Universidade do Estado do Rio de Janeiro

Centro de Tecnologia e Ciências

Faculdade de Engenharia

Leonardo Fernandes Ferreira

Modelo Deep Learning para Dinâmica dos Fluidos Computacional

Rio de Janeiro

2019

Leonardo Fernandes Ferreira

Modelo Deep Learning para Dinâmica dos Fluidos Computacional

Projeto de graduação apresentado, como requisito parcial para obtenção do grau de Engenheiro Mecânico, a Faculdade de Engenharia, da Universidade do Estado do Rio de Janeiro.

Orientadores: Gustavo Rabello dos Anjos

Karla Tereza Figueiredo Leite

Norberto Mangiavacchi

Rio de Janeiro

2019

Ficha elaborada pelo autor através do
Sistema para Geração Automática de Ficha Catalográfica da Rede Sirius - UERJ

F383 Ferreira, Leonardo Fernandes
Modelo Deep Learning para Dinâmica dos Fluidos Computacional / Leonardo Fernandes Ferreira. - 2019.
40 f.

Orientador: Norberto Mangiavacchi
Trabalho de Conclusão de Curso apresentado à Universidade do Estado do Rio de Janeiro, Faculdade de Engenharia, para obtenção do grau de bacharel em Engenharia Mecânica.

1. dinâmica dos fluidos computacional - Monografias. 2. método do reticulado de boltzmann - Monografias. 3. deep learning - Monografias. 4. redes neurais - Monografias. I. Mangiavacchi, Norberto. II. Universidade do Estado do Rio de Janeiro. Faculdade de Engenharia. III. Título.

CDU 621

Leonardo Fernandes Ferreira

Modelo Deep Learning para Dinâmica dos Fluidos Computacional

Projeto de graduação apresentado, como requisito parcial para obtenção do grau de Engenheiro Mecânico, a Faculdade de Engenharia, da Universidade do Estado do Rio de Janeiro.

Aprovado em: 11 de dezembro de 2019

Banca Examinadora:

Prof. Dr. Norberto Mangiavacchi (Orientador)
Faculdade de Engenharia da UERJ

Profª Dra. Karla Teresa Figueiredo Leite (Co-orientadora)
Instituto de Matemática e Estatística da UERJ

Prof. Dr. Daniel José Nahid Mansur Chalhub
Faculdade de Engenharia da UERJ

Prof. Dr. Gustavo Rabello dos Anjos (Co-orientador)
Universidade Federal do Rio de Janeiro - UFRJ - COPPE

Rio de Janeiro

2019

RESUMO

FERREIRA, Leonardo Fernandes. *Modelo Deep Learning para Dinâmica dos Fluidos Computacional*. 43 f. Projeto Final (Graduação em Engenharia Mecânica) - Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro (UERJ), Rio de Janeiro, 2019.

A dinâmica dos fluidos computacionais (CFD) é um campo de estudo de grande interesse e com diversas aplicações, e que, devido a complexidade e extensividade dos cálculos envolvidos, pode demandar um grande tempo computacional para ser executada. Por outro lado, as Redes Neurais Artificais (RN) são um campo que já mostrou-se útil em diversas áreas do conhecimento, inclusive em dinâmica de fluidos. Com o intuito de se explorar uma relação entre essas áreas, e investigar possíveis melhorias no ramo de CFD, foram criadas simulações de escoamentos monofásicos usando-se o Método do Reticulado de Boltzmann, e então aplicadas técnicas de RN para recriar as simulações, considerando apenas o primeiro frame da sequencia da dinâmica produzido pelo simulador. Os resultados obtidos a partir dele mostram-se coerentes com os resultados esperados, e a generalização do método apresenta resultados promissores. Assim, utilização de técnicas de RN revela-se uma boa prova de conceito de suas aplicações na área de CFD.

Palavras-chave: dinâmica dos fluidos computacional, método do reticulado de boltzmann, deep learning, redes neurais convolucionais

ABSTRACT

The computational fluid dynamics (CFD) is a field of study of great interest and with several applications, and that, due to the complexity and extensiveness of the calculations involved, may require a large computational time to perform. On the other hand, Artificial Neural Networks (NN) is a field that has been shown to be useful in several areas of knowledge, including fluid dynamics. In order to explore a relationship between these areas, and investigate possible improvements in the CFD branch, single-phase flow simulations were created using the Lattice Boltzmann Method, and then NN techniques were used to recreate the simulations, considering only the first frame of the dynamics sequence produced by the simulator. The results obtained from it are consistent with the expected results, and the generalization of the method presents promising results. Thus, the use of NN techniques proves to be a good proof of concept for their CFD applications.

Keywords: computational fluid dinamics, lattice boltzmann method, neural network, convolutional neural network.

LISTA DE FIGURAS

Figura 1 Discretização do método do reticulado Boltzmann. 12
Figura 2 Ilustração do passo de Colisão em um modelo. 13
Figura 3 Ilustração do passo de propagação do LBM 14
Figura 4 Ilustração da condição de não escorregamento no LBM 15
Figura 5 Diagrama de rede neural e representação das operações realizadas nela. 17
Figura 6 Representação do método do gradiente. 19
Figura 7 Exemplo da operação de convolução. 21
Figura 8 Exemplo da operação de convolução transposta. 22
Figura 9 Exemplos de objetos gerados 23
Figura 10 Visão geral da arquitetura da rede 26
Figura 11 Exemplo de resultado 28
Figura 12 Teste de generalização com um único objeto. 29
Figura 13 Comparação visual da velocidade, divergência e vorticidade das simulações. 30
Figura 14 Evolução do erro médio quadrático entre as simulações em função do tempo. 31
Figura 15 Teste de generalização com vários objetos. 32
Figura 16 Valores de velocidade ao longo de um corte feito em X=200, e sua respectiva transformada de Fourier. 33
Figura 17 Bloco residual básico usado na construção da rede. 39
Figura 18 Primeira parte do modelo, o *encoder*. 40
Figura 19 Segunda parte, a evolução temporal 41
Figura 20 Terceira parte, o *decoder* 42

LISTA DE TABELAS

Tabela 1 Parâmetros das camadas do encoder e decoder 43
Tabela 2 Parâmetros das camadas de evolução temporal. 43

SUMÁRIO

1 **INTRODUÇÃO** 9

1.1 **Motivação** 9

1.2 **Objetivo** 9

1.3 **Descrição da monografia** 10

2 **FUNDAMENTAÇÃO TEÓRICA** 11

2.1 **Método do reticulado de Boltzmann** 11

2.1.1 Condições de contorno 14

2.2 **Redes Neurais** 15

2.2.1 Backpropagation 18

2.3 **Redes Neurais Convolucionais** 19

2.3.1 Convolução Transposta 21

3 **MÉTODO** 23

3.1 **Geração do dataset** 23

3.2 **Arquitetura da RN** 24

4 **RESULTADOS E DISCUSSÃO** 27

CONCLUSÃO 35

REFERÊNCIAS 36

5 **APÊNDICE A** 39

1 INTRODUÇÃO

A área de aprendizado de máquinas, especialmente o ramo de redes neurais, tem sido atualmente um grande centro de atenção, e tem visto aplicações nas mais diversas áreas de estudos. O campo de dinâmica dos fluidos computacional (CFD) mostra-se um grande candidato a aplicação dessas técnicas, e também tem sido um foco de interesse.

Recentemente alguns estudos foram publicados sobre aplicações de redes neurais na área de CFD. Alguns trabalhos investigam as melhorias no tempo de computação de algumas técnicas de CFD, através da substituição de termos complexos por aproximações simples geradas pela rede [1,2]. Outros geram simulações da evolução física de fumaças [3] ou gotas [4], partes também de interesse na área.

Nesse trabalho foi investigado o comportamento a dinâmica de escoamentos turbulentos, assim como feito em [5–7]. E foi treinado um modelo que produz simulações que mostraram-se semelhantes às produzidas pelo simulador original.

1.1 Motivação

A dinâmica dos fluidos computacional é um ramo da dinãmica dos fluidos, que lida com a análise e solução numérica de problemas de fluidodinâmica [8]. É muito usada em áreas como engenharia para cálculo de aero e hidrodinâmica [9, 10], em geologia para análise de materiais porosos [11], e até medicina para modelagem do fluxo de fármacos no interior de artérias [12], entre outras.

Apesar da importância, os métodos de simulação e CFD são conhecidos também por serem lentos, necessitando de muito tempo computacional para serem executados, devido a quantidade de cálculos relativos ao processo físico envolvido.

1.2 Objetivo

Tendo em vista o interesse e limitações na área, foi proposto um método para realizar a simulação com o uso de redes neurais (RN), e que pode ser importante para reduzir o tempo de simulação, uso de memória e até melhorar a exatidão de CFD [4, 13].

O processo de solução se inicia por meio da redução de dimensionalidade do problema pelo uso de Redes Neurais Convolucionais (ConvNet) [14]. O modelo possui

três partes, a primeira que aprende a transformar a simulação original em uma representação dimensionalmente reduzida dela; a segunda que é responsável pelo aprendizado da evolução temporal nesse espaço de dimensão reduzida; e a terceira, que faz a transformação inversa da representação da simulação para a dimensão original.

Para o aprendizado do processo pelas redes neurais é fundamental a inserção de uma grande quantidade de dados de ótima qualidade. Dessa forma, o método do Reticulado de Boltzmann (*Lattice Boltzmann Method (LBM)*), que é um método usado para simulação de fluido, derivado das equações de Boltzmann, e originado do *Lattice Gas Automata* [15], foi usado para geração do banco de dados de escoamentos turbulentos para alimentar as redes neurais.

1.3 **Descrição da monografia**

O restante do trabalho é dividido em mais três partes: o Capítulo 2 contém a fundamentação teórica necessária para compreensão das técnicas desenvolvidas e aplicadas a respeito do LBM e RN; o Capítulo 3, apresenta a configuração das simulações realizadas com o LBM e descreve o modelo criado e as configurações usadas; e o Capítulo 4, que apresenta e analisa os resultados obtidos pela RN, através da comparação com os dados obtidos pelo simulador, gerados para o teste. As considerações finais são feitas em CONCLUSÃO, junto com propostas de melhorias para o projeto. O APÊNDICE A apresenta detalhes mais técnicos e especificações da arquitetura, além da parametrização da RN.

2 FUNDAMENTAÇÃO TEÓRICA

Para se treinar uma rede neural é necessário que se tenha dados do processo que deseja-se aprender, nesse caso, as simulações da dinâmica de fluido. Para tal, foi escolhido o LBM em 2D para gerar as simulações usadas.

2.1 Método do reticulado de Boltzmann

O LBM é uma alternativa para a dinâmica dos fluidos computacional tradicional, usado para a simulação de fluidos, e que, ao invés de resolver as equações de Navier-Stokes, resolve as equações de Boltzmann, em suas formas discretas, para simular o fluxo de um fluido Newtoniano [15].

O modelo é uma abordagem estatística, e contém uma função de distribuição de densidade $f(\mathbf{x}, \mathbf{p}, t)$, que depende da posição $\mathbf{x}$ e momento $\mathbf{p}$ de cada partícula individual num determinado instante de tempo t. Esta função representa o estado exato do sistema, e a partir dele é possível calcular o desenvolvimento temporal deste.

