UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS QUIXADÁ

BACHARELADO EM ENGENHARIA DE SOFTWARE

EDUARDO DA SILVA LOPES FILHO

COMPARANDO ALGORITMOS DE APRENDIZADO PROFUNDO PARA O PROBLEMA DE DETECÇÃO DE DISTRAÇÃO DE MOTORISTAS A PARTIR DE IMAGENS

QUIXADÁ

2018

EDUARDO DA SILVA LOPES FILHO

COMPARANDO ALGORITMOS DE APRENDIZADO PROFUNDO PARA O PROBLEMA DE DETECÇÃO DE DISTRAÇÃO DE MOTORISTAS A PARTIR DE IMAGENS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software do Campus Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Software.
Área de Concentração: Computação.

Orientador: Prof. Me. Regis Pires Magalhães

Coorientadora: Profa. Dra. Ticiana Linhares Coelho da Silva

QUIXADÁ

2018

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

L851c Lopes Filho, Eduardo da Silva.
Comparando algoritmos de aprendizado profundo para o problema de detecção de distração de motoristas a partir de imagens / Eduardo da Silva Lopes Filho. – 2018.
49 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Engenharia de Software, Quixadá, 2018.
Orientação: Prof. Me. Regis Pires Magalhães.
Coorientação: Prof. Dr. Ticiana Linhares Coelho da Silva.

1. Aprendizagem Profunda. 2. Visão Computacional. 3. Rede Neural Convolucional. I. Título.
CDD 005.1

EDUARDO DA SILVA LOPES FILHO

COMPARANDO ALGORITMOS DE APRENDIZADO PROFUNDO PARA O PROBLEMA DE DETECÇÃO DE DISTRAÇÃO DE MOTORISTAS A PARTIR DE IMAGENS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software do Campus Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Software. Área de Concentração: Computação.

Aprovada em: ____/____/_____

BANCA EXAMINADORA

Prof. Me. Regis Pires Magalhães (Orientador)
Universidade Federal do Ceará (UFC)

Profa. Dra. Ticiana Linhares Coelho da Silva (Coorientadora)
Universidade Federal do Ceará (UFC)

Prof. Dr. Marcos Antonio de Oliveira
Universidade Federal do Ceará (UFC)

À minha família e minha namorada e amiga.

AGRADECIMENTOS

Agradeço primeiramente aos meus pais Eduardo e Geny, por todos os sacrifícios feitos por mim, não só durante a faculdade, mas sempre, por nunca tentar limitar meus sonhos e sempre expandir meus horizontes. À minha mãe, por sempre dar um jeito de me ajudar. Ao meu pai, por todos os ensinamentos me passados enquanto trabalhávamos juntos em nosso sítio. Obrigado por sempre acreditarem em mim!

Agradeço à Deus por sempre me proporcionar sabedoria para que eu possa sempre tomar as melhores decisões possíveis.

Agradeço aos meus irmãos Amaro Sérgio, Bruno e minha cunhada Paloma, pelo apoio constante e por todas as vezes que acordaram cedo ou foram dormir tarde para me buscarem ou levarem para pegar ônibus para ir à faculdade. Vocês são incríveis!

Aos meus avos Sérgio, Elvira e Elizete por serem sempre exemplos de sabedoria, conduta, caráter, e honestidade para que eu possa sempre me guiar pelos seus ensinamentos.

Agradeço à minha sobrinha Maria Eduarda, por alegrar minha vida e meus fins de semana. Estude muito e seja melhor do que eu! Você tem um grande futuro pela frente.

Agradeço à Thacyla Milena, pela sua amizade, por todo carinho, amor, companheirismo, por ter me escutado, pelos conselhos, por ter me dado força para continuar nesta caminhada e por sempre estar do meu lado, mesmo com tanta distância.

Agradeço aos meus tios Joaquim, Raimundo e Cláudia e aos meus primos Sérgio e Bruna por todo o apoio sempre e principalmente pela grande ajuda em um dos momentos mais críticos e difíceis da minha graduação.

Agradeço ao Professor Regis Pires Magalhães por ter aceitado ser meu co-orientador, em um trabalho passado e meu orientador neste trabalho, mesmo sem me conhecer. Obrigado por todos os conselhos, ensinamentos e pela ótima orientação neste trabalho! Foi uma grande honra para mim, ter você como orientador.

Agradeço à Professora Ticiana Linhares por ter me orientado e me acompanhado neste trabalho, por todas as ajudas, dicas, críticas, pelos conselhos, ensinamentos, seu esforço para sempre me ajudar e por toda sua gentileza. Foi um imenso prazer trabalhar com você.

Agradeço ao Professor Marcos Antonio de Oliveira por pelas orientações quando fui bolsista, quando fui seu orientando na versão anterior do meu Trabalho de Conclusão de Curso e por ter aceitado fazer parte da minha banca de defesa desse novo trabalho.

Agradeço à Professora Diana Braga por ser uma verdadeira mãe de todos os alunos

do curso de Engenharia de Software, por ter me dado a oportunidade de trabalhar com você em um projeto já no primeiro ano de faculdade. Que Deus abençoe você e sua família!

Agradeço ao Professor Carlos Igor Ramos Bandeira por ter me apresentado tão bem ao maravilhoso universo do Aprendizado de Máquina, que hoje é a área que eu ainda me vejo trabalhando daqui à 50 anos.

Agradeço aos meus amigos Marcos Flávio, Gustavo Aires, Júlio Serafim, Francisco Wanderson, José Murilo, Lucas Vieira, Caio Melo e Letícia Aguiar pela amizade, apoio e companheirismo durante essa jornada. Foi um prazer percorrer esse caminho ao lado de vocês! As noites acordados não foram em vão.

Agradeço aos meus amigos Rafael Costa e Isac Cavalcante por todas as grandes ajudas e dicas que me deram no desenvolvimento deste trabalho. Rafael, não vejo a hora de te ver 100% novamente.

Agradeço aos meus amigos e colegas da Trixlog, em especial ao Douglas Henrique que apostou em mim, ao Leidson Melo que me auxilia e ajuda todos os dias e ao Daniel Alves por ter acreditado em mim desde o começo e por ter me dado a oportunidade de trabalhar com flexibilidade quando eu precisava ir à Quixadá e além disso, por ser um humano incrível.

Agradeço, de coração, à toda comunidade acadêmica da UFC Campus Quixadá, por me proporcionar o melhor ambiente possível de aprendizado. À todos o funcionários do Campus por sempre fazerem o seu melhor para ajudar aos alunos, em especial ao Venício de Oliveira e a Natália Pinho.

