1 e θ_2 que minimizam as distâncias entre os exemplos de treinamento $y^{(i)}$ e os valores previstos pelo modelo $\hat{y}^{(i)} = h_\Theta^{(i)}(x)$, $i = 1, 2, \ldots, m$:

$$e^{(i)} = y^{(i)} - \hat{y}^{(i)} \tag{2.1}$$

A variável $e^{(i)}$ também é designada por erro de estimação.

O algoritmo do gradiente tem como objetivo encontrar o mínimo de uma função, designada por função custo J (função de perda ou *loss function*), usando um esquema iterativo, onde em cada passo de iteração se toma a direção negativa do gradiente da função, a qual corresponde à direção de declive máximo. A água que desce morro abaixo, por efeito da gravidade, segue o método da descida pelo gradiente, uma vez que o curso d'água segue pelo caminho de declividade máxima. Em inglês, o algoritmo do gradiente é comumente chamado de *gradient descent* (*Stochastic Gradient Descent* (SGD)).

É usual que o ajuste dos exemplos de treinamento seja realizado pelo método dos mínimos quadrados, segundo o qual a hipótese a ser adotada é aquela que torna mínimo o erro quadrático médio (*Mean Square Error* (MSE)) de estimação (função custo). Ou seja, o algoritmo do gradiente busca minimizar a média quadrática $E(\,e^2)$

$$J(\Theta) = E(\,e^2) = \frac{1}{2m}\sum_{i=1}^{m}\left[h_\Theta(x^{(i)}) - y^{(i)}\right]^2 \tag{2.2}$$

em que a função custo $J(\Theta)$ é dividida por 2 por convenção.

```
repeat
    $\theta_j \leftarrow \theta_j - \eta \frac{\partial}{\partial \theta_j} J(\theta_j), \quad j = 0, 1, 2$
until atingir a convergência;
```

Algoritmo 1: Algoritmo do gradiente em RLM para $\Theta = [\theta_0, \theta_1, \theta_2]^T$.

O algoritmo do gradiente é dado acima, em que η denota o passo de adaptação ou taxa de aprendizado (*learning rate*).

As RNA são ferramentas indicadas para implementação de regressão não linear multivariada (17), (49). Assim, justifica-se a abordagem utilizada neste trabalho.

O aprendizado supervisionado utilizado neste projeto, também designado por aprendizado profundo ou DL (43), envolveu o treinamento de redes neurais com múltiplas camadas (ou profundas).

2.2 A Noção de Redes Neurais

O *The Handbook of Artificial Intelligence* apresenta a seguinte definição operacional[6] para IA (50):

"*Artificial Intelligence (AI) is the part of computer science concerned with designing intelligent computer systems, that is, systems that exhibit the characteristics we associate with intelligence in human behavior - understanding language, learning, reasoning, solving problems, and so on.*"

"Inteligência Artificial (IA) é a parte da ciência da computação preocupada em projetar sistemas computacionais inteligentes, isto é, sistemas que exibem

[6] Não há consenso sobre a definição de IA.

as características que associamos à inteligência no comportamento humano - compreensão da linguagem, aprendizagem, raciocínio, resolução de problemas e assim por diante." (tradução livre)

Existem duas linhas principais de pesquisa em IA: a conexionista e a simbólica. De acordo com Boden (51), ambas foram inspiradas no artigo seminal intitulado *A Logical Calculus of the Ideas Immanent in Nervous Activity* (1943), de Warren Sturgis McCulloch e Walter Pitts (52), a primeira teoria computacional moderna[7]. A literatura reconhece que a pesquisa realizada por McCulloch e Pitts é o trabalho pioneiro em IA (17).

A abordagem conexionista usa RNA. A linha simbólica, às vezes chamada de IA simbólica (51), segue a tradição logicista e teve John McCarthy e Allen Newell, entre outros, como alguns de seus grandes expoentes (54).

A Fig. 2 ilustra o Modelo de neurônio artificial McCulloch e Pitts (MCP) (52).

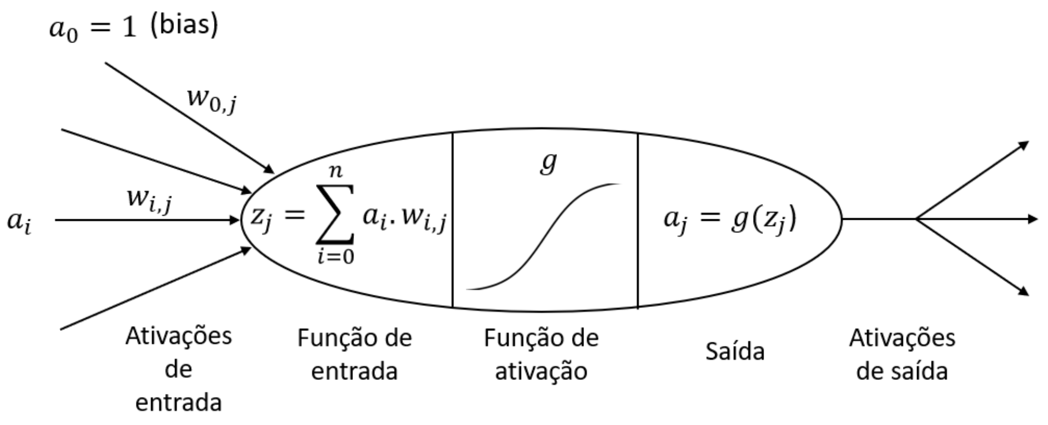


Figura 2 – Modelo de McCulloch e Pitts

O neurônio (ou unidade) artificial da Fig. 2 possui as seguintes características:

- sinais (ativações) de entrada $(a_1, a_2, \ldots, a_n)$ são bits 0 ou 1. O sinal externo $a_0 = 1$ é o *bias*;
- pesos sinápticos do j-ésimo neurônio: $w_{0,j}, w_{1,j}, \ldots, w_{n,j}$; e
- a_j denota o sinal de saída (ou ativação de saída) do j-ésimo neurônio, dado por:

$$z_j = \sum_{i=0}^{n} w_{i,j} a_i \tag{2.3}$$

$$a_j = g(z_j) = \{0, 1\} \tag{2.4}$$

[7] A teoria é considerada moderna porque emprega a noção matemática de computação estabelecida por Alan Turing em artigo publicado em 1936 (53).

em que $g(z)$ é uma função de ativação não linear. A saída é binária (bit 0 ou bit 1); por conseguinte, diz-se que o MCP tem a propriedade "tudo ou nada".

A função de ativação do MCP é a função de Heaviside (função degrau unitário)

$$g(z) = \begin{cases} 1, & z \geq 0 \\ 0, & z < 0 \end{cases} \tag{2.5}$$

Atualmente, as redes neurais usam outras funções de ativação, que viabilizam o treinamento da rede por meio do algoritmo de retropropagação do erro (*backpropagation* ou *"backprop"*) (55), (56), (57) , (58)[8], tais como a função de ativação designada por REctified Linear Unit (Relu), dada por

$$g(z) = \max\{0, z\}. \tag{2.6}$$

A Fig. 3 ilustra a função Relu.

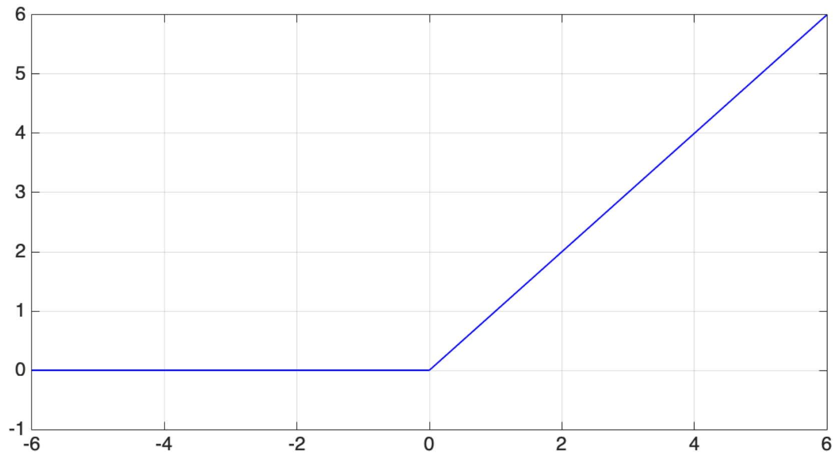


Figura 3 – função Relu.

É importante ressaltar que o papel do algoritmo *backpropagation* é calcular o gradiente em RNA com múltiplas camadas (43), enquanto que outro algoritmo, como o SGD, é o responsável pela atualização dos parâmetros da RNA.

Uma RNA é um sistema de processamento paralelo distribuído, inspirado na estrutura de processamento do cérebro humano. A Fig. 4 mostra um exemplo de rede neural densamente conectada com uma camada oculta (*hidden layer*). A técnica de RNA é uma forma de computação não algorítmica de funções.

Na Fig. 4, o vetor $\boldsymbol{x} = [x_0\, x_1\, x_2\, x_3]^T$, em que $x_0 = 1$ é o *bias*, contém os atributos ou sinais de entrada (camada de entrada) x_1, x_2, x_3. A camada intermediária, designada

[8] O algoritmo de retropropagação foi descoberto de forma independente várias vezes.

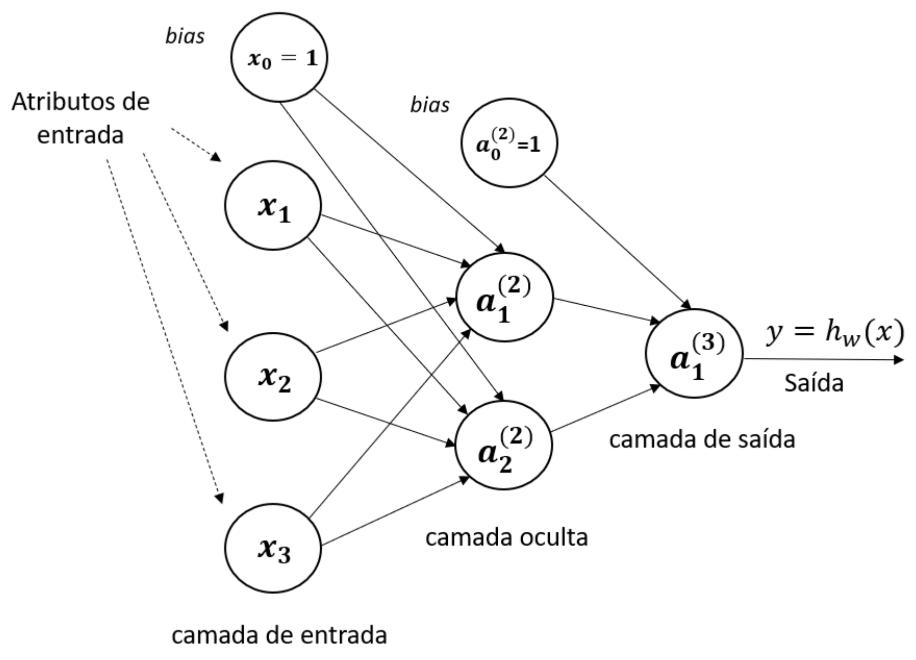


Figura 4 – Exemplo de RNA com uma camada oculta.

por oculta, possui três unidades ($a_0^{(2)}$, $a_1^{(2)}$ e $a_2^{(2)}$), em que $a_0^{(2)} = 1$ é o *bias*. A camada de saída possui somente a unidade $a_1^{(3)} = h_w(x)$.

Antes de prosseguir, é necessário apresentar a intuição por trás do fato de que as modernas RNA são capazes de aproximar funções não lineares com precisão arbitrária, pelo menos em teoria (teorema da aproximação universal (59)). A explicação a seguir considera uma rede com apenas uma camada de $N + 1 = M$ neurônios em paralelo, onde cada neurônio é excitado pelo sinal de entrada $\boldsymbol{a} = \{a_0, a_1, \ldots, a_N\}$.

Reescrevendo (2.3) na forma vetorizada[9]

$$z_i = \boldsymbol{W}^T \boldsymbol{a} \tag{2.7}$$

em que se adotou a notação $\boldsymbol{W}$ para o vetor de pesos sinápticos, $\boldsymbol{a}$ para o vetor de entradas e T denota a transposição de matrizes[10].

Considere a Transformada Discreta de Fourier (TDF) de um sinal de entrada $\boldsymbol{b} = \{b_0, b_1, \ldots, b_N\}$ dada por[11]

$$B[k] = \begin{cases} \sum\limits_{i=0}^{N} (e^{-j2\pi \frac{k}{M} i}) b_i, & 0 \leq k \leq N \\ 0, & \text{para os demais valores} \end{cases} \tag{2.8}$$

[9] O índice j da variável z foi trocado por i por conveniência.
[10] Assume-se que os vetores são do tipo coluna, como é usual na literatura de processamento de sinais (60).
[11] A unidade imaginária é denotada por j.

e a transformação inversa correspondente, chamada Transformada Discreta Inversa de Fourier (TDIF) (61)

$$z_i = \begin{cases} \sum_{k=0}^{N} (\frac{1}{M} e^{j2\pi \frac{k}{M} i}) B[k], & 0 \leq k \leq N \\ 0, & \text{para os demais valores} \end{cases} \tag{2.9}$$

Reescrevendo (2.9) na forma vetorizada, obtém-se

$$z_i = \boldsymbol{W}^T \boldsymbol{B} \tag{2.10}$$

em que $\boldsymbol{W} = \{(\frac{1}{M} e^{j2\pi \frac{k}{M} i})\}, 0 \leq k \leq N$, e $\boldsymbol{B} = \{B_0, B_1 \ldots B_N\}$.

Compare (2.10) e (2.7). Observe que essas equações são iguais se $\boldsymbol{a} = \boldsymbol{B}$. Portanto, a Eq. (2.10) sugere que seria possível representar a função $z = f(a)$ através de uma rede neural que usa os pesos $\{\frac{1}{M} e^{j2\pi \frac{k}{M} i}\}, 0 \leq k \leq N$. A Eq. (2.10) é parecida com uma série de Fourier.

Fourier mostrou em 1807 que uma função arbitrária e aperiódica $f(t)$ definida em um intervalo finito T_0 pode ser reconstruída a partir de uma série trigonométrica chamada série de Fourier (62). Portanto, "não há nada de novo sob o sol ". Os engenheiros eletricistas estão bem familiarizados com essa noção.

O teorema da aproximação universal afirma que uma rede *feedforward* com uma entrada linear e pelo menos uma camada oculta de unidades artificiais com uma função de ativação não linear pode aproximar qualquer "função"[12] de um espaço de dimensão finita para outro com erro insignificante, desde que a rede receba um número suficiente de unidades (43), (59). No entanto, o teorema não diz qual deve ser o número de unidades na camada oculta. Também não se tem garantia de que o algoritmo de treinamento será capaz de aprender essa função. Isso poderá ocorrer devido à existência de mínimos locais na função de custo a ser otimizada.

2.3 Aprendizado Profundo

Muitos pensam que DL é uma nova tecnologia. Contudo, isto não corresponde à realidade dos fatos. O DL surgiu na década de 1940 do século passado, mas não com esse nome (43).

Segundo Goodfellow, Bengio e Courville (43), houve três ondas de pesquisa e desenvolvimento em DL:

1. O DL com o nome de cibernética nos anos 1940-1960.

[12] Na verdade, qualquer função de Borel. Este conceito está além do escopo deste trabalho e não será discutido.

2. O DL conhecido como conexionismo (*connectionism*) nas décadas de 1980 e 1990.
3. O DL atual, que ressurgiu em 2006, com o trabalho de Hinton, Osindero e Teh (63).

Os primeiros algoritmos de DL (os da primeira onda) eram inspirados em modelos computacionais de aprendizado biológico, ou seja, modelos de como o aprendizado ocorre em um cérebro biológico. Tais algoritmos foram designados por RNA.

Goodfellow, Bengio e Courville destacam que (43, p. 14),

> "*The modern term deep learning goes beyond the neuroscientific perspective on the current breed of machine learning models. It appeals to a more general principle of learning multiple levels of composition, which can be applied in machine learning frameworks that are not necessarily neurally inspired*".
>
> "A terminologia moderna aprendizagem profunda vai além da perspectiva neurocientífica da atual geração de modelos de aprendizado de máquina. Apela para um princípio mais geral de aprendizagem de múltiplos níveis de composição, que pode ser aplicado em estruturas de aprendizado de máquina que não são necessariamente inspiradas em estruturas neuronais biológicas." (tradução livre)

Na década de 1980, a segunda onda das redes neurais emergiu de um movimento científico denominado Conexionismo (paradigma conexionista) ou processamento paralelo distribuído, que surgiu no contexto das ciências cognitivas.

Segundo o Conexionismo, um número grande de unidades computacionais simples pode atingir um comportamento inteligente quando interligadas em rede.

A segunda onda das RNA durou até meados dos anos 1990. Naquela época, o treinamento de uma RNA era extremamente custoso do ponto de vista computacional. Por conseguinte, o mercado perdeu o interesse pela tecnologia, pois era inviável do ponto de vista econômico.

A terceira onda das RNA (a atual) é oriunda de uma contribuição seminal dada pelo grupo de pesquisa liderado por Geoffrey Hinton, da Universidade de Toronto, em 2006. Hinton, Osindero e The demonstraram que um tipo de rede neural denominada *deep belief* poderia ser treinada com alta eficiência por meio da estratégia *greedy layer-wise pretraining* (63). O artigo de Hinton, Osindero e Teh disparou uma nova onda de pesquisas em RNA, a qual popularizou o termo *Deep Learning*.

O termo *deep* (profundo) em DL denota a ideia de um modelo em rede que possui a capacidade de particionar a representação de uma entrada em múltiplas camadas. O número de camadas de um modelo corresponde à profundidade da rede (64).

A Fig. 5 mostra um modelo de DL do tipo DFNN com seis (6) camadas para classificação de dígitos manuscritos. Reconhecer dígitos escritos à mão é um problema importante em muitas aplicações de visão computacional, tais como classificação automática de correspondências por código postal e leitura automática de cheques (17). Yan LeCun, prêmio Turing de 2018, é um dos pioneiros na área (65), (66), (67).

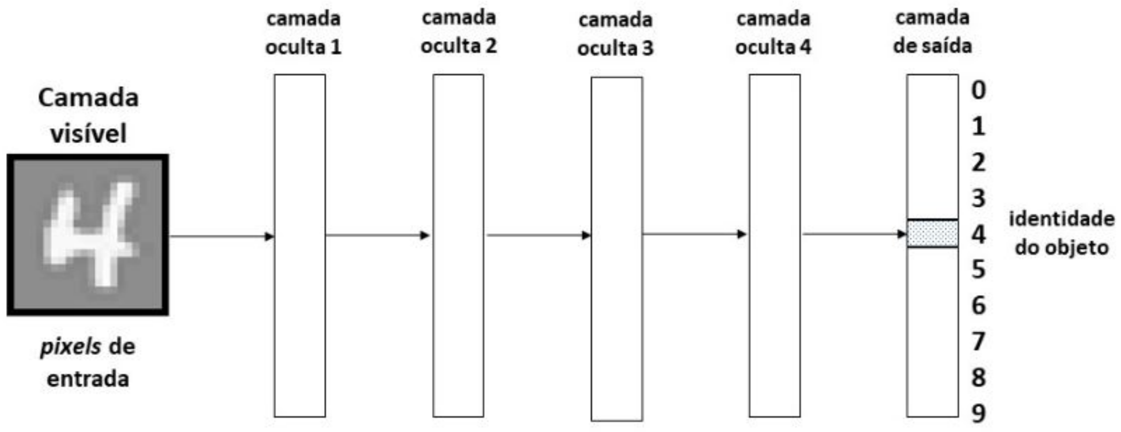


Figura 5 – DFNN com seis camadas.

A função (modelo ou hipótese) que mapeia o conjunto de *pixels* de entrada em um dado objeto (no caso o dígito 4) é composta pelas últimas cinco (5) camadas. Tal modelo, além de ser não linear, é extremamente complicado. A estratégia do DL é desdobrar esse modelo não linear/complexo em uma cascata de transformações sucessivas de menor complexidade, de modo que cada camada do DL execute uma operação de transformação ou filtragem mais simples. Por exemplo, a primeira camada oculta da Fig. 5 poderia identificar bordas na imagem, ao passo que à segunda camada oculta caberia identificar cantos e contornos. Os algoritmos de classificação dos dígitos são executados pelos neurônios (unidades) da camada de saída.

A Fig. 6 mostra a relação entre IA, ML e DL.

A Fig. 7 ilustra como funciona o DL (25). Na inicialização, valores aleatórios são atribuídos aos pesos $\boldsymbol{w}$ da rede; logo, o valor inicial do resíduo é alto. No entanto, no decorrer do treinamento da rede, os pesos são ajustados incrementalmente na direção correta; ao mesmo tempo, o valor da função custo diminui. Este é o *loop* de treinamento, que, sendo repetido o suficiente, normalmente dezenas de iterações (mini-bateladas ou *mini-batches*) em milhares de exemplos, produz pesos que minimizam a função de custo. Uma rede é considerada treinada quando o mínimo da função custo é atingido.

O DL alcançou os seguintes avanços, todos em áreas historicamente difíceis de ML (64), (68):

- classificação de imagem super-humana;
- reconhecimento de voz em nível quase humano;

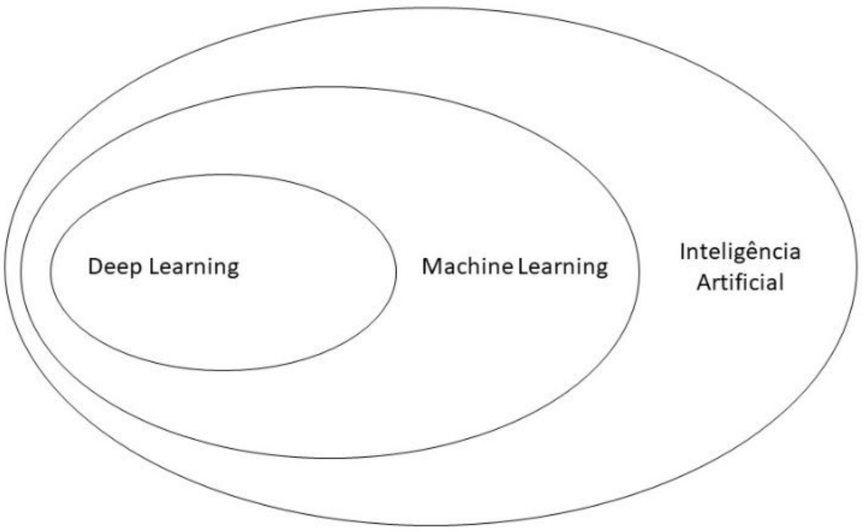


Figura 6 – relação entre IA, ML e DL.

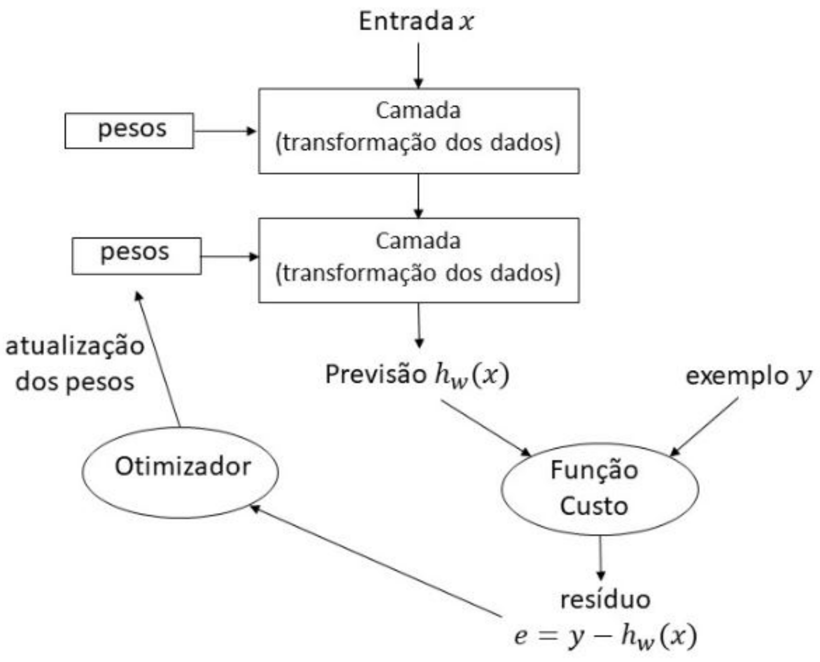


Figura 7 – Como funciona o DL.

- transcrição de escrita à mão quase humana;
- tradução automática;
- conversão de texto em fala;
- assistentes digitais como a Alexa (Amazon);
- condução autônoma de veículo ao nível quase humano;

- segmentação de anúncios (usada por empresas como Google, Baidu e Bing); e
- capacidade de responder perguntas em linguagem natural; e
- desempenho super-humano em jogos como xadrez, shogi e Go.

No entanto, o DL possui limitações. As arquiteturas atuais de RNA não têm o poder de rastrear flutuações randômicas dos dados de treinamento em tempo real. Em outras palavras, o aprendizado profundo ainda não é adaptativo. Observe que o treinamento de um modelo de DL em uma estação de trabalho dedicada pode levar dias ou semanas. Isso se deve à sua complexidade computacional.