Como mostrado pela Figura 1 [16], cada índice da função f pode ser interpretado como a frequência da distribuição da densidade microscópica do fluido em cada direção discreta do modelopodendo a densidade macroscópica ρ ser definida como o somatório de toda essa distribuição em cada posição determinada:

$$\rho = \sum_{a=0}^{8} f_a$$

Já a velocidade $\mathbf{v}$, é a média das velocidades microscópicas $\mathbf{e_a}$, multiplicadas pela respectiva densidade f_a:

$$\mathbf{v} = \frac{1}{\rho} \sum_{a=0}^{8} f_a \mathbf{e_a}$$

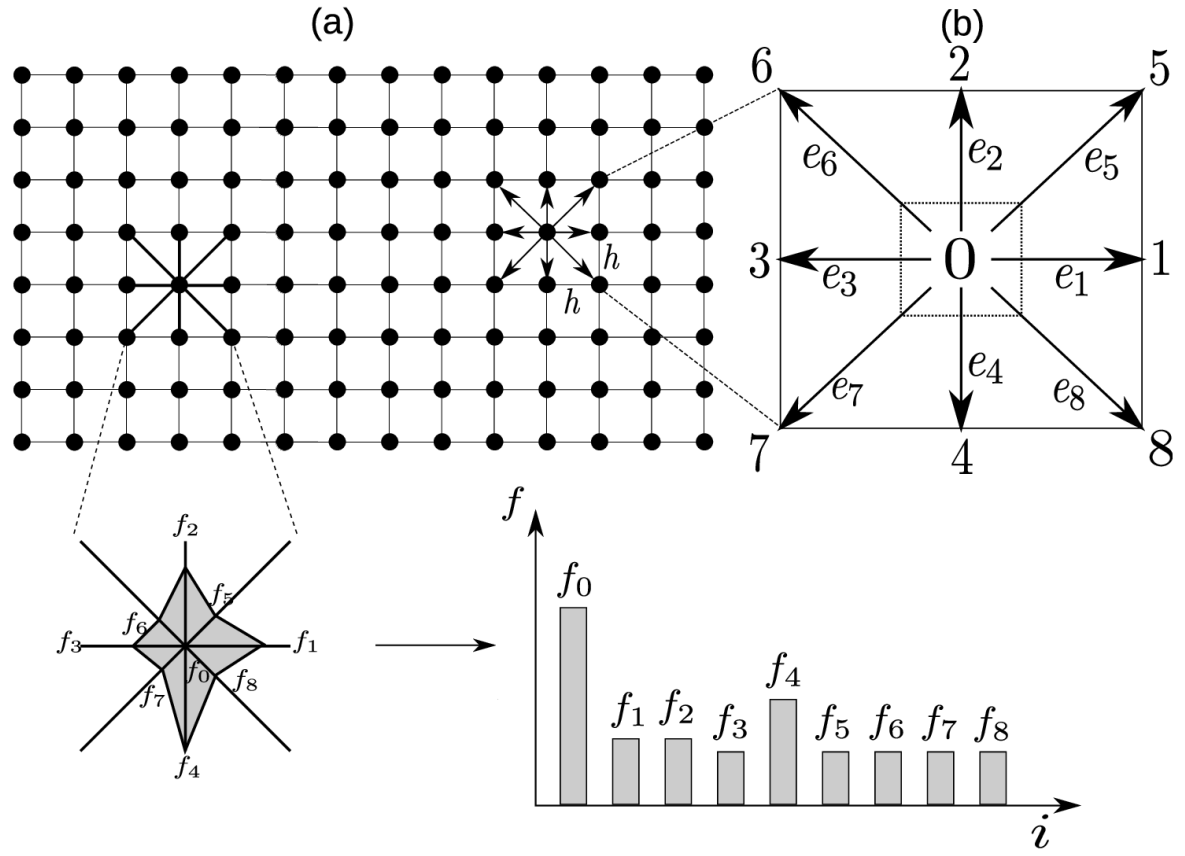


Figura 1 Discretização do método do reticulado Boltzmann: (a) reticulado padrão do método, e histograma da função de distribuição de densidade (f_a) de uma partícula; (b) direções do modelo

Fonte: KUMAR, 2015, f. 169.

Para o cálculo do desenvolvimento temporal da função f, o algoritmo é dividido em duas partes, que juntas são definidas como

$$\underbrace{f_a(\mathbf{x}+\mathbf{e_a}\Delta t, t+\Delta t)}_{\text{Propagação}} = f_a(\mathbf{x},t) - \underbrace{\frac{f_a(\mathbf{x},t)-f_a^{eq}(\mathbf{x},t)}{\tau}}_{\text{Colisão}}$$

E individualmente, podem ser mais bem descritas assim:

- Colisão

  A colisão retrata o desenvolvimento da função f ao longo do tempo, e representa um modelo de colisão das partículas do fluido, ou seja, como elas interagem entre si. Neste trabalho, o modelo de colisão usado foi o de Bhatnagar–Gross–Krook (BGK) [17], que é uma escolha bem robusta para simulações de escoamento monofásico, e

que é definido como:

$$\frac{\partial f_a}{\partial t} = \mathbf{e}_a \cdot \nabla f_a = \frac{f_a - f_a^{eq}}{\tau} \qquad a = (0, ..., 8)$$

Onde f_a são as funções de distribuição de densidade de fluido; $\mathbf{e_a}$ são as direções discretas possíveis; τ é o tempo de relaxação, um parâmetro adimensional relacionado a viscosidade cinemática ν:

$$\nu = c_s^2(\tau - 0.5\delta t)$$

f_a^{eq} é a função de distribuição em equilíbrio, é definida como:

$$f_a^{eq} = \rho\,\omega_a\left[1 + \frac{\mathbf{e}_a \cdot \mathbf{v}}{c_s^2} + \frac{(\mathbf{e}_a \cdot \mathbf{v})^2}{2c_s^4} - \frac{\mathbf{v}^2}{2c_s^2}\right]$$

Onde ρ é a densidade do fluido, $\mathbf{v}$ é a velocidade, $c_s = \frac{c}{\sqrt{3}}$ é a velocidade do som, em unidades do reticulado, $c = \frac{\partial x}{\partial t} = 1$ é a constante do reticulado; os fatores de ponderamento w_a são definidos como:

$$w_a = \begin{cases} 4/9, & a = 0 \\ 1/9, & a = 1, 2, 3, 4 \\ 1/36, & a = 5, 6, 7, 8 \end{cases}$$

Como mostra a Figura 2, as funções de distribuição são recalculadas de acordo com a equação de equilíbrio, porém a velocidade e densidade macroscópica permanecem inalteradas.

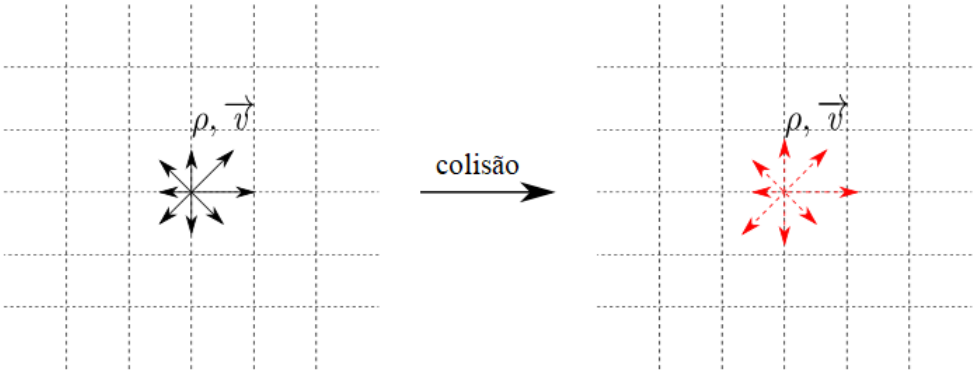


Figura 2 Ilustração, em um único nodo, do passo de Colisão em um modelo.

Fonte: KUMAR, 2015

- Propagação

  O passo de propagação descreve o espalhamento das partículas de uma célula em células vizinhas, e é definido matematicamente como:

$$f_a(\mathbf{x} + \mathbf{e_a}\Delta t, t + \Delta t) = \tilde{f}_a(\mathbf{x}, t)$$

  Onde f_a e $\tilde{f}_a$ são respectivamente os estados pré e pós colisão da função de distribuição, que dependem da posição $\mathbf{x}$ e do tempo t. A distribuição permanece com magnitude constante, porém é propagada para os nodos vizinhos de acordo com sua direção.

Figura 3 Ilustração, em um único nodo, do passo de propagação.
Fonte: KUMAR, 2015

2.1.1 Condições de contorno

Para o cálculo de qualquer resultado significativo, é necessário definir condições de contorno na simulação, ou seja, como o fluido se comporta nas extremidades dela.

As condições usadas, foram as de paredes estáticas, conhecida como *bounceback boundaries*, para as partes de cima e baixo da simulação, e de Neumann [18] para o fluxo de entrada e saída, a esquerda e a direita, respectivamente.

Como mostra a Figura 4 , [19], no passo de propagação a distribuição de densidade é propagada à parede. Para se manter a velocidade desta nula, a distribuição é invertida em direção, e é refletida novamente de volta para a célula que a refletiu originalmente.

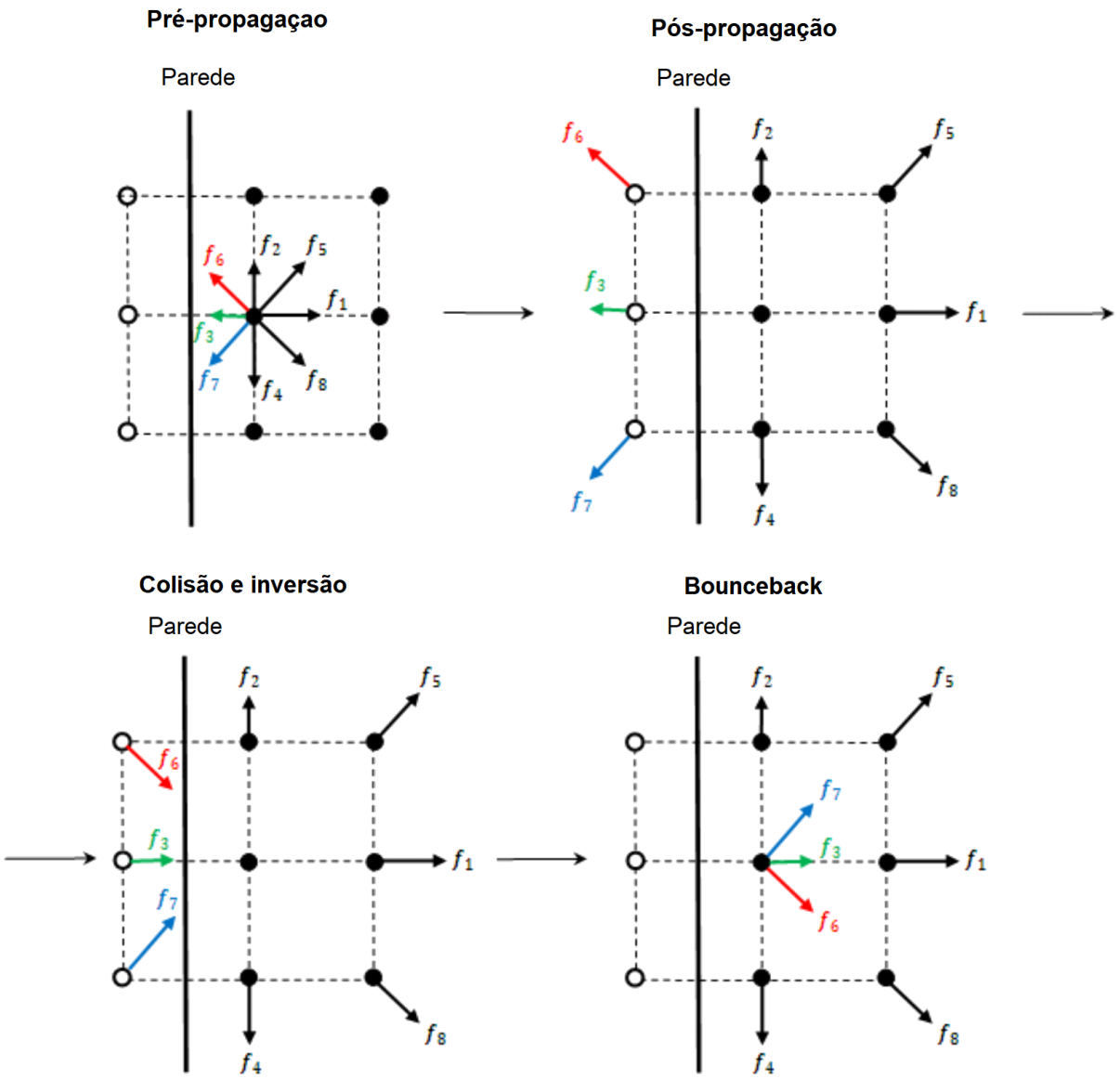


Figura 4 Ilustração da condição de não escorregamento no LBM.
Fonte: BAO, MESKAS; 2011

Já as condições de Neumann, são feitas através da escolha de uma velocidade **v** desejada, e a consequente computação da função f_a^{eq} equivalente, que é salva e repetidamente sobrepõe os valores de f_a nas posições de fronteira em cada passo de tempo.