Ao Doutorando em Engenharia Elétrica, Ednardo Moreira Rodrigues, e seu assistente, Alan Batista de Oliveira, aluno de graduação em Engenharia Elétrica, pela adequação do *template* utilizado neste trabalho para que o mesmo ficasse de acordo com as normas da biblioteca da Universidade Federal do Ceará (UFC).

“É melhor cair em contradição do que do oitavo andar.”

(Falcão)

RESUMO

Uma das principais causas de acidentes trânsito atualmente, é a distração de motoristas enquanto dirigem. Seja a distração pelo uso de celular ao volante ou por conversar olhando para o passageiro. Uma possível forma de diminuir o número de acidentes é alertar ao motorista quando ele estiver distraído. A tarefa mais difícil de avisar ao motorista quando estiver distraído, é detectar a distração. Uma forma de se conseguir isso é a partir do uso de Aprendizado Profundo. Dessa forma, este trabalho tem como objetivo descobrir a melhor técnica para classificação de imagens para o problema da detecção de distração de motoristas. Para isso, foram testadas diversas arquiteturas de Redes Neurais Convolucionais, para descobrir qual a melhor arquitetura para resolver este problema.

Palavras-chave: Aprendizado Profundo. Visão Computacional. Redes Neurais Convolucionais. VGG. Inception. ResNet

ABSTRACT

One of the major causes of traffic accidents today is the distraction of drivers while driving. It may be the distraction by using cell phones behind the wheel or by talking looking at the passenger. One possible way to reduce the number of accidents is to alert the driver when he is distracted. The most difficult task of telling the driver when he is distracted is to detect distraction. One way to achieve this is through the use of Deep Learning. Thus, this work aims to discover the best technique for classification of images for the problem of detecting the distraction of drivers. To this end, several architectures of Convolutional Neural Networks were tested to find out the best architecture to solve such problem.

Keywords: Deep Learning. Computer Vision. Convolutional Neural Networks. VGG. Inception. ResNet

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos, o uso de meios de transporte terrestres tem aumentado consideravelmente. Ao mesmo tempo em que o número de acidentes de trânsito tem crescido de forma análoga. Porém, o crescimento no número de acidentes não é uma simples proporção relacionada apenas ao número de veículos em circulação. Muitos outros fatores podem influenciar no número de acidentes de trânsito, como por exemplo: vias defeituosas, ingestão de bebidas alcoólicas ou substancias ilícitas, imprudência, distração do motoristas, dentre outros.

Dos fatores citados anteriormente, um dos maiores causadores de acidentes é o fator de distração do motorista. Estudos afirmam que dirigir usando celular causa efeitos parecidos com dirigir alcoolizado (PICKRELL, 2016). Em contrapartida, dos fatores citados anteriormente, o fator distração do motorista é um dos mais fáceis de se combater no momento em que se dirige.

Para se ter ideia, dos 89.396 acidentes registrados pela Polícia Rodoviária Federal, 34.439 - mais de 38% - foram causados por conta de distração do motorista (PRF Polícia Rodoviária Federal, 2018). Ou seja, cerca de 38% dos acidentes ocorridos em 2017 poderiam ter sido evitados com uma ação básica: prestar atenção unicamente no trânsito.

Infelizmente, ter a atenção voltada unicamente ao trânsito não é uma ação tão fácil de ser feita por diversos fatores. Entre eles:

- É comum que em grandes cidades motoristas passem horas no trânsito todos os dias no translado casa - trabalho;
- Passando-se muito tempo dirigindo, é comum receber ligações ou mensagens enquanto dirige;
- É difícil concentrar-se em uma única atividade durante muito tempo.

Em contrapartida, quando existe algo ou alguém "forçando"o motorista a focar apenas em dirigir, se torna menos provável que haja distração. Assim, dar à algo ou alguém a função exclusiva de não deixar o motorista perder a atenção no transito, se torna uma boa medida para tentar diminuir o número de acidentes causados por distração do motorista.

Entretanto, delegar essa tarefa a um humano não é viável, tanto financeiramente quanto tecnicamente - uma vez que com um tempo um humano pode se distrair também. Dessa forma, se torna mais viável buscar outra forma de avisar ao motorista que ele está se distraindo. Isso pode ser feito por meio de um sistema computacional que captura imagens, detecta se o motorista está distraído, e o avisa, caso esteja.

Um sistema computacional para realizar essa tarefa possui claramente três partes

bem distintas: uma que captura uma imagem; outra que consegue distinguir se um motorista está distraído baseado em uma imagem; e outra que, dado que o motorista está distraído, avisa ao motorista. Para a criação desse sistema o grande desafio é o de dar a um sistema computacional a habilidade de determinar se um motorista está distraído, baseado em imagens.

Imagens são para computadores apenas matrizes de números que representam as cores que o olho humano enxerga. Logo, um sistema que vai analisar imagens para determinar se um motorista está distraído, tomará essa decisão baseado em matrizes de números. Problemas desse tipo são difíceis, quando não impossíveis, de se resolver por meio de algoritmos "tradicionais" – onde as decisões são tomadas a partir de estruturas de decisão - uma vez que não existem, necessariamente, padrões do tipo: "Dada um matriz A, se o elemento $A_{i,j}$ possuir um determinado número x, então essa imagem representa um determinado objeto".

Para a resolução desse tipo de problema, normalmente, são utilizados algoritmos de Aprendizado de Máquina, que são algoritmos que conseguem aprender determinados padrões de forma automática, conseguindo detectar esses padrões e assim melhores resultados (GRUS, 2015).

Existem diversas técnicas de Aprendizado de Máquina. E uma das mais utilizadas atualmente é a de Rede Neural. Uma Rede Neural é um modelo preditivo motivado pela forma como o cérebro funciona. Imaginando-se o cérebro como uma coleção de neurônios conectados, cada neurônio examina as saídas dos neurônios que o alimentam, fazem um cálculo e, em seguida, disparam, se o cálculo exceder um determinado limiar, ou não, caso contrário (GRUS, 2015). Redes Neurais, normalmente, são organizadas em camadas. De forma que cada camada pode possuir vários neurônios. As camadas podem ser totalmente conectadas ou parcialmente conectadas, dependendo do modelo de Rede Neural.

Uma vez que o poder de processamento e o acesso a grandes quantidades de dados deixou de ser uma barreira, as Redes Neurais - e demais técnicas de Aprendizado de Máquina - passaram a ser mais estudadas. Logo começaram a surgir arquiteturas de Redes Neurais com muitas camadas, possibilitando o aprendizado de características cada vez mais específicas. Então surgiu o termo Aprendizado Profundo (*Deep Learning*) para identificar técnicas que utilizam Redes Neurais com um grande número de camadas e capaz de precessar um grande número de parâmetros (PATTERSON; GIBSON, 2017).

