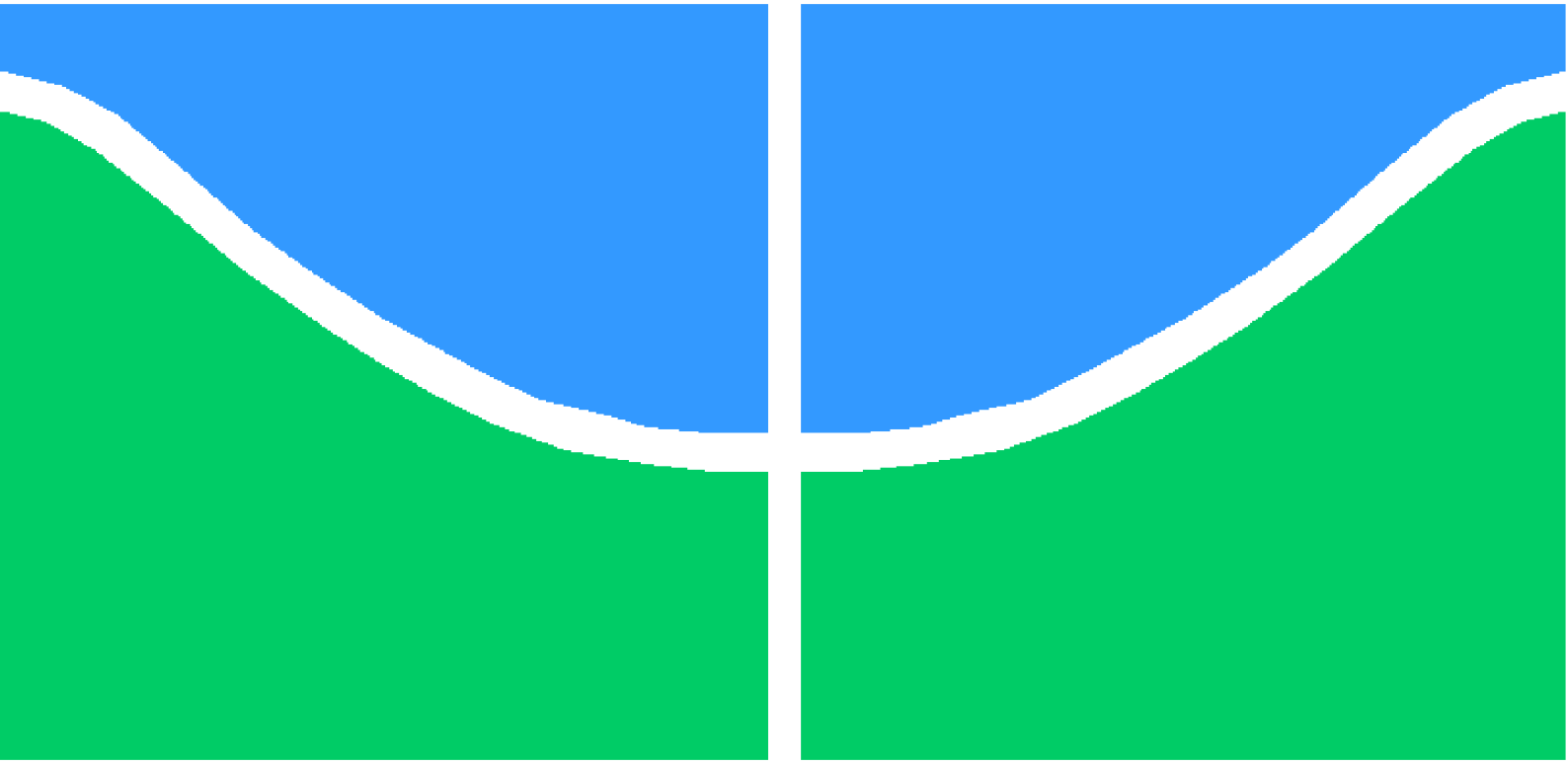


Universidade de Brasília - UnB
Faculdade UnB Gama - FGA
Engenharia Aeroespacial

Rede Neural Convolucional para classificação de imagens SAR de desflorestamento na Amazônia

Autor: Diogo Filipe Sens
Orientador: Professor Doutor Giancarlo Santilli

Brasília, DF
2021

Diogo Filipe Sens

Rede Neural Convolucional para classificação de imagens SAR de desflorestamento na Amazônia

Monografia submetida ao curso de graduação em (Engenharia Aeroespacial) da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em (Engenharia Aeroespacial).

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Professor Doutor Giancarlo Santilli

Brasília, DF

2021

Diogo Filipe Sens
Rede Neural Convolucional para classificação de imagens SAR de desflorestamento na Amazônia/ Diogo Filipe Sens. – Brasília, DF, 2021-
111 p. : il. (algumas color.) ; 30 cm.

Orientador: Professor Doutor Giancarlo Santilli

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2021.

1. Radar de Abertura Sintética (SAR). 2. Rede Neural Convolucional (CNN). I. Professor Doutor Giancarlo Santilli. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Rede Neural Convolucional para classificação de imagens SAR de desflorestamento na Amazônia

CDU 02:141:005.6

Diogo Filipe Sens

Rede Neural Convolucional para classificação de imagens SAR de desflorestamento na Amazônia

Monografia submetida ao curso de graduação em (Engenharia Aeroespacial) da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em (Engenharia Aeroespacial).

Trabalho aprovado. Brasília, DF, (13 de novembro de 2021):

Professor Doutor Giancarlo Santilli
Orientador

Professor Doutor Paolo Gessini
Convidado 1

Professor Doutor Domenico Simone
Convidado 2

Brasília, DF
2021

À minha esposa Ana Silvia,
por todo apoio, dedicação e paciência
ao longo desses anos

Agradecimentos

O trabalho de conclusão de curso representa o encerramento do ciclo da graduação, que se inicia com a aprovação no vestibular e início das aulas, passa pelos anos de estudo ao longo do curso, até chegar no momento da colação de grau e recebimento do diploma. Nessa jornada, eu não poderia deixar de começar a agradecer à minha esposa Ana Silvia, cujo apoio desde o momento em que decidi cursar (novamente) uma graduação foi fundamental. Sem ela, essa empreitada certamente teria sido impensável.

Devo agradecer também à toda equipe de professores da Faculdade do Gama da Universidade de Brasília, em particular aos professores do curso de Engenharia Aeroespacial, cuja dedicação são o fator fundamental para que a UnB seja um espaço de excelência na produção do conhecimento científico no campo das tecnologias aeoroespaciais. Em meio a tantos mestres que me lecionaram durante esses anos, dedico especiais agradecimentos aos seguintes professores: Jungpyo Lee, Olexiy Shynkarenko, Artem Andrianovi e Arthur Bertoldi, por todo o apoio nos trabalhos realizados no Laboratório de Propulsão Química (CPL) da UnB junto com a equipe de foguetes da UnB, a Capital Rocket Team, projeto o qual dediquei 4 dos 5 anos da faculdade; ao professor William Reis, pela orientação no projeto de *cubesats* junto à equipe Gama Cube Design; ao professor Ronni Amorim, que, assim como o professor Olexiy, me ofereceram orientação nos projetos de iniciação científica; aos professores Tais Calliero Tognetti, Gabriela Possa, Matheus Bernardini e Wytler Cordeiro, com os quais trabalhei como monitor ao longo do curso; aos professores Sébastien Rondineau e Domenico Simone, por terem dado todo apoio possível nas tentativas em ingressar em programas de pós-graduação no exterior; ao professor Paolo Gessini, pela orientação dada nos trabalhos de estágio no Laboratório de Sistemas Espaciais (LaSE) da UnB; e, naturalmente, ao professor Giancarlo Santilli, não somente pelas orientações durante a produção deste trabalho, mas também por ter aberto o caminho para esse interessantíssimo campo do sensoriamento remoto.

Agradeço também a toda equipe técnica do campus da FGA: os membros da secretaria; a equipe de limpeza, que tantas vezes nos ajudou a organizar toda a bagunça que fazíamos em nosso trabalho de construção dos foguetes na Capital; a equipe de segurança, responsável pelo controle do acesso aos contêineres onde os trabalhos da equipe eram realizados e cuja convivência era cotidiana; à equipe do Restaurante Universitário e a todos cujo trabalho torna o funcionamento do campus possível.

Por fim, não poderia deixar de dedicar um grande abraço de agradecimento a todos os amigos e amigas que fiz ao longo desses 5 anos, seja nas matérias em comum que pegamos, seja principalmente nos projetos extracurriculares, que exigiram tanta dedicação

de nossa parte, mas cujo aprendizado e convivência certamente valeram a pena. Aos membros antigos, presentes e futuros da Capital Rocket Team, Zenit Aerospace, Gama Cube Design e de todas as outras equipes da FGA nas quais não tive o privilégio de trabalhar, deixo o meu mais sincero agradecimento por tudo o que aprendi com vocês. Vocês são um dos pilares fundamentais que tornam a FGA essa instituição de excelência no campo.

A todos e todas que fizeram parte de minha vida nos últimos 5 anos de Engenharia Aeroespacial na UnB, meu agradecimento por tudo.

"Há, hoje, portanto, dois movimentos internacionais: um em nível do sistema financeiro, da informação, do domínio do poder efetivamente das potências; e outro, uma tendência ao internacionalismo dos movimentos sociais. [...] A Amazônia é um exemplo vivo dessa nova geopolítica, pois nela se encontram todos esses elementos. Constitui um desafio para o presente, não mais um desafio para o futuro. [...] É imperativo o uso não predatório das fabulosas riquezas naturais que a Amazônia contém e também do saber das suas populações tradicionais que possuem um secular conhecimento acumulado para lidar com o trópico úmido. Essa riqueza tem de ser melhor utilizada. Sustar esse padrão de economia de fronteira é um imperativo internacional, nacional e também regional.".

(Bertha Becker, Geopolítica da Amazônia)

Resumo

O presente trabalho visa desenvolver uma rede neural convolucional (CNN) capaz de classificar imagens de satélite SAR da Floresta Amazônica que contenham indicadores de desmatamento. CNN é uma arquitetura de aprendizagem de máquina usada para reconhecimento de padrões em imagens digitais, como reconhecimento de caracteres ou de rostos humanos. SAR é um tipo de imageamento ativo (radar) por satélite que não sofre interferência de nuvens, comuns na região amazônica. O objetivo é automatizar a detecção de sinais de desmatamento na floresta, de modo a reduzir o tempo de resposta das autoridades competentes. A acurácia do modelo desenvolvido atingiu um valor próximo de 90%.

Palavras-chaves: Rede Neural Convolucional. Imagem SAR. Floresta Amazônica. Desflorestamento.

Abstract

The present work aims to develop a convolutional neural network (CNN) capable of classifying SAR satellite images from the Amazon Forest that contain deforestation indicators. CNN is a machine learning architecture used for pattern recognition in digital images, such as character or human face recognition. SAR is a type of active satellite imagery (radar) that does not suffer from interference from clouds, common in the Amazon region. The objective is to automate the detection of signs of deforestation in the forest, in order to reduce the response time of the competent authorities. The accuracy of the developed model reached a value close to 90%.

Key-words: Convolutional Neural Network. SAR Images. Amazon Rainforest. Deforestation.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ANN	*Artificial Neural Network* (Rede Neural Artificial)
ASI	*Agenzia Spaziale Italiana* (Agência Espacial Italiana)
CBERS	*China Brazil Earth Resources Satellite* (Satélite Sino-Brasileiro de Recursos Terrestres)
CNN	*Convolutional Neural Network* (Rede Neural de Convolução)
CNSA	*China National Space Administration* (Administração Espacial Nacional da China)
CONAE	*Comisión Nacional de Actividades Espaciales* (Comissão Nacional de Atividades Espaciais - Argentina)
CSA	*Canadian Space Agency* (Agência Espacial Canadense)
DLR	*Deutsches Zentrum für Luft-und Raumfahrt* (Centro Aeroespacial Alemão)
DNN	*Deep Neural Network* (Rede Neural Profunda)
ESA	*European Space Agency* (Agência Espacial Europeia)
IA	Inteligência Artificial
IBGE	Instituto Brasileiro de Geografia e Estatística
INPE	Instituto Nacional de Pesquisas Espaciais
INTA	*Instituto Nacional de Técnica Aeroespacial* (Instituto Nacional de Tecnologia Aeroespacial - Espanha)
JAXA	*Japan Aerospace Exploration Agency* (Agência de Exploração Aeroespacial do Japão)
KARI	*Korea Aerospace Research Institute* (Instituto de Pesquisa Aeroespacial da Coreia da Sul)
LTU	*Linear Threshold Unit* (Unidade de Limite Linear)
MAE	*Mean Absolute Error* (Erro Absoluto Médio)
MSE	*Mean Square Error* (Erro Médio ao Quadrado)

NASA	*National Aeronautics and Space Administration* (Administração Nacional da Aeronáutica e Espaço)
PRODES	Programa de Monitoramento do Desmatamento da Floresta Amazônica Brasileira por Satélite
ReLU	*Rectified Linear Unit* (Unidade Linear Retificada)
RMSE	*Root Mean Square Error* (Raiz do Erro Médio ao Quadrado)
SAR	*Synthetic Aperture Radar* (Radar de Abertura Sintética)
SNAP	*Sentinel Aplication Platform* (Plataforma de Aplicação Sentinel)
UKSA	*United Kingdom Space Agency* (Agência Espacial do Reino Unido

Lista de símbolos

σ^0	Coeficiente de retroespalhamento
ω_i	Peso do i-ésimo neurônio em uma rede neural artificial
σ	quantidade de energia reletida pela seção transversal para o receptor
A	Peso unidade de área
P_r	Potência do sinal recebida
P_t	Potência do sinal transmitida
G_t	Ganho da antena de transmissão
A_r	Área da antena de recepção
R	Distância entre o imageador e o alvo
RC_2	razão cross-polarizada
x_i	i-ésimo sinal de entrada em uma rede neural artificial
z_i	i-ésimo sinal de saída de um neurônio artificial e sinal de entrada da função de ativação
$\sigma(z)$	função de ativação logística ou sigmoide
$H(z)$	função de ativação de unidade linear retificada (ReLU)
$\hat{y}_i$	i-ésimo sinal de saída da rede neural artificial (ReLU)
y_i	i-ésimo valor real de treinamento supervisionado
θ_i	i-ésimo vetor de pesos de um determinado neurônio
η	grau ou taxa de aprendizagem da rede

Sumário

Introdução

A floresta amazônica é um bioma que compreende por volta de metade do território brasileiro. Logo, monitorar as atividades que geram impacto ambiental, em particular o desmatamento, é uma tarefa que somente pode ser realizada a contento utilizando-se de sistemas de sensoriamento remoto, em particular, de satélites de observação da Terra. Atualmente, o Brasil conta com dois satélites próprios capazes de gerar imagens dessa natureza: o satélite CBERS-4B, desenvolvido em conjunto com a China em um programa de cooperação (*China-Brazil Earth Research Satellite*) em vigor há mais de 30 anos; e o Amazonia-1, primeiro satélite projetado, desenvolvido e integrado inteiramente no Brasil.

Além disso, o Brasil conta com uma série de programas de monitoramento da região amazônica, sendo os mais conhecidos o PRODES e o DETER. Com base no banco de dados gerado pelos satélites mencionados, esses programas geram informações que servem de subsídio para as autoridades competentes tomarem medidas de combate às atividades ilegais na Amazônia, sejam essas informações alertas diários de foco de desmatamento, ou levantamentos anuais da taxa de desmatamento do bioma.

No entanto, ainda que seja importante que o país tenha fontes próprias de geração de imagens por satélite, os sistemas que o Brasil possui utilizam sensores óticos, ou seja, sensores capazes de produzir imagem no espectro da luz visível de maneira passiva, tendo o Sol como fonte de radiação. Esse é o sistema mais utilizado pelos satélites de observação que atualmente orbitam a Terra; porém, ele conta com algumas desvantagens. Primeiro, o mais evidente, os satélites óticos somente podem gerar imagens durante o dia, o que, a princípio, não é um grande empecilho para o acompanhamento de atividades que ocorrem ao longo de um certo período (como é o caso da expansão de uma região desmatada ao longo de semanas ou meses).

Segundo, devido às características físicas da radiação eletromagnética na faixa do espectro visível, o imageamento ótico é bloqueado pela presença de nuvens. Essa desvantagem é relevante para o nosso trabalho, pois a região amazônica é extremamente úmida, com grande formação de nuvens em determinadas épocas do ano, o que exige, por exemplo, que a equipe de monitoramento faça uma extrapolação para preencher essa lacuna e poder estimar a taxa anual de desmatamento.

Para evitar esses obstáculos, é necessário o uso de sistemas ativos de imagemamento, que utilizam radar para emitir radiação na faixa do micro-ondas. Esses satélites, por terem uma fonte ativa de iluminação, podem produzir imagens a qualquer hora do dia. Além disso, a radiação na frequência do micro-ondas sofre menos interferência de nuvens, contornando esse importante desafio na região amazônica. Ainda que o Brasil

não tenha sistemas próprios de imagemamento por radar, há uma série de programas em outros países com essa capacidade, com destaque para o programa Sentinel, da Agência Espacial Europeia, que fornece um banco de dados de imagens de maneira gratuita.

O imageamento por satélite com radar de abertura sintética (SAR), além de superar esses obstáculos físicos presentes no sensoriamento ótico, tem outras capacidades úteis para o monitoramento da Amazônia. Dependendo da banda de frequência trabalhada, a radiação emitida consegue penetrar na matéria do alvo do imageamento. A banda X, por exemplo, de frequência maior, consegue fazer imagens da parte superior da copa das árvores de uma floresta. Já as bandas S e L, de frequência menor, são capazes de atravessar essa copa, imageando o tronco das árvores ou mesmo o solo da floresta. Essa capacidade é útil,por exemplo, para estimativa da biomassa da região alvo ou, para fins de controle do desmatamento, para detectar descritores dessa atividade, como a abertura de clareiras ou pistas clandestinas.

Ainda que as imagens SAR apresentem essas vantagens, recomendadas para o monitoramento da região amazônica, seu uso ainda demanda etapas de processamento, de modo a tornar a imagem legível para o usuário. Uma dessas etapas é a fase de classificação de imagens, que consiste justamente em indicar se a imagem produzida retratou ou não o objeto desejado (i.e. se a imagem retrata um foco de desmatamento). Essa é uma etapa que consome bastante tempo e é ainda muito dependente de mão de obra humana, o que aumenta a janela de tempo entre o imageamento e a ação prática tomada com base nas informações dessa imagem (o envio de agentes para coibir a atividade ilegal).

Felizmente, houve um avanço significativo - poderia-se dizer até revolucionário - em um campo tecnológico relevante para o problema em tela: a computação visual e o aprendizado de máquina. Computação visual é a grande área de estudo que busca desenvolver a capacidade de captação, processamento e interpretação de informação visual por uma máquina. Aprendizado de máquina (*machine learning*), por sua vez, é uma subárea do campo da inteligência artificial (IA) que busca desenvolver a capacidade de uma máquina de, autonomamente, aperfeiçoar a execução de uma determinada atividade (digamos, a classificação de uma imagem), com base em uma experiência pretérita.

Há várias formas de você desenvolver uma aplicação em aprendizado de máquina, cada qual com suas aplicações e particularidades. A estratégia comum, porém, é trabalhar com um grande banco de dados que sirva para treinar a máquina, de modo que ela consiga extrapolar esse treinamento e ser capaz de realizar sua tarefa diante de uma nova informação. Por exemplo, podemos reunir um banco de dados com diversas imagens SAR da Amazônia, contendo tanto imagens da floresta preservada quanto da floresta com presença de regiões desmatadas, cada qual pré-classificada com esses rótulos (desmatada ou não). Então desenvolvemos um algoritmo que, com base nesse banco de dados, identificará as características em comum entre as imagens de cada conjunto (as imagens de floresta

preservada são mais homogêneas, as imagens com desmatamento tem mais contraste, com bordas bem definidas ao longo da clareira). Assim, na presença de uma nova informação, uma nova imagem, esse algortimo será capaz de classificar a presença do desmatamento ou não com base na presença ou não dessas características aprendidas com base no banco de dados.

Uma ferramenta de aprendizagem de máquina bastante adequada para esse tipo de problema é a rede neural convolucional (*convolutional neural network*, CNN). Redes neurais, um subgrupo dentro do aprendizado de máquina, são algoritmos desenhados de modo a emular o funcionamento do cérebro humano. É composto por uma série de funções aninhadas, que formam uma rede de nós interligados em camadas, as quais podem ser ativadas ou desativadas de maneira análoga à sinapse entre dois neurônios. A rede neural de convolução é um tipo particular de rede neural que utiliza um conjunto dessas camadas para encontrar padrões em comum em partes de uma imagem, como uma janela que vai rastreando essa imagem para encontrar pontos de contraste, identificáveis em mais de um caso de imagens semelhantes, mas não necessariamente idênticas.

Um exemplo clássico é a classificação de rostos humanos. Naturalmente, cada indivíduo tem um rosto diferente, por mais parecidos que duas pessoas sejam. Porém, existe claramente um padrão de rosto humano: a posição dos olhos, nariz e boca, e o formato mais ou menos ovalado do próprio rosto. Ainda que essas características variem de pessoa para pessoa (narizes maiores ou menores, rostos mais finos ou não), as linhas gerais dessas caracteríticas são padronizadas. Treinando um algoritmo como o CNN, é possível identificar em uma imagem qualquer se há um rosto humano em qualquer ponto daquela imagem (um recurso muito utilizado pelos serviços de redes sociais). A mesma ideia é aplicável a outros casos.

A proposta deste trabalho é desenvolver uma rede neural de convolução que seja capaz de classificar imagens SAR da Amazônia de modo a identificar marcadores visuais (como clareiras) que acusem a presença de atividade ilegal na região. Para tanto, iremos olhar com mais detalhe três pontos importantes: o que é SAR e quais são os sistemas disponíveis que produzem esse tipo de imagem; o que é uma rede neural convolucional, no contexto maior do apredizado de máquina; e como funciona a dinâmica de desmatamento na Amazônia e quais são os mecanismos que o Brasil tem para controlar ou mesmo coibir essa atividade.

Em seguida, detalharemos mais a parte metodológica do trabalho: que tipo de banco de dados será utilizado e como ele será adquirido; se haverá necessidade de pré-processamento antes do uso para o treinamento; como será feito esse pré-processamento; de que maneira a rede neural será desenvolvida; como será seu treinamento; e quais serão os parâmetros a serem usados para medir sua eficiência. Por fim, será indicado quais são os resultados esperados uma vez que o algoritmo esteja pronto.

Na sequência, explicaremos a escolha da região alvo imageada (São Félix do Xingu, Pará) para a composição do banco de dados, as características do local e as razões para sua seleção. Passamos a explicar o processo de pré-processamento das imagens no programa SNAP, desde a coleta na plataforma Copernicus, da Agência Espacial Europeia (ESA) até à subdivisão das imagens em diversas amostras, para composição do banco de dados de treinamento da rede. Explicamos o processo de classificação manual de cada amostra, necessário para a realização de uma aprendizagem supervisionada, bem como a divisão desses amostras entre o grupo de treinamento, o de validação e o de teste. Explicamos o desenvolvimento da rede neural convolucional e o seu processo de treinamento.

Por fim, apresentamos os resultados de acurácia alcançado em diversos cenários de teste. O modelo desenvolvido chegou a uma acurácia muito próxima da desejada (89,32%). Discutimos esse resultado, bem como fizemos apontamentos sobre possíveis caminhos que o trabalho pode seguir no futuro. Encerramos o trabalho com uma conclusão que fez uma retrospectiva do trabalho realizado. As referências bibliográficas, os código desenvolvidos e as imagens porduzidas encontram-se na parte final do trabalho.

Parte I

Referencial Teórico

1 Radar de Abertura Sintética (SAR)

Este capítulo será dedicado a explicar as técnicas de observação da Terra, em particular o uso do radar de abertura sintética (SAR). O objetivo não é ser uma discussão exaustiva da ciência do sensoriamento remoto como um todo, uma área do conhecimento deveras extensa, mas sim apresentar os conceitos fundamentais necessários para a compreensão do presente trabalho.

