Estudo do Desempenho Aerodinâmico de *Scramjets* usando Redes Neurais Profundas

Ângelo de Carvalho Paulino[1], Angelo Passaro[1]
[1]Instituto Tecnológico de Aeronáutica (ITA), São José dos Campos/SP – Brasil

***Resumo* – A área de hipersônica tem atraído interesse mundial devido a suas aplicações espaciais e militares. Este trabalho explora o uso de Redes Neurais Profundas (DNNs, do inglês "*Deep Neural Networks*") para prever o desempenho aerodinâmico de *scramjets*. Os *datasets* de treinamento e teste foram gerados usando dados de processos de otimização computacional, cobrindo diferentes altitudes e números de Mach. Nesse contexto, os resultados mostram que as DNNs treinadas podem prever com precisão o empuxo líquido em condições de voo não ótimas. As DNNs treinadas não só substituem cálculos computacionalmente custosos, mas também aproveitam dados da otimização que são potencialmente descartáveis para melhorar sua capacidade de generalização. Contribuições incluem a capacidade das DNNs facilitarem o pré-projeto de *scramjets*, economizando recursos computacionais e fornecendo uma metodologia rápida para o estudo de desempenho em diferentes condições de voo.**

***Palavras-Chave* – Hipersônica, Redes Neurais Profundas, *Scramjets*.**

I. INTRODUÇÃO

O desenvolvimento de tecnologias hipersônicas no cenário internacional tem sido foco de crescente interesse, particularmente nas áreas de proteção térmica, combustão supersônica a ar aspirado (*scramjet*) e Veículos Hipersônicos (VH). As aplicações de interesse são amplas, desde seu uso para fins militares, tais como armas de longo alcance e alta velocidade, que são mais difíceis de serem detectadas e possuem maior capacidade de penetração [1] até seu uso como um estágio para acesso mais barato ao espaço [2]-[3]. Um seleto grupo de países, incluindo Austrália, Brasil, China, Estados Unidos, Índia, Rússia e União Europeia, têm realizado ensaios em voo de motores e veículos [1], [4]-[7]. Mais recentemente, há relatos do uso de Inteligência Artificial (IA) como valioso auxílio para projetos, operação e testes envolvendo aplicações em Hipersônica [8]-[12].

Modelos de aprendizado profundo (DL, do inglês "*Deep Learning*") têm se destacado nessa área [8], [13]. Trabalhos recentes têm aplicado com sucesso técnicas de DL para a solução de problemas em Hipersônica. Vários aspectos são abordados, desde guiagem e controle de voo [14]-[16], até quantificação e predição das propriedades dos escoamentos [17]-[18], estimativas de fluxo de calor [19], otimização de trajetória [20]-[21], estimativa de cargas aerodinâmicas nas asas [22] e até mesmo reações químicas pós-choque [13], dentre outras.

Araújo et al. [23] apresenta o uso de meta-heurísticas aplicadas a uma otimização multiobjetivo das seções de compressão de um motor *scramjet* genérico em diferentes condições de voo (altitude e velocidade) a fim de, ao mesmo tempo, maximizar o empuxo líquido ("*F_net*") e minimizar o arrasto aerodinâmico de um dado *design*, obedecendo a uma série de restrições de projeto. Tais restrições envolvem a mitigação da probabilidade de ocorrer o "*unstart*" (interrupção da combustão supersônica), ao satisfazer-se a condição de que a relação de pressão através das ondas de choque permaneça abaixo do gradiente de pressão adverso dado pelo limite de Korkegi [23]-[25], além de requisitos prescritos de temperatura e pressão. As otimizações realizadas por [23] consideram valores discretos de altitude, de 25, 30 e 35 km, e números de Mach inteiros no intervalo fechado "6 a 10". Com seus resultados, os autores sugerem que é possível projetar uma geometria *scramjet* fixa que ao mesmo tempo maximize o empuxo líquido e seja capaz de operar em uma faixa mais ampla de operação, ao invés de em apenas altitudes e número de Mach pré-estabelecidos. Contudo, averiguar essa possibilidade carece de adequada investigação para condições de voo diferentes das condições ótimas, o que é computacionalmente custoso e — por muitas abordagens — inviável. Portanto, o uso de algoritmos de DL pode se mostrar útil para tratar tais questões.

Em [26], os autores geraram 150 amostras usando *Knowledge Based Engineering* (KBE), e depois treinaram uma DNN com essas amostras para estimar o peso máximo de decolagem (MTOW, do inglês *Maximum Take-Off Weight*) para diferentes configurações de asas de um VH propulsado por um motor *turboramjet* (e não *scramjet*), impondo um voo de cruzeiro limitado a Mach 5 e altitude máxima de 28km. DNNs são também usadas para estimar o empuxo não-instalado, dessa vez de um motor *scramjet*, por meio da modificação do posicionamento e ângulo dos suportes de injeção de combustível na câmara de combustão, a partir de 100 amostras geradas com uso de dinâmica de fluidos computacional (CFD, do inglês "*Computer Fluid Dynamics*"), para um número de Mach fixado em 8 e altitude fixada em 32km [27].