2.2 Redes Neurais

Redes Neurais Artificiais são métodos de aprendizado computacional inspiradas no funcionamento do cérebro humano. São organizadas em camadas de processadores denominados de neurônios artificiais, que realizam o processamento e a transmissão de

informações entre as sinapses que conectam os neurônios.

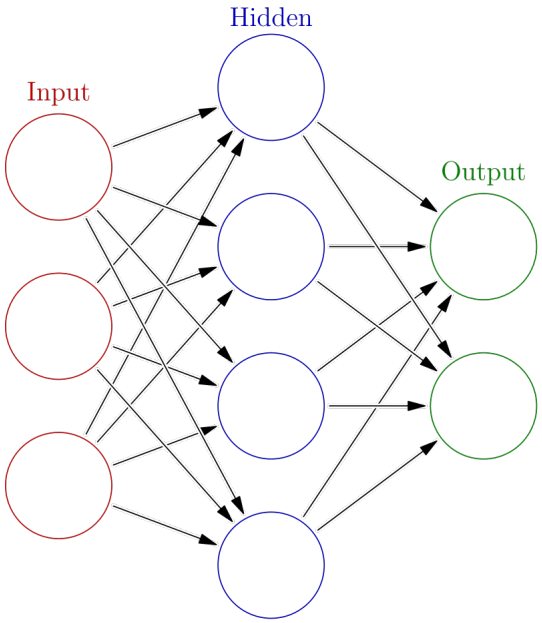
O neurônio que é estrutura básica das redes neurais é o neurônio artificial, e simula as funções do seu equivalente biológico. Já as conexões entre os neurônios são modeladas como parâmetros a serem aprendidos durante o processo de treinamento, e são denominadas pesos. Em cada neurônio é computada uma função cujo argumento é a soma do produto entre os valores computados pelos neurônios das camadas anteriores e o respectivo peso, ilustrado pela Figura 5, e definida como:

$$a = \sum_{i=1}^{n} \omega_i x_i \tag{2.1}$$

Esse cálculo, porém, apresenta um resultado linear em função dos valores de entrada. Para a computação de padrões mais complexos, é necessário aplicar uma função não linear nesses somatório, conhecida como função de ativação. Dentre as funções mais usadas, estão a $tanh(a)$ ou a *rectified linear unit*, ReLU [20], definida como:

$$f(a) = max(0, a) \tag{2.2}$$

Cada nodo representado na Figura 5, [21], possui um valor, e cada seta representa um peso que multiplica os valores do nodo de origem, e são somados no nodo de chegada.



Fonte: GLOSSER, 2013

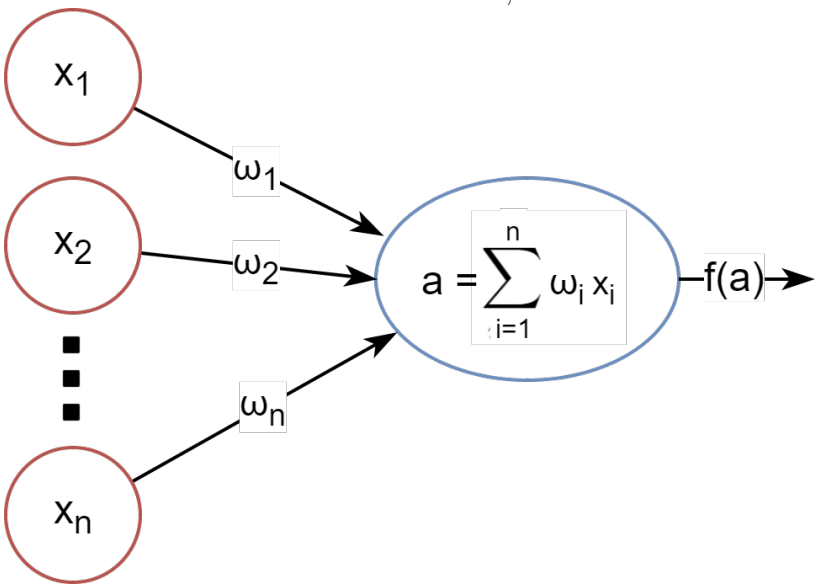


Figura 5 Diagrama de rede neural e representação das operações realizadas nela, que ocorrem em cada nodo da RN

Dado o *Input* x, o processo de aprendizado e treinamento da RN, consiste no ajuste dos valores dos pesos ω, visando produzir o resultado y esperado. Para tal processo de ajustamento dos pesos, é usado um algoritmo chamado de *backpropagation*.

2.2.1 Backpropagation

Os pesos sinápticos da rede são inicializados com valores aleatórios, possibilitando calcular, dado o *Input* x, qual o *Output* $\hat{y}$ que ela produz. Então, dado o resultado real y, denominado rótulo ou *target*, é possível calcular o erro produzido pela rede, chamado de custo J, através do uso de uma medida escolhida de precisão, como o Erro Quadrático Médio (EQM) (2.3) [22].

$$J = \frac{1}{m}\sum_{m}(y_m - \hat{y}_m)^2 \tag{2.3}$$

Após o cálculo do custo, é desejado a minimização deste, de modo que o resultado da rede seja mais próximo do real esperado. Como o resultado produzido pela rede depende de seus pesos ω, e J depende de y que é o *Output* da rede, é possível calcular a derivada do custo em função dos parâmetros $\frac{\partial J}{\partial \omega}$, e, com o intuito de reduzir o custo, atualizar os pesos ω, através da multiplicação da derivada por um parâmetro chamado de *learning rate* α, procedimento conhecido como método do gradiente [23] (2.4). O método é aplicado para todas as amostras do *dataset* em várias iterações, até que o custo J pare de diminuir, como ilustra a Figura 6.

$$\omega := \omega - \alpha\frac{\partial J}{\partial \omega} \tag{2.4}$$

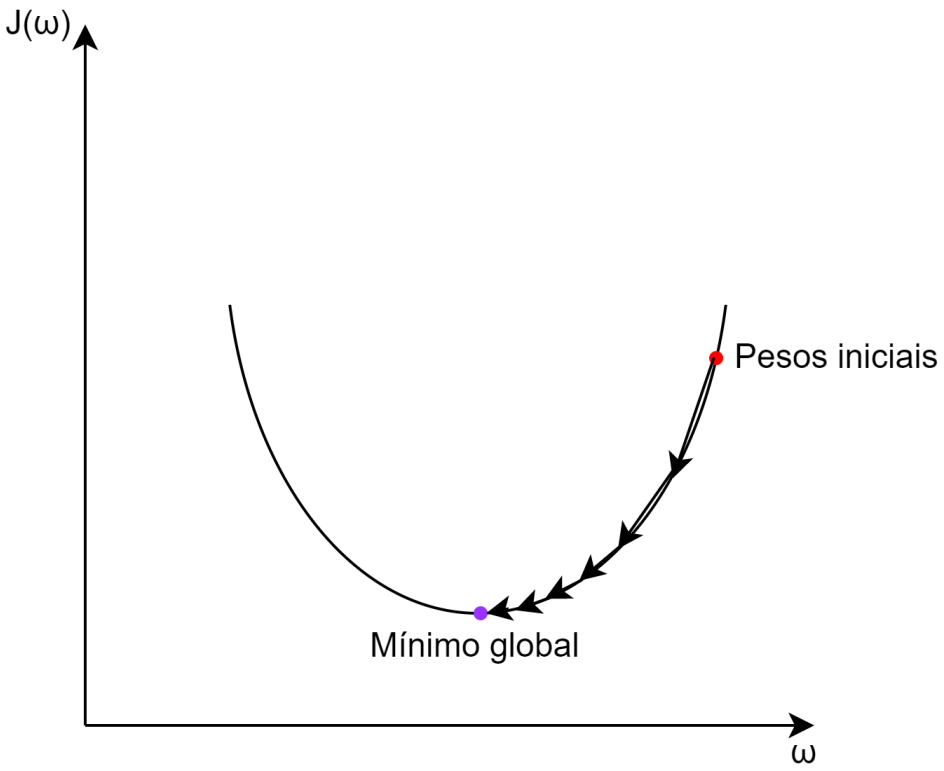


Figura 6 Representação simplificada do método do gradiente em função de apenas um parâmetro ω. Cada seta indica uma iteração do método, e sua convergência ao mínimo.

Os valor de α deve ser bem escolhido: não pode ser muito grande, para evitar que os pesos ultrapassem o mínimo e divirjam, nem deve ser muito pequeno, para evitar que a convergência seja demorada e leve várias iterações. Geralmente é um valor entre 0 e 1.

Esse processo já foi melhorado através do uso de técnicas mais complexas no cálculo de α, como o *Adaptative Moment Estimation* (Adam) [24], e que foi usado nesse projeto. Ele usa técnicas como o cálculo da média móvel do gradiente, e cálculo de momento, para tornar mais rápida e robusta a convergência dos valores de peso.

As Rede Neural apresentam muitas vantagens em sua aplicação para o problema. Assim, nesse trabalho foi usado um tipo mais novo e específico de Rede Neural bastante apropriado para problemas que envolvam imagens: as Redes Convolucionais. [25]

2.3 **Redes Neurais Convolucionais**

As Redes Neurais Convolucionais (ConvNets) são uma classe especial de Redes Neurais, principalmente usadas na área de processamento de imagens, uma área que envolve o tratamento de imagens para diversos fins, já que nelas as correlações espaciais

de um *pixel* e seus vizinhos são extraídas através das operações de convolução. Na área de dinâmica dos fluidos, a observação dessa correlação deve ser também muito forte, já que, em princípio, a velocidade de um ponto local do fluido influencia mais as velocidades dos pontos da vizinhança do que as de pontos distantes.

Nessa redes, dada uma imagem f, o *kernel* h, que tem a dimensão escolhida e valores inicializados aleatoriamente, realiza-se a operação discreta de convolução (2.5) sobre a imagem, conforme ilustrado pela Figura 7.

$$(f * h)(m, n) = \sum_j \sum_k h[j, k] f[m - j, n - k] \tag{2.5}$$

Conforme a Figura 7, [26], em cada localização, o produto entre o elemento do *kernel* (a) e os elementos da imagem (b) que ele sobrepõe é computado, e a soma dos resultados é a saída para aquela localização (c). Em uma rede neural, os valores do *kernel* são aprendidos durante o treinamento desta.

