INSTITUTO DE ENGENHARIA NUCLEAR

FILIPE SANTANA MOREIRA DO DESTERRO

APRIMORAMENTO NA PREDIÇÃO DE DOSES EM CASOS DE ACIDENTES NUCLEARES UTILIZANDO DEEP NETS E GPU

Rio de Janeiro
2018

FILIPE SANTANA MOREIRA DO DESTERRO

APRIMORAMENTO NA PREDIÇÃO DE DOSES EM CASOS DE ACIDENTES NUCLEARES UTILIZANDO DEEP NETS E GPU

Dissertação apresentada ao Programa de Pós-graduação em Ciência e Tecnologia Nucleares do Instituto de Engenharia Nuclear da Comissão Nacional de Energia Nuclear como parte dos requisitos necessários para a obtenção do Grau de Mestre em Ciência em Engenharia Nuclear – Profissional em Métodos Computacionais

Orientadores: Prof. Dr. Claudio Marcio do Nascimento Abreu Pereira
Dr. Adino Américo Heimlich Almeida

Rio de Janeiro
2018

DESTERRO Moreira Santana, Filipe

Aprimoramento na predição de doses em casos de acidentes nucleares utilizando deep nets e gpu / Filipe Santana Moreira do Desterro – Rio de Janeiro: CNEN/IEN, 2018.

xi, 53f. : il.; 31 cm.

Orientadores: Claudio Marcio do Nascimento Abreu Pereira e Adino Americo Heimlich Almeida

Dissertação (Mestrado em Ciência e Tecnologia Nucleares) – Instituto de Engenharia Nuclear, PPGIEN, 2018.

1. Inteligência artificial. 2. DeepNet. 3. Deep learning.
4. Dispersão Atmosférica de Radionuclídeos.

AGRADECIMENTOS

Primeiramente a Deus que foi meu maior orientador durante todo o curso.
Ao professor e orientador Claudio Marcio pela oportunidade concedida e por sua orientação e amizade ao longo desta jornada.
Ao meu orientador Adino por seus conselhos durante a pesquisa trabalho.
Ao meu grande amigo Marcelo Carvalho que meu auxiliou em grande parte desta caminhada.
A minha querida esposa Ana Carolina que ao vem me acompanhando nas minhas aventura e desventuras.
A Comissão Nacional de Energia Nuclear que proveu fomentos para o meu desenvolvimento durante o programa.

APRIMORAMENTO NA PREDIÇÃO DE DOSES EM CASOS DE ACIDENTES NUCLEARES UTILIZANDO DEEP NETS E GPU

Filipe Santana Moreira do Desterro

DISSERTAÇÃO SUBMETIDA AO PROGRAMA DE PÓS GRADUAÇÃO EM CIÊNCIA E TECNOLOGIA NUCLEARES DO INSTITUTO DE ENGENHARIA NUCLEAR DA COMISSÃO NACIONAL DE ENERGIA NUCLEAR COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS E TECNOLOGIA NUCLEARES.

Aprovada por:

_____
Prof. Cláudio Márcio do Nascimento Abreu Pereira, D.Sc.

_____
Dr. Adino Américo Heimlich Almeida, D. Sc.

_____
Prof. Celso Marcelo Franklin Lapa, D. Sc.

_____
Prof. Roberto Schirru, D.Sc.

_____
Prof. André Luís da Silva Pinheiro, D. Sc.

RIO DE JANEIRO, RJ - BRASIL
MARÇO DE 2018

RESUMO

Recentemente, o uso de dispositivos móveis foi proposto para a medição da avaliação da dose durante acidentes nucleares. A ideia é apoiar equipes de campo, fornecendo uma estimativa aproximada do mapa de distribuição de dose na proximidade da usina de energia nuclear (UEN), sem a necessidade de se conectar aos sistemas da UEN. A fim de fornecer essa execução autônoma, um conjunto de redes neurais artificiais (RNA) é proposto em substituição aos tradicionais sistemas de dispersão atmosférica de radionuclídeo (DAR) que utilizam modelos físicos complexos que demandam um excessivo tempo de processamento.

Uma limitação observada nessa abordagem é o treinamento muito demorado das RNAs. Além disso, se o número de parâmetros de entrada aumenta, o desempenho de RNAs tradicionais, como o *Multilayer-Perceptron* (MLP) com treinamento de *backpropagation* ou Redes Neurais de Regressão Geral (GRNN), é afetado, prejudicando sensivelmente a predição. Este trabalho centra-se no estudo de tecnologias computacionais para melhoria das RNAs a serem usadas na aplicação móvel, bem como seus algoritmos de treinamento.

Contudo, para refinar a aprendizagem e permitir melhores estimativas de dose, são necessárias arquiteturas de RNA mais complexas. As RNAs com muitas camadas (muito mais do que um número típico de camadas), às vezes referidas como Redes Neurais Profundas ou *Deep Neural Networks* (DNN), por exemplo, demonstraram obter melhores resultados. Por outro lado, o treinamento de tais RNAs é muito lento. Deste modo, com o objetivo de permitir o uso desses DNNs em um tempo de treinamento razoável. É proposta uma solução de programação paralela, usando a Unidade de Processamento Gráfico (GPU). Neste contexto, este trabalho utilizou o framework *TensorFlow* para desenvolver Redes Neurais Profundas com 9 camadas. Como resultado, *speedups* entre 50 e 100 vezes (dependendo das arquiteturas RNA comparadas) foram alcançadas no processo de treinamento, sem afetar a qualidade dos resultados obtidos (estimativas de dose).

Palavras-chave: Inteligência artificial, DeepNet, Deep learning, Dispersão Atmosférica, de Radionuclídeos.

ABSTRACT

Recently, the use of mobile devices has been proposed for the measurement of dose evaluation during nuclear accidents. The idea is to support field teams, providing a rough estimate of the dose distribution map in the vicinity of the nuclear power plant (NPP), without the need to connect to the NPP systems. In order to provide this autonomous execution, a set of artificial neural networks (ANNs) is proposed instead of the traditional atmospheric dispersion of radionuclides (ADR) systems that use complex physical models that require an excessive processing time.

One limitation observed in this approach is the very time-consuming training of ANN. In addition, if the number of input parameters increases, the performance of standard ANNs, such as Multilayer-Perceptron (MLP) with backpropagation training or General Regression Neural Networks (GRNN), is affected, leading to an irrational prediction. Thus, work focuses on the study of computational technologies to improve the RNAs to be used in the mobile application, as well as their training algorithms.

However, to refine learning and allow better dose estimates, more complex ANN architectures are required. Layer ANNs (much more than a typical number of layers), sometimes referred to as Deep Neural Networks (DNNs), for example, have been shown to perform better. On the other hand, the training of such ANNs is very slow. Thus, in order to allow the use of these DNNs in a reasonable training time. With this, a parallel programming solution is proposed, using the Graphics Processing Units (GPU). In this context, this work used the TensorFlow framework to develop deep neural networks with 9. As a result, speedups between 50 and 100 times (depending on the ANN architectures compared) were achieved in the training process, without affecting the quality of the results obtained dose).

Keywords: Artificial Intelligence, DeepNet, Deep Learning, Atmospheric Dispersion of Radionuclides.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ADR	- Atmospheric Dispersion of Radionuclides
ANN	- Artificial Neural Network
CNAAA	- Central Nuclear Almirante Álvaro Alberto
CNEN	-Conselho Nacional de Energia Nuclear
CPU	- Central Processing Unit (Unidade Central de Processamento)
cuDNN	- Deep Neural Network library
DAR	-Dispersão Atmosférica de Radionuclídeos
GPU	- Graphics Processor Unit (Unidade Gráfica de Processamento)
IPCC	- Intergovernmental Panel on Climate Change
NPP	-Nuclear Power Plant
PNUMA	- Programa das Nações Unidas para o Meio Ambiente
ONU	- Organização das Nações Unidas
OMM	- Organização Meteorológica Mundial
PEL	- Plano de Emergência Local
PEE	- Plano de Emergência Externo
RNA	- Rede Neural Artificial
SM	- Streaming Multiprocessor
SIMD	- Single Instruction Multiple Data
SDAR	- Sistema de Dispersão Atmosférica de Radionuclídeos
ULA	- Unidade Lógica e Aritmética
ZPE	- Zonas de Planejamento de Emergência

SUMÁRIO

1. INTRODUÇÃO

Um dos grandes desafios do século XXI tem sido o fornecimento de energia com um alto fator de utilização, item fundamental ao desenvolvimento humano, associado à uma redução da emissão de gases do efeito estufa. Até este momento, as alternativas de geração de energia não intermitente e independente de condições meteorológicas adversas, se concentram na geração pela queima de carvão e na energia nuclear proveniente da fissão do átomo de urânio.

O acordo de Paris de 2015 (TOLLEFSON, 2015) propôs uma limitação e redução gradativa nas emissões de gases do efeito estufa nos países signatários, grande parte das quais oriundas da produção de energia elétrica por meio da queima de carvão e gás natural. Estes compromissos, se cumpridos, serão uma forma de reduzir a projeção, prevista pelos modelos matemáticos, da elevação da temperatura média do planeta.

O *Intergovernmental Panel On Climate Change* (IPCC) é um organismo multidisciplinar, formado pelo Programa das Nações Unidas para o Meio Ambiente (PNUMA) e pela Organização Meteorológica Mundial (OMM) em 1988. No mesmo ano o IPCC foi aceito pela Organização das Nações Unidas (ONU). Com a contribuição de cientistas, o IPCC tem como objetivo desenvolver pesquisas que auxiliem nas decisões contra potenciais impactos ambientais e socioeconômicos provenientes das mudanças climáticas no mundo.

Para alcançar as metas de diminuir o aquecimento global, foram firmados em Paris, baseado em sugestões do IPCC, o acordo para a criação e desenvolvimento dos já existentes meios de produção não poluentes de energia elétrica. A energia nuclear tem sido a opção tecnológica, desenvolvida por muitos países, que o IPCC sugestiona como uma fonte de energia para a substituição da utilização de combustíveis fósseis.

Atualmente no mundo, existem mais de 440 usinas nucleares, correspondendo a 17% da produção de energia elétrica em todo planeta (ACKEMAN, 2001) [1]. No Brasil existem 2 usinas nucleares Angra 1 e Angra 2, estas fazem parte da Central Nuclear Almirante Álvaro Alberto (CNAAA). Neste mesmo local coexiste uma terceira usina Angra 3, mas que ainda está em construção. De acordo com a Eletronuclear, empresa que detém o controle da central nuclear

[1] Disponível em: http://www.ipcc.ch/ipccreports/tar/wg3/index.php?idp=128

brasileira, as potências instaladas nas duas usinas somam-se 1990 megawatts nominais, representando uma parcela de 30% da energia consumida pelo estado do Rio de Janeiro (ELETRONUCLEAR, 2011).

Uma central nuclear é construída junto com um rígido projeto de segurança. Esse projeto tem como meta a proteção dos indivíduos e do meio ambiente dos riscos radiológicos. A fim de cumprir essa meta, o projeto de segurança é formulado em diversos pontos da construção de uma usina nuclear, desde o projeto inicial, fabricação, construção, chegando até a operação. Além disso, o planejamento para possíveis acidentes, também é uma etapa crucial no projeto de segurança de uma usina, pois através dele são estipulados meios de ação que devem ser tomados em caso de acidentes, conferindo assim a uma central nuclear um alto padrão de segurança se diferenciando do padrão usado pela indústria convencional.

1.1. APRESENTAÇÃO DO PROBLEMA

As usinas nucleares brasileiras, Angra 1, Angra 2 e Angra 3 tem seu projeto de segurança descritos em diversos documentos, dentre eles os documentos: Critérios de Segurança Adotados Para As Usinas Nucleares Angra 1, Angra 2 e Angra 3 (ELETROBRAS, 2011). As usinas dispõe de diversas barreiras de segurança para assegurar que acidentes mais sérios sejam contidos sem atingir o entorno. Ainda assim, existe um plano de emergência, para atuar e assegurar o bem-estar dos trabalhadores e da população nas vizinhanças da CNAAA em casos de liberação acidental de material nuclear.

O plano de emergência é formado por outros dois planos. O Primeiro é o Plano de Emergência Local (PEL). O PEL foi traçado pela Eletronuclear utilizando as normas da Comissão Nacional de Energia Nuclear (CNEN). As ações definidas no PEL têm a intenção de proteger os trabalhadores e indivíduos que estejam dentro da CNAAA e qualquer área da Eletronuclear, que são a Vila Residencial de Praia Brava e a região de Piraquara de Fora. As equipes que fazem parte deste plano estão de prontidão 24h por dia e são colocadas em simulações de acidentes 10 vezes por ano, 5 em cada usina, para treinamento e aprimoramento do PEL.

O segundo plano é e o Plano de Emergência Externo (PEE). O PEE teve sua primeira edição em 1978, mas em 1994, tendo como coordenação a Subsecretaria de Defesa Civil do Estado do Rio de Janeiro, teve seu título mudado para Plano de Emergência Externo do Estado do Rio de Janeiro (PEE/RJ). No PEL são implementadas quatro ações específicas determinadas pelas até então Zonas

de Planejamento de Emergência (ZPE), que são áreas a redor da CNAAA, denominadas de ZPE-3, ZPE-5, ZPE-10 e ZPE-15. Cada ZPE é delimitado por um círculo a partir de um ponto central, localizado na usina Angra 1, o raio de cada círculo tem respectivamente 3 km, 5 km, 10 km e 15 km. Os treinamentos com os órgãos responsáveis e a população é realizada de dois em dois anos. Dentro do plano de emergência, um aspecto vital em caso de acidentes com liberação de material nuclear é uma estimativa rápida e precisa da dispersão atmosférica dos radionuclídeos (DAR) liberados, atualmente realizada através do Sistema de Dispersão Atmosférica de Radionuclídeos (SDAR), que realiza os cálculos de transporte e difusão dos radionuclídeos em forma de nuvens (plumas) radioativas nas ZPEs. Com este sistema é possível obter uma estimativa da dispersão da pluma radioativa na atmosfera permitindo que as equipes tomem decisões acertadas para diminuir os possíveis impactos ambientais.

O funcionamento dos SDAR envolve a simulação de modelos físicos complexos que demandam elevado poder computacional. O SDAR realiza simulações, como: predição de termos fonte, cálculos de campos de vento, dispersão de plumas, deposição de radionuclídeos e predição de doses equivalentes. Contudo, para alcançar bons resultados as simulações dependem de quão realista e refinados são os modelos computacionais, mas para alcançar a qualidade ideal são necessários computadores potentes.

O SDAR da Central Nuclear Almirante Álvaro Alberto (CNAAA) possui uma estrutura modular, composta por 4 módulos: Termo Fonte, Campo de Vento, Dispersão e Projeção. Uma característica desse sistema é o seu alto custo computacional que requer uma plataforma computacional de alto desempenho para sua execução (PINHEIRO, 2017).

Em virtude disso, (PEREIRA et al, 2017) em seu trabalho propôs a utilização de uma rede neural artificial, na predição de dose de radionuclídeos para uso em dispositivos móveis, mais especificamente smartphones, com o objetivo de fornecer suporte adicional às decisões das equipes de campo quando os sistemas de informação da planta não estão disponíveis.