Neste contexto, o objetivo principal deste trabalho é investigar a capacidade de modelos de DL baseados em Redes Neurais Profundas (DNNs, do inglês "*Deep Neural Networks*") aproximarem adequadamente os dados obtidos no processo de otimização realizado por [23], visando gerar DNNs capazes de eventualmente substituir os cálculos computacionalmente dispendiosos de processos de simulação e/ou de otimização para avaliar o desempenho aerodinâmico de veículos *scramjet* genéricos em diferentes condições de voo não estudadas anteriormente. Os resultados mostraram que as DNNs treinadas foram capazes de representar adequadamente o espaço de soluções. A metodologia aplicada neste trabalho se diferencia de [26] e [27] pelo uso de DNNs treinadas com centenas de milhares de amostras oriundas de processos de otimização multiobjetivo de diferentes configurações da seção de compressão de *scramjets*, visando aferição do desempenho aerodinâmico em uma ampla faixa de operação em termos de altitude e número de Mach.

II. FUNDAMENTAÇÃO TEÓRICA

A. Redes Neurais Profundas e Deep Learning

O *Machine Learning* (ML) é a única abordagem viável para se construir sistemas baseados em IA que possam operar em ambientes complexos [28]. As Redes Neurais (NNs, do inglês, *Neural Networks*) são um conceito fundamental em ML, inspiradas na estrutura e função do cérebro biológico [29]. Uma NN consiste em unidades interconectadas chamadas neurônios artificiais, que processam informações em camadas [30]. Esses neurônios artificiais imitam vagamente os neurônios biológicos, recebendo entradas de outros neurônios, aplicando uma função de ativação e enviando um sinal de saída para a próxima camada [31].

O DL é um subcampo do ML que utiliza NNs com múltiplas camadas ocultas, empilhadas entre as camadas de entrada e saída [32], as quais são conhecidas como DNNs [13], [28], [31]. Essas camadas ocultas permitem que a rede aprenda representações cada vez mais intrincadas dos dados, permitindo-lhe resolver problemas que estão além das capacidades de redes superficiais com menos camadas [28], [30], [33]. O aprendizado profundo revolucionou vários campos devido à sua capacidade de aprender padrões a partir de grandes quantidades de dados, alcançando alta precisão em tarefas como reconhecimento de imagem, processamento de linguagem natural e reconhecimento de fala [30].

B. Otimização no treinamento dos modelos

Algoritmos de otimização são particularmente interessantes para resolver problemas difíceis cuja complexidade inviabiliza a busca por soluções ótimas exatas em que se avalia todo o espaço de soluções possíveis [34]. No caso particular da otimização aplicada às NN, métodos de otimização podem ser usados para ajustar iterativamente seus parâmetros internos a fim de minimizar uma função de perda (*loss function*) predefinida que mede o desempenho do modelo nos dados de treinamento [28].

À medida em que se minimiza a *loss function* — ou apenas *loss* — entre o valor predito e o valor de referência/verdade de campo, também se maximiza o índice de acerto, ou acurácia, e ambos podem ser utilizados como métrica de desempenho como visto em [26], [27], [28]. Quanto mais próximo for o *loss* de zero, e a acurácia de 100%, melhor é o desempenho do modelo.

Os métodos de otimização desempenham um papel essencial no aprendizado profundo. O método *Adaptive Moment Estimation* (ADAM) se mostra uma técnica eficaz e amplamente adotada [35], [36], em virtude de facilitar a navegação eficiente na superfície de perdas e contribuir para a obtenção de melhores desempenhos. Por este motivo, este foi o método de otimização escolhido para aplicação neste trabalho. Porém, apesar de o algoritmo ADAM ser frequentemente considerado uma técnica robusta quanto à escolha de hiperparâmetros, é comum que o hiperparâmetro *learning rate* (*lr*) precise ser ajustado a fim de se obter melhores resultados [28], [37].

C. Ajuste de Hiperparâmetros

Hiperparâmetros são configurações que controlam o processo de aprendizagem de um modelo de aprendizagem profunda, mas não são aprendidos diretamente a partir dos dados em si [28]. Exemplos de hiperparâmetros incluem o número de camadas ocultas, o número de neurônios em cada camada, o *lr* e a escolha da função de ativação. A seleção dos hiperparâmetros ideais é de grande importância para alcançar o desempenho desejado em DNNs [28], [38].

No caso particular deste trabalho, a **pesquisa em grade** [38] se mostra mais interessante para variar o hiperparâmetro *lr*, em virtude de haver poucos parâmetros de entrada e de se viabilizar um maior controle sobre o processo de treinamento.

III. METODOLOGIA

Neste trabalho, treinou-se uma DNN para a predição do desempenho aerodinâmico de configurações da seção de compressão de um motor scramjet. O treinamento da DNN faz uso de uma base de dados gerada a partir de estudos anteriores que utilizaram algoritmos estocásticos de otimização computacional. A implementação computacional é realizada em linguagem Python. Nas próximas subseções será detalhado o processo metodológico utilizado para a definição da arquitetura, treinamento e otimização da DNN.

A. Bases de Dados (Datasets)

Durante o processo de otimização realizado em [23] foram geradas várias soluções-tentativa (indivíduos) em que se obtinham, dentre outros, valores de empuxo líquido ("*F_net*") com base em determinados parâmetros de entrada, correspondentes às condições de voo e características geométricas de *scramjets* genéricos. Todos os parâmetros de entrada utilizados neste trabalho constam da Tabela 1.