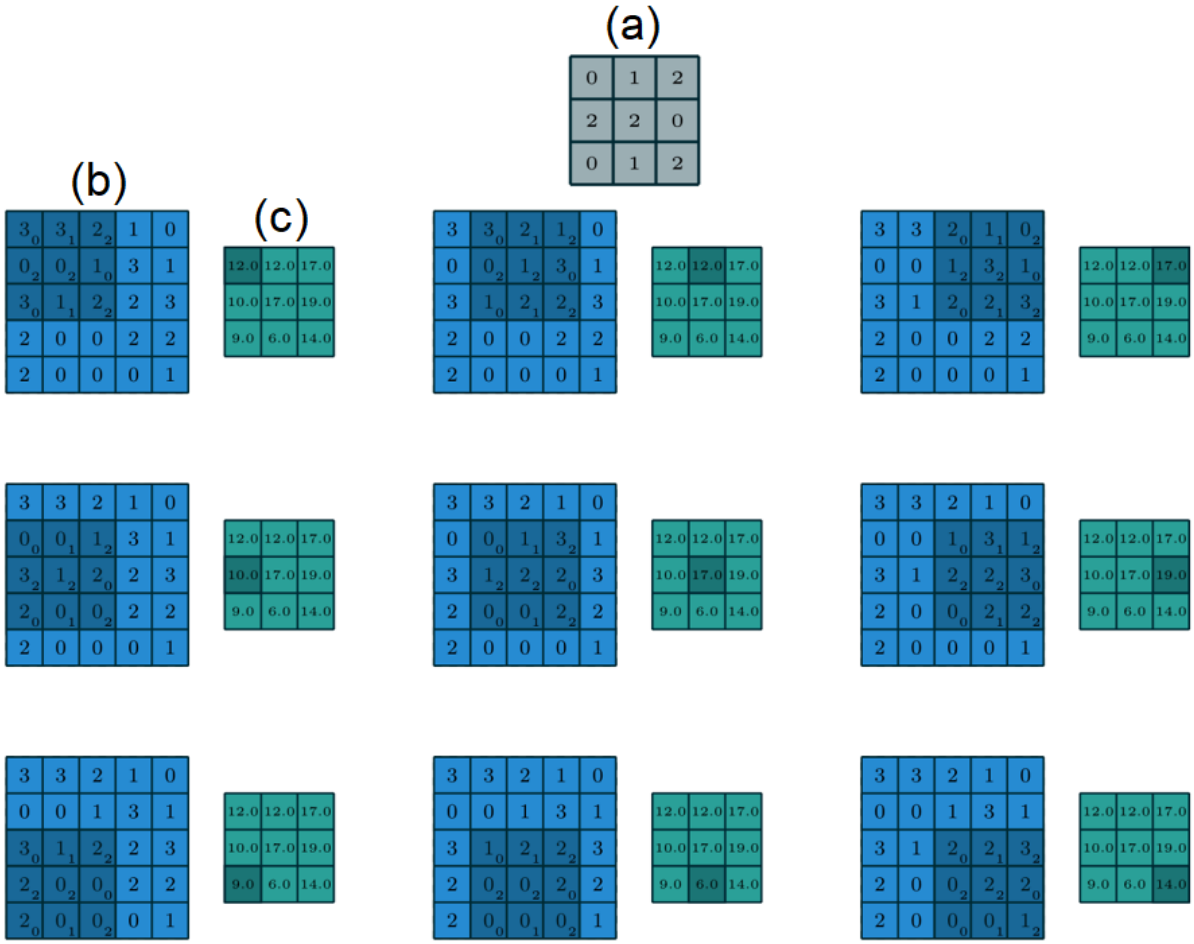


Figura 7 Exemplo da operação discreta de convolução.
Fonte: DUMOULIN, VISIN; 2016

2.3.1 Convolução Transposta

A convolução transposta, ou também chamada as vezes de deconvolução, foi proposta para ser a operação inversa a da convolução [26]. Conforme a Figura 8, com o propósito de se aumentar o tamanho da imagem de Input (a) da rede neural, é inserido uma margem, geralmente composta de elementos de valor zero, entre cada *pixel* do Input (b) e depois é feita a operação de convolução (c).

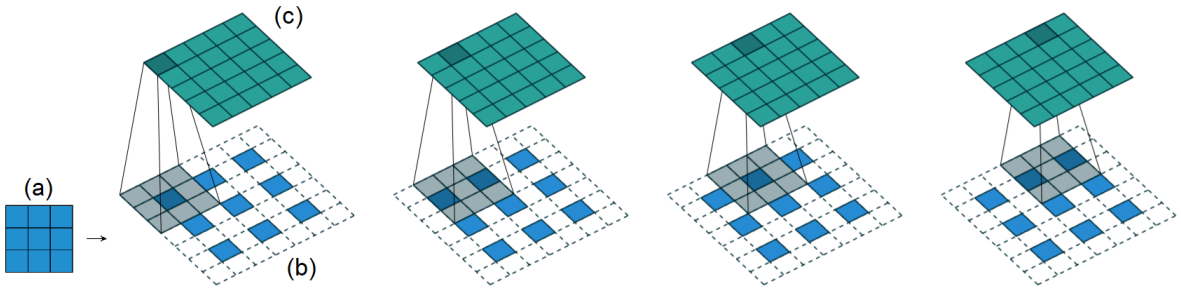


Figura 8 Exemplo da operação discreta de convolução transposta.

Fonte: DUMOULIN, VISIN; 2016

3 MÉTODO

3.1 Geração do dataset

Para gerar as simulações de fluido, foi usado o LBM D2Q9, ou seja, com 2 dimensões e 9 distribuições de velocidade, em um *grid* de 256x256 pixels, com condições de contorno já descritas. As velocidades de entrada e saída determinadas foram de 0,08 unidades do reticulado (lu) para a direita, que é uma velocidade relativamente alta, a viscosidade é de $\nu = 0,01\ lu$. Esses fatores foram escolhidos de modo a obter um escoamento turbulento.

Com o objetivo de criar um treinamento com situações básicas que possam servir de base para outras mais complexas, os obstáculos contidos na simulação foram aleatoriamente gerados, sendo circulares e variando em quantidade, de 2 a 8 objetos, tamanho, de 3 a 20 pixels de raio, e coordenada, variando a posição do centro de [10, 246], tanto vertical, quanto horizontalmente, como exemplifica a Figura 9. Esses obstáculos são gerados como uma matriz booleana, contendo 1 em cada posição que tem um obstáculo presente, e 0 nas posições sem nenhum deles.

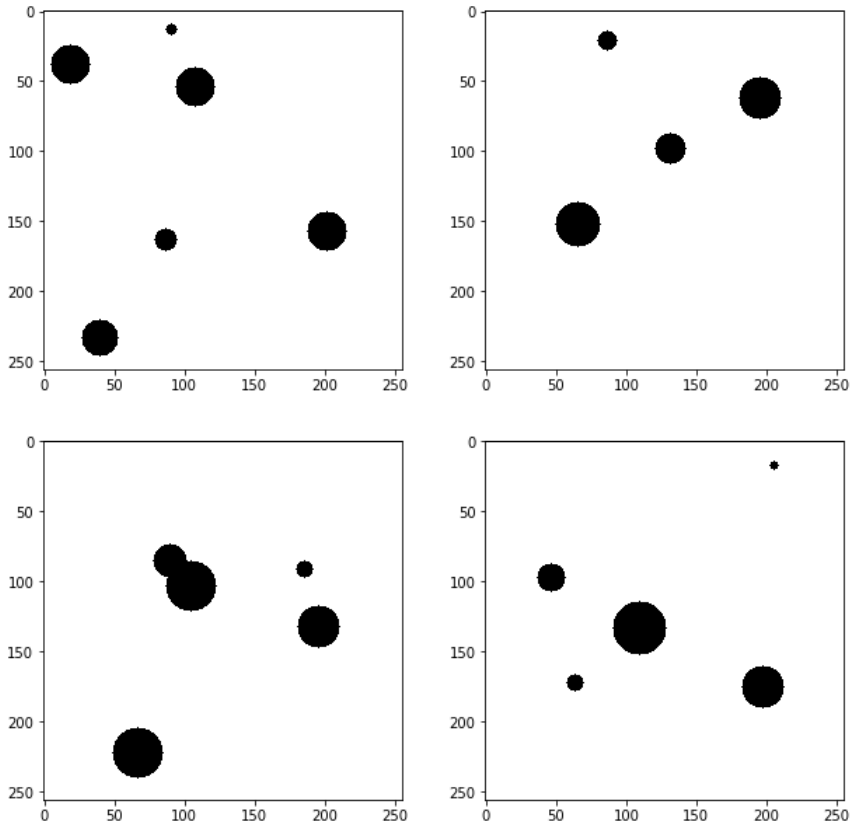


Figura 9 Exemplos de objetos gerados

Essa geometria foi escolhida por sua simplicidade, porém a junção entre um ou mais círculos podem ainda produzir efeitos mais complexos de turbulência, que servem como exemplo para a rede em casos de geometria mais complexas.

Dadas essas condições, e velocidade inicial nula nas outras células da simulação, foram geradas 45 simulações de 100 passos de tempo cada, contendo os valores das 9 direções da função de distribuição de velocidade f_a do LBM, sendo que entre cada passo de tempo, as etapas de colisão e propagação foram executadas 120 vezes. 40 dessas simulações foram usadas no treinamento do modelo de RN, e 5 delas foram usados como validação dos resultados de treinamento.

3.2 **Arquitetura da RN**

Conforme a Figura 10, o modelo de RN usado é dividido em três partes principais: uma, o *encoder* (a), que usa camadas de convolução para a redução dimensional da simulação; outra, a de evolução temporal (b), para o cálculo do próximo passo de tempo nesse domínio de dimensão reduzida; e outra, o *decoder* (c), que usa as camadas de convolução transposta para restaurar a representação de dimensão reduzida a dimensão original. Essa arquitetura conhecida como encoder-decoder já foi usada de forma semelhante em problemas de simulação de sistemas dinâmicos, como em [4, 6, 13, 27].

No modelo, dada uma simulação desejada para se computar, a representação dos objetos presentes nela é processada pela rede de Objeto *Encoder*, e gera 2 resultados em um domínio de dimensão reduzida, o Obj Mult e o Obj Soma, que são reservados para o uso ao longo do desenvolvimento da dinâmica da simulação.

Depois, dado um *frame* inicial gerado no LBM como referência, a representação reduzida dele é computada através do uso da rede (a) f_a *Encoder*. Então os 2 resultados Obj Mult e Obj Soma são, respectivamente e em ordem, multiplicados e somados a essa representação. Esse passo reforça a permanência temporal dos objetos na simulação, como usado também em outras tarefas, e em [13, 28].

Esse primeiro *frame* pode ser recuperado através do uso do f_a *Decoder*, ou o processo de desenvolvimento temporal da dinâmica pode ser continuado através do uso da rede (b). Essa rede tem o objetivo de realizar a transformação de $f_a(t)$ para $f_a(t+1)$, tarefa essa não trivial, já que é realizada nesse domínio reduzido, que possivelmente não tem correlação direta com a dinâmica de fluidos vista e entendida no domínio original.

O conjunto de desenvolvimento da simulação pode ser repetido indefinidamente para se obter o desenvolvimento do escoamento. E, para se obter simulações em que a evolução do escoamento não é de interesse, o *decoder* pode ser usado apenas quando desejado a obtenção do *frame* no domínio original.

Cada parte da RN foi descrita em detalhe, juntamente com parâmetros e aprofundamentos sobre a arquitetura da rede, em APÊNDICE A, e são apresentadas de maneira geral pela Figura 10.