Um dos modelos de Aprendizado Profundo mais utilizados tem sido as Redes Neurais Convolucionais (*Convolutional Neural Networks*, ou CNN). Na verdade, as Redes

Neurais Convolucionais são as principais responsáveis pela popularização das técnicas de Aprendizado Profundo, juntamente com o avanço das Unidades de Processamento Gráficos (GPU) (PATTERSON; GIBSON, 2017).

Dessa forma, o presente trabalho tem foco no problema de descobrir se um motorista está distraído baseado em imagens estáticas. Como solução, este trabalho faz uma análise de diversas arquiteturas de Redes Neurais Convolucionais, que são testadas para descobrir qual a arquitetura mais adequada para este problema.

O objetivo principal deste trabalho é descobrir a melhor técnica para classificação de imagens para o problema da detecção de distração de motoristas. Além disso, ele traz como contribuições uma discussão acerca das diversas arquiteturas testadas, levantando hipóteses dos motivos pelos quais determinadas arquiteturas possuem melhores resultados que outras. Com isso será possível basear-se neste trabalho para compreender melhor a natureza deste problema, e aplicar soluções parecidas para problemas semelhantes. Além disso, a disponibilização de uma imagem Docker [1] com um ambiente com diversas ferramentas e bibliotecas para o uso de Aprendizado Profundo, que pode ser utilizada por qualquer pessoa.

O presente trabalho está estruturado da seguinte forma: No Capítulo 1, é dada uma introdução ao problema; no Capítulo 2, é feita uma explanação em relação aos principais termos e técnicas utilizados no trabalho; no Capítulo 3, são mostrados alguns trabalhos relacionados ao presente trabalho; no Capítulo 4, é explicitada a metodologia utilizada neste trabalho; no Capítulo 5, são mostrados os experimentos realizados com base na metodologia adotada, e seus respectivos resultados; e por fim, no Capítulo 6, são mostradas as conclusões deste trabalho, bem como os trabalhos futuros.

[1] <https://www.docker.com>

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, são apresentados os conceitos fundamentais utilizados neste trabalho. A Seção 2.1, apresenta de uma forma geral os conceitos de Aprendizado de Máquina; na Seção 2.2, são apresentados os conceitos de Redes Neurais; na Seção 2.3, são apresentados os conceitos de Aprendizado Profundo e algumas das principais arquiteturas de Redes Neurais Convolucionais; por fim, na Seção 2.4, são apresentadas as métricas que são utilizadas neste trabalho.

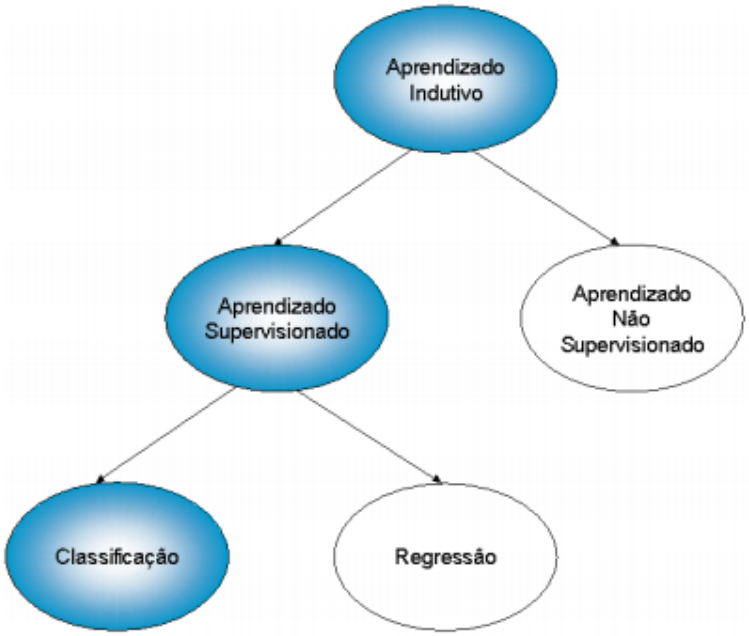
2.1 Aprendizado de Máquina

Aprendizado de Máquina é uma subárea da Inteligencia Artificial que tem como objetivo o desenvolvimento de técnicas computacionais sobre o aprendizado e construção de sistemas capazes de adquirir conhecimento de forma automática (MONARD; BARANAUSKAS, 2003).

O Aprendizado de Máquina possui um tipo bastante genérico, que é o Aprendizado Indutivo onde a aquisição de conhecimento se dá por meio de um conjunto de exemplos (NICOLETTI, 1994). Como pode-se observar na Figura 1, existem dois principais tipos de Aprendizado Indutivo:

- **Aprendizado Supervisionado:** Forma de aprendizado onde é fornecido ao algoritmo de aprendizado, um conjunto de exemplos de treinamento para os quais o rótulo da classe associada é conhecido (MONARD; BARANAUSKAS, 2003). Existem dois principais tipos de problemas que podem ser resolvidos por meio de Aprendizado Supervisionado:
  - **Problemas de Classificação:** Um problema é dito Problema de Classificação quando tem como objetivo determinar a classe a qual um determinado elemento pertence, sendo o número de classes um número discreto (FRIEDMAN *et al.*, 2001).
  - **Problemas de Regressão:** Um problema é dito Problema de Regressão quando tem como objetivo determinar valores quantitativos, tratando-se de valores contínuos (FRIEDMAN *et al.*, 2001).
- **Aprendizado Não Supervisionado:** Forma de aprendizado em que não é fornecido nenhum rótulo ao algoritmo, de forma que o algoritmo deve ser capaz de agrupar os dados de forma que os dados mais semelhantes fiquem mais próximos, formando os *clusters* (REZENDE, 2003; OCHI *et al.*, 2004)

Figura 1 – Hierarquia do Aprendizado de Máquina



Fonte: Monard e Baranauskas (2003).

Dada a natureza do problema ao qual o presente trabalho busca resolver – descobrir qual o tipo de distração do motorista dada uma imagem – é fácil perceber que as técnicas utilizadas são as que aparecem coloridas na Figura 1: Aprendizado Indutivo, Aprendizado Supervisionado e Classificação.