Os dados oriundos das rotinas de otimização utilizados como *datasets* para treinar modelos de DNN neste trabalho constam da Tabela 2. O primeiro *dataset* contempla apenas uma condição de voo (número de Mach = 7 e altitude de 30 km), enquanto o segundo é composto por um conjunto mais amplo de condições de voo, com números de Mach variando de 6 a 10 e altitudes de voo variando de 25 a 35 km, com diferentes temperaturas mínimas na entrada da câmara de combustão.

TABELA I. DESCRIÇÃO DOS PARÂMETROS DE ENTRADA.

Parâmetro	Descrição
h3	Altura da câmara de combustão
y0	Intensidade do choque para a 1ª rampa
y1	Intensidade do choque para a 2ª rampa
y2	Intensidade do choque para a 3ª rampa
y3	Intensidade do choque para a 4ª rampa
T0	Temperatura do freestream*
P0	Pressão do freestream*
M0	Número de Mach do freestream*
Tmin	Temperatura mínima de entrada na câmara de combustão*
F_net	Empuxo líquido gerado pelo scramjet**

* Parâmetros de entrada presentes apenas no dataset 2
** Parâmetro "alvo" do treinamento

TABELA II. QUANTIDADE DE AMOSTRAS POR DATASET, DADAS AS CONDIÇÕES DE VOO DAS ROTINAS DE OTIMIZAÇÃO.

Nome do Dataset	Número de Mach "M0"	Altitude "Z" (Km)	Temperatura mínima do ar na entrada da câmara de combustão - "Tmin" (K)	Parâmetros de entrada	Número de amostras	Total de amostras no dataset
Dataset 1	7	30	800	5	19.800	19.800
Dataset 2	6	25	1.100	9	19.800	236.640
	6	30	800	9	19.800	
	6	35	800	9	19.800	
	7	25	1.100	9	19.800	
	7	30	800	9	19.800	
	8	25	1.100	9	19.800	
	8	30	800	9	19.800	
	8	35	800	9	18.840	
	9	25	1.100	9	19.800	
	9	25	800	9	19.800	
	9	30	800	9	19.800	
	10	30	800	9	19.800	

B. Implementação computacional

Existem várias bibliotecas em linguagem de programação que fornecem rotinas de treinamento para modelos de DNN. A biblioteca escolhida para este estudo foi a biblioteca PyTorch, que fornece modelos de DL em estilo de programação imperativo, na linguagem Python, apresentando depuração facilitada e consistência com outras bibliotecas de computação científica populares, enquanto permanece eficiente e suporta aceleradores de hardware, tais como GPUs [39], disponível sem custos e amplamente adotado, conforme mais de 19.500 citações em trabalhos científicos encontrados na plataforma Scopus até esta data.

Adicionalmente à biblioteca PyTorch foram utilizadas rotinas de outras bibliotecas em Python: para a gerar gráficos foram utilizadas as bibliotecas Matplotlib [40] e Plotly [41]; para lidar com estruturas de dados tabulares foi utilizada a biblioteca Pandas [42]; para ferramentas estatísticas e ML, Scikit-learn [43] e estruturas de dados para cálculos numéricos a biblioteca NumPy [44].

Foi desenvolvido um código computacional em Python para realizar o tratamento e a preparação dos dados presentes nos *datasets*, bem como para a criação da arquitetura das DNNs utilizadas (Fig. 1), realização do treinamento de modelos de DNN com dados de processos de otimização de veículos *scramjet* e ainda os estudos de aplicação das DNNs treinadas.

C. Arquitetura da DNN

Para definir a arquitetura a ser utilizada, foram inicialmente testados modelos simples de DNNs disponíveis na biblioteca PyTorch [39], frequentemente utilizados para demonstrar funcionalidades básicas. A análise das perdas (*loss*) ao modelar os dados do *dataset 1* revelou que esses modelos de exemplo não eram capazes de capturar os padrões essenciais dos parâmetros escolhidos para este estudo, indicando um caso de *underfitting*. Portanto, tornou-se necessário modificar esses modelos e aumentar gradualmente sua complexidade para alcançar níveis de acurácia satisfatórios.

A função de perda adotada durante o treinamento foi o "erro quadrático médio" (MSE, do inglês *Mean-Square Error*) [28], e alguns resultados foram apresentados em termos da raiz desse valor (RMSE, do inglês *Root Mean-Square Error*) [45].

O processo de testes e verificação envolveu aumentar iterativamente o número de camadas ocultas e de neurônios em cada camada oculta, realizando novas avaliações nos dados de teste do *dataset 1* a fim de aferir tanto o *loss* quanto o índice R^2 a cada iteração do processo de definição da rede.

O modelo inicial possuía apenas 4 camadas ocultas e um total de 102 neurônios, e foi gradativamente evoluindo até o modelo final com 8 camadas ocultas e um total de 679 neurônios. Cada arquitetura gerada foi treinada com o *dataset 1*, e a arquitetura foi definida de forma permanente quando se alcançou um *loss* inferior a 1e-4 e um R^2 superior a 95% no conjunto de testes.