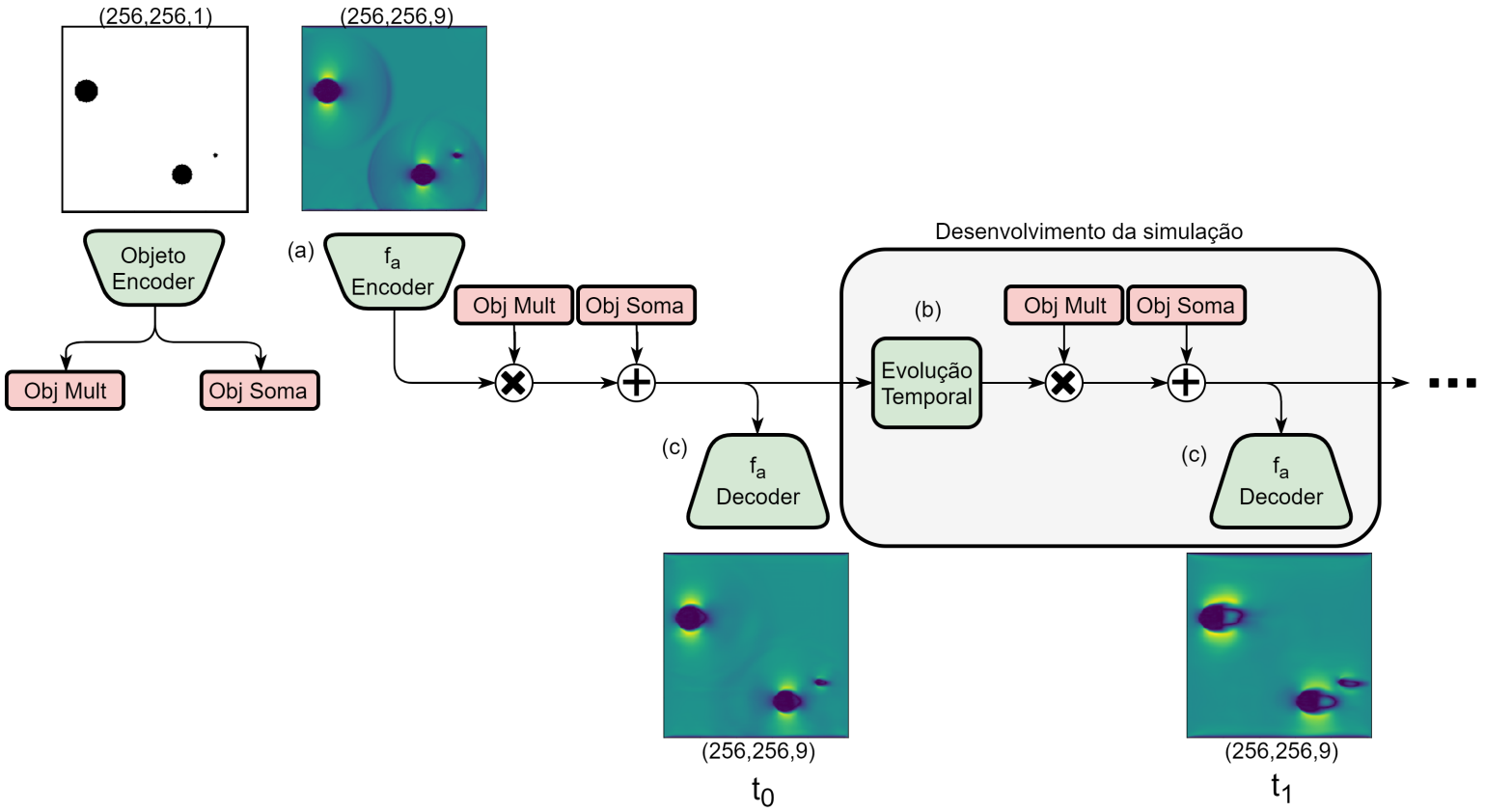


Figura 10 Visão geral da arquitetura da rede. Os números entre parênteses representam a dimensão dos elementos referentes.

4 RESULTADOS E DISCUSSÃO

Treinado o modelo usando os parâmetros descritos em APÊNDICE A, foram geradas simulações com ele, para avaliar os resultados obtidos. Para tal, foram usados arranjos de objetos que não foram usados durante o treinamento, para que se possa avaliar a capacidade da rede de produzir bons resultados em novas situações.

Foram usados arranjos como Figura 11, que foram criados usando posições aleatórias para os objetos; e outros mais diversos, como Figura 12, Figura 15, para avaliar a capacidade de generalização da rede em situações mais diversas, e então foi calculada a velocidade absoluta do fluido, que é representada nas imagens em unidade chamada Unidades do Reticulado.

Para avaliar a exatidão do modelo produzido, foi necessário usar uma métrica diferente da usada no treinamento da RN, o Erro Quadrático Médio, já que este não tem relação direta com as propriedades físicas do escoamento, e apresenta discrepância devido ao comportamento caótico do escoamento. E, como a métrica de avaliação pode variar com a aplicação desejada, assim como [1], foi usada uma métrica com significado mais geral, que é o cálculo do divergente do campo de velocidade do fluido (4.1) a cada passo de tempo, Figura 11. Nos gráficos, são demonstrados pelas linhas a média desses valores, e suas amplitudes, pelos valores sombreados.

$$\boldsymbol{\nabla} \cdot \mathbf{v}(x, y) = \frac{\delta \mathbf{v}(x, y)}{\delta x} + \frac{\delta \mathbf{v}(x, y)}{\delta y} \tag{4.1}$$

A Figura 11 apresenta os resultados para uma das simulações geradas junto com as de treinamento da rede neural, seguindo as mesmas regras desta, porém não usada no treino em si. É possível notar semelhança entre as simulações produzidas pelo LBM e pela RN, e um baixo erro na divergência, que mostra uma boa generalização para situações como as usadas no treinamento. As funções de distribuição dessa simulação apresentam um MSE total de 0.7319260698897175.

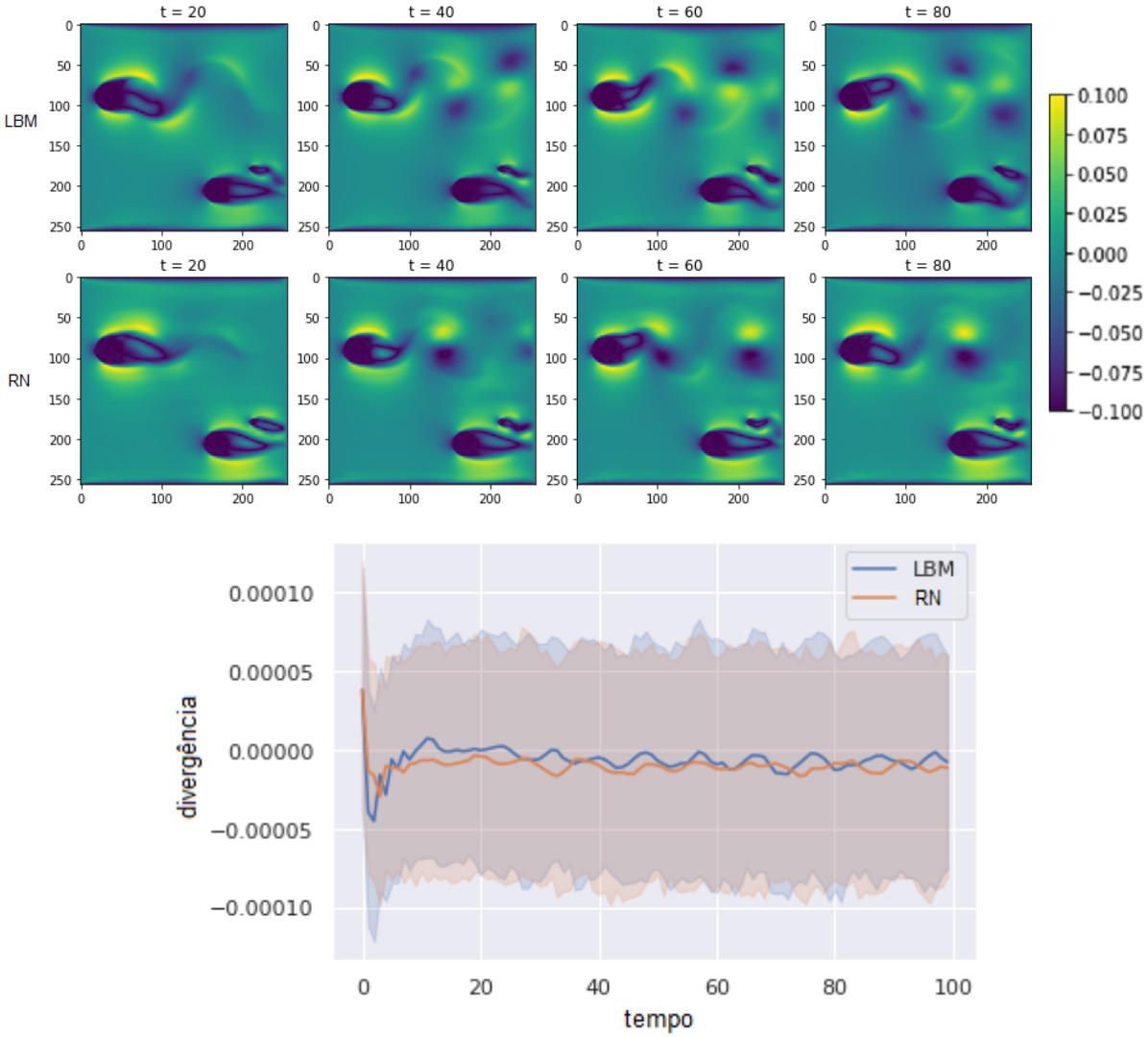


Figura 11 Exemplo de resultado obtido no conjunto de teste da Rede Neural.

A Figura 12 apresenta um teste mais específico, foi criada uma simulação com um único objeto de raio 20 pixels centrada no eixo y. Essa é uma simulação mais simples, mas que extrapola o conjunto de treinamento, além de permitir uma visualização melhor dos efeitos de turbulência isoladamente. É possível visualizar que não há uma grande diferença entre os valores de divergência das simulações. As funções de distribuição dessa simulação apresentam um MSE total de 0.9218580352696122.

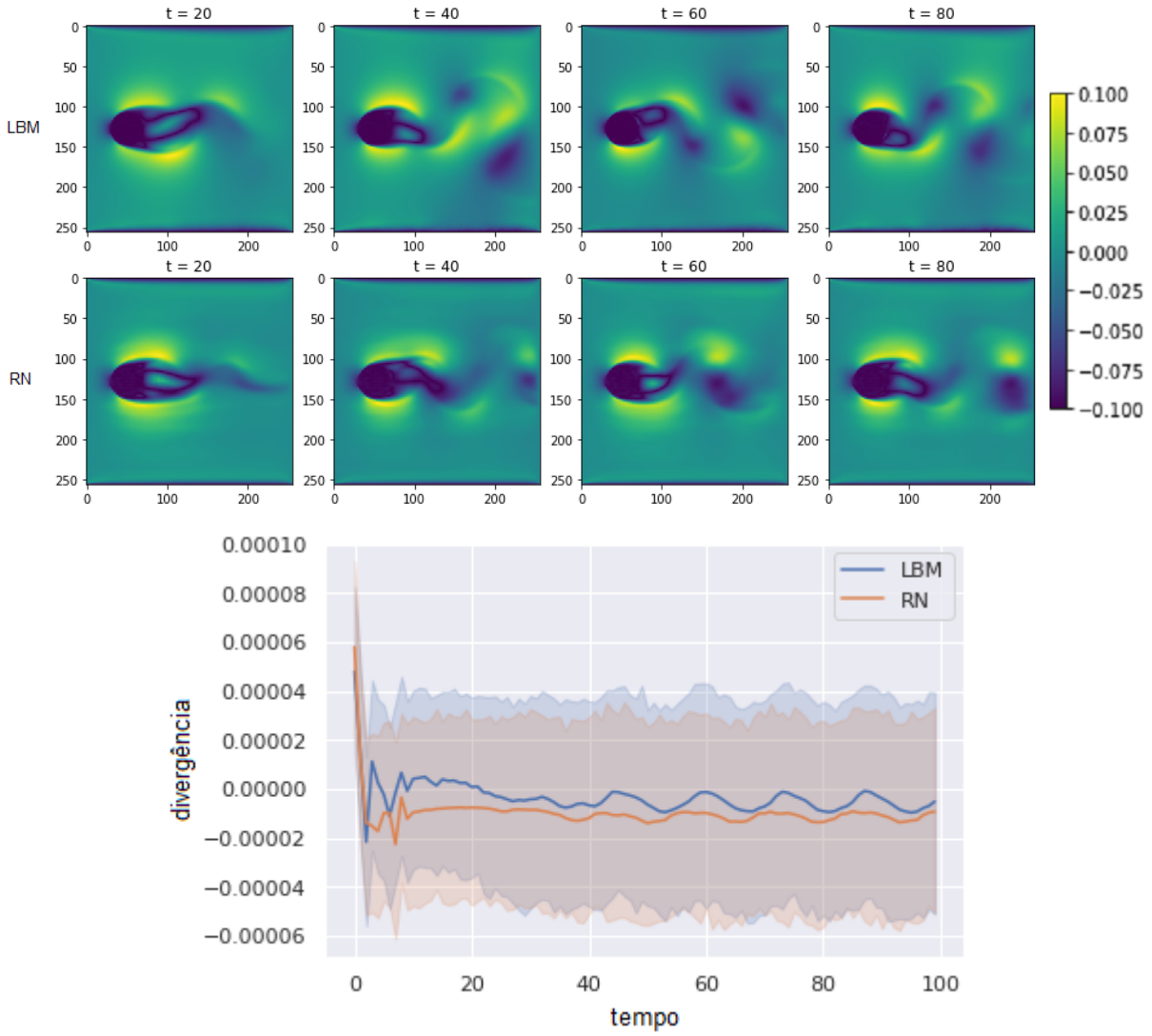


Figura 12 Teste de generalização com um único objeto, e sua respectiva divergência.