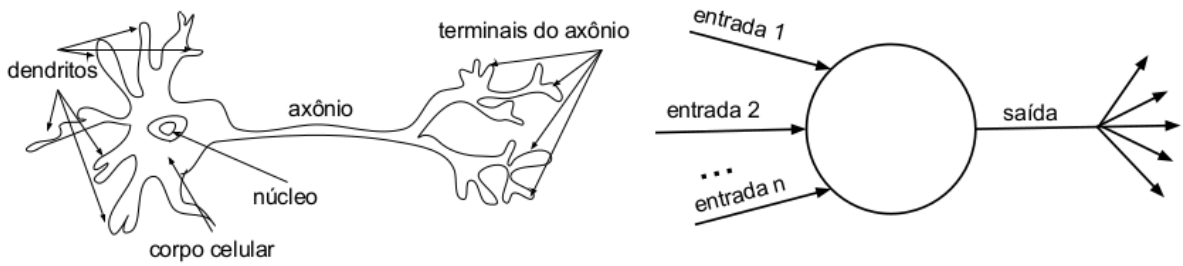
2.2 Redes Neurais

Uma Rede Neural é um modelo preditivo motivado pela forma como o cérebro funciona. Imaginando-se o cérebro como uma coleção de neurônios conectados, cada neurônio examina as saídas dos neurônios que o alimentam, fazem um cálculo e, em seguida, disparam, se o cálculo exceder um determinado limiar, ou não, caso contrário (GRUS, 2015).

Assim como neurônios biológicos possuem dendritos por onde recebem estímulos, um corpo no qual tais sinais são processados, e um axônio responsável por emitir para fora do neurônio um sinal de saída, também os neurônios artificiais possuem um conjunto de canais de entrada, etapas de processamento e uma saída que pode ser ligada a outros neurônios, como pode ser visto na Figura 2 (VASCONCELOS; GONZALEZ, 2017).

Ainda hoje, são utilizados os fundamentos do modelo básico de neurônio artificial, denominado *Perceptron*, proposto em (GRUS, 2015). Partindo de conceitos propostos pela neurociência, ROSENBLATT introduziu a ideia de associar um peso a cada conexão de entrada

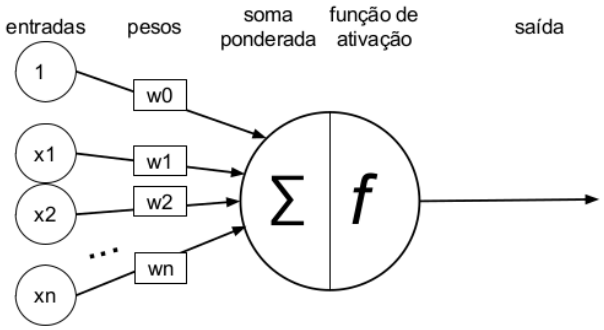
Figura 2 – Neurônio Biológico e Neurônio Artificial



Fonte: Vasconcelos e Gonzalez (2017).

do neurônio artificial, de maneira a ponderar sua influência para uma determinada tarefa, como pode ser visto na Figura 3.

Figura 3 – Perceptron: a combinação linear das entradas x_i ponderadas pelos pesos w_i é transformada pela função de ativação f na saída y emitida pelo neurônio



Fonte: Vasconcelos e Gonzalez (2017).

Com apenas um *perceptron* é possível resolver problemas binários, no qual as possíveis saídas são linearmente separáveis. Portanto, ele é capaz de aprender as funções booleanas de negação (NOT), "e" (AND) e "ou" (OR). Entretanto, utilizando-se apenas um *perceptron* não é possível modelar, por exemplo, a função booleana "ou exclusivo" (XOR) (VASCONCELOS; GONZALEZ, 2017).

Para conseguir resolver problemas mais complexos, faz-se necessária a organização de diversos *perceptrons*. Existem diferentes formas de se organizar um conjunto de *perceptrons* para formar uma Rede Neural, porém este trabalho é focado nas chamadas Redes Neurais Alimentadas para Frente (*Feedforward Neural Networks*) ou Perceptron Múltiplas Camadas (*Multi-layer Perceptron*). De acordo com Vasconcelos e Gonzalez (2017), uma estrutura em camadas é composta por:

- Uma camada inicial, chamada de Camada de Entrada, responsável pela leitura dos dados que serão processados;
- Uma ou mais camadas intermediárias, chamadas de Camadas Ocultas, responsáveis pelo

processamento propriamente dito;

- Uma última camada, chamada Camada de Saída, responsável por emitir o resultado do processamento.

Com Redes Neurais é possível resolver uma gama de problemas variados, desde reconhecimento de manuscritos à detecção facial (GRUS, 2015). Além disso, Redes Neurais não são focadas em apenas um tipo de problema. Com Redes Neurais é possível resolver Problemas de Classificação, Regressão, Clusterização, entre outros (VASCONCELOS; GONZALEZ, 2017).

2.3 Aprendizado Profundo

Atualmente, Aprendizado Profundo tornou-se difícil de se definir, pois sua definição veio mudando aos poucos durante a ultima década. Uma definição comum de aprendizado profundo é a de que Aprendizado Profundo se trata de Aprendizado de Máquina com Redes Neurais com mais de 2 camadas ocultas. Porém, se assim fosse, seria possível dizer que Aprendizado Profundo é usado desde a década de 80. Segundo (PATTERSON; GIBSON, 2017) o conceito o Aprendizado Profundo transcende arquiteturas e, juntamente com o aumento do poder de processamento, tem dado resultados incríveis em relação aos estilos de Redes Neurais anteriores. O autor ainda aponta alguns aspectos importantes sobre Aprendizado Profundo:

- Mais neurônios que as redes anteriores;
- Formas mais complexas de conectar camadas /neurônios em Redes Neurais;
- Aumento gigantesco no poder de computação disponível para treinamento;
- Extração automática de características.

Assim, a definição de (PATTERSON; GIBSON, 2017) – e que é adotada neste trabalho – de Aprendizado Profundo, é que são Redes Neurais com um grande número parâmetros e camadas que pertencem a um dos principais tipos de arquiteturas que são apresentadas nas subseções a seguir.