Além disso, o DL não é como a matemática ou a física, nas quais avanços teóricos podem ser alcançados com giz e quadro-negro. O DL, assim como a IA, é uma ciência empírica (69), pois a nossa compreensão de por quê e quando o DL funciona permanece limitada. Por exemplo, não há critério de projeto para o número de camadas na rede, muito menos para o número de neurônios em uma camada oculta. O campo é impulsionado por descobertas experimentais. Mas é claro, existem melhores práticas a serem seguidas.

2.4 Métodos de Estimação *Data-driven* de SOC

Com base na estrutura, computação e processo de execução, Lipu et al (18) classificam os algoritmos de estimação *data-driven* de SOC em seis tipos principais (os sub-tipos também estão indicados):

- rede neural *feed-forward* ou DFNN:
  - rede neural profunda (*Deep Neural Networks* (DNN));
  - rede neural de função de base radial;
  - *extreme learning machine* (máquina de aprendizado extremo);
  - rede neural *wavelet*; e
  - rede neural *time-delay*.
- classificação e regressão:
  - SVM; e
  - *random forest*.
- probabilísticos:
  - regressão do processo gaussiano; e
  - rede *Deep Belief*.

- redes neurais recorrentes (*Recurrent Neural Networks* (RNN)):
  - *non-linear autoregressive with exogenous inputs* (não linear autorregressiva com entradas exógenas);
  - *Long Short-Term Memory* (LSTM); e
  - *Gated Recurrent Unit* (GTU).
- algoritmo baseado em regras:
  - sistema de inferência adaptativa Neuro-Fuzzy; e
  - lógica Fuzzy.
- algoritmo híbrido:
  - baseado em otimização híbrida; e
  - *hybrid DL* (algoritmo de DL híbrido).

Chemali et al (14) comparam o desempenho das DFNN com os de outros algoritmos relevantes que foram propostos desde a segunda metade dos anos 2.000. O artigo mostra que o erro de estimação do SOC obtido com o algoritmo de DFNN proposto pelos autores é menor que o dos seguintes métodos:

- *Model Adaptive-Improved Extended Kalman Filter* (EKF) (70);
- *Adaptive* EKF *with Neural Networks* (71);
- *Adaptive Unscented Kalman Filter* (AUKF) *with Extreme Machine Learning* (72);
- *Fuzzy Neural Networks* (NN) *with Genetic Algorithm* (73); e
- *Radial Bias Function* NN (74).

Além disso, Chemali et al (14) mostram que as DFNN permitem que o comportamento de uma célula de íons de lítio em diferentes temperaturas seja bem modelado pelos pesos de uma única rede neural, a qual consegue estimar com grande acurácia e robustez o SOC em uma dada faixa de temperatura (ex.: 0^o C a 40^o C). Note-se que a estratégia de ECM não possui esta flexibilidade. O ECM de uma célula a 20^o C é diferente do ECM da mesma célula a 40^o C.

Diante do quanto exposto, optou-se pela modelagem com algoritmos do DL do tipo DFNN, pois, apesar de serem os mais simples, apresentam bons resultados, conforme relatado pela literatura recente (75). Esta hipótese de pesquisa foi confirmada pelos resultados obtidos pelo projeto.

2.5 Construção do Modelo e Medição de Desempenho em DL

Em aprendizado de máquina, o modelo deve apresentar um bom desempenho para entradas novas e não vistas, não apenas para aquelas em que o algoritmo foi treinado (é o desafio central do ML). Tal habilidade é chamada de generalização. O que separa o aprendizado de máquina da otimização tradicional é que se quer que o erro de generalização, ou erro de teste, seja o mais baixo possível (17), (43). Para fazer isso, precisa-se de um conjunto de teste. Este projeto seguiu a melhor prática de segregar o conjunto de teste em dois conjuntos distintos: conjuntos de validação e de teste (25). O conjunto de validação é usado para ajustar os hiperparâmetros da rede (número de camadas, número de unidades por camadas, taxa de aprendizado, etc. (43)).

O problema do *overfitting* (sobreajuste do modelo), que ocorre quando o *gap* (lacuna) entre o erro de treinamento e o erro de validação/teste é muito grande, não tem sido devidamente abordado na literatura recente de estimatição de SOC de baterias de íons de lítio usando aprendizado profundo. Portanto, é preciso aplicar conceitos da teoria de aprendizagem estatística nesta área de investigação (43). O processo de mitigar o *overfitting* é denominado regularização, que pode ser definido como qualquer modificação que se faz em um algoritmo de aprendizagem com o objetivo de reduzir seu erro de validação/generalização, mas não seu erro de treinamento. Para se visualizar este fenômeno, pode-se traçar as curvas de aprendizado de treinamento e de validação, conforme ilustrado pela Fig. 8. Esta figura mostra que os erros de treinamento (*training loss*) e de validação (*validation loss*) se comportam de maneira diferente. O eixo horizontal representa o número de épocas de treinamento. O eixo vertical é a função de perda. Observe que o modelo começa a apresentar *overfitting* por volta da quinta época.

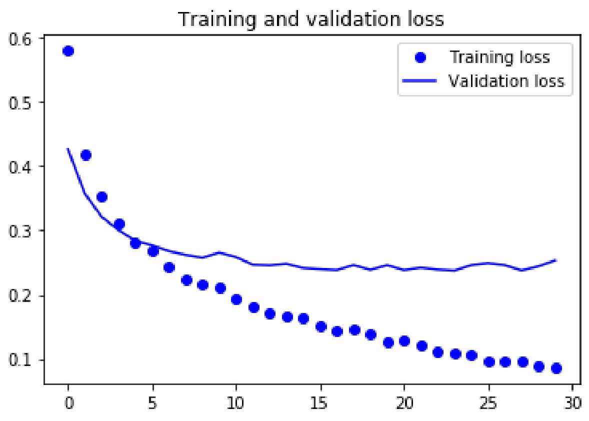


Figura 8 – erros de treinamento e de validação.

Observe-se que a literatura de aprendizado de máquina apresenta um arcabouço sólido para a solução de problemas de aprendizado profundo, tais como avaliação de modelos e ataque ao *overfitting*, entre outros (43), (25), (76). Os resultados satisfatórios obtidos por este projeto em termos de um erro de generalização baixo levam em consideração o *overfitting*, conforme será visto no próximo capítulo.

Como mencionado anteriormente, deve-se considerar o problema de otimização no contexto de DL, que difere completamente dos algoritmos de otimização tradicionais de várias maneiras. Neste trabalho, utilizou-se algoritmos com regras de aprendizagem adaptativas, tais como RMSProp e Adam, que incluem o conceito de *momentum*, permitindo uma convergência mais rápida do que o do SGD, ao custo de mais computação. A taxa de aprendizado é um dos hiperparâmetros mais difíceis de configurar em uma RNA, pois afeta significativamente o desempenho do modelo. Observe que pequenos valores da taxa de aprendizado resultam em uma convergência lenta do modelo de DL. Por outro lado, se a taxa de aprendizado for muito grande, a descida do gradiente pode ultrapassar o mínimo. Pode falhar em convergir, ou mesmo divergir

2.6 Aprendizado Estatístico

Esta seção apresenta uma breve revisão dos principais conceitos sobre processos estocásticos ou aleatórios (também chamados de processos de geração de dados pela literatura de aprendizado profundo (43)) e aprendizado estatístico que foram usados neste trabalho. O objetivo é apenas indicar o que é importante saber, sem entrar em detalhes. O leitor interessado deverá consultar a literatura apropriada (76), (77).

Definition 2.6.0.1 (Processo Estocástico)**.** *Seja T um conjunto arbitrário. Um processo estocástico é uma família $\{\mathbf{x}_t, t \in T\}$, tal que, para cada $t \in T$, $\mathbf{x}_t$ é uma variável aleatória.* ■

Quando o conjunto T é o conjunto dos números inteiros $\mathbb{Z}$, então $\{\boldsymbol{x}_t\}$ é um processo estocástico de tempo discreto (ou sequência aleatória); $\{\boldsymbol{x}_t\}$ é um processo estocástico de tempo contínuo se T representa o conjunto dos números reais $\mathbb{R}$.

A variável aleatória $\boldsymbol{x}_t$ é, de fato, uma função de dois argumentos $\boldsymbol{x}(t, \zeta)$, $t \in T$, $\zeta \in \Omega$, dado que seja definida sobre o espaço amostral Ω. Para cada $\zeta \in \Omega$ tem-se uma realização, trajetória ou série temporal x_t. O conjunto de todas as realizações é denominado *ensemble*. Cada trajetória é uma função ou uma sequência não aleatória e para um dado t, x_t é um número.

Um processo $\boldsymbol{x}_t$ é completamente especificado por suas distribuções finito-dimensionais ou funções de distribuição de probabilidade de ordem n:

$$F_{\boldsymbol{x}}(x_1, x_2, \ldots, x_n; t_1, t_2, \ldots, t_n) = P\{\boldsymbol{x}(t_1) \leq x_1, \boldsymbol{x}(t_2) \leq x_2, \ldots, \boldsymbol{x}(t_n) \leq x_n\} \qquad (2.11)$$

em que $t_1, t_2, \ldots, t_n$ são quaisquer elementos de T e $n \geq 1$.

A função de distribuição de probabilidade de primeira ordem também é conhecida como função de distribuição acumulada.

A função densidade de probabilidade é dada por:

$$f_{\boldsymbol{x}}(x_1, x_2, \ldots, x_n; t_1, t_2, \ldots, t_n) = \frac{\partial^n F_{\boldsymbol{x}}(x_1, x_2, \ldots, x_n; t_1, t_2, \ldots, t_n)}{\partial x_1 \partial x_2 \ldots \partial x_n}. \tag{2.12}$$

Aplicando a fórmula da densidade de probabilidade condicional,

$$f_{\boldsymbol{x}}(x_k | x_{k-1}, \ldots, x_1) = \frac{f_{\boldsymbol{x}}(x_1, \ldots, x_{k-1}, x_k)}{f_{\boldsymbol{x}}(x_1, \ldots, x_{k-1})}, \tag{2.13}$$

em que $f_{\boldsymbol{x}}(x_1, \ldots, x_{k-1}, x_k)$ denota $f_{\boldsymbol{x}}(x_1, \ldots, x_{k-1}, x_k; t_1, \ldots, t_{k-1}, t_k)$, de forma repetida sobre $f_{\boldsymbol{x}}(x_1, \ldots, x_{n-1}, x_n)$ obtém-se a regra da cadeia da probabilidade

$$f_{\boldsymbol{x}}(x_1, x_2, \ldots, x_n) = f_{\boldsymbol{x}}(x_1) f_{\boldsymbol{x}}(x_2 | x_1) f_{\boldsymbol{x}}(x_3 | x_2, x_1) \ldots f_{\boldsymbol{x}}(x_n | x_{n-1}, \ldots, x_1). \tag{2.14}$$

Quando $\boldsymbol{x}_t$ é uma sequência de variáveis aleatórias mutuamente independentes, (2.14) pode ser reescrita como

$$f_{\boldsymbol{x}}(x_1, x_2, \ldots, x_n) = f_{\boldsymbol{x}}(x_1) f_{\boldsymbol{x}}(x_2) \ldots f_{\boldsymbol{x}}(x_n). \tag{2.15}$$

Definition 2.6.0.2 (Processo Puramente Estocástico)**.** *Um processo puramente estocástico* $\{\mathbf{x}_t, t \in \mathbb{Z}\}$ *é uma sequência de variáveis aleatórias mutuamente independentes.* ■

Definition 2.6.0.3 (Processo IID)**.** *Um processo Independente e Identicamente Distribuído (IID)* $\{\mathbf{x}_t, t \in \mathbb{Z}\}$*, denotado por* $\mathbf{x}_t \sim$ *IID, é um processo puramente estocástico e identicamente distribuído.* ■

Como já foi dito anteriormente, o desafio central em ML é que o algoritmo tenha um bom desempenho sobre o conjunto de teste. Os conjuntos de treinamento e de teste são gerados pelo mesmo processo aleatório. Tipicamente, assume-se que os exemplos em cada conjunto são independentes e que os conjuntos de treinamento e de teste sejam identicamente distribuídos. Estas hipóteses são conhecidas coletivamente como premissas IID.

O teorema *no free lunch* (78) diz que, considerando a média de todos os processos de geração de dados possíveis, qualquer algoritmo de aprendizado de máquina tem a mesma taxa de erro quando avaliado em exemplos não observados anteriormente. Ou seja, não existe, pelo menos em teoria, um algoritmo de aprendizado de máquina que seja melhor do que todos os outros para todos os casos.

Contudo, o teorema é válido apenas quando se trabalha com a média de todos os os possíveis processos geradores de dados. Mas isso não ocorre no mundo real, pois o processo aleatório está sujeito a restrições físicas. Assim, em aplicações práticas, é realista projetar algoritmos que possuam um bom desempenho para um dado processo gerador de dados.

Indo além, o objetivo da pesquisa em algoritmos de ML não é procurar por um algoritmo de aprendizado universal. Em vez disso, tem-se que entender quais são as características estocásticas do *dataset* que se tem à disposição de forma a projetar, validar e testar um algoritmo que seja eficiente para aquele conjunto de dados.

3 Conclusões

Os resultados obtidos fornecem fortes evidências empíricas de que as redes neurais profundas alimentadas para frente ou DFNN conseguem estimar o SOC de uma célula de íons de lítio com grande acurácia.

Este trabalho confirmou que a estimação do SOC por meio de DFNN apresenta as seguintes vantagens em relação métodos que são orientados ao modelo (*model driven*):

1. as DFNN conseguem estimar com grande precisão a dependência funcional não linear que existe entre tensão, corrente e temperatura (quantidades observáveis) e quantidades não observáveis, como o SOC; e

2. o problema da identificação dos parâmetros (como para um ECM) é evitado.

Especificamente, os estudos empíricos/computacionais realizados mostram que:

- É possível estimar o SOC de uma célula de íons de lítio Panasonic 18650PF com erro menor que 0.12% (métrica MSE) por meio de algoritmos de DFNN (vide os artigos dos apêndices C e D). Este resultado corrobora a grande acurácia que pode ser obtida quando a estimação do SOC é realizada por arquiteturas de DL, conforme relatado pela literatura recente.

- A escolha do algoritmo de otimização (SGD, AdaGrad, RMSprop, Adam, etc. (43)) deve ser feita caso a caso.

- A escolha do algoritmo de otimização afeta o desempenho do modelo de DFNN usado.

Para trabalhos futuros, recomenda-se que i) a estimação do SOC de uma célula de íons de lítio em diferentes temperaturas e ii) outras arquiteturas de redes neurais *feed-forward* e recorrentes sejam investigados.

Referências

1 BOLZON, A. da S. *Implementação e Estudo de um Conversor Bidirecional CC-CA Monofásico de Dois Estágios como Interface entre uma Nanorrede e a Rede Elétrica.* Tese (Dissertação de Mestrado) — Universidade Federal de Minas Gerais, Escola de Engenharia, Minas Gerais, 2014. Citado na página 15.

2 RAJEEV, A. *Principles of Cyber-Physical Systems.* [S.l.]: The MIT Press, 2015. Citado na página 15.

3 CAO, Y.; JIANG, T.; HAN, Z. A Survey of Emerging M2M Systems: Context, Task, and Objective. *IEEE Internet of Things Journal*, p. 1246 – 1258, 2016. Citado na página 15.

4 SANDIA National Laboratories. *DOE/EPRI 2013 Electricity Storage Handbook in Collaboration with NRECA*. USA, 2013. Citado 5 vezes nas páginas 15, 16, 53, 54 e 98.

5 HANNAN, M. A. et al. Review of Energy Storage Systems for Electric Storage Vehicle Applications: Issues and Challenges. *Renewable and Sustainable Energy Reviews*, Vol. 69, p. 771 – 789, 2016. Citado 5 vezes nas páginas 16, 53, 77, 89 e 107.

6 WHITTINGHAM, M. S. History, Evolution, and Future Status of Energy Storage. *Proceedings of the IEEE*, Vol. 100, p. 1518 – 1534, 2012. Citado 9 vezes nas páginas 16, 54, 75, 76, 87, 88, 96, 105 e 106.

7 LUO, X. et al. Overview of Current Development in Electrical Energy Storage Technologies and the Application Potential in Power System Operation. *Applied Energy*, Vol. 137, p. 511 – 536, 2015. Citado 6 vezes nas páginas 16, 54, 76, 88, 96 e 106.

8 BYRNE, R. H. et al. Energy Management and Optimization Methods for Grid Energy Storage Systems. *IEEE ACCESS*, 2018. Citado 5 vezes nas páginas 16, 54, 76, 88 e 106.

9 MOTAPON, S. N. et al. A Generic Electrothermal Li-ion Battery Model for Rapid Evaluation of Cell Temperature Temporal Evolution. *IEEE Transactions on Industrial Electronics*, Vol. 64, n. 2, p. 998 – 1007, February 2017. Citado 10 vezes nas páginas 16, 17, 54, 55, 77, 78, 88, 98, 107 e 108.

10 HURIA, T. et al. High fidelity electrical model with thermal dependence for characterization and simulation of high power lithium battery cells. In: *IEEE International Electric Vehicle Conference*. [S.l.]: IEEE, 2012. Citado 8 vezes nas páginas 17, 54, 55, 76, 78, 98, 106 e 108.

11 REN, L. et al. Remaining useful life prediction for lithium-ion battery: A deep learning approach. *IEEE Access*, v. 6, p. 50587–50598, 2018. Citado 6 vezes nas páginas 17, 55, 77, 78, 89 e 107.

12 JEON, D. H. Numerical Modeling of Lithium Ion Battery for Predicting Thermal Behavior in a Cylindrical Cell. *Current Appl. Phys.*, Vol. 14, n. 2, p. 196 –205, Feb. 2014. Citado 5 vezes nas páginas 17, 55, 78, 98 e 108.

13 LI, J. et al. An Electrochemical-Thermal Model Based on Dynamics Responses for Lithium Iron Phosphate Battery. *J. Power Sources*, Vol. 255, p. 130 –143, June 2014. Citado 5 vezes nas páginas 17, 55, 78, 98 e 108.

14 CHEMALI, E. et al. State-of-charge estimation of li-ion batteries using deep neural networks: a machine learning approach. *Journal of Power Sources*, v. 400, p. 242–255, 2018. Citado 14 vezes nas páginas 17, 26, 36, 55, 68, 71, 77, 78, 81, 88, 90, 98, 108 e 111.

15 HANNAN, M. A. et al. Toward Enhanced State of Charge Estimation of Lithium-ion Batteries Using Optimized Machine Learning Techniques. *Scientific Reports, Nature*, v. 10, n. 4687, 2020. Citado 4 vezes nas páginas 17, 77, 98 e 108.

16 ALI, M. U. et al. Towards a smarter battery management system for electric vehicle applications: a critical review of lithium-ion battery state of charge estimation. *Energies*, v. 12, n. 446, 2019. Citado 3 vezes nas páginas 17, 98 e 108.

17 RUSSEL, S.; NORVIG, P. *Artificial Intelligence: A Modern Aproach*. 4th ed.. ed. [S.l.]: Pearson, 2020. Citado 6 vezes nas páginas 17, 23, 27, 28, 33 e 37.

18 LIPU, M. S. H. et al. Data-driven state of charge estimation of lithium-ion batteries: Algorithms, implementation factors, limitations and future trends. *Journal of Cleaner Production*, v. 277, 2020. Citado 9 vezes nas páginas 18, 26, 35, 96, 97, 98, 99, 106 e 107.

19 LI, Z. et al. On state-of-charge determination for lithium-ion batteries. *Journal of Power Sources*, v. 348, p. 281–301, 2017. Citado 4 vezes nas páginas 18, 88, 96 e 106.

20 KOLLMEYER, P. Panasonic 18650PF Li-ion Battery Data. *Mendeley Data*, v. 1, 2018. Disponível em: <https://data.mendeley.com/datasets/wykht8y7tg/1>. Acesso em: 09/02/ 2020. Citado 15 vezes nas páginas 19, 68, 71, 74, 77, 80, 84, 89, 90, 93, 99, 103, 104, 110 e 113.

21 LIMA, A. B. de; SALLES, M. B. C.; CARDOSO, J. R. *State-of-Charge Estimation of a Li-Ion Battery using Deep Forward Neural Networks*. 2020. Disponível em: <https://arxiv.org/abs/2009.09543>. Citado na página 19.

22 LIMA, A. B. de; SALLES, M. B. C.; CARDOSO, J. R. *State-of-charge Estimation of a Li-ion Battery using Deep Learning and Stochastic Optimization*. 2020. Disponível em: <https://arxiv.org/abs/2011.09673>. Citado na página 19.

23 LIMA, A. B. de; SALES, M. B. C.; CARDOSO, J. R. State-of-Charge Estimation of the Panasonic 18650PF Li-ion Cell using Deep Learning Models and Algorithms with Adaptive Learning Rates. *International Journal of Engineering Research and Applications*, v. 10, p. 30–34, december 2020. Citado na página 19.

24 LIMA, A. B. de; SALES, M. B. C.; CARDOSO, J. R. Data-driven state-of-charge estimation of the Panasonic 18650PF Li-ion cell using deep forward neural networks. In: *INDUSCON 2021*. [S.l.: s.n.], 2021. Citado na página 19.

25 CHOLLET, F. *Deep Learning with Python*. 1st. ed. [S.l.]: Manning Publications, 2018. Citado 12 vezes nas páginas 19, 20, 25, 33, 37, 56, 63, 66, 67, 68, 71 e 97.

26 GULLI, A.; KAPOOR, A.; PAL, S. *Deep Learning with TensorFlow 2 and Keras*. 2nd. ed. [S.l.]: Packt>, 2019. Citado 3 vezes nas páginas 19, 20 e 71.

27 SKANSI, S. *Introduction to Deep Learning: From Logical Calculus to Artificial Intelligence*. 1st. ed. [S.l.]: Springer, 2018. Citado 2 vezes nas páginas 19 e 71.

28 CHOLLET, F.; ALLAIRE, J. J. *Deep Learning with R*. 1st. ed. [S.l.]: Manning Publications, 2017. Citado na página 19.

29 MATHWORKS. *MATLAB for Deep Learning*. 2020. Disponível em: <https://www.mathworks.com/solutions/deep-learning.html>. Acesso em: 08/07/ 2020. Citado na página 19.

30 GOOGLE. *TensorFlow*. 2020. Disponível em: <https://www.tensorflow.org/?hl=pt-br>. Acesso em: 07/07/ 2020. Citado 3 vezes nas páginas 20, 71 e 81.

31 PASZKE, A. et al. *PyTorch*. 2020. Disponível em: <https://pytorch.org>. Acesso em: 07/07/ 2020. Citado 2 vezes nas páginas 20 e 71.

32 APACHE. *MXNet*. 2020. Disponível em: <https://mxnet.apache.org>. Acesso em: 07/07/ 2020. Citado 2 vezes nas páginas 20 e 71.

33 MICROSOFT. *Microsoft Cognitive Toolkit*. 2020. Disponível em: <https://docs.microsoft.com/en-us/cognitive-toolkit/>. Acesso em: 07/07/ 2020. Citado 2 vezes nas páginas 20 e 71.

34 FACEBOOK. *Caffe2*. 2020. Disponível em: <https://caffe2.ai>. Acesso em: 07/07/ 2020. Citado 2 vezes nas páginas 20 e 71.

35 GOOGLE. *Colaboratory*. 2020. Disponível em: <https://colab.research.google.com/notebooks/intro.ipynb>. Acesso em: 07/07/ 2020. Citado 2 vezes nas páginas 20 e 71.

36 GitHub. *GitHub Code Repository*. 2020. Disponível em: <https://github.com>. Acesso em: 01/06/ 2020. Citado na página 20.

37 Project Jupyter. *The Jupyter Notebook*. 2020. Disponível em: <https://jupyter.org/>. Acesso em: 07/07/ 2020. Citado na página 20.

38 PÉREZ, F.; GRANGER, B. E. IPython: a System for Interactive Scientific Computing. *Computing in Science and Engineering*, IEEE Computer Society, v. 9, n. 3, p. 21–29, may 2007. ISSN 1521-9615. Disponível em: <https://ipython.org>. Citado na página 20.

39 GOODFELLOW, I. J. et al. *Generative Adversarial Networks*. 2014. Disponível em: <https://arxiv.org/abs/1406.2661>. Citado na página 20.

40 GRAVES, A.; WAYNE, G.; DANIHELKA, I. Neural Turing Machines. *CoRR*, abs/1410.5401, 2014. Disponível em: <https://arxiv.org/abs/1410.5401>. Citado na página 20.

41 Anaconda. *Anaconda Open-Source Distribution*. 2020. Disponível em: <https://www.anaconda.com/products/individual>. Acesso em: 07/07/ 2020. Citado na página 21.

42 FUEGI, J.; FRANCIS, J. Lovelace babbage and the creation of the 1843 notes. *IEEE Annals of the History of Computing*, v. 25, n. 4, p. 16–26, 2003. Citado na página 23.