Após os vários testes, foi empiricamente determinado um modelo de DNN capaz de lidar adequadamente com as características das amostras relativas ao problema em questão. Sua arquitetura está representada na Fig. 1. A primeira camada (Entrada) recebe uma matriz onde o número de linhas reflete a quantidade de amostras de cada *Fold* e o número de colunas corresponde ao número de parâmetros de entrada selecionados para o processo de treinamento/predição. A última camada (Saída), por sua vez, representa um tensor que fornece o valor predito para a variável de interesse, "*F_net*".

O número de neurônios nas camadas ocultas também foi determinado empiricamente e a função de ativação do tipo "*rectified linear unit*" (ReLU), cujo uso é recomendado na literatura [28], é utilizada em cada camada.

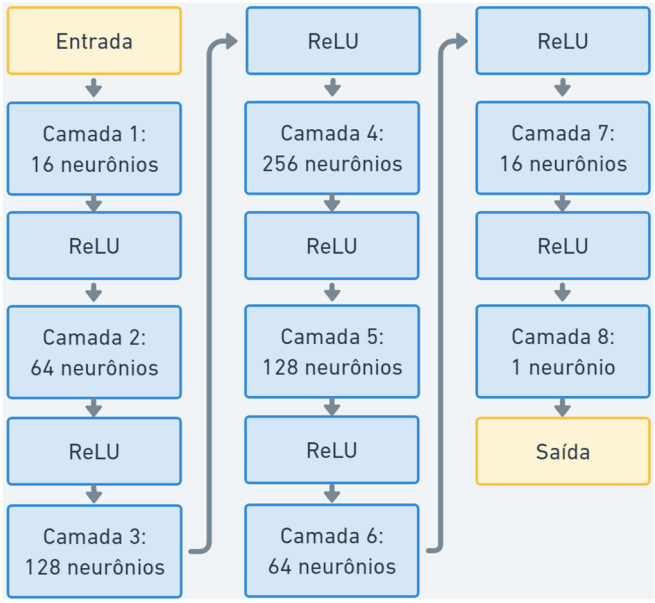

Fig. 1. Arquitetura das redes utilizadas para os experimentos.

D. Experimentos

Para todos os experimentos, foi utilizada uma abordagem de *cross-validation* (CV) chamada "7FCV", que tem sido adotada na literatura com bons resultados [46]-[47], onde se usam 5 partições dos dados para efetuar o treinamento, uma partição para validação/ajuste de hiperparâmetros, e uma partição para testes de generalização (conjunto "*Test Data*").

Assim, formam-se 6 combinações possíveis para se utilizar 5 partes para treinamento e 1 para validação. Dessa forma, cada uma das 6 combinações possíveis se constitui em um único *Fold*, contendo tanto os dados para treinamento quanto os correspondentes dados de validação. Para o *dataset 1*, cada um dos *Folds* ficou com 14.121 amostras para treinamento e 2.825 para validação. Já para o *dataset 2*, cada *Fold* possui 168.994 amostras para treino e 33.799 para validação. Ambos os *datasets* foram normalizados entre [0,1].

Para o ajuste de hiperparâmetros, foi adotada a abordagem de "pesquisa em grade" para determinar o hiperparâmetro "*lr*", em virtude de haver poucos parâmetros de entrada e de se viabilizar maior controle sobre o processo de treinamento da DNN. Os valores de *lr* foram variados entre os valores discretos [0,0002; 0,0005; 0,001; ou 0,002] nos testes.

Enquanto os dados do *dataset 1* foram usados apenas para a definição da arquitetura, os dados do *dataset* 2 foram utilizados para os experimentos a fim de se avaliar a capacidade preditiva da arquitetura proposta em condições amplas de voo, a saber, números de Mach variando de 6 a 10 e altitudes de voo variando de 25 a 35 km, com diferentes temperaturas mínimas na entrada da câmara de combustão.

IV. RESULTADOS E DISCUSSÕES

Com a metodologia estabelecida, foram gerados resultados que tratam dos diferentes aspectos da abordagem exploratória já apresentada. Ainda, a arquitetura proposta na metodologia foi treinada e avaliada em diversas condições. Os resultados viabilizaram a aferição da capacidade preditiva das DNNs geradas tanto em modelar adequadamente conjuntos de dados de processos de otimização quanto em oferecer estimativas razoáveis para o desempenho aerodinâmico de *scramjets* em diferentes condições de voo.

A. Realização de Experimentos

Para os experimentos foi utilizado o *dataset 2*, com uma busca em grade para valores de *lr* iguais a [0,0002; 0,0005; 0,001; ou 0,002], com um número de épocas ajustado para o valor fixo de 20.000. Os resultados constam das Figs. 2 a 5. Ao se analisar os resultados para o *dataset* filtrado, depreendeu-se que:

- O melhor desempenho da arquitetura adotada pôde prever o comportamento de "*F_net*" adequadamente, com um RMSE, de **2,183e-3**. O desempenho do melhor modelo **excedeu os 99%**, com um valor de **R^2 maior que 99,98%**.
- Não houve diferença de desempenho estatisticamente significativa entre os *Folds*, exceto no tocante ao Fold 6, que se mostrou consideravelmente pior que os demais. O menor valor geral de loss foi obtido com o ***Fold 2***, ao passo que o ***Fold 5*** apresentou a mediana mais baixa e a menor variabilidade, sugerindo um desempenho mais consistente e possivelmente melhor em termos de perda.
- Os valores de *lr* iguais a 0,001 e 0,002 apresentaram medianas muito semelhantes, com destaque para um menor valor de *loss* ao se utilizar *lr 0,002*. Estes valores de *lr* podem ser mais favoráveis à obtenção de bons desempenhos em investigações futuras.