2.3.1 Redes Neurais Recorrentes

Redes Neurais Recorrentes fazem parte da família de Redes Neurais Alimentadas para Frente. Porém, diferentes das Redes Neurais convencionais, as Redes Neurais Recorrentes possuem a habilidade de enviar informações ao longo do tempo. Dessa forma, esse tipo de Rede Neural modela o aspecto de tempo dos dados criando ciclos na rede – daí o “recorrente”, do nome

(PATTERSON; GIBSON, 2017). Técnicas de Aprendizado de Máquina comuns e demais tipos de Redes Neurais, não modelam a influência do tempo em seus modelos, assumindo que o tempo não é uma informação importante. No entanto, o fator tempo pode ser de extrema importância para alguns conjuntos de dados. Pensando nisso, uma técnica muito utilizada para considerar o tempo em alguns modelos, é dar como entrada para o modelo não apenas um dado, mas uma janela de dados, por exemplo: anterior, atual e próximo. Porém essa técnica possui a limitação de considerar apenas o tempo da janela, podendo deixar de aprender algum padrão temporal de duração maior que o período da janela predefinida. Já com as Redes Neurais Recorrentes não há esse problema, uma vez que não há uma restrição de tamanho de janela. Redes Neurais Recorrentes podem possuir ciclos nas conexões. Isto permite modelar o comportamento temporal e melhorar a taxa de acerto em domínios como: séries temporais, linguagem, áudio e texto (PATTERSON; GIBSON, 2017).

2.3.2 Redes Neurais Recursivas

Assim como as Redes Neurais Recorrentes, as Redes Neurais Recursivas possuem a capacidade de lidar com entradas de tamanhos diferentes. Porém, as Redes Neurais Recursivas conseguem modelar estruturas hierárquicas no conjunto de dados. A desconstrução de cenas, por exemplo, ainda é um problema não trivial. E com a natureza recursiva de problemas de desconstrução, o desafio não é apenas identificar objetos, mas identificar como os objetos estão relacionados na cena. Além de desconstrução de cena, Redes Neurais Recursivas podem ser utilizadas em processamento de linguagem natural, transcrição de áudio-para-texto, entre outros.

2.3.3 Redes Neurais Pré-treinadas não Supervisionadas

Fazem parte desse tipo das Redes Neurais Pré-treinadas não Supervisionadas: as Autoencodificadoras (*Autoencoders*) – redes neurais capazes de aprender representações comprimidas de conjuntos de dados, normalmente utilizadas para redução de dimensionalidade; as Redes de Crenças Profundas (*Deep Belief Networks*, ou DBNs) – redes capazes de aprender probabilisticamente, a reconstruir as entradas; e as Redes Adversariais Generativas (*Generative Adversarial Networks*, ou GANs) – uma arquitetura composta por duas Redes Neurais. Uma responsável por gerar novos dados e outra por descobrir os dados que são gerados. Assim as duas redes aprendem e ao final do treinamento, a rede geradora pode ser capaz de gerar dados muito parecidos com dados reais (PATTERSON; GIBSON, 2017).

2.3.4 Redes Neurais Convolucionais

Redes Neurais Convolucionais, que são o foco desde trabalho, representam um modelo de Aprendizado Profundo baseado na organização do córtex visual dos animais. A analogia feita é de que as células do córtex visual são sensíveis a pequenas sub-regiões do campo de visão. Essas pequenas sub-regiões são divididas em mosaicos de forma a cobrir todo o campo de visão. Essas células são adequadas para explorar as fortes correlações espaciais locais encontradas nos tipos de imagens que nosso cérebro processa, de forma a atuar como filtros locais sobre o espaço de entrada. Assim, as células mais simples são ativadas quando detectam padrões mais simples, como arestas e células mais complexas são ativadas quando detectam padrões mais específicos – geralmente baseados em diversos resultados de várias células simples (PATTERSON; GIBSON, 2017).

Para entender melhor, Redes Perceptron Multicamadas normais recebem as entradas em vetores unidirecionais que transformam os dados e os enviam para uma ou mais camadas ocultas totalmente conectadas, então os resultados são retornados pela camada de saída. O problema dessa abordagem, principalmente tratando com imagens, é que é difícil conseguir escalar. Tendo-se como exemplo um conjunto de dados com imagens de tamanho 32x32 pixels com 3 canais de informação RGB tem-se 3072 pesos por neurônio na primeira camada oculta, onde facilmente há vários neurônios. Além disso, é normal que haja várias camadas ocultas, multiplicando mais ainda esses pesos (PATTERSON; GIBSON, 2017).

Desde 2010, anualmente, acontece uma competição chamada ILSVRC (*Imagenet Large Scale Visual Recognition Challenge*) que tem como objetivo avaliar algoritmos de detecção de objectos e classificação de imagens (RUSSAKOVSKY *et al.*, 2015). A partir de 2012, os algoritmos com melhores resultados têm sido as Redes Neurais Convolucionais, de forma que todos os anos, as arquiteturas de Redes Neurais Convolucionais que atingem os melhores resultados se tornam arquiteturas “famosas”. Essas arquiteturas são comumente utilizadas para a resolução de diversos outros problemas.

Assim, neste trabalho são analisadas as seguintes arquiteturas:

VGG: A arquitetura VGG (SIMONYAN; ZISSERMAN, 2014) foi proposta baseada na arquitetura AlexNet (ZEILER; FERGUS, 2014) – arquitetura com o melhor desempenho no ILSVRC de 2012 e se tornou um marco na utilização de Redes Neurais Convolucionais.

A principal característica abordada em (SIMONYAN; ZISSERMAN, 2014) é a profundidade. É mostrado que pode-se extrair um maior número de características utilizando-se

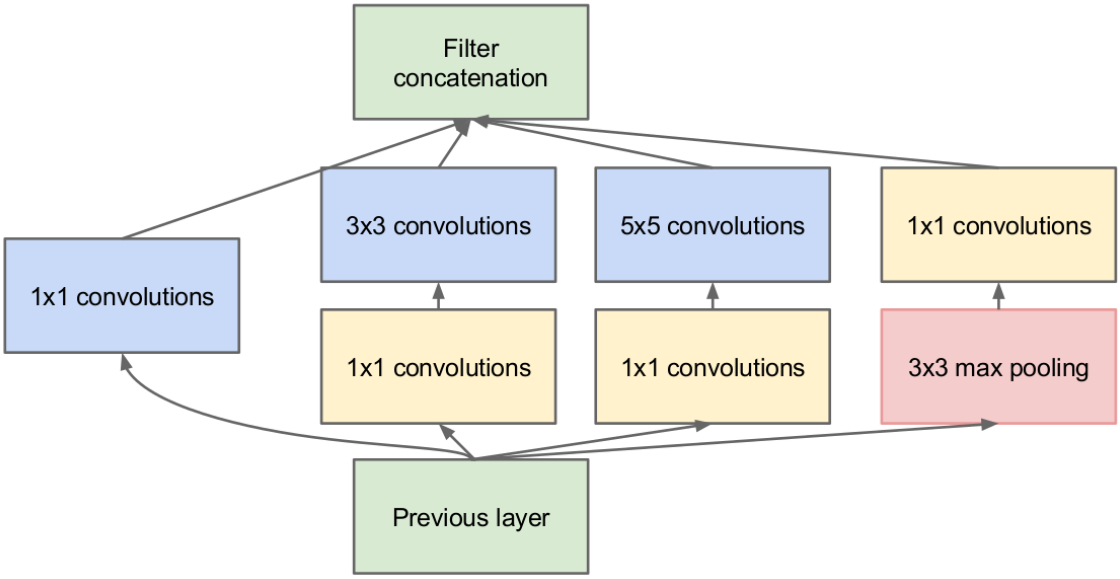
arquiteturas com mais camadas de convoluções e convoluções pequenas (3x3, nesse caso). Com isso, a arquitetura VGG-16 – VGG de 16 camadas com pesos sinápticos – conseguiu ótimos resultados no ILSVRC de 2014, e embora não tenha vencido a competição se tornou uma das arquiteturas mais usadas nos últimos anos tanto a VGG-16, quanto a VGG-19. Essa arquitetura recebe uma imagens RGB de dimensões 224x224.