Antes de discutirmos as técnicas de classificação de imagens por satélite, ou mesmo a natureza do objeto a ser observado (a Floresta Amazônica), é importante entendermos o funcionamento da fonte geradora dessas imagens, de modo a compreendermos melhor o que podemos produzir de conhecimento útil a partir dessas imagens.

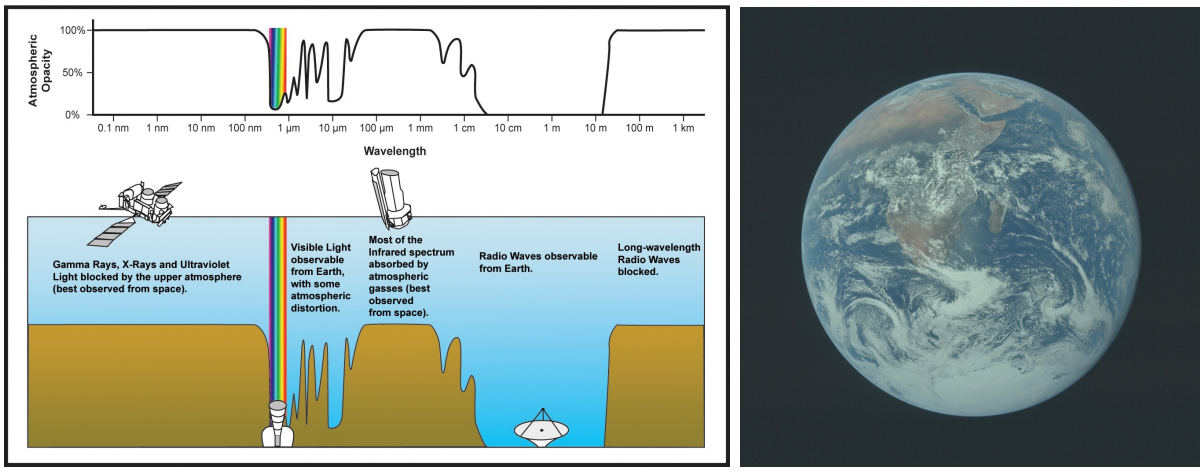
1.1 Sistemas de observação da Terra

No contexto da observação da Terra, *sensoriamento remoto* é a ciência da obtenção à distância de imagens da superfície do planeta "por meio da detecção e medição quantitativa das respostas das interações da radiação eletromagnética com os materiais terrestres"(LILLESAND; KIEFER; CHIPMAN, 2004). Em outras palavras, é pela detecção, por meio de um sensor, da radiação eletromagnética que interage com um objeto que é feito o imageamento da superfície da Terra. Logo, a natureza dessa radiação é essencial para definir esse processo de sensoriamento.

Grosso modo, podemos dividir os sistemas de observação da Terra em duas categorias: sistemas passivos e sistemas ativos. Sistemas passivos utilizam fontes naturais de radiação (i.e. o Sol) para fazer o imageamento, enquanto os sistemas ativos utilizam de uma fonte artificial própria, o que permite, por exemplo, produzir imagens mesmo na ausência de fonte natural (MENESES; ALMEIDA, 2012).

Outra característica relevante é a frequência ou comprimento de onda da radiação que será detectado pelo sensor. Como é sabido, o espectro eletromagnético é dividido em faixas espectrais conforme suas frequências, cada qual com suas características específicas, como as características referentes à interação dessa radiação com a matéria na qual ela incide (HSU, 2019). Em outras palavras, diferentes faixas espectrais comportam-se de maneira diferente ao interagir com um determinado material.

A Figura 1 mostra com mais clareza essa ideia. Podemos observar em (a) que determinadas faixas espectrais são completamente absorvidas pela atmosfera terrestre. Por um lado, isso nos protege da radiação nociva vinda do cosmos; por outro, essa opacidade limita a possibilidade de observarmos a superfície da Terra a algumas janelas espectrais.



(a) (b)

Figura 1 – (a) Opacidade atmosférica (HSU, 2019) e (b) *The Blue Marble* (NASA, 1972)

O espectro da luz visível, por exemplo, é uma dessas janelas (ainda que ela sofra com a opacidade das nuvens, como podemos ver em (b)), pela qual os sistemas passivos de observação utilizam-se de sensores óticos para captar a luz visível proveniente do Sol.

1.1.1 Radar de abertura sintética (SAR)

Ainda com base na Figura 1, percebemos que há outra banda espectral, na faixa do micro-ondas, abaixo do infravermelho em frequência, em que a opacidade da atmosfera é menor, sendo possível a obtenção de imagens da superfície da Terra. É nessa faixa que operam os sistemas ativos de sensoriamento, uma vez que é nessa faixa em que os radares atuam.

No entanto, uma vez que essa é uma banda espectral de frequência menor, comparativamente à luz visível, o comprimento de onda da radiação nesse espectro é proporcionalmente maior, o que exige uma antena de recepção maior (em alguns caos por volta de 10 metros) (MENESES; ALMEIDA, 2012), o que, para um sistema aeroespacial a ser lançado em órbita, é inviável.

Para contornar essa limitação, fora proposto um sistema capaz de emular o alcance de um antena desse porte, manobrando o satélite para que o sensor que irá captar o feixe refletido pela superfície da Terra se comporte como uma antena virtual (ou sintética), de mesma dimensão. Esse é o funcionamento do Radar de Abertura Sintética, ou SAR (*syntethic array radar*). A partir das plataformas SAR é que passou a ser viável o imageamento na faixa do micro-ondas (MENESES; ALMEIDA, 2012).

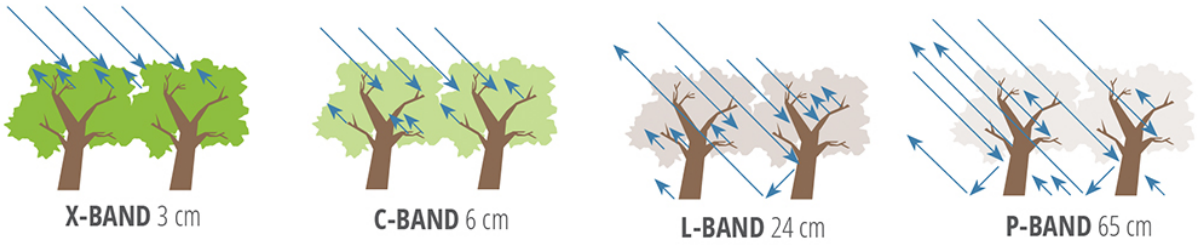


Figura 2 – Penetração do imageamento SAR (NASA, 2020)

1.2 Bandas SAR (X, C, S, L, P)

Uma vantagem importante do imageamento em micro-ondas é que as ondas de maior comprimento penetram mais em um dado material, dependendo de suas características (NASA, 2020). No caso da faixa SAR, essa característica é particularmente interessante para imagens de regiões com vegetação (cf. Figura 2).

Para ficar mais clara essa ideia, a Figura 3 mostra a divisão da faixa espectral SAR em bandas (X, C, S, L P). Os sistemas que operam na banda X, mais comuns atualmente, conseguem imagear somente a parte superior da copa das das árvores. Já os que operam nas bandas C ou S conseguem passar por essas copas, imageando o tronco das árvores.

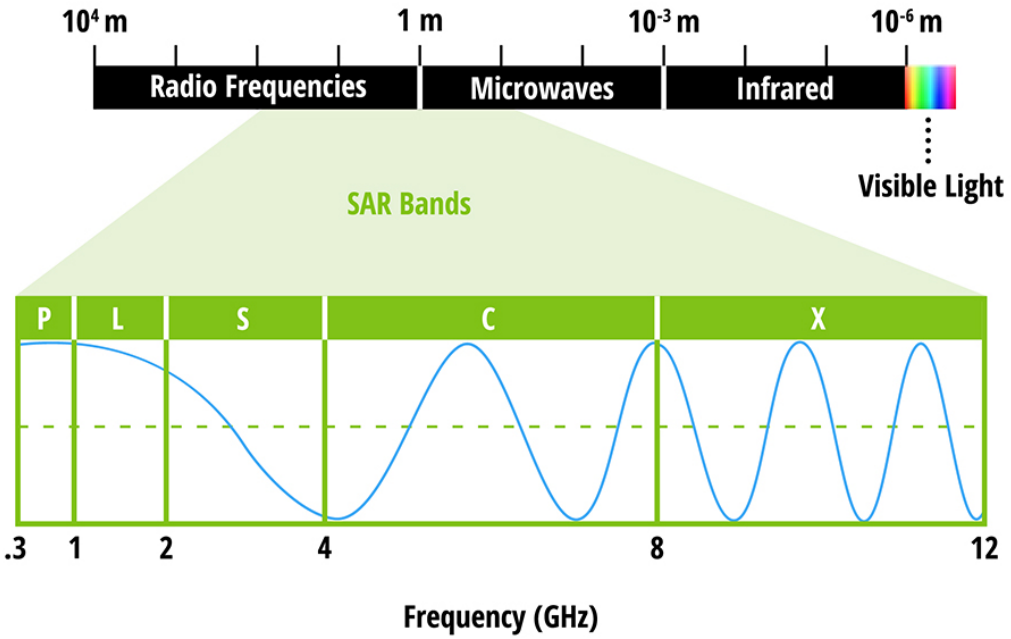


Figura 3 – Bandas espectrais SAR (NASA, 2020)

1.3 Imageamento SAR

As imagens feitas por uma plataforma SAR são compostas com base nas informações coletadas do sinal emitido que interagiu com o alvo e retornou ao sensor do satélite. Informações da onda como comprimento, amplitude, fase e polarização são relevantes para caracterizar a natureza do alvo, que pode ser representada em uma imagem de escala de cinza pouco discernível ao olho humano. Isso torna a interpretação de uma imagem SAR reconhecidamente menos intuitiva do que uma imagem ótica.

Para fins deste trabalho, será dado enfoque a duas características do sinal enviado e recebido por uma plataforma SAR: a polarização da onda e a forma de interação com o alvo, conhecido como retroespalhamento. O sinal, no entanto, é mais complexo, contendo informações de fase, quadratura e coerência que podem ser usadas para outras análises do alvo, que fogem ao escopo do trabalho (DINIZ, 2019).

1.3.1 Polarimetria

O sinal emitido e recebido por uma plataforma SAR é uma onda eletromagnética, a qual vibra em dois planos perpendiculares principais e que pode ser polarizada em referência em um desses planos ou de outro modo (cf. Figura 4). Por convenção, a polarização com base em um desses planos é chamada de polarização horizontal ou vertical.

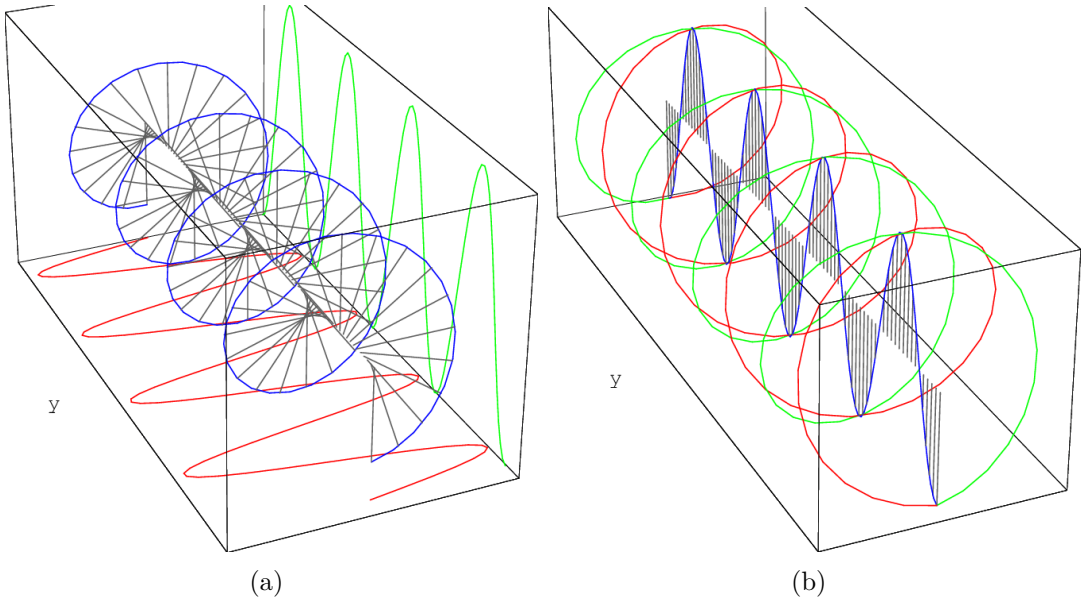


Figura 4 – (a) Polarização circular decomposta em polarizações lineares e (b) Polarização linear decomposta em polarizações circulares (BECKERT; FALCKE, 2002)

Essas polarizações definem os modos de aquisição de imagem SAR, a depender de como o sinal emitido ou o sinal recebido é polarizado:

Polarização paralela: tanto o sinal emitido quanto o sinal recebido estão na mesma polarização, seja a vertical (modo VV), seja a horizontal (modo HH).

Polarização cruzada: os sinais emitido e recebido estão em polarizações distintas (VH ou HV).

O modo de polarização do sinal é relevante, pois alvos distintos interagem de maneiras distintas com cada um dos modos, o que pode servir para caracterizar esses alvos, como veremos adiante quando abordarmos o retroespalhamento do sinal (DINIZ, 2019).

1.3.2 Mecanismos de retroespalhamento

A onda eletromagnética emitida pela plataforma SAR interage com o alvo de maneiras distintas, a depender da natureza do material que compõe esse alvo. A forma como esse sinal é refletido por uma superfície, por exemplo, irá depender da rugosidade dessa superfície. Um objeto tridimensional, aquele em que um sinal consegue penetrar em sua composição interna, também irá interagir com esse sinal de maneira característica.

Em suma, o espalhamento da onda pelo alvo que for refletido de volta para a plataforma SAR (o retroespalhamento) pode revelar características relevantes desse alvo. Nesse sentido, três mecanismos de espalhamento são mais conhecidos: o espalhamento superficial, o de reflexão dupla (*double bounce*) e o volumétrico (DINIZ, 2019). O primeiro ocorre em superfícies planas (rugosas ou não) e tendem a interagir com o alvo apenas uma vez, quando são refletidas de volta. O segundo ocorre na presença de superfícies perpendiculares, como prédios em um centro urbano, o que permite que o sinal seja refletido duas vezes antes de voltar ao sensor do satélite. O terceiro é associado a alvos tridimensionais, com alguma penetração do sinal em seu interior, como copa de árvores (cf. Figura 5).

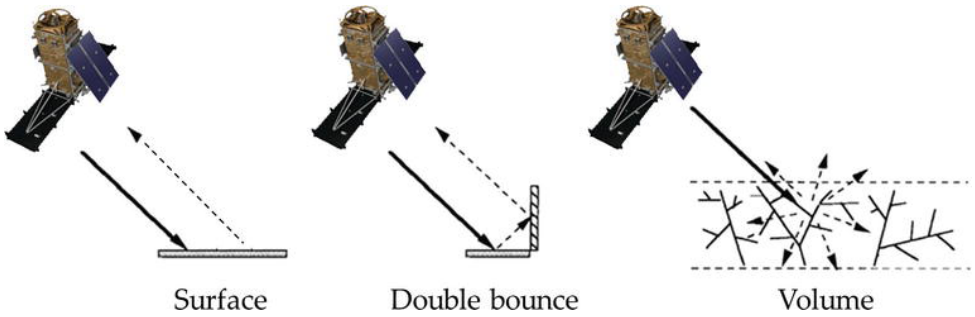


Figura 5 – Principais mecanismos de espalhamento (DABBOOR; BRISCO, 2018)

Diferentes polarizações correspondem mais a determinados mecanismos. As polarizações paralelas são mais associadas ao espalhamento superficial e ao *double-bouncing* e as polarizações cruzadas, ao espalhamento volumétrico. Essa característica é particu-

larmente útil para o presente trabalho, como veremos quando formos distinguir regiões desmatadas de áreas florestais.

1.3.3 Coeficiente de Retroespalhamento

Como visto, o sinal de um imageador carrega várias informações da onda, como fase e quadratura, que pode fornecer dados relevantes do alvo. Porém, para fins deste trabalho, vamos nos ater a um dado específico que é o coeficiente de retroespalhamento (σ^0). Esse coeficiente é a quantidade de energia refletida pela seção transversal para o receptor (σ) por unidade de área (DINIZ, 2019):

$$\sigma^0 = \frac{\sigma}{A} \tag{1.1}$$

Essa seção transversal, ou área do retroespalhamento efetivo do alvo é encontrada pela equação do radar, em que a potência do sinal recebida (P_r) é diretamente proporcional à potência transmitida (P_t), ao ganho da antena de transmissão (G_t), a área da antena de recepção (A_r) e ao σ, e inversamente proporcional à distância do alvo (R) à quarta potência:

$$P_r = \frac{P_t G_t A_r \sigma}{(4\pi)^2 R^4} \tag{1.2}$$

Dado que a plataforma imageadora tema a informação das outras variáveis, o valor de σ pode ser calculado para cada unidade elementar da área imageada (píxel), formando uma matriz de valores numéricos que, se divididos pela unidade de área para calcular σ^0, resultará em valores entre 0 e 1, que podem ser usados para compor uma imagem digital em escala de cinza.

Como diferentes polarizações interagem de maneira diferente com o alvo, diferentes modos (VV/HH, VH/HV) resultarão em diferentes coeficientes de retroespalhamento para um mesmo alvo. Essas diferentes matrizes de valores de σ^0 podem ser associadas a canais de um arquivo raster, formando uma imagem que pode ser trabalhada em programas de georreferenciamento ou, no caso deste trabalho, ser usada para classificação de imagem por meio de uma rede neural.

1.3.4 Índices Polarimétricos

Uma distinção importante do imageamento SAR em comparação com o imageamento ótico é a quantidade de bandas espectrais disponíveis em um dado sistema. Enquanto no imageamento ótico é possível trabalhar com várias faixas espectrais em uma mesma plataforma, o que permite realizar análises multiespectrais, como cálculos de ín-

dices, o mesmo não pode ser feito com imagem SAR, que conta com apenas uma banda espectral (X, C, L, P, a depender da plataforma).

No entanto, é possível fazer o cálculo de índices com base na resposta polarimétrica, como por exemplo com os diferentes coeficientes de retroespalhamento de cada modo de polarização. Uma plataforma que contenha todos os 4 modos certamento será capaz de fornecer as informações necessárias para os mais diversos índices; porém, mesmo com plataformas com apenas dois modos, como é o caso do satélite Sentinel-1, é possível fazer esse tipo de análise (DINIZ, 2019).

Um índice particularmente interessante para o presente trabalho é a razão cross-polarizada, que é a razão entre o coeficiente de retroespalhamento no modo VH pelo coeficiente do modo VV, que são justamente os modos presentes no Sentinel-1.

$$RC_2 = \frac{\sigma^0 VH}{\sigma^0 VV} \tag{1.3}$$

Esse índice é utilizado para realizar a discriminação entre o espalhamento superficial e o volumétrico, aplicável para casos em que se busca distinguir regiões florestadas e regiões desmatadas.

Mais do que isso, se pegarmos as matrizes de coeficientes VH e VV, juntamente com a matriz das razões cross-polarizadas correspondentes, e usarmos como canais de um arquivo raster, podemos compor uma imagem RGB passível de utilização em uma rede neural que classifique imagens coloridas de uma maneira geral.

1.3.5 Ruído *Speckle*

Um último aspecto relevante do imageamento SAR a ser abordado é a presença de ruídos na formação da imagem, em especial o ruído *speckle*. Esse é um ruído multiplicativo que é formado pela interação dos feixes coerentes do sinal e levam a variação de ecos nas células de resolução (DINIZ, 2019). O resultado é a formação aleatória de áreas claras e escuras na imagem, dando um aspecto granulado (efeito sal e pimenta).

O ruído *speckle* é uma das razões que dificultam a interpretação visual de uma imagem SAR da forma como ela é composta pela plataforma imageadora. Para coletar informações relevantes dessas imagens, elas precisam passar por pré-processamentos. No caso do ruído *speckle*, é necessário aplicar um filtro que selecione porções da imagem e retirem um valor médio dos píxeis contidos nessa janela. Esse valor médio vai depender do tipo de filtro que se está aplicando, seja um filtro no domínio da frequência (filtro *multilooking*), seja um filtro no domínio do espaço, que se subdividem em filtros não adaptativos (média, mediana, moda) e filtros adaptativos (Lee, Gamma) (DINIZ, 2019).

O pré-processamento de imagens SAR ficará mais claro na parte metodológica.

1.4 Plataformas SAR

É importante termos o conhecimento de quais plataformas capazes de produzir imagens SAR estão atualmente em órbita, bem como projetos a serem lançados num futuro próximo. A Tabela 1 traz informações sobres esses satélites, com base nos dados disponíveis no Portal de Observação da Terra Agência Espacial Europeia (ESA).

Satélite	Agência/Empresa	País/Região	Banda SAR	Lançamento
RADARSAT-2	CSA	Canadá	C	2007
TerraSAR-X	DLR	Alemanha	X	2007
TanDEM-X	DLR	Alemanha	X	2010
KOMPSAT-5	KARI	Coreia do Sul	X	2013
ALOS-2	JAXA	Japão	L	2014
PAZ	INTA	Espanha	X	2014
Sentinel-1	ESA	Europa	C	2014
Gaofen-3	CNSA	China	C	2016
Asnaro-2	JAXA	Japão	X	2018
Capella 1/Denali	Capella Space	EUA	X	2018
COSMO-SkyMed	ASI	Itália	X	2018
NovaSAR-S1	UKSA	Reino Unido	S	2018
SAOCOM 1	CONAE	Argentina	L	2018
BIOMASS	ESA	Europa	L	2022
Tandem-L	DLR	Alemanha	L	2022

Tabela 1 – Satélites SAR de Observação da Terra (ESA, 2021)

- RADARSAT-2: missão conjunta entre a Agência Espacial Canadense (CSA) e a empresa MacDonald Dettwiler. Propósito de monitoramento do meio ambiente, gerenciamento de recursos naturais e vigilância da costa marítima.

- TerraSAR-X: Missão conjunta entre a Agência Espacial Alemã (DLR) e a empresa EADS Astrium. Tem o duplo propósito de produzir dados científicos (geológicos, oceanográficos), bem como estimular o mercado de observação da Terra.

- TanDEM-X: Desenvolvido para trabalhar em conjunto com o TerraSAR-X. Tem o propósito de produzir dados para o modelo digital de elevação (DEM) da superfície da Terra.

- KOMPSAT-5: Missão do Instituto de Pesquisa Aeroespacial da Coreia do Sul (KARI) para observação de recursos naturais e vigilância de desastres naturais.

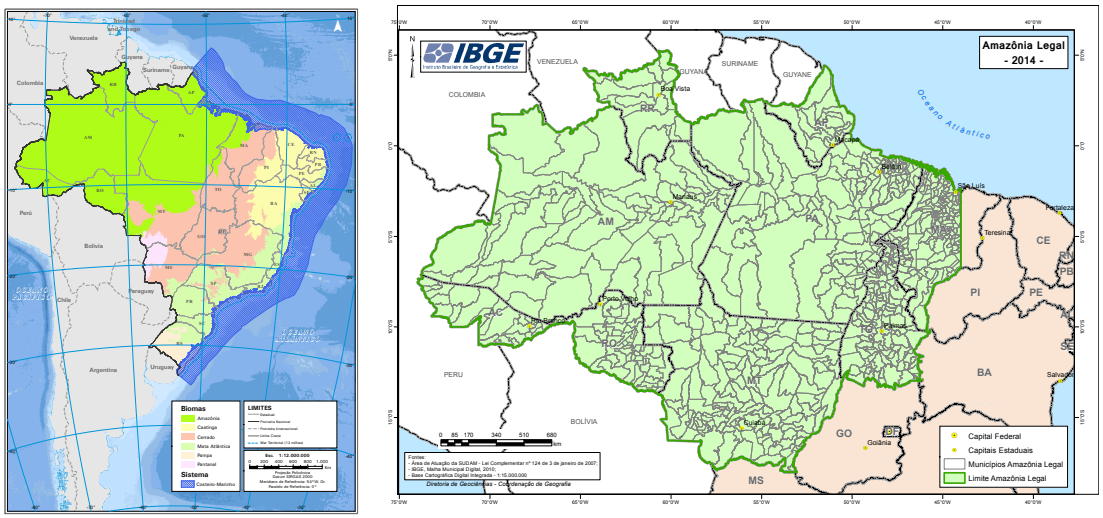
- ALOS-2: Um dos mais avançados sistemas SAR atualmente em órbita. Desenvolvido pela Agência Espacial Japonesa (JAXA) para vigilância de desastres, monitoramento de recursos terrestres, agricultura e monitoramento global de florestas.