Como mostra a Figura 13, é possível também visualizar a evolução da divergência e vorticidade na simulação, revelando semelhanças entre elas, apesar de nunca terem sido explicitamente calculadas no modelo.

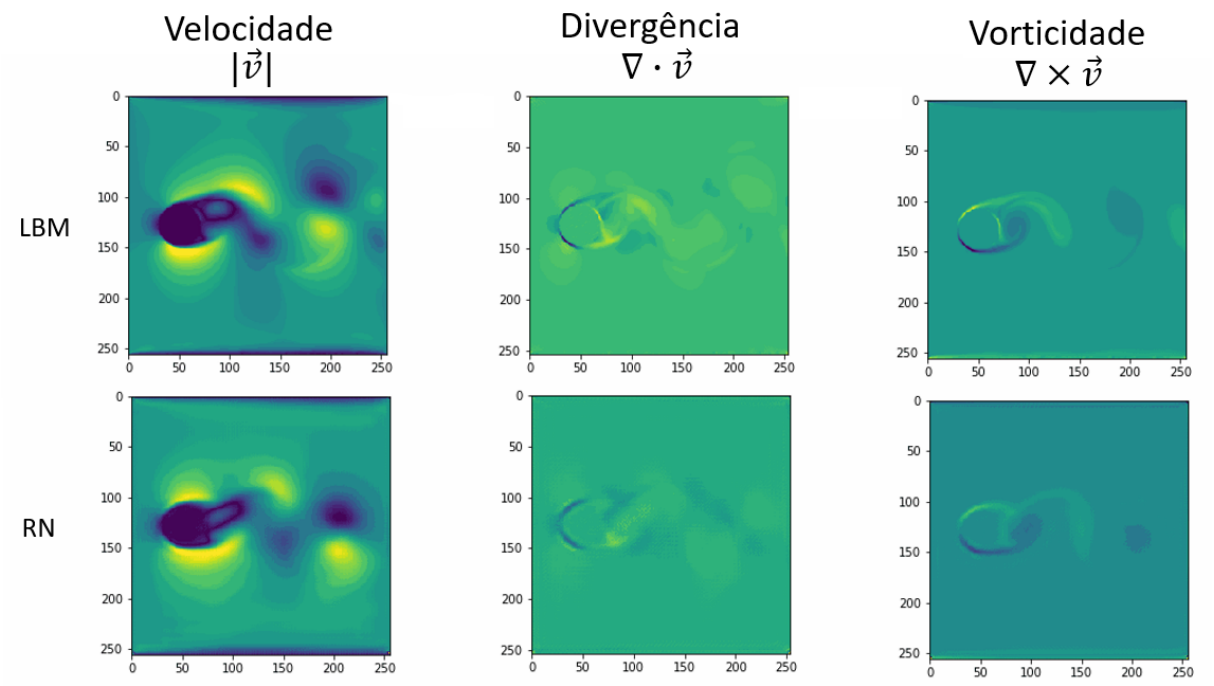


Figura 13 Comparação visual da velocidade, divergência e vorticidade das simulações.

Como a Figura 14 mostra, a análise do MSE entre as simulações da RN e LBM mostra um aumento contínuo do erro de acordo com o tempo, certamente devido a caoticidade do problema. O MSE olha apenas os valores absolutos dos temos da distribuição f_a, e após qualquer pequena diferença, esses erros vão se acumulando, e não oferecem uma interpretação da física do que ocorre na simulação.

Em comparação, se adicionarmos um valor aleatório ε, percentualmente com valores de até, respectivamente, 10, 25 e 35% dos valores de f_a original, temos uma diferença no MSE inicial em relação a simulação do LBM. Esse valor serve como referência para a comparação da evolução do erro da RN, e é possível ver que com o tempo o erro ultrapassa todos esses valores de base.

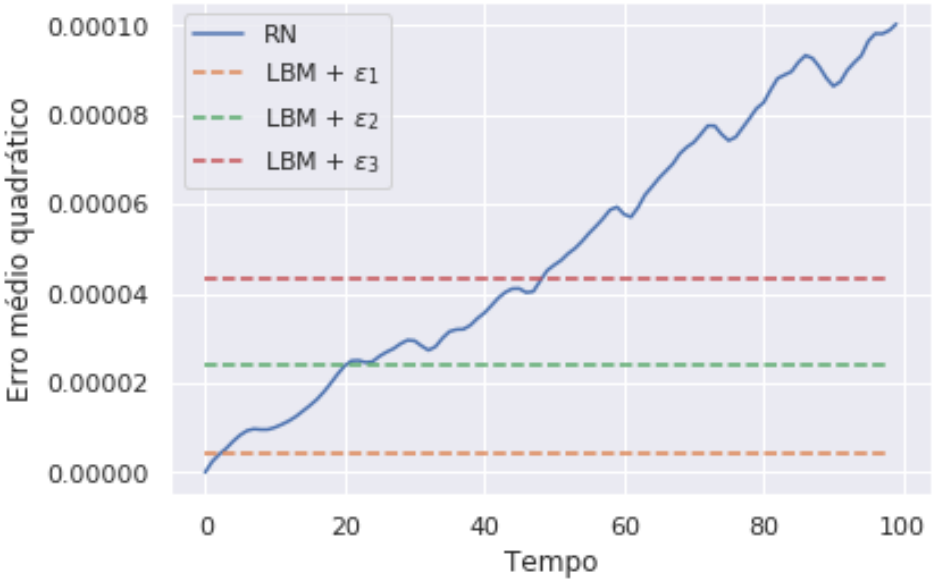


Figura 14 Evolução do erro médio quadrático entre as simulações em função do tempo, e valores de referência.

Já na simulação da Figura 15, foram criados 16 objetos de raio 3 de pixels distribuídos ao longo de $y = 50$. Esse escoamento pode representar um corte bidimensional de um fluxo por uma grade, porém sem as relações tridimensionais da turbulência. Os valores de divergência da RN oscilam bastante, mas mantém-se próximos ao do LBM, e aparentam se estabilizar com o decorrer da simulação. As funções de distribuição dessa simulação apresentam um MSE total de 0.08802398939879666, certamente menor devido a maior dispersão das turbulências.

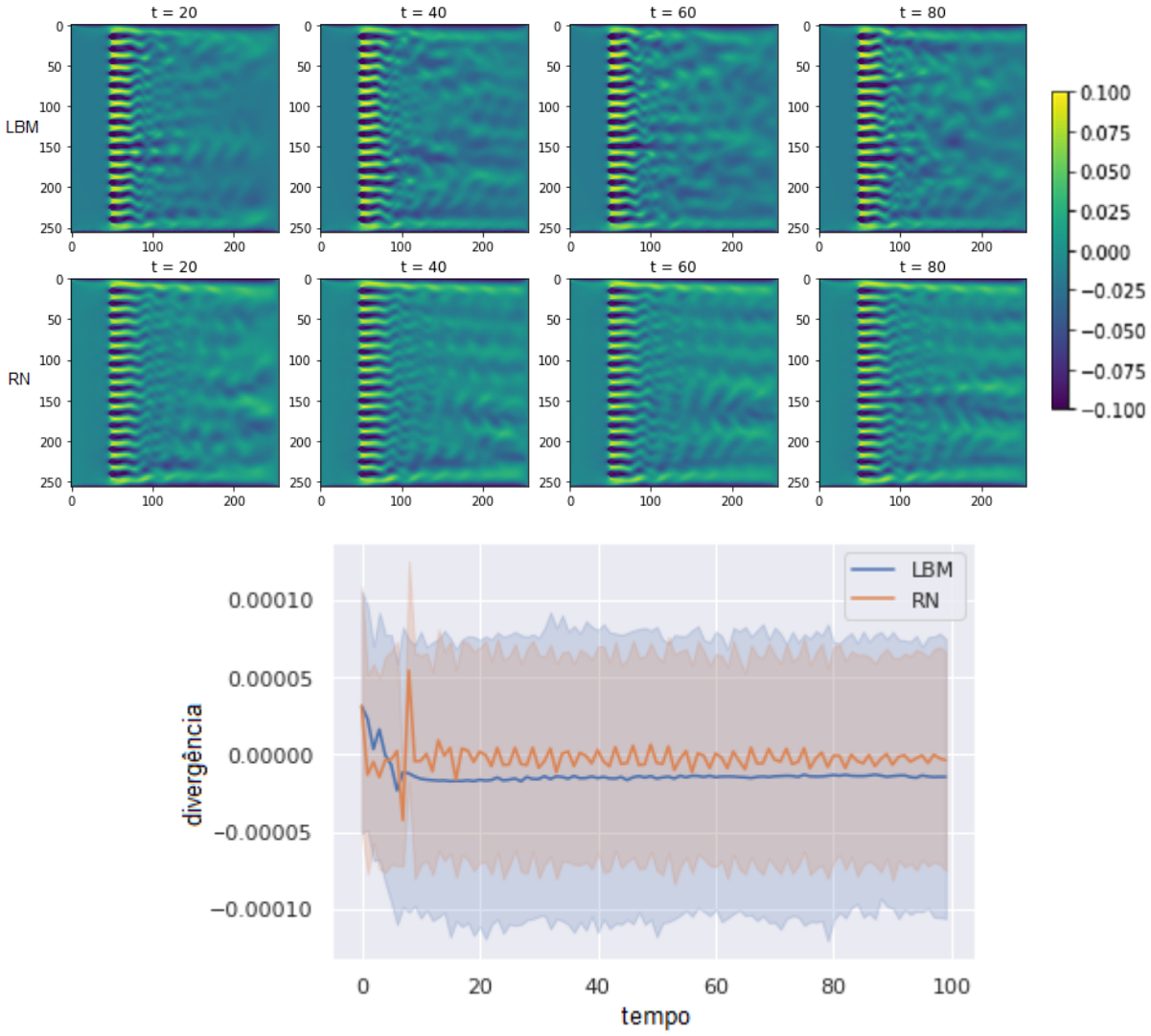


Figura 15 Teste de generalização com vários objetos, e sua respectiva divergência.