43 GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. 1st. ed. [S.l.]: MIT Press, 2016. Citado 23 vezes nas páginas 23, 27, 29, 31, 32, 37, 38, 41, 55, 56, 61, 62, 64, 77, 80, 81, 84, 88, 89, 92, 97, 107 e 111.

44 LIMA, A. B. de; BARRETO, M. R. P.; AMAZONAS, J. R. de A. Spectra-based sentiment analysis. *INFOCOMP*, v. 17, n. 2, p. 01–06, december 2018. Citado na página 23.

45 NG, A. *Machine Learnig – Coursera*. 2019. Citado na página 23.

46 LANTZ, B. *Machine Learning with R*. 1st. ed. [S.l.]: Packt Publishing, 2013. Citado na página 24.

47 CHEN, B.; MEDINI, T.; SHRIVASTAVA, A. SLIDE: In Defense of Smart Algorithms over Hardware Acceleration for Large-Scale Deep Learning Systems. *arXiv*, march 2019. Citado 2 vezes nas páginas 25 e 55.

48 SILVER, D. et al. General Reinforcement Learning Algorithm. *arXiv*, 2017. Disponível em: <https://arxiv.org/pdf/1712.01815.pdf?utm_campaign=nathan.ai%20newsletter&utm_medium=email&utm_source=Revue%20newsletter>. Citado 2 vezes nas páginas 25 e 63.

49 HAYKIN, S. *Neural Networks and Learning Machines*. third ed. [S.l.]: Pearson, 2009. Citado na página 27.

50 ANDERSON, R. et al. Chapter I – Introduction. In: BARR, A.; FEIGENBAUM, E. A. (Ed.). *The Handbook of Artificial Intelligence*. [S.l.]: Elsevier, 1981. v. 1. Citado 2 vezes nas páginas 27 e 58.

51 BODEN, M. A. *AI Its Nature and Future*. 1st. ed. [S.l.]: Oxford University Press, 2016. Citado 3 vezes nas páginas 28, 58 e 62.

52 MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, v. 5, p. 115–133, 1943. Disponível em: <http://www.cs.cmu.edu/~epxing/Class/10715/reading/McCulloch.and.Pitts.pdf>. Citado 2 vezes nas páginas 28 e 58.

53 TURING, A. M. On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, v. 42, p. 230–265, 1936. Citado 2 vezes nas páginas 28 e 58.

54 NEWELL, A. Physical symbom systems. *Cognitive Science*, v. 4, p. 135–183, April–June 1980. Citado 2 vezes nas páginas 28 e 62.

55 BRYSON, J. A. E.; HO, Y. *Applied Optimal Control*. [S.l.]: Blaisdell, 1969. Citado na página 29.

56 WERBOS, P. J. *Beyond Regression: New tools for prediction and analysis in the behavioral sciences*. Doctoral Thesis — Harvard, 1974. Citado 2 vezes nas páginas 29 e 64.

57 PARKER, D. B. Technical Report, *Learning Logic*. 1985. Citado 2 vezes nas páginas 29 e 64.

58 LECUN, Y. A Learning Scheme for Asymmetric Threshold Network. In: BIENENSTOCK, E.; FOGELMAN, F.; WEISBUCH, G. (Ed.). *Disordered systems and biological organization*. [S.l.]: Springer Verlag, 1986. Citado 2 vezes nas páginas 29 e 64.

59 CYBENKO, G. Approximation by superpositions of a sigmoidal function. *Math. Control Signal Systems*, v. 2, p. 303–314, 1989. Disponível em: <https://doi.org/10.1007/BF02551274>. Citado 4 vezes nas páginas 30, 31, 59 e 61.

60 SAYED, A. H. *Fundamentals of Adaptive Filtering*. [S.l.]: Wiley-Interscience, 2003. Citado na página 30.

61 OPPENHEIM, A. V.; SCHAFER, R. W. *Processamento de Sinais em Tempo Discreto*. 3a. ed. [S.l.]: Pearson, 2019. Citado na página 31.

62 LATHI, B. P. *Signal Processng and Linear Systems*. [S.l.]: Oxford University Press, 1998. Citado 2 vezes nas páginas 31 e 61.

63 HINTON, G. E.; OSINDERO, S.; TEH, Y. A fast learning algorithm for deep belief nets. *Neural Computation*, v. 18, p. 1527–1554, 2006. Citado 2 vezes nas páginas 32 e 66.

64 CHOLLET, F.; ALLAIRE, J. J. *Deep Learning with R*. 1st ed.. ed. [S.l.]: Manning Publications, 2017. Citado 4 vezes nas páginas 32, 33, 68 e 71.

65 LECUN, Y. Generalization and network design strategies. In: PFEIFER, R. et al. (Ed.). *Connectionism in Perspective*. Zurich, Switzerland: Elsevier, 1989. Citado na página 33.

66 LECUN, Y. et al. Handwritten digit recognition with a back-propagation network. In: TOURETZKY, D. S. (Ed.). *Advances in Neural Information Processing Systems 2*. Morgan-Kaufmann, 1990. p. 396–404. Disponível em: <http://papers.nips.cc/paper/293-handwritten-digit-recognition-with-a-back-propagation-network.pdf>. Citado na página 33.

67 LECUN, Y. et al. Gradient-based learning applied to document recognition. *Proceeedings of the IEEE*, v. 86, n. 11, November 1998. Citado na página 33.

68 SILVER, D.; AL. et. Mastering chess and shogi by sef-play with a general reinforcement learning algorithm. *Science*, v. 362, p. 1140–1144, dec. 2018. Citado 3 vezes nas páginas 33, 55 e 67.

69 SIMON, H. A. Artificial intelligence: an empirical science. *Artificial Intelligence - Elsevier*, v. 77, p. 95–127, 1996. Citado na página 35.

70 SEPASI, S.; GHORBANI, R.; LIAW, B. Y. Improved extended Kalman filter for state of charge estimation of battery pack. *Journal of Power Sources*, v. 255, p. 368–376, 2014. Citado 5 vezes nas páginas 36, 55, 78, 99 e 108.

71 CHARKHGARD, M.; FARROKHI, M. State-of-Charge Estimation for Lithium-Ion Batteries Using Neural Networks and EKF. *IEEE Transactions on Industrial Electronics*, v. 57, dec. 2010. Citado 5 vezes nas páginas 36, 55, 78, 99 e 108.

72 DU, J.; LIU, Z.; WANG, Y. State of charge estimation for li-ion battery based on model from extreme learning machine. *Control Engineering Practice*, v. 26, p. 11–19, 2014. Citado 5 vezes nas páginas 36, 55, 78, 99 e 108.

73 LEE, Y.-S.; WANG, W.-Y.; KUO, T.-Y. Soft Computing for Battery State-of-Charge (BSOC) Estimation in Battery String Systems. *IEEE Transactions on Industrial Electronics*, v. 55, n. 1, jan. 2014. Citado 5 vezes nas páginas 36, 55, 78, 99 e 108.

74 CHANG, W.-Y. Estimation of the state of charge for a LFP battery using a hybrid method that combines a RFF neural network, an OLS algorithm and AGA. *Electrical Power and Energy Systems*, v. 53, p. 603– 611, 2013. Citado 5 vezes nas páginas 36, 55, 79, 99 e 108.

75 CHEMALI, E. *Intelligent State-of-Charge and State-of-Health Estimation Framework for Li-ion Batteries in Electrified Vehicles Using Deep Learning Techniques.* Doctoral Thesis — McMaster university, 2018. Citado na página 36.

76 MURPHY, K. P. *Machine Learning: A Probabilistic Approach.* [S.l.]: The MIT Press, 2012. Citado 5 vezes nas páginas 37, 38, 56, 57 e 64.

77 PAPOULIS, A. *Probability, Random Variables, and Stochastic Processes.* Third. [S.l.]: McGraw-Hill, 1996. Citado 2 vezes nas páginas 38 e 64.

78 WOLPERT, D. H. The Lack of A Priori Disctintions Between Learning Algorithms. *Neural Computation*, v. 8, p. 1341–1390, 1996. Citado 2 vezes nas páginas 39 e 66.

79 ZHAO, R. et al. A compact unified methodology via a recurrent neural network for accurate modeling of lithium-ion battery voltage and state-of-charge. In: *2017 IEEE Energy Conversion Congress and Exposition (ECCE).* [S.l.: s.n.], 2017. p. 5234–5241. Citado 3 vezes nas páginas 55, 68 e 77.

80 LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *Nature*, v. 521, p. 436–444, 2015. Citado na página 55.

81 MALLAT, S. Understanding deep convolutional networks. *Phil. Trans. R. Soc. A*, 2016. Disponível em: <http://rsta.royalsocietypublishing.org>. Citado na página 55.

82 MINAEE, S.; ABDOLRASHIDI, A. A. Deep-emotion: Facial expression recognition using attentional convolutional network. *arXiv*, february 2019. Disponível em: <http://arxiv.org/abs/1902.01019>. Citado na página 55.

83 CHEMALI, E. et al. Long short-term memory networks for accurate state-of-charge estimation of li-ion batteries. *IEEE Transactions on Industrial Electronics*, v. 65, n. 8, p. 6730–6739, 2018. Citado 3 vezes nas páginas 55, 68 e 77.

84 KOLLMEYER, P.; HACKL, A.; EMADI, A. Li-ion battery model performance for automotive drive cycles with current pulse and eis parameterization. In: *2017 IEEE Transportation Electrification Conference and Expo (ITEC).* [S.l.: s.n.], 2017. p. 486–492. Citado 3 vezes nas páginas 55, 68 e 77.

85 RUSSEL, S.; NORVIG, P. *Inteligência Artificial.* 3a ed.. ed. [S.l.]: Rio de Janeiro: Elsevier, 2013. Citado na página 58.

86 OPPENHEIM, A. V.; SCHAFER, R. W. *Discrete-Time Signal Processing.* 3rd. ed. [S.l.]: Pearson, 2009. Citado na página 60.

87 ACM. *A. M. Turing Award 2018.* 2020. Https://awards.acm.org/about/2018-turing. Acesso em: 07/07/ 2020. Citado na página 62.

88 CUN, L. et al. Handwritten Digit Recognition with a Back-Propagation Network. In: *Advances in Neural Information Processing Systems*. [S.l.]: Morgan Kaufmann, 1990. p. 396–404. Citado na página 62.

89 MCCARTHY, J.; FEIGENBAUM, E. A. In Memoriam: Arthur Samuel: Pioneer in Machine Learning. *AI Magazine*, v. 11, n. 3, p. 10, Sep. 1990. Disponível em: <https://www.aaai.org/ojs/index.php/aimagazine/article/view/840>. Citado na página 62.

90 SAMUEL, A. L. Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development*, v. 3, n. 3, p. 210–229, 1959. Citado na página 63.

91 ERTEL, W. *Introduction to Artificial Intelligence*. 2nd. ed. [S.l.]: Springer, 2017. Citado na página 63.

92 MITCHELL, T. M. *Machine Learning*. [S.l.]: McGraw-Hill Dscience/Eng./Math, 1997. Citado na página 63.

93 WIDROW, B.; HOFF, M. E. Adaptive switching circuits. *IRE WESCON Convention Record*, p. 96–104, 1960. Citado na página 64.

94 ROSENBLATT, F. The Perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, v. 65, p. 386–408, 1958. Citado na página 64.

95 RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning Internal Representations by Error Propagation. In: RUMELHART, D. E.; MCCLELLAND, J. L. (Ed.). *Parallel Distributed Processing: Explorations in The Microstructure of Cognition*. [S.l.]: Foundations, Cambridge, MA: Bradford Books/MIT Press, 1986. v. 1. Citado 2 vezes nas páginas 64 e 100.

96 JR., A. E. B.; HO, Y. C. *Applied Optimal Control*. [S.l.]: Blaisdell, 1969. Citado na página 64.

97 Department for Transportation. *HM Government: The Road to Zero: Next steps towards cleaner road transport and delivering our Industrial Strategy*. 2018. Citado 2 vezes nas páginas 75 e 87.

98 HUANG, Z. et al. Convolutional Gated Recurrent Unit - Recurrent Neural Network for State-of-Charge Estimation of Lithium-Ion Batteries. *IEEEAccess*, v. 7, p. 93139–93149, 2019. Citado na página 77.

99 ZAHID, T. et al. State of charge estimation for electric vehicle power battery using advanced machine learning algorithm under diversified drive cycles. *Energy*, v. 162, p. 871–882, 2018. Citado na página 77.

100 TIELEMAN, T.; HINTON, G. *Lecture 6.5 - RMSProp, COURSERA: Neural Networks for Machine Learning*. [S.l.], 2012. Citado 2 vezes nas páginas 77 e 107.

101 KINGMA, D. P.; BA, J. L. Adam: A Method fot Stochastic Optimization. In: *ICLR conference,*. [S.l.: s.n.], 2015. Citado 2 vezes nas páginas 77 e 107.

102 DUCHI, J.; HAZAN, E.; YORAM, S. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, v. 12, n. 2121–2159, 2011. Citado na página 77.

103 CHOLLET, F. *Keras: the Python deep learning API*. 2021. Disponível em: <https://keras.io/>. Citado 3 vezes nas páginas 81, 82 e 102.

104 HOW, D. N. T. et al. State-of-Charge estimation of Li-Ion battery in electric vehicles: a deep neural network approach. *IEEE Transactions on Industry Applications*, v. 56, n. 5, september/october 2020. Citado 3 vezes nas páginas 88, 96 e 106.

105 MARTINS, F. et al. Analysis of Fossil Fuel Energy Consumption and Environmental Impacts in European Countries. *Energies*, v. 12, n. 6, 2019. Citado 2 vezes nas páginas 95 e 96.

106 JOHNSON, B. *Now is the time to plan our green recovery*. 2020. Citado 2 vezes nas páginas 96 e 105.

107 SABOUR, S.; FROSST, N.; HINTON, G. E. Dynamic routing between capsules. In: *Proc. 31st Int. Conf. Neural Inf. Process. Syst.* [S.l.]: NIPS, 2017. p. 3859–3869. Citado 2 vezes nas páginas 97 e 107.

108 RICHARDSON, F.; REYNOLDS, D.; DEHAK, N. Deep neural networks approaches to speaker and language recognition. *IEEE Signal Process. Lett.*, v. 22, n. 10, p. 1671–1675, october 2015. Citado 2 vezes nas páginas 97 e 107.

109 VASWANI, A. et al. Attention is all you need. In: *Proc. 31st Int. Conf. Neural Inf. Process. Syst.* [S.l.]: NIPS, 2017. p. 5998–6008. Citado 2 vezes nas páginas 97 e 107.

110 AGGARWAL, C. C. *Neural Networks and Deep Learning: A Textbook*. [S.l.]: Cham, Switzerland: Springer, 2018. Citado na página 97.

111 United States Environmental Protection Agency – EPA. *Dynamometer Drive Schedules*. [S.l.], 2021. Disponível em: <https://www.epa.gov/vehicle-and-fuel-emissions-testing/dynamometer-drive-schedules>. Citado na página 99.

112 YUAN, B.; PARHI, K. K. Early stopping criteria for energy-efficient low-latency belief-propagation polar code decoders. *IEEE Trans. Signal. Proc.*, v. 62, n. 24, december 2014. Citado na página 100.

Apêndices

APÊNDICE A – State-of-Charge Estimation of a Li-Ion Battery using Deep Forward Neural Networks

Authors: Alexandre B. de Lima, Maurício B. C. Salles, and José Roberto Cardoso.
Electronic preprint: <https://arxiv.org/abs/2009.09543>, september, 2020.

A.1 Abstract

This article presents two Deep Forward Networks with two and four hidden layers, respectively, that model the drive cycle of a Panasonic 18650PF lithium-ion (Li-ion) battery at a given temperature using the K-fold cross-validation method, in order to estimate the State of Charge (SOC) of the cell. The drive cycle power profile is calculated for an electric truck with a 35kWh battery pack scaled for a single 18650PF cell. We propose a machine learning workflow which is able to fight overfitting when developing deep learning models for SOC estimation. The contribution of this work is to present a methodology of building a Deep Forward Network for a lithium-ion battery and its performance assessment, which follows the best practices in machine learning.

Keywords: Artificial Intelligence, Deep Learning, Electrical Energy Storage, Li-ion battery, and State-Of-Charge.

A.2 Introduction

A.2.1 State of the Art and Trends in Li-ion Battery Estimation

Energy storage acts as a mediator between variable loads and variable sources. Electricity storage is not new. Volta invented the modern battery in 1799. Batteries were implemented in telegraph networks in 1836 (4). The Rocky River Hydroelectric Power Plant in New Milford, Connecticut, was the first major electrical energy storage (EES) system project in the United States. The plant used hydroelectric storage technology through Pumped Hydroelectric Storage (PHS) pumping.

This research is motivated by the study of the application of EES systems in the area of sustainable energy sources.

Hannan et al. (5) present a detailed taxonomy of the types of energy storage systems taking into account the form of energy storage and construction materials: mechanical, electrochemical (rechargeable and flow batteries), chemical, electrical (ultracapacitor or superconducting magnetic coil), thermal and hybrid.

Recently, industry and academia have given great importance to the electrification of the transport system, given the need to reduce the emission of greenhouse gases. Hybrid electric vehicles, such as the Toyota Prius, or fully electric vehicles, such as the various Tesla models, the Nissan Leaf and the GM Volt, are successful cases in the United States (6).

The advancement of EES technologies enabled the emergence of the iPod, smartphones and tablets with lithium-ion (li-ion) batteries. If renewable sources, such as solar and wind, become prevalent, the EES will be one of the critical components of the electricity grid, given the intermittent nature of these energy sources (6, 7). EES systems are necessary even when renewable sources are connected to the grid, because it is necessary to smooth the energy supply. For example, the EES of a building or factory can be charged during hours of reduced demand and supply/supplement energy demand during peak hours.

EES technology consists of the process of converting a form of energy (almost always electrical) to a form of storable energy, which can be converted into electrical energy when necessary. EES has the following functions: to assist in meeting the maximum electrical load demands, to provide time-varying energy management, to relieve the intermittency of renewable energy generation, to improve energy quality/reliability, to serve remote loads and vehicles, to support the realization of smart grids, improve the management of distributed/standby power generation and reduce the import of electricity during peak demand periods (7), (8).

An EES (which can connect to the network or operate in stand-alone mode) consists of two main subsystems: i) storage and ii) power electronics. Such subsystems are complemented by other components that include monitoring and control systems (4).

Lithium-ion battery technology has attracted the attention of industry and academia for the past decade. This is mainly due to the fact that lithium-ion batteries offer more energy, higher power density, higher efficiency and lower self-discharge rate than other battery technologies such as NiCd, NiMH, etc. (9).

The efficient use of the lithium-ion battery requires the supervision of a Battery Management System (BMS), as it is necessary that the battery operates under appropriate conditions of temperature and charge (State-Of-Charge (SOC)) (10). The cell temperature produces deleterious effects on the open circuit voltage, internal resistance and available capacity and can also lead to a rapid degradation of the battery if it operates above a given temperature threshold. Therefore, the modeling of the battery is of paramount importance, since it will be used by the BMS to manage the operation of the battery (9).

There are two methods of battery modeling: i) model-driven and ii) data-driven (based on data that is collected from the device) (11).

Electrothermal models, which belong to the category of model-driven methods, are commonly classified as: i) electrochemical or ii) based on Equivalent Circuit Models (ECM) (9, 10).

Electrochemical models are based on partial differential equations (12) and are able to represent thermal effects more accurately than ECM (13). However, the first class of models requires detailed knowledge of proprietary parameters of the battery manufacturer: cell area, electrode porosity, material density, electrolyte characteristics, thermal conductivity, etc. This difficulty can be eliminated by characterizing the battery using a thermal camera and thermocouples. But this solution is expensive, time consuming and introduces other challenges such as the implementation of dry air purge systems, ventilation, security, air and water supply, etc. Electrochemical models demand the use of intensive computing systems (10).

On the other hand, the ECM-based approach has been used for computational/numerical analysis of batteries (10). In this case, the objective is to develop an electrical model that represents the electrochemical phenomenon existing in the cell. The level of complexity of the model is the result of a compromise between precision and computational effort. Note that an extremely complex and accurate ECM may be unsuitable for application in embedded systems.

The most recent literature show that the machine learning approach, based on deep learning algorithms is the state of the art in the area (11, 79, 14, 47, 43, 80, 81, 82, 68, 83, 84). Machine learning is a branch of AI, as will be seen in section A.3.

Chemali et al (14) compared the performance of Deep Neural Networks (DNN) with those of other relevant algorithms that have been proposed since the second half of the 2000s. The article shows that the SOC estimation error obtained with deeep learning is less than the following methods:

- Model Adaptive-Improved Extended Kalman Filter (EKF) (70);
- Adaptive EKF with Neural Networks (71);
- Adaptive Unscented Kalman Filter (AUKF) with Extreme Machine Learning (72);
- Fuzzy NN with Genetic Algorithm (73); and
- Radial Bias Function Neural Network (74).

Estimating the SOC of lithium ion cells in a BMS by means of deep learning offers at least two significant advantages over model driven approaches, namely: i) neural networks are able to estimate the non linear functional dependence that exists between voltage, current and temperature (observable quantities) and unobservable quantities, such as SOC, with great precision and ii) the problem of identifying ECM parameters is avoided.

A.2.2 Contribution of the paper

In relation to the literature already mentioned, this paper takes a step back as far as reasons related to the methodology adopted for the construction and performance measurement of a deep learning model are concerned.

First, we work with Deep Feedforward Networks (DFN) as a baseline family model, as they form the basis of many important commercial applications (43). Our preliminary results show that a simple architecture with four hidden layers is already quite interesting. We do not work with Recurrent Neural Networks (RNN) because they are suitable for sequential data processing problems such as machine translation.

Second, the central challenge in machine learning is that our model has to perform well on new, unseen inputs, not just those on which our algorithm was trained. This ability is called generalization. What separates machine learning from traditional optimization is that we want the generalization error, or test error, to be as low as possible (43). To do this, we need a test set. In this work we followed the best practice of breaking the test set into two separate sets: validation and test sets (25). That is, the generalization power of the models were measured against validation and test sets. The validation set is used to fine tune the network hyperparameters.

Third, as a corollary of generalizatin, the central problem in machine learning, namely overfitting, has not been properly addressed, to the best of our knowledge, in the recent mainstream literature of SOC estimation of Li-ion batteries using deep learning. Thus, we have to apply concepts from statistical learning theory (43). Overfitting occurs when the gap between the training error and generalization error is too large. The processing of mitigating overfitting is called regularization, which can be defined as any modification we make to a learning algorithm with the goal of reducing its generalization error but not its training error. To see this phenomenon, one has to plot the training and generalization learning curves, see Fig. 9 for instance.

Note that the literature of machine learning presents a solid framework for solving deep learning problems, such as model evaluation and the attack on overfitting, among others (43, 25, 76). The satisfactory result obtained in this article in terms of a low generalization error takes overfitting into account.

Fourth, as mentioned before, we have to consider the optimization problem in the context of deep learning, which completely differs from traditional optimization algorithms

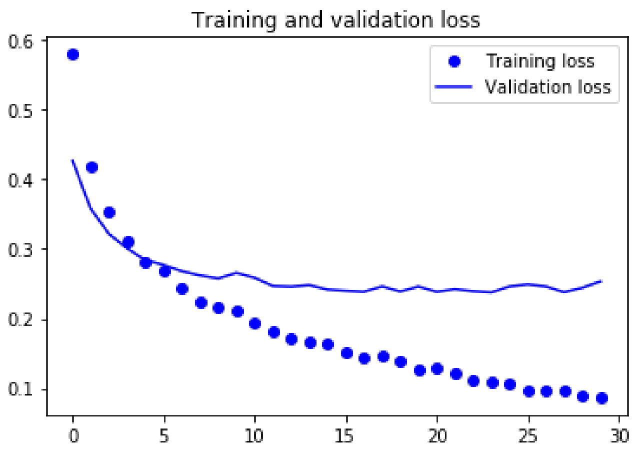


Figura 9 – Training and generalization errors behave differently. The horizontal axis represents the number of epochs. The vertical axis is the loss function. Note that the model starts to overfit around the fifth epoch.

in several ways. In this work, we use algorithms with adaptive learning rules, such as RMSProp and Adam, which include the concept of momentum, allowing faster convergence than Stochastic Gradient Descent (SGD), at the cost of more computation. The learning rate is one of the most difficult hyperparameters of a artificial neural network (ANN) to be configured as it significantly affects the model performance. Note that small values of the learning rate result in a slow convergence of deep learning. On the other hand, if the learning rate is too large, gradient descent can overshoot the minimum. It may fail to converge, or even diverge.

Fifth, the validation error is estimated by taking the average validation error across K trials. We use a simple, but popular solution, called K-fold cross-validaton (Fig. 10), which consists of splitting the available training data into two partitions (training and validation), instantiating K identical models, for each fold $k \in \{1, 2, \ldots, K\}$, and training each one on the training partitions, while evaluating on the validation partition. The validation score for the model used is then the average of the K validation scores obtained. This procedure allows network hyperparameters to be adjusted so that overfitting is mitigated (76, p. 23). It is usual to use about 80% of the data for the training set, and 20% for the validation set. Note that the validation scores may have a high variance with regard to the validation split. Therefore, K-fold cross-validaton help us improve the reliability when evaluating the generalization power of the model.