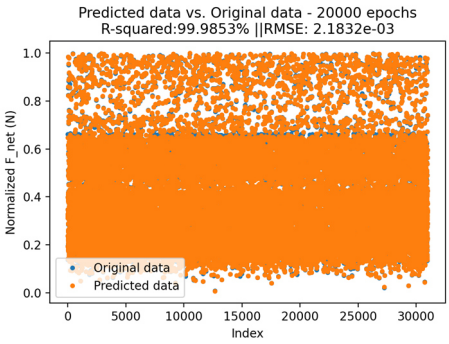

Fig. 2 – Comparativo de valores normalizados de *F_net*: Original X Predito.

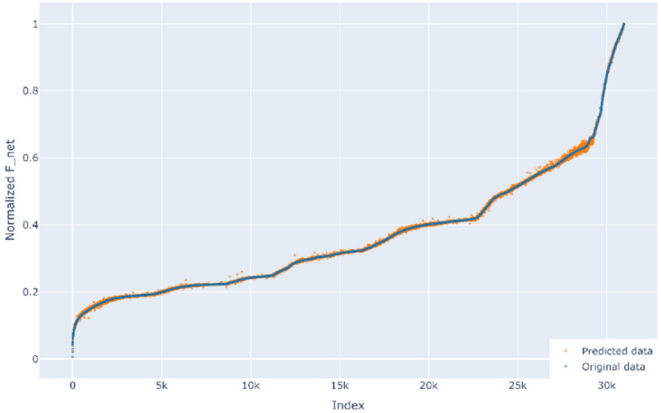

Fig. 3 – Comparativo de valores normalizados **ordenados** de *F_net*: Original X Predito.

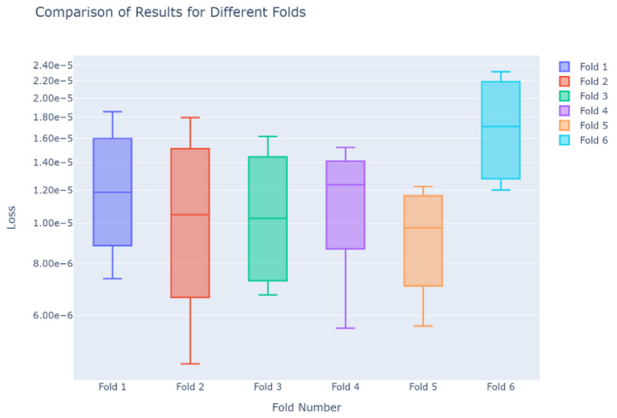

Fig. 4 – Comparativo de desempenho por *Fold*.

Fig. 5 – Comparativo de desempenho por *learning rate*.

B. Exploração do desempenho

Com as DNNs treinadas e capazes de estimar os valores de "*F_net*", é possível agora estimar o desempenho de um veículo *scramjet* genérico com configurações fixas de parâmetros de entrada fora dos valores ótimos de altitude e número de Mach. Isso é importante porque o *scramjet* genérico adotado não apresenta partes móveis e é necessário avaliar seu desempenho em condições de voo fora daquelas para as quais foi projetado. Dos 9 parâmetros de entrada, 3 dependem das condições do escoamento ("T0", "P0" e "M0") e outros 5 dependem apenas da geometria do veículo ("h3", "y0", "y1", "y2" e "y3"). Quanto ao parâmetro "Tmin", identificou-se que os melhores valores de "*F_net*" foram obtidos com valores de "Tmin" iguais a 800K. Assim, posto que "T0" e "P0" dependem da altitude de voo, e "M0" apenas da velocidade do voo (em altitudes fixas), foi gerada uma tabela de valores preditos pelas redes a fim de evidenciar o valor de "*F_net*" considerando: altitude variável, número de Mach variável, e configurações geométricas fixas, para um valor de "Tmin"=800K.

A Fig. 6 mostra o resultado dos valores "*F_net*" preditos para condições de voo substancialmente diferentes daquelas para as quais a geometria foi considerada ótima em [23] pela melhor DNN treinada para estimação de "*F_net*". Cabe ressaltar que a DNN foi capaz de identificar valores de "*F_net*" ligeiramente superiores (até 1%) àquele obtido pela geometria fixa nas condições nominais (obtidas como resultado do processo de otimização) tanto para o número de Mach nominal quanto para números de Mach que não haviam sido previamente testados devido às limitações computacionais inerentes ao processo de otimização.

Além disso, a Fig. 6 dá uma noção acurada do comportamento de uma geometria fixa em condições de voo muito diferentes das nominais relativas ao envelope de voo previamente considerado "ótimo". Tal fato é comprovado pelos valores de "*F_net*", ou seja, do empuxo líquido, que se mostraram sempre positivos mesmo no pior cenário, que seria o equivalente a Mach 6 e 35km de altitude. Tal achado fornece a confiança necessária para a realização de testes de avaliação mais elaborados da geometria, tais como os realizados por CFD ou ainda por meios experimentais.