- PAZ: Baseado no projeto TerraSAR-X.Possui finalidade dual (civil/defesa), tendo sido parcialmente financiado pelo Ministério da Defesa da Espanha.
- Sentinel-1: Parte da família de satélites GEOSS (*Global Earth Observation System of Systems*). Uma das principais fornecedoras de imagens do banco da agência europeia ESA, o qual servirá de fonte para o presente trabalho.
- Gaofen-3: Primeiro satélite civil desenvolvido pela agência chinesa (CSNA) a usar SAR banda C. Usado para monitoramento de desastres, meteorologia e recursos hídricos.
- Asnaro-2: Projeto desenvolvido pela agência japonesa em continuação do projeto anterior Asnaro, que era um satélite ótico.
- Capella 1/Denali: Primeira constelação de satélites SAR desenvolvido totalmente por uma empres privada, a Capella Space, de Palo Alto, Califórnia, EUA. Composta por 36 microssatélites em órbita baixa (500km de altitude). Funciona na banda X.
- COSMO-SkyMed: Constelação desenvolvida pela Agência Espacial Italiana e o Ministério da Defesa da Itália, possui finalidade dual (civil/defesa). Um acordo de cooperação permite que o governo brasileiro (CENSIPAM) tenha acesso ao banco de imagens produzido por esse sistema (SILVA et al., 2019)
- NovaSAR-S1: Sistema de observação desenvolvida pela Agência Espacial do Reino Unido (UKSA), um dos únicos a funcionar na banda S. Voltado para o monitoramento de desastres, agricultura e observação de florestas e recursos hídricos.
- SAOCOM 1: Satélite SAR desenvolvido pela agência argentina (CONAE), o único exemplar de um país do hemisfério sul. Funciona em banda L, com maior penetração nos objetos observados. Lançado por um foguete Falcon 9, da SpaceX.
- BIOMASS: Projeto ambicioso da Agência Espacial Europeia (ESA) de monitoramento de florestas de modo a mensurar o ciclo global de carbono, importante para entender os mecanismos de emissão de gases de efeito estufa, em particular aqueles ligados ao desflorestamento. Previsto para ser lançado em 2022, quando estiver disponível certamente será uma das principais ferramentas de monitoramento da gestão e preservação do bioma amazônico, bem como outras regiões florestais ameaçadas do planeta.
- Tandem-L: Continuação do trabalho da agência alemã com TanDEM-X, outro sistema previsto para atuação na mensuração do ciclo de carbono nas florestas, além de observação de fenômenos de glaciação, oceanografia, bem como observação tridimensional de formação de florestas, corpos de gelo e deformações na superfície da Terra com precisão milimétrica.

2 Indicadores de desmatamento

2.1 A Floresta Amazônica

A Amazônia é o principal bioma do território brasileiro e a principal reserva florestal tropical do mundo. Ocupa uma área de 4.212.742 km^2, o que corresponde a 49,5% do território do país (IBGE, 2019). Sua vegetação predominante é classificada como floresta ombrófila densa (46,18% do bioma) e floresta ombrófila aberta (22,68% do bioma), que se caracterizam por apresentar cobertura vegetal perene e dossel (extrato superior da floresta) alto, chegando a 50m de altura (que são características relevantes para o imageamento aéreo). Para fins de gestão territorial, o governo brasileiro adota o conceito de Amazônia Legal (cf. Figura 6), delimitação que incorpora uma área maior que o bioma amazônico (5,5 milhões de km^2), por englobar, por exemplo, a totalidade dos estados do Mato Grosso e Tocantins e grande parte do estado do Maranhão (IBGE, 2014). As políticas públicas de uso do território amazônico, bem como de preservação desse bioma, tomam por base esse recorte geográfico.



(a) (b)

Figura 6 – (a) Biomas brasileiros e (b) Amazônia Legal (IBGE, 2014)

Uma das principais atividades econômicas da região é a produção agropecuária, organizada sob regime de fronteiras agrícolas (IBGE, 2021a), no qual a produção se expande a partir de uma região tradicional (como a Região Centro Sul do país) em direção a terras disponíveis para atividade agropecuária. No estado do Mato Grosso, por exemplo, é observável o aumento da atividade agrícola, particularmente a produção intensiva de grãos, na região norte do estado (onde se localiza a parte do bioma amazônico que fica

nesse estado), seguindo o corredor de integração aberto pela BR-163 (Cuiabá-Santarém). O município de Lucas do Rio Verde, por exemplo, é responsável sozinho por 10% da produção de soja no Brasil (IBGE, 2021a).

Esse tipo de atividade econômica causa um grande impacto ambiental na região, uma vez que a fronteira agrícola é aberta por meio do desflorestamento da área ocupada, em geral por meio da queimada da vegetação para abertura do pasto, ou posteriormente para a área de plantio. Não somente essa ocupação intensiva afeta diretamente o ecossistema local, mas também contribui para a mudança climática, uma vez que essas queimadas são um dos principais contribuidores da emissão de gases de efeito estufa no Brasil (IBGE, 2015).

2.2 Desmatamento na Amazônia

A dinâmica de desmatamento da Amazônia é bastante característica, tendo em vista que a floresta é bastante densa, como vimos na caracterização do bioma, e o acesso a ela é, inicialmente, bastante limitado. O primeiro passo para a ação de desmatamento é, portanto, a abertura de caminhos e estradas na floresta, a partir de suas bordas, de modo a dar acesso aos extrativistas a regiões antes inacessíveis, em um processo conhecido como fragmentação da floresta (SILVA et al., 2019).

Esse processo de fragmentação florestal decorre daquilo que é intitulado como desmatamento por degradação florestal (INPE, 2019), em que há uma modificação da paisagem florestal de maneira gradativa, com o corte seletivo de determinadas árvores (mais nobres e de maior valor econômico), tornando essa paisagem mais heterogênea, se comparada com o padrão prévio da floresta preservada. Essa degradação altera o microclima da região, reduzindo sua umidade e tornando-a mais propensa a queimadas (SILVA et al., 2019), as quais são responsáveis pela retirada do restante da cobertura vegetal, liberando o solo para destinação a atividades agropecuárias (INPE, 2019).

Diante dessa dinâmica característica, é possível apontar essa abertura inicial de estradas e clareiras como um índice precursor de um foco de desmatamento. Quanto mais cedo esses índices forem identificados, menor será o intervalo de tempo para que as autoridades competentes tomem as medidas cabíveis e evitem as etapas seguintes da atividade de desmatamento (cf. Figura 7).

2.3 Sistemas de Monitoramento da Amazônia no Brasil

Cabe definirmos quais são os mecanismos de monitoramento e controle do desmatamento da Amazônia disponíveis no Brasil, de modo a ver em qual contexto o presente trabalho pode ser inserido. O principal órgão reponsável pelo sensoriamento remoto no

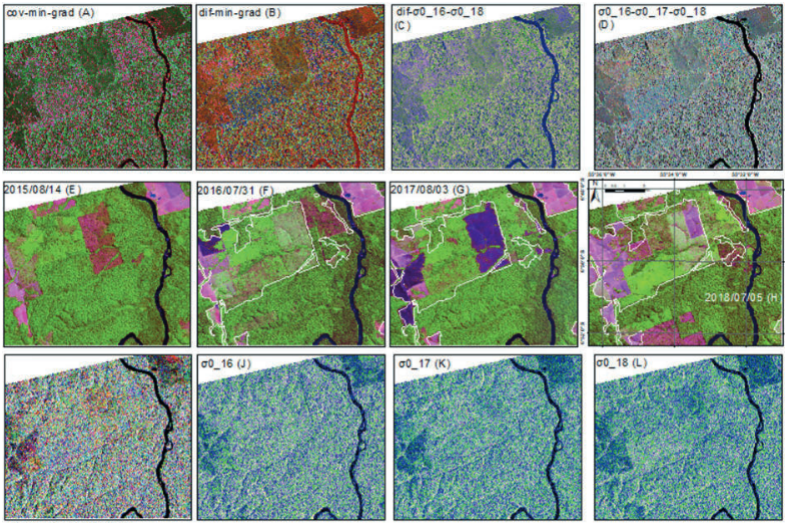


Figura 7 – Imagens SAR no município de Novo Progresso, com polígonos do desmatamento (SILVA et al., 2019)

país é o Instituto Nacional de Pesquisas Espaciais (INPE), o principal órgão de execução para o desenvolvimento das pesquisas espaciais no âmbito civil e cuja finalidade é "realizar pesquisas científicas, desenvolvimento tecnológico, atividades operacionais e capacitação de recursos humanos nos campos da Ciência Espacial e da Atmosfera, da *Observação da Terra*, da Previsão de Tempo e Estudos Climáticos, da Engenharia e Tecnologia Espacial e áreas do conhecimento correlatas, consoante à política definida pelo Ministério [da Ciência, Tecnologia e Inovação]"(Regimento Interno do Instituto Nacional de Pesquisas Espaciais, artigo 4º., grifo nosso). Porém, conforme veremos adiante, há também outras iniciativas de monitoramento da Amazônia fora do âmbito do INPE.

2.3.1 PRODES

O Programa de Monitoramento do Desmatamento da Floresta Amazônica Brasileira por Satélite (PRODES) é o projeto do INPE responsável pelo cálculo anual da taxa de desmatamento na Amazônia. Em funcionamento desde 1988, o PRODES faz uso das imagens fornecidas pelos satélites LANDSAT 8/OLI, CBERS 4 e IRS-2 (INPE, 2020).

Esses satélites contam com sensores óticos, o que acarreta os desafios já mencionados, decorrentes de áreas cobertas por nuvens, que precisam ser estimadas, de modo a ser realizado o cálculo da taxa de desmatamento para o período de um ano. Outro limitador para o sensoriamento feito por esse programa é que ele é capaz somente de fazer a detecção do corte raso da floresta, ou seja o processo de remoção completa da cobertura florestal (INPE, 2019). Os dados fornecidos pelo PRODES servem de base para o governo federal realizar as políticas públicas para o tema da preservação do bioma Amazônico.

2.3.2 DETER / DETER-2

O DETER é o programa do INPE de alertas diários de focos de desmatamento da Amazônia, que servem de suporte aos órgãos de fiscalização, em particular o Instituto Brasileiro do Meio Ambiente e dos Recursos Naturais Renováveis (IBAMA). Os alertas são emitidos com base nos dados fornecidos pelo PRODES e, embora ambos os programas utilizem a mesma base de dados, o DETER não pode servir de comparativo para taxa de desmatamento mensal (INPE, 2021a).

2.3.3 DETER Intenso

Diante da percepção das limitações trazidas pelos sensores óticos dos satélites tradicionalmente utilizados pelo INPE, o instituto decidiu em 2019 avançar nos seus mecanismos de monitoramento, por meio da integração dessas imagens óticas com as imagens SAR fornecidas pelo satélite Sentinel-1. Além de ampliar o escopo da faixa espectral do imageamento, o DETER Intenso também utiliza-se de ferramentas de tecnologia de informação, como o uso de computação em nuvem e o desenvolvimento de um algoritmo de priorização de alvos, no intuito de otimizar seus trabalhos, o que permitiu, por exemplo, reduzir a taxa de revisitação para 1 ou dois dias (INPE, 2021b).

Das ferramentas disponíveis no Brasil para o monitoramento do desmatamento na Amazônia, o DETER Intenso é claramente o mais relevante para o presente trabalho. Infelizmente, devido a essa ferramenta ainda estar em fase de consolidação, suas informações ainda não estão abertas para o público geral, sendo destinadas exlusivamente para os órgãos de fiscalização.

2.3.4 Terra Amazon

O TerraAmazon é uma ferramenta de sistema de imageamento georreferenciado (GIS) desenvolvida pelo INPE usada no monitoramento da Amazônia. É utilizada pelo instituto nos seus diversos projetos, como o PRODES e o DETER, e está aberta para o uso do público em geral (INPE, 2021c).

2.3.5 Terra Brasilis / Terra Class

TerraBrasilis é o Portal de divulgação dos dados de monitoramento de vegetação do INPE (ASSIS et al., 2019).

2.3.6 Amazônia SAR / Sipam SAR

Iniciativa de monitoramento da Amazônia que não é conduzida pelo INPE, e sim pelo Centro Gestor e Operacional do Sistema de Proteção da Amazônia (Censipam),órgão

ligado ao Ministério da Defesa. Esse órgão é formado por agentes de diversos órgãos federais que atuam nessa área de preservação, como o Departamento da Polícia Federal (DPF), o IBAMA, o Instituto Nacional de Colonização e Reforma Agrária (INCRA) e o Instituto Chico Mendes de Convervação da Biodiversidade (ICMBio). Faz uso de imagens SAR fornecidas pelo satélite COSMO-SKyMed (DEFESA, 2020).

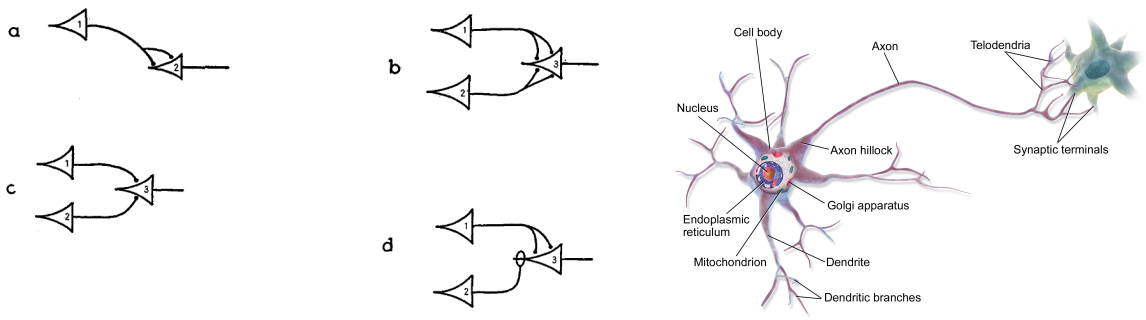
3 Redes Neurais Convolucionais (CNN)

Neste capítulo iremos explanar sobre as ferramentas computacionais que serão utilizadas no trabalho. É importante termos claros os conceitos de aprendizagem de máquina (*machine learning*), redes neurais (*neural networks*) e visão computacional, bem como em que sentido estes serão utilizados para a classificação das imagens.

3.1 Redes Neurais Artificiais

Grosso modo, uma rede neural artificial (ANN, em inglês) é um algoritmo cuja arquitetura emula o funcionamento do sistema nervoso biológico. Do mesmo modo que o voo das aves inspirou a invenção do avião, ainda que seu funcionamento seja distinto (ANDERSON, 1989), imitar mecanismos da natureza faz parte do desenvolvimento tecnológico desde há muito tempo.

No caso das ANN, a primeira proposição foi feita em 1943 pelos matemáticos Warren S. McCulloch e Walter Pitts no artigo *A logical calculus of the ideas immanent in nervous activity* (MCCULLOCH; PITTS, 1943). O modelo proposto era composto por uma ou mais entradas binárias (ligado/desligado) seguidas por uma única saída binária, o que era capaz de representar qualquer tipo de proposição lógica ('e', 'ou', 'não' e suas combinações), como pode ser visto na Figura 8(a). A ideia é um neurônio artificial enviar um sinal artificial binário ao neurônio seguinte, do mesmo modo que uma sinapse biológica (cf. 8(b)).



(a) ANN representando portas lógicas (MCCULLOCH; PITTS, 1943)

(b) Neurônio biológico (GéRON, 2017)

Figura 8 – Redes neurais artificiais e biológicas.

3.1.1 Perceptron

Em 1958, o psicólogo Frank Rosenblatt evoluiu o trabalho com ANN ao propor o modelo que ele chamou de perceptron (ROSENBLATT, 1958). Essa arquitetura é baseada em um neurônio artificial diferente do proposto por McCulloch e Pitts, em que as entradas (*inputs*) não são apenas sinais binários, mas números $(x_1, x_2, ...x_n)$ a serem somados de maneira ponderada, conforme pesos distintos $(w_1, w_2, ...w_n)$, soma essa servindo de entrada para uma função degrau qualquer, comumente a função Heaviside. Esse tipo de neurônio é chamado de unidade de limite linear (LTU, *linear threshold unit*) (GéRON, 2017).

$$z = w_1x_1 + w_2x_2 + ... + w_nx_n = \mathbf{w^T} \cdot \mathbf{x} \tag{3.1}$$

$$H(z) = \begin{cases} 0 & z \leq 0 \\ 1 & z \geq 0 \end{cases} \tag{3.2}$$

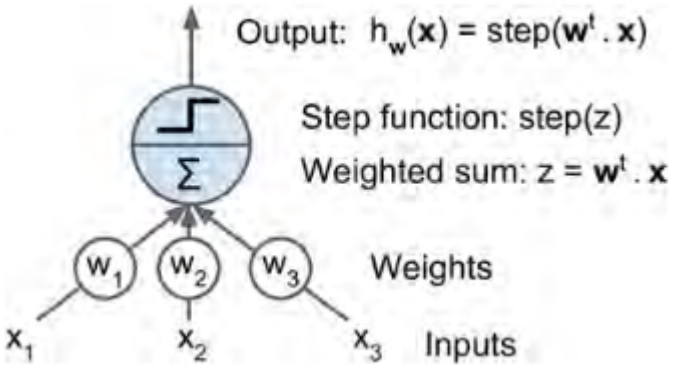


Figura 9 – Unidade de limite linear (LTU) (GéRON, 2017)

O perceptron é formado por uma camada de vários LTUs que são cada um ligados a todos os sinais de entrada, mais um neurônio de viés (*bias neuron*), que garante mais um grau de liberdade ao modelo (cf. Figura 10).

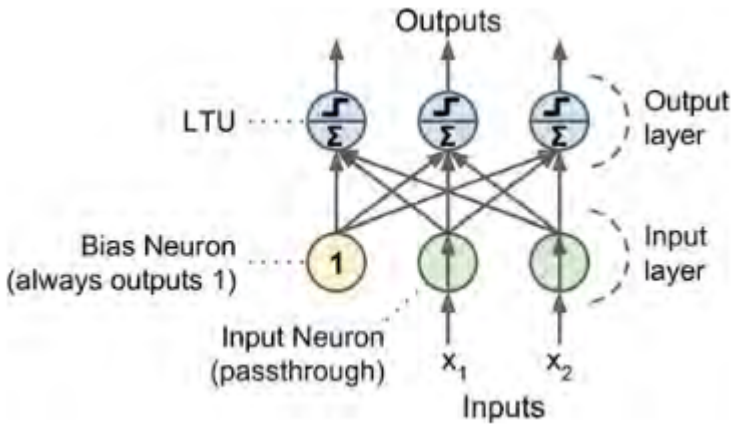


Figura 10 – Perceptron com neurônio de viés igual a 1 (GéRON, 2017)

Importante ressaltar que os pesos w_i definem quais sinais de entrada são relevantes para o sinal de saída (*output*) desejado. Assim, em se tratando de treinamento da rede, usa-se sinais de entrada e de saída conhecidos, de modo a encontrar esses pesos, os quais serão utilizados em sinais de entrada de teste que resultarão em sinais de saída cujo padrão será aquele aprendido nos sinais de entrada de treino.

É possível ampliar o perceptron proposto por Rosenblatt e utilizar mais de uma camada de neurônios LTU, conforme podemos ver na Figura 11.

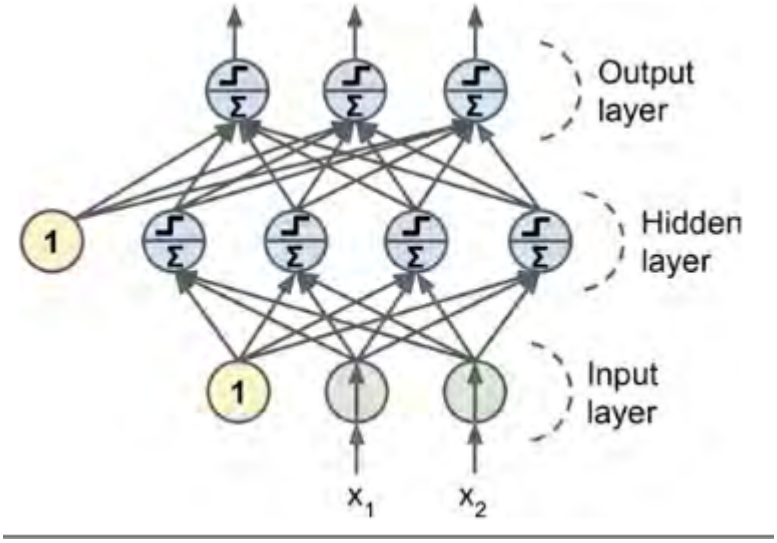


Figura 11 – Perceptron com mais de uma camada (GéRON, 2017)

Uma rede perceptron de mais de uma camada é conhecida como rede neural profunda (*deep neural netwotk, DNN*) (GéRON, 2017).

3.1.2 Função de ativação

Na seção anterior, fora explicado que uma rede neural simples consiste na soma ponderada de vários sinais de entrada que alimenta uma função degrau cujo resultado é o sinal de saída. Essa função é conhecida como função de ativação, pois é ela que ativa ou não o sinal de saída, conforme a forma da função degrau (no caso da função Heaviside, quando $z \geq 0$). Porém, há outras funções de ativação que podem ser utilizadas, mais recomendadas para redes neurais multicamadas (GéRON, 2017).

- função logística (sigmoide): $\sigma(z) = 1(1 + e^{-z})$
- função tangente hiperbólica: $tanh(z) = 2\sigma 2z - 1$
- *Rectified Linear Unit* (ReLU): $H(z) = \begin{cases} 0 & z \leq 0 \\ z & z \geq 0 \end{cases}$

3.1.3 Treinamento da Rede

O propósito de um algoritmo de aprendizagem de máquina é executar uma tarefa sem ter sido explicitamente programado para ela, ou, em outras palavras, "aprender da experiência E referente a alguma tarefa T e alguma medida de performance P, de modo que a performance em T, medida por P, melhora com a experiência E"(SAMUEL, 1959; MITCHELL, 1997). Assim, é importante entendermos o método de treinamento/aprendizagem de uma rede neural artificial.

3.1.3.1 Funções de Erro (*Loss Functions*)

Para fins do presente trabalho, vamos nos ater ao aprendizado supervisionado, em que os dados de treinamento contém os valores de saída desejado (GéRON, 2017). Desse modo, treinar a rede significa encontrar valores para os pesos w_i (como vimos com o perceptron) que minimizem o erro entre o sinal de saída $\hat{y}$ da rede e o valor real y indicado pelos dados de treinamento. Para medir o erro de n sinais de entrada, algumas funções de erro (*loss function*) podem ser usadas, como o erro médio ao quadrado (MSE, Equação 3.3), o erro da raiz média ao quadrado (RMSE, Equação 3.4) e o erro médio absoluto (MAE, Equação 3.5) (AGGARWAL, 2018).