A.2.3 Organization of the paper

The remainder of the paper is organized as follows. Section A.3 presents an overview of AI, machine learning and deep learning for the reader who is not familiar with the subject. Section A.4 presents our experimental results. Finally, section A.5 presents our

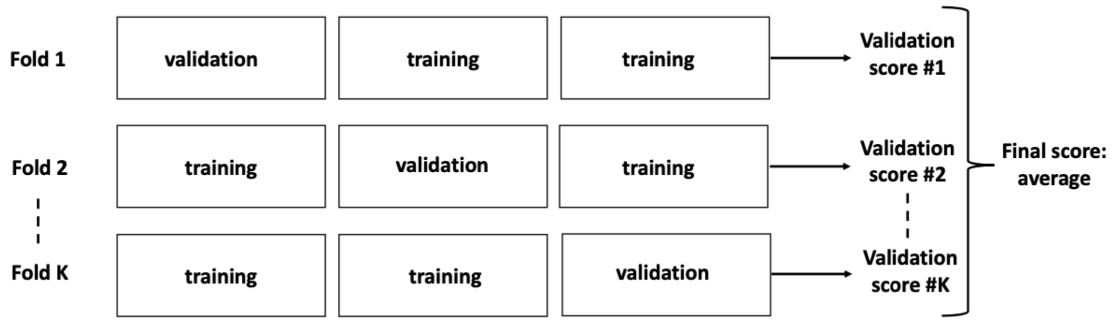


Figura 10 – K-fold cross-validation.

conclusions.

A.3 Artificial Intelligence and Deep Learning

A.3.1 The Notion of Neural Nets

The Handbook of Artificial Intelligence presents the following operational definition[1] for Artificial Intelligence (AI) (50):

> "Artificial Intelligence (AI) is the part of computer science concerned with designing intelligent computer systems, that is, systems that exhibit the characteristics we associate with intelligence in human behavior - understanding language, learning, reasoning, solving problems, and so on".

There are two main lines of research in AI: the connectionist and the symbolic. According to Boden (51), both were inspired by the seminal article entitled *A Logical Calculus of the Ideas Immanent in Nervous Activity* (1943), by Warren Sturgis McCulloch and Walter Pitts (52), the first modern computational theory[2]. The literature recognizes that the research carried out by McCulloch and Pitts is the pioneering work in AI (85).

Fig. 11 illustrates the McCulloch and Pitts artificial neuron model.

The artificial neuron (or unit) of Fig. 11 has the following characteristics:

- input signals (activations) $(a_1, a_2, \ldots, a_n)$ are 0 or 1 bits. The external signal $a_0 = 1$ is known as the bias[3];

[1] There is no consensus on the concept of intelligence.
[2] The theory is considered modern because it employs the mathematical notion of computation established by Turing in 1936 (53).
[3] The bias plays the role of the intercept b in the simple linear regression model $y = wx + b$.

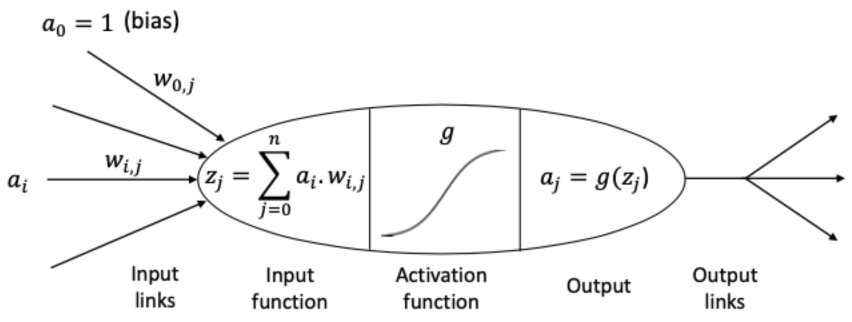


Figura 11 – McCulloch and Pitts model

- synaptic weights of the j-th neuron: $w_0, w_1, \ldots, w_n$; and
- a_j denotes the output signal (or output activation) of the jth neuron, given by

$$z_j = \sum_{i=0}^{n} w_{i,j} a_i \tag{A.1}$$

$$a_j = g(z_j) = \{0, 1\} \tag{A.2}$$

where $z = f(a)$ is the input function and $g(z)$ is a nonlinear activation function. The output is binary (bit 0 or bit 1); therefore, the McCulloch and Pitts model is said to have the " all or nothing " property.

The activation function of the McCulloch and Pitts model is the Heaviside function (unit step function)

$$g(z) = \begin{cases} 1, & z \geq 0 \\ 0, & z < 0 \end{cases} \tag{A.3}$$

$g(z) = 1$ se $z \geq 0$ ou $g(z) = 0$ para $z < 0$.

We take the opportunity to make a necessary digression on the neuron model of Fig. **??**, in order to present the intuition behind the fact that modern ANN (see Fig. 12) are able to approximate nonlinear functions with arbitrary precision, at least in theory (universal approximation theorem (59)). The explanation below considers a network with only one layer of $N + 1 = M$ neurons in parallel, where each neuron is excited by the input signal $\boldsymbol{a} = \{a_0, a_1, \ldots, a_N\}$, not necessarily binary. This layer is known as the hidden layer.

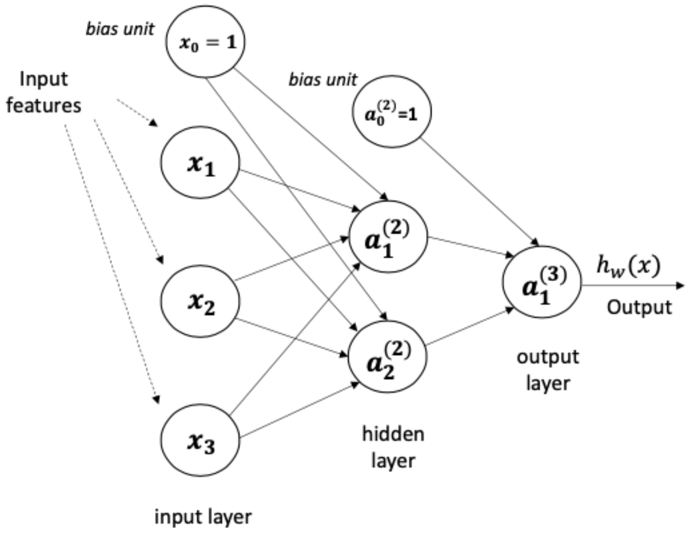


Figura 12 – Example of a densely connected neural net with one hidden layer. The hidden layer has three units ($a_0^{(2)}$, $a_1^{(2)}$, and $a_2^{(2)}$). The input features are the input signals $\boldsymbol{x}$. The function $\boldsymbol{y} = \boldsymbol{h_w}(\boldsymbol{x})$ is called the model or hypothesis.

Let's rewrite (A.1) in a vectorized form

$$z_i = \boldsymbol{W}^T \boldsymbol{a} \tag{A.4}$$

where we adopt the notation $\boldsymbol{W}$ for the vector of synaptic weights, $\boldsymbol{a}$ for the vector of entries and T denotes the transposition of matrices [4].

Consider the Discrete Fourier Transform (DFT) [5] of an input signal $\boldsymbol{b} = \{b_0, b_1, \ldots, b_N\}$ given by

$$B[k] = \begin{cases} \sum_{i=0}^{N} (e^{-j2\pi \frac{k}{M} i}) b_i, & 0 \leq k \leq N \\ 0, & \text{otherwise} \end{cases} \tag{A.5}$$

and the corresponding inverse transformation, called Inverse Discrete Fourier Transform (IDFT) (86)

$$z_i = \begin{cases} \sum_{k=0}^{N} (\frac{1}{M} e^{j2\pi \frac{k}{M} i}) B[k], & 0 \leq k \leq N \\ 0, & \text{otherwise} \end{cases} \tag{A.6}$$

Rewriting (A.6) in vectorized form, we obtain

$$z_i = \boldsymbol{W}^T \boldsymbol{B} \tag{A.7}$$

where $\boldsymbol{W} = \{(\frac{1}{M} e^{j2\pi \frac{k}{M} i})\}, 0 \leq k \leq N$, and $\boldsymbol{B} = \{B_0, B_1 \ldots B_N\}$.

[4] This article assumes that vectors are always column vectors, as it is usual in the signal processing area.

[5] The imaginary unit is represented by j.

Compare (A.7) and (A.4). Note that these equations are equal if $\boldsymbol{a} = \boldsymbol{B}$. Therefore, Eq. (A.7) suggests that it would be possible to represent the function $z = f(a)$ through a neural network that uses the weights $\{\frac{1}{M}e^{j2\pi\frac{k}{M}i}\}, 0 \leq k \leq N$. Eq. (A.7) looks like a Fourier series.

Remember that Fourier showed in 1807 that an arbitrary and aperiodic function $f(t)$ defined in a finite interval T_0 can be reconstructed from a trigonometric series called the Fourier series (62). So "there is nothing new under the sun". Electrical engineers are well familiarized with this notion.

As mentioned before, the universal approximation theorem states that a feedforward network with a linear input and at least one hidden layer of artificial units with an non linear activation function can approximate any "function"[6] from one finite-dimensional space to another with any desired nonzero amount of error, provided that the network is given enough units (43, 59). However, the theorem does not say what the number of units in the hidden layer should be. We also have no guarantees that the training algorithm will be able to learn that function. This may be due to the existence of local minimums in the cost function to be optimized.

Nowadays, the activation function called REctified Linear Unit (Relu) (see Fig13), given by

$$g(z) = \max\{0, z\} \tag{A.8}$$

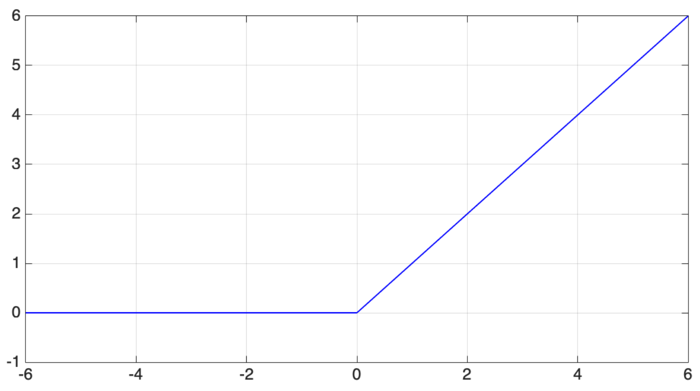


Figura 13 – Relu activation funcion.

is commonly used in ANN (43).

An ANN is a distributed parallel processing system, inspired by the processing structure of the human brain. The ANN technique is a form of non-algorithmic computation of functions.

[6] In fact, any Borel function. This concept is beyond the scope of this paper and will not be discussed.

The connectionist approach uses ANN. The symbolic line, sometimes called the symbolic AI ("Good Old Fashioned AI (GOFAI) (51)) follows the logical tradition and had John McCarthy and Allen Newell, among others, as some of its great exponents (54).

Since 2006, the connectionist line has gained prominence, due in large part to the fundamental contributions of the researchers Yoshua Bengio, Geoffrey Hinton, and Yann LeCun, who were awarded the 2018 A. M. Turing Award (87). Nowadays AI research is dominated by systems that use ANN, also known as Deep Learning (43). Fig. 15 shows a deep learning model for handwritten digit recognition (a classical problem in computer vision), first solved by LeCun et al (88). The model has one input layer, four hidden layers, and one output layer.

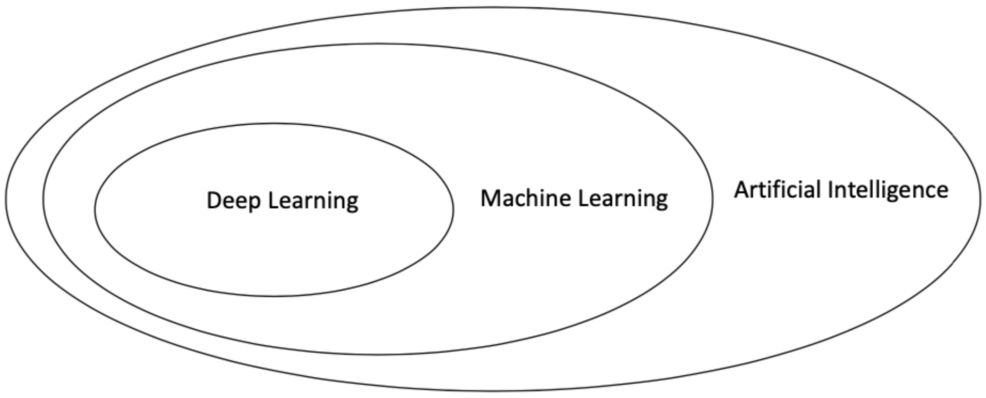


Figura 14 – Relationship between AI, machine learning and deep learning.

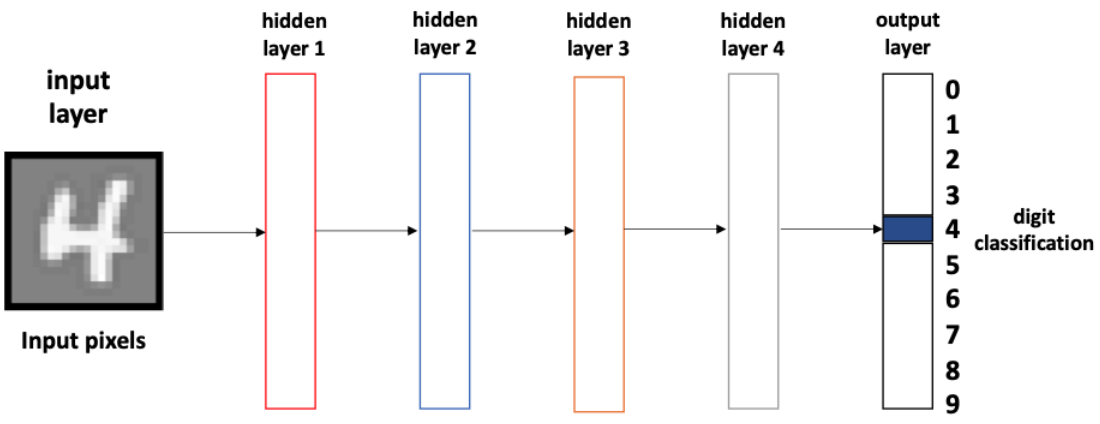


Figura 15 – Handwritten Digit Recognition with a 6 layers ANN.

A.3.2 Machine Learning

Starting at 1952, Arthur Samuel, the pioneer of machine learning in the era of digital computers (89), wrote a series of programs for IBM computers that learned to play checkers [7] through a learning process that was not based on neural networks. The program was shown on TV at 1956, making a strong impression on public opinion (90, 91).

Tom Mitchell gives a definition of machine learning (92, p. 2)

> " A computer is said to **learn** from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E".

As Samuel have shown, checkers is a problem that can be solved using machine learning algorithms. According to Mitchell's definition, we have that:

- E = " training " of the program, which consists of the program playing a sufficiently large number[8] of checkers games against itself.
- T = playing games.
- P = percent of games won against opponents.

Machine learning algorithms can be divided into four groups (25):

- Supervised Learning: in which the man provides the input data (training set), as well as the expected responses from the data (a label is attached to each data). The output of the processing is a set of rules, which will be applied on new input data, in order to produce original responses (inferences of the classification or regression type).
- Unsupervised Learning: for training a descriptive model capable of recognizing patterns. There is no teacher. Applications: discover shopping patterns in supermarkets and data clustering tasks (eg, identifying groups of people who share common interests such as sports, religion or music).
- Self-supervised Learning: is a type of supervised learning; there are still labels involved (because learning needs to be supervised by something), but they are generated from the input data, usually using a heuristic algorithm.
- Reinforcement Learning: a program is trained to interact in a environment through actions to achieve a goal. Learning uses a rewards and/or punishment mechanism. The AlphaZero algorithm[9], developed by Google DeepMind, uses this technique (48).

[7] The work was developed during Samuel's free time.
[8] For example, hundreds of thousands of games.
[9] AlphaZero defeated the world champion of Go, Lee Sedol, in a challenge sponsored by Google in 2016 in South Korea.

Supervised learning, which is used in this work, involves the training of a model or hypothesis. To this end, given a set of training data (or training set), the learning algorithm aims to fit a model that minimizes a metric of choice.

In 1959, Bernard Widrow and Martian E. Hoff discovered the famous machine learning algorithm called Least Mean Square (LMS) (93) The LMS algorithm belongs to the family of stochastic gradient algorithms and uses the method of optimization of the gradient descent, which is also used in deep learning, whose objective is to find the values of the parameters that minimize a given function, called objective function or cost function.

Adaline implements supervised machine learning, since the desired response for each input pattern is provided.

In 1958, Frank Rosenblatt presented the theory of a hypothetical nervous system called the Perceptron (94). Perceptron is a single-layer ANN with a learning rule capable of implementing a linear classifier.

A Perceptron follows the feed-forward processing model, from left (inputs) to right (output (s)).

The 1980s were marked by the discovery of the backpropagation algorithm, which updates the gradient for multilayer networks (95). The wide dissemination of the results in the collection *Parallel Distributed Processing* (95), edited by Rumelhart and McClelland, caused great excitement in the areas of Computer Science and Psychology.

In fact, the backpropagation algorithm was discovered independently several times (96, 56, 57, 58).

A.3.3 Statistical Learning

This section introduces a very short review of the main concepts about stochastic processes (also called data-generating processes by the literature of deep learning (43)) and statistical learning that are used in this paper. The purpose is just to indicate what is important to be known without entering into the details. The interested reader should refer to the appropriate literature (76, 77).

Definition A.3.3.1 (Stochastic Process)**.** *Let T be an arbitrary set. A stochastic process is a family $\{\mathbf{x}_t, t \in T\}$, such that, for each $t \in T$, $\mathbf{x}_t$ is a random variable.* ■

When the set T is the set of integer numbers $\mathbb{Z}$, then $\{\boldsymbol{x}_t\}$ is a discrete time stochastic process (or random sequence); $\{\boldsymbol{x}_t\}$ is a continuous time stochastic process if T is taken as the set of real numbers $\mathbb{R}$.

The random variable $\boldsymbol{x}_t$ is, in fact, a function of two arguments $\boldsymbol{x}(t,\zeta)$, $t \in T$, $\zeta \in \Omega$, given that it is defined over the sample space Ω. For each $\zeta \in \Omega$ we have a realization, trajectory or time series x_t. The set of all realizations is called ensemble. Each trajectory is a function or a non-random sequence and for each fixed t, x_t is a number.

A process $\boldsymbol{x}_t$ is completely specified by its *finite-dimensional distributions* or n-order probability distribution functions, as:

$$F_{\boldsymbol{x}}(x_1, x_2, \ldots, x_n; t_1, t_2, \ldots, t_n) = P\{\boldsymbol{x}(t_1) \leq x_1, \boldsymbol{x}(t_2) \leq x_2, \ldots, \boldsymbol{x}(t_n) \leq x_n\} \tag{A.9}$$

in which $t_1, t_2, \ldots, t_n$ are any elements of T and $n \geq 1$.

The first order probability distribution function is also known as Cumulative Distribution Function - CDF.

The *probability density function - PDF* is given by:

$$f_{\boldsymbol{x}}(x_1, x_2, \ldots, x_n; t_1, t_2, \ldots, t_n) = \frac{\partial^n F_{\boldsymbol{x}}(x_1, x_2, \ldots, x_n; t_1, t_2, \ldots, t_n)}{\partial x_1 \partial x_2 \ldots \partial x_n}. \tag{A.10}$$

Applying the conditional probability density formula,

$$f_{\boldsymbol{x}}(x_k | x_{k-1}, \ldots, x_1) = \frac{f_{\boldsymbol{x}}(x_1, \ldots, x_{k-1}, x_k)}{f_{\boldsymbol{x}}(x_1, \ldots, x_{k-1})}, \tag{A.11}$$

in which $f_{\boldsymbol{x}}(x_1, \ldots, x_{k-1}, x_k)$ denotes $f_{\boldsymbol{x}}(x_1, \ldots, x_{k-1}, x_k; t_1, \ldots, t_{k-1}, t_k)$, repeatedly over

$$f_{\boldsymbol{x}}(x_1, \ldots, x_{n-1}, x_n)$$

we get the probability chain rule

$$f_{\boldsymbol{x}}(x_1, x_2, \ldots, x_n) = f_{\boldsymbol{x}}(x_1) f_{\boldsymbol{x}}(x_2|x_1) f_{\boldsymbol{x}}(x_3|x_2, x_1) \ldots f_{\boldsymbol{x}}(x_n|x_{n-1}, \ldots, x_1). \tag{A.12}$$

When $\boldsymbol{x}_t$ is a sequence of *mutually independent* random variables, (A.12) can be rewritten as

$$f_{\boldsymbol{x}}(x_1, x_2, \ldots, x_n) = f_{\boldsymbol{x}}(x_1) f_{\boldsymbol{x}}(x_2) \ldots f_{\boldsymbol{x}}(x_n). \tag{A.13}$$

Definition A.3.3.2 (Purely Stochastic Process)**.** *A purely stochastic process* $\{\mathbf{x}_t, t \in \mathbb{Z}\}$ *is a sequence of mutually independent random variables.* ■

Definition A.3.3.3 (IID Process)**.** *An Independent and Identically Distributed (IID) process* $\{\mathbf{x}_t, t \in \mathbb{Z}\}$*, denoted by* $\mathbf{x}_t \sim IID$*, is a purely stochastic and identically distributed process.* ■

As mentioned earlier, the central challenge in machine learning is that the algorithm performs well on the test set. The training and test sets are generated by the same data-generating process. Typically, it is assumed that the examples in each set are independent and that the training and test sets are identically distributed. These assumptions are collectively known as IID.

The **no free lunch theorem** (78) says that, considering the average over all the possible data-generating processes, any machine learning algorithm has the same error rate when evaluated on previously unobserved examples. That is, there is not, at least in theory, a machine learning algorithm that is better than all others for all cases.

However, the no free lunch theorem is valid only when working with the average over all the possible data-generating processes. Fortunately, that does not happen in real life, as the physical data-generating process of a Li-Ion cell is a result of the restrictions imposed by the real world over all the possible data-generating processes. Thus, in practical applications, it is realistic to think about designing algorithms that have a good performance for a given Li-Ion cell.

Going further, the goal of research in machine learning is not to look for a universal learning algorithm. Instead, we have to understand what the stochastic characteristics of our dataset are, in order to design, validate and test an algorithm that is efficient for that specific data set.

A.3.4 ANN as Deep Learning

Interest in ANNs has resurfaced with the advent of the Deep Belief Nets in 2006 (63). The work of Hinton, Osindero and Teh demonstrated that a type of DNN could be trained with high efficiency. Their research triggered the current wave of research in ANN, which popularized the term deep learning.

Deep learning denotes the idea of a neural network with multiple hidden layers that has the ability to partition the representation of an entry into multiple layers (see Fig.**??**). The number of layers in a model corresponds to the depth of the network (25).

The success of deep learning today is due to: i) the emergence of Big Data, which made it possible to store data for training in databases with tens of millions of examples, ii) the advent of Graphics Processing Unit (GPU), and iii) advances in algorithms.

Fig. 16 illustrates how deep learning works (25). The variables $\boldsymbol{x}$ and $\boldsymbol{y}$ denote the input signal (training example) and the desired signal (target), respectively. The function $\boldsymbol{h_w}(\boldsymbol{x})$ is the mathematical model or hypothesis. The estimation error (residual or loss score) is given by $\boldsymbol{e} = \boldsymbol{y} - \boldsymbol{h_w}(\boldsymbol{x})$. At startup, random values are assigned to the $\boldsymbol{w}$ weights of the network, so the value of the initial residue is high. However, in the course of processing the training examples, the weights are adjusted incrementally in the correct direction; at the same time, the value of the loss function decreases. This is the training loop, which, being repeated enough times, typically dozens of iterations over thousands of examples, produces weights that minimize the cost function. A network is considered trained when the minimum of the cost function is reached.

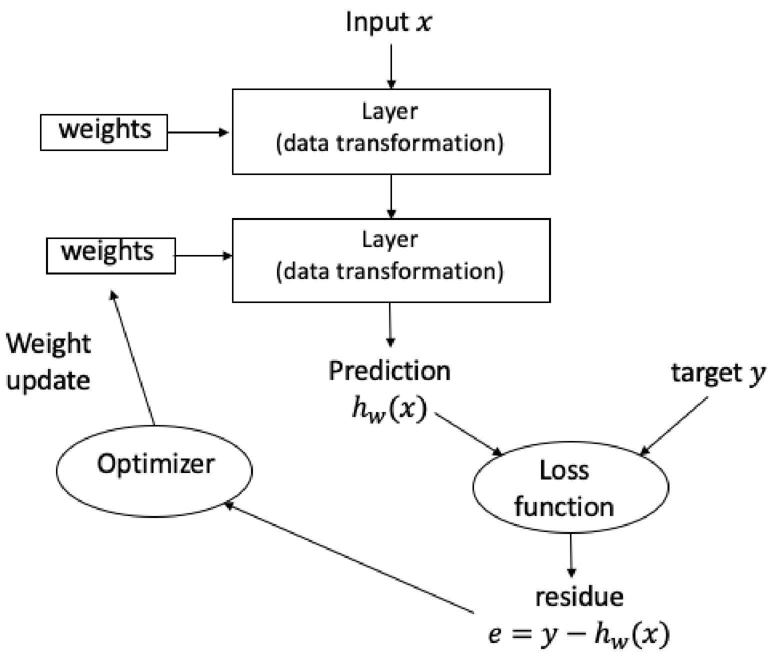


Figura 16 – How deep learning works.