1.2. OBJETIVOS

Inspirado na investigação realizada por (PEREIRA, et al, 2017), que enfatiza a importância do desenvolvimento de uma ferramenta capaz de estimar distribuições espaciais de dose, de forma eficiente e independente da CNAAA, no caso de uma eventual perda de comunicação com sistemas de segurança da CNAAA durante um acidente, e ainda, motivado pelas limitações do mesmo

trabalho, é proposta uma abordagem que permite ampliar o escopo de uso de RNAs na estimativa de dose, permitindo:

i. Um aprendizado mais eficiente (mais preciso e mais rápido);
ii. A solução de problemas mais complexos com mais variáveis (parâmetros de entrada das RNAs);

Para isso, o trabalho tem como principal objetivo desenvolver uma Rede Neural Profunda (do inglês, *Deep Net*) com um processo de aprendizagem supervisionado otimizado, utilizando computação paralela baseada em Unidades de Processamento Gráfico (GPU) para acelerar o treinamento, de forma a permitir que o número de variáveis e padrões de treinamento sejam aumentados, e o treinamento ocorra em tempo hábil, permitindo que a previsão da distribuição das doses na região de interesse seja mais precisa e realística.

Esta rede neural foi configurada para ter cinco camadas, uma de entrada, três ocultas e uma de saída. Os dados de produção usados para validar a rede neural, tiveram um erro médio de 1,050. Além disso, para o treinamento da rede neural artificial foram usados exemplos padronizados, com características de um possível acidente na CNAAA. Cada exemplo é formado por quatro variáveis: direção do vento, velocidade do vento, posição X e posição Y. O número total de exemplos para o treinamento foi 3.825. Para essa configuração de 5 x 4 x 3.825 (camadas x variáveis x exemplos), foi preciso um tempo de treinamento de 4:30h, com uma rede neural *Perceptron Multilayer*.

Em investigações preliminares desta dissertação, buscando conseguir uma maior precisão nas previsões, foi incorporado um número maior de exemplos no treinamento e uma variável adicional foi inserida. Contudo, o aumento do número de variáveis de quatro para cinco, aliado ao maior número de exemplos fez com que o tempo de treinamento crescesse de horas para dias. O acréscimo no número de camadas ocultas da rede neural artificial tornava impossível a execução do treinamento pelo sistema, pois ele não suportava o tamanho do problema. Além disso, mesmo nos casos em que a execução era possível, mesmo que demorasse muitos dias, outros problemas apareciam como o problema de desaparecimento de gradiente (*vanish gradient problem*), (SUTSKEVER, 2014).

1.3. TRABALHOS RELACIONADOS

O problema de custo computacional é sempre a fonte de barreiras no desenvolvimento e criação de soluções computacionais. Atualmente busca-se soluções alternativas para diminuir o custo computacional sem comprometer a integridade dos sistemas. Uma solução é o uso de redes neurais artificiais para poder melhorar e até substituir alguns sistemas. O seu uso em determinados problemas computacionais complexo pode obter resultados parecidos com software executados por grandes computadores de alto desempenho.

O treinamento de redes neurais para determinação de dose em áreas de acidente já obteve soluções por outras ferramentas de *machine learning*, como a pesquisa *Development of a mobile dose prediction system based on artificial neural network for NPP emergencies with radioactive material releases* (PEREIRA, et al, 2017). Neste trabalho aplicamos dois métodos de *machine learning* para determinação da dose de radiação em um domínio computacional 43x67.

O primeiro foi realizado utilizando a *General Regression Neural Network* (GRNN) (SPECHT, 1991) otimizado com Algoritmo Genético (GOLDBERG, 1989) e teve um tempo de treinamento de 00:19h, mas apesar do tempo reduzido ele não mostrou bons resultados, como mostra a Tabela 1.

Tabela 1 - Estatística da GRNN utilizando Algoritmo Genético

	Conjunto de Treino	Conjunto de Teste	Conjunto de Produção
Números de Exemplo	3.825	2.448	6.120
Erro médio absoluto	0,011	1,810	1,217
Erro máximo absoluto	1,088	74,060	98,538
Coeficiente de correlação	1,0000	0,9822	0,9882
Gerações (população = 20)	31		
Tempo de Treinamento	00:19h		

Já o segundo foi uma rede neural *Multi-Layer Perceptrons* (MLP), onde foi preciso um tempo de treinamento de 04:30h no conjunto de produção, que representa dados reais, resultou em um erro médio de 1,050 como mostrado na Tabela 2.

Tabela 2 - Estatística da Rede Neural Multi-Layer Perceptrons

	Conjunto de Treino	Conjunto de Teste	Conjunto de Produção
Números de Exemplo	3.825	2.448	6.120
Erro médio absoluto	0,691	1,113	1,050
Erro máximo absoluto	28,772	82,884	98,065
Coeficiente de correlação	0,9980	0,9906	0,9909
Épocas de aprendizado	40.000		
Tempo de Treinamento	04:30h		

A partir destes testes outro problema foi encontrado, a dificuldade de aplicar em redes neurais artificiais um número maior de camadas, variáveis e padrões de treinamento. Problemas como o tempo de processamento das épocas e o problema de desaparecimento de gradiente são características que impedem a melhora nos resultados de redes neurais artificiais. Uma solução para este tipo de problema é o uso de redes neurais que utilizam aprendizagem profunda.

1.4. JUSTIFICATIVA

O treinamento de redes neurais profundas (*Deep Nets*), que usam computação paralela baseada em GPU para acelerar o tempo de treinamento utilizando grande quantidade de exemplos, é capaz de prover uma melhoria significativa nas previsões de dose através da RNA com um aprendizado substancialmente mais rápido. Além disso, a melhoria de eficiência no aprendizado permite a utilização de um número maior de parâmetros na previsão, tornando a estimativa mais realística.

O termo *deep learning*, nos últimos anos, vem ganhando popularidade tanto na área acadêmica quanto na corporativa. De modo que, *deep learning* vem sendo tópico de estudo de diversos artigos científicos na área de inteligência artificial. Além disso, na parte corporativa, grandes empresas, como por exemplo Google, Microsoft, Baidu e Nvidia estão investindo fortemente em projetos relacionados a *deep learning*.

Ademais, a arquitetura mais empregada na área de *deep learning* para realizar esse aprendizado em camadas de abstração, são as redes neurais com aprendizado profundo (*Deep Nets*). Estas redes ganharam força atualmente por se mostrarem como uma estrutura altamente paralelizável, e por isso diversas maneiras de paralelização estão sendo testadas.

Entretanto, atualmente a forma que vem sendo mais difundida é a utilização de GPUs para o treinamento paralelo de *Deep Nets* (JÜRGEN SCHMIDHUBER, 2015), já que estas possuem um custo reduzido e um grande poder de processamento, devido ao seu grande número de processadores. Com GPUs agora pode-se processar grande número de informação de maneira bem rápida, reduzindo o processamento de semanas para dias. Contudo, desenvolver estas redes neurais profundas nesse ambiente impõe uma certa complexidade em seu desenvolvimento, devido a problemas como o de desaparecimento de gradiente que impedem que o avanço do aprendizado aconteça, podendo levar a um resultado ineficiente no treinamento da rede neural.

Entretanto, a literatura traz diversos exemplos de problemas sendo resolvidos *com Deep Nets*. Como por exemplo a empresa Nvidia que está trazendo além de GPUs, diversos materiais para o desenvolvimento de novas redes neurais com aprendizado profundo utilizando *Deep Neural Network library* (cuDNN), uma biblioteca que auxilia na configuração de uma *Deep Net* deixando a computação paralela transparente para o desenvolvedor (CHETLUR, at al, 2014).

2. FUNDAMENTAÇÃO TEÓRICA

2.1. O SISTEMA DE DISPERSÃO ATMOSFÉRICA DE RADIONUCLÍDEOS

A segurança é um tópico extremamente relevante na área nuclear, principalmente quando está relacionado as centrais nucleares. Pois, embora risco de acidentes em centrais nucleares seja pequeno, mas suas consequências são severas como pode-se observar nos casos de Tchernobil e Fukushima. Em virtude disso, o aspecto de segurança nas centrais não engloba somente a prevenção de acidentes, como também a preparação e planejamento para acidentes e situações de emergência (SANTOS, et al, 2017).

De modo que, toda central nuclear possui um plano de emergência, que em caso de um acidente envolvendo liberação de material radioativo para o meio externo, são definidas ações, como: busca de abrigo e evacuação, que devem ser tomadas de acordo com o nível de liberação para reduzir as consequências do acidente e minimizar a exposição do público à radiação. Além disso, caso o acidente seja grave, com intensa liberação de material radioativo, onde é necessário realizar evacuação, uma estimativa precisa da dispersão do material radioativo na atmosfera é um fator essencial para ajudar o processo de tomada de decisão para orientar as pessoas para longe das possíveis áreas afetadas. Nesse sentido, para prever o transporte e a difusão de material radioativo, os Sistemas de Dispersão Atmosférica de Radionuclídeos (DAR) são utilizados. Estes sistemas, para funcionar eficazmente e para poder ajudar o processo de tomada de decisão em tempo real em uma situação de emergência, processam informações de entrada sobre o termo fonte (os materiais radioativos liberados, atividades e localização), condições meteorológicas (vento, umidade e precipitação) e características geográficas (topografia, terreno, solo, etc.) (SANTOS, et al, 2017).

2.1.1. O Sistema de Controle Ambiental

A Central Nuclear Almirante Álvaro Alberto (CNAAA) também utiliza um sistema DAR, este é o Sistema de Controle Ambiental (SCA). Este sistema tem como principal atribuição servir como uma ferramenta para auxiliar na tomada de decisão, caso haja a necessidade de se realizar uma evacuação, devido a um possível acidente nuclear com liberação de radionuclídeos para o meio ambiente.

O SCA possui uma estrutura modular, sendo composto pelos seguintes módulos: Termo de Fonte, Campo de Vento, Dispersão de Pluma e Pluma e Projeção, na Figura 1 é possível observar essa estrutura:

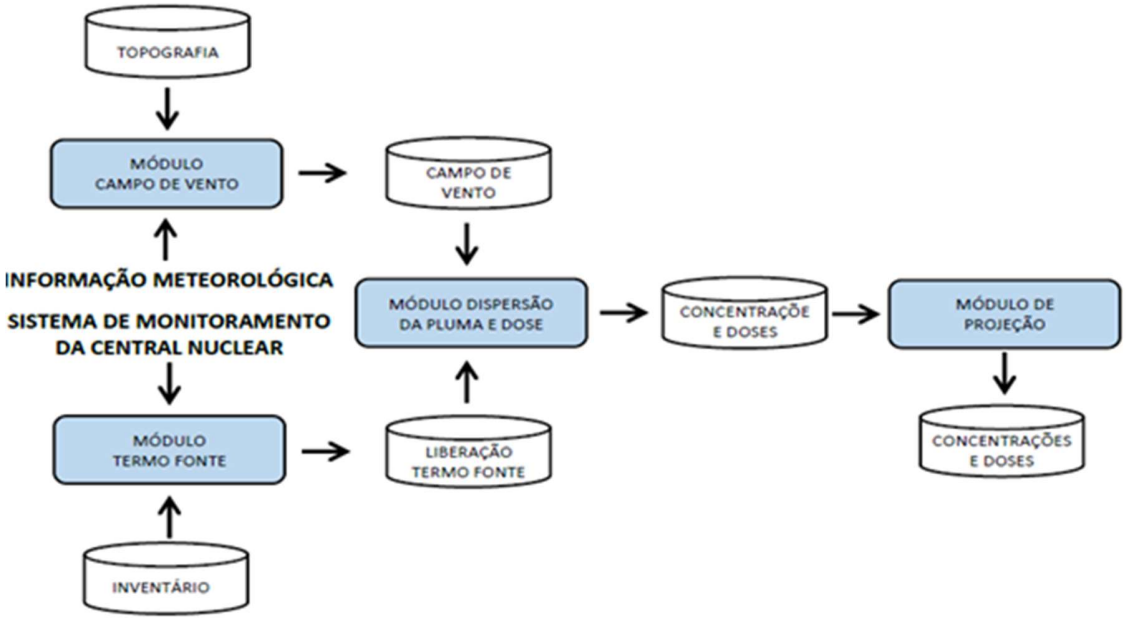

Figura 1- Estrutura Modular do SCA. Fonte: Pinheiro, 2017

Em situações de emergência, o tempo de resposta é um fator crucial, de modo que, as decisões tomadas nestas situações devem ser feitas de forma rápida, mas também precisam ser precisas. Neste contexto, o sistema DAR entra em ação, fornecendo aos tomadores de decisão informações importantes relacionadas a situação do acidente, em termos das consequências para a população dentro da área delimitada pelo sistema. Contudo, este um sistema que só deve ser ativado essencialmente em situações de emergências ou para simulações de treinamento destas situações. Neste sentido, o sistema só pode ser ativado de duas maneiras: de forma automática e de forma manual. Na forma automática, o sistema é ativado no modo de operação em emergência sempre que uma das seguintes situações acontecem:

- Sinal de injeção de segurança;
- Alarme de alto nível de radiação em qualquer um dos monitores de radiação;

- Alarme de alta atividade nas salas dos compressores do sistema de tratamento de rejeitos gasosos;
- Baixa pressão nos tanques do sistema de tratamento de rejeitos gasosos; e
- Alta pressão no tanque de alívio do pressurizador (pressão de ruptura do disco).

Enquanto que na forma manual, o sistema pode ser ativado tanto no modo de operação de emergência, no caso de um acidente real ou no modo de operação de simulação, para treinamento e/ou análise.

No modo de operação em emergência, é considerado a existência de um acidente real e só pode ser desativado pelo pessoal da Proteção Radiológica, com ordem expressa da Gerência de Situações de Emergência (FURNAS CENTRAIS ELÉTRICAS S.A., 1987). Já no caso do modo de operação em simulação, o sistema simula um acidente, mas pode ser desativado a qualquer tempo. O objetivo do SDAR da CNAAA, consiste na avaliação da dose radiológica no meio ambiente e o cálculo da dose e da previsão de dose recebida por um indivíduo em um determinado ponto afastado da fonte geradora de radiação, que no caso de um acidente nuclear, seria a própria usina nuclear. Para que a dose radiológica neste ponto seja devidamente calculada, é necessário determinar como a concentração dos radionuclídeos liberados no acidente variam espacialmente e temporalmente na atmosfera. Por sua vez, para computo da distribuição espacial e da evolução temporal do processo de transporte dos radionuclídeos emitidos, é necessário que seja calculada a quantidade de material radioativo liberado durante o acidente e como foi o transporte e difusão deste material desde a sua origem até o seu destino.

Como o transporte e difusão ocorrem em meio atmosférico é necessário levarmos em consideração as variações com o tempo das condições meteorológicas como a velocidade e a estabilidade do vento. Atualmente o sistema leva em consideração um intervalo de tempo de 15 minutos denominado de ciclo. Em cada ciclo, as condições meteorológicas são mantidas constantes para que sejam efetuados os cálculos necessários para a avaliação da dose radiológica em um determinado ponto.

A cada ciclo do sistema os seguintes cálculos são realizados:

A. Avaliação da taxa de liberação de radionuclídeos para a atmosfera;
B. Determinação do campo de velocidade e estabilidade do vento;

C. Cálculo do transporte e difusão do material radioativo na atmosfera;
D. Avaliação da distribuição espacial da dose; e
E. Cálculo da dose decorrente de projeções do ciclo atual.

A avaliação da taxa de liberação dos radionuclídeos para a atmosfera, depende fundamentalmente do tipo de acidente (real ou simulado), cada acidente em particular possui suas próprias características que interferem na quantidade dos radionuclídeos liberados. A atividade inicial de cada radionuclídeo disponível para liberação depende do acidente ocorrido e do tipo de radionuclídeo considerado, identificando-se os caminhos pelos quais essa liberação pode ocorrer. Por conta destas particularidades, o operador da usina ao diagnosticar/simular o acidente ocorrido, deve informar ao sistema através de uma linha de comando qual o acidente em questão para que a atividade total liberada no ciclo possa ser calculada pelo modelo de termo fonte, fazendo um balanço da atividade através de cada caminho possível compatível com o acidente, de acordo com o status dos diversos componentes dos caminhos.