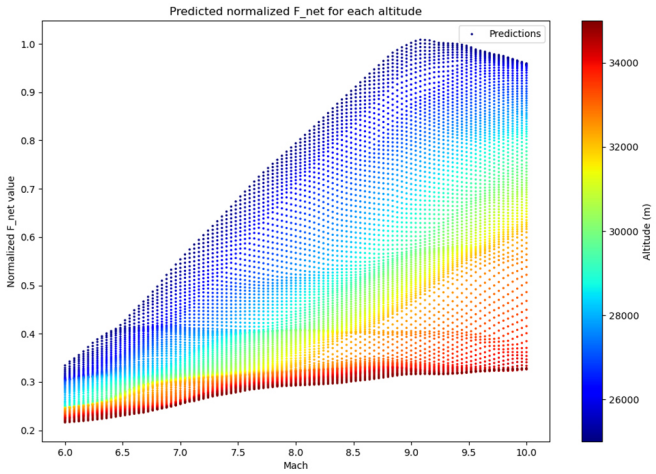

Fig. 6 – Valores normalizados de "*F_net*" preditos para diferentes condições de voo (altitude e número de Mach) pela melhor DNN treinada.

Desse modo, a metodologia apresentada neste trabalho foi capaz de realizar a avaliação da viabilidade de se utilizar geometrias fixas em faixas amplas de condições de operação, suprindo a necessidade que já havia sido apontada por [23].

V. CONCLUSÃO

Este trabalho avaliou a capacidade de diferentes modelos de DNN preverem adequadamente o desempenho aero-dinâmico de configurações variadas de veículos *scramjet* em condições de voo diferentes das consideradas ótimas. Investigou-se diferentes valores do hiperparâmetro "*lr*" e combinações de dados de *cross-validation* no treinamento dos modelos com amostras oriundas de processos de otimização computacional. Os resultados mostraram que as DNNs treinadas obtiveram bom desempenho em prever o valor de "*F_net*" (mais de 99% de acurácia) para uma dada configuração geométrica fornecida como parâmetro de entrada. Assim, o uso das DNNs treinadas se mostrou útil para analisar, já na fase de pré-projeto, a viabilidade de se utilizar geometrias fixas de *scramjets* para uma faixa mais ampla de operação em termos de altitude e número de Mach.

Os modelos treinados de "*F_net*" auxiliam no estudo de diferentes configurações de *scramjets* para operação em diversas condições de voo. Este trabalho fornece uma metodologia capaz de avaliar cada uma das soluções ótimas em condições de voo muito diferentes das nominais, não estudadas previamente, contribuindo para um pré-projeto mais robusto visando voos reais.

Como trabalhos futuros sugere-se a investigação da capacidade preditiva das DNNs treinadas neste trabalho com respeito a faixas de altitude e número de Mach que extrapolem aquelas com as quais as DNNs foram treinadas. Assim, seria possível avaliar o comportamento (em termos de "*F_net*") de uma dada geometria em regiões que não foram originalmente contempladas pelo processo de otimização.

REFERÊNCIAS

[1] R. Speier, G. Nacouzi, C. Lee, and R. Moore, Hypersonic Missile Nonproliferation: Hindering the Spread of a New Class of Weapons. Santa Monica, CA: RAND Corporation, 2017. doi: 10.7249/RR2137.

[2] G. Russo, C. Voto, and R. Savino, "S4 - A demonstrator of HYPLANE, a single stage suborbital spaceplane and a hypersonic business jet," Acta Astronaut., vol. 183, pp. 244–254, Jun. 2021, doi: 10.1016/j.actaastro.2021.03.025.

[3] G. Scarlatella et al., "Assessment of Mission Capabilities of a Reusable Heavy-lift Launch Vehicle Concept with Aerospike Engine," in AIAA SCITECH 2024 Forum, Reston, Virginia: American Institute of Aeronautics and Astronautics, Jan. 2024, p. Orlando.

[4] E. T. Curran, "Scramjet Engines: The First Forty Years," J. Propuls. Power, vol. 17, no. 6, pp. 1138–1148, Nov. 2001, doi: 10.2514/2.5875.

[5] D. Sziroczak and H. Smith, "A review of design issues specific to hypersonic flight vehicles," Prog. Aerosp. Sci., vol. 84, pp. 1–28, Jul. 2016, doi: 10.1016/J.PAEROSCI.2016.04.001.

[6] I. da S. Rêgo, A. Passaro, D. Carinhana, M. A. S. Minucci, and É. J. de Oliveira, "Brazilian suborbital rockets for hypersonic flight testing: A review and a market perspective," in Proceedings of the Intl. Astro. Congress, IAC, International Astronautical Federation, IAF, Jan. 2019.

[7] J. F. de A. Martos, I. da S. Rêgo, S. N. P. Laiton, B. C. Lima, F. J. Costa, and P. G. de P. Toro, "Experimental Investigation of Brazilian 14-X B Hypersonic Scramjet Aerospace Vehicle," Int. J. Aerosp. Eng., vol. 2017, pp. 1–10, Jan. 2017, doi: 10.1155/2017/5496527.

[8] W. Wang and J. Ma, “A review: Applications of machine learning and deep learning in aerospace engineering and aero-engine engineering,” Adv. Eng. Innov., vol. 6, no. 1, pp. 54–72, Feb. 2024, doi: 10.54254/2977-3903/6/2024060.