InceptionV3: A arquitetura GoogLeNet (SZEGEDY *et al.*, 2015) foi a arquitetura de Rede Neural Convolucional com o melhor desempenho no ILSVRC de 2014. A arquitetura recebeu esse nome em homenagem ao trabalho de LECUN *et al.*, considerado o primeiro uso de Redes Neurais Convolucionais modernas. Embora tenha ganho o ILSVRC de 2014, a principal contribuição de (SZEGEDY *et al.*, 2015) não foi a arquitetura da rede como um todo, mas o módulo (bloco de construção) principal que foi desenvolvido, chamado *Inception*.

Na Figura 4, é mostrado o módulo *Inception* que tem como característica principal, evitar um consumo muito alto de recursos. Isso é possível, pois utilizando-se módulos *Inception* "empilhados" tem-se um bom desempenho na retropropagação. Além disso, essa arquitetura possui um número relativamente pequeno, se comparado com as redes VGG, por exemplo. Após o sucesso da GoogLeNet, em (SZEGEDY *et al.*, 2016) são apresentadas algumas melhorias para o módulo *Inception* e para arquitetura da rede, como um todo. Entre as melhorias, está a substituição uma convolução de 5x5 por duas convoluções de 3x3, entre outras. Com as melhorias apresentadas, as novas arquiteturas chamadas InceptionV2 e InceptionV3 passaram a obter melhores resultados que a GoogLeNet e vêm sendo bastante utilizadas. Assim, neste trabalho será utilizada a arquitetura InceptionV3, que tem como entrada imagens RGB de dimensões 299x299.

ResNet: Semelhante à GoogLeNet, a arquitetura ResNet (JAN; LIU, 2015) tem como principal contribuição, não a arquitetura da rede como um todo. Mas sim, seu principal bloco de construção. No caso da ResNet, uma representação do bloco *Residual Learning* é mostrada na Figura 5. Em (JAN; LIU, 2015), os autores se propõem a resolver o problema da degradação do erro e da taxa de acerto com o incremento da profundidade das Redes Neurais Convolucionais. Teoricamente, quanto mais profundas as redes, melhores devem ser os resultados. Porém, foi mostrado em (JAN; LIU, 2015) que em redes "planas", a partir de uma certa profundidade, o aumento do número de camadas tende a produzir piores resultados. Para resolver esse problema, os autores propõem o bloco *Residual Learning*, que, basicamente, adiciona um "atalho" de um ponto i a um ponto $i+2$. Dessa forma, evitando o efeito de degradação com o aumento

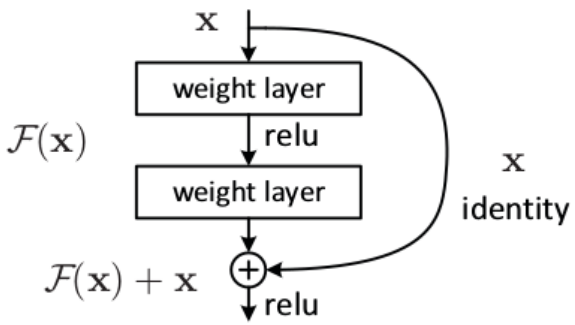
Figura 4 – Módulo Inception



Fonte: Szegedy *et al.* (2015).

de camadas. A arquitetura ResNet-152 obteve os melhores resultados no ILSVRC de 2015. Além da ResNet-152, também são propostas ResNets com várias profundidades, por exemplo: ResNet-34, ResNet-50 e ResNet-101. Neste trabalho são utilizadas ResNet-50 e ResNet-101, por serem arquiteturas consideravelmente mais profundas que VGG-16 e InceptionV3, porém não tão profundas quanto ResNet-152 – que demanda mais recursos computacionais. Essa arquitetura recebe uma imagens RGB de dimensões 224x224.

Figura 5 – *Residual Learning*: um bloco de construção



Fonte: Jan e Liu (2015).

2.4 Métricas

Uma das coisas mais importantes ao avaliar diferentes técnicas de Aprendizado de Máquina, é a escolha de quais métricas avaliar, uma vez que existem métricas mais indicadas

para cada tipo de problema. Assim, nesta seção são apresentadas as métricas utilizadas neste trabalho para avaliar as diversas arquiteturas analisadas.

2.4.1 Logarithmic Loss

O *Logarithmic Loss* (ou Logloss), é uma métrica de desempenho para avaliar as predições de probabilidades de uma determinada entrada pertencer a uma determinada classe. Essa métrica possui valores de 0 a 1, que pode ser vista como a porcentagem de confiabilidade de uma classificação. E como se trata de uma medida de *loss*, quanto menor, melhor. Sendo 0, um valor de erro perfeito (BROWNLEE, 2016).

O *Logarithmic Loss* é denotado pela seguinte fórmula quando o número de classes $M = 2$:

$$loss = -(y\,log(p) + (1-y)log(1-p)) \tag{2.1}$$

e pela seguinte fórmula quando o número de classes $M \geq 2$:

$$loss = -\sum_{c=1}^{M} y_{o,c}\,log(p_{o,c}) \tag{2.2}$$

onde:

- M : Número de possíveis classes;
- o : Uma determinada observação;
- c : Uma determinada classe;
- y : Indicador binário (0 ou 1) de quando uma classe c é a predição correta para uma observação o;
- p : Probabilidade de uma predição de um modelo que uma observação o pertença à classe c.

2.4.2 Taxa de Acerto

A Taxa de Acerto (ou *accuracy*) é porcentagem de predições feitas corretamente em relação a todas as predições feitas. A Taxa de Acerto é a métrica mais utilizada na avaliação de algoritmos para problemas de classificação (BROWNLEE, 2016).