Deep learning has achieved the following advances, all in historically difficult areas of machine learning, such as (68, 25):

- nonlinear regression;
- superhuman image classification;
- voice recognition at an almost human level;
- transcription of almost human handwriting;
- automatic translation;
- text to speech conversion;
- autonomous vehicle driving at an almost human level; and
- superhuman performance in games like chess, shogi and Go.

However, deep learning has limitations. Current ANN architectures do not have the power to track statistical changes of the training data in real time. In other words, deep learning is not yet adaptive. Note that training a DNN at a dedicated workstation can take days or weeks. This is due to the computational complexity of deep learning.

Furthermore, the science of deep learning is not like mathematics or physics, in which theoretical advances can be achieved with a chalk and a blackboard. Deep learning is an engineering science (64), as it does not yet have a mathematical formalism like that of the area of adaptive filtering. For example, as we have stated before, there is no design criterion for the number of layers in the network, much less for the number of neurons in a hidden layer. The field is driven by experimental discoveries. But of course, there are best practices to be followed (25).

A.3.5 A Workflow for Approaching Deep Learning Problems

In this paper, we follow an adapted version of the supervised machine learning workflow proposed by Chollet (see Fig. 17) (25):

1. Choose a reliable dataset. If you do not find a dataset, collect the data of interest, and annotate it with labels.
2. Choose how you will measure success on your problem. Which metrics will you monitor on your validation data?
3. Determine your evaluation protocol: K-fold cross-validation? Which portion of the data should you use for validation?
4. Develop a baseline model with statistical power.
5. Develop a model that overfits.
6. Regularize your model and tune its hyperparameters, based on performance on the validation data.

A.4 Experimental Results

A.4.1 Deep Learning Methodology and Modeling

We selected the 2.9 Ah Panasonic 18650PF Li-ion Battery Data provided by Dr. Phillip Kollmeyer, University of Wisconsin-Madison (20). Note that this dataset has been used by some of the top works in the area (79, 14, 83, 84).

A series of nine drive cycles were made public, among them a Neural Network (NN) cycle, which is the cycle of our interest. More specifically, the simulations in this section present the results for data collected at a temperature of 25^o C.

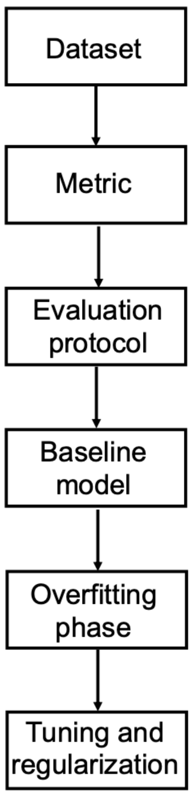


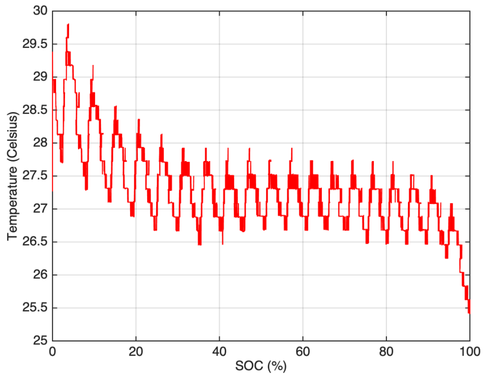
Figura 17 – Deep learning workflow.

The NN drive cycle was designed to have some additional dynamics which are useful for training neural networks. The drive cycle power profile is calculated for an electric Ford F150 truck with a 35kWh battery pack scaled for a single 18650PF cell.

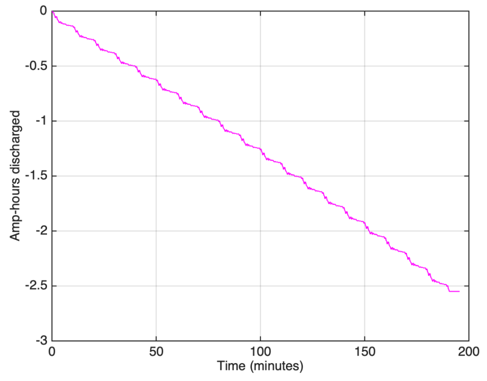
Fig. 18 shows the following 2.9 Ah Panasonic 18650PF Li-ion cell characteristic curves:

- temperature (^{o} C) vs SOC (%);
- amp-hours discharged vs time (minutes);
- voltage (V) vs time (minutes);
- current (A) vs time (minutes);
- temperature (^{o} C) vs time (minutes); and
- voltage (V) vs SOC (%).

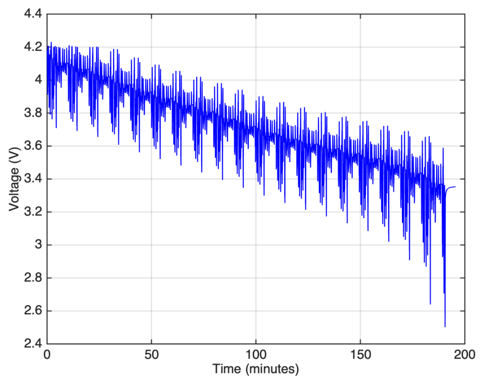
The input data ($\boldsymbol{x}$) or features are: $x_1 = v(t)$ (voltage in V), $x_2 = i(t)$ (current in A), and $x_3 = T(t)$ (temperature in ^{o}C), where t denotes time in seconds. We see no reason for the inclusion of extra features in the hypothesis space, as the other data collected in



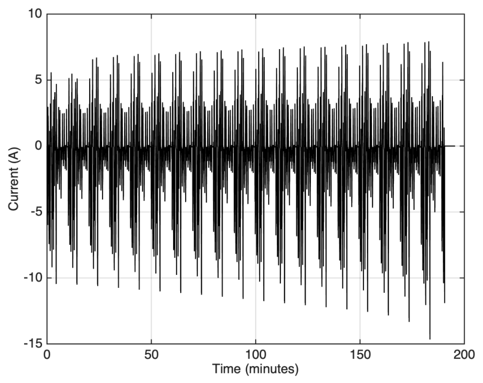
(a)



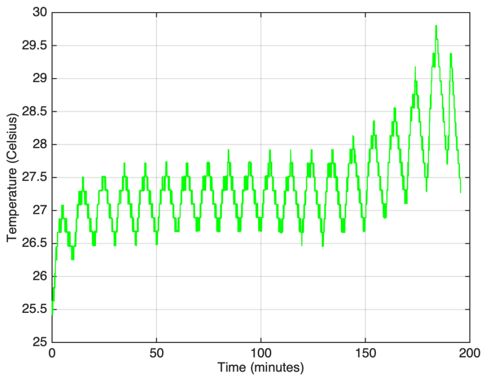
(b)



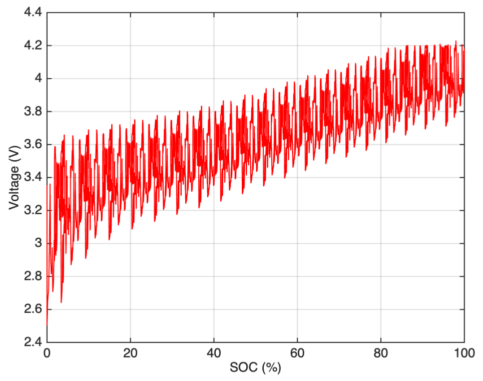
(c)



(d)



(e)



(f)

Figura 18 – in (a), (b), (c), (d), (e), and (f) we have: temperature (^{o} C) vs SOC (%), amp-hours discharged vs time (minutes), voltage (V) vs time (minutes), current (A) vs time (minutes), temperature (^{o} C) vs time (minutes), and voltage (V) vs SOC (%), respectively.

(20) are: i) Wh (measured watt-hours, with Wh counter reset after each charge, test, or drive cycle), Power (measure power in watts), and Chamber-Temp-degC (measured chamber temperature in degrees Celsius). The output variable or target ($\boldsymbol{y}$) is the SOC (%). The dataset has $116,982$ examples, which were divided examples for training, validation, and testing, respectively.

We applied feature normalization on the input data using the formula

$$\boldsymbol{x}_{\text{normalized}} = \frac{\boldsymbol{x} - \mu_{\boldsymbol{x}}}{\sigma_{\boldsymbol{x}}} \tag{A.14}$$

where $\mu_{\boldsymbol{x}}$ and $\sigma_{\boldsymbol{x}}$ denote the mean and standard deviation of $\boldsymbol{x}$.

We choose Mean Absolute Error (MAE) as the performance metric (14) and the K-fold cross-validation as the basic method to fight overfitting. Notwithstanding, we have also used weight regularizers and dropout layers with the same goal.

The literature review indicated that the `Python` language is the most suitable for this research, for it has several free and open source frameworks for deep learning. In addition, `Python` is the language most used by the machine learning community (25, 26, 27). Other options like `R` also have great machine learning libraries (64).

Open source deep learning frameworks such as TensorFlow (30), PyTorch (31), MXNet (32) and Microsoft Cognitive Toolkit (CNTK) (33)[10] have stood out in recent years. However, TensorFlow 2 and the Keras API offer some differentials, such as the possibility of executing codes in Google Collaboratory, or "Colab" (35), just using a browser and with free access to CPU/GPU (and even to Google's Tensor Processor Unit (TPU)).

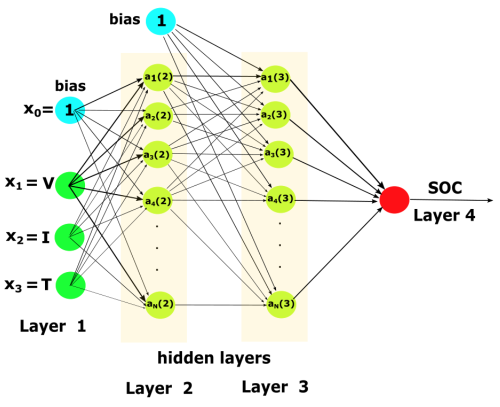
In addition, TensorFlow offers a browser-based visualization tool called TensorBoard, whose main objective is to help the user visually monitor everything that happens inside their model during training (25). TensorBoard automates some features such as visualizing the learning curve of neural networks.

Thus, we decided to use the `Python` language and the TensorFlow 2 framework in conjunction with the Keras API. We coded/prototyped the deep learning model in the Spyder IDE (`Python` 3.7) of the Anaconda package manager.

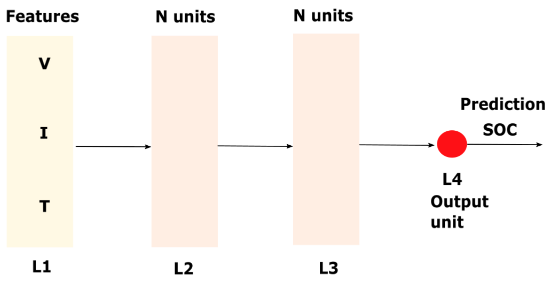
A.4.2 Tuning of Hyperparameters

Fig. 19 shows the architecture of a densely connected DFN with four (4) layers (two hidden layers). Fig. 20 shows the learning curves using Adam optimizer and 4-fold cross-validations. Note that overfitting manifests itself as a gap between the validation MAE (red plot) and the training MAE (blue plot) in those figures.

[10] The Caffe2 (34) library has been absorbed by PyTorch.



(a)



(b)

Figura 19 – (a) and (b): densely connected 4-layer DFN and its schematic version, respectively.

We tuned the hyperparameters of the deep learning model using L^1/L^2 parameter norm penalties and adding two extra dropout layers.

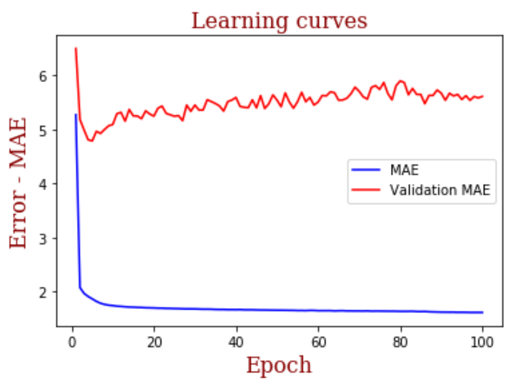
In L^2 regularization, the cost added to the objective function is proportional to the square root of the sum of the square values of the weight coefficients (L^2 norm – $||\boldsymbol{w}||_2^2$), whereas in L^1 regularization the cost added to the objetive function is proportional to the sum of the absolute values of the weight coefficients (L^1 norm – $||\boldsymbol{w}||_1$).

Fig. 21 shows the architecture of a DFN with six (6) layers, where we have, after the input layer (layer 1), in sequence, two pairs of a 256 units/hidden layer followed by a dropout layer, then the final layer. The neural net uses 8-fold cross-validations. In Fig. 22, note that overfitting occurs only around 35 epochs. Thus, this model has a greater power of generalization than the previous ones, as expected.

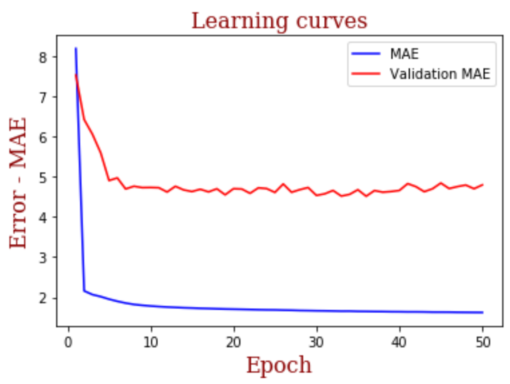
A.4.3 SOC Estimation Results

The DFN model with two hidden layers, 256 units/hidden layer, batch size of 128 and without regularization achieves the best SOC's estimate on the test set, with a MAE of approximately 1.60%.

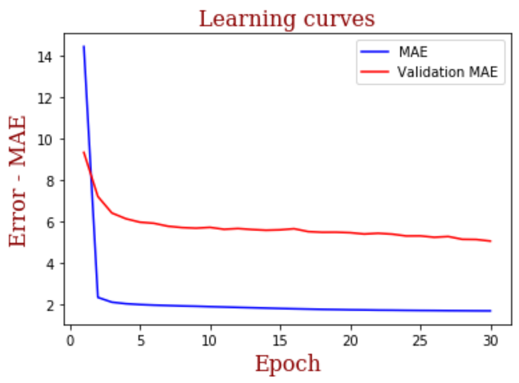
However, as indicated by the learning curves in Fig. **??**, the DFN model with four hidden layers (two pairs of a 256 units/hidden layer followed by a dropout layer with a dropout rate of 0.5) has a greater power of generalization than the previous model. The MAE obtained on the test set was approximately 2.0% in this case.



(a)



(b)



(c)

Figura 20 – in (a), (b), and (c), we have learning curves for: 256 units/hidden layer, batch size of 64 and 100 epochs; 256 units/hidden layer, batch size of 128 and 50 epochs; and 256 units/hidden layer, batch size of 256, and 30 epochs, respectively.

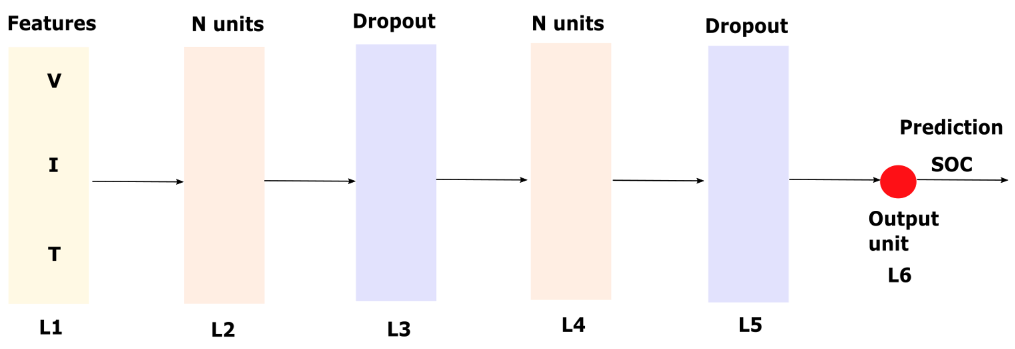


Figura 21 – DFN with six (6) layers.

A.5 Conclusions

This paper presents two simple DFN models with two and four hidden layers, respectively, using an optimizer with adaptive learning rules, and the Relu activation function, in order to estimate the State of Charge (SOC) of a Panasonic 18650PF lithium-

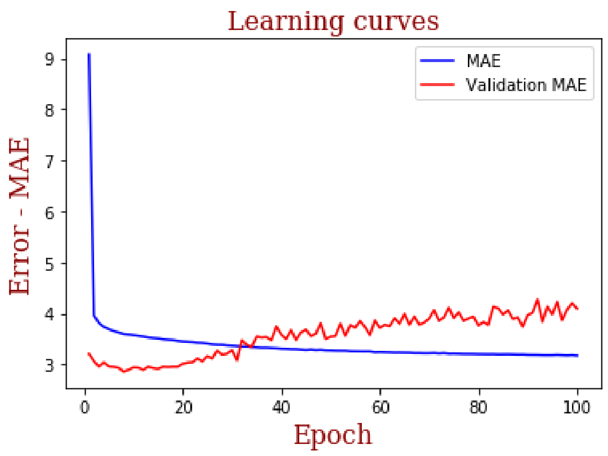


Figura 22 – learning curves for a model with 256 units/hidden layer, batch size of 128, dropout rate of 0.5.

ion battery of the Neural Network (NN) drive cycle of dataset (20) using the K-fold cross-validation method.

The DFN model with four hidden layers presents a better power of generalization, not only because it has a greater capacity in terms of more layers, but also due to the application of additional regularization techniques such as dropout layers and parameter norm penalties. The contribution of this work is to present a methodology of building a DFN for a lithium-ion battery and its performance assessment, which follows the best practices in machine learning.

APÊNDICE B – State-of-charge Estimation of a Li-ion Battery using Deep Learning and Stochastic Optimization

Authors: Alexandre B. de Lima, Maurício B. C. Salles, and José Roberto Cardoso.
Electronic preprint: <https://arxiv.org/abs/2011.09673>, november, 2020.

B.1 Abstract

This article presents a novel empirical study for the estimation of the State of Charge (SOC) of a lithium-ion (Li-ion) battery which uses a deep learning model with three hidden layers. We model a series of ten vehicle drive cycles that were applied to a Panasonic 18650PF Li-ion cell. Our results show that the choice of the optimization algorithm affects the model performance. The proposed model was able to achieve an error smaller than 1.0% in all drive cycles.

Keywords: Deep Learning, Electrical Energy Storage, Li-ion battery, and State-Of-Charge.

B.2 Introduction

In the last decade, government, industry and academia have given great importance to the electrification of the transport system, motivated by the need to reduce the emission of greenhouse gases. Hybrid electric vehicles, such as the Toyota Prius, or fully electric vehicles, such as the various Tesla models, the Nissan Leaf and the Chevy Bolt, are successful cases in the USA (6).

The United Kingdom took a bold step in this direction in 2018. The Executive Summary of the document *The Road to Zero: Next steps towards cleaner road transport and delivering our Industrial Strategy* (United Kingdom) states that (97):

> "Our strategy is built around a core mission: to put the UK at the forefront of the design and manufacturing of *zero emission vehicles* and for all new cars and vans to be effectively zero emission by 2040.
>
> (...)

as we move to the mass adoption of ultra low emission vehicles, more infrastructure will be needed and we want to see improvements to the consumer experience of using it. Our vision is for current and prospective *electric vehicle* drivers to be able to easily locate and access charging infrastructure that is affordable, efficient and reliable.

(...)

The electricity system of 2050 will look very different from today?s. There will be more low carbon generation, and *new technologies such as battery storage* and onsite generation will play a bigger role. Decarbonising how we heat our homes and businesses will also bring further changes to the demands placed on the energy system." (our highlight)

The advancement of Electrical Energy Storage (EES) technologies enabled the emergence of the iPod, smartphones and tablets with lithium-ion (li-ion) batteries. Also, EES (cited above as "battery storage") will be one of the critical components of the new electricity grid, given the intermittent nature of renewable energy sources (6), (7). EES systems are necessary even when renewable sources are connected to the grid, because it is necessary to smooth the energy supply. For example, the EES of a building or factory can be charged during hours of reduced demand and supply/supplement energy demand during peak hours.

EES technology consists of the process of converting a form of energy (almost always electrical) to a form of storable energy, which can be converted into electrical energy when necessary. EES has the following functions: to assist in meeting the maximum electrical load demands, to provide time-varying energy management, to relieve the intermittency of renewable energy generation, to improve energy quality/reliability, to serve remote loads and vehicles, to support the realization of smart grids, improve the management of distributed/standby power generation and reduce the import of electricity during peak demand periods (7), (8).

The efficient use of the Li-ion battery requires the supervision of a Battery Management System (BMS), as it is necessary that the battery operates under appropriate conditions of temperature and charge (State-Of-Charge (SOC)) (10). The SOC can be measured using the Coulomb counting method of (B.1)

$$\mathrm{SOC} = \mathrm{SOC}_0 - \frac{\int_o^t I_{\mathrm{bat}}\, dt}{Q_n} \tag{B.1}$$

where SOC_0 is the initial value of SOC, I_{bat} is the battery current and Q_n is the nominal capacity in Ah. It should be noted that the cell temperature produces deleterious effects on the open circuit voltage, internal resistance and available capacity and can also lead to a rapid degradation of the battery if it operates above a given temperature threshold.

Therefore, the modeling of the battery is of paramount importance, since it will be used by the BMS to manage the operation of the battery (9).

The recent literature suggests that the machine learning approach, based on deep learning algorithms is the state of the art in this area (98), (99), (11), (79), (14), (83), (84), (15).

One of the great challenges in deep learning is the optimization of the neural network. Although the Stochastic Gradient Descent (SGD) algorithm (and its variants) is very popular, a learning rate too small leads to painfully slow convergence, while a learning rate too high can destabilize the algorithm, causing oscillations or divergence (43).

On the other hand, we have at our disposal algorithms with adaptive learning rates such as RMSProp (100) and Adamax (43), (101).

To the best of our knowledge, there is no study on SOC estimation of Li-ion batteries that investigates the effect of choosing the optimization algorithm on the performance of the deep learning model. That is our contribution. The question we pose is: which optimization algorithm should we choose?

For example, (98) “postulates” the use of the Adagrad algorithm (102) for the optimization of a convolutional gated recurrent unit (CNN - GRU) architecture. However, as we will see in section B.4, sometimes the “good old fashioned” SGD can beat an algorithm with adaptive learning rules like Adagrad, at the cost of speed, as “there is no free lunch”. Things are more complicated in the field of neural network optimization (43).

This work uses a deep learning model with three hidden layers. We model a series of ten vehicle drive cycles that were applied to a Panasonic 18650PF Li-ion battery (20). More specifically, we compare the performance of the following algorithms: RMSProp (100), Adamax (101), and SGD (43). Our results show that the choice of the optimization algorithm affects the model performance. The issue of regularization (parameter norm penalties, early stopping, dropout, etc.) is outside the scope of this work. Nevertheless, the power of generalization of the proposed deep learning model is high, since it was obtained a Mean Absolute Error (MAE)/Mean Squared Error (MSE) lower than 1.0% for all of the ten drive cycles.

The remainder of the paper is organized as follows. Section B.3 presents the state of the art and trends in Li-ion battery parameter estimation. Section B.4 presents our experimental results. Finally, section B.5 presents our conclusions.

B.3 State of the Art and Trends in Li-ion Battery Estimation

Energy storage acts as a mediator between variable loads and variable sources.

Hannan et al. (5) present a detailed taxonomy of the types of energy storage systems taking into account the form of energy storage and construction materials: mechanical, electrochemical (rechargeable and flow batteries), chemical, electrical (ultracapacitor or superconducting magnetic coil), thermal and hybrid. Li-ion battery technology has attracted the attention of industry and academia for the past decade. This is mainly due to the fact that Li-ion batteries offer more energy, higher power density, higher efficiency and lower self-discharge rate than other battery technologies such as NiCd, NiMH, etc. (9).

There are two methods of battery modeling: i) model-driven and ii) data-driven (based on data that is collected from the device) (11).

Electrothermal models, which belong to the category of model-driven methods, are commonly classified as: i) electrochemical or ii) based on Equivalent Circuit Models (ECM) (10), (9).