[9] S. Ai, J. Song, and G. Cai, “A real-time fault diagnosis method for hypersonic air vehicle with sensor fault based on the auto temporal convolutional network,” Aerosp. Sci. Technol., vol. 119, p. 107220, Dec. 2021, doi: 10.1016/j.ast.2021.107220.

[10] T. McCall, K. Seyed Alavi, L. Rana, and B. Chudoba, “Artificial Intelligent Research Assistant for Aerospace Design Synthesis—Solution Logic,” in 22nd AIAA International Space Planes and Hypersonics Systems and Technologies Conference, Reston, Virginia: American Institute of Aeronautics and Astronautics, Sep. 2018. doi: 10.2514/6.2018-5387.

[11] L. Cui, A. Liu, C. Yv, and T. Quan, “Learning Algorithm for Tracking Hypersonic Targets in Near Space,” in Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, vol. 227 LNICST, 2018, pp. 206–214. doi: 10.1007/978-3-319-73447-7_24.

[12] Â. de C. Paulino and A. Passaro, “Investigação cientométrica sobre aplicações de Redes Neurais em Hipersônica,” in XXV Simpósio de Aplicações Operacionais em Áreas de Defesa, São José dos Campos, Brasil, 2023, p. 1.

[13] Z. Mao, L. Lu, O. Marxen, T. A. Zaki, and G. E. Karniadakis, “DeepM&Mnet for hypersonics: Predicting the coupled flow and finite-rate chemistry behind a normal shock using neural-network approximation of operators,” J. Comput. Phys., vol. 447, 2021, doi: 10.1016/j.jcp.2021.110698.

[14] Y. Shou, T. Yan, B. Xu, and F. Sun, “Integrated guidance and control of hypersonic flight vehicle with coordinated mission requirement and input constraint,” Int. J. Robust Nonlinear Control, 2023, doi: 10.1002/rnc.6607.

[15] X. Bu, C. Hua, M. Lv, and Z. Wu, “Flight Control of Waverider Vehicles with Fragility-avoidance Prescribed Performance,” IEEE Trans. Aerosp. Electron. Syst., pp. 1–15, 2023, doi: 10.1109/TAES.2023.3251314.

[16] J. Lv, C. Wang, and Y. Kao, “Adaptive fixed-time quantized fault-tolerant attitude control for hypersonic reentry vehicle,” Neurocomputing, vol. 520, pp. 386–399, Feb. 2023, doi: 10.1016/j.neucom.2022.11.057.

[17] C. Fujio and H. Ogawa, “Deep-learning prediction and uncertainty quantification for scramjet intake flowfields,” Aerosp. Sci. Technol., vol. 130, p. 107931, Nov. 2022, doi: 10.1016/j.ast.2022.107931.

[18] E. Ozbenli, P. Vedula, K. Vogiatzis, and E. Josyula, “Numerical solution of hypersonic flows via artificial neural networks,” in AIAA Scitech 2020 Forum, 2020. doi: 10.2514/6.2020-1233.

[19] D. Ding, H. Chen, Z. Ma, B. Zhang, and H. Liu, “Heat flux estimation of the cylinder in hypersonic rarefied flow based on neural network surrogate model,” AIP Adv., vol. 12, no. 8, 2022, doi: 10.1063/5.0108757.

[20] Y. WANG and Z. JIANG, “Theories and methods for designing hypersonic high-enthalpy flow nozzles,” Chinese J. Aeronaut., vol. 35, no. 1, pp. 318–339, 2022, doi: 10.1016/j.cja.2021.01.018.

[21] J. Shi, J. Wang, L. Su, Z. Ma, and H. Chen, “A Neural Network Warm-Started Indirect Trajectory Optimization Method,” Aerospace, vol. 9, no. 8, 2022, doi: 10.3390/aerospace9080435.

[22] A. Beachy, H. Bae, I. Boyd, and R. Grandhi, “Emulator embedded neural networks for multi-fidelity conceptual design exploration of hypersonic vehicles,” in AIAA Scitech 2021 Forum, American Institute of Aeronautics and Astronautics Inc, AIAA, 2021, pp. 1 – 24. [Online].

[23] P. P. B. Araújo, R. Y. Tanaka, C. A. Silva, A. Passaro, and P. G. P. Toro, “Multi-objective optimization of a hypersonic airbreathing vehicle,” Phys. Fluids, vol. 36, no. 2, Feb. 2024, doi: 10.1063/5.0181366.

[24] R. H. Korkegi, “Comparison of Shock-Induced Two- and Three-Dimensional Incipient Turbulent Seperation,” AIAA J., vol. 13, no. 4, pp. 534–535, Apr. 1975, doi: 10.2514/3.49750.

[25] R. H. Korkegi, “A Simple Correlation for Incipient-Turbulent Boundary-Layer Separation due to a Skewed Shock Wave,” AIAA J., vol. 11, no. 11, pp. 1578–1579, Nov. 1973, doi: 10.2514/3.50637.

[26] D. Chen, Y. Li, J. Guo, and Y. Li, “Estimation of hypersonic vehicle weight using Physics-Informed neural network supported by knowledge based engineering,” Expert Syst. Appl., vol. 195, p. 116609, Jun. 2022, doi: 10.1016/j.eswa.2022.116609.