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \tag{3.3}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2} \tag{3.4}$$

$$MAE = \frac{\sum_{i=1}^{n}|\hat{y}_i - y_i|}{n} \tag{3.5}$$

3.1.3.2 Descida Gradiente (*Gradient Descent*)

Para encontrar o valor mínimo de erro, basta encontrar o valor mínimo da função de erro, ou seja, encontrar o ponto em que a derivada da função seja igual a zero. Para entendermos melhor, vamos pegar como referência a Equação 3.3. Vamos também, para fins de nomenclatura, nomear os pesos de cada neurônio de uma dada rede como θ. Assim, o valor de saída da rede $\hat{y}$ pode ser definido como

$$\hat{\mathbf{y}} = \theta^T \cdot \mathbf{x} \tag{3.6}$$

em que θ é um vetor dos parâmetros (pesos) de cada neurônio e x é o vetor (ou matriz de vetores, dependendo da dimensão dos dados de entrada (GéRON, 2017). Aplicando a Equação 3.6 na Equação 3.3, temos

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\theta^T \cdot \mathbf{x^{(i)}} - y^{(i)})^2 \tag{3.7}$$

Como são os valores de θ que nos interessam (para quais valores de θ teremos a função de erro em seu mínimo), derivamos a função para esses valores:

$$\frac{\partial}{\partial\theta_j}MSE(\theta) = \frac{2}{m}\sum_{i=1}^{m}(\theta^T \cdot \mathbf{x}^{(i)} - \mathbf{y}^{(i)})x_j^{(i)} \tag{3.8}$$

A derivada parcial mostrada na Equação 3.8 é calculada de maneira iterativa, usando o vetor gradiente de j termos $\nabla_\theta MSE(\theta)$ (GéRON, 2017)

$$\theta^{(nextstep)} = \theta - \eta\nabla_\theta MSE(\theta) \tag{3.9}$$

em que η é o grau de aprendizagem (*learning rate*). A esse processo iterativo de encontrar o valor mínimo da função de erro é dado o nome de descida gradiente (*gradient descent*). Interessante notar que esse processo, por trabalhar com derivadas, demanda que a rede contenha funções de ativação com derivadas bem definidas (o que justifica o uso de sigmoide, por exemplo, em vez de uma função degrau, como nos perceptrons tradicionais).

3.1.3.3 Retropropagação (*Backpropagation*)

Em redes neurais de uma única camada, esse processo de descida gradiente é bastante direto. Porém, em redes multicamadas, é necessário usar outros métodos, como a retropropagação (*backpropagation*). Por esse método, em uma fase direta (*forward phase*) o sinal de saída é calculado com base em um dado valor para os pesos θ, e esse resultado é comparado com os valores pré-classificados (*labelled*) dos dados de treinamento, bem como os valores da derivada da função de perda. Depois, numa fase inversa (*backward phase*), o gradiente da função de perda é calculado a partir dos dados de saída e usado para calcular novos valores para os pesos θ da rede. O processo ocorre de maneira iterativa (AGGARWAL, 2018).

3.2 Rede Neural Convolucional

As redes neurais convolucionais (*convolutional neural network, CNN*) são um tipo de rede neural desenvolvida no campo da computação visual, área do conhecimento que visa "criar sistemas autônomos que possam executar tarefas que a visão humana consegue"(HUANG, 1996) (tradução nossa). Essa arquitetura fora desenvolvida a partir dos trabalhos de Kunihiko Fukushima, que propôs em 1980 um modelo de rede capaz de reconhecer padrões visuais sem ser afetado por mudanças de posição (FUKUSHIMA, 1980).

As CNN são compostas pelos elementos conhecidos de uma rede neural profunda (camada de entrada, camada escondida, função de ativação), mais um elemento novo: a camada de convolução (*convolutional layer*). Pensando cada pixel de uma imagem digital como um sinal de entrada do *input layer*, cada neurônio da camada de convolução estará ligada somente aos neurônios de entrada correspontes ao seu campo receptivo (GéRON, 2017), em vez de estar ligado a todos, como no perceptron tradicional. Uma segunda camada convolucional é ligada a um retângulo de neurônios (cr. Figura) da primeira camada.

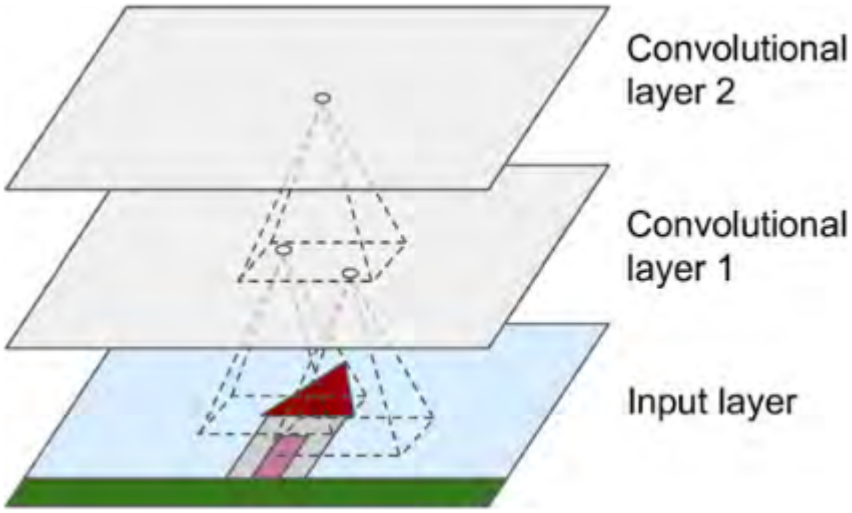


Figura 12 – Camadas da rede neural convolucional (GéRON, 2017)

A CNN também é composta por uma camada de junção (*pooling layer*) que, assim como a camada de convolução, é ligada a apenas alguns neurônios da camada anterior, agregando os sinais de entrada (com uma função de máximo ou de média, por exemplo), de modo a reduzir o tamanho da imagem de entrada, o que melhora o desempenho computacional do processamento da rede (GéRON, 2017).

Para fins do presente trabalho, essa arquitetura é bastante adequada, pois a rede convolucional é capaz de reconhecer padrões em imagens, como a presença de uma clareira de desmatamento em uma imagem do topo de uma floresta.

Parte II

Metodologia, Resultados e Discussões

4 Considerações Iniciais

Antes de abordarmos em detalhe a metodologia de trabalho adotada, é importante fazermos algumas considerações sobre as ferramentas que utilizaremos, bem como ao resultado que se espera chegar com esse trabalho.

4.1 Acesso às imagens SAR

Utilizaremos as imagens fornecidas pelo portal Copernicus da ESA, que fornece o imageamento produzido pelos satélites Sentinel, em particular os satélites Sentinel-1A e 1B, que fazem o sensoriamento em SAR. Para o pré-processamento dessas imagens, utilizaremos o programa SNAP, desenvolvido pela ESA e voltado para o trabalho com as imagens que a agência fornece.

4.2 Linguagem de programação utilizada

Para o desenvolvimento da rede neural, será utilizada a linguagem Python. Essa linguagem de programação tem-se destacado no uso para ciência de dados, com o desenvolvimento de ferramentas dedicadas para essa tareda, como as bibliotecas Keras, o Numpy e o Tensorflow. Ademais, por ser uma linguagem não proprietária (ao contrário, por exemplo, do Matlab), um algoritmo desenvolvido em Python tem um alcance maior para desenvolvedores que por ventura queiram consultar, utilizar ou mesmo aperfeiçoar esse trabalho, uma vez que este esteja publicado.

4.3 Ferramentas de desenvolvimento

O principal ambiente de desenvolvimentos dos algoritmos foi o Jupyter Notebook, ferramenta intuitiva de programação em Python, que utiliza a máquina local para o processamento dos *scripts*. Também utilizamos o Google Colab, ambiente muito semelhante ao Jupyter, porém utiliza computação em nuvem, o que demanda conexão com a Internet.

4.4 Máquina utilizada

O computador utilizado para o treinamento da rede será um ideapad 330S com processador AMD Ryzen 7, GPU Radeon 540 2GB, RAM 8GB. o computador apresentou desempenho razoável para as tarefas de preparação do banco de dados.

Porém, para o treinamento efetivo da rede neural, o poder computacional da máquina foi insuficiente, sendo necessário utilizar a ferramenta Google Colab, que dá acesso *online* a uma placa gráfica (GPU) de 12 GB, na versão livre, e 24 GB na versão profissional. Para o treinamento da rede em mais de 10 épocas (ver adiante) foi necessário utilizar a versão profissional.

4.5 Análise de desempenho e resultados esperados

Os modelos de classificação serão avaliados conforme a acurácia de sua predição, ou seja, a capacidade da rede de classificar corretamente uma imagem não contida no banco de dados que fez seu treinamento. Quanto menor for o valor da função de perda calculado, maior essa acurácia. O nível de acurácia que se espera alcançar é de um valor maior que 90%.

5 Região Alvo Escolhida

A região escolhida para ser alvo de sensoriamento foi a região do município de São Félix do Xingu, Pará. De acordo com levantamento do PRODES (PRODES, 2020), esse foi o município da Amazônia legal que mais desmatou no período até 2020, com um total de área desmatada de $19.886,2\ km^2$, o que representa 23.6% da área total de $84.253\ km^2$ (maior que o território da Áustria, cf. Figura 13) do município.

Figura 13 – Município de São Félix do Xingu, Pará (ABREU, 2021)

São Félix do Xingu é um município com população estimada de 135.732 habitantes, renda per capita de R$ 11.187,00, índice de desenvolvimento humano (IDH) de 0,594 (médio) e percentual da população com rendimento nominal mensal per capita de até 1/2 salário mínimo de 44,8% (IBGE, 2021b). A atividade agropecuária representa 36% da economia do município, principalmente pecuária bovina. A atividade industrial compõe 12,6% da economia local, principalmente extração mineral (serviços - 21,4% - e administração pública - 30% - compõem o restante da atividade econômica) (IBGE, 2018).

O município localiza-se no sudeste do estado do Pará, no encontro do Rio Fresco com o Rio Xingu, e faz parte da região conhecida como Arco do Desmatamento (cf. Figura 14). Essa região estende-se do leste do Pará, passa pelo norte do Mato Grosso, Rondônia e chega ao sul do Acre e é caracterizada por um intenso desmatamento do bioma amazônico, decorrente em grande parte da construção de rodovias (BR-230, BR-163, BR-364 ou, no caso de São Félix do Xingu, a PA-279) que possibilitaram o acesso a regiões da floresta amazônica antes inacessíveis (MESSIAS et al., 2021).

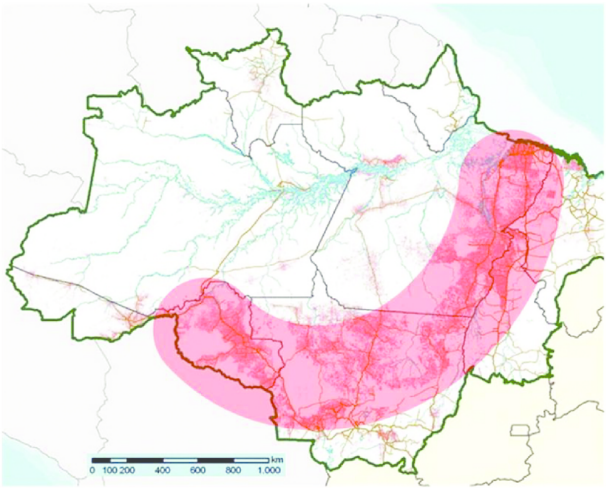


Figura 14 – Arco do Desmatamento (LUI; MOLINA, 2009)

Devido às características do município, São Félix do Xingu apresenta um perfil bastante rico para o imageamento aéreo voltado à identificação de desmatamento, com luagres totalmente desmatados, outros totalmente preservados e outros em condição parcial de desmatamento (cf. Figura 15). Esse contraste de imagens auxiliará a classificação feita pela rede neural.

Figura 15 – Imagem aérea da sede municipal de São Félix do Xingu (Bing Maps)

6 Formação do Banco de Imagens

6.1 Seleção de Imagens

A Figura 16 mostra a área compreendida pelo imageamento SAR que iremos utilizar no presente trabalho. As imagens foram feitas pelo satélite Sentinel-1A, em passagem descendente, no dia 05/07/2021 e coletadas na plataforma Copernicus da ESA. Essa área tem um total de $43,688\ km^2$ e as coordenadas geográficas de seus vértices encontram-se na Tabela 2.

Figura 16 – Imagem Selecionada na Plataforma Copernicus

	Latitude	Longitude
1	−6.301901332488	−50.84648361667641
2	−5.750021259252666	−53.07839660376309
3	−7.8202442866330735	−51.178185301798784
4	−7.264698302929794	−53.4181139411352

Tabela 2 – Coordenadas da área imageada

O modo de aquisição de dado feito por esse imageamento foi o de faixa ampla interferométrica (*Interferometric Wide Swath Mode*) ou IW, que se caracteriza pela divisão da faixa (*swath*) de sensoriamento em três subfaixas, pelas quais o sensor vai alternando ciclicamente, conforme a Figura 17.

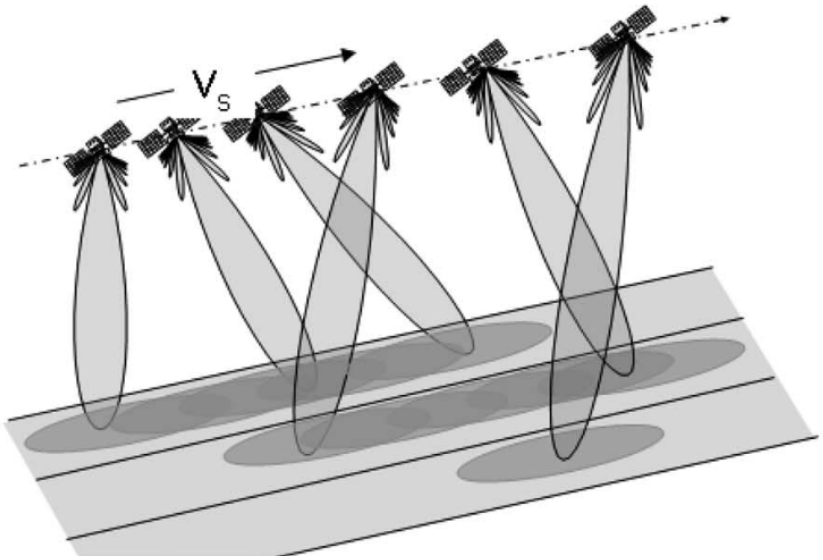


Figura 17 – Modo de Aquisição IW (BOURBIGOT, 2016)

Essas subfaixas são, por sua vez, subdivididas em nove passagens, ou *bursts* (cf. Figura 18), que são processadas como imagens separadas (BOURBIGOT, 2016), o que para fins de montagem do banco de dados para treinamento da rede neural, como veremos adiante, é bastante conveniente.

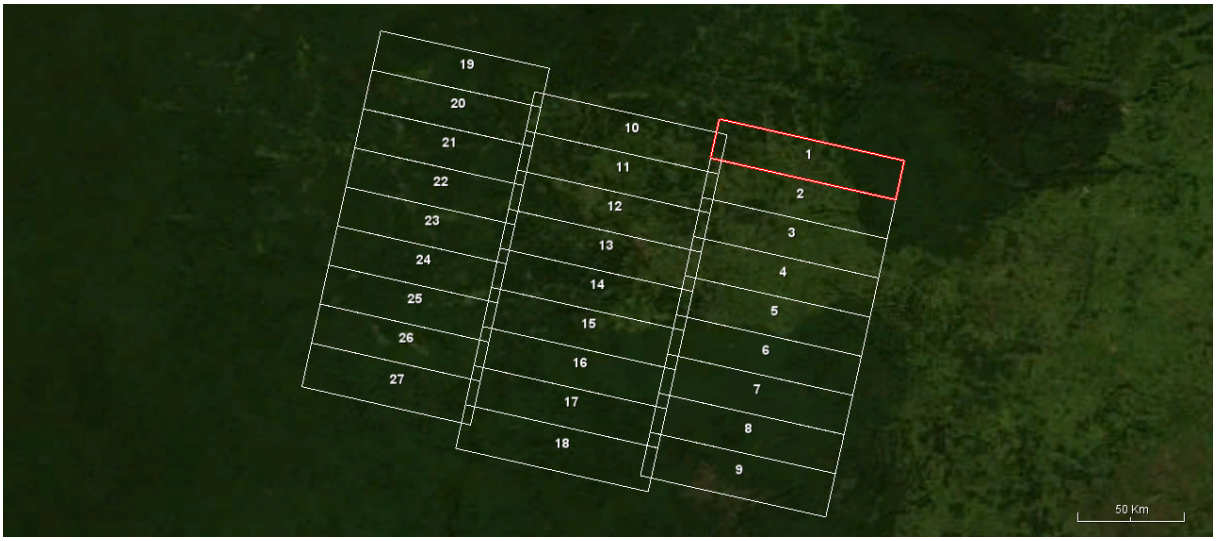


Figura 18 – Sub-*swaths* e seus respectivos *bursts*

As imagens SAR são notoriamente menos intuitivas para a visão humana do que as imagens óticas, sendo estas usadas como referência para o usuário tanto na plataforma Copernicus quanto no programa SNAP (cf. Figuras 16 e 18). A título de ilustração, a figura 19 mostra o imageamento contido no primeiro *burst* da sub-faixa IW 1 (destacado na Figura 18) da forma como ela é fornecida pelo Copernicus.

Figura 19 – Imagem SAR (*Swath* IW 1, *Burst* 1) sem pré-processamento

Para que as imagens SAR possam ser trabalhadas adequadamente, é necessário realizar seu pré-processamento, de modo que o resultado final fique conforme a Figura 20.

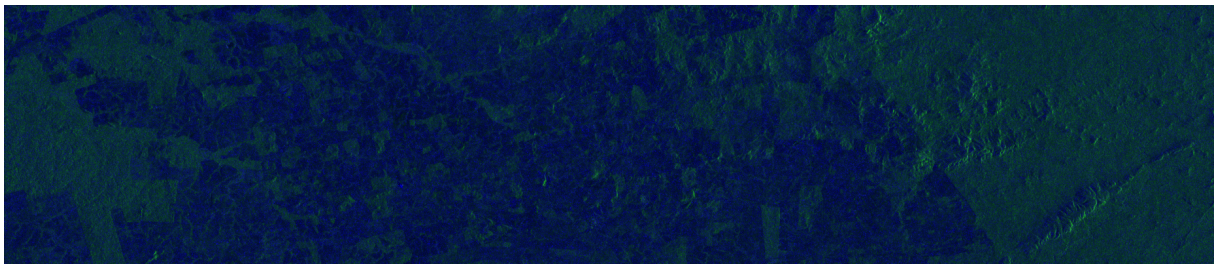


Figura 20 – Imagem SAR (*Swath* IW 1, *Burst* 1) após o pré-processamento

6.2 Pré-Processamento das imagens SAR no SNAP

O trabalho de pré-processamento neste trabalho foi feito no programa SNAP e teve como base a dissertação de mestrado *Avaliação do potencial dos dados polarimétricos Sentinel-1A para mapeamento do uso e cobertura da terra na região de Ariquemes - RO*, de Juliana Maria Ferreira de Souza Diniz (DINIZ, 2019), e as etapas seguem conforme a Figura 21, item a), com adaptações.

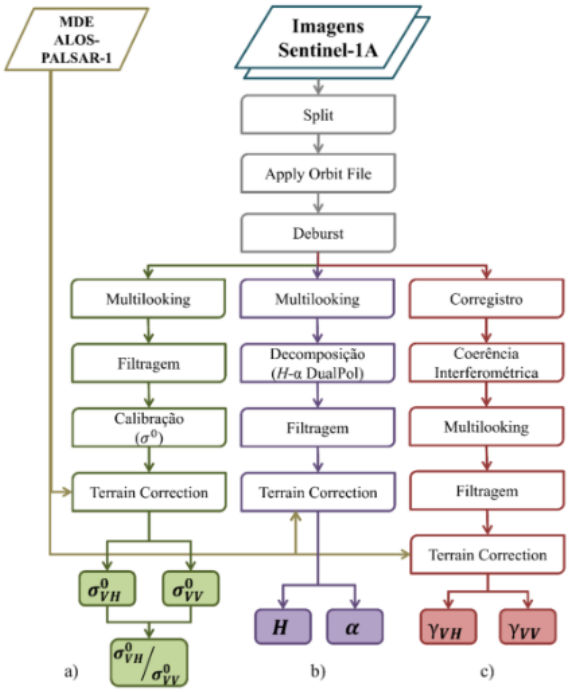


Figura 21 – Etapas de pré-processamento de imagens SAR no SNAP (DINIZ, 2019)

Split: os arquivos SAR fornecidos pela plataforma Sentinel são via de regra de tamanho considerável (4,5 GB o arquivo utilizado neste trabalho). Dependendo do computador que se está utilizando, o processamento do arquivo inteiro é muito demorado ou simplesmente não há poder computacional para isso. O melhor é subdividir a imagem nas subfaixas e *bursts* fornecidas pelo modo IW de imageamento. Como o propósito deste trabalho é formar um banco de dados com um bom número de amostras para a rede neural, optamos por dividir a imagem nos 27 *bursts* disponíveis, replicando as etapas seguintes de processamento nos 27 novos arquivos gerados.

Apply Orbit File: o arquivo fornecido pelo Copernicus fornece uma série de metadados referentes ao imageamento do alvo, incluindo informações dos vetores de estado de órbita do satélite. Essas informações não estão atualizadas no arquivo e, caso se deseje trabalhar com dados de georreferenciamento, é melhor atualizar esses dados usando esse comando. Para o nosso trabalho essa informação não era relevante, mas para fins de boas práticas fizemos a atualização mesmo assim.

Remove Thermal Noise: imagens SAR tendem a ter um ruído térmico aditivo, principalmente entre as subfaixas. É interessante filtrar esse ruído com esse comando, porém é importante ressaltar que há perda de informação de atributos polarimétricos (fase e quadratura). Se estivéssemos fazendo uma decomposição polarimétrica, essa filtragem não poderia ser feita. Porém como estamos trabalhando somente com coeficiente de retroespalhamento, optamos por aplicar o filtro.

Calibration: essa é a operação que fornece a informação que precisamos, o coeficiente de retroespalhamento, por meio de uma calibração radiométrica que, conforme documentação do próprio SNAP, é calculado a partir da equação:

$$\sigma_i^0 = \frac{DN_i^2}{A_i^2} \tag{6.1}$$

em que DN_i é o número digital do i-ésimo pixel e A_i é um coeficiente de calibração obtida por tabela de pesquisa de calibração (DINIZ, 2019). Essa operação realiza a calibração para os dois modos de polarização (VH e VV) que o Sentinel-1 fornece.

Deburst: existe uma descontinuidade entre os *bursts* de uma imagem IW, que fica bem evidente quando a imagem formada pela operação *Split* é formada por mais de um *burst*. Por mais que não seja o caso do presente trabalho, optamos por realizar essa etapa, para eliminar possíveis distorções nas bordas das imagens.

Multilooking: como mencionado anteriormente, imagens SAR são passíveis de ruído *speckle*, e o filtro *multilooking* minimiza os efeitos desse ruído, bem como faz uma correção de píxel. Optou-se por usar a opção padrão do SNAP de 4 *range lookings* e 1 *azimuth looking*.

Speckle Filter: Além do *multilooking*, é interessante passar um filtro especificamente contra os efeitos do *speckle*, para diminuir a aparência granulada da imagem. Vale ressaltar que uma aplicação muito intensa desse filtro acaba degradando informações da imagem, reduzindo sua resolução.

Terrain Correction: o imageamento SAR é passível de distorções provocadas pelo formato do terreno alvo, como formação de sombras em morros e encostas. Essa operação visa reduzir essas distorções, além de orientar a imagem com base nas informações de georreferenciamento do alvo, o que faz a imagem inclinar na mesma proporção da inclinação da órbita do Sentinel-1 (98.18º). Essa orientação é bastante inconveniente para o nosso trabalho, pois dificulta a divisão de cada um dos *bursts* em um número maior de amostras menores. O ganho dessa correção não é relevante para o tipo de detecção que estamos realizando, então optamos por não realizar essa operação.

Terminadas essas etapas de pré-processamento, salvamos o resultado final em um arquivo raster (extensão GEOTiff) com dois canais, cada um armazenando os coeficientes de retroespalhamento para cada uma das polarizações ($\sigma^0 VH$ e $\sigma^0 VV$).

6.3 Processamento das Imagens em Python

6.3.1 Converter arquivos raster em tensores

Encerrada a fase de trabalho com o SNAP, passou-se para o trabalho com Python, com o objetivo de formar o banco de dados para a rede neural. Para isso, era necessário converter as informações contidas nos arquivos rasteres de imagens em vetores (*arrays*) de dados, ou matrizes de dados contendo os coeficientes de retroespalhamento. Esses vetores serão as informações de entrada (*inputs*) da rede neural convolucional. Esse trabalho de conversão foi feito com o *script* desenvolvido no Apêndice A.1 (*raster_to_array.ipynb*).