Electrochemical models are based on partial differential equations (12) and are able to represent thermal effects more accurately than ECM (13). However, the first class of models requires detailed knowledge of proprietary parameters of the battery manufacturer: cell area, electrode porosity, material density, electrolyte characteristics, thermal conductivity, etc. This difficulty can be eliminated by characterizing the battery using a thermal camera and thermocouples. But this solution is expensive, time consuming and introduces other challenges such as the implementation of dry air purge systems, ventilation, security, air and water supply, etc. Electrochemical models demand the use of intensive computing systems (10).

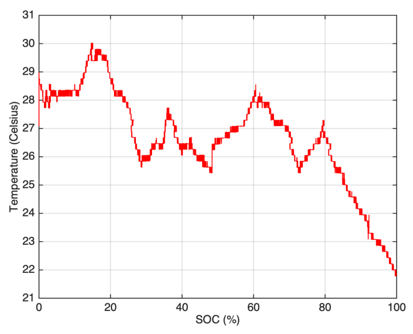
On the other hand, the ECM-based approach has been used for computational/numerical analysis of batteries (10). In this case, the objective is to develop an electrical model that represents the electrochemical phenomenon existing in the cell. The level of complexity of the model is the result of a compromise between precision and computational effort. Note that an extremely complex and accurate ECM may be unsuitable for application in embedded systems.

Chemali et al (14) compared the performance of Deep Neural Networks (DNN) with those of other relevant algorithms that have been proposed since the second half of the 2000s. The article shows that the SOC estimation error obtained with deeep learning is less than the following methods:

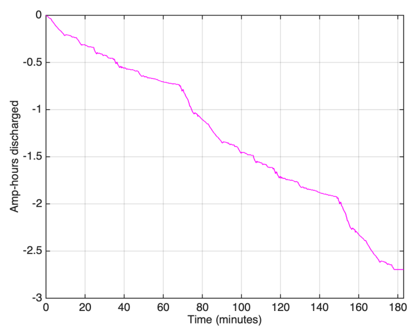
- Model Adaptive-Improved Extended Kalman Filter (EKF) (70);
- Adaptive EKF with Neural Networks (71);
- Adaptive Unscented Kalman Filter (AUKF) with Extreme Machine Learning (72);
- Fuzzy NN with Genetic Algorithm (73); and

- Radial Bias Function Neural Network (74).

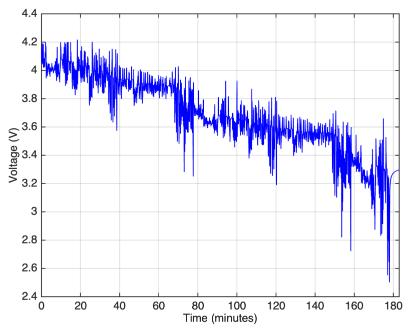
Estimating the SOC of lithium ion cells in a BMS by means of deep learning offers at least two significant advantages over model driven approaches, namely: i) neural networks are able to estimate the non linear functional dependence that exists between voltage, current and temperature (observable quantities) and unobservable quantities, such as SOC, with great precision and ii) the problem of identifying ECM parameters is avoided.



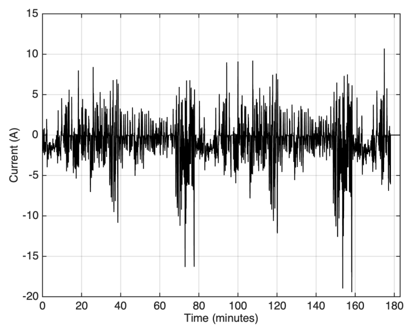
(a)



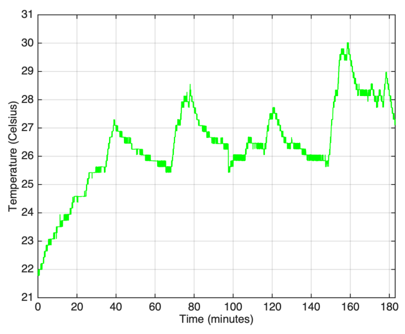
(b)



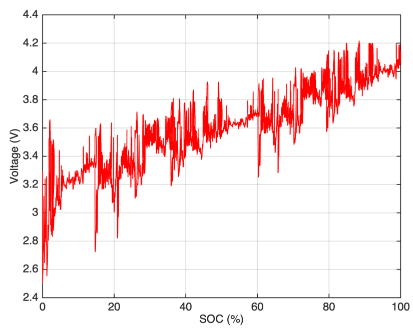
(c)



(d)



(e)



(f)

Figura 23 – in (a), (b), (c), (d), (e), and (f) we have (for the drive Cycle 1): temperature (^{o} C) vs SOC (%), amp-hours discharged vs time (minutes), voltage (V) vs time (minutes), current (A) vs time (minutes), temperature (^{o} C) vs time (minutes), and voltage (V) vs SOC (%), respectively.

B.4 Experimental Results

We selected the *Panasonic 18650PF Li-ion Battery Data* of (20). This dataset contains a series of ten drive cycles: Cycle 1, Cycle 2, Cycle 3, Cycle 4, US06, HWFTa, HWFTb, UDDS, LA92, and Neural Network (NN). Cycles 1-4 consist of a random mix of US06, HWFET, UDDS, LA92, and Neural Network drive cycles. The drive cycle power profile is calculated from measurement for an electric Ford F150 truck with a 35kWh battery pack scaled for a single 18650PF cell. We consider only the tests at the temperature of 25^o C, as the objective of this paper is not to assess the performance of the deep learning model at different temperatures, but to empirically investigate the question of stochastic optimization in the context of the problem of SOC estimation with deep neural nets.

For instance, Fig. 23 shows the following 2.9 Ah Panasonic 18650PF Li-ion cell characteristic curves for Cycle 1.

- temperature (^{o} C) vs SOC (%);
- amp-hours discharged vs time (minutes);
- voltage (V) vs time (minutes);
- current (A) vs time (minutes);
- temperature (^{o} C) vs time (minutes); and
- voltage (V) vs SOC (%).

We applied feature normalization on the input data using the formula

$$\boldsymbol{x}_{\text{normalized}} = \frac{\boldsymbol{x} - \mu_{\boldsymbol{x}}}{\sigma_{\boldsymbol{x}}} \tag{B.2}$$

where $\mu_{\boldsymbol{x}}$ and $\sigma_{\boldsymbol{x}}$ denote the mean and standard deviation of $\boldsymbol{x}$.

We divided the datasets into training and validations sets. The validation error is estimated by taking the average validation error across $K = 4$ trials. We use a simple, but popular solution, called K-fold cross-validaton (Fig. 24), which consists of splitting the available training data into two partitions (training and validation), instantiating K identical models, for each fold $k \in \{1, 2, \dots, K\}$, and training each one on the training partitions, while evaluating on the validation partition. The validation score for the model used is then the average of the K validation scores obtained. This procedure allows network hyperparameters to be adjusted so that overfitting is mitigated (43) . It is usual to use about 80% of the data for the training set, and 20% for the validation set. Note that the validation scores may have a high variance with regard to the validation split. Therefore,

K-fold cross-validaton help us improve the reliability when evaluating the generalization power of the model.

We used MAE and MSE as performance metrics for the generalization (test) error (14), (43). After the validation phase using K-fold cross-validaton, the deep learning model is trained using the entire training data and its performance is evaluated against an unseen test set. This the test phase (final phase).

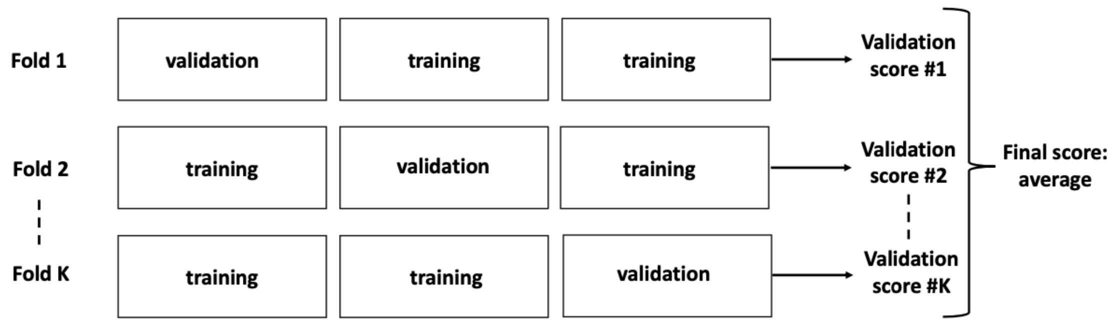


Figura 24 – K-fold cross-validation.

The code was developed in `Python`, using TensorFlow 2 and the Keras API (30), (103).

Our simulations confirm the design matrix chosen by (14), as our best results were found with the following features (inputs): $x_1 = V(t)$ (voltage in V), $x_2 = \bar{V}(t)$ (average voltage in V), $x_3 = \bar{I}(t)$ (average current in A), and $x_4 = \bar{T}(t)$ (average temperature in ^{o}C), where t denotes time in seconds. We used moving averages over 400 past samples in order to smooth the last three features. This procedure improved the estimation of the SOC. The Deep Forward Network (DFN) has three hidden layers, each hidden layer having 256 units (see Fig.25). We used the REctified Linear Unit (Relu) activation function, given by

$$g(z) = \max\{0, z\}. \tag{B.3}$$

Table 1 shows the SOC estimates obtained with SGD, RMSProp and Adamax. We used the default parameters for the Keras `SGD`, `RMSProp`, and `Adamax` classes. The `SGD`class, as well as the others, uses mini-batch gradient descent on n training examples,

$$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} - \eta.\nabla J(\theta; \boldsymbol{x}^{(i:i+n)}; \boldsymbol{y}^{(i:i+n)}) \tag{B.4}$$

where $\boldsymbol{\theta}$ denotes the network parameters, η is the learning rate (also called step size), $\nabla(.)$ is the gradient, and $J(.)$ is the cost function.

RMSProp is given by the following equations:

$$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} - \frac{\eta}{\sqrt{E[g^2]_k + \epsilon}}.g_k \tag{B.5}$$

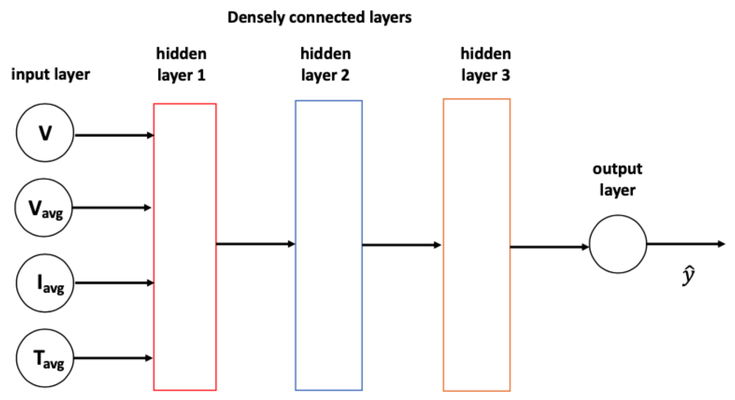


Figura 25 – DFN with three (3) hidden layers. The output is denoted by $\hat{y}$. The network has a total of $133,121$ trainable parameters. The first, second, and third hidden layers have $1,280$, $65,792$, and $65,792$ parameters, respectively. The output layer has 257 parameters (bias plus 256 units of the last hidden layer).

where

$$E[g^2]_k = 0.9E[g^2]_{k-1} + 0.1g_k^2 \tag{B.6}$$

and

$$g_k = \nabla_{\boldsymbol{\theta}_k} J(\boldsymbol{\theta}_k). \tag{B.7}$$

Adamax is a variant of the ADAptive Moment Estimation (Adam) algorithm based on the infinity norm. Adam has the following update rule:

$$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} - \frac{\eta}{\sqrt{\hat{v}_k} + \epsilon}\hat{m}_k \tag{B.8}$$

where

$$\hat{m}_k = \frac{m_k}{1 - \beta_1^k} \tag{B.9}$$

and

$$\hat{v}_k = \frac{v_k}{1 - \beta_w^k}. \tag{B.10}$$

The variables m_k and v_k are estimates of the first and the second moments of the gradients, respectively:

$$m_k = \beta_1 m_{k-1} + (1 - \beta_1)g_k \tag{B.11}$$

$$v_k = \beta_2 v_{k-1} + (1 - \beta_2)g_k^2. \tag{B.12}$$

The proponents of Adam indicate the default values of 0.9 for β_1, 0.999 for β_2, and 10^{-7} for ϵ (103).

The results of Table 1 show that:

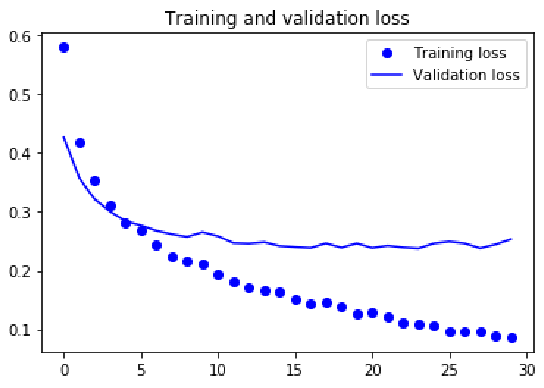


Figura 26 – Training and generalization errors behave differently. The horizontal axis represents the number of epochs. The vertical axis is the loss function. Note that the model starts to overfit around the fifth epoch.

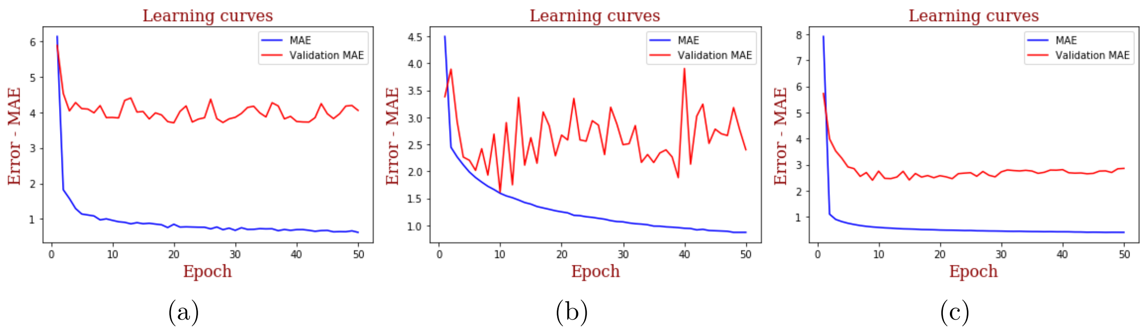


Figura 27 – in (a), (b), and (c) we have: MAE (SGD) vs Epoch, MAE (RMSProp) vs Epoch, and MAE (Adamax) vs Epoch, respectively.

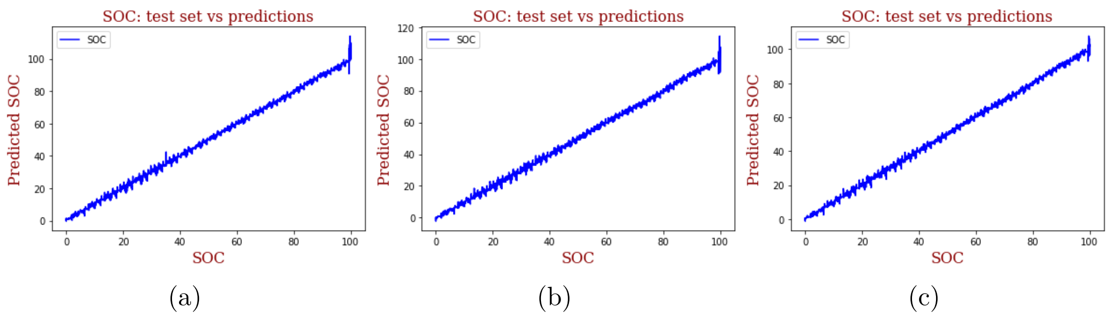


Figura 28 – in (a), (b), and (c) we have: Predicted SOC (SGD) vs Measured SOC, Predicted SOC (RMSProp) vs Measured SOC, and Predicted SOC (Adamax) vs Measured SOC, respectively.

	SGD		**RMSProp**		**Adamax**	
Drive Cycle	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**
Cycle 1	0.81	1.04	0.96	1.55	0.45	0.43
Cycle 2	0.65	0.77	2.83	1.04	0.47	0.47
Cycle 3	0.68	0.91	1.19	2.24	0.57	0.63
Cycle 4	0.48	0.54	0.88	1.29	0.54	0.61
US06	1.01	2.08	1.11	2.15	0.60	0.77
HWFTa	0.40	0.35	0.39	0.34	0.41	0.32
HWFTb	0.41	0.36	1.33	2.54	0.37	0.28
UDDS	0.68	0.70	1.13	2.09	0.49	0.44
LA92	0.59	0.70	0.83	1.19	0.63	0.71
NN	1.08	1.66	0.68	0.90	0.48	0.45

Tabela 1 – MAE and MSE for all drive cycles.

- the choice of the optimization algorithm must be made on a case-by-case basis. Although the family of algorithms with adaptive learning rates (RMSProp, Adam, Adadelta, etc.) has become very popular in the deep learning community nowadays, one can not claim, a priori, that a given optimization algorithm is the best algorithm (43, p. 302);
- the choice of the optimization algorithm affects the model performance. Also note the good performance obtained with the hidden 3-layer DFN model, who was able to achieve a MAE/MSE smaller than 1.0% in all drive cycles.

Figs. 27 and 28 show, for the NN drive cycle, the learning curves (validation phase) and the prediction curves (test phase) for the SOC, respectively.

B.5 Conclusions

We present a preliminary empirical study on the impact of the optimization algorithm on the performance of the deep learning model, in the context of the estimation of the SOC of a Li-ion battery in ten distinct scenarios. For this, we used the dataset (20).

Our results indicate that: i) the choice of the optimization algorithm must be made on a case-by-case basis, and ii) the choice of the optimization algorithm affects the model performance.

Although the family of algorithms with adaptive learning rates (RMSProp, Adam, Adadelta, etc.) has become very popular in the deep learning community nowadays, one can not claim, a priori, that a given optimization algorithm is the best algorithm.

Finally, it is important to highlight the good performance obtained with the hidden 3-layer DFN model, who was able to achieve a MAE/MSE smaller than 1.0% in all drive cycles.

APÊNDICE C – State-of-Charge Estimation of the Panasonic 18650PF Li-ion Cell using Deep Learning Models and Algorithms with Adaptive Learning Rates

Authors: Alexandre B. de Lima, Maurício B. C. Salles, and José Roberto Cardoso. Journal: International Journal of Engineering Research and Applications (IJERA), v. 10, p. 30–34, 2020.

C.1 Abstract

This article presents a novel empirical study for the estimation of the State-of-Charge (SOC) of the Panasonic 18650PF lithium-ion (Li-ion) cell using deep learning models and algorithms with adaptive learning rates. Specifically, we model a vehicle drive cycle designed for training neural networks. Our results suggest that the choice of the optimization algorithm affects the performance of the model and that a Deep Forward Network (DFN) with four hidden layers is the model of optimal capacity when considering 256 units per layer. This optimal DFN is able to estimate the SOC of the 18650PF Li-ion cell with an error smaller than 0.12 % over a 25^o C dataset using the Adam optimization algorithm.

Keywords: Deep Learning, Electrical Energy Storage, Li-ion battery, State-of-Charge.

C.2 Introduction

In the last decade, government, industry and academia have given great importance to the electrification of the transport system, motivated by the need to reduce the emission of greenhouse gases. Hybrid electric vehicles, such as the Toyota Prius, or fully electric vehicles, such as the various Tesla models, the Nissan Leaf and the Chevy Bolt, are successful cases in the USA. The United Kingdom took a bold step in this direction in 2018 (6), (97).

The advancement of Electrical Energy Storage (EES) technologies enabled the emergence of the iPod, smartphones and tablets with lithium-ion (li-ion) batteries. Also, EES will be one of the critical components of the new electricity grid, given the intermittent nature of renewable energy sources (7). EES systems are necessary even when renewable sources are connected to the grid, because it is necessary to smooth the energy supply. For example, the EES of a building or factory can be charged during hours of reduced demand and supply/supplement energy demand during peak hours.

EES technology consists of the process of converting a form of energy (almost always electrical) to a form of storable energy, which can be converted into electrical energy when necessary.

EES has the following functions: to assist in meeting the maximum electrical load demands, to provide time-varying energy management, to relieve the intermittency of renewable energy generation, to improve energy quality/reliability, to serve remote loads and vehicles, to support the realization of smart grids, improve the management of distributed/standby power generation and reduce the import of electricity during peak demand periods (8).

The efficient use of the Li-ion battery requires the supervision of a Battery Management System (BMS), as it is necessary that the battery operates under appropriate conditions of temperature and State-of-Charge (SOC). The BMS has to estimate, in real-time, the amount of energy stored in a given system, such as a battery pack of a Electric Vehicle (EV). Such a task is not trivial, since it is not possible to directly measure the amount of energy stored in the system. In fact, the estimation of SOC is probably one of the biggest challenges in the field of battery research (19).

For instance, SOC can be measured using the Coulomb counting method given by (6)

$$\mathrm{SOC} = \mathrm{SOC}_0 - \frac{\int_0^t I_{\mathrm{bat}}\, dt}{Q_n} \tag{C.1}$$

where SOC_0 is the initial value of SOC, I_{bat} is the battery current and Q_n is the nominal capacity in Ah. It should be noted that the cell temperature produces deleterious effects on the open circuit voltage, internal resistance and available capacity and can also lead to a rapid degradation of the battery if it operates above a given temperature threshold. Therefore, the modeling of the battery is of paramount importance, since it will be used by the BMS to manage the operation of the battery (9).

The recent literature suggests that the machine learning approach, based on deep learning algorithms is the state of the art in the area of SOC estimation (14), (104).

One of the great challenges in deep learning is the optimization of the neural network. Although the Stochastic Gradient Descent (SGD) algorithm (and its variants) is very popular, a learning rate too small leads to painfully slow convergence, while a learning rate too high can destabilize the algorithm, causing oscillations or divergence (43).

On the other hand, we have at our disposal algorithms with adaptive learning rates such as AdaGrad, RMSProp, and Adam[1].

This work uses Deep Forward Networks (DFN) or MultiLayer Perceptrons (MLP) as baseline models. We investigate the effect of choosing the optimization algorithm on the performance of the deep learning models, especially with regard to the SOC estimate of a lithium-ion cell. The questions we ask are: a) which optimization algorithm should we choose? b) Which DFN offers the optimal capacity[2]?

We model a vehicle drive cycle designed for training neural networks which were applied to a Panasonic 18650PF Li-ion cell (20). More specifically, we compare the performance of the following optimization algorithms for training deep models: SGD, AdaGrad, RMSProp, and Adam (these last three use methods that adapt the learning rate parameter of the training algorithm).

The issue of regularization (parameter norm penalties, early stopping, dropout, etc.) is outside the scope of this paper.

The remainder of the paper is organized as follows. Section C.3 presents the state of the art and trends in Li-ion battery SOC estimation. Section C.4 presents our experimental results. Finally, section C.5 presents our conclusions.

C.3 State of the Art and Trends in Li-ion Battery Estimation

Energy storage acts as a mediator between variable loads and variable sources.

Hannan et al. (5) present a detailed taxonomy of the types of energy storage systems taking into account the form of energy storage and construction materials: mechanical, electrochemical (rechargeable and flow batteries), chemical, electrical (ultracapacitor or superconducting magnetic coil), thermal and hybrid. Li-ion battery technology has attracted the attention of industry and academia for the past decade. This is mainly due to the fact that Li-ion batteries offer more energy, higher power density, higher efficiency and lower self-discharge rate than other battery technologies such as NiCd, NiMH, etc.

There are two methods of battery modeling: i) model-driven and ii) data-driven (based on data that is collected from the device) (11).

Electrothermal models, which belong to the category of model-driven methods, are commonly classified as: i) electrochemical or ii) based on Equivalent Circuit Models (ECM).

Electrochemical models are based on partial differential equations and are able to represent thermal effects more accurately than ECM. However, the first class of models

[1] Chapter 8 of reference (43) presents a brief, but very instructive, review of such algorithms

[2] In deep learning, the number of learnable parameters in a model is often referred to as the model's capacity (determined by the number of layers and the number of units per layer) (43).

requires detailed knowledge of proprietary parameters of the battery manufacturer: cell area, electrode porosity, material density, electrolyte characteristics, thermal conductivity, etc. This difficulty can be eliminated by characterizing the battery using a thermal camera and thermocouples. But this solution is expensive, time consuming and introduces other challenges such as the implementation of dry air purge systems, ventilation, security, air and water supply, etc. Electrochemical models demand the use of intensive computing systems.

On the other hand, the ECM-based approach has been used for computational/numerical analysis of batteries. In this case, the objective is to develop an electrical model that represents the electrochemical phenomenon existing in the cell. The level of complexity of the model is the result of a compromise between precision and computational effort. Note that an extremely complex and accurate ECM may be unsuitable for application in embedded systems.

Estimating the SOC of lithium ion cells in a BMS by means of deep learning offers at least two significant advantages over model driven approaches, namely: i) neural networks are able to estimate the non linear functional dependence that exists between voltage, current and temperature (observable quantities) and unobservable quantities, such as SOC, with great precision and ii) the problem of identifying ECM parameters is avoided (14).