A taxa de liberação média do ciclo é calculada, dividindo-se a atividade total liberada no ciclo, pela quantidade de bufadas liberadas no mesmo ciclo. O valor da taxa de liberação média é tomado como a atividade liberada por uma bufada no ciclo. Uma vez calculada a taxa de liberação média por ciclo, o sistema deve então calcular a distribuição espacial do campo de velocidade do vento baseado nos valores médios do ciclo obtidos nas torres meteorológicas, da velocidade do vento (magnitude e direção).

O campo de vento é tornado consistente em massa após um processo de extrapolação e interpolação a partir dos valores fornecidos, através de um processo interativo que elimina as divergências do campo interpolado. Com o campo de vento devidamente conhecido pelo processo acima descrito e distribuído na malha tridimensional, o cálculo do transporte e da difusão dos radionuclídeos na atmosfera é então iniciado e possui um modelo de bufadas tridimensional com trajetória lagrangeana variável e difusão gaussiana. Cada bufada possui uma taxa de liberação média que foi calculada pelo módulo termo fonte e é liberada e transportada durante um intervalo de advecção, cujo valor é definido a cada ciclo, com a velocidade local do campo de vento no ponto que corresponde ao início do intervalo de advecção. A contribuição de uma bufada para a concentração em cada ponto da malha é determinada pelo modelo gaussiano ao longo da trajetória percorrida pela bufada durante o ciclo e a concentração em cada ponto da malha é o somatório das

contribuições médias das bufadas naquele determinado ponto, durante o ciclo que estiver sendo processado.

Os seguintes efeitos físicos são considerados no modelo de transporte e difusão (COPPE/UFRJ - NUCLEAR, LABORATÓRIO DE ANÁLISE E SEGURANÇA, 1987).

- Efeito de esteira ("building wake") causado pelos edifícios da usina;
- Elevação da pluma (empuxo térmico e/ou quantidade de movimento devido à velocidade de saída do material liberado);
- Depleção seca (deposição de particulados e iodos no solo);
- Depleção molhada (deposição de particulados e iodos em caso de chuva durante o transporte atmosférico);
- Decaimento radioativo;
- Reflexão no solo; e
- Variação do coeficiente de dispersão quando a trajetória da bufada atravessa regiões com diferentes classes de estabilidade.

Ao final de cada ciclo, são geradas tabelas que possuem a concentração média de iodos e particulados a nível do solo e para gases nobres a nível do solo, 50m, 125m e 275m. Com base nas tabelas geradas, são calculadas a taxa de dose média no ciclo e as doses acumuladas desde o início do acidente para a tireoide e pulmão, devido à inalação de iodos, particulados e gases nobres e para o corpo inteiro devido à imersão na nuvem radioativa onde somente existe a contribuição dos gases nobres usando-se o modelo de pluma finita. Assumindo a persistência das condições do ciclo atual, tais como condições meteorológicas e taxa de liberação dos radionuclídeos para a atmosfera, o sistema fornece projeções para uma e duas horas, para as taxas de dose média e doses acumuladas, contados a partir do final do ciclo atual.

2.2. COMPUTAÇÃO PARALELA

A computação paralela é uma forma de computação, baseada no conceito de dividir para conquistar, no qual grandes problemas são divididos em problemas menores os quais são resolvidos

de forma simultânea como mostra a Figura 2. Em seguida cada uma dessas partes é reagrupada retornando a solução do problema integral.

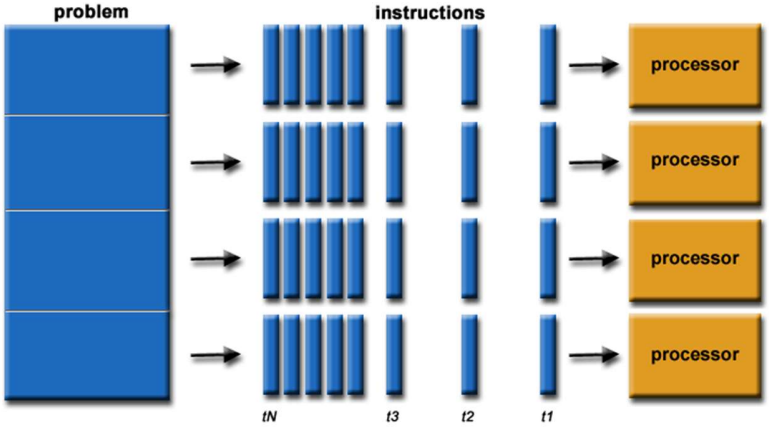

Figura 2- Conceito do Processamento Paralelo
Fonte: Introduction to Parallel Computing (Barei, B, s.d)

Nos últimos anos, o interesse na computação paralela se intensificou, uma vez que devido às limitações físicas do silício, os processadores estão evoluindo não em relação a frequência máxima, mas sim em número máximo de núcleos de processamento em um chip. A GPU é um acelerador que utiliza a arquitetura de múltiplos núcleos (multicores), no quais milhares de núcleos simples são colocadas dentro de um único chip, permitindo que milhares de tarefas sejam executadas em paralelo. Atualmente, as GPUs são dispositivos que entregam um grande poder de processamento, comparado a supercomputadores do passado, apresentando um baixo custo por watt/hora de operação.

Neste contexto, diversas aplicações na área do aprendizado de máquina estão fazendo uso da computação paralela baseada em GPU. Pois, embora a fase de aprendizagem de um algoritmo de aprendizagem típico requer aritmética de ponto de flutuante de alta precisão e um ambiente de software altamente flexível, a fase de utilização (pós-treinamento) geralmente pode fazer uso de hardware consideravelmente mais simples (LECUN, s.d).

2.2.1. A Graphic Processor Unit - GPU

Na última década, as GPUs que antes eram dispositivos voltados somente para processamento gráfico, sendo utilizada principalmente na indústria dos games, se tornaram referência quando o assunto é resolução de problemas complexos, que demandam massivo poder de processamento, da área da ciência e da engenharia. De modo que, a empresa que detectou o potencial das GPUs para além do processamento gráfico foi a NVIDIA, esta começou a desenvolver GPUs capazes de trabalhar com aplicações de uso geral e criou a arquitetura e linguagem de programação CUDA (*Compute Unified Device Architecture*), que viabilizou que algoritmos feitos para se comunicar diretamente com a GPU fossem desenvolvidos, de maneira a fazer uso dos poderosos processadores gráficos da GPU para processar dados de aplicações gerais (MATOS, 2016).

Embora sejam similares, as CPUs e as GPUs possuem diferenças fundamentais em suas a arquiteturas, como mostra a Figura 3. Enquanto que a CPU é produzida para ser flexível, com capacidade de controlar diversos tipos de programas, e em virtude disso possuindo poucas unidades de processamento lógico (ULA), mas grandes unidades de controle e memória em seu chip. As GPUs em contrapartida possuem poucas unidades lógicas e de memória, sendo a maior parte do seu chip é composta por ULAs.

Figura 3- Diferença das Arquiteturas da CPU e da GPU.
Fonte: David, 2013

A nível de *hardware*, o principal componente da GPU, é o Streaming Multiprocessor (SM), é nele em que o processamento acontece, uma vez que uma GPU geralmente possui vários SMs e

dentro deles estão os processadores CUDA que realizam as operações matemáticas para as Threads (SANTOS, et al, 2017). A Figura 4 mostra a estrutura da SM de GPU NVIDIA Pascal, é importante salientar, que mesmo sendo da mesma arquitetura, a quantidade de SMs na GPU é diferente de modelo para modelo.

Figura 4 - Arquitetura da SM das GPUs NVIDIA Pascal.
Fonte: GeForce GTX 1080 Whitepaper

2.3. REDES NEURAIS ARTIFICIAIS

O cérebro humano é formado por bilhões de neurônios, sendo o principal órgão do sistema nervoso no corpo humano, responsável pelo aprendizado do ser humano. Diante de tantos

neurônios o cérebro é uma estrutura altamente complexa e robusta, tornando eficiente o trabalho de armazenamento, atualização e compartilhamento de informação. Segundo (KOCH, 1999), o funcionamento do cérebro começa com estímulos nos neurônios da periferia para estruturas mais profundas do cérebro, onde ocorre o aprendizado e armazenamento da memória.

Um neurônio é definido como uma célula, um outro nome dado a um neurônio é soma, onde acontece todo o metabolismo da célula. Outras duas partes importantes é o axônio e dendritos, responsáveis pela transmissão de mensagens do neurônio. Podemos representar uma mensagem entre neurônios a partir de um axônio de um neurônio para o dendrito de outro neurônio, chamando esse fenômeno de sinapse (GURGEL, 2014).

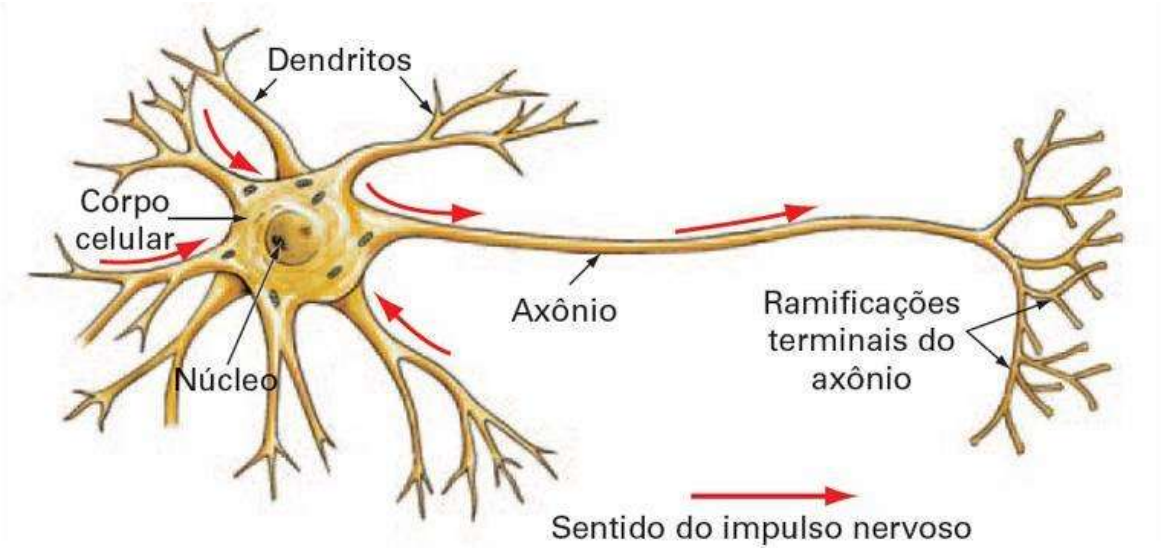

Figura 5- Representação esquemática da estrutura do neurônio
Fonte: http://deeplearningbook.com.br/o-neuronio-biologico-e-matematico/ acessado em: 01/03/2018

Um neurônio recebe sinais através de seus dendritos e enviados para os axônios, podendo seguir adiante ou não. Durante a passagem do sinal dentro do neurônio, ele pode ser amplificado ou atenuado, pois dependendo do dendrito que está associado ele recebe um peso pelo qual o sinal é multiplicado. Os pesos podem ser comparados a memória.

Inspirados no neurônio biológico pesquisadores desenvolveram um neurônio matemático, criando o que chamamos de Redes Neurais Artificiais (RNA), um modelo de aprendizagem de máquina baseado no cérebro humano. Esses modelos foram criados de forma matemática para representar um neurônio artificial.

2.3.1 Neurônios artificiais

A primeira representação e aceita pela comunidade acadêmica de um neurônio artificial foi feita por Warren McCulloch e Walter Pitts em 1943, onde é implementado de forma simplificada o funcionamento e componentes de um neurônio. O neurônio artificial funciona de maneira simples somando ponderadamente todas as entradas, aplicada a uma função e passando o resultado a frente.

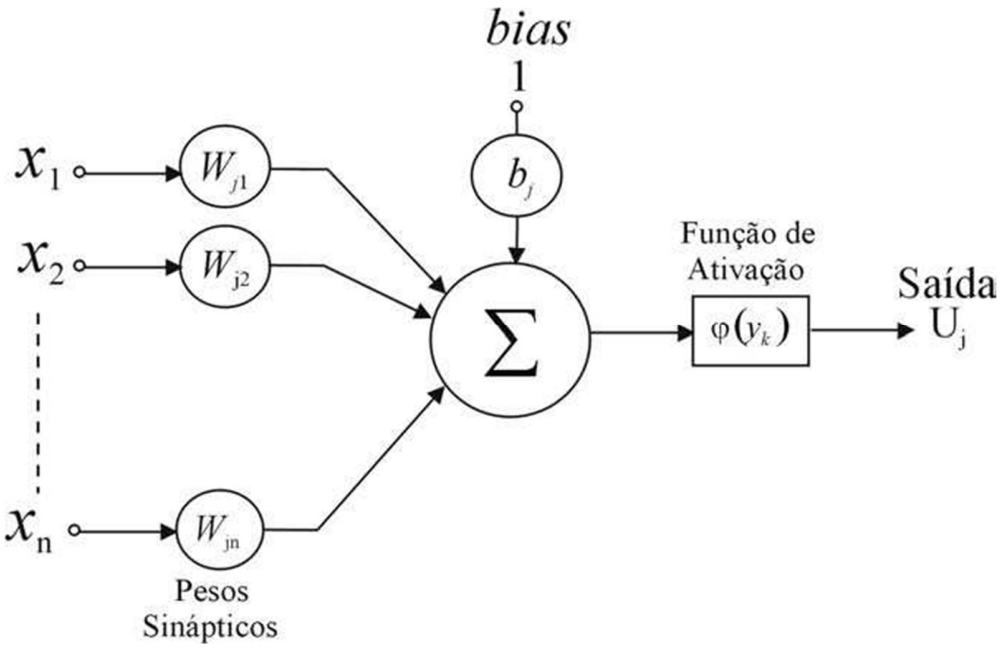

Figura 6 - Modelo simples de um neurônio artificial.
Fonte: http://www.gsigma.ufsc.br/~popov/aulas/rna/neuronio_artificial/index.html acessado em: 01/03/2018

- Sinais de entrada { x_1, x_2, ... x_n }: Representam os dados que alimentam o neurônio, normalmente são entregues normalizados para influenciar na eficiência computacional.
- Pesos sinápticos { W_1, W_2, ... W_n }: Esses valores são aprendidos com o treinamento da rede neural, eles ponderam os valores de entrada.
- Combinador Linear { $\sum$ }: Soma todos os valores de entrada, já ponderados pelos pesos sinápticos, buscando encontrar um potencial de ativação
- Bias { b_j }: A partir do conhecimento fornecido, ele permite um grau maior adaptação da rede neural., fazendo com que a rede amplie seu aprendizado.

- Função de ativação { $\varphi(Y_k)$ }: A função de ativação limita a saída da rede neural em um intervalo de valores, ativando ou não aquele neurônio.
- Sinal de saída { y }: Este é o resultado de saída da rede, ele também pode ser usado com entrada em outros neurônios.

2.4. DEEP NETS

Rede neurais profundas ou *Deep Nets*, são redes neurais artificiais constituídas por múltiplas camadas ocultas entre as camadas de entrada e saída e são utilizadas na modelagem de sistemas hierárquicos complexos.

Até os anos 90, o uso de *Deep Net* era considerado impossível pela grande parte dos pesquisadores, porém em 2006, Hilton, et al. publicou um artigo detalhando o treinamento uma rede neural profunda e demostrando uma precisão maior que 98%. A partir desta nova concepção, o interesse da comunidade cientifica voltou-se para modelos de redes neurais com aprendizado profundo, fazendo com que 10 anos depois o avanço nos modelos gerou resultados nunca visto antes na história do aprendizado de máquina (AURLIEN, 2017)(HILTON, 2006).