Para cada arquivo raster, foi criado um vetor tridimensional (tensor) com dois canais, cada um contendo os coeficientes de cada polarização (VH e VV). Além disso, fora calculada a razão cross-polarizada entre os dois coeficientes, e o resultado fora adicionado como um terceiro canal nos arquivos correspondentes.

Na sequência, houve a remoção das bordas pretas que o SNAP gera quando são salvos os arquivos rasteres e não acrescentam nenhuma informação relevante, e a inversão horizontal da ordem dos vetores, uma vez que as imagens SAR do Sentinel-1 são registradas de maneira espelhada quando o satélite está em órbita descendente (o que é corrigido quando se faz a correção de terreno no SNAP, etapa que pulamos no pré-processamento).

O resultado desse processamento foi a criação de: 9 tensores de dimensão 1450 x 6800 x 3 (subfaixa 1), 9 tensores de dimensão 1450 x 6100 x 3 (subfaixa 2) e 9 tensores de dimensão 1450 x 5900 x 3 (subfaixa 3). Esses tensores foram subdivididos e tensores

menores, de dimensão 363 x 358 x 3, num padrão de 4 linhas e 19, 17 6 16 colunas, a depender da subfaixa (IW1, IW2 e IW3, respectivamente), resultando em um total de 1872 novos tensores, que foram salvos em arquivos de texto (formato .txt). Cada nova amostra foi numerada de acordo com a subfaixa (de 1 a 3), o *burst* (de 1 a 9), a linha (de 1 a 4) e a coluna (de 1 a 19, 17 ou 16) a que pertencia.

A dimensão para os novos tensores foi escolhida de modo a gerar um número considerável de amostras para o banco de dados da rede neural, mas que tivessem tamanho suficiente para possibilitar a interpretação visual da cena que a imagem formada pelos tensores (ver adiante). O formato também foi escolhido para aproveitar o maior número de píxeis possíveis da imagem original.

6.3.2 Converter Tensores em arquivos de imagem

Feito o cálculo da razão cross-polarizada, podemos usar as informações contidas nos três canais de cada tensor e formar uma imagem colorida RGB ($R = \sigma^0 VH$, $G = \sigma^0 VV$, $B = CR_2$). Para fins de *input* da rede neural, essa plotagem das imagens não é necessária; porém essas imagens são importantes para que se tenha uma interpretação visual que permita a criação de rótulos correspondentes a essas imagens (ver adiante). As imagens foram criadas pelo *script* do Apêndice A.2 (*array_to_image.ipynb*).

Foram gerados arquivos de imagem (formato .png) para cada um dos 1872 tensores gerados pelo *script* anterior. Para fins de documentação, optamos por colocar no Apêndice 2 as imagens geradas a partir dos *bursts* inteiros, de modo a termos uma noção global de como as imagens SAR ficaram na totalidade. Que fique claro, no entanto, que cada um dos *bursts* contidos no apêndice foi subdividido em uma média de 70 amostras cada.

6.3.3 Classificação manual e organização do banco de dados

Como mencionado anteriormente, o tipo de aprendizagem de máquina que pretendemos desenvolver é do tipo supervisionada. Em outras palavras, um conjunto de imagens contendo as classificações (rótulos) desejadas será fornecido à rede neural, de modo a ela aprender os padrões que definem a classificação, para que posteriormente possa classificar autonomamente informações novas, não contidas no conjunto de treinamento.

É comum no desenvolvimento de algoritmos de aprendizagem de máquina utilizar banco de dados de acesso livre, em que as amostras já estão devidamente classificadas e prontas para o uso. Porém, no presente trabalho estamos lidando com um conjunto de imagens de produção própria, o que demandou fazermos o trabalho de classificação manualmente.

Decidimos classificar as imagens em 4 categorias: totalmente desmatado, parcialmente desmatado, parcialmente preservado,totalmente preservado (com representação

numérica de 1 a 4, respectivamente). Se a cena mostrada na imagem não contivesse qualquer traço de desmatamento (ou de floresta em pé), era classificada como 'totalmente preservada' (ou 'totalmente desmatada'). As outras duas categorias eram aplicadas a imagens em que somente parte da cena apresentava marca de desmatamento, mas contendo parcelas de floresta preservada; se a parcela preservada era maior do que a desmatada, era classificada como 'parcialmente preservada', e vice-versa (cf. Figura 22).

Boa parte das amostras tinham nitidez suficiente para fazer uma interpretação adequada da cena. Como vimos, a razão cross-polarizada consegue contrastar o retroespalhamento volumétrico (presente na copa das árvores) e o superficial (presente no chão descoberto). De todo modo, foram utilizadas imagens óticas de alta resolução (Google Maps, Bing Maps e SNAP) como referência, em particular para as amostras cuja interpretação fosse menos clara.

Feita a classificação manual de todas as amostras, suas respectivas representações numéricas foram salvas em um vetor, de modo a estarem correlacionadas às suas respectivas amostras, contidas em um tensor de dimensões 1872x363x358x3, conforme *script* contido no Apêndice A.3 (*split_train_test_dataset.ipynb*). As amostras foram randomizadas (junto com os respectivos rótulos) e divididas entre imagens de treinamento, imagens de validação e imagens de teste, numa proporção de 70%, 15% e 15%, respectivamente.

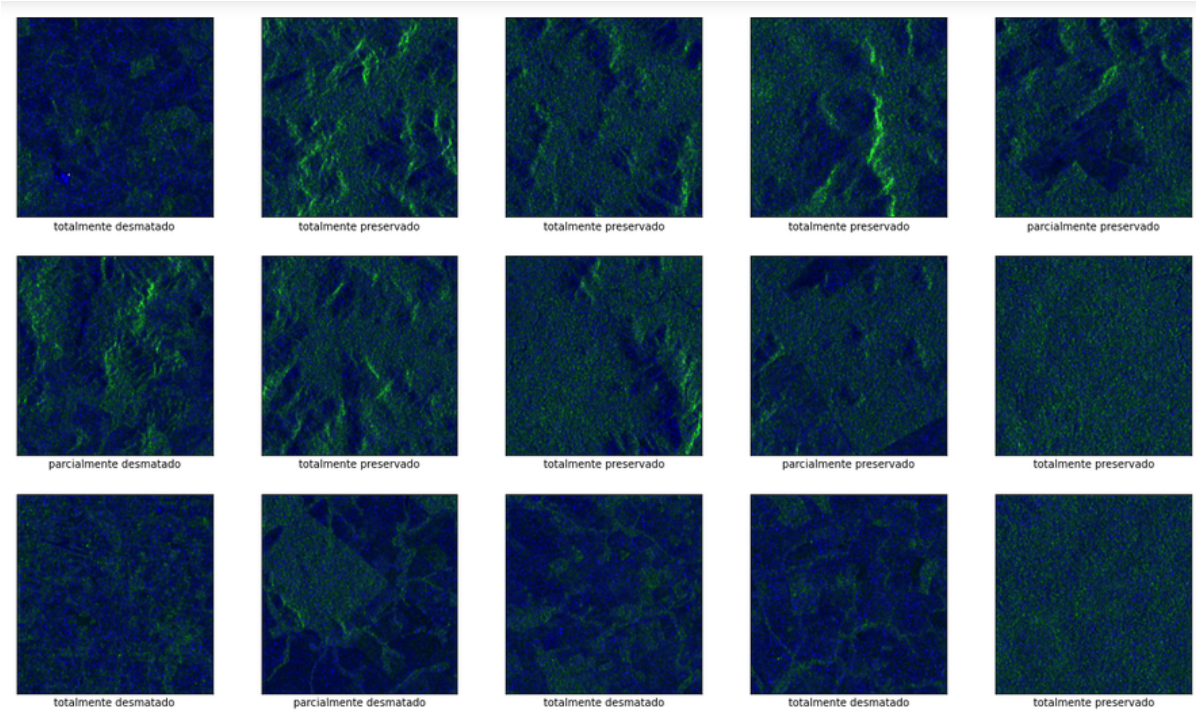


Figura 22 – Exemplos da classificação manual das amostras

7 Desenvolvimento da Rede Neural

7.1 Arquitetura da Rede Neural

Após algumas iterações de teste, definimos a seguinte arquitetura da rede neural (cf. Figura 23 e Apêndice C):

- camada de entrada com o formato das imagens (363x358x3);
- 1ª. camada de convolução (64 filtros, *kernel* 2x2);
- 1ª. camada de *pooling* (formato 2x2);
- 2ª. camada de convolução (128 filtros, *kernel* 2x2);
- 2ª. camada de *pooling* (formato 2x2);
- 3ª. camada de convolução (256 filtros, *kernel* 2x2);
- 3ª. camada de *pooling* máximo (formato 2x2);
- 4ª. camada de convolução (512 filtros, *kernel* 2x2);
- 4ª. camada de *pooling* máximo (formato 2x2);
- 5ª. camada de convolução (1024 filtros, *kernel* 2x2);
- 5ª. camada de *pooling* máximo (formato 2x2);
- camada de achatamento (*flatten*);
- 1ª. camada densa (1024 neurônios);
- 1ª. camada de descarte (*dropout*) de razão $0,5$;
- 2ª. camada densa (1024 neurônios);
- 2ª. camada de descarte (*dropout*) de razão $0,5$;
- camada densa de saída (2, 3 ou 4 saídas).

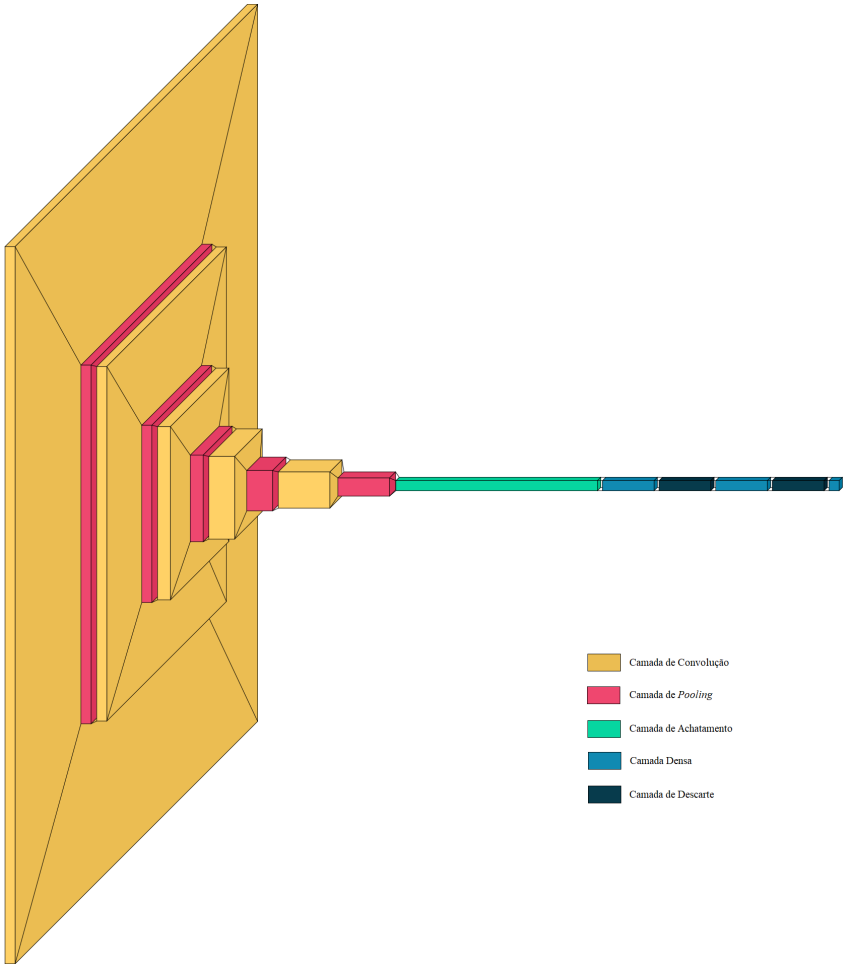


Figura 23 – Formato da rede neural convolucional

Essa arquitetura foi desenvolvida com base em um conhecido modelo de rede de classificação de imagens (VGG16), desenvolvido por (SIMONYAN; ZISSERMAN, 2015) e vencedor do *ImageNet Large Scale Visual Recognition Challenge 2014* (ILSVRC2014). Algumas considerações sobre as camadas utilizadas:

Camada de convolução: camada que executa a principal operação da rede neural convolutional. Um filtro reconhecedor de padrões passa uma janela (*kernel*) por toda a imagem, executando uma convolução entre as duas matrizes e armazenando o valor calculado em uma matriz resultante. O número de filtros define a profundidade (número de canais) dessa matriz.

Camada de *pooling* máximo: camada que opera a redução das dimensões dos valores de entrada, a depender do tamanho da janela do filtro (2x2 reduz as dimensões à metade). O filtro pode selecionar o valor médio dentro da janela ou, no caso escolhido, o valor máximo.

Camada de achatamento: apenas altera o formato dos valores de entrada, colocando-os em um vetor de uma dimensão.

Camada Densa: uma camada comum em que todos os neurônios estão conectados aos sinais de entrada.

Camada de Descarte: camada que descarta (reduz a zero) os valores de entrada numa frequência determinada (0, 5 no caso), o que permite reduzir o *overfitting* do modelo.

Camada de Saída: camada densa com número de saídas correspondentes ao número de rótulos que classificam as amostras.

O *script* que desenvolveu esse modelo de rede neural e realizou seu subsequente treinamento encontra-se no Apêndice A.4 (*cnn_training.ipynb*).

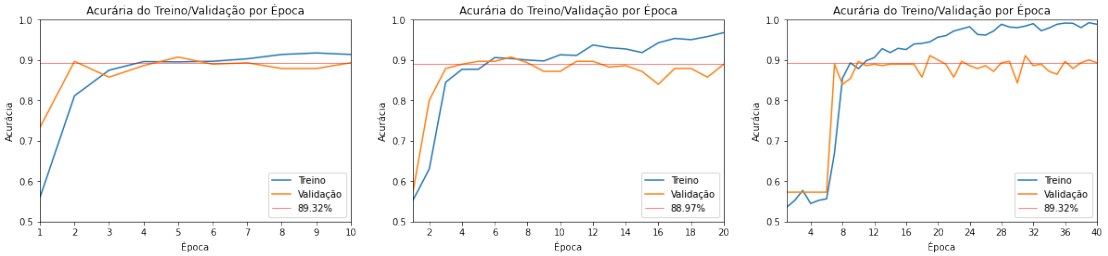
7.2 Treinamento da Rede Neural

O treinamento do modelo com base em 70% das amostras e seus respectivos rótulos ocorreu em três cenários: 4 classificações distintas ('totalmente desmatado', 'parcialmente desmatado', 'parcialmente preservado' e 'totalmente preservado'); 3 classificações distintas ('totalmente desmatado', 'parcialmente desmatado' e totalmente preservado'); e 2 classificações distintas ('desmatado' e 'preservado'). Esses cenários foram escolhidos quando se percebeu que, com os quatro rótulos originais da classificação manual, não se estava conseguindo aumentar a acurácia da validação (conferência do modelo com base em 15% das amostras que não fizeram parte do treinamento), e o modelo tendia ao *overfitting*.

A hipótese levantada foi que era difícil para o modelo fazer a distinção entre o que era parcialmente preservado e parcialmente desmatado. Assim, foi feita a troca dos rótulos 'parcialmente preservado' para 'parcialmente desmatado', em um cenário, e a troca de ambos para 'totalmente desmatado', que passou a ser considerado apenas 'desmatado', frente ao outro conjunto que passou também a ser apenas considerado 'preservado'.

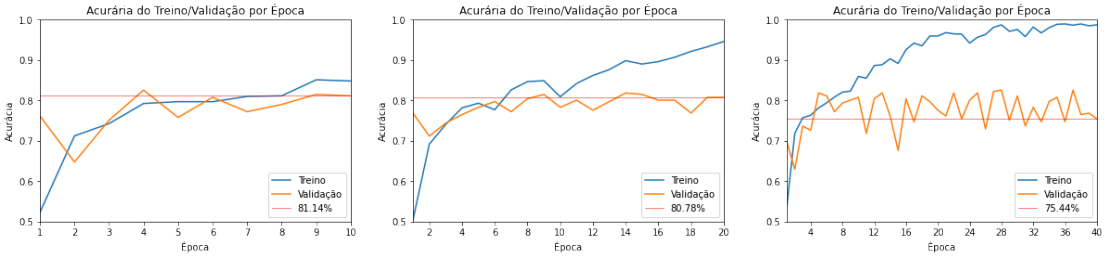
Encerrada a fase de validação, foram feitos os testes do modelo utilizando-se os 15% restantes das amostras. Os resultados dos testes podem ser conferidos a seguir.

8 Resultados



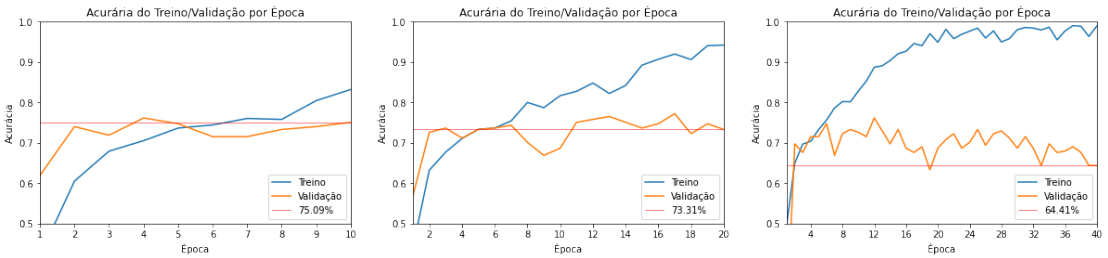
(a) (b) (c)

Figura 24 – Acurácia do modelo com 2 rótulos e (a) 10 épocas (b) 20 épocas e (c) 40 épocas



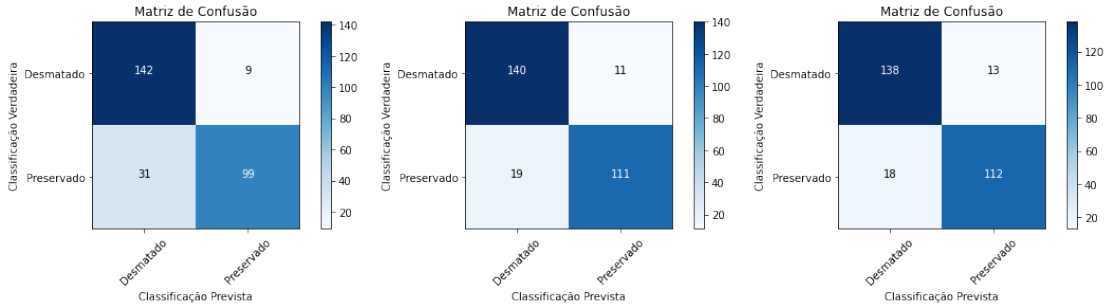
(a) (b) (c)

Figura 25 – Acurácia do modelo com 3 rótulos e (a) 10 épocas (b) 20 épocas e (c) 40 épocas



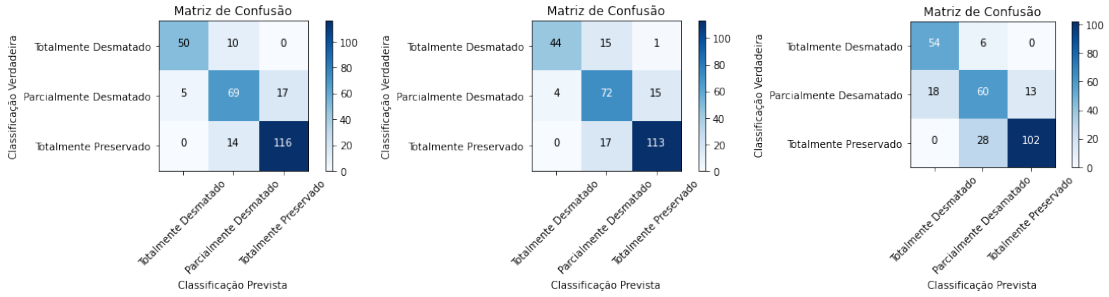
(a) (b) (c)

Figura 26 – Acurácia do modelo com 4 rótulos e (a) 10 épocas (b) 20 épocas e (c) 40 épocas



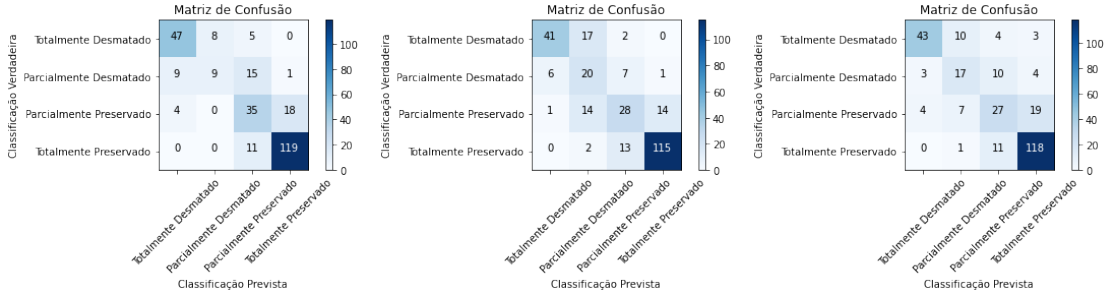
(a) (b) (c)

Figura 27 – Matriz de confusão do modelo com 2 rótulos e (a) 10 épocas (b) 20 épocas e (c) 40 épocas



(a) (b) (c)

Figura 28 – Matriz de confusão do modelo com 3 rótulos e (a) 10 épocas (b) 20 épocas e (c) 40 épocas



(a) (b) (c)

Figura 29 – Matriz de confusão do modelo com 4 rótulos e (a) 10 épocas (b) 20 épocas e (c) 40 épocas

9 Discussão

Uma primeira constatação que os dados permitem é a confirmação da hipótese de que menos rótulos levou a uma maior acurácia de validação, como pode ser visto nos gráficos da Figura 24, bem como a uma maior acurácia de teste, como pode ser visto nas matrizes de confusão na Figura 27. Além disso, constatou-se que o modelo alcança um nível de acurária bem cedo, com 10 épocas ou menos (época é o número de vezes que o modelo faz o treinamento por todo o *dataset*); continuar o treinamento por mais épocas, por sua vez, leva ao *overfitting* (a acurácia de treinamento muito alta e muito descolada da acurácia de validação), tendência que se intensifica com um número de rótulos.

Um ponto importante a ser levado em consideração é que existe uma assimetria de avaliação dos erros de classificação ocorridos em todos os cenários e mostrados pelas matrizes de confusão. Para fins de detecção de desmatamento na Amazônia, é melhor ocorrer um falso positivo (acusar a presença de desmatamento quando na verdade não há) do que um falso negativo (a presença real de uma atividade extrativista sem que tenha sido devidamente detectada). Nesse sentido, mais uma vez os resultados apontam que trabalhar com dois rótulos é o melhor caminho, pois tanto no cenário de 10 épocas (31 contra 9), quanto no de 20 épocas (19 contra 11) quanto no de 40 épocas (18 contra 13) o falso positivo predominou. Em outros cenários, à excessão do cenário de 3 rótulos e 40 épocas (46 contra 19) todos os outros apresentaram uma predominância de falsos negativos.

Nenhum cenário conseguiu chegar à acurácia desejada de mais de 90%; porém os melhores cenários chegaram a um valor muito próximo (89,32%). Consideramos o resultado satisfatório, tendo em vista o tamanho reduzido do banco de dados (menos de 2 mil amostras), se comparado aos *datasets* disponíveis na Internet, com 50 ou 60 mil amostras, bem como ao poder computacional disponível para o trabalho. Deixamos como indicação para um futuro trabalho, o desenvolvimento de um outro algoritmo, integrado com a ferramenta SNAP, que automatize grande parte das tarefas de pré-processamento, de modo a produzir mais amostras em menos tempo e otimizar a parte de classificação manual e, por fim, o treinamento da rede neural, testando em todo o ciclo a alteração de outros parâmetros, como diferentes filtros *speckle* ou coeficientes polarimétricos além da razão cross-polarizada, o que mais otimizará a acurácia da classificação automatizada.