Para uma análise dos valores de velocidade, é possível pegar um corte em X=200, e analisando as médias de velocidades locais ao longo de t = [40,100] 4.2, é possível se chegar nos gráficos da Figura 16, onde a primeira imagem mostra a média de distribuição de velocidades ao longo do corte, e a segunda mostra a análise de Fourier dessas distribuição. A transformada retorna um número complexo para cada um dos componentes da velocidade, e para a sua visualização, foi feita a multiplicação pelo conjugado desse número, e então tirado seu logaritmo, com os valores de maiores frequência deslocados para as extremidades.

$$\bar{x}(y,t) = \frac{1}{m}\sum_{t}^{m} x(y,t) \tag{4.2}$$

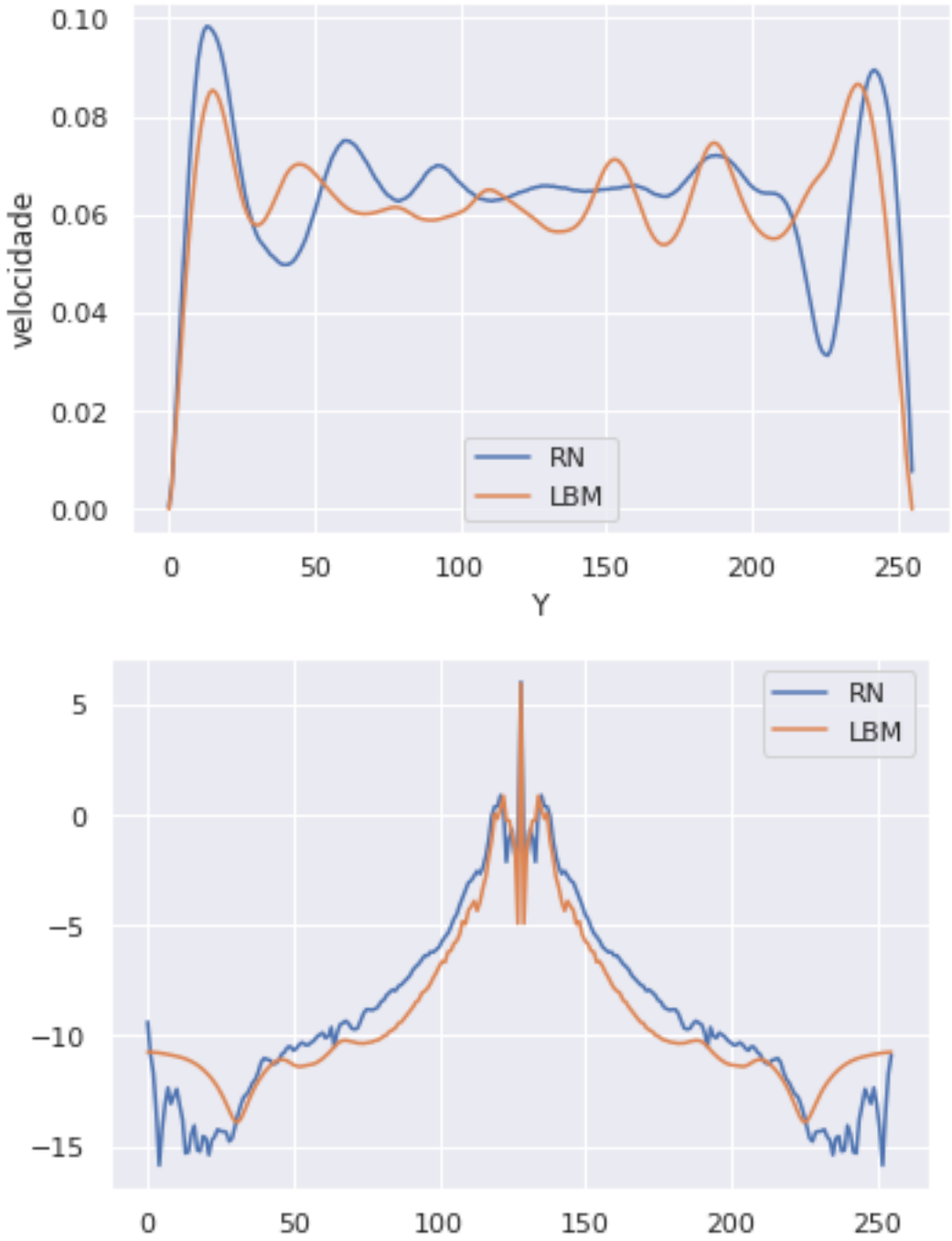


Figura 16 Valores de velocidade ao longo de um corte feito em X=200, e sua respectiva transformada de Fourier.

Essa análise mostra uma certa semelhança nas distribuições de velocidade; apesar de visivelmente elas aparecerem diferentes, a RN consegue criar valores próximos dos reais para a distribuição de velocidades do fluido.

Com esses três resultados, é possível se ter uma noção de como os resultados da rede se comportam após o treino. Em simulações mais diferenciadas do conjunto de treino,

a rede apresenta resultados de divergência um pouco mais distantes dos verdadeiros, mas como sugere o primeiro resultado do conjunto de teste, isso pode ser minimizado através de uma maior diversificação do treinamento para incluir casos mais parecidos com os que serão trabalhados.

A rede apresenta uma boa convergência temporal, ou seja, os resultados não possuem valores que produzem grandes divergências na simulação a ponto de gerar um erro que a corrompa. Qualitativamente as simulações apresentam boas características, como a presença de turbulências e vórtices que interagem entre si, produzindo resultados interessantes.

CONCLUSÃO

CFD é uma área de enorme interesse em diversos campos de aplicação, porém, seu elevado custo computacional pode ser um obstáculo para a utilização do método. O estudo de aplicações de soluções orientadas a dados, embora recente, tem mostrado excelentes resultados, e através do uso de técnicas de Aprendizado de Máquinas, torna-se possível a simulação de fluidos de maneira mais eficiente.

Os resultados apresentados pelo trabalho, mostram que é possível usar redes neurais para a obtenção de boas simulações de escoamentos turbulentos, e mostram também que a arquitetura encoder-decoder da rede neural é adequada para tarefa, e a redução dimensional aplicada por ela apresenta também uma possibilidade na diminuição do tempo levado para se gerar simulação.

Uma diversificação maior das simulações de treinamento, assim como um aumento nesse número de exemplos, podem auxiliar na capacidade de generalização da rede. Para tal, pode ser necessário também um aumento no número de parâmetros da rede, e isso deve ser testado experimentalmente.

Dado a complexidade do problema, ainda é possível realizar melhorias no método. Ambos os campos de redes neurais e dinâmica dos fluidos computacional tem apresentado grandes avanços nos últimos anos, e o uso de novas técnicas dessas áreas podem melhorar ainda mais a qualidade das simulações.

REFERÊNCIAS

[1] Tompson, J.; Schlachter, K.; Sprechmann, P.; Perlin, K. Accelerating Eulerian Fluid Simulation With Convolutional Networks. *arXiv e-prints*, p. arXiv:1607.03597, Jul 2016.

[2] White, C.; Ushizima, D.; Farhat, C. Fast Neural Network Predictions from Constrained Aerodynamics Datasets. *arXiv e-prints*, p. arXiv:1902.00091, Jan 2019.

[3] CHU, M.; THUEREY, N. Data-driven synthesis of smoke flows with cnn-based feature descriptors. *ACM Transactions on Graphics*, Association for Computing Machinery (ACM), v. 36, n. 4, p. 1–14, Jul 2017. ISSN 0730-0301. Disponível em: <http://dx.doi.org/10.1145/3072959.3073643>.

[4] Wiewel, S.; Becher, M.; Thuerey, N. Latent-space Physics: Towards Learning the Temporal Evolution of Fluid Flow. *arXiv e-prints*, p. arXiv:1802.10123, Feb 2018.

[5] Zhang, W.; Zhu, L.; Liu, Y.; Kou, J. Machine learning methods for turbulence modeling in subsonic flows over airfoils. *arXiv e-prints*, p. arXiv:1806.05904, Jun 2018.

[6] MORTON, J.; WITHERDEN, F.; JAMESON, A.; KOCHENDERFER, M. *Deep Dynamical Modeling and Control of Unsteady Fluid Flows*. 2018.

[7] GUO, X.; LI, W.; IORIO, F. Convolutional neural networks for steady flow approximation. In: *KDD '16*. [S.l.: s.n.], 2016.

[8] WILCOX, D. C. *Turbulence modeling for CFD*. [S.l.]: DCW industries La Canada, CA, 1998.

[9] BAYSAL, O.; ELESHAKY, M. E. Aerodynamic design optimization using sensitivity analysis and computational fluid dynamics. *AIAA Journal*, v. 30, n. 3, p. 718–725, 1992.

[10] SANDERSE, B.; PIJL, S. van der; KOREN, B. Review of computational fluid dynamics for wind turbine wake aerodynamics. *Wind Energy*, v. 14, n. 7, p. 799–819, 2011.

[11] XU, G.; LUXBACHER, K. D.; RAGAB, S.; XU, J.; DING, X. Computational fluid dynamics applied to mining engineering: a review. *International Journal of Mining, Reclamation and Environment*, Taylor Francis, v. 31, n. 4, p. 251–275, 2017.

[12] LUCENA, R. M.; MANGIAVACCHI, N.; PONTES, J.; ANJOS, G.; MCGINTY, S. On the transport through polymer layer and porous arterial wall in drug-eluting stents. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, v. 40, n. 12, p. 572, Nov 2018. ISSN 1806-3691.

[13] Hennigh, O. Lat-Net: Compressing Lattice Boltzmann Flow Simulations using Deep Neural Networks. *arXiv e-prints*, p. arXiv:1705.09036, May 2017.

[14] KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: PEREIRA, F.; BURGES, C. J. C.; BOTTOU, L.; WEINBERGER, K. Q. (Ed.). *Advances in Neural Information Processing Systems 25*. [S.l.]: Curran Associates, Inc., 2012. p. 1097–1105.

[15] PERUMAL, D. A.; DASS, A. K. A review on the development of lattice boltzmann computation of macro fluid flows and heat transfer. *Alexandria Engineering Journal*, v. 54, n. 4, p. 955 – 971, 2015. ISSN 1110-0168.

[16] KUMAR, K. *Multi-scale multiphase modelling of granular flows*. 169,172 p. Tese (Doutorado), Jan 2015.

[17] BHATNAGAR, P. L.; GROSS, E. P.; KROOK, M. A model for collision processes in gases. i. small amplitude processes in charged and neutral one-component systems. *Phys. Rev.*, American Physical Society, v. 94, p. 511–525, May 1954.

[18] SUKOP, M. C.; THORNEJR., D. T. *Lattice Boltzmann Method, Fundamentals and Engineering Applications with Computer Codes*. [S.l.]: Springer, 2006. 39 - 54 p.

[19] BAO, Y.; MESKAS, J. *Lattice Boltzmann Method for Fluid Simulations*. 04 2014.

[20] AGARAP, A. F. *Deep Learning using Rectified Linear Units (ReLU)*. 2018.

[21] GLOSSER.CA. *Artificial neural network*. 2019. Disponível em: <http://web.archive.org/web/20191014071635/https://en.wikipedia.org/wiki/Artificial_neural_network>.

[22] LECUN, Y. A theoretical framework for back-propagation. 08 2001.

[23] TADDY, M. *Business Data Science: Combining Machine Learning and Economics to Optimize, Automate, and Accelerate Business Decisions*. [S.l.: s.n.], 2019. 303-307 p.

[24] KINGMA, D. P.; BA, J. Adam: A Method for Stochastic Optimization. dec 2014. Disponível em: <http://arxiv.org/abs/1412.6980>.

[25] GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016.

[26] Dumoulin, V.; Visin, F. A guide to convolution arithmetic for deep learning. *arXiv e-prints*, p. arXiv:1603.07285, Mar 2016.

[27] Greydanus, S.; Dzamba, M.; Yosinski, J. Hamiltonian Neural Networks. *arXiv e-prints*, p. arXiv:1906.01563, Jun 2019.

[28] Vondrick, C.; Pirsiavash, H.; Torralba, A. Generating Videos with Scene Dynamics. *arXiv e-prints*, p. arXiv:1609.02612, Sep 2016.

[29] Lin, T.; Stich, S. U.; Kshitij Patel, K.; Jaggi, M. Don't Use Large Mini-Batches, Use Local SGD. *arXiv e-prints*, p. arXiv:1808.07217, Aug 2018.