C.4 Experimental Results

We selected the Panasonic 18650PF Li-ion Battery Data of (20). This dataset contains a series of ten drive cycles: Cycle 1, Cycle 2, Cycle 3, Cycle 4, US06, HWFTa, HWFTb, UDDS, LA92, and Neural Network (NN). Cycles 1-4 consist of a random mix of US06, HWFET, UDDS, LA92, and Neural Network drive cycles. The drive cycle power profile is calculated from measurement for an electric Ford F150 truck with a 35kWh battery pack scaled for a single 18650PF cell.

We consider only the tests at the temperature of 25^o C for the NN drive cycle, which combines portions of US06 and LA92 drive cycles, and was designed to have some additional dynamics which are useful for training neural networks. Note that the objective of this paper is not to assess the performance of the DFN models at different temperatures, but to empirically investigate the question of stochastic optimization in the context of the problem of SOC estimation with deep neural nets.

For instance, Fig. 29 shows the following 2.9 Ah Panasonic 18650PF Li-ion cell characteristic curves:

- temperature (^{o} C) vs SOC (%);

- amp-hours discharged vs time (minutes);
- voltage (V) vs time (minutes);
- current (A) vs time (minutes);
- temperature (^{o} C) vs time (minutes); and
- voltage (V) vs SOC (%).

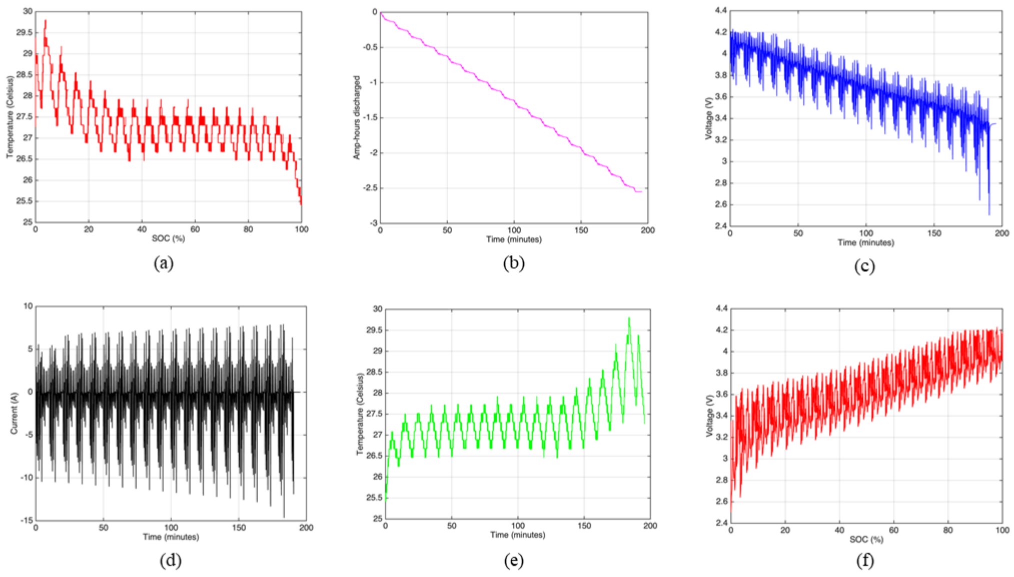


Figura 29 – in (a), (b), (c), (d), (e), and (f) we have: temperature (^{o}C) vs. SOC (%), amp-hours discharged vs. time (minutes), voltage (V) vs. time (minutes), current (A) vs. time (minutes), temperature (o C) vs. time (minutes), and voltage (V) vs. SOC (%), respectively.

We applied feature normalization on the input data using the formula

$$\boldsymbol{x}_{\text{normalized}} = \frac{\boldsymbol{x} - \mu_{\boldsymbol{x}}}{\sigma_{\boldsymbol{x}}} \tag{C.2}$$

where $\mu_{\boldsymbol{x}}$ and $\sigma_{\boldsymbol{x}}$ denote the mean and standard deviation of $\boldsymbol{x}$.

We divided the NN dataset into training and validations sets. The validation error is estimated by taking the average validation error across $K = 4$ trials. We use a simple, but popular solution, called K-fold cross-validaton, which consists of splitting the available training data into two partitions (training and validation), instantiating K identical models, for each fold $k \in \{1, 2, \ldots, K\}$, and training each one on the training partitions, while evaluating on the validation partition. The validation score for the model used is then the average of the K validation scores obtained. It is usual to use about 80% of the data for the training set, and 20% for the validation set. Note that the validation scores may have a high variance with regard to the validation split. Therefore, K-fold cross-validaton help us improve the reliability when evaluating the generalization power of the model (43).

We use Mean Absolute Error (MAE) and Mean Squared Error (MSE) as performance metrics for the generalization (test) errors. After the validation phase using K-fold cross-validaton, the DFN model is trained using the entire training data and its performance is evaluated against an unseen test set. This is the test phase (final phase).

The features of the input layer are: voltage, current, and temperature. We vary the depth of the models, using DFN with two to five hidden layers (HL), in order to assess the effect of depth on generalization power, each hidden layer having 256 units. We used the REctified Linear Unit (Relu) activation function, given by

$$g(z) = \max\{0, z\}. \tag{C.3}$$

Tables 2 and 3 show the MAE and MSE, respectively, for the SOC estimates obtained with SGD, AdaGrad, RMSProp, and Adam optimizers. Fig. 2 shows the learning curves (MAE and loss function) of the training phase for the DFN with four hidden layers using the Adam algorithm.

	2 HL	**3 HL**	**4 HL**	**5 HL**
Optimizer	**MAE**	**MAE**	**MAE**	**MAE**
SGD	0.32	0.45	0.32	0.50
AdaGrad	0.84	0.63	0.53	0.49
RMSProp	0.41	0.67	0.56	0.32
Adam	0.46	0.26	0.21	0.22

Tabela 2 – MAE vs. optimization algorithms.

	2 HL	**3 HL**	**4 HL**	**5 HL**
Optimizer	**MAE**	**MAE**	**MAE**	**MAE**
SGD	0.25	0.42	0.25	0.51
AdaGrad	1.49	0.85	0.60	0.52
RMSProp	0.37	0.75	0.56	0.29
Adam	0.41	0.14	0.11	0.13

Tabela 3 – MSE vs. optimization algorithms.

The results of Tables 2 and 3 show that:

- the choice of the optimization algorithm must be made on a case-by-case basis. Although the family of algorithms with adaptive learning rates (AdaGrad, RMSProp, and Adam in this paper) has become very popular in the deep learning community nowadays, one can not claim, a priori, that a given optimization algorithm is the best algorithm. For instance, the results obtained for the DFN with four hidden layers and the SGD algorihtm are better than those obtained for AdaGrad and RMSProp;

- the choice of the optimization algorithm affects the model performance;
- the DFN with four hidden layers is able to estimate the SOC with a MAE of 0.21% and MSE of 0.11% (best results) using Adam, being, therefore, the model of optimal capacity when considering 256 units per layer.

It should be noted that we do not perform ensemble-average, as we are dealing with a real dataset, i. e., we just have access to one realization of the data-generating process.

C.5 Conclusion

We present a preliminary empirical study on the impact of the optimization algorithm on the performance of the deep learning model, in the context of the estimation of the SOC of a Li-ion battery. For this, we used the Panasonic 18650PF Li-ion Battery Data (20).

Our results indicate (for the 18650PF Li-ion cell) that: i) the choice of the optimization algorithm must be made on a case-by-case basis, ii) the choice of the optimization algorithm affects the model performance, iii) the DFN with four hidden layers (256 units per layer) is able to estimate the SOC with with a MAE of 0.21% and MSE of 0.11% using the Adam optimizer, being, therefore, the model of optimal capacity.

APÊNDICE D – Data-driven state-of-charge estimation of the Panasonic 18650PF Li-ion cell using deep forward neural networks

Authors: Alexandre B. de Lima, Maurício B. C. Salles, and José Roberto Cardoso. Accepted for presentation at the 14th IEEE International Conference on Industry Applications (INDUSCON) that will take place in São Paulo, Brazil during August 16-18, 2021.

D.1 Abstract

The State-of-Charge (SOC) is a key parameter for the proper functioning of the Battery Management System (BMS) of lithium-ion (Li-ion) batteries, and indicates the amount of charge remaining in the battery. In this work, we present a novel empirical study for the data-driven estimation of the SOC of the Panasonic 18650PF Li-ion cell using Deep Forward Neural Networks (DFNN) and optimization algorithms with adaptive learning rates. Specifically, we model the Urban Dynamometer Driving Schedule (UDDS) drive cycle. Our results suggest that the choice of the optimization algorithm affects the performance of the model and that a DFNN with five hidden layers is the model of optimal capacity when considering 256 units per layer. This optimal DFNN is able to estimate the SOC of the 18650PF Li-ion cell with an error smaller than 0.12 % over a 25^o C dataset using the Adamax optimization algorithm.

Keywords: Deep Learning, Electrical Energy Storage, Li-ion battery, State-of-Charge.

D.2 Introduction

Electrical Energy Storage (EES) and Eletric Vehicle (EV) have attracted a lot of attention recently due to environmental issues such as global warming, air pollution, and fossil fuel depletion (105).

EES technology consists of the process of converting a form of energy (almost always electrical) to a form of storable energy, which can be converted into electrical energy when necessary. EES has the following functions: to assist in meeting the maximum electrical load demands, to provide time-varying energy management, to relieve the intermittency of renewable energy generation, to improve energy quality/reliability, to serve remote loads and vehicles, to support the realization of smart grids, improve the management of distributed/standby power generation and reduce the import of electricity during peak demand periods (7).

The advancement of EES technologies enabled the emergence of the iPod, smartphones and tablets with lithium-ion (Li-ion) batteries. Also, EES will be one of the critical components of the new electricity grid, given the variable nature of renewable energy sources (6), (7). EES systems are necessary even when renewable sources are connected to the grid, because it is necessary to smooth the energy supply. For example, the EES of a building or factory can be charged during hours of reduced demand and supply/supplement energy demand during peak hours.

In the last decade, government, industry and academia have given great importance to the electrification of the transport system, motivated by the need to reduce the emission of greenhouse gases (18), considering that renewable energy is a key issue to decrease fossil fuel usage and to the shift to low carbon energy systems (105). Hybrid electric vehicles, such as the Toyota Prius, or fully electric vehicles, such as the various Tesla models, the Nissan Leaf and the Chevy Bolt, are successful cases in the USA (6). Recently, the Prime Minister of the United Kingdom confirmed that Her Majesty's government is anticipating a ban on the sale of new gasoline and diesel-powered cars and vans from 2040 until 2030 (106).

The efficient use of a Li-ion battery requires the supervision of a Battery Management System (BMS), as it is necessary that the battery operates under appropriate conditions of temperature and State-of-Charge (SOC). The BMS has to estimate, in real-time, the amount of energy stored in a given system, such as a battery pack of an EV. Such a task is not trivial, since it is not possible to directly measure the amount of energy stored in the system. In fact, the estimation of SOC is probably one of the biggest challenges in the field of battery research (19).

In mathematical terms, SOC can be defined as (104)

$$\mathrm{SOC} = \frac{Q_{\mathrm{availabe}}}{Q_n} \times 100\% \tag{D.1}$$

where Q_{availabe} is the remaining battery charge and Q_n is the nominal capacity (both in Ah).

SOC is an inner state of a battery and can be empirically determined using the Coulomb counting method given by (19)

$$\mathrm{SOC} = \mathrm{SOC}_0 - \frac{\int_0^t I_{\mathrm{bat}\,dt}}{Q_n} \times 100\% \tag{D.2}$$

where SOC_0 is the initial value of SOC, I_{bat} is the battery current and Q_n is the nominal capacity. SOC depends on battery cell temperature, material degradation, electrochemical reactions, and aging cycles (18). It should be noted that the cell temperature produces deleterious effects on the open circuit voltage, internal resistance and available capacity and can also lead to a rapid degradation of the battery if it operates above a given temperature threshold. Accurate SOC estimation of a battery is critical to controlling charging, discharging and extending the battery's life cycle.

Therefore, the modeling of the battery is of paramount importance, since it will be used by the BMS to manage the operation of the battery.

The idea of applying an augmented version of the traditional machine learning algorithm known as Artificial Neural Network (ANN) to the estimation of the SOC of Li-ion batteries is fairly new. This strategy is usually called Deep Learning (DL). DL has been breaking records in domains such as computer vision, speech recognition, and Natural Language Processing (NLP) since the early 2010s (107), (108), (109).

One of the challenges in DL is the optimization of the neural network. Although the Stochastic Gradient Descent (SGD) algorithm (and its variants) is very popular, a learning rate too small leads to painfully slow convergence, while a learning rate too high can destabilize the algorithm, causing oscillations or divergence.

On the other hand, we have at our disposal algorithms with adaptive learning rates such as Adam, Adamax, AdaGrad, and RMSProp, for instance (110).

Overfitting is a central problem in machine learning and occurs when the gap between the *training* error and *generalization* (or *test*) error is too large. The processing of mitigating overfitting is called *regularization*, which can be defined as any modification we make to a learning algorithm with the goal of reducing its generalization error but not its training error.

We use Deep Forward Neural Networks (DFNN) or MultiLayer Perceptrons (MLP) as our baseline models (18). We investigate the effect of choosing the optimization algorithm on the performance of the DL models, especially with regard to the SOC estimate of a Li-ion cell. The questions we ask are: a) which optimization algorithm should we choose? b) Which DFNN offers the optimal capacity[1]?

As far as regularization issues are concerned, i. e., how we mitigate the test error of the algorithms under investigation, this work utilizes a very popular strategy in machine learning known as *early stopping* (43), (25).

The remainder of the work is organized as follows. Section D.3 presents the state of

[1] In DL, the number of learnable parameters in a model is often referred to as the model's capacity (determined by the number of layers and the number of units per layer) (43). Consider a plot of the error (training/test errors - vertical axis) versus capacity (horizontal axis). Tipically, at the left end of the graph, training error and test error are both high. As we increase capacity, training error decreases, but the gap between training and test error increases. Eventually, the size of this gap outweighs the decrease in training error, and we enter the *overfitting regime*, where capacity is too large, above the *optimal capacity*.

the art and trends in Li-ion battery SOC estimation. Section D.4 presents our experimental results. Finally, section D.5 presents our conclusions.

D.3 State of the Art in Li-ion Battery Estimation

Energy storage acts as a mediator between variable loads and variable sources. Electricity storage is not new. Volta invented the modern battery in 1799. Batteries were implemented in telegraph networks in 1836 (4). The Rocky River Hydroelectric Power Plant in New Milford, Connecticut, was the first major electrical energy storage (EES) system project in the United States. The plant used hydroelectric storage technology through Pumped Hydroelectric Storage (PHS).

In general, the literature records two well known methods for SOC estimation: i) model-based and ii) data-driven (based on data that is collected from the device) (18).

Electrothermal models, which belong to the category of model-driven methods, are commonly classified as: i) electrochemical or ii) based on Equivalent Circuit Models (ECM) (10), (9).

Electrochemical models are based on partial differential equations (12) and are able to represent thermal effects more accurately than ECM (13). However, the first class of models requires detailed knowledge of proprietary parameters of the battery manufacturer: cell area, electrode porosity, material density, electrolyte characteristics, thermal conductivity, etc. This difficulty can be eliminated by characterizing the battery using a thermal camera and thermocouples. But this solution is expensive, time consuming and introduces other challenges such as the implementation of dry air purge systems, ventilation, security, air and water supply, etc. Electrochemical models demand the use of intensive computing systems (10).

On the other hand, the ECM-based approach has been used for computational/numerical analysis of batteries (10). In this case, the objective is to develop an electrical model that represents the electrochemical phenomenon existing in the cell. The level of complexity of the model is the result of a compromise between precision and computational effort. Note that an extremely complex and accurate ECM may be unsuitable for application in embedded systems.

The recent literature suggests that the machine learning approach, based on DL algorithms is the state of the art in the area of SOC estimation of Li-ion batteries (16), (15).

Chemali et al (14) compared the performance of Deep Neural Networks (DNN) with those of other relevant algorithms that have been proposed since the second half of the 2000s. The article shows that the SOC estimation error obtained with DL is less than the following methods:

- Model Adaptive-Improved Extended Kalman Filter (EKF) (70);
- Adaptive EKF with Neural Networks (71);
- Adaptive Unscented Kalman Filter (AUKF) with Extreme Machine Learning (72);
- Fuzzy NN with Genetic Algorithm (73); and
- Radial Bias Function Neural Network (74).

According to Lipu et al (18),

> (...) The data-driven SOC estimation approaches require limited knowledge about the battery internal characteristics (...) in comparison to model-based approaches (...) Besides, the data-driven algorithms can operate without battery model, thus considerably time and human efforts can be avoided to develop complex mathematical rules and relationships in mapping the battery dynamics as well as determining battery model parameters.

Estimating the SOC of Li-ion cells in a BMS by means of DL offers at least two significant advantages over model driven approaches, namely: i) neural networks are able to estimate the non linear functional dependence that exists between voltage, current and temperature (observable quantities) and unobservable quantities, such as SOC, with great precision and ii) the problem of identifying ECM parameters is avoided.

D.4 Experimental Results

We selected the Panasonic 18650PF Li-ion Battery Data of (20). This dataset contains a series of ten drive cycles: Cycle 1, Cycle 2, Cycle 3, Cycle 4, US06, HWFTa, HWFTb, UDDS, LA92, and Neural Network (NN). Cycles 1-4 consist of a random mix of US06, HWFET, UDDS, LA92, and Neural Network drive cycles. The drive cycle power profile is calculated from measurement for an electric Ford F150 truck with a 35kWh battery pack scaled for a single 18650PF cell.

We consider only the tests at the temperature of 25^o C for the Urban Dynamometer Driving Schedule (UDDS) drive cycle, which is commonly called the "LA4" or "the city test" and represents city driving conditions (111). Note that the objective of this paper is not to assess the performance of the DFNN models at different temperatures, but to empirically investigate the problem of data-driven SOC estimation of the Panasonic 18650PF Li-ion battery using DFNN and algorithms with adaptive learning rates.

For instance, Fig. 30 shows the following 2.9 Ah Panasonic 18650PF Li-ion cell characteristic curves:

- temperature (^{o}C) vs. SOC (%);
- amp-hours discharged vs. time (minutes);
- voltage (V) vs. time (minutes);
- current (A) vs. time (minutes);
- temperature (^{o}C) vs. time (minutes), and;
- voltage (V) vs. SOC (%).

We applied feature normalization on the input data using the formula:

$$\boldsymbol{x}_{\text{normalized}} = \frac{\boldsymbol{x} - \mu_{\boldsymbol{x}}}{\sigma_{\boldsymbol{x}}} \tag{D.3}$$

where $\mu_{\boldsymbol{x}}$ and $\sigma_{\boldsymbol{x}}$ denote the mean and standard deviation of $\boldsymbol{x}$.

Let

$$\{(x_1, \text{SOC}_1), (x_2, \text{SOC}_2), \ldots, (x_i, \text{SOC}_i), \ldots, (x_m, \text{SOC}_m)\}$$

be a training set with m examples, where (x_i, SOC_i) and SOC_i denote the i-th training example and the i-th SOC target variable (output of DFNN), respectively. We use Mean Absolute Error (MAE), given by,

$$\text{MAE} = \frac{1}{m}\sum_i (|\text{SOC}_i^* - \text{SOC}_i|) \tag{D.4}$$

where SOC_i^* gives the prediction of the model (DFNN) for input x_i, and Mean Squared Error (MSE) defined as

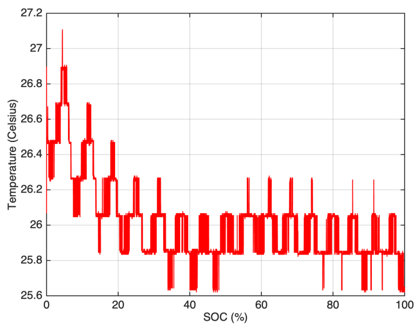
$$\text{MSE} = \frac{1}{m}\sum_i (\text{SOC}_i^* - \text{SOC}_i)^2 \tag{D.5}$$

as performance metrics for the generalization errors.

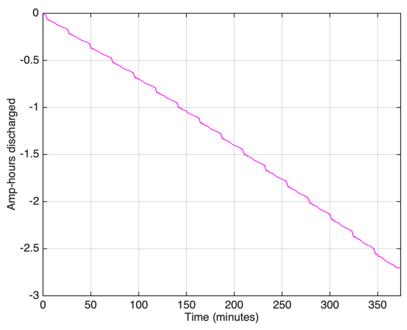
In order to prevent overfitting, the backpropagation algorithm (95) is used in conjunction with the early stopping training scheme, which stops training when the monitored metric has stopped improving (112).

The features of the input layer are: voltage, current, and temperature. We vary the depth of the models, using DFNN with two to six hidden layers, in order to assess the effect of depth on generalization power, each hidden layer having 256 units. We used the REctified Linear Unit (Relu) activation function (Fig. 31), given by

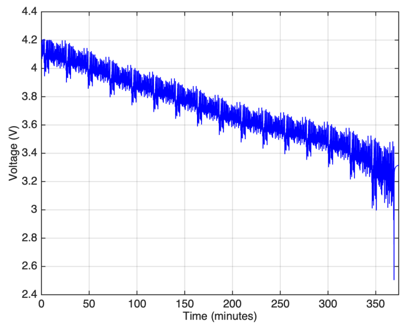
$$g(z) = \max\{0, z\}. \tag{D.6}$$



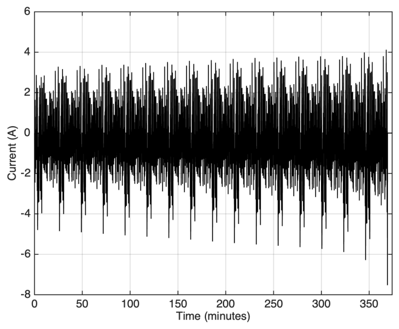
(a)



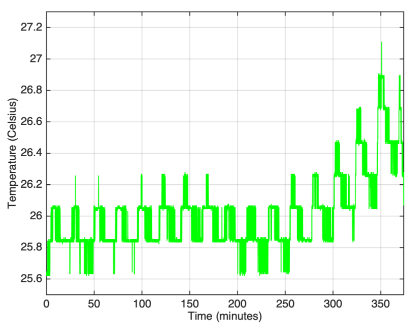
(b)



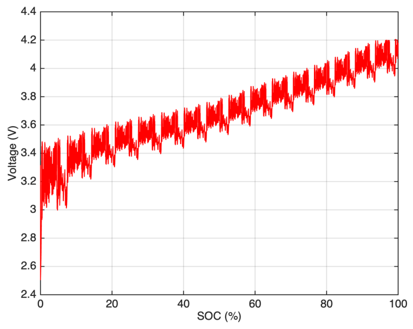
(c)



(d)



(e)



(f)

Figura 30 – in (a), (b), (c), (d), (e), and (f) we have (for the UDDS drive Cycle): temperature (^{o} C) vs SOC (%), amp-hours discharged vs time (minutes), voltage (V) vs time (minutes), current (A) vs time (minutes), temperature (^{o} C) vs time (minutes), and voltage (V) vs SOC (%), respectively.

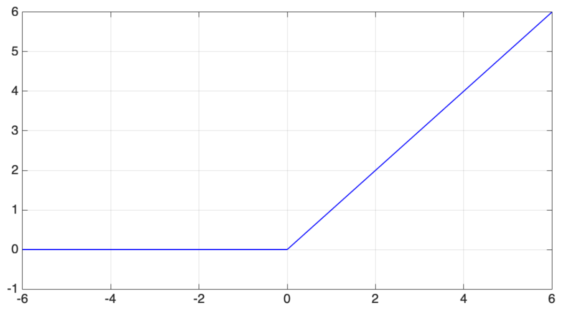


Figura 31 – Relu activation funcion.

Tables 4 and 5 show the MAE and MSE, respectively, for the SOC estimates obtained using the Nadam, Adam, Adamax, SGD, AdaGrad, RMSProp, FTLR[2], and Adadelta optimizers (all the eight optimizers that can be instantiated using the Tensorflow 2.x keras API)(103)[3].

Fig. 32 shows the learning curves (MAE and loss function) of the training phase for the DFNN with five hidden layers (HL) using the RMSProp algorithm.

	2 HL	**3 HL**	**4 HL**	**5 HL**	**6 HL**
Optimizer	**MAE**	**MAE**	**MAE**	**MAE**	**MAE**
Nadam	0.65	0.41	0.26	0.29	0.34
Adam	0.31	0.35	0.37	0.35	0.29
Adamax	0.37	0.31	0.34	0.24	0.30
SGD	0.45	0.42	0.40	0.32	0.41
AdaGrad	0.57	0.47	0.43	0.37	0.37
RMSProp	0.66	0.49	0.25	0.31	0.25
FTLR	0.55	0.52	0.47	0.45	0.46
Adadelta	0.57	0.44	0.40	0.37	0.35

Tabela 4 – MAE vs. optimization algorithms.