10 Conclusão

A Amazônia é um dos mais importantes ativos territoriais que o Brasil possui. É a maior reserva de recursos genéticos e de água doce superficial que conhecemos no mundo. Ao mesmo tempo, é o lar de mais de 20 milhões de pessoas somente na Amazônia Legal brasileira. É um espaço dinâmico, que envolve diversas interações sociais, econômicas e culturais, por vezes conflitantes. A gestão territorial desse imenso bioma é um desafio geopolítico, ambiental e humanitário.

Na medida em que a sociedade global vem percebendo a urgência da adoção de um modo de produção socialmente justo, economicamente próspero e ambientalmente sustentável, de modo a mitigar os piores efeitos da crise climática, o Brasil precisa corresponder adequadamente a esse anseio, reforçando seus esforços para preservação de seu principal patrimônio ecológico.

Nessa tarefa, o sensoriamento remoto da região amazônica é fundamental, e respostas efetivas com base nele para combater a degradação do bioma são cada vez mais urgentes. Agentes como o INPE, o IBAMA, a Polícia Fedral e outros órgãos de controle e monitoramento são essenciais para coibir práticas ilegais de desmatamento. Infelizmente, devido às características específicas da região, como grande presença de nuvens, dificultam essa tarefa de monitoramento.

O imageamento por radar, por meio das plataformas SAR, são um meio de contornar esses obstáculos. Trabalhando na faixa espectral do micro-ondas, a radiação emitida por esses imageadores sofrem uma interferência menor dos efeitos atmosféricos. Ademais, sua capacidade de penetrar no material dos alvos, a depender da banda com que se esteja trabalhando, permite coletar informações inacessíveis caso se trabalhe com o imageamento ótico. A desvantagem mais evidente desse tipo de sensoriamento é a dificuldade de interpretação pelo olho humano de imagens formadas fora do espectro visível.

Para superar essa dificuldade, é interessante o uso de ferramentas computacionais que auxiliem - ou mesmo executem efetivamente - a tarefa de detecção dos alvos de interesse. A visão computacional é um campo da tecnologia que busca desenvolver ferramentas que emulem, e de certo modo superem, a capacidade do olho humano. Entre essas ferramentas está as redes neurais artificais de classificação de imagens, em particular a rede neural convolucional, uma arquitetura muito eficiente para encontrar padrões comuns em uma mesma categoria de imagens.

Neste trabalho, procuramos desenvolver um modelo de rede neural convolucional que fosse capaz de identificar imagens SAR contendo focos de desmatamento na Floresta Amazônica com um alto grau de acurácia (mais de 90%). Para tanto, fora montado um

banco de dados de imageamento da região de São Félix do Xingu, Pará, município localizado no Arco do Desmatamento e, até o ano de 2020, o lugar com a maior área acumulada de desmatamento. A dinâmica territorial dessa localidade gerou uma grande variedade de cenas, de imagens de florestas totalmente preservadas, muitas delas localizadas em reservas indígenas, a regiões totalmente desmatadas. Esse contraste de imagens permitiu a criação de um banco de dados que servisse de subsídio para o treinamento da rede neural.

O resultado final chegou bem próximo da meta pretendida (89,32%). Mais do que isso, o presente trabalho abordou o ciclo completo do sensoriamento, desde a escolha motivada do alvo, a coleta do material bruto na plataforma imageadora, o pré-processamento das imagens, de modo a torná-las legíveis, a formação do banco de dados, o desenvolvimento do algoritmo de classificação e, por fim, o resultado e a interpretação dos dados fornecidos pela rede neural. Futuramente, o trabalho pode continuar a ser desenvolvido para englobar outros aspectos desse campo do sensoriamento por radar, como uso de outros filtros de pré-processamento, cálculo de outros coeficientes polarimétricos ou mesmo o desenvolvimento de outras arquiteturas que otimizem o trabalho de detecção de atividades extrativistas no bioma amazônico. A adoção de recursos tecnológicos que possibilitem a preservação e o desenvolvimento sustentável da região é o âmago da economia verde que o Brasil - e o mundo - precisa adotar.

Referências

ABREU, R. L. de. *Mapa de São Félix do Xingu, Pará*. 2021. Disponível em: <https://bit.ly/3aW4GBY>. Citado 2 vezes nas páginas 15 e 59.

AGGARWAL, C. C. *Neural Networks and Deep Learning: A Textbook*. [S.l.]: Springer, 2018. Citado 2 vezes nas páginas 52 e 53.

ANDERSON, J. D. *Introduction to Flight*. third. [S.l.]: McGraw-Hill, 1989. Citado na página 49.

ASSIS, L. F. F. G. et al. Terrabrasilis: A spatial data analytics infrastructurefor large-scale thematic mapping. *International Journal of Geo-Information*, v. 8, n. 513, 2019. Citado na página 46.

BECKERT, T.; FALCKE, H. Circular polarization of radio emission from relativistic jets. *Astronomy & Astrophysics*, 2002. Citado 2 vezes nas páginas 15 e 36.

BOURBIGOT, M. *Sentinel-1 Product Definition*. [S.l.], 2016. Citado 2 vezes nas páginas 15 e 62.

DABBOOR, M.; BRISCO, B. Wetlands management - assessing risk and sustainable solutions. In: _____. [S.l.]: IntechOpen, 2018. cap. Wetland Monitoring and Mapping Using Synthetic Aperture Radar. Citado 2 vezes nas páginas 15 e 37.

DEFESA, M. da. *Censipam Projeto Amazônia*. 2020. Disponível em: <https://bit.ly/30mnfdu>. Citado na página 47.

DINIZ, J. M. F. de S. *Avaliação do potencial dos dados polarimétricos Sentinel-1A para mapeamento do uso e cobertura da terra na região de Ariquemes - RO*. Dissertação (Mestrado) — INPE, feb 2019. Citado 7 vezes nas páginas 15, 36, 37, 38, 39, 63 e 64.

ESA. *Earth Observation Missions Database*. 2021. Disponível em: <https://bit.ly/3v9a90d>. Citado 2 vezes nas páginas 17 e 40.

FUKUSHIMA, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, n. 36, p. 193 202, 1980. Citado na página 53.

GéRON, A. *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. first. [S.l.]: O'Reilly, 2017. Citado 7 vezes nas páginas 15, 49, 50, 51, 52, 53 e 54.

HSU, H. S. U. *Atmospheric Absorption & Transmission*. 2019. Disponível em: <https://bit.ly/3nauak9>. Citado 3 vezes nas páginas 15, 33 e 34.

HUANG, T. S. Computer vision: Evolution and promise. *19th CERN School of Computing*, 1996. Citado na página 53.

IBGE. *Amazônia Legal*. 2014. Disponível em: <https://bit.ly/3555wJr>. Citado 2 vezes nas páginas 15 e 43.

IBGE. *Indicadores de desenvolvimento sustentável*. IBGE, 2015. Disponível em: <https://bit.ly/3dEUDDn>. Citado na página 44.

IBGE. *Produto Interno Bruto dos Municípios*. 2018. Disponível em: <https://bit.ly/3aS2v2m>. Citado na página 59.

IBGE. *Biomas e sistema costeiro-marinho do Brasil: compatível com a escala 1:250.000*. Rio de Janeiro, 2019. Disponível em: <https://bit.ly/3lQxTlb>. Citado na página 43.

IBGE. *Fronteira Agrícola – Amazônia Legal*. 2021. Disponível em: <https://bit.ly/3lKbF48>. Citado 2 vezes nas páginas 43 e 44.

IBGE. *São Félix do Xingu*. 2021. Disponível em: <https://bit.ly/3n9GpOx>. Citado na página 59.

INPE. *Metodologia Utilizada nos Projetos PRODES e DETER*. 2019. Disponível em: <https://bit.ly/2G84G6h>. Citado 2 vezes nas páginas 44 e 45.

INPE. *PRODES - Amazônia*. 2020. Disponível em: <https://bit.ly/2Smzp1z>. Citado na página 45.

INPE. *DETER*. 2021. Disponível em: <https://bit.ly/2RKDfog>. Citado na página 46.

INPE. *DETER INTENSO*. 2021. Disponível em: <https://bit.ly/2GurzAE>. Citado na página 46.

INPE. *Portal TerraAmazon*. 2021. Acessado em 06/05/2021. Disponível em: <https://bit.ly/2SgbLUv>. Citado na página 46.

LILLESAND, T. M.; KIEFER, R. W.; CHIPMAN, J. W. *Remote Sensing and Image Interpretation*. fifth. [S.l.]: Wiley, 2004. Citado na página 33.

LUI, G. H.; MOLINA, S. M. G. Ocupação humana e transformação das paisagens na amazônia brasileira. *Amazônica - Revista de Antropologia*, v. 1, n. 1, p. 200–228, apr 2009. Citado 2 vezes nas páginas 15 e 60.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, v. 5, p. 115–133, 1943. Citado na página 49.

MENESES, P. R.; ALMEIDA, T. de. *Introdução ao Processamento de Imagens de Sensoriamento Remoto*. [S.l.]: Editora UnB, 2012. Citado 2 vezes nas páginas 33 e 34.

MESSIAS, C. G. et al. Análise das taxas de desmatamento e seus fatores associados na amazônia legal brasileira nas últimas três décadas. *RA'EGA*, v. 52, p. 18 –41, nov 2021. Citado na página 59.

MITCHELL, T. M. *Machine Learning*. [S.l.]: McGraw-Hill, 1997. Citado na página 52.

NASA. *View of the Earth as seen by the Apollo 17 crew traveling toward the moon*. 1972. Disponível em: <https://go.nasa.gov/31jwefH>. Citado 2 vezes nas páginas 15 e 34.

NASA. *What is Synthetic Aperture Radar?* 2020. Disponível em: <https://go.nasa.gov/2EQLbOT>. Citado 2 vezes nas páginas 15 e 35.

PRODES. *Desmatamento nos Municípios da Amazônia Legal para o ano de 2020*. 2020. Disponível em: <https://bit.ly/3vI7ix3>. Citado na página 59.

ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Cornell Aeronautical Laboratory*, v. 65, n. 6, 1958. Citado na página 50.

SAMUEL, A. L. Some studies in machine learning using the game of checkers. *IBM Journal*, v. 3, n. 3, jul 1959. Citado na página 52.

SILVA, C. A. et al. Análise qualitativa do desmatamento na floresta amazônica a partir de sensores sar, Óptico e termal. *Anuário do Instituto de Geociências - UFRJ*, v. 42, n. 4, p. 18–29, dec 2019. Citado 4 vezes nas páginas 15, 41, 44 e 45.

SIMONYAN, K.; ZISSERMAN, A. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. Disponível em: <https://arxiv.org/pdf/1409.1556.pdf>. Citado na página 70.

Apêndices

APÊNDICE A – Códigos Python

A.1 *raster_to_array.ipynb*

```
#libraries
from osgeo import gdal
import numpy as np

# opening the list of swath array shapes
swath_array_shapes = []

#looping through the swaths
for i in range(3):
    for j in range(9):

        # reading the i_j swath
        dataset = (gdal.Open(
            r'tif_files/S1A_IW_SLC__1SDV_20210705T090737_'
            + '20210705T090804_038640_048F41_253E_split_'
            + str(i + 1) + '_' + str(j + 1)
            + '_Orb_TNR_Cal_deb_ML_Lee3.tif'))

        #getting the raster bands
        band1 = dataset.GetRasterBand(1) # Red channel
        band2 = dataset.GetRasterBand(2) # Green channel
        b1 = band1.ReadAsArray()
        b2 = band2.ReadAsArray()
        b3 = np.divide(b1, b2) # Blue channel
        b3 = np.nan_to_num(b3)
        swath_array = np.dstack((b1, b2, b3))

        #remove black boards
        while np.prod(sum(swath_array)) == 0:
            if sum(sum(swath_array[:,0,:])) == 0:
                swath_array = np.delete(swath_array, 0, 1)
            if sum(sum(swath_array[:,-1,:])) == 0:
                swath_array = np.delete(swath_array, -1, 1)
```

```
    if sum(sum(swath_array[0,:,:])) == 0:
        swath_array = np.delete(swath_array, 0, 0)
    if sum(sum(swath_array[-1,:,:])) == 0:
        swath_array = np.delete(swath_array, -1, 0)

#flip array horizontally
swath_array = np.flip(swath_array, 1)

#filling the swath array shapes list
swath_array_shapes.append(swath_array.shape)

#saving 3D-arrays as 2D-txt
reshaped_array = swath_array.reshape(swath_array.shape[0], -1)
np.savetxt("txt_files/swath_" + str(i + 1) + "_" + str(j + 1)
           + ".txt", reshaped_array)

#dimensions of cropped samples in each swath
x_len = 358
y_len = 363

#number of crooped samples for each row in each swath
if i == 0:
    samples = 19
elif i == 1:
    samples = 17
else:
    samples = 16

#looping through samples in a given swath
for k in range(4):
    for l in range(samples):
        cropped_array = swath_array[
            0 + y_len * k : y_len * (1 + k),
            0 + x_len * l : x_len * (1 + l), :]
        reshaped_array = cropped_array.reshape(
            cropped_array.shape[0], -1)
        (np.savetxt("txt_files/cropped_" + str(i + 1) + "_"
                    + str(j + 1) + "_" + str(k + 1) + "_" +
                    str(l + 1) + ".txt", reshaped_array))
```

```
#saving the swath array shapes list as txt
np.savetxt("txt_files/swath_array_shapes.txt", swath_array_shapes)
```

A.2 array_to_image.ipynb

```
#libraries
import numpy as np
import matplotlib.pyplot as plt

#loading swath array shapes list
swath_array_shapes = np.loadtxt("txt_files/swath_array_shapes.txt")
    .astype('int')

#looping through swaths
count = 0
cropped_array_shape = [363, 358, 3]

for i in range(3):
    for j in range(9):
        #reading swath array 2D-txt file
        reshaped_swath_array = np.loadtxt("txt_files/swath_"
                                          + str(i + 1) + "_"
                                          + str(j + 1) + ".txt")

        #reshaping to 3D-array
        original_swath_array = (reshaped_swath_array.reshape(
                                reshaped_swath_array.
                                shape[0], reshaped_swath_array
                                .shape[1] // swath_array_shapes
                                [count][2],
                                swath_array_shapes[count][2]))

        #plotting swath RGB image
        f = plt.figure(figsize = (original_swath_array.shape[1]
                                  / 200, original_swath_array
                                  .shape[0] / 200))
        plt.axis('off')
        plt.imshow(original_swath_array, cmap='gray', vmin=0,
```

```
            vmax=255)

#saving swath image as png file
plt.savefig('png_files/swath_' + str(i + 1) + '_'
            + str(j + 1) + '.png')

#swath counter
count = count + 1

#number of crooped samples for each row in each swath
if i == 0:
    samples = 19
elif i == 1:
    samples = 17
else:
    samples = 16

#looping through samples in a given swath
for k in range(4):
    for l in range(samples):

        #reading cropped array 2D-txt file
        reshaped_cropped_array = (np.loadtxt(
            "txt_files/cropped_" + str(i + 1) + "_"
            + str(j + 1) + "_" + str(k + 1) + "_"
            + str(l + 1) + ".txt"))

        #reshaping to 3D-array
        original_cropped_array = (
            reshaped_cropped_array.reshape(
            reshaped_cropped_array.shape[0],
            reshaped_cropped_array.shape[1]
            // cropped_array_shape[2],
            cropped_array_shape[2]))

        #plotting cropped RGB image
        f = plt.figure(figsize = (original_cropped_array
                        .shape[1] / 50, original_cropped_array
                        .shape[0] / 50))
```

```
                plt.axis('off')
                plt.imshow(original_cropped_array, cmap='gray',
                           vmin=0, vmax=255)

                #saving cropped image as png file
                (plt.savefig('png_files/cropped_' + str(i + 1) + '_'
                             + str(j + 1) + '_' + str(k + 1) + '_'
                             + str(l + 1) + '.png'))
```

A.3 split_train_test_dataset.ipynb

```
# libraries
import numpy as np
import matplotlib.pyplot as plt
import random

# creating labels for image samples:
# 1 - totally deforested
# 2 - partially deforested
# 3 - partially preserved
# 4 - totally preserved
labels = np.array([[
   2,3,3,2,2,2,2,1,1,2,3,3,3,4,4,4,4,4,4, # swath 1 split 1 line 1
   3,2,2,2,1,1,1,1,1,1,1,2,3,3,4,4,4,4,4, # swath 1 split 1 line 2
   3,4,3,2,1,1,1,1,1,1,1,1,1,1,3,4,4,4,4, # swath 1 split 1 line 3
   1,3,2,1,1,1,1,1,1,1,1,1,1,1,1,3,4,4,4, # swath 1 split 1 line 4

   2,2,1,1,1,1,1,1,1,1,1,1,1,1,2,3,4,4,4, # swath 1 split 2 line 1
   3,1,1,1,1,2,2,2,2,1,1,2,2,1,2,2,4,4,4, # swath 1 split 2 line 2
   4,3,3,2,1,1,1,1,1,1,1,1,2,2,2,3,3,4,4, # swath 1 split 2 line 3
   3,2,1,2,2,1,3,2,1,1,1,2,2,3,3,2,1,3,4, # swath 1 split 2 line 4

   2,3,3,3,2,2,3,2,1,2,1,1,2,2,1,1,1,2,4, # swath 1 split 3 line 1
   2,2,3,3,2,1,2,1,2,1,1,1,1,1,1,1,1,1,1, # swath 1 split 3 line 2
   2,3,1,1,1,1,2,1,1,1,1,1,1,1,1,1,1,1,1, # swath 1 split 3 line 3
   1,3,3,1,1,2,1,1,1,1,1,1,1,1,1,1,1,1,1, # swath 1 split 3 line 4

   1,2,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1, # swath 1 split 4 line 1
   1,1,1,1,1,1,1,1,1,1,1,1,1,2,2,3,2,3,4, # swath 1 split 4 line 2
```

```
1,1,1,2,3,2,1,1,1,1,1,1,1,1,1,1,2,3,3, # swath 1 split 4 line 3
1,1,2,2,1,1,1,1,2,1,1,1,1,1,1,4,4,4,4, # swath 1 split 4 line 4

1,1,2,2,1,1,1,1,1,1,1,1,1,1,2,3,3,3,3, # swath 1 split 5 line 1
2,4,1,2,1,1,1,1,1,1,1,1,1,3,3,4,3,4,4, # swath 1 split 5 line 2
4,4,1,1,1,1,1,1,1,1,1,1,1,3,4,4,4,4,4,4, # swath 1 split 5 line 3
4,4,4,2,2,3,3,3,4,3,3,3,4,4,4,4,4,4,4, # swath 1 split 5 line 4

3,4,4,3,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4, # swath 1 split 6 line 1
4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4, # swath 1 split 6 line 2
4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4, # swath 1 split 6 line 3
4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4, # swath 1 split 6 line 4

4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4, # swath 1 split 7 line 1
4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4, # swath 1 split 7 line 2
4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4, # swath 1 split 7 line 3
4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4, # swath 1 split 7 line 4

4,4,4,3,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4, # swath 1 split 8 line 1
4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4, # swath 1 split 8 line 2
4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4, # swath 1 split 8 line 3
4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4, # swath 1 split 8 line 4

4,4,4,4,3,3,3,3,4,4,4,4,4,4,4,4,4,4,4,4, # swath 1 split 9 line 1
4,4,4,4,4,4,4,4,4,4,3,3,3,4,4,4,4,4,4,4, # swath 1 split 9 line 2
4,3,4,4,4,4,4,3,3,4,4,4,4,4,4,4,4,4,4,4, # swath 1 split 9 line 3
4,4,4,4,4,4,3,2,1,1,1,2,2,2,2,3,2,3,3,2, # swath 1 split 9 line 4

1,1,1,1,1,1,1,2,3,2,3,3,2,2,3,3,3,     # swath 2 split 1 line 1
2,1,1,1,1,1,1,1,1,1,1,2,1,2,2,3,3,     # swath 2 split 1 line 2
1,1,1,1,1,1,2,1,1,1,2,1,1,1,1,2,2,     # swath 2 split 1 line 3
1,2,2,1,1,1,2,1,1,1,1,1,2,2,2,3,3,     # swath 2 split 1 line 4

1,1,1,1,1,1,1,1,1,1,1,1,1,2,2,3,4,1,     # swath 2 split 2 line 1
1,1,1,2,1,1,1,1,2,1,1,1,2,2,1,2,2,     # swath 2 split 2 line 2
2,2,1,3,2,2,1,2,2,2,1,1,1,1,1,1,2,     # swath 2 split 2 line 3
3,3,3,2,1,2,2,1,2,2,1,1,1,1,1,1,1,     # swath 2 split 2 line 4

3,3,3,2,1,2,2,1,3,2,1,1,1,1,2,2,2,     # swath 2 split 3 line 1
```

```
3,3,1,3,3,2,1,2,1,1,1,1,1,2,2,3,2,     # swath 2 split 3 line 2
2,1,3,3,3,3,2,2,2,2,1,1,1,1,2,3,1,     # swath 2 split 3 line 3
3,3,2,2,2,3,4,3,1,1,2,1,1,1,2,2,1,     # swath 2 split 3 line 4

3,3,2,2,1,1,2,2,1,2,1,1,1,2,1,1,1,     # swath 2 split 4 line 1
3,3,2,1,2,1,1,1,2,2,2,2,1,1,1,1,1,     # swath 2 split 4 line 2
2,2,1,2,2,2,2,2,2,2,3,2,2,1,1,2,1,     # swath 2 split 4 line 3
3,2,3,3,2,2,3,3,3,3,3,2,1,1,3,3,2,     # swath 2 split 4 line 4

3,3,3,3,3,2,3,3,2,3,3,2,1,2,2,4,2,     # swath 2 split 5 line 1
4,4,3,3,2,3,2,2,1,1,1,2,3,2,3,4,4,     # swath 2 split 5 line 2
4,4,4,3,2,3,3,2,2,2,3,2,2,2,2,3,3,     # swath 2 split 5 line 3
2,4,3,3,3,2,2,3,3,3,2,1,2,2,2,3,3,     # swath 2 split 5 line 4

3,3,2,3,3,2,3,3,3,3,2,2,2,2,3,3,4,     # swath 2 split 6 line 1
3,3,3,3,2,3,3,1,1,3,3,3,3,3,4,4,4,     # swath 2 split 6 line 2
3,3,2,3,2,3,2,3,3,2,3,4,4,4,4,4,4,     # swath 2 split 6 line 3
3,3,4,4,3,3,3,2,2,3,4,4,4,4,4,4,4,     # swath 2 split 6 line 4

3,4,4,3,4,2,3,3,4,4,4,4,4,4,4,4,4,     # swath 2 split 7 line 1
4,3,3,3,3,3,3,4,4,3,4,4,4,4,4,4,4,     # swath 2 split 7 line 2
4,4,4,3,4,4,4,4,4,4,4,4,4,4,4,4,4,     # swath 2 split 7 line 3
3,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,     # swath 2 split 7 line 4

4,4,4,3,3,4,4,4,4,4,4,4,4,4,4,4,4,     # swath 2 split 8 line 1
4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,     # swath 2 split 8 line 2
3,4,4,4,4,4,4,4,4,4,3,4,4,4,4,4,4,     # swath 2 split 8 line 3
4,3,4,4,4,4,4,4,4,4,4,4,4,3,4,4,4,     # swath 2 split 8 line 4

4,3,4,4,4,4,4,4,4,4,4,4,4,4,4,4,3,     # swath 2 split 9 line 1
4,4,3,4,4,4,4,4,4,4,4,3,4,4,4,4,4,     # swath 2 split 9 line 2
4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,     # swath 2 split 9 line 3
4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,     # swath 2 split 9 line 4

3,4,3,3,3,4,3,4,4,4,4,4,3,2,3,3,       # swath 3 split 1 line 1
4,3,3,2,4,4,4,4,4,4,4,4,3,1,1,1,       # swath 3 split 1 line 2
4,4,4,4,3,3,4,4,4,4,4,4,4,3,3,2,       # swath 3 split 1 line 3
4,4,4,3,3,3,4,4,4,4,4,3,4,4,3,2,       # swath 3 split 1 line 4
```

```
4,2,3,3,4,4,4,4,4,4,3,3,3,4,3,2,      # swath 3 split 2 line 1
4,4,3,4,4,4,3,4,3,3,2,1,2,3,4,3,      # swath 3 split 2 line 2
4,4,4,4,4,4,4,4,3,2,2,3,2,2,2,3,      # swath 3 split 2 line 3
3,3,3,4,4,4,4,2,3,4,3,3,2,3,2,3,      # swath 3 split 2 line 4

3,3,3,2,3,3,2,2,3,3,2,1,2,2,2,3,      # swath 3 split 3 line 1
2,3,4,3,2,3,3,2,3,2,2,2,1,1,1,1,      # swath 3 split 3 line 2
2,3,3,2,3,2,2,3,2,2,2,2,1,1,1,2,      # swath 3 split 3 line 3
2,3,3,4,2,3,3,4,4,3,3,2,2,1,1,2,      # swath 3 split 3 line 4

2,3,3,3,2,3,2,3,4,4,3,2,2,2,2,2,      # swath 3 split 4 line 1
2,3,4,4,2,3,2,3,3,4,3,3,3,2,3,3,      # swath 3 split 4 line 2
2,4,4,4,3,2,3,2,2,3,2,3,3,3,4,4,      # swath 3 split 4 line 3
3,4,4,4,3,3,3,3,2,3,3,3,3,3,3,4,      # swath 3 split 4 line 4

3,4,4,3,2,4,2,2,3,2,2,2,2,3,4,4,      # swath 3 split 5 line 1
2,3,3,4,3,4,3,2,3,4,4,4,4,4,3,3,      # swath 3 split 5 line 2
3,4,4,3,3,3,4,3,3,2,3,3,4,3,2,3,      # swath 3 split 5 line 3
3,4,3,2,2,3,3,4,4,3,4,3,4,4,3,3,      # swath 3 split 5 line 4

4,4,3,3,4,3,3,3,3,3,4,4,3,3,3,3,      # swath 3 split 6 line 1
3,4,3,3,3,3,3,4,4,3,3,4,4,3,4,3,3,    # swath 3 split 6 line 2
4,4,3,3,3,4,4,4,3,3,2,3,3,4,3,3,      # swath 3 split 6 line 3
3,4,4,4,4,4,4,4,4,4,3,4,4,3,3,2,      # swath 3 split 6 line 4

3,4,4,4,4,4,4,4,4,4,4,3,3,3,3,3,      # swath 3 split 7 line 1
4,4,4,4,4,4,4,4,4,4,4,3,3,3,4,4,3,    # swath 3 split 7 line 2
4,4,4,4,4,4,4,4,4,4,4,4,3,3,4,4,3,    # swath 3 split 7 line 3
3,4,4,4,4,4,4,4,4,4,4,4,4,4,3,3,3,    # swath 3 split 7 line 4

3,4,4,4,4,4,4,4,3,3,3,4,3,3,3,3,      # swath 3 split 8 line 1
4,4,4,4,4,4,4,4,3,2,3,3,3,3,3,2,      # swath 3 split 8 line 2
4,4,4,4,4,4,4,4,4,4,4,3,3,3,3,3,3,    # swath 3 split 8 line 3
4,4,4,4,4,4,4,4,4,4,4,4,3,4,4,4,4,    # swath 3 split 8 line 4

4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,    # swath 3 split 9 line 1
4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,  # swath 3 split 9 line 2
4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,  # swath 3 split 9 line 3
4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4     # swath 3 split 9 line 4
```

```
    ]]).transpose()

#image sample formats
cropped_array_shape = [363, 358, 3]
number_swaths = 3
number_splits = 9
number_sample_rows = 4

#empty image sample list
image_samples = []

#loading image samples txt files
for i in range(number_swaths):

    #number of columns is different for each swath
    if i == 0:
        number_sample_columns = 19
    elif i == 1:
        number_sample_columns = 17
    else:
        number_sample_columns = 16

    for j in range(number_splits):
        for k in range(number_sample_rows):
            for l in range(number_sample_columns):

                #reading cropped array 2D-txt file
                reshaped_cropped_array = (np.loadtxt(
                    "txt_files/cropped_" + str(i + 1) + "_"
                    + str(j + 1) + "_" + str(k + 1) + "_"
                    + str(l + 1) + ".txt"))

                #reshaping to 3D-array
                original_cropped_array = (
                    reshaped_cropped_array.reshape(
                    reshaped_cropped_array.shape[0],
                    reshaped_cropped_array.shape[1]
                    // cropped_array_shape[2],
                    cropped_array_shape[2]))
```

```
                #filling image sample list up
                image_samples.append(original_cropped_array)

#shuffling the samples indexes
random_indexes = np.arange(len(labels))
np.random.seed(len(labels))
np.random.shuffle(random_indexes)

#setting the size of training, validating and testing datasets
train_images_size = round(len(labels)*0.7)
valid_images_size = round(len(labels)*0.15)
test_images_size = round(len(labels)*0.15)

#empty datasets
train_images = []
valid_images = []
test_images = []
train_labels = []
valid_labels = []
test_labels = []

#splitting the samples in train, val and testing datasets
for i in range(train_images_size):
    train_images.append(image_samples[random_indexes[i]])
    train_labels.append(labels[random_indexes[i]])

for i in range(train_images_size, train_images_size
               + valid_images_size):
    valid_images.append(image_samples[random_indexes[i]])
    valid_labels.append(labels[random_indexes[i]])

for i in range(train_images_size + valid_images_size,
               train_images_size + valid_images_size
               + test_images_size):
    test_images.append(image_samples[random_indexes[i]])
    test_labels.append(labels[random_indexes[i]])

#changing list to array
```

```
train_images = np.array(train_images)
valid_images = np.array(valid_images)
test_images = np.array(test_images)
train_labels = np.array(train_labels)
valid_labels = np.array(valid_labels)
test_labels = np.array(test_labels)

#saving arrays as npy files
np.save("npy_files/train_images.npy", train_images)
np.save("npy_files/valid_images.npy", valid_images)
np.save("npy_files/test_images.npy", test_images)
np.save("npy_files/train_labels.npy", train_labels)
np.save("npy_files/valid_labels.npy", valid_labels)
np.save("npy_files/test_labels.npy", test_labels)

#exploratory data analysis
class_names = ['totalmente desmatado',
               'parcialmente desmatado',
               'parcialmente preservado',
               'totalmente preservado']

plt.figure(figsize=(20,20))
for i in range(25):
    plt.subplot(5, 5, i + 1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i])
    plt.xlabel(class_names[train_labels[i][0] - 1])
plt.show()
```