A Taxa de Acerto é denotada pela seguinte fórmula:

$$Taxa\,de\,Acerto = \frac{Quantidade\,de\,Acertos}{Quantidade\,de\,Dados\,Classificados} \tag{2.3}$$

2.4.3 Matriz de Confusão

A Matriz de Confusão é uma representação bastante útil para a Taxa de Acerto de um modelo com duas ou mais classes. É uma tabela que possui uma linha e uma coluna para cada classe. Cada célula possui o número – ou porcentagem – de predições da classe da linha atual que pertencem à classe da coluna atual (BROWNLEE, 2016).

A partir da Matriz de Confusão, é possível retirar algumas informações sobre cada classe, que são utilizadas para calcular diversas métricas (HAN *et al.*, 2011), que são:

- **Verdadeiro Positivo (TP)**: Dados que foram corretamente classificados pelo classificador;
- **Verdadeiro Negativo (TN)**: Dados corretamente classificados como não pertencentes à uma determinada classe;
- **Falso Positivo (FP)**: Dados não pertencentes à uma classe, classificados como pertencentes;
- **Falso Negativo (FN)**: Dados pertencentes á uma classe, classificados como não pertencentes.

Na Tabela 1, é possível ver um exemplo de Matriz de Confusão. Nesse exemplo, pode-se observar que esse modelo hipotético classificou 47 gatos como ratos, 32 cachorros como ratos e 296 ratos como ratos. Dessa forma, esse modelo hipotético poderia ser melhorado para não "confundir" tanto ratos com cachorros.

A partir da Tabela 1 é possível identificar os valores de TP, TN, FP, FN. Por exemplo, levando-se em consideração a classe gato: $TP = 333$, pois 333 gatos foram classificados como gatos; $TN = 310 + 43 + 32 + 296 = 681$, pois são os dados que não são de gatos e não foram classificados como gatos; $FP = 14 + 47 = 61$, pois são dados de cachorros e ratos que foram classificados como de gatos; e $FN = 23 + 18 = 41$, pois são os dados de gatos que foram classificados como cachorros e ratos.

Tabela 1 – Um exemplo de Matriz de Confusão

		classificadas		
		gato	cachorro	rato
classes reais	gato	333	23	18
	cachorro	14	310	43
	rato	47	32	296

Fonte: Elaborado pelo Próprio Autor

2.4.4 Precision

A Taxa de acerto é uma boa métrica para se ter uma visão geral do desempenho do classificador. Porém não permite uma análise classe por classe, o que pode ser um problema caso hajam classes com maior número de dados que outras, de forma que um bom desempenho geral pode ser causado por um bom desempenho em apenas uma classe (BROWNLEE, 2016).

As métricas *precision*, *recall* e *f1-score*, permitem a análise "local" de cada classe, com os valores que podem ser retirados da Matriz de Confusão. A métrica *precision*, para uma classe *A*, é o número de dados corretamente classificados como da classe *A* dividido pelo número de dados classificados como da classe *A* (RASCHKA; MIRJALILI, 2017). A *precision* é denotada pela seguinte fórmula:

$$precision = \frac{TP}{TP+FP} \tag{2.4}$$

2.4.5 Recall

Recall é uma métrica bem semelhante à *precision*, é também conhecida como Taxa de Verdadeiros Positivos, para uma classe *A*, é o número dados corretamente classificados como da classe *A* dividido pelo número dados, realmente, da classe *A* (RASCHKA; MIRJALILI, 2017). O *recall* é denotado pela seguinte fórmula:

$$recall = \frac{TP}{TP+FN} \tag{2.5}$$

2.4.6 F1-score

O *f1-score* é, basicamente, a média harmônica entre *precision* e *recall*. Com *precision* é possível detectar problemas de falsos positivos e com o *recall* é possível detectar problemas de falsos negativos. Assim, como a média harmônica tende a punir valores baixos, com *f1-score* é possível ter uma melhor noção do desempenho do classificador para uma determinada classe (RASCHKA; MIRJALILI, 2017). *F1-score* é denotado pela seguinte fórmula:

$$F1 = 2\frac{precision * recall}{precision + recall} \tag{2.6}$$

2.5 Conclusão

Neste capítulo, foram apresentados os principais conceitos necessários ao entendimento deste trabalho. Com os conceitos apresentados, é possível compreender o capítulo

seguinte, que é o Capítulo 3, onde são apresentados os trabalhos relacionados.

3 TRABALHOS RELACIONADOS

Neste capítulo, é apresentado o trabalho de (KUTILA *et al.*, 2007), que propõe um módulo para detecção de distração de motoristas. É apresentado também, o trabalho de (AGUIAR; GUERRA, 2017) que utiliza Redes Neurais Convolucionais pré-treinadas para classificação de imagens histopatológicas para o auxílio no diagnóstico de câncer de mama.

3.1 DRIVER DISTRACTION DETECTION WITH A CAMERA VISION SYSTEM

Em (KUTILA *et al.*, 2007), é apresentada uma solução para o problema de detecção de distração de motoristas para utilização como um módulo de Avaliação de Atividade de Cabine (*Cockpit Activity Assessment*, ou CAA) que pode ser utilizado em conjunto, por exemplo, com Sistemas Avançados de Assistência ao Motorista (*Advanced Driver Assistance Systems*, ou ADAS).

A solução apresentada para o problema descrito acima é um módulo capaz de detectar a distração visual e cognitiva dos motoristas. A distração visual é detectada a partir dos movimentos da cabeça e do olhar. E a distração cognitiva é detectada a partir de sensores em direção à pista, das informações disponibilizadas pelos sensores do veículo, além de também utilizar as informações de movimentos de cabeça e olhos.

Dessa forma, a solução apresentada possui duas tarefas principais:

- **Detecção de Distração Visual:** A tarefa de detecção de distração visual é feita utilizando um produto disponível comercialmente: o *faceLab* da *Seeing Machines*[1]. Com esse produto é possível coletar as informações de movimentos da cabeça e olhos do motorista. Com isso, é possível determinar se o motorista está olhando para os retrovisores, para frente, etc.
- **Detecção de Distração Cognitiva:** A tarefa de detecção de distração cognitiva é feita utilizando o algoritmo de classificação SVM, tendo como entrada informações de sensores em direção à pista, sensores do próprio veículo e informações de movimentos de cabeça e olhos. Todas essas informações juntas são utilizadas para determinar se há, ou não, distração cognitiva.