The results of Tables 4 and 5 show that:

- the choice of the optimization algorithm must be made on a case-by-case basis. Although algorithms with adaptive learning rates such as Nadam, Adam, Adamax, AdaGrad, RMSProp, FTLR, and Adadelta, have become very popular in the DL community nowadays, one can not claim, a priori, that a given optimization algorithm is the best algorithm. For instance, the results obtained for the DFNN with five HL and the SGD algorihtm are better than those obtained for Adam and AdaGrad;
- the choice of the optimization algorithm affects the model performance;

[2] FTLR denotes "Follow The (Proximally) Regularized Leader".
[3] We utilized Tensorflow with the Keras DL framework in the experimental setup of this work.

	2 HL	**3 HL**	**4 HL**	**5 HL**	**6 HL**
Optimizer	**MSE**	**MSE**	**MSE**	**MSE**	**MSE**
Nadam	0.60	0.41	0.18	0.14	0.21
Adam	0.19	0.23	0.25	0.23	0.16
Adamax	0.23	0.18	0.20	0.11	0.18
SGD	0.35	0.30	0.29	0.20	0.29
AdaGrad	0.58	0.38	0.31	0.25	0.25
RMSProp	0.69	0.46	0.14	0.19	0.15
FTLR	0.58	0.45	0.38	0.35	0.37
Adadelta	0.57	0.33	0.28	0.25	0.22

Tabela 5 – MSE vs. optimization algorithms.

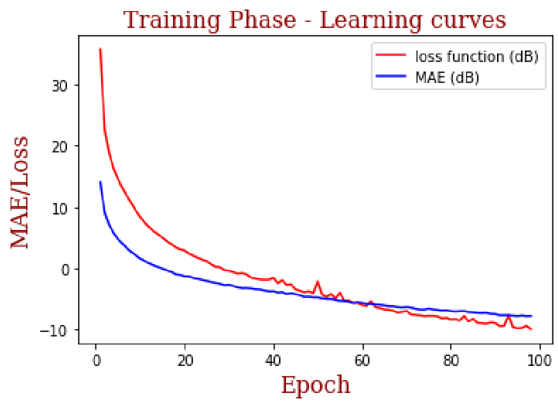


Figura 32 – Learning curves of the training phase.

- the DFNN with five HL is able to estimate the SOC with a MAE of 0.24% and MSE of 0.11% (best results) using Adamax, being, therefore, the model of optimal capacity when considering 256 units per layer.

It should be noted that we do not perform ensemble-average, as we are dealing with a real dataset, i. e., we just have access to one realization of the data-generating process.

D.5 Conclusion

We present a preliminary empirical study on the impact of the optimization algorithm on the performance of the DFNN model, in the context of the estimation of the SOC of a Li-ion battery. For this, we used the dataset (20).

Our results indicate, for the 18650PF Li-ion cell (UDDS drive cycle 25^o C) that:

- the choice of the optimization algorithm must be made on a case-by-case basis;

- the choice of the optimization algorithm affects the model performance;
- the DFNN with five hidden layers (256 units per layer) is able to estimate the SOC with with a MAE of 0.24% and MSE of 0.11% using the Adamax optimizer, being, therefore, the model of optimal capacity.

We present a preliminary empirical study on the impact of the optimization algorithm on the performance of the deep learning model, in the context of the estimation of the SOC of a Li-ion battery. For this, we used the Panasonic 18650PF Li-ion Battery Data (20).

Our results indicate (for the 18650PF Li-ion cell) that: i) the choice of the optimization algorithm must be made on a case-by-case basis, ii) the choice of the optimization algorithm affects the model performance, iii) the DFN with four hidden layers (256 units per layer) is able to estimate the SOC with with a MAE of 0.21% and MSE of 0.11% using the Adam optimizer, being, therefore, the model of optimal capacity.

APÊNDICE E – The relevance of the optimization algorithm on the data-driven estimation of the state-of-charge of the Panasonic 18650PF lithium-ion cell using deep feedforward neural networks

Authors: Alexandre B. de Lima, Maurício B. C. Salles, and José Roberto Cardoso. Submitted to the Future Technologies Conference (FTC) to be held from 28-29 October 2021, Vancouver, Canada.

E.1 Abstract

This article presents a novel empirical study for the estimation of the State of Charge (SOC) of a lithium-ion (Li-ion) battery which uses a Deep Feedforward Neural Network (DFNN) with three hidden layers. We model a series of ten vehicle drive cycles that were applied to the Panasonic 18650PF Li-ion cell. Our results show that the choice of the optimization algorithm affects the model performance. The proposed model was able to achieve an error smaller than 1.0% over a dataset of 25^o C in all drive cycles.

Keywords: Deep Learning, Electrical Energy Storage, Li-ion battery, State-of-Charge.

E.2 Introduction

In the last decade, government, industry and academia have given great importance to the electrification of the transport system, motivated by the need to reduce the emission of greenhouse gases. Hybrid electric vehicles, such as the Toyota Prius, or fully Electric Vehicles (EV), such as the various Tesla models, the Nissan Leaf and the Chevy Bolt, are successful cases in the USA (6). Recently, the Prime Minister of the United Kingdom confirmed that Her Majesty's government is anticipating a ban on the sale of new gasoline and diesel-powered cars and vans from 2040 until 2030 (106).

The advancement of Electrical Energy Storage (EES) technologies enabled the emergence of the iPod, smartphones and tablets with lithium-ion (li-ion) batteries. Also, EES will be one of the critical components of the new electricity grid, given the intermittent nature of renewable energy sources (6), (7). EES systems are necessary even when renewable sources are connected to the grid, because it is necessary to smooth the energy supply. For example, the EES of a building or factory can be charged during hours of reduced demand and supply/supplement energy demand during peak hours.

EES technology consists of the process of converting a form of energy (almost always electrical) to a form of storable energy, which can be converted into electrical energy when necessary. EES has the following functions: to assist in meeting the maximum electrical load demands, to provide time-varying energy management, to relieve the intermittency of renewable energy generation, to improve energy quality/reliability, to serve remote loads and vehicles, to support the realization of smart grids, improve the management of distributed/standby power generation and reduce the import of electricity during peak demand periods (7), (8).

The efficient use of the Li-ion battery requires the supervision of a Battery Management System (BMS), as it is necessary that the battery operates under appropriate conditions of temperature and charge (State-Of-Charge (SOC)) (10). The BMS has to estimate, in real-time, the amount of energy stored in a given system, such as in a battery pack of an EV. This task is not trivial, since it is not possible to directly measure the amount of energy stored in the system. In fact, the estimation of SOC is probably one of the biggest challenges in the field of battery research (19).

In mathematical terms, SOC can be defined as (104)

$$\mathrm{SOC} = \frac{Q_{\text{availabe}}}{Q_n} \times 100\% \tag{E.1}$$

where Q_{availabe} is the remaining battery charge and Q_n is the nominal capacity (both in Ah).

SOC is an inner state of a battery and can be empirically determined using the Coulomb counting method given by (19)

$$\mathrm{SOC} = \mathrm{SOC}_0 - \frac{\int_0^t I_{\text{bat}}\, dt}{Q_n} \times 100\% \tag{E.2}$$

where SOC_0 is the initial value of SOC, I_{bat} is the battery current and Q_n is the nominal capacity. SOC depends on battery cell temperature, material degradation, electrochemical reactions, and aging cycles (18). It should be noted that the cell temperature produces deleterious effects on the open circuit voltage, internal resistance and available capacity and can also lead to a rapid degradation of the battery if it operates above a given temperature threshold. Accurate SOC estimation of a battery is critical to controlling charging, discharging and extending the battery's life cycle.

Therefore, the modeling of the battery is of paramount importance, since it will be used by the BMS to manage the operation of the battery.

The idea of applying an augmented version of the traditional machine learning algorithm known as Artificial Neural Network (ANN) to the estimation of the SOC of Li-ion batteries is fairly new. This strategy is usually called Deep Learning (DL). DL has been breaking records in domains such as computer vision, speech recognition, and Natural Language Processing (NLP) since the early 2010s (107), (108), (109).

One of the great challenges in DL is the optimization of the neural network. Although the Stochastic Gradient Descent (SGD) algorithm (and its variants) is very popular, a learning rate too small leads to painfully slow convergence, while a learning rate too high can destabilize the algorithm, causing oscillations or divergence (43).

On the other hand, we have at our disposal algorithms with adaptive learning rates such as RMSProp (100) and Adamax (101).

We use Deep Feedforward Neural Networks (DFNN) or MultiLayer Perceptrons (MLP) as our baseline models (18). We investigate the effect of choosing the optimization algorithm on the performance of the DFNN model, especially with regard to the estimation of the SOC of a Li-ion cell. The question we ask is: which optimization algorithm should we choose?

The issue of regularization for DL is outside the scope of this work.

The remainder of the paper is organized as follows. Section E.3 presents the state of the art and trends in Li-ion battery parameter estimation. Section E.4 presents our experimental results. Finally, section E.5 presents our conclusions.

E.3 State of the Art and Trends in Li-ion Battery Estimation

Energy storage acts as a mediator between variable loads and variable sources.

Hannan et al. (5) present a detailed taxonomy of the types of energy storage systems taking into account the form of energy storage and construction materials: mechanical, electrochemical (rechargeable and flow batteries), chemical, electrical (ultracapacitor or superconducting magnetic coil), thermal and hybrid. Li-ion battery technology has attracted the attention of industry and academia for the past decade. This is mainly due to the fact that Li-ion batteries offer more energy, higher power density, higher efficiency and lower self-discharge rate than other battery technologies such as NiCd, NiMH, etc. (9).

There are two methods of battery modeling: i) model-driven and ii) data-driven (based on data that is collected from the device) (11).

Electrothermal models, which belong to the category of model-driven methods, are commonly classified as: i) electrochemical or ii) based on Equivalent Circuit Models (ECM) (10), (9).

Electrochemical models are based on partial differential equations (12) and are able to represent thermal effects more accurately than ECM (13). However, the first class of models requires detailed knowledge of proprietary parameters of the battery manufacturer: cell area, electrode porosity, material density, electrolyte characteristics, thermal conductivity, etc. This difficulty can be eliminated by characterizing the battery using a thermal camera and thermocouples. But this solution is expensive, time consuming and introduces other challenges such as the implementation of dry air purge systems, ventilation, security, air and water supply, etc. Electrochemical models demand the use of intensive computing systems (10).

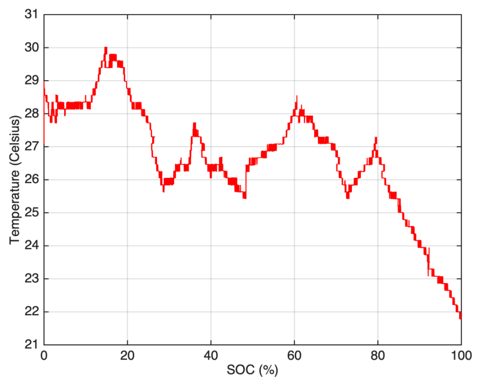
On the other hand, the ECM-based approach has been used for computational/numerical analysis of batteries (10). In this case, the objective is to develop an electrical model that represents the electrochemical phenomenon existing in the cell. The level of complexity of the model is the result of a compromise between precision and computational effort. Note that an extremely complex and accurate ECM may be unsuitable for application in embedded systems.

Chemali et al (14) compared the performance of DFNN with those of other relevant algorithms that have been proposed since the second half of the 2000s. The article shows that the SOC estimation error obtained with deeep learning is less than the following methods:

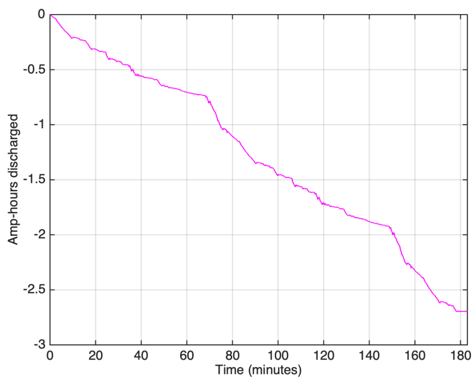
- Model Adaptive-Improved Extended Kalman Filter (EKF) (70);
- Adaptive EKF with Neural Networks (71);
- Adaptive Unscented Kalman Filter (AUKF) with Extreme Machine Learning (72);
- Fuzzy NN with Genetic Algorithm (73); and
- Radial Bias Function Neural Network (74).

In fact, the recent literature indicates that the machine learning approach, based on DL algorithms is the state of the art in the area of SOC estimation of Li-ion batteries (16), (15).

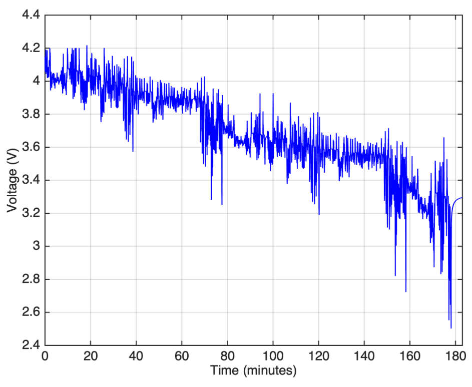
Estimating the SOC of lithium ion cells in a BMS by means of deep learning offers at least two significant advantages over model driven approaches, namely: i) neural networks are able to estimate the non linear functional dependence that exists between voltage, current and temperature (observable quantities) and unobservable quantities, such as SOC, with great precision and ii) the problem of identifying ECM parameters is avoided.



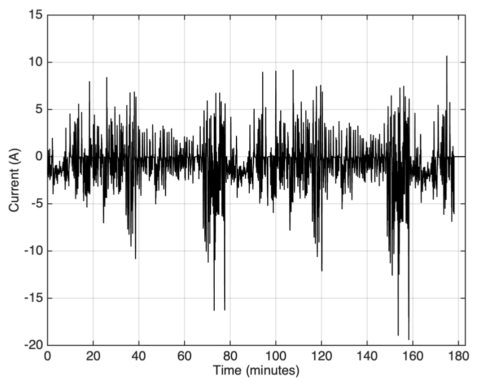
(a)



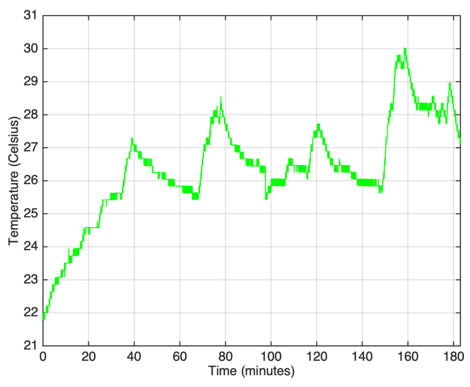
(b)



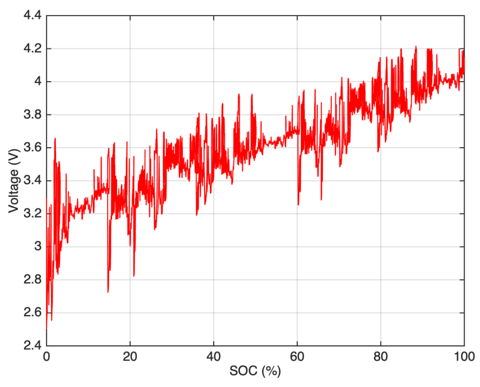
(c)



(d)



(e)



(f)

Figura 33 – in (a), (b), (c), (d), (e), and (f) we have (for the drive Cycle 1): temperature (^{o} C) vs SOC (%), amp-hours discharged vs time (minutes), voltage (V) vs time (minutes), current (A) vs time (minutes), temperature (^{o} C) vs time (minutes), and voltage (V) vs SOC (%), respectively.

E.4 Experimental Results

We selected the Panasonic 18650PF Li-ion Battery Data of (20). This dataset contains a series of ten drive cycles: Cycle 1, Cycle 2, Cycle 3, Cycle 4, US06, HWFTa, HWFTb, UDDS, LA92, and Neural Network (NN). Cycles 1-4 consist of a random mix of US06, HWFET, UDDS, LA92, and Neural Network drive cycles. The drive cycle power profile is calculated from measurement for an electric Ford F150 truck with a 35kWh battery pack scaled for a single 18650PF cell.

We consider only the tests at the temperature of 25^o C, as the objective of this paper is not to assess the performance of the DFNN model at different temperatures, but to empirically investigate the question of stochastic optimization in the context of the problem of SOC estimation with deep neural nets.

For instance, Fig. 33 shows the following 2.9 Ah Panasonic 18650PF Li-ion cell characteristic curves for Cycle 1.

- temperature (^{o} C) vs SOC (%);
- amp-hours discharged vs time (minutes);
- voltage (V) vs time (minutes);
- current (A) vs time (minutes);
- temperature (^{o} C) vs time (minutes); and
- voltage (V) vs SOC (%).

We applied feature normalization on the input data using the formula

$$\boldsymbol{x}_{\text{normalized}} = \frac{\boldsymbol{x} - \mu_{\boldsymbol{x}}}{\sigma_{\boldsymbol{x}}} \tag{E.3}$$

where $\mu_{\boldsymbol{x}}$ and $\sigma_{\boldsymbol{x}}$ denote the mean and standard deviation of $\boldsymbol{x}$.

Let

$$\{(x_1, \text{SOC}_1), (x_2, \text{SOC}_2), \ldots, (x_i, \text{SOC}_i), \ldots, (x_m, \text{SOC}_m)\}$$

be a training set with m examples, where (x_i, SOC_i) and SOC_i denote the i-th training example and the i-th SOC target variable (output of DFNN), respectively. We use Mean Absolute Error (MAE), given by,

$$\text{MAE} = \frac{1}{m} \sum_i (|\text{SOC}_i^* - \text{SOC}_i|) \tag{E.4}$$

where SOC_i^* gives the prediction of the model (DFNN) for input x_i, and Mean Squared Error (MSE) defined as

$$\text{MSE} = \frac{1}{m}\sum_i (\text{SOC}_i^* - \text{SOC}_i)^2 \tag{E.5}$$

as performance metrics for the generalization errors.

As far as cross-validation for model validation is concerned, we divided the datasets into training, validation[1], and test sets[2]. The validation error is estimated by taking the average validation error across $K = 4$ trials. We use a simple, but popular solution, called K-fold cross-validaton (Fig. 34), which consists of instantiating K identical models, for each fold $k \in \{1, 2, \ldots, K\}$, and training each one on the training partitions, while evaluating on the validation partition. The validation score for the model used is then the average of the K validation scores obtained. This procedure allows network hyperparameters to be adjusted so that overfitting is mitigated (43). Note that the validation scores may have a high variance with regard to the validation split. Therefore, K-fold cross-validaton help us improve the reliability when evaluating the generalization power of the model.

After the validation phase using K-fold cross-validaton, the DFNN model is trained using the entire training data (excluding the test set) and its performance is evaluated against an unseen test set. This the test phase (final phase).

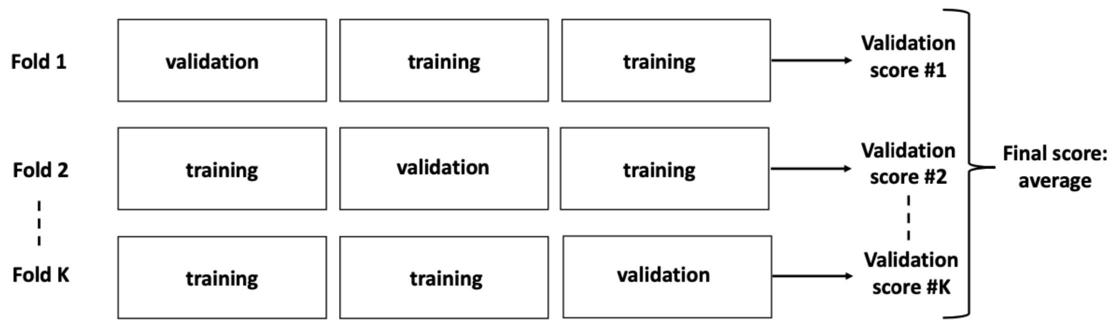


Figura 34 – K-fold cross-validation.

Our simulations confirm the design matrix chosen by (14), as our best results were found with the following features (inputs): $x_1 = V(t)$ (voltage in V), $x_2 = \bar{V}(t)$ (average voltage in V), $x_3 = \bar{I}(t)$ (average current in A), and $x_4 = \bar{T}(t)$ (average temperature in ^{o}C), where t denotes time in seconds. We used moving averages over 400 past samples in order to smooth the last three features. This procedure improved the estimation of the SOC. The DFNN has three hidden layers (HL), each HL having 256 units (see Fig.36).

[1] Also known as cross-validation set.

[2] For instance, a usual way to break down a dataset is: i) training set (60%), validation set (20%), and test set (20%).

We used the REctified Linear Unit (Relu) activation function, given by

$$g(z) = \max\{0, z\}. \tag{E.6}$$

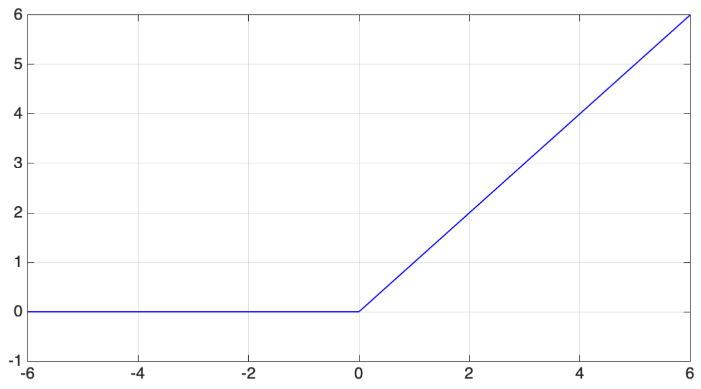


Figura 35 – Relu activation funcion.

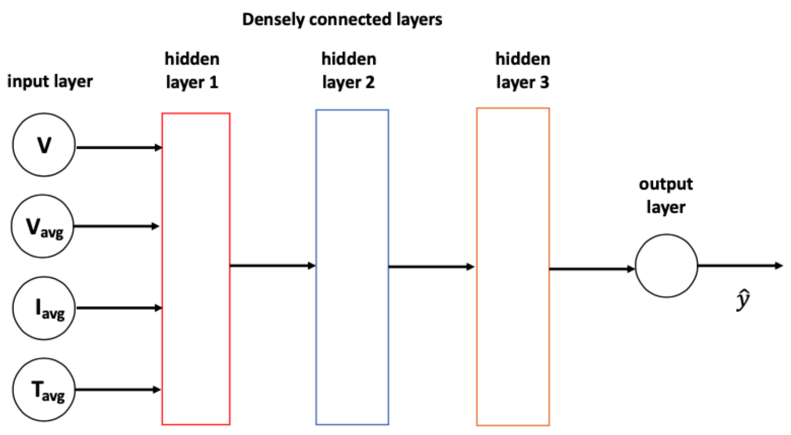


Figura 36 – DFNN with three (3) hidden layers. The output is denoted by $\hat{y}$.

Table 6 shows the SOC estimates obtained with SGD, RMSProp and Adamax. In each row, the best result is indicated in blue, whereas the worst result is in red.

The results of Table 6 show that:

- the choice of the optimization algorithm must be made on a case-by-case basis. Although the family of algorithms with adaptive learning rates (RMSProp, Adamax, Adam, etc.) has become very popular in the DL community nowadays, one can not claim, a priori, that a given optimization algorithm is the best algorithm. For instance, the results obtained for the Cycle 4 and LA92 drive cycles with the SGD algorithm are better than those obtained wirt RMSProp and Adamax; and

	SGD		RMSProp		Adamax	
Drive Cycle	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**
Cycle 1	0.81	1.04	0.96	1.55	0.45	0.43
Cycle 2	0.65	0.77	2.83	1.04	0.47	0.47
Cycle 3	0.68	0.91	1.19	2.24	0.57	0.63
Cycle 4	0.48	0.54	0.88	1.29	0.54	0.61
US06	1.01	2.08	1.11	2.15	0.60	0.77
HWFTa	0.40	0.35	0.39	0.34	0.41	0.32
HWFTb	0.41	0.36	1.33	2.54	0.37	0.28
UDDS	0.68	0.70	1.13	2.09	0.49	0.44
LA92	0.59	0.70	0.83	1.19	0.63	0.71
NN	1.08	1.66	0.68	0.90	0.48	0.45

Tabela 6 – MAE and MSE for all drive cycles. In each row, the best result for MAE/MSE is indicated in blue, whereas the worst result for MAE/MSE is in red.

- the choice of the optimization algorithm affects the model performance

Also note that our DFNN with 3 HL is able to estimate the SOC with a MAE/MSE smaller than 1.0% in all drive cycles, when considering 256 units per layer.

E.5 Conclusion

We present a preliminary empirical study on the impact of the optimization algorithm on the performance of a DFNN model with 3 HL, in the context of the estimation of the SOC of a Li-ion cell in ten distinct scenarios. For this, we used the dataset (20).

Our results indicate, for the 18650PF Li-ion cell (all drive cycles at 25^o C) that:

- the choice of the optimization algorithm must be made on a case-by-case basis; and
- the choice of the optimization algorithm affects the model performance.

Finally, it is important to highlight the good performance obtained with our 3 HL DFNN model, who was able to achieve a MAE/MSE smaller than 1.0% in all drive cycles, when considering 256 units per layer.