[30] TETKO, I. V.; LIVINGSTONE, D. J.; LUIK, A. I. Neural network studies. 1. comparison of overfitting and overtraining. *Journal of Chemical Information and Computer Sciences*, v. 35, n. 5, p. 826–833, 1995.

[31] He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. *arXiv e-prints*, p. 6, Dec 2015.

5 APÊNDICE A

O treinamento da rede neural foi feito usando o otimizador Adam, com *learning rate* de 10^{-3}, e β_1, β_2 e ϵ de, respectivamente, $0,9$; $0,999$ e 10^{-7}, a *loss* usada foi a de Erro Quadrático Médio.

O treinamento foi feito gerando-se os próximos 5 *frames* da simulação, dados um *frame* incial, e a matriz booleana representativa dos objetos presentes. O *batch size* [29] usado foi de 8, e numero de *epochs* foi de 20, de modo que o treinamento foi interrompido antes que o modelo começasse o *overfitting* [30].

Para construção das redes, foram usados blocos residuais [31] , que são agrupamentos das camadas de convolução da rede. Os parâmetros padrões das convoluções são apresentados na Figura 17, a não ser que seja dito outros na Tabela 1 e Tabela 2. Para as funções de ativação de *LeakyReLU*, foi usado um coeficiente angular $\alpha = 0.3$.

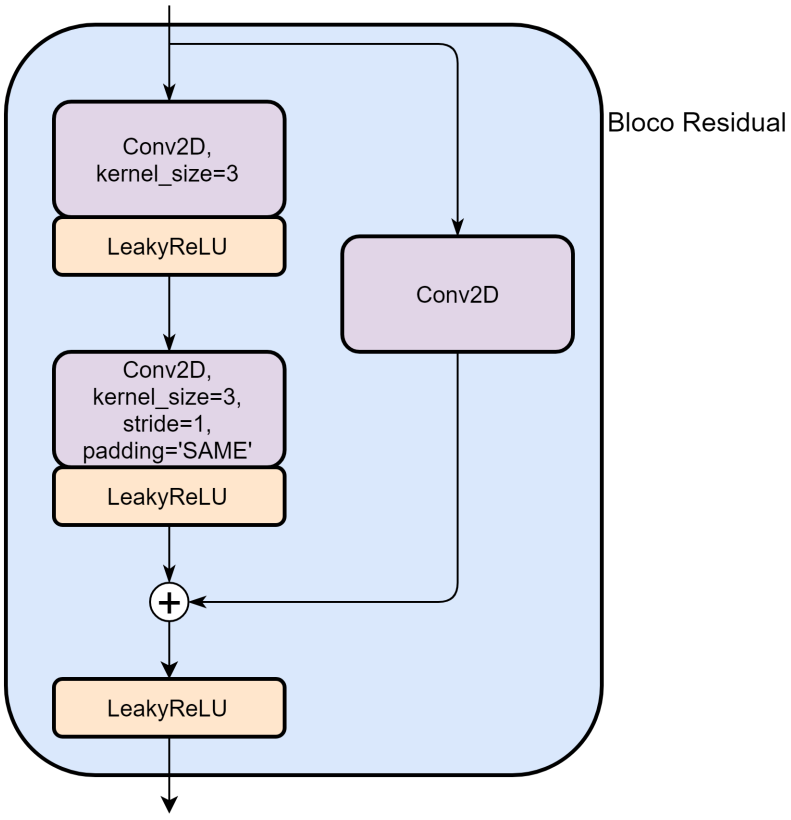


Figura 17 Bloco residual básico usado na construção da rede.

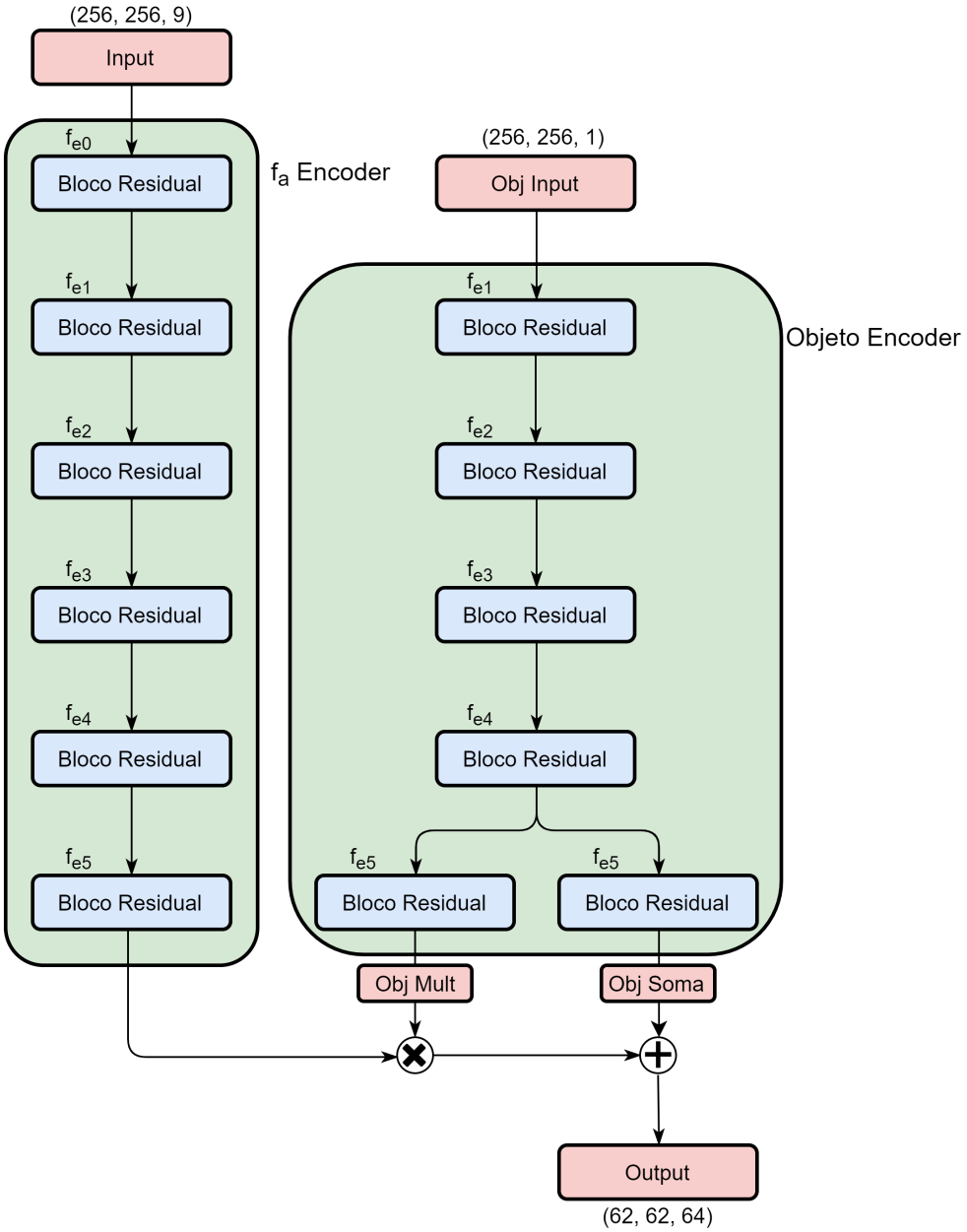


Figura 18 Primeira parte do modelo, o *encoder*.

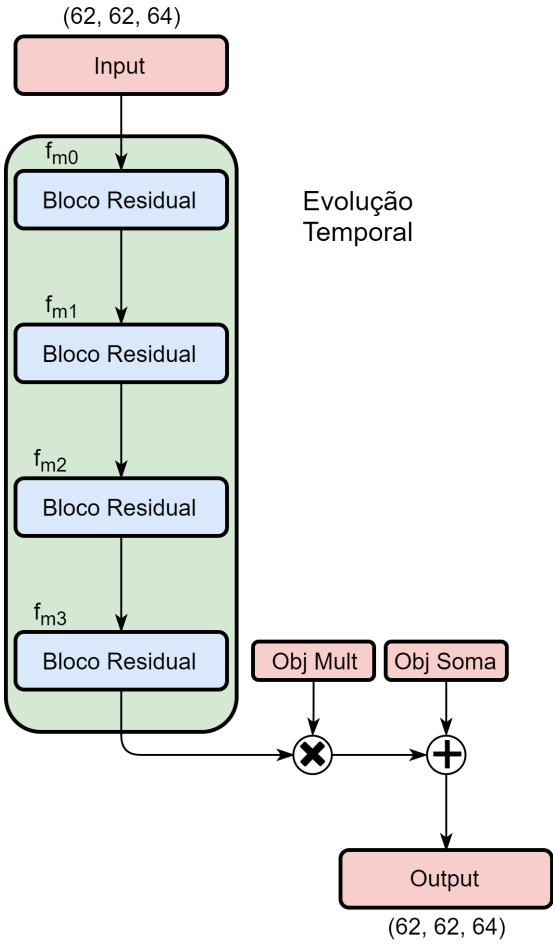


Figura 19 Segunda parte, a evolução temporal

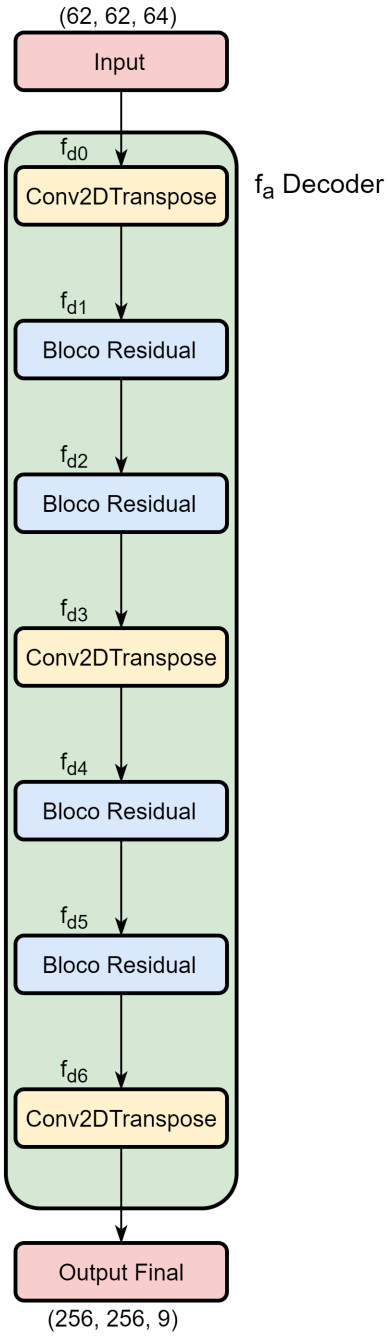


Figura 20 Terceira parte, o *decoder*

Camada	Operação	Features	Kernel	Stride	Padding
f_{e0}	Conv2D	18	4	1	Same
f_{e1}	Conv2D	32	4	2	Valid
f_{e2}	Conv2D	32	3	1	Same
f_{e3}	Conv2D	64	4	2	Valid
f_{e4}	Conv2D	64	3	1	Same
f_{e5}	Conv2D	64	3	1	Same
f_{d0}	Conv2DTranspose	32	4	2	-
f_{d1}	Conv2D	32	3	1	Same
f_{d2}	Conv2D	32	3	1	Same
f_{d3}	Conv2DTranspose	16	4	2	-
f_{d4}	Conv2D	16	3	1	Same
f_{d5}	Conv2D	16	3	1	Same
f_{d6}	Conv2DTranspose	9	3	1	-

Tabela 1 Parâmetros das camadas do encoder e decoder

Camada	Operação	Features	Kernel	Stride	Padding
f_{m0}	Conv2D	64	64	1	Same
f_{m1}	Conv2D	64	64	1	Same
f_{m2}	Conv2D	64	64	1	Same
f_{m3}	Conv2D	64	64	1	Same

Tabela 2 Parâmetros das camadas de evolução temporal.