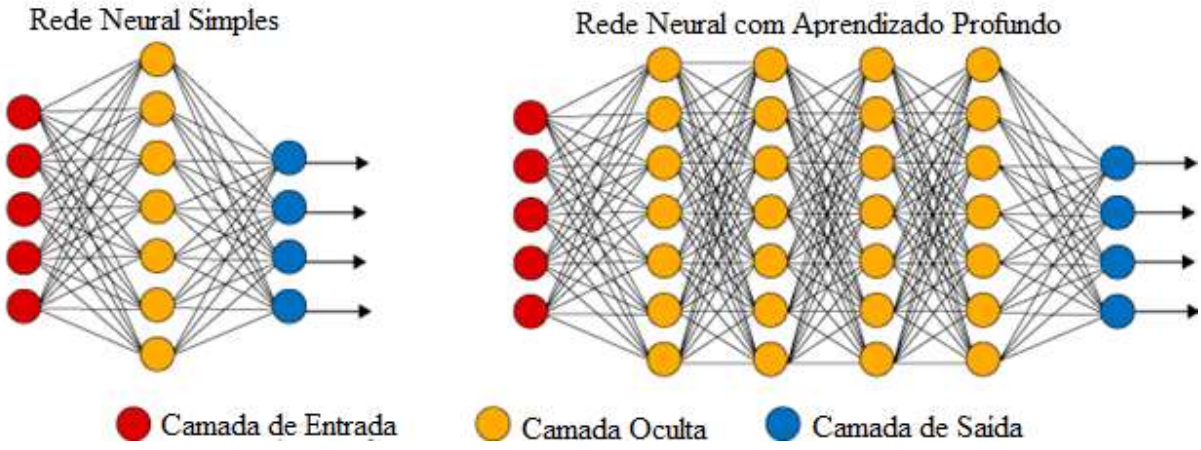

Figura 7 - Diferença entre Rede neural simples e Rede neural profunda (*Deep Nets*) (Fonte: https://www.quora.com/What-is-the-difference-between-Neural-Networks-and-Deep-Learning)

Tecnicamente a *Deep Net* tem seu conceito baseado em redes *multilayer perceptron network,* contudo, a diferença da *Deep Net* está em sua profundidade, mais especificamente no

número de camadas que compõem a rede. De modo que, a *Deep Net* de uma maneira bem simplificada pode ser definida como uma *multilayer perceptron network* com muitas camadas ocultas (BULENT, at al, 2017) e grande quantidade de neurônios.

A complexidade da *Deep Net* está no treinamento de suas camadas, em que cada camada de nós é treinada com um grupo de recursos diferente das outras camadas, baseado na saída da camada anterior, aumentando assim a complexidade de cada camada (DEEPLEARNING4J, s.d).

Diversos modos foram testados para solucionar a complexidade da *Deep Net*, e uma das formas de solução veio com o avanço do hardware, mais especificamente com o poder de processamento de GPUs, que possuem diversos núcleos capazes de executar instruções de forma paralela, sendo exponencialmente mais eficiente que os processadores tradicionais, *Central Process Unit* ou Unidade Central de Processamento (CPU), que trabalham de forma sequencial e possuem poucos núcleos.

O algoritmo de treinamento projetado sobre a computação paralela utilizando GPU, tornam o treinamento da rede neural profunda superior às que são executadas pela CPU. Resolvendo problemas como o de desaparecimento de gradiente, e usando componentes computacionais de acesso mais fácil resultando em um custo computacional mais baixo, tornando a *Deep Net* como um grande potencial de melhoramento em novos treinamentos de redes, que anteriormente foram treinadas por outros algoritmos de aprendizado sem paralelismo.

2.4.1. Arquitetura da Memória

A organização de memória em algoritmo paralelo é crucial a fim de obter a melhor performance do paralelismo, porém essa organização contém uma complexidade especial e precisa de planejamento anterior à codificação (SÁSKYA, 2014).

Sendo necessário, dois vetores iniciais para a primeira camada oculta. Um vetor de entrada que será constituído do número de neurônios multiplicado pelo número de casos de entrada que serão usados. O vetor de saída da camada terá o mesmo tamanho que a quantidade de neurônios da camada que está sendo computada. Como a paralelização ocorre por camada, os dados necessários são correspondentes a camada em questão. Com isso todas as camadas terão estrutura de vetores de entrada e saída.

Vetores lineares de pesos serão necessários para o armazenamento dos valores, para realizar a saída dos neurônios. Uma estrutura para cada camada irá guardar todos os pesos de cada neurônio da camada que está sendo computada.

O armazenamento de valores delta utilizados na retropropagação serão armazenados de maneira parecida aos vetores de saída. Já os valores das derivadas são armazenados em vetores semelhantes aos vetores de pesos.

2.4.2. Treinamento Paralelo

Para a implementação de uma rede neural com aprendizado profundo de forma paralela, será preciso uma nova forma de pensar no desenvolvimento do algoritmo, pois o algoritmo paralelo é executado de forma diferente de um algoritmo sequencial. Apesar de grandes resultados serem obtidos a partir desse modelo de rede neural, a filosofia da rede neural com aprendizado profundo não é muito diferente de uma rede neural simples, a diferença aqui é que com a chegada de GPUs, a execução de muitos neurônios com camadas maiores se tornou possível, visto que temos hoje GPUs como a GeForce GTX TITAN Z com 5760 núcleos que podem ser executados ao mesmo tempo em forma de threads e blocos.

Em uma lógica direta, os neurônios de uma rede neural serão associados diretamente a *threads* da GPU, desta forma o algoritmo permitirá executar todos os neurônios ao mesmo tempo. Todavia, sabendo que a rede neural é executada enviando sinais para frente de camada para camada, fazendo que neurônios em camadas ocultas só sejam ativados depois de receberem sinais de outros neurônios que foram ativados anteriormente, sendo assim a execução ao mesmo tempo de todos os neurônios não iria gerar o resultado esperado.

Visto que uma rede neural que usa retro propagação precisa de muitos casos de entrada para obter um bom resultado em seu treinamento, como por exemplo em uma *convolutional neural network* (CNN), que precisa de muitos casos de entradas para identificar e selecionar características diferentes de um mesmo objeto em imagens diferentes, permitindo assim que a rede identifique estes objetos em diferentes contextos com exatidão. Como cada camada avalia todos os casos de entrada igualmente, a paralelização da rede pode ocorrer na forma em que é mapeado um *Kernel* para cada camada da rede, de forma que cada camada dos casos de entrada é atribuído a um bloco de *threads*, assim cada caso terá um bloco *threads*, e cada neurônio da camada é associado a uma *Thread* do bloco (NVIDIA, 2015). Desta forma o controle de execução das camadas pode ser feito

de forma fácil, onde cada *Kernel* é executado, representa inúmeros casos sendo executado em uma só camada. A Figura 8 demonstra essa solução.

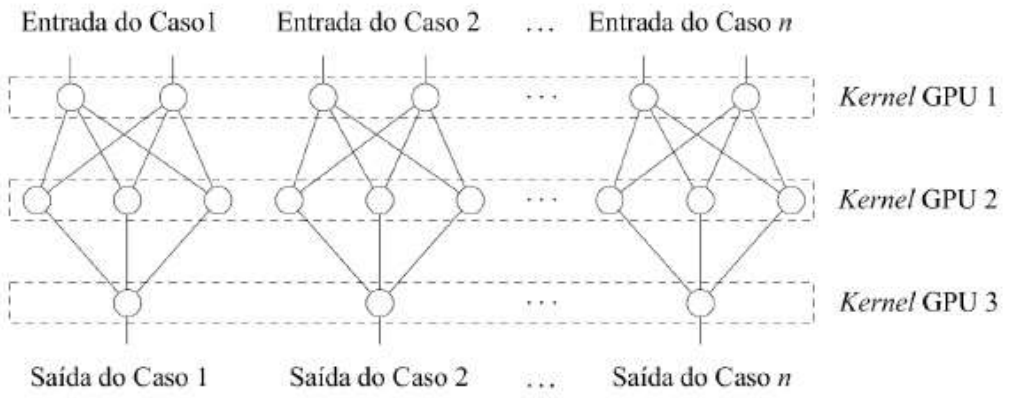

Figura 8 - Detalhamento de um tipo de treinamento paralelo (NETO, et al, 2015)

2.4.2.1. Propagação para Frente

Nesta etapa será chamada à função correspondente a cada camada da rede de forma sequencial, sendo que a execução da função será feita de forma paralela resultado da saída de cada neurônio da camada. Parâmetros usados são: vetor de peso, índice inicial do vetor de peso relacionado com os neurônios da camada, a quantidade de pesos e o vetor de entrada. Quantidade de blocos de threads será a quantidade de casos de entrada que serão usados, e a quantidade de threads serão a quantidade neurônios da respectiva camada. Com a multiplicação dos pesos pelos valores de entrada a função de ativação dos neurônios é executada que resultará no valor de saída do neurônio.

2.4.2.2. Cálculo do Valor de Delta

Esta etapa será bem parecida com a primeira, com a diferença em que aqui será feita a diferença entre os vetores de saída da referida camada com o vetor de saída esperado da rede neural.

2.4.2.3. Cálculo do Valor da Derivada de Erro

O cálculo do valor da derivada de erro também é feito de forma paralela, mas a estrutura é diferente das funções anteriores. O número de blocos de threads continua o mesmo, mas a quantidade de *threads* é alterada para o resultado entre a multiplicação do número de neurônios vezes número de pesos por neurônios. Assim poderá ser feito o cálculo da derivada de erro, que consiste na multiplicação do valor delta pertencente ao peso em questão pelo valor de saída que este peso está associado.

2.4.2.4. Atualização dos Pesos

A função que irá atualizar os pesos é a mais simples, pois ela só terá um bloco de threads, e a quantidade threads será o número total de pesos da rede neural. Assim cada thread fará atualização do peso, combinando o valor da derivada de erro com o seu peso associado.

2.4.3. Funções de Ativação

Alterações nos pesos dos neurônios durante o treinamento podem fazer com que o comportamento da RNA mude completamente, de maneira descontrolada, onde a saída da RNA para um exemplo pode ter um erro muito pequeno e para outras saídas o erro pode aumentar de maneira indesejável. Para controlar esse problema, é incluído um componente matemático, as funções de ativação, onde pequenas mudanças nos pesos e bias causem apenas pequenas alterações nas saídas.

A função de ativação é parte fundamental para o aprendizado durante o treinamento da RNA. A função de ativação decide se a informação gerada pelo neurônio é relevante ou não, tornando esse neurônio ativo ou desativado dentro da rede neural artificial. A equação 1 demonstra de forma simplista como é implementado a função de ativação matematicamente.

$$Y = Função\ de\ Ativação\ (\sum(\ Peso * entrada) + bias) \qquad (1)$$

Sem a função de ativação, os pesos e bias realizam uma transformação linear, sendo fácil de resolver computacionalmente, todavia limitada em problemas complexos. A função de ativação

é uma transformação não linear do sinal de entrada da RNA. As tarefas complexas são facilmente executadas com funções de ativação não lineares.

O desenvolvimento de *Deep Nets* se tornou atraente com as funções de ativação, pois o aumento na aprendizagem se mostrou exponencial. Foi com a função de ativação Relu que redes neurais profundas ganharam notoriedade, permitindo um avanço no uso de *Deep Nets*. Outras funções foram desenvolvidas onde iremos explorar adiante.

2.4.3.1. *Rectified Linear Unit (ReLU)*

A Relu é uma função de ativação frequentemente utilizada em *Deep Nets*. Seu uso é motivado por desativar neurônios que respondam com resultados menores que zero, assim o treinamento pode ser facilmente computado e tornando a *Deep Net* mais eficiente. Foi proposto pela primeira vez no artigo *Restricted Boltzmann Machines*[9]. A função de ativação ReLU é definida formalmente por:

$$ReLU\ (x) = \max\{0, x\} \qquad ReLU'(x) = \begin{cases} 1, & se\ x\ \geq 0 \\ 0, & c.c. \end{cases} \tag{2}$$

2.4.3.2. *Exponential Linear Unit (ELU)*

A função de ativação ELU é frequentemente utilizada em redes profundas e valores negativos são admitidos. Esta função produz uma média de ativação mais próxima de zero, fazendo com que o gradiente de aprendizado esteja próximo do gradiente natural [7]. ELUs matematicamente são assim representadas:

$$ELU\ (x, \alpha) = \begin{cases} x, & se\ x\ \geq 0 \\ \alpha(e^x - 1), & c.c \end{cases} \quad ELU'(x, \alpha) = \begin{cases} 1, & se\ x\ \geq 0 \\ ELU(x, \alpha) + \alpha, & c.c. \end{cases} \tag{3}$$

Tecnicamente as ELUs poderiam ser mais ineficientes que as Relus, por precisar realizar uma quantificação maior e não realizar a desativação de neurônios, todavia em testes realizados na literatura ela demostrou um desempenho semelhantes ao da Relu.

2.5. TENSORFLOW

O Tensorflow é o framework desenvolvido pela equipe de *machine learning* da Google de código livre, que usa um compilado[2] baseado em C++. Seu objetivo é proporcionar a facilidade de desenvolvimento de algoritmos de *Deep Nets* para diferentes problemas que usam grande quantidade de dados sem se preocupar com a programação paralela, utilizando como base a biblioteca cuDNN[3]. Nativamente o seu uso pode ser realizado em diversas plataformas, com suporte em *datacenters*, *clusters*, desktops e até dispositivos móveis como *tablets* e *smartphones*. Com a possibilidade do uso de uma linguagem de alto nível no desenvolvimento de aplicações, a biblioteca pode ser usada em diversos modelos de aprendizagem e algoritmos de treinamento sem alterar o núcleo do sistema. Assim sendo, o Tensorflow proporciona um desenvolvimento fácil de novos algoritmos e uma manutenção rápida dos sistemas de *machine learning*, facilitando assim a utilização do mesmo em diversos dispositivos (ABADI, 2016).

Para representação do TensorFlow, ele usa um gráfico de fluxo de dados, o termo em sua literatura é *Dataflow graph*. Ele consiste em que cada parte do fluxo de dados pode ser observado a computação e algoritmos de aprendizagem. Esse fluxo é representado por nós onde o Tensorflow representa um operador matemático, permitindo que camadas complexas sejam construídas facilmente (ABADI, 2015).

[2] Disponível em: https://www.tensorflow.org/

[3] "A biblioteca NVIDIA CUDA® Deep Neural Network (cuDNN) é uma biblioteca de primitivas acelerada por GPU para redes neurais profundas." disponível em: <https://developer.nvidia.com/cudnn> ultimo acesso: 01/02/2018.