Os dados de teste foram coletados com um carro de passageiro da SEAT e um caminhão da Volvo. Os testes e ajustes dos algoritmos desenvolvidos foram feitos remotamente.

[1] <http://www.seeingmachines.com>

Os dados do caminhão foram coletados de 12 motoristas profissionais, de 21 a 59 anos de idade e de 2 à 39 anos de experiência. Já os dados do carro de passageiro foram coletados de 3 motoristas comuns (não profissionais), de 5 a 10 anos de experiência.

Os testes foram feitos em diferentes ambientes como autoestradas, zonas rurais e urbanas. As distrações foram artificialmente introduzidas no conjunto de testes: As distrações cognitivas foram adicionadas pedindo aos motoristas que fizessem operações aritméticas e as distrações visuais foram introduzidas pedindo aos motoristas para ler sequencias de números de figuras coladas no velocímetro, rádio, retrovisores, etc.

Os resultados mostram que o módulo possui uma taxa de acerto em torno de 84% para detecção de que o motorista estava atento. O que ajuda a responder à pergunta de pesquisa do trabalho que é: "A distração visual e cognitiva pode ser detectada com o auxilio de visão computacional em junção com outros dados?". Nesse caso, há fortes evidências de que sim.

Assim como em (KUTILA *et al.*, 2007), este trabalho tem como objetivo detectar distração dos motoristas utilizando aprendizado de máquina. Por outro lado, em (KUTILA *et al.*, 2007) é utilizado um hardware de terceiros que faz boa parte da tarefa de detecção de distração visual e cognitiva. Outro ponto é que neste trabalho não é feita uma diferenciação entre distração visual e cognitiva, em busca de descobrir se um modelo com apenas detecção visual de distração é capaz de detectar distrações em geral. Além disso, as técnicas utilizadas são diferentes, neste trabalho é feito o uso de Aprendizado Profundo, já em (KUTILA *et al.*, 2007) é utilizado SVM.

3.2 TRANSFERÊNCIA DE CONHECIMENTO UTILIZANDO APRENDIZADO PROFUNDO PARA CLASSIFICAÇÃO DE IMAGENS HISTOPATOLÓGICAS

Em (AGUIAR; GUERRA, 2017), é atacado o problema da classificação de imagens histopatológicas para utilização como sistemas de auxílio ao diagnóstico médico (Computer Aided Diagnosis, ou CAD). Em (AGUIAR; GUERRA, 2017), o principal foco da classificação de imagens histopatológicas é na detecção de câncer de mama, o tipo mais comum de câncer entre as mulheres no mundo. No Brasil, por exemplo, corresponde a cerca de 28% dos casos de câncer entre mulheres.

Para classificação de imagens, o autor cita as redes de Aprendizado Profundo como o estado da arte para classificação de imagens. Além disso, o autor cita outros autores que tentaram resolver esse problema utilizando Aprendizado Profundo treinando as Redes Neurais do zero. Dessa forma, o autor buscou resolver esse problema de uma outra forma: utilizando

Redes Neurais Convolucionais pre treinadas (ou Transferência de Conhecimento), para extração de características.

As características extraídas das Redes Neuras de Aprendizado Profundo foram utilizadas para o treinamento de alguns classificadores: *1-Nearest Neighbour* (1-NN), *Quadratic Linear Analysis* (QDA), *Support Vector Machine* (SVM) e *Random Forest of decision trees* (RF). Além disso, as arquiteturas de Redes Neurais Convolucionais utilizadas, foram: VGG-16, VGG-19 e InceptionV3.

O conjunto de dados utilizado em (AGUIAR; GUERRA, 2017) foi o *BreaKhis*[2], um conjunto de dados construído por (SPANHOL *et al.*, 2016) com apoio do Laboratório P&D[3]. O conjunto de dados possui imagens de biópsias feitas a partir da extração de nódulos de 82 pacientes diferentes, de forma que as imagens geradas a partir das análises histopatológicas são classificadas como tumores malignos ou benignos. Além disso, as imagens possuem diferentes magnitudes (ampliações feitas atraves da análise no microscópio), que são de: 40x, 100x, 200x e 400x.

O trabalho possui 3 principais etapas:

- **Pré-Processamento da Base de Dados:** Nessa etapa, o autor faz o redimensionamento das imagens, que possuem as dimensões 700x460 pixels, para que possam ser dadas como entradas para as Redes Neurais.
- **Extração de Características:** Após o redimensionamento das imagens de acordo com o esperado como entrada em cada arquitetura, utilizando a biblioteca Keras, foram carregadas as arquiteturas VGG-16, VGG-19 e InceptionV3, todas com os pesos sinápticos treinados no conjunto de dados *ImageNet*. Para cada magnitude foram gerados os vetores de características com base no modelo interno de extração de características de cada arquitetura.
- **Treino e Teste:** Após a extração de características, os vetores de características gerados, foram dados como entrada e teste para os classificadores citados acima. Os dados foram divididos em 70% para treino e 30% para teste.

Os resultados do trabalho de (AGUIAR; GUERRA, 2017) mostram que o classificador SVM obteve melhores resultados na classificação em todas as arquiteturas. Além disso, a arquitetura que proporcionou melhores resultados foi a InceptionV3 - com a ressalva de que InceptionV3 não foi testada com outros classificadores, mas apenas SVM.

[2] https://omictools.com/breast-cancer-histopathological-database-tool
[3] http://www.prevencaoediagnose.com.br/

Semelhante à (AGUIAR; GUERRA, 2017), o presente trabalho se utiliza de Redes Neurais pré treinadas na base de dados *ImageNet* para classificação de imagens. Inclusive, as arquiteturas utilizadas em (AGUIAR; GUERRA, 2017), também são utilizadas neste trabalho. Por outro lado, neste trabalho a tarefa de classificação é feita pela própria Rede Neural, alterando-se as ultimas camadas da rede, que são responsáveis pela classificação, removendo o mapeamento original de 1000 classes e adicionando um mapeamento para a quantidade de classes utilizadas nesse trabalho.

3.3 Conclusão

Neste capítulo, foram apresentados os trabalhos de (KUTILA *et al.*, 2007) e (AGUIAR; GUERRA, 2017). Foram explicitadas suas características, suas semelhanças e diferenças com o trabalho atual. Na Tabela 2 é feito um resumo das principais características dos trabalhos relacionados e deste trabalho. No Capítulo 4, a seguir, é mostrada a metodologia adotada para o desenvolvimento deste trabalho.

Tabela 2 – Comparativo das características de técnicas utilizadas, tipo de problema e domínio de problema abordados pelas trabalhos