[27] A. C. Ispir, B. H. Saracoglu, T. Magin, and A. Coussement, “A methodology for estimating hypersonic engine performance by coupling supersonic reactive flow simulations with machine learning techniques,” Aerosp. Sci. Technol., vol. 140, p. 108501, Sep. 2023, doi: 10.1016/j.ast.2023.108501.

[28] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, 2016. [Online]. Available: https://www.deeplearningbook.org/

[29] W. McCulloch and W. Pitts, “A Logical Calculus of the Ideas Immanent in Nervous Activity (1943),” in Ideas That Created the Future, vol. 5, no. 4, The MIT Press, 2021, pp. 79–88. doi: 10.7551/mitpress/12274.003.0011.

[30] J. Schmidhuber, “Deep Learning in neural networks: An overview,” Neural Networks, vol. 61, pp. 85–117, Jan. 2015, doi: 10.1016/J.NEUNET.2014.09.003.

[31] S. S. Haykin, Neural networks and learning machines, Third. Upper Saddle River, NJ: Pearson Education, 2009.

[32] Y. Lecun, Y. Bengio, and G. Hinton, “Deep learning,” Nature, vol. 521, no. 7553, pp. 436–444, May 2015, doi: 10.1038/NATURE14539.

[33] G. E. Hinton, S. Osindero, and Y.-W. Teh, “A Fast Learning Algorithm for Deep Belief Nets,” Neural Comput., vol. 18, no. 7, pp. 1527–1554, Jul. 2006, doi: 10.1162/neco.2006.18.7.1527.

[34] B. Doğan and T. Ölmez, “A new metaheuristic for numerical function optimization: Vortex Search algorithm,” Inf. Sci. (Ny)., vol. 293, no. February, pp. 125–145, Feb. 2015, doi: 10.1016/j.ins.2014.08.053.

[35] G. Habib and S. Qureshi, “Optimization and acceleration of convolutional neural networks: A survey,” J. King Saud Univ. - Comput. Inf. Sci., vol. 34, no. 7, pp. 4244–4268, Jul. 2022, doi: 10.1016/j.jksuci.2020.10.004.

[36] Y. Tian, Y. Zhang, and H. Zhang, “Recent Advances in Stochastic Gradient Descent in Deep Learning,” Mathematics, vol. 11, no. 3, p. 682, Jan. 2023, doi: 10.3390/math11030682.

[37] I. Loshchilov and F. Hutter, “Decoupled weight decay regularization,” in Proceedings of the 7th International Conference on Learning Representations (ICLR), 2019.

[38] J. Bergstra, J. B. Ca, and Y. B. Ca, “Random Search for Hyper-Parameter Optimization Yoshua Bengio,” J. Mach. Learn. Res., vol. 13, pp. 281–305, 2012, Accessed: Mar. 26, 2024. [Online]. Available: http://scikit-learn.sourceforge.net.

[39] A. Paszke et al., “PyTorch: An Imperative Style, High-Performance Deep Learning Library,” in Advances in Neural Information Processing Systems, H. Wallach, H. Larochelle, A. Beygelzimer, F. d\textquotesingle Alché-Buc, E. Fox, and R. Garnett, Eds., Curran Associates, Inc., 2019.

[40] J. D. Hunter, “Matplotlib: A 2D Graphics Environment,” Comput. Sci. Eng., vol. 9, no. 3, pp. 90–95, 2007, doi: 10.1109/MCSE.2007.55.

[41] P. T. Inc., “Collaborative data science.” Plotly Technologies Inc., Montreal, QC, 2015. [Online]. Available: https://plot.ly

[42] W. McKinney, “Data Structures for Statistical Computing in Python,” in Proceedings of the 9th Python in Science Conference, S. van der Walt and J. Millman, Eds., 2010. doi: 10.25080/Majora-92bf1922-00a.

[43] F. Pedregosa et al., “Scikit-learn: Machine Learning in Python,” J. Mach. Learn. Res., vol. 12, pp. 2825–2830, 2011.

[44] C. R. Harris et al., “Array programming with NumPy,” Nature, vol. 585, no. 7825, pp. 357–362, Sep. 2020, doi: 10.1038/s41586-020-2649-2.

[45] T. Chai and R. R. Draxler, “Root mean square error (RMSE) or mean absolute error (MAE)? -Arguments against avoiding RMSE in the literature,” Geosci. Model Dev., vol. 7, no. 3, 2014.

[46] Â. de C. Paulino, L. N. F. Guimarães, and E. H. Shiguemori, “Hybrid Adaptive Computational Intelligence-based Multisensor Data Fusion applied to real-time UAV autonomous navigation,” Intel. Artif., vol. 22, no. 63, pp. 162–195, May 2019, doi: 10.4114/intartif.vol22iss63pp162-195.

[47] Â. de C. Paulino, L. N. F. Guimaraes, and E. H. Shiguemori, “Assessment of Noise Impact on Hybrid Adaptive Computational Intelligence Multisensor Data Fusion Applied to Real-Time UAV Autonomous Navigation,” IEEE Lat. Am. Trans., vol. 18, no. 02, pp. 295–302, Feb. 2020, doi: 10.1109/TLA.2020.9085283.