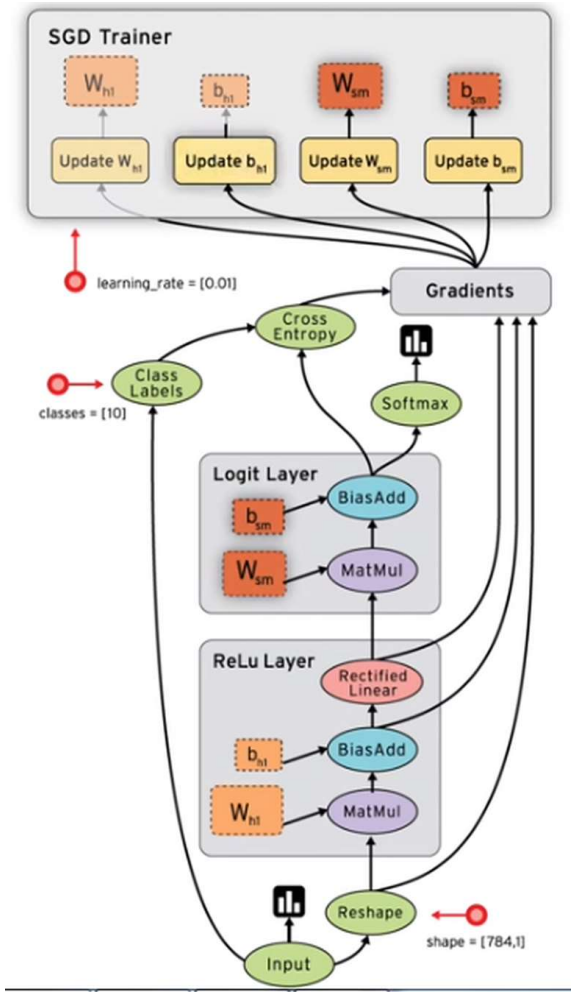

Figura 9- Detalhamento do funcionamento da rede neural no Tensorflow.
Fonte: https://www.tensorflow.org/programmers_guide/graphs acessado: 01/03/2018

Uma aplicação TensorFlow tem duas fases: na primeira é definido o programa (arquitetura da rede neural e regras de atualização) como representado na Figura 9; a segunda fase executa uma versão do programa otimizada para usar os recursos disponíveis. Esse adiamento da execução permiti a otimização do Tensorflow com informações globais sobre o cálculo.

Como os nós do Tensorflow não tem uma dependência implícita, o processamento paralelo pode ser controlado, suportando assim múltiplas execuções simultânea de seus *subgraphs*. Tensorflow se torna um framework muito versátil tanto para pesquisas quanto para a indústria.

3. IMPLEMENTAÇÃO DA DEEP NET

A implementação foi pensada para a utilização da *Deep Net* na predição de doses radioativas em torno da CNAAA. Sendo assim, para o desenvolvimento da *Deep Net* o framework Tensorflow foi utilizado.

O trabalho foi dividido em duas fases: o foco da primeira fase foi melhorar a predição da rede neural apresentada na literatura (PEREIRA, et al, 2017); enquanto a segunda fase centrou-se no aperfeiçoamento da rede para que esta apresentasse uma predição mais realista, de modo que, nesta etapa um volume maior de informações foi incorporado ao treinamento da rede.

Na realização da investigação foi preciso utilizar componentes de hardware que permitiram executar a *Deep Net*, eles são:

Tabela 3 - Hardware usado para processamento.

Placa de Vídeo	Geforce GTX 1050_{Ti}4GB 128 Bit GDDR5
Processador	Intel® Core™ i7 CPU 960 @ 3.20GHz × 8
Memória	7,8 GiB

É importante salientar que estes componentes não são de uso industrial, mas mesmo com um poder computacional relativamente baixo, ótimos resultados foram encontrados.

3.1 CONJUNTO DE DADOS

Os dados são a parte fundamental para o desenvolvimento deste trabalho, pois estes são usados para o treinamento da *Deep Net*. Sendo assim, o SDAR da CNAAA foi utilizado para simular cenários personalizados para todo o conjunto de acidentes postulados e condições meteorológicas já observadas. A partir desta simulação, o conjunto de dados foi gerado, em seguida esse conjunto foi dividido em 3 subconjuntos o de: treinamento, teste e produção.

Como será demonstrado, a *Deep Net* apresentada neste trabalho, utiliza o conjunto de dados que tem a mesma origem de trabalhos já apresentados, como o de (PEREIRA, et al, 2017) e

(DESTERRO, et al, 2017). Ademais, um segundo conjunto também foi preparado para contemplar novos testes mostrados mais adiante em outra validação. O número de padrões para cada conjunto está apresentado na Tabela 4 e Tabela 5.

Tabela 4 - Conjunto de dados 1

Dados para:	Número de exemplos
Treinamento	3.825
Teste	2.448
Produção	6.120

Tabela 5 - Conjunto de dados 2

Dados para:	Número de exemplos
Treinamento	15.299
Teste	9.792
Produção	24.480

Estes dados foram normalizados para melhorar a eficiência das *Deep Nets*. A equação 1 demonstra como a normalização foi feita.

$$X_N = \frac{(X - \overline{X})}{S} \tag{1}$$

Onde: X_N é o valor normalizado, X é o valor original, $\overline{X}$ é a média e S é o desvio padrão.

3.2 ARQUITETURA DA DEEP NET

Durante os testes, foi buscado encontrar uma arquitetura que pudesse alcançar o objetivo de diminuir o tempo de treinamento e obter resultados mais precisos. Essa busca aconteceu de forma empírica, realizando mudanças na quantidade de camadas e neurônios e usando diferentes funções de ativações. O algoritmo de treinamento usado na *Deep Net* foi o gradiente descendente.

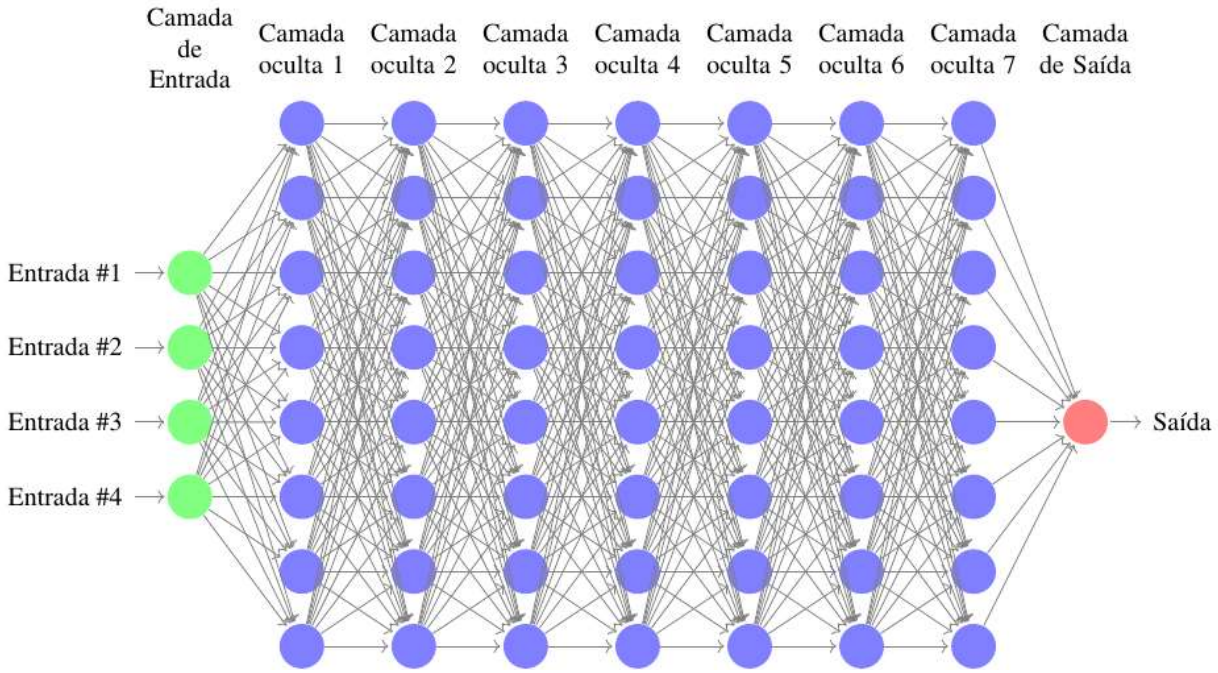

Figura 10 - Imagem ilustrativa da arquitetura da *Deep Net*

A arquitetura das *Deep Net*s será avaliada por meio de comparações dos resultados e tempo de execução do treinamento com redes neurais já presente na literatura. As arquiteturas serão ilustradas em quatro tabelas, onde cada tabela é construída com o tipo de função de ativação, quantidade de neurônios em cada camada e a camada referente. Nestas arquiteturas foi usado o conjunto de dados 1 na realização do treinamento.

Tabela 6 - Arquitetura da Deep Net 1

Função de Ativação	Neurônios	Camada
Entrada	4	1
relu	32	2
relu	16	3
relu	8	4
relu	4	5
Saída	1	6

Tabela 7 - Arquitetura da Deep Net 2

Função de Ativação	Neurônios	Camada
Entrada	4	1
relu	64	2
relu	32	3
relu	16	4
relu	8	5
elu	4	6
Saída	1	7

Tabela 8 - Arquitetura da Deep Net 3

Função de Ativação	Neurônios	Camada
Entrada	4	1
relu	128	2
relu	64	3
relu	32	4
relu	16	5
elu	8	6
elu	4	7
Saída	1	8

Tabela 9 - Arquitetura da Deep Net 4

Função de Ativação	Neurônios	Camada
Entrada	4	1
relu	256	2
relu	128	3
relu	64	4
elu	32	5
elu	16	6
elu	8	7
elu	4	8
Saída	1	9

Uma arquitetura semelhante à mostrada na Tabela 9, foi construída para receber mais dados e características do problema, entretanto foram incluídos mais neurônios a suas camadas para atender o aumento do número de padrões no conjunto de dados, fazendo-se necessário um número maior de recursos computacionais. Ilustrado na Tabela 10.

Tabela 10 - Arquitetura da Deep Net 5

Função de Ativação	Neurônios	Camada
Saída	5	1
relu	320	2
relu	160	3
relu	80	4
elu	40	5
elu	20	6
elu	10	7
elu	5	8
Saída	1	9

O funcionamento do treinamento também foi mantido, aumentando somente o número de épocas de treinamento. O algoritmo de aprendizagem também foi mantido.

3.3 ALGORITMO DA DEEP NET

Aqui será ilustrado o algoritmo usado para os testes com a *Deep Net*. Durante seu desenvolvimento foi procurada uma maneira em que diversos testes pudessem ser realizados sem que o núcleo do sistema fosse alterado. Na Figura 11 até a Figura 15 são mostrados parte do código usado para os testes.

Na Figura 11 é ilustrado a definição das variáveis meteorológicas que serão incluídas na primeira camada. Também aqui será definido a saída da última camada, que será uma dose radioativa predita.

```
38  # Input & Output Place-Holders
39  with tf.name_scope("IO"):
40      inputs = tf.placeholder(tf.float32, [None, numFeatures], name="X")
41      outputs = tf.placeholder(tf.float32, [None, 1], name="Yhat")
42
```

Figura 11 - Definição da Entrada e Saída da Deep Net

Na Figura 12 é ilustrada a definição da arquitetura da *Deep Net*. Da linha 42 a 58 é definido o tipo de conexões de dependência entre cada camada, entre as linhas 52 e 60 são definidos o número de neurônios em cada camada.

```
44  with tf.name_scope("LAYER"):
45      # network architecture
46      Layers = [numFeatures, 5, 320, 160, 80, 40, 20, 10, 5, 1]
47      h1 = tf.Variable(tf.random_normal([Layers[0], Layers[1]], 0, 0.1, dtype=tf.float32), name="h1")
48      h2 = tf.Variable(tf.random_normal([Layers[1], Layers[2]], 0, 0.1, dtype=tf.float32), name="h2")
49      h3 = tf.Variable(tf.random_normal([Layers[2], Layers[3]], 0, 0.1, dtype=tf.float32), name="h3")
50      h4 = tf.Variable(tf.random_normal([Layers[3], Layers[4]], 0, 0.1, dtype=tf.float32), name="h4")
51      h5 = tf.Variable(tf.random_normal([Layers[4], Layers[5]], 0, 0.1, dtype=tf.float32), name="h5")
52      h6 = tf.Variable(tf.random_normal([Layers[5], Layers[6]], 0, 0.1, dtype=tf.float32), name="h6")
53      h7 = tf.Variable(tf.random_normal([Layers[6], Layers[7]], 0, 0.1, dtype=tf.float32), name="h7")
54      h8 = tf.Variable(tf.random_normal([Layers[7], Layers[8]], 0, 0.1, dtype=tf.float32), name="h8")
55      hout = tf.Variable(tf.random_normal([Layers[8], Layers[9]], 0, 0.1, dtype=tf.float32), name="hout")
56
57      b1 = tf.Variable(tf.random_normal([Layers[1]], 0, 0.1, dtype=tf.float32), name="b1")
58      b2 = tf.Variable(tf.random_normal([Layers[2]], 0, 0.1, dtype=tf.float32), name="b2")
59      b3 = tf.Variable(tf.random_normal([Layers[3]], 0, 0.1, dtype=tf.float32), name="b3")
60      b4 = tf.Variable(tf.random_normal([Layers[4]], 0, 0.1, dtype=tf.float32), name="b4")
61      b5 = tf.Variable(tf.random_normal([Layers[5]], 0, 0.1, dtype=tf.float32), name="b5")
62      b6 = tf.Variable(tf.random_normal([Layers[6]], 0, 0.1, dtype=tf.float32), name="b6")
63      b7 = tf.Variable(tf.random_normal([Layers[7]], 0, 0.1, dtype=tf.float32), name="b7")
64      b8 = tf.Variable(tf.random_normal([Layers[8]], 0, 0.1, dtype=tf.float32), name="b8")
65      bout = tf.Variable(tf.random_normal([Layers[9]], 0, 0.1, dtype=tf.float32), name="bout")
66
```

Figura 12 - Definição da Arquitetura da *Deep Net*

Na Figura 13 é ilustrado a definição de cada camada com o número de neurônio definido e as conexões definidas anteriormente na Figura 12. As funções de ativação também são definidas aqui.

```
# Define the Layer operations as a Python funtion
def model(inputs, layers):
    [h1, b1, h2, b2, h3, b3, h4, b4, h5, b5, h6, b6, h7, b7, h8, b8, hout, bout] = layers

    y1 = tf.matmul(inputs, h1) + b1

    y2 = tf.add(tf.matmul(y1, h2), b2)
    y2 = tf.nn.relu(y2)

    y3 = tf.add(tf.matmul(y2, h3), b3)
    y3 = tf.nn.relu(y3)

    y4 = tf.add(tf.matmul(y3, h4), b4)
    y4 = tf.nn.relu(y4)

    y5 = tf.add(tf.matmul(y4, h5), b5)
    y5 = tf.nn.elu(y5)

    y6 = tf.add(tf.matmul(y5, h6), b6)
    y6 = tf.nn.elu(y6)

    y7 = tf.add(tf.matmul(y6, h7), b7)
    y7 = tf.nn.elu(y7)

    y8 = tf.add(tf.matmul(y7, h8), b8)
    y8 = tf.nn.elu(y8)

    yret = tf.matmul(y8, hout) + bout

    return yret
```

Figura 13 - Definição das Camadas e Neurônios da *Deep Net*

Na Figura 14 é ilustrado a definição do funcionamento da *Deep Net*, isso incluí o tipo de algoritmo usado no treinamento, a taxa de aprendizado e a função de custo.

```
# Define the operations that are performed
with tf.name_scope("train"):
    learning_rate = 0.05
    yout = model(inputs, [h1, b1, h2, b2, h3, b3, h4, b4, h5, b5, h6, b6, h7, b7, h8, b8, hout, bout])

    cost_op = tf.reduce_mean(tf.square(yout - outputs))

    train_op = tf.train.GradientDescentOptimizer(learning_rate).minimize(cost_op)
```

Figura 14 - Definição do Funcionamento da *Deep Net*

Na Figura 15 é ilustrado como o treinamento da *Deep Net* é executado pelo Tensorflow. Aqui neste ponto pode ser observado a transparência da computação paralela que fica por conta do Tensorflow A definição da quantidade das épocas é feita também nesta etapa, juntamente com a definição de parada para avaliação do conjunto de teste.

```
134
135 max_epochs = 60000
136 with tf.Session() as sess:
137
138     # initialize the variables
139     init = tf.global_variables_initializer()
140     sess.run(init)
141
142     while True:
143
144         sess.run(train_op, feed_dict={inputs: train_x, outputs: train_y})
145
146         # Update the user every 1000 epochs
147         if epoch % 1000 == 0:
148             cost = sess.run(cost_op, feed_dict={inputs: train_x, outputs: train_y})
149             print("Epoch: %d - Error: %.8f - Error Teste: %.8f" % (real_epoch, cost, cost_test))
150
151             if epoch >= max_epochs:
152                 print("STOP!")
153                 break
154
155         epoch += 1
156
```

Figura 15 - Definição da Execução do Treinamento da *Deep Net*

4. TREINAMENTO E RESULTADOS

Nesta seção serão abordados os experimentos realizados com as *Deep Net*s quantificando os ganhos obtidos em aceleração do treinamento, aumento da precisão e diminuição do erro, analisando por meios de comparações e validações de dados seus resultados junto as suas configurações com versões não paralelas apresentadas pela literatura.

4.1 TREINAMENTOS COM CONJUNTO DE DADOS 1

Nesta fase, os experimentos foram arquitetados para analisar o comportamento e os resultados gerados pela *Deep Net* em relação a RNA que não usa a computação paralela no treinamento. Sendo assim, foi utilizado um número total de exemplos para o treinamento de 3.825 exemplos e para testes de 2.448 exemplos. A validação foi feita em cima de 6.120 exemplos em produção para avaliar a eficácia da *Deep Net*.

4.1.1. Treinamento Deep Net 1

Neste experimento foi configurado 4 camadas ocultas e funções de ativação do tipo Relu. O resultado obtido não se mostrou bom em relação ao apresentado pela literatura, entretanto a velocidade de execução obteve bons resultados.

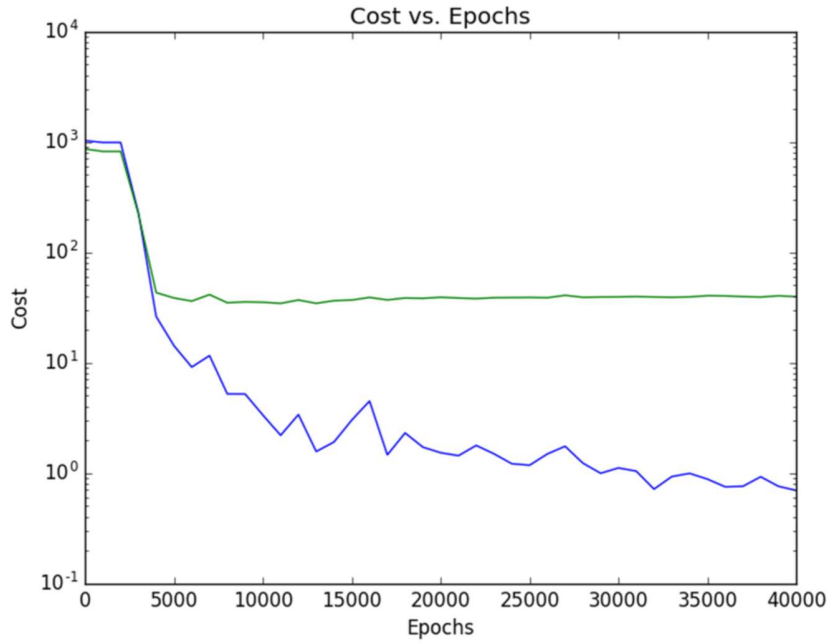

Figura 16- Progresso do erro médio absoluto durante o treinamento da *Deep Net* 1.
(linha azul: dados de treinamento; linha verde: dados de teste)

Tabela 11 - Dados do treinamento Deep Net 1

	Conjunto de Treino	Conjunto de Teste	Conjunto de Produção
Números de Exemplo	3.825	2.448	6.120
Erro médio absoluto	0,3250	1,5328	1,1311
Erro máximo absoluto	8,8292	62,7339	89,726
Erro médio quadrático	0,6926	39,4485	25,1566
r2 score (coef determination)	0,6926	0,9517	25,1566
Épocas de aprendizado	40.000		
Tempo de Treinamento	00:03:41h		

4.1.2. Treinamento Deep Net 2

O experimento seguinte configurado com 5 camadas ocultas, a camada incluída tem a função de ativação configurada na última camada. O resultado do erro médio apresentou melhoras com essas modificações.

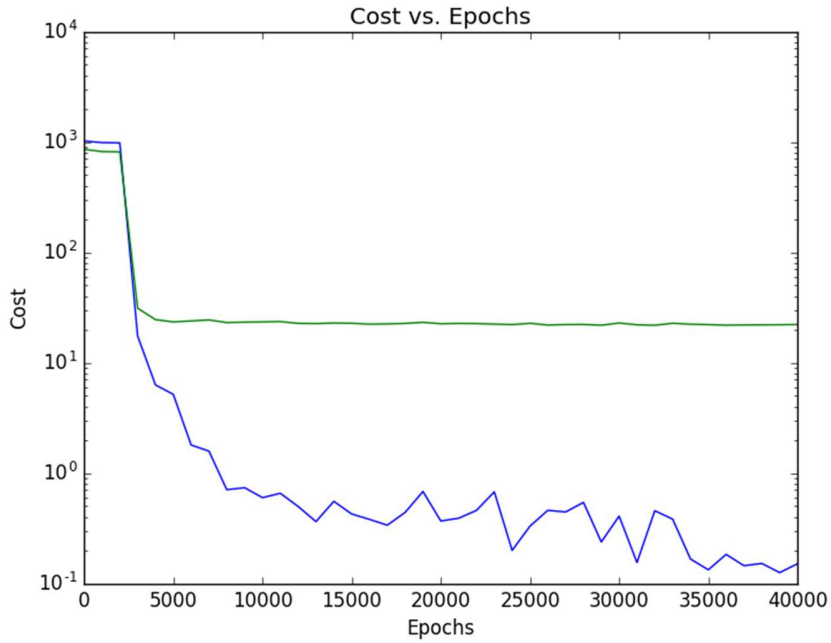

Figura 17- Progresso do erro médio absoluto durante o treinamento da *Deep Net* 2
(linha azul: dados de treinamento; linha verde: dados de teste)

Tabela 12 - Dados do treinamento Deep Net 2

	Conjunto de Treino	Conjunto de Teste	Conjunto de Produção
Números de Exemplo	3.825	2.448	6.120
Erro médio absoluto	0,1444	1,0835	0,9016
Erro máximo absoluto	6,0992	54,1352	103,024
Erro médio quadrático	0,1505	22,14	18,5744
r2 score (coef determination)	0,1505	0,9729	18,5744
Épocas de aprendizado	40.000		
Tempo de Treinamento	00:06:11h		

4.1.3. Treinamento Deep Net 3

A configuração de camadas neste experimento teve a inclusão da função de ativação Elu em duas camadas, resultando assim em melhores resultados em comparação com os últimos treinamentos.

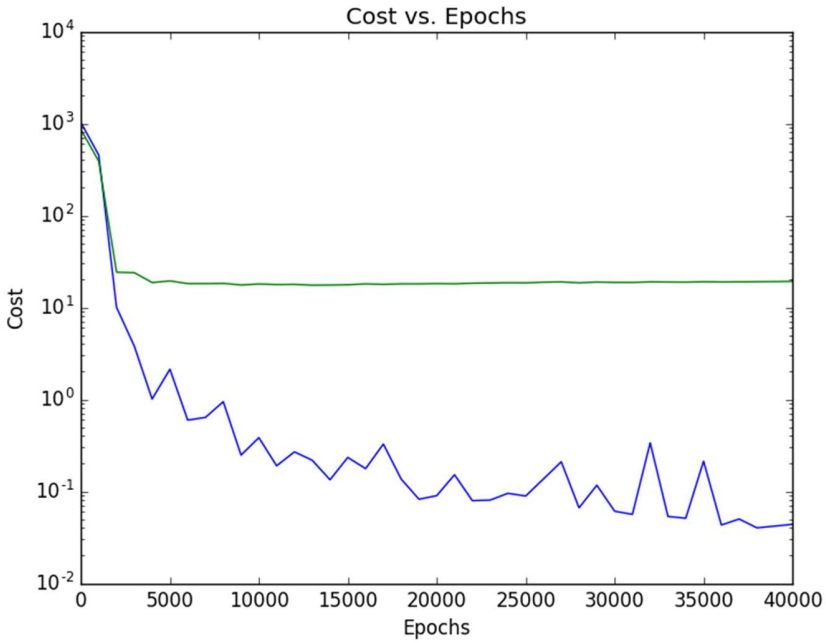

Figura 18- Progresso do erro médio absoluto durante o treinamento da *Deep Net* 3 (linha azul: dados de treinamento; linha verde: dados de teste)

Tabela 13 - Dados do treinamento Deep Net 3

	Conjunto de Treino	Conjunto de Teste	Conjunto de Produção
Números de Exemplo	3.825	2.448	6.120
Erro médio absoluto	0,0886286	1,03754	0,853251
Erro máximo absoluto	2,54128	41,2815	88,6679
Erro médio quadrático	0,043787	19,2	16,5252
r2 score (coef determination)	0,043787	0,976509	16,5252
Épocas de aprendizado	40.000		
Tempo de Treinamento	00:10:30h		

4.1.4. Treinamento Deep Net 4

Para o ultimo experimento com 7 camadas ocultas, tendo 4 funções de ativação do tipo Elu, os resultados mostraram um melhor aprendizado da rede que nas configurações anteriores.

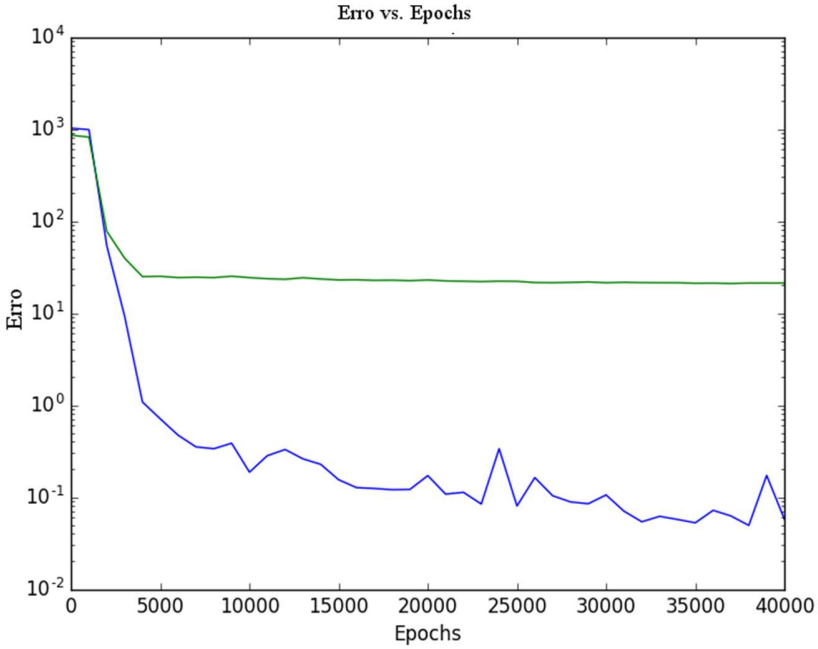

Figura 19 - Progresso do erro médio absoluto durante o treinamento da *Deep Net* 4 (linha azul: dados de treinamento; linha verde: dados de teste)

Tabela 14 - Dados do treinamento Deep Net 4

	Conjunto de Treino	Conjunto de Teste	Conjunto de Produção
Números de Exemplo	3.825	2.448	6.120
Erro médio absoluto	0,091	1,016	0,7999
Erro máximo absoluto	2,382	50,228	88,677
Erro médio quadrático	0,058	21,186	17,11
r2 score (coef determination)	0,058	0,974	17,11
Épocas de aprendizado	40.000		
Tempo de Treinamento	00:02:44h		

4.1.5. Analise dos Resultados

A *Deep Net* nestes experimentos demonstrou não só uma aceleração no tempo de treinamento, como também uma precisão maior a cada configuração de treinamento, tendo assim um erro menor na predição de doses.

Um comportamento observado durante o treinamento foi o erro absoluto médio no conjunto de teste, onde não houve alterações significativas a partir de cinco mil épocas, demonstrando que o tempo de treinamento ainda poderia diminuir caso o treinamento fosse realizado com menos épocas.

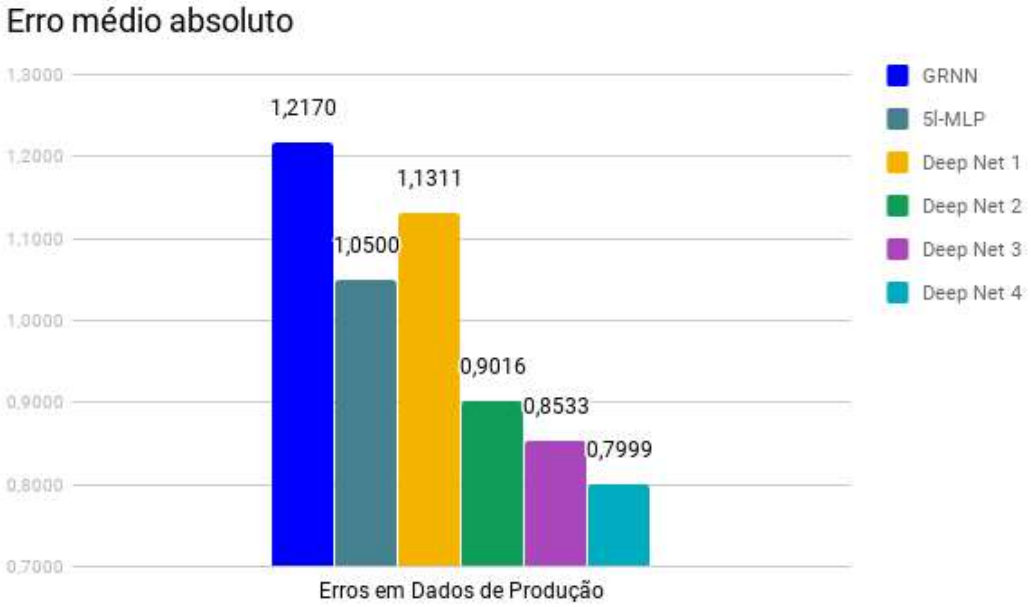

Figura 20- Diferença dos erros médios absolutos entre a *Deep Net* e RNAs

Outro comportamento que pode ser observado, foi a melhora no aprendizado da *Deep Net* com o uso de mais camadas ocultas, demonstrado na Figura 20, com o erro médio absoluto decaindo a cada experimento com mais camadas. A cada camada incluída nos experimentos cominou no aumento do tempo de treinamento, como mostrado na Figura 21. O aumento no tempo de treinamento, não prejudicou o bom desempenho da *Deep Net*, esta demonstrou precisar de um tempo bem menor do que a RNA desenvolvida em outro trabalho.

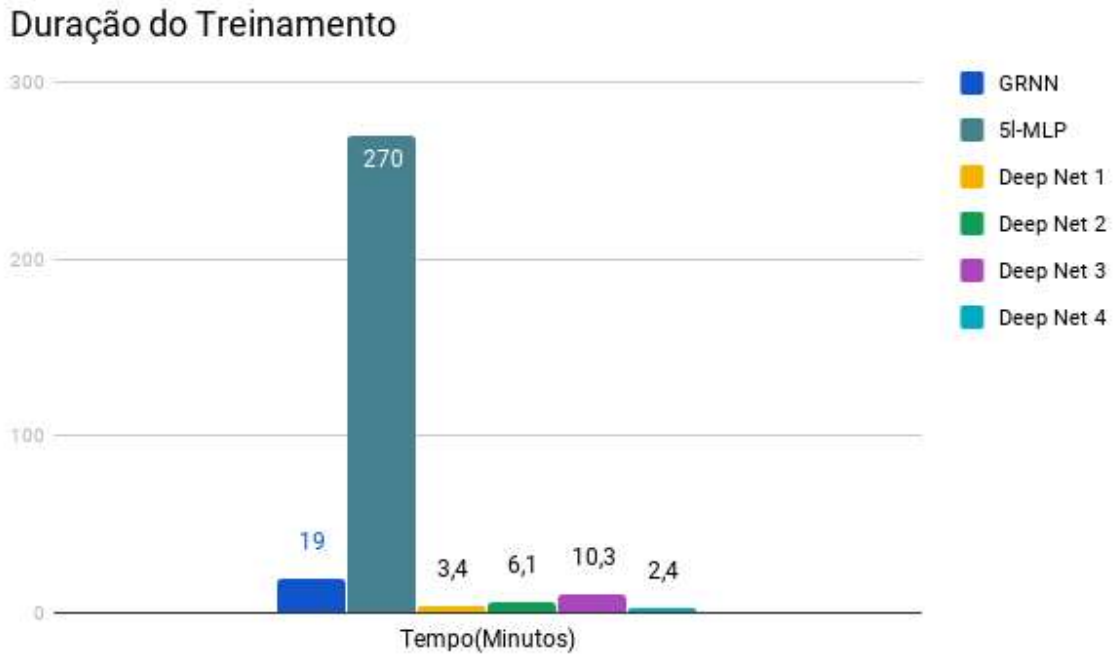

Figura 21- Diferença de tempos de duração do treinamento entre *Deep Net* e RNAs

O tempo de treinamento no ultimo experimento precisou somente de dois minutos e quarenta e quatro segundos na melhor performance, demostrando um grande avanço, e ainda assim com erro médio absoluto 23% menor que RNAs da literatura (PEREIRA, et al, 2017). O erro médio absoluto de 23% foi calculado com base nos resultados na predição no conjunto de produção, esses exemplos não foram submetidos a *Deep Net* durante o treinamento.

4.2 TREINAMENTOS COM CONJUNTO DE DADOS 2

Buscando tornar os resultados mais realistas, a *Deep Net* foi treinada novamente contemplando mais uma informação aos exemplos usados. A informação inserida aos exemplos foi o tempo. Esse tempo dentro do conjunto de dados varia de 0 a 60 minutos, dando a rede neural a possibilidade de predizer a dose em um ponto no mapa em uma faixa de até uma hora. Neste treinamento foi utilizado um número total de 15.299 exemplos.

A inclusão de uma nova característica e o aumento de dados de treinamento na mesma rede, resultou em resultados mais realísticos, diminuindo o erro médio absoluto e melhorando a predição de dose.

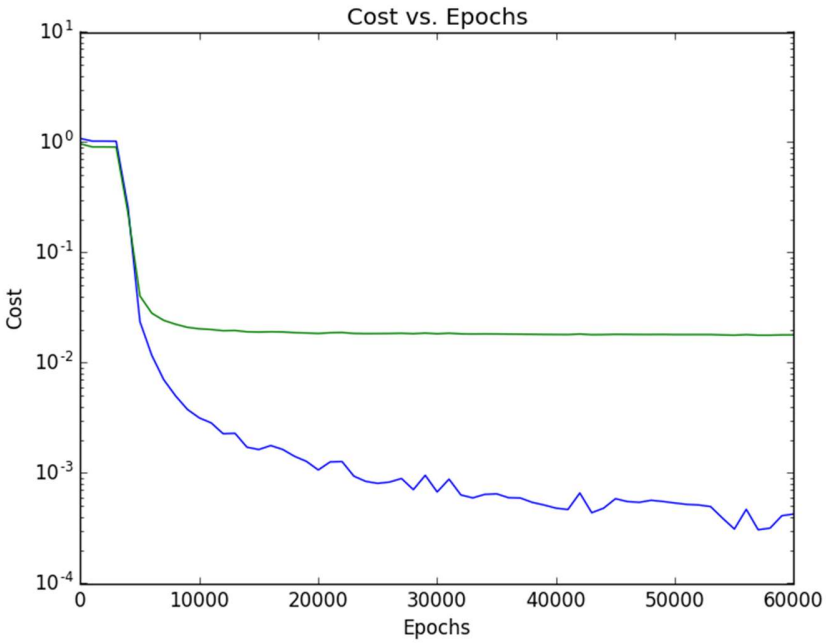

Figura 22- Progresso do erro médio absoluto durante o treinamento da *Deep Net* 5 (linha azul: dados de treinamento do segundo treinamento; linha verde: dados de teste)

Entretanto o tempo de treinamento aumentou consideravelmente, como é mostrado na Tabela 15.

Tabela 15 - Primeiro treinamento com cinco características e 60.000 épocas

	Conjunto de Treino	Conjunto de Teste	Conjunto de Produção
Números de Exemplo	15.299	9.792	2.4480
Erro médio absoluto	0,0099	0,0296	0,0253
Erro máximo absoluto	0,1798	2,1363	3,4102
Erro médio quadrático	0,0004	0,0179	0,0155
r2 score (coef determination)	0,0004	0,9802	0,0155
Épocas de aprendizado	60.000		
Tempo de Treinamento	03:31:27h		

Entretanto pode se observar que na Figura 22 que a linha do erro médio absoluto no conjunto de teste converge para um erro a partir de 10.000 épocas, marcando um tempo de trinta e um minutos, como é mostrado na Figura 23.

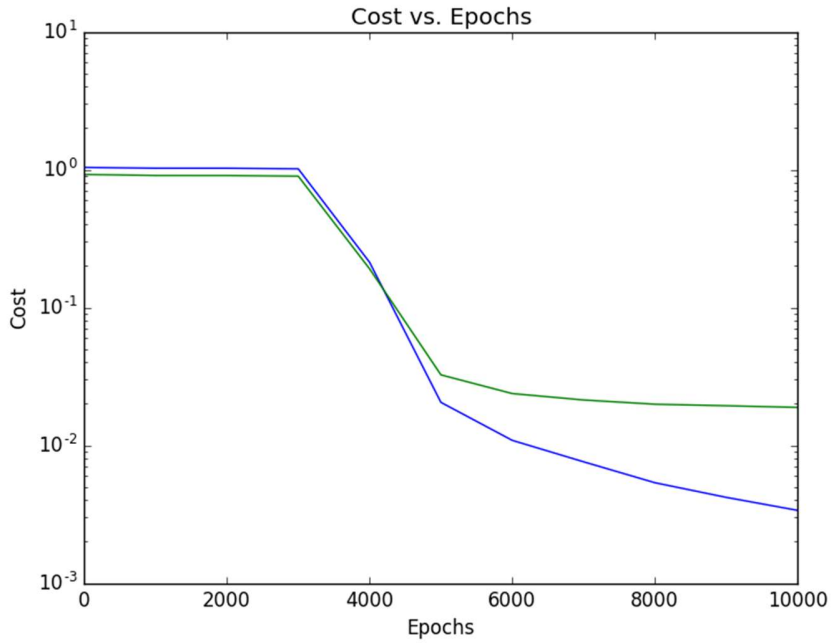

Figura 23- Progresso do erro médio absoluto com 10.000 épocas de treinamento
(linha azul: dados de treinamento do segundo treinamento; linha verde: dados de teste)

4.2.1. Comparação de Plumas

Algumas plumas foram geradas para observar seu comportamento após um acidente simulado, sendo o epicentro de liberação de radionuclídeo na Usina Nuclear Angra 1.

4.2.1.1. Pluma no tempo 15

- Velocidade do Vento: 4 m/s
- Direção do Vento 146º
- Tempos: 15
- Domínio Computacional: 43x67

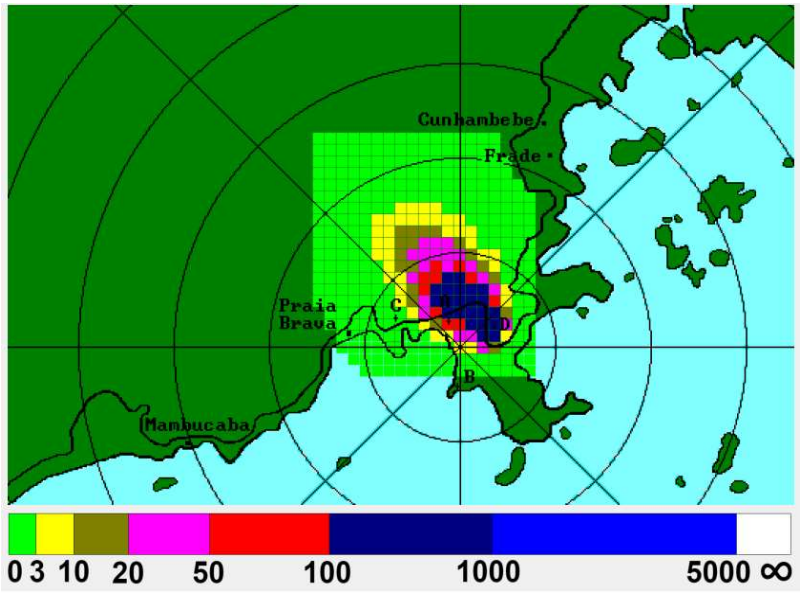

Figura 24 - Pluma no tempo 15 com dados do SCA (mRem/h)

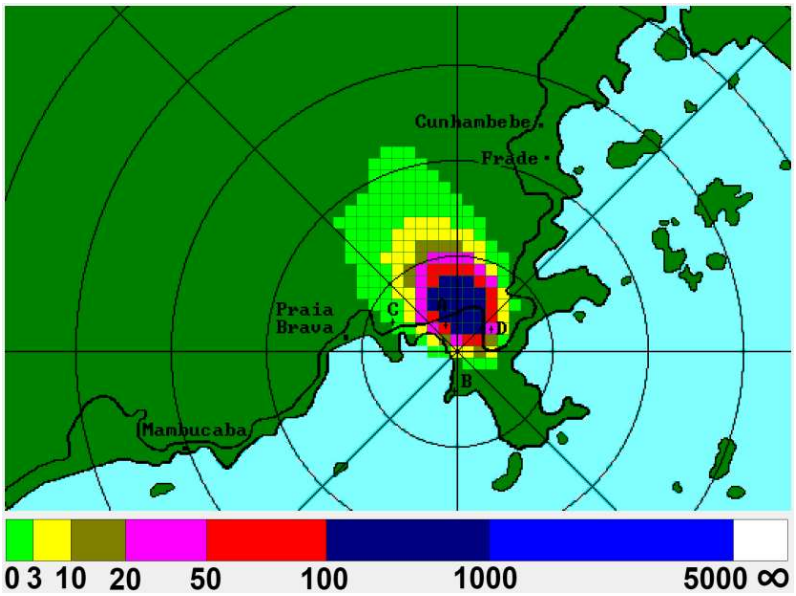

Figura 25 - Pluma no tempo 15 com predição da *Deep Net* (mRem/h)

Podemos observar que as plumas ficaram bem semelhantes nos primeiros 15 minutos do acidente, demonstrando assim um bom desempenho da *Deep Net* para esse caso.

4.2.1.2. Pluma no tempo 30

- Velocidade do Vento: 4 m/s
- Direção do Vento 146º
- Tempos: 30
- Domínio Computacional: 43x67

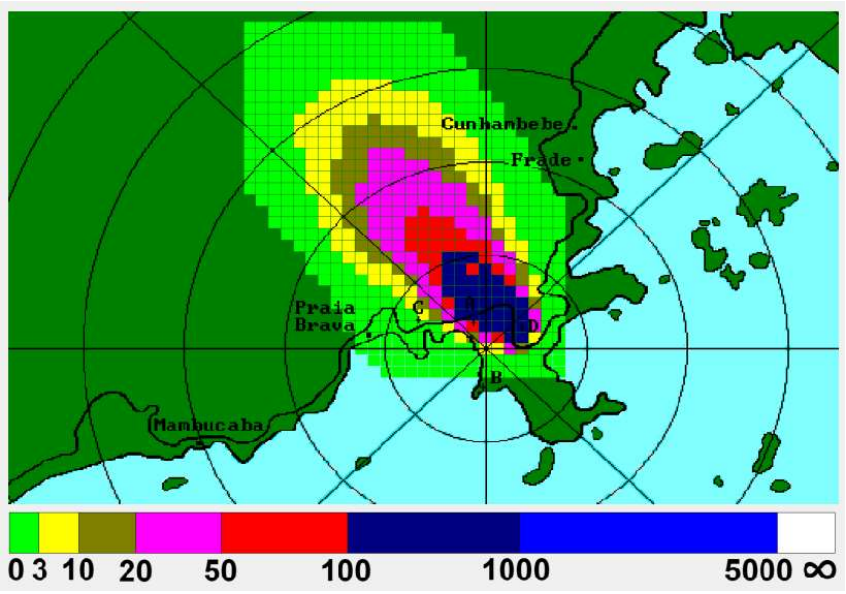

Figura 26 - Pluma no tempo 30 com dados do SCA (mRem/h)

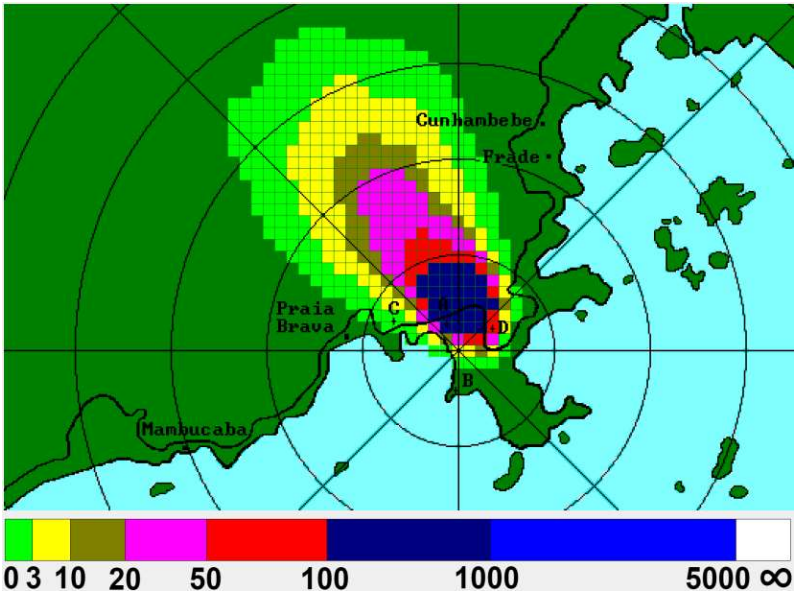

Figura 27 - Pluma no tempo 30 com predição da *Deep Net* (mRem/h)

Podemos observar que, assim como nos 15 minutos, a predição funcionou bem nos 30 minutos, com plumas bem semelhantes.

4.2.1.3. Pluma no tempo 45

- Velocidade do Vento: 4 m/s
- Direção do Vento 146º
- Tempos: 45
- Domínio Computacional: 43x67

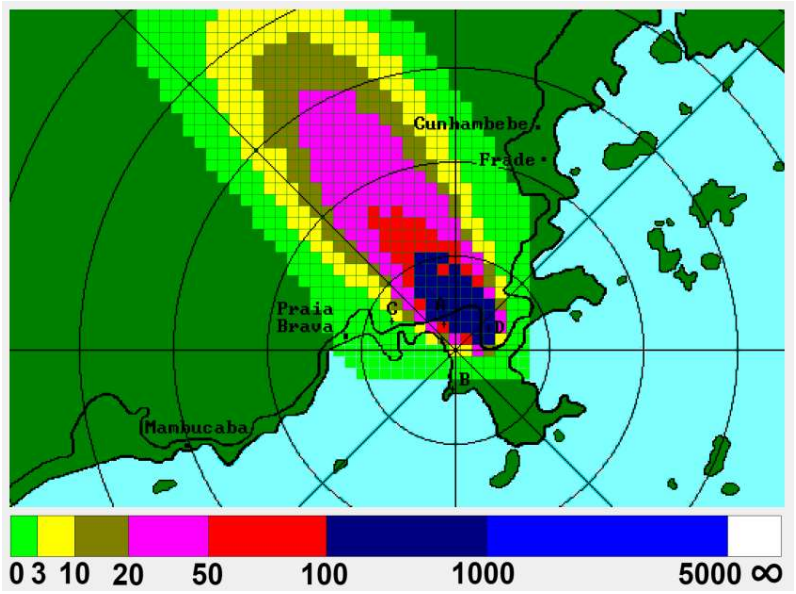

Figura 28 - Pluma no tempo 45 com dados do SCA (mRem/h)

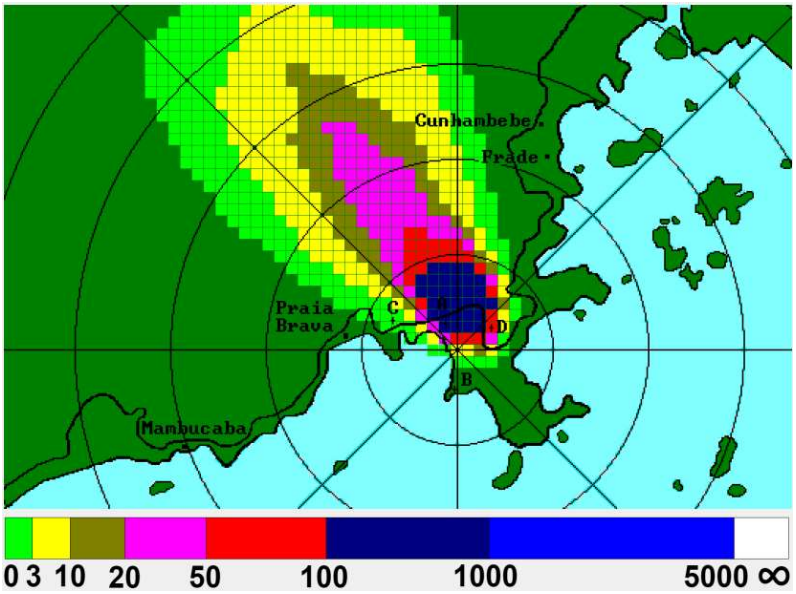

Figura 29 - Pluma no tempo 45 com predição da *Deep Net* (mRem/h)

Já nos 45 minutos, podemos observar uma pequena variação na pluma, visivelmente ela apresenta um erro maior para a predição da *Deep Net*.

4.2.1.4. Pluma no tempo 60

- Velocidade do Vento: 4 m/s
- Direção do Vento 146º
- Tempos: 60
- Domínio Computacional: 43x67

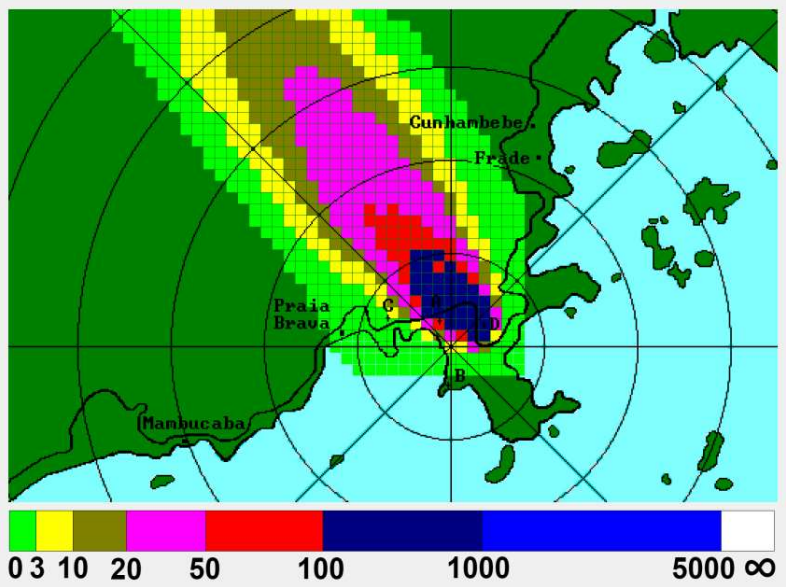

Figura 30 - Pluma no tempo 60 com dados do SCA (mRem/h)

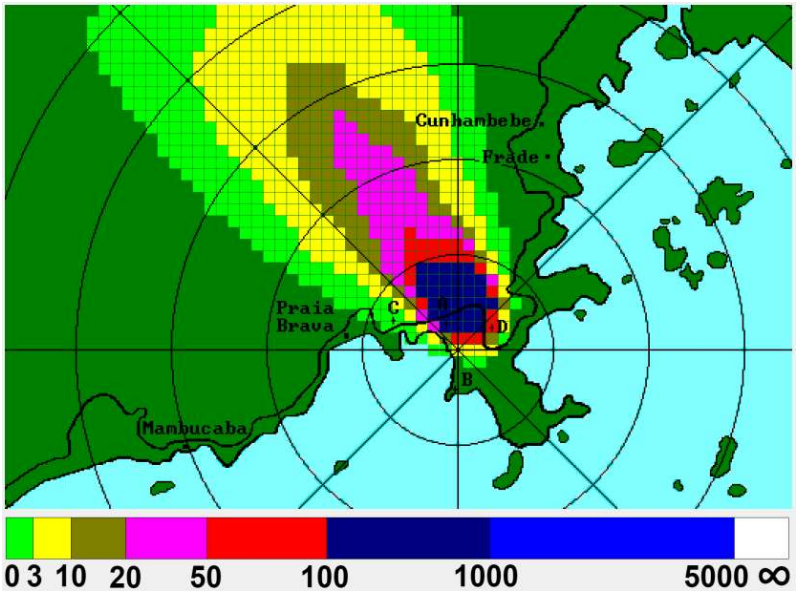

Figura 31 - Pluma no tempo 60 com predição da *Deep Net* (mRem/h)

Com 60 minutos após o acidente, a *Deep Net* apresentou um erro bem grande para as grandes distâncias do acidente, como é visto nas plumas.

5. CONCLUSÃO E TRABALHOS FUTUROS

Este trabalho propõe a utilização de redes neurais profundas (*Deep Nets*) treinadas com GPU na previsão espacial de dose em casos de acidentes nucleares que causem a dispersão atmosférica de radionuclídeos.

A versões da *Deep Net,* foram elaboradas utilizando o *framework* Tensorflow em conjunto com a linguagem Python. As arquiteturas da *Deep Net* desenvolvidas tiveram seus resultados comparados com os resultados do SDAR da CNAAA, aproximando se muito com os resultados originais.

Para validar a hipótese, foi realizado uma série de treinamentos, utilizando dados gerados pelo SDAR a partir de acidentes simulados na CNAAA. Os dados usados foram divididos em dois conjuntos de exemplos, onde o primeiro conjunto de dados representava o conjunto incialmente usado no treinamento de RNA encontrado na literatura (PEREIRA, et al, 2017), o segundo conjunto com mais dados que o primeiro, foi formado para contemplar mais informações de um acidente.

Foi possível observar que, em geral, a *Deep Net* proposta nesse trabalho teve resultados melhores que as RNAs comuns, tanto em termos de eficiência do aprendizado quanto com relação ao tempo de treinamento.

Os resultados dos treinamentos que utilizaram o conjunto de dados 1, obtiveram a um erro 23% menor na predição de doses, com uma aceleração do treinamento de 110 vezes, demonstrando assim a eficiência direta da *Deep Net*.

Pode-se observar na Tabela 15, que o treinamento através do conjunto de dados 2, apresentou um erro próximo a 10^{-2} com um tempo de treinamento de trinta e um minutos. Este resultado pode ser comparado a outras abordagens apresentadas na literatura (PEREIRA, et al, 2017), que relatam erros de 1,050 para um tempo de treinamento de quatro horas e meia. Mesmo com uma dimensionalidade da entrada maior, a *Deep Net* demonstrou uma capacidade efetiva de aprendizado, o que não ocorrera com as RNAs convencionais.

Isso demonstra que o custo computacional pode ser reduzido quando utilizado esse tipo de proposta, possibilitando que outros problemas mais complexos e de maior dimensionalidade possam ser resolvidos com *Deep Net*s treinadas com GPUs.

Sendo assim, espera-se que esse trabalho contribua para motivar o desenvolvimento de novas aplicações na área de engenharia nuclear que possam usar *Deep Net* para solucionar problemas complexos e de alto custo computacional.

Em relação a trabalhos futuros, são sugeridos os seguintes tópicos:

- A utilização do algoritmo de otimização por enxame de partículas – PSO para melhorar a arquitetura da *Deep Net*;
- Aumentar o número de variáveis aos exemplos a fim de alcançar uma predição ainda mais próxima da realidade.

REFERÊNCIAS

Barney B.,***Introduction to Parallel Computing***, Lawrence Livermore National Laboratory, [s.d].

Bulent Ayhan, Chiman Kwan, ***Application of Deep Belief Network to Land Cover Classification Using Hyperspectral Images, Advances in Neural Networks*** - ISNN, pages 269 - 276, 2017.

Bengio, Yoshua.; Courvile, Aaron; Vincent, Pascal. ***Representation Learning: A Review and New Perspectives*****, Department of computer science and operations research**, U. Montreal and Canadian Institute for Advanced Research (CIFAR), 2014.

Chetlur, Sharan; Woolley, Cliff; Vandermersch, Philippe; Cohen, Jonathan; Tran, John; Catanzaro, Bryan; Shelhamer, Evan, **cuDNN: Efficient Primitives for Deep Learning,** ARXIV, 2014.

Ciresan DC, Meier U, Gambardella LM, Schmidhuber J., ***Deep, big, simple neural nets for handwritten digit recognition***, Neural Comput, 2010.

David B. Kirk and Wen-mei W. Hwu, Acknowledgements, **In Programming Massively Parallel Processors (Second Edition)**, Morgan Kaufmann, Boston, 2013.

Gurgel, Sáskya Thereza Alves. **Análise de técnicas de implementação paralela para treinamento de redes neurais em GPU**. Dissertação (Mestrado em Informática) - Universidade Federal da Paraíba, João Pessoa, 2014.

Hinton, G. E., Osindero, S. and Teh, Y. **A fast learning algorithm for deep belief nets**. Neural Computation, 18, pp 1527-1554, 2006.

PPE/UFRJ - NUCLEAR, LABORATÓRIO DE ANÁLISE E SEGURANÇA. SCA-MOD - Sistema de Controle Ambiental - Modelagem. Rio de Janeiro: [s.n.], v. 3, 1987.

FURNAS CENTRAIS ELÉTRICAS S.A. **Especificação Funcional do Sistema de Controle Ambiental (SCA)**. Rio de Janeiro: [s.n.], 1987.

Pereira M.N.A., Schirru R., Gomes K., Cunha J., ***Development of a mobile dose prediction system based on artificial neural networks for NPP emergencies with radioactive material releases***. Annals of Nuclear Energy, Vol.105, pp.219-225, 2017.

Deeplearning4j Development Team. Deeplearning4j: ***Open-source distributed deep learning for the JVM***, Apache Software Foundation License 2.0. http://deeplearning4j.org.[s.d].

Eletronuclear, **Critérios De Segurança Adotados Para As Usinas Nucleares Angra 1, Angra 2 E Angra 3**, Rio De Janeiro, 2011.

Desterro, Filipe S. M., Almeida, Adino A. H., Pereira, Claudio M. N. A., ***Improvement Of Radiation Dose Estimation Due To Nuclear Accidents Using Deep Neural Network And Gpu***, Annals of INAC, 2017.

Frank Ackerman, Erik Alsema, Harry Audus, Jeroen de Beer, Ranjan K. Bose, John Davison, Paul Freund, Jochen Harnisch, Gilberto de M. Jannuzzi, Anja Kollmuss, Changsheng Li, Evan Mills, Kiyoyuki Minato, Steve Plotkin, Sally Rand, A. Schafer, David Victor, Arnaldo C. Walter, John J. Wise, Remko Ybema, ***Technological and Economic Potential of Greenhouse Gas Emissions Reduction Contents***, Annals of IPCC, Working Group III: Mitigation, 2001.

Guimaraes, Leonam, **Panorama da Energia Nuclear no Brasil e no Mundo - Edição 2016**, Rio De Janeiro, 2016.

Gooodfelllow, Ian ; Bengio, Yoshua.; Courville, Aaaro, ***Deep Learning***, MIT Press 2016.

Ivakhnenko, A. G., Lapa, V.G., ***Cybernetic Predicting Devices***, Kiev: Naukova Dumka, 1965.

Tollefson J, Weiss K., ***Nations Adopt Historic Global Climate Accord***, Nature, Vol. 528, Pages 315-316, 17 December 2015.

Jürgen Schmidhuber, ***Deep learning in neural networks: An overview***, Neural Networks, Volume 61, Pages 85-117, ISSN 0893-6080, 2015.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. **Backpropagation applied to handwritten zip code recognition**. Neural Computation, 1(4):541–551, 1989.

LeCun Y, Corda B. , Farabet C., **A Study of Parallel Computing for Machine Learning: which Platform for which Application? ,** [s.l], [s.d]

Koch Christof, Laurent Gilles, **Complexity and the Nervous System**, American Association for the Advancement of Science, Vol. 284, Issue 5411, pp. 96-98, 1999.

Peter Norvig, Stuart Russell, **Inteligencia Artificial: Referência Completa Para Cursos De Computaçao**, Elsevier, 2004.

Pinheiro, André. **Modelo Computacional Paralelo Baseado em GPU para Cálculo do Campo de Vento de um Sistema de Dispersão Atmosférica de Radionuclídeos**. Rio de Janeiro: Tese (Doutorado em Engenharia Nuclear) - Universidade Federal do Rio de Janeiro, 2017.

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. **TensorFlow: Large-scale machine learning on heterogeneous systems**, 2015. Software available from tensorflow.org.

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng, Google Brain, ***TensorFlow: A System for Large-Scale Machine Learning***, November 2–4, Savannah, GA, USA. pages 265-283, 2016.

Matos D., **GPU e Deep Learning,** [s.l], 2016.

Neto F., Simão V.s, Lacerda. W., **Implementação de Redes Neurais em Plataforma GPU**, Congresso Brasileiro de Inteligência Computacional, 2015.

NVIDIA, ***Deep Learning in a Nutshell: History and Training***, 2015.

Russell, Stuart J.; Norvig, Peter, ***Artificial Intelligence: A Modern Approach (2nd ed.)***, Upper Saddle River, New Jersey: Prentice Hall, 27, 32--58, 968--972, 2003.

SANTOS, Marcelo C. Dos; PEREIRA, Claudio M. N. A.; SCHIRRU, Roberto; PINHEIRO, André, **Gpu-based Parallel Computing In Real-time Modeling Of Atmospheric Transport And Diffusion Of Radioactive Material**, INAC, 2017.

Sáskya, G., **Análise de Técnicas de Implementação Paralela para Treinamento de Redes Neurais em GPU**, Dissertação (Mestrado em Informática) - Universidade Federal da Paraíba, 2014

Sutskever, Ilya and Vinyals, Oriol and Le, Quoc V, **Sequence to Sequence Learning with Neural Networks**, Annals of Advances in Neural Information Processing Systems 27, pages 3104-3112, 2014.

Warren McCulloch, Walter Pitts: **A Logical Calculus of the Ideas Immanent in Nervous Activity**. In: Palm G., Aertsen A. (eds) Brain Theory. Springer, Berlin, Heidelberg, 1943.