A.4 *cnn_training.ipynb*

```
#accessing Google Drive

from google.colab import drive
drive.mount('/content/drive')

#libraries
```

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import MaxNLocator
import tensorflow as tf

from tensorflow import keras
from tensorflow.keras import datasets, layers, models
from keras.models import Sequential
from keras.layers.core import Flatten, Dense, Dropout
from keras.layers.convolutional import Convolution2D, MaxPooling2D

#functions

def number_of_labels(number_labels):
  if number_labels == 4:

    #keeping 4 labels
    #(changing label range)
    train_labels[train_labels == 1] = 0
    valid_labels[valid_labels == 1] = 0
    train_labels[train_labels == 2] = 1
    valid_labels[valid_labels == 2] = 1
    train_labels[train_labels == 3] = 2
    valid_labels[valid_labels == 3] = 2
    train_labels[train_labels == 4] = 3
    valid_labels[valid_labels == 4] = 3

  elif number_labels == 3:

    #changing from 4 to 3 labels
    #(merging partially deforested and partially preserved)
    train_labels[train_labels == 3] = 2
    valid_labels[valid_labels == 3] = 2
    train_labels[train_labels == 1] = 0
    valid_labels[valid_labels == 1] = 0
    train_labels[train_labels == 2] = 1
    valid_labels[valid_labels == 2] = 1
    train_labels[train_labels == 4] = 2
```

```
    valid_labels[valid_labels == 4] = 2

  elif number_labels == 2:

    #changing from 4 to 2 labels
    #(merging partially deforested, partially preserved
    #and totally deforested)
    train_labels[train_labels == 3] = 1
    valid_labels[valid_labels == 3] = 1
    train_labels[train_labels == 2] = 1
    valid_labels[valid_labels == 2] = 1
    train_labels[train_labels == 1] = 0
    valid_labels[valid_labels == 1] = 0
    train_labels[train_labels == 4] = 1
    valid_labels[valid_labels == 4] = 1

#loading datasets

train_images = np.load("/content/drive/MyDrive/TCC2/npy_files/"
                       + "train_images.npy")
valid_images = np.load("/content/drive/MyDrive/TCC2/npy_files/"
                       + "valid_images.npy")
train_labels = np.load("/content/drive/MyDrive/TCC2/npy_files/"
                       + "train_labels.npy")
valid_labels = np.load("/content/drive/MyDrive/TCC2/npy_files/"
                       + "valid_labels.npy")

#choosing the number of labels

number_labels = int(input("Number of labels? "))
number_epochs = int(input("Number of epochs? "))

number_of_labels(number_labels)

#creating convolutional model

model = models.Sequential()

model.add(layers.Conv2D(64, (3, 3), activation='relu',
```

```
                        input_shape=(363, 358, 3)))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(256, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(512, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(1024, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Flatten())
model.add(layers.Dense(1024, activation='relu'))
model.add(layers.Dropout(rate=0.5))
model.add(layers.Dense(1024, activation='relu'))
model.add(layers.Dropout(rate=0.5))
model.add(layers.Dense(number_labels, activation='softmax'))

#compiling the model

model.compile(optimizer = 'Adam',
              loss = 'sparse_categorical_crossentropy',
              metrics = ['accuracy'])

#training the model

history = model.fit(x = train_images,
                    y = train_labels,
                    validation_data = (valid_images, valid_labels),
                    batch_size = 10,
                    epochs = number_epochs,
                    verbose = 2)

#evaluating the model
```

```
test_loss, test_acc = model.evaluate(valid_images, valid_labels,
                                     verbose=2)

#plotting the chart

plt.figure().gca().xaxis.set_major_locator(MaxNLocator(integer=True))
plt.plot(list(np.arange(1,number_epochs+1)),
         history.history['accuracy'], label='Treino')
plt.plot(list(np.arange(1,number_epochs+1)),
         history.history['val_accuracy'], label = 'Validação')
plt.axhline(test_acc, color='r',
            label=str(round(test_acc*100, 2)) + "%", linewidth=0.5)
plt.xlabel('Época')
plt.ylabel('Acurácia')
plt.title('Acurária do Treino/Validação por Época')
plt.ylim([0.5, 1])
plt.xlim([1, number_epochs])
plt.legend(loc='lower right')
plt.plot()

#saving the model

model.save('/content/drive/MyDrive/TCC2/models/cnn_' + str(number_labels)
           + '_Labels_' + str(number_epochs) + '_epochs.h5')

#model architecture summary

model.summary()
```

A.5 *cnn_testing.ipynb*

```
#accessing Google Drive

from google.colab import drive
drive.mount('/content/drive')

#libraries
import numpy as np
from tensorflow.keras.models import load_model
```

```
from sklearn.metrics import confusion_matrix
import itertools
import matplotlib.pyplot as plt

def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Matriz de Confusão',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
            horizontalalignment="center",
            color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('Classificação Verdadeira')
    plt.xlabel('Classificação Prevista')

def number_of_labels(number_labels):
  if number_labels == 4:
```

```
    #keeping 4 labels
    #(changing labels range)
    test_labels[test_labels == 1] = 0
    test_labels[test_labels == 2] = 1
    test_labels[test_labels == 3] = 2
    test_labels[test_labels == 4] = 3

  elif number_labels == 3:

    #changing from 4 to 3 labels
    #(merging partially deforested and partially preserved)
    test_labels[test_labels == 3] = 2
    test_labels[test_labels == 1] = 0
    test_labels[test_labels == 2] = 1
    test_labels[test_labels == 4] = 2

  elif number_labels == 2:

    #changing from 4 to 2 labels
    #(merging partially deforested, partially preserved
    #and totally deforested)
    test_labels[test_labels == 3] = 1
    test_labels[test_labels == 2] = 1
    test_labels[test_labels == 1] = 0
    test_labels[test_labels == 4] = 1

#loading testing dataset

test_images = np.load("/content/drive/MyDrive/TCC2/npy_files/"
                      + "test_images.npy")
test_labels = np.load("/content/drive/MyDrive/TCC2/npy_files/"
                      + "test_labels.npy")

#choosing the number of labels

number_labels = int(input("Number of labels? "))
number_epochs = int(input("Number of epochs? "))
```

```
number_of_labels(number_labels)

#loading the model

model = load_model('/content/drive/MyDrive/TCC2/models/cnn_'
                   + str(number_labels) + '_Labels_' + str(number_epochs)
                   + '_epochs.h5')
test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)

#making prediction with test dataset

predictions = model.predict(x=test_images, steps=len(test_images),
                            verbose=1)
predictions = np.round(predictions)

#plotting confusion matrix

cm = confusion_matrix(y_true=test_labels,
                      y_pred=np.argmax(predictions, axis=-1))
if number_labels == 4:
  cm_plot_labels = ['Totalmente Desmatado',
                    'Parcialmente Desmatado',
                    'Parcialmente Preservado',
                    'Totalmente Preservado']
elif number_labels == 3:
  cm_plot_labels = ['Totalmente Desmatado',
                    'Parcialmente Desamatado',
                    'Totalmente Preservado']
elif number_labels == 2:
  cm_plot_labels = ['Desmatado', 'Preservado']

plot_confusion_matrix(cm=cm, classes=cm_plot_labels)
```

APÊNDICE B – Imagens SAR depois de processadas

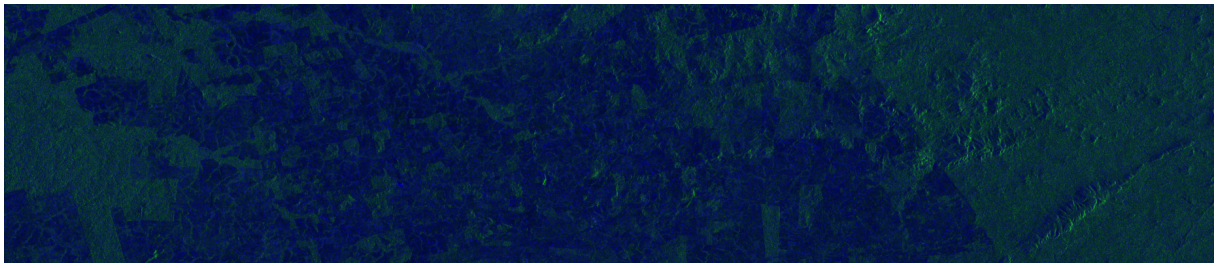


Figura 30 – Subfaixa 1, *Burst* 1

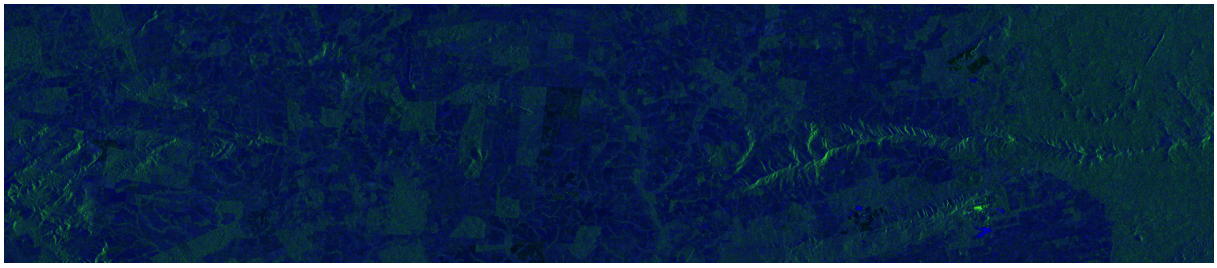


Figura 31 – Subfaixa 1, *Burst* 2

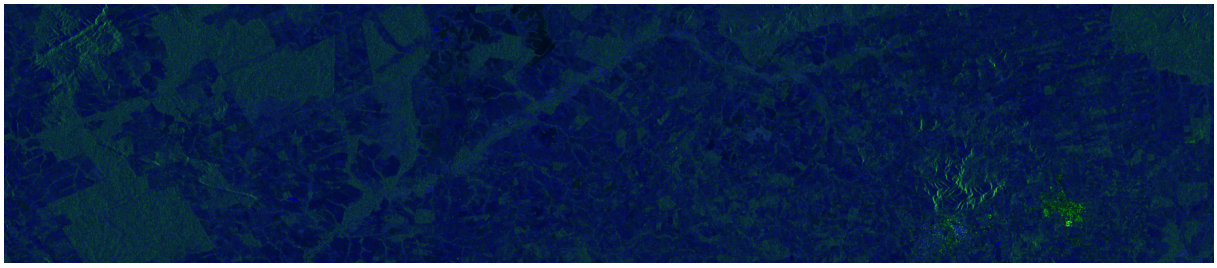


Figura 32 – Subfaixa 1, *Burst* 3

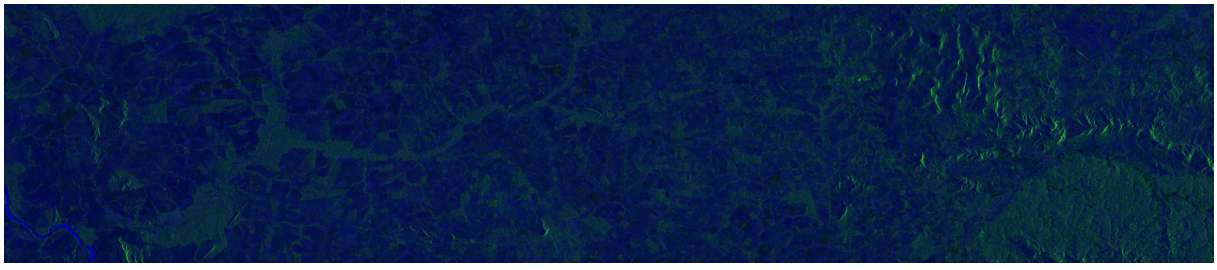


Figura 33 – Subfaixa 1, *Burst* 4

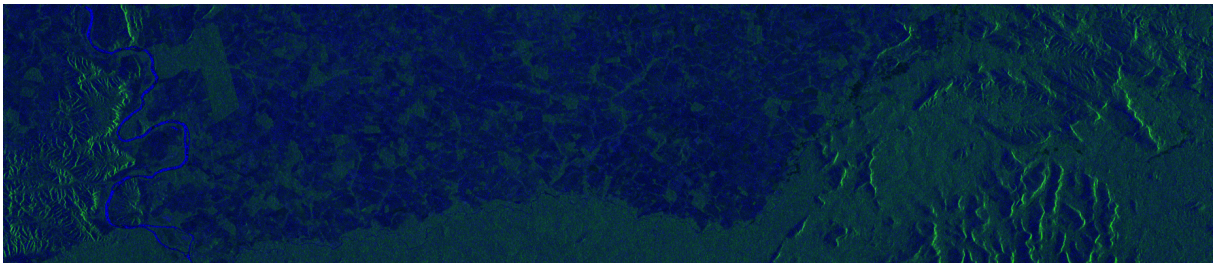


Figura 34 – Subfaixa 1, *Burst* 5

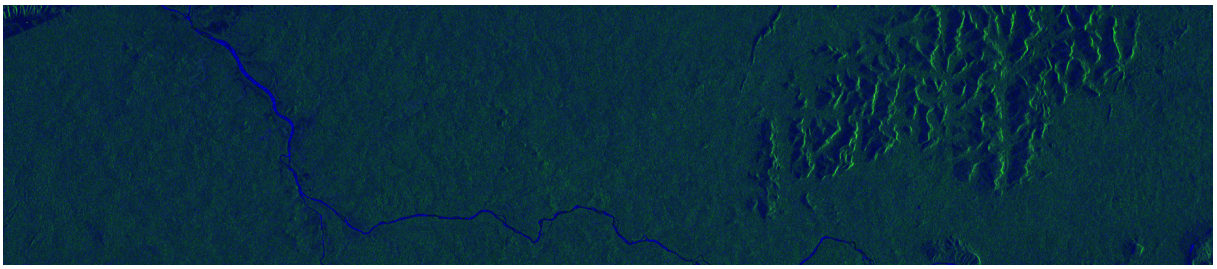


Figura 35 – Subfaixa 1, *Burst* 6

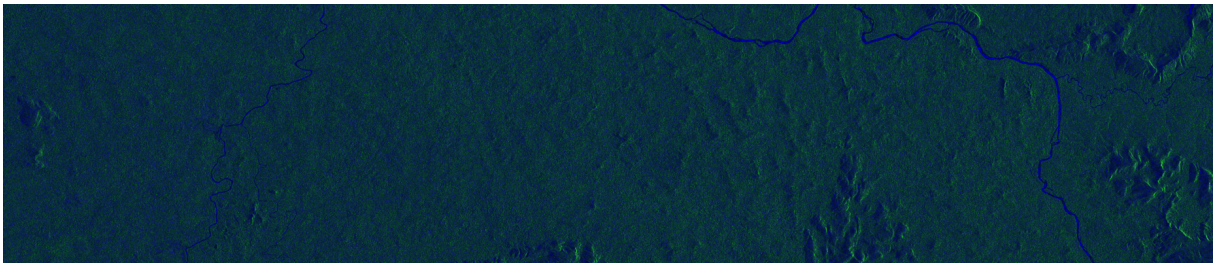


Figura 36 – Subfaixa 1, *Burst* 7

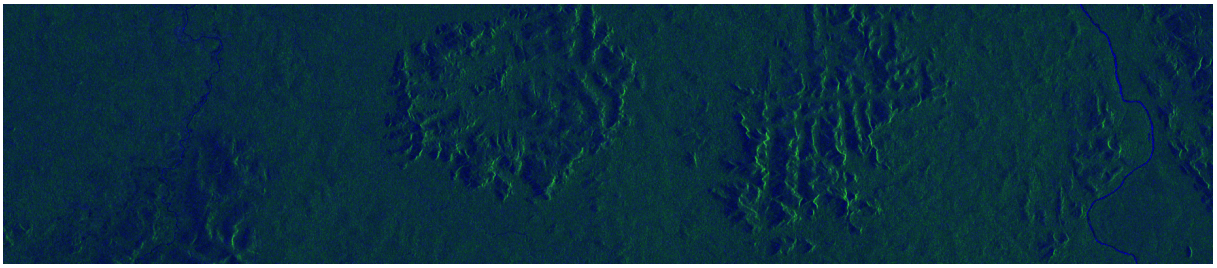


Figura 37 – Subfaixa 1, *Burst* 8

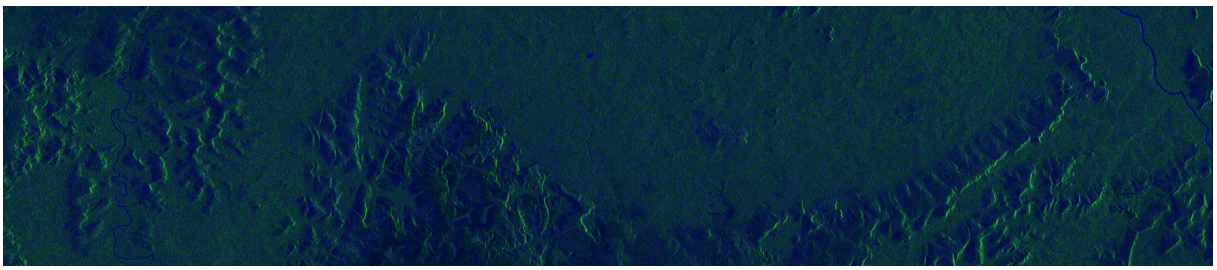


Figura 38 – Subfaixa 1, *Burst* 9

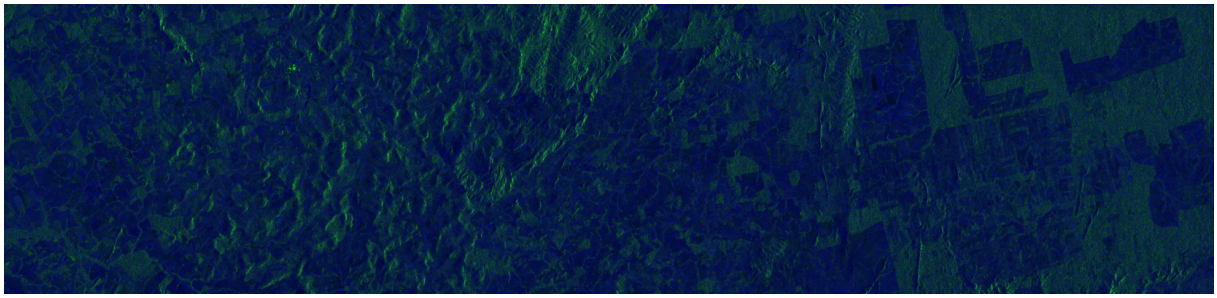


Figura 39 – Subfaixa 2, *Burst* 1

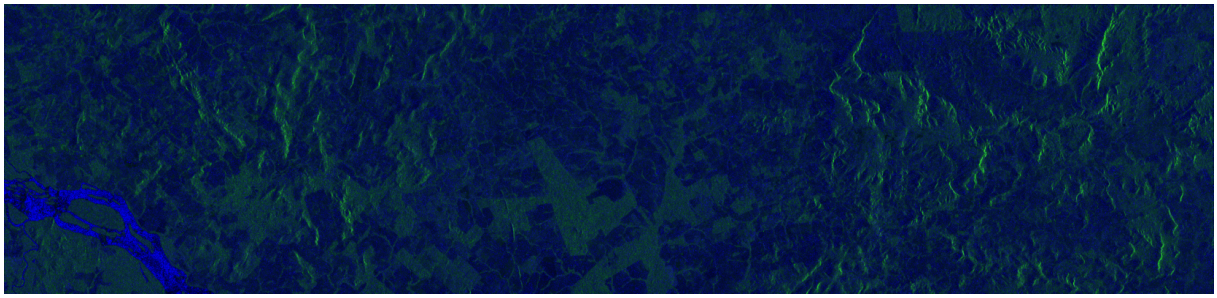


Figura 40 – Subfaixa 2, *Burst* 2

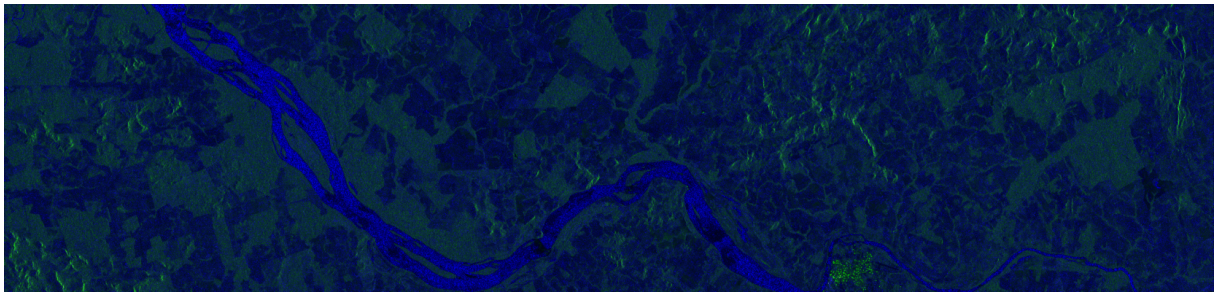


Figura 41 – Subfaixa 2, *Burst* 3

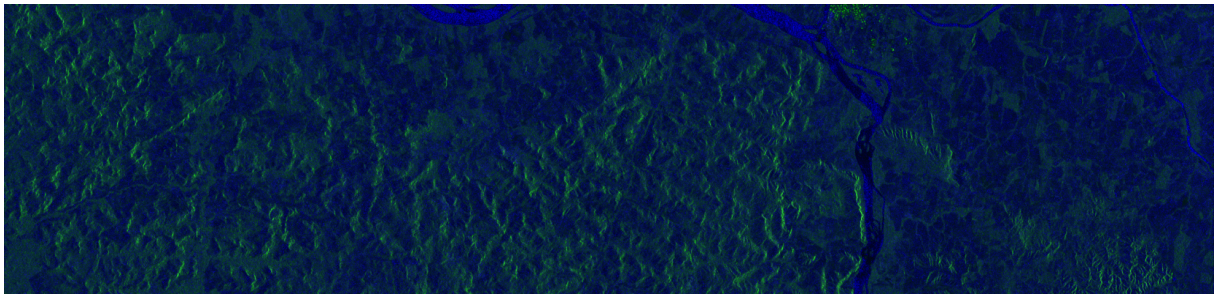


Figura 42 – Subfaixa 2, *Burst* 4

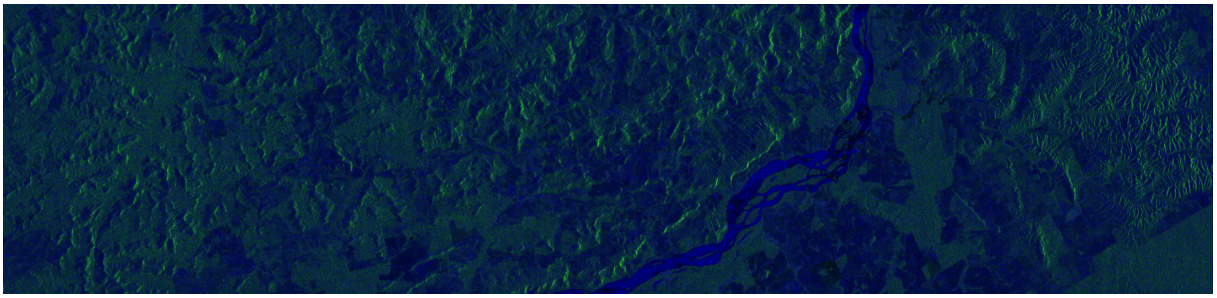


Figura 43 – Subfaixa 2, *Burst* 5

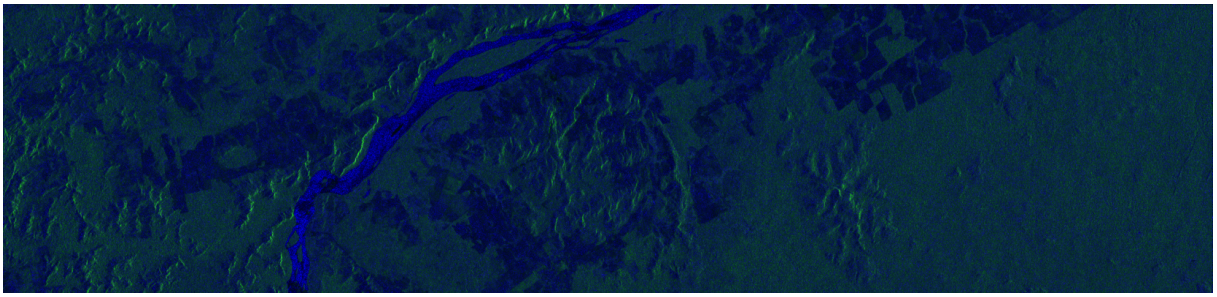


Figura 44 – Subfaixa 2, *Burst* 6

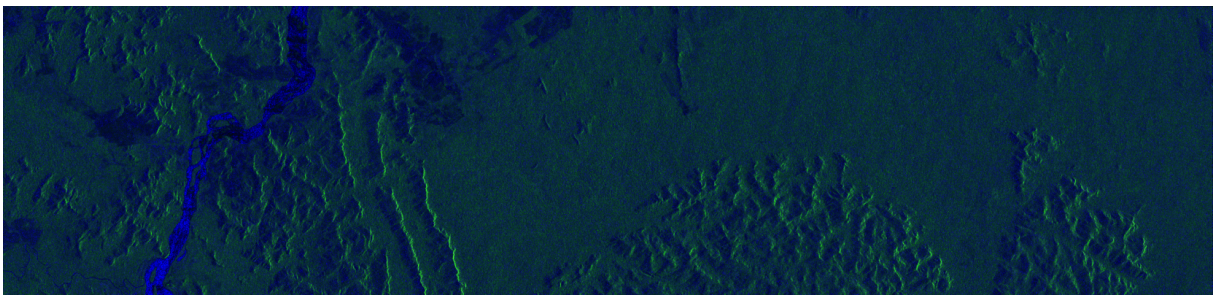


Figura 45 – Subfaixa 2, *Burst* 7

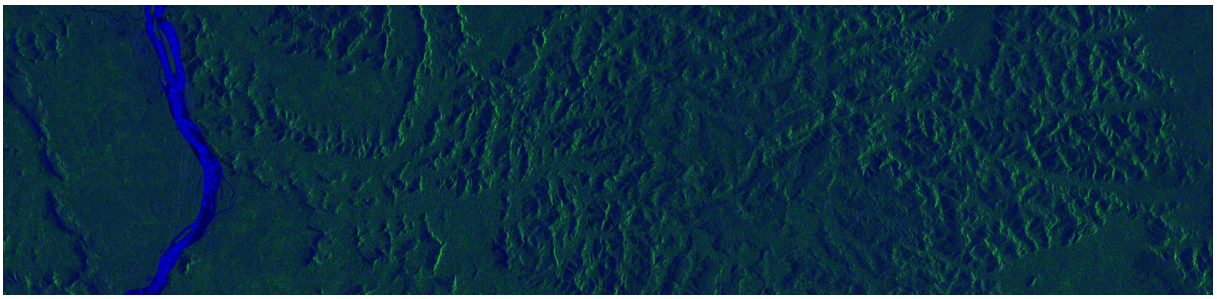


Figura 46 – Subfaixa 2, *Burst* 8

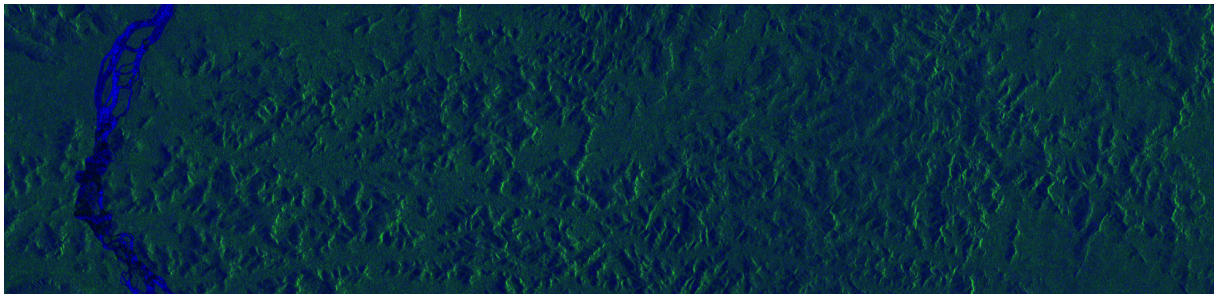


Figura 47 – Subfaixa 2, *Burst* 9

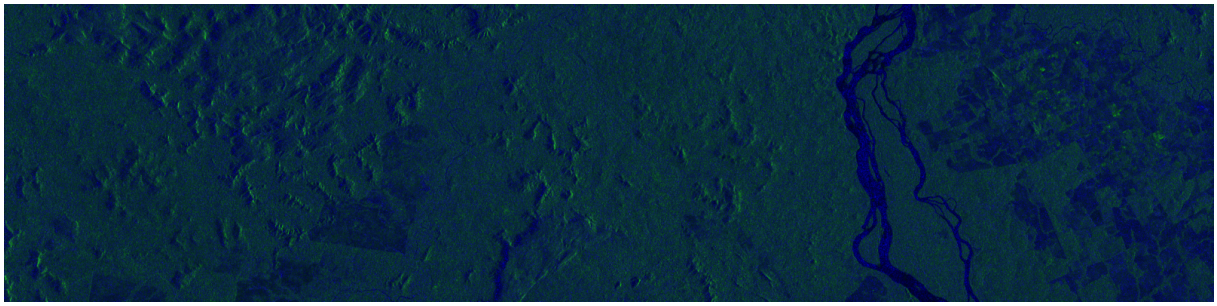


Figura 48 – Subfaixa 3, *Burst* 1

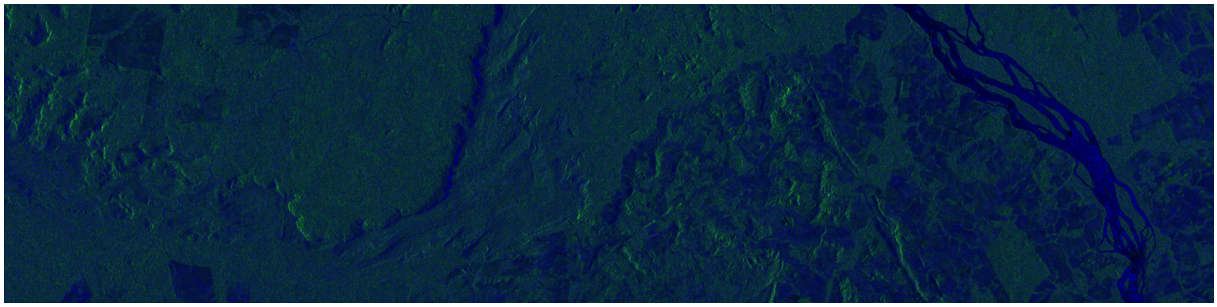


Figura 49 – Subfaixa 3, *Burst* 2

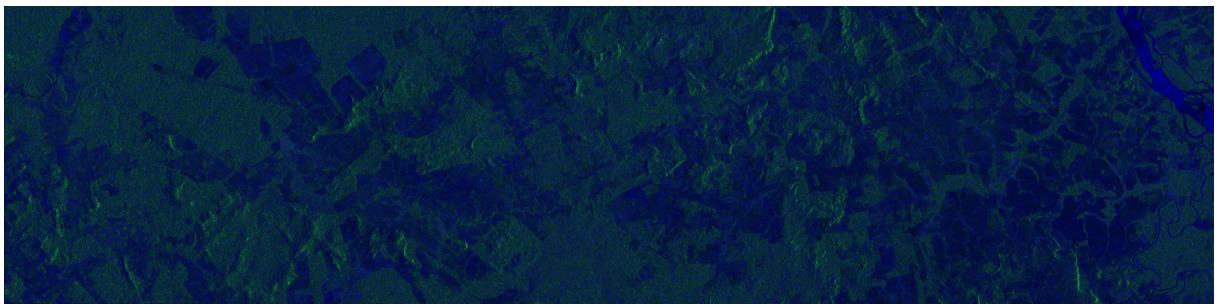


Figura 50 – Subfaixa 3, *Burst* 3

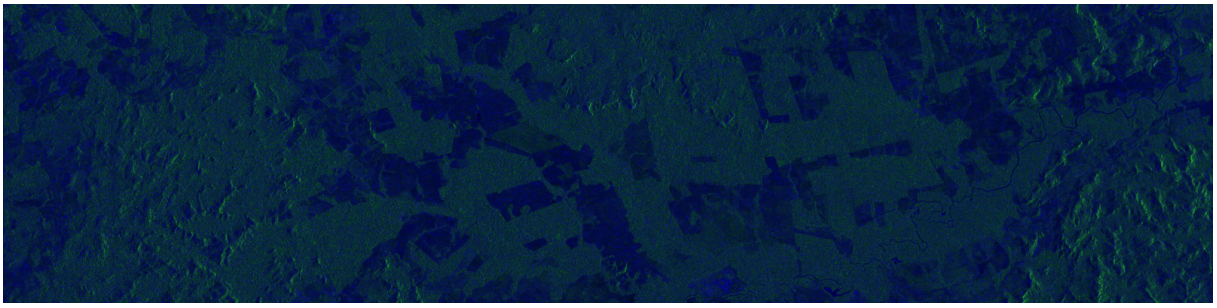


Figura 51 – Subfaixa 3, *Burst* 4

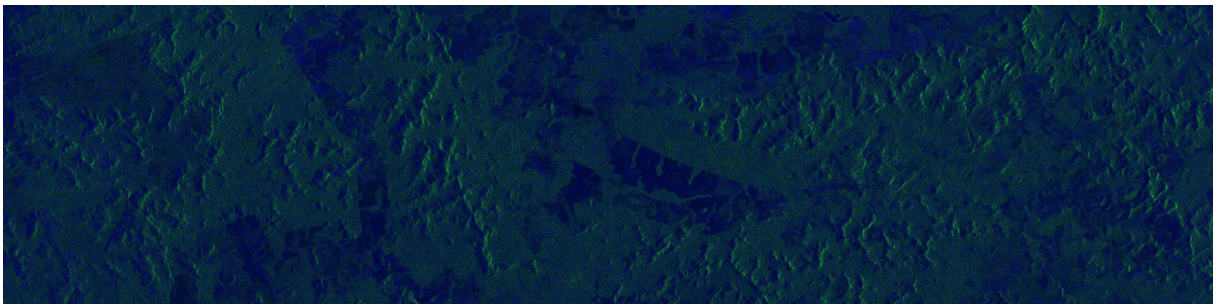


Figura 52 – Subfaixa 3, *Burst* 5

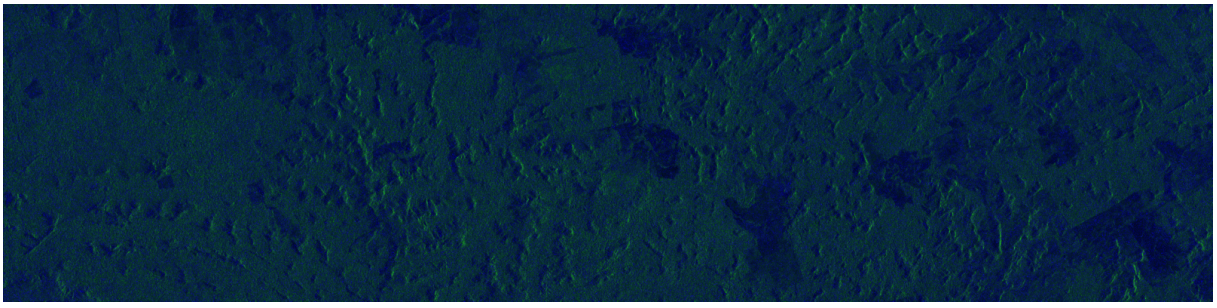


Figura 53 – Subfaixa 3, *Burst* 6

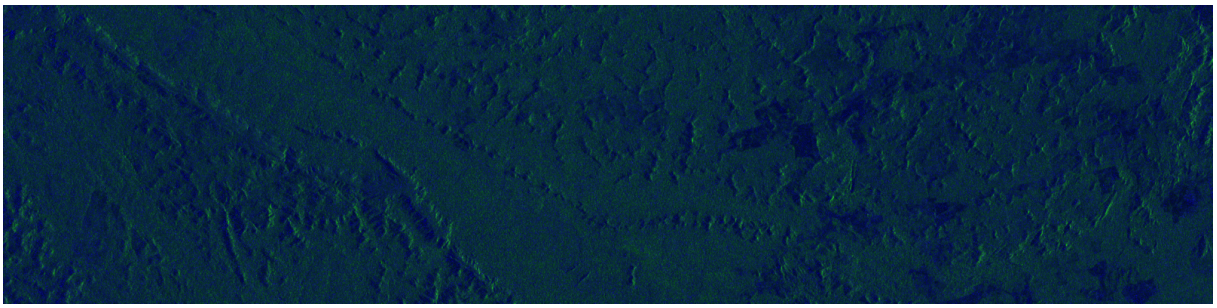


Figura 54 – Subfaixa 3, *Burst* 7

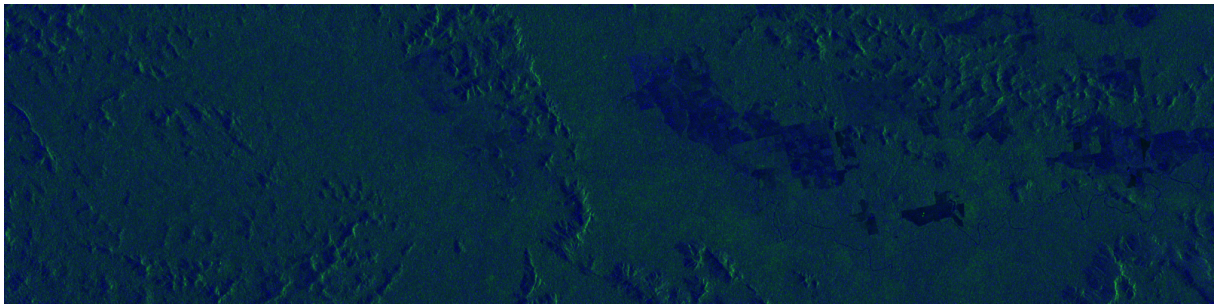


Figura 55 – Subfaixa 3, *Burst* 8

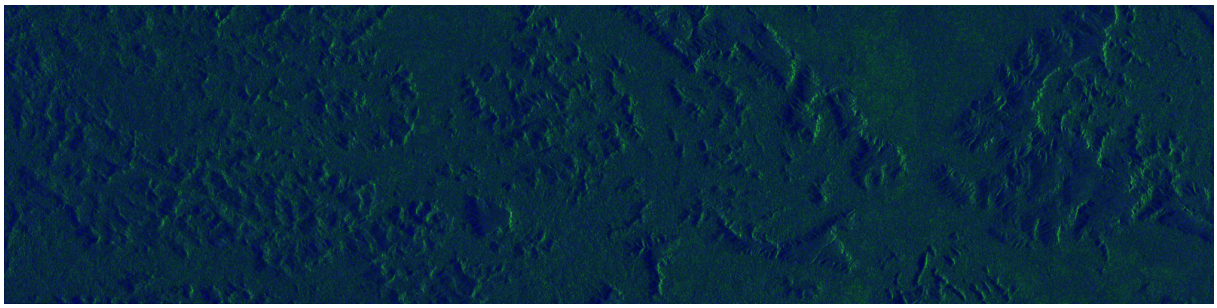


Figura 56 – Subfaixa 3, *Burst* 9

APÊNDICE C – Camadas da Rede Neural Convolucional

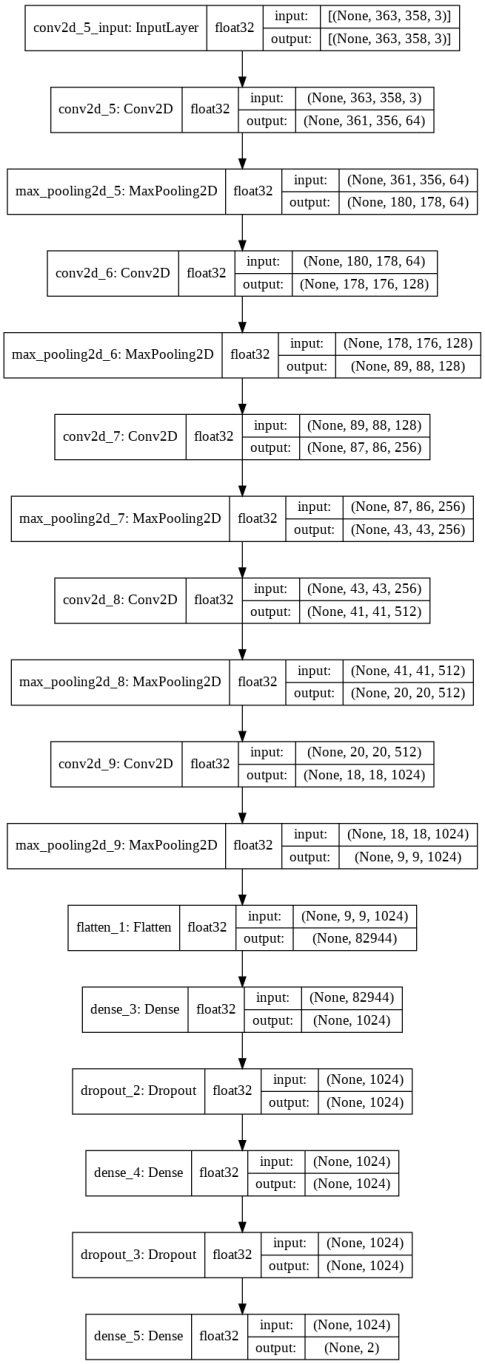


Figura 57 – Camadas da rede neural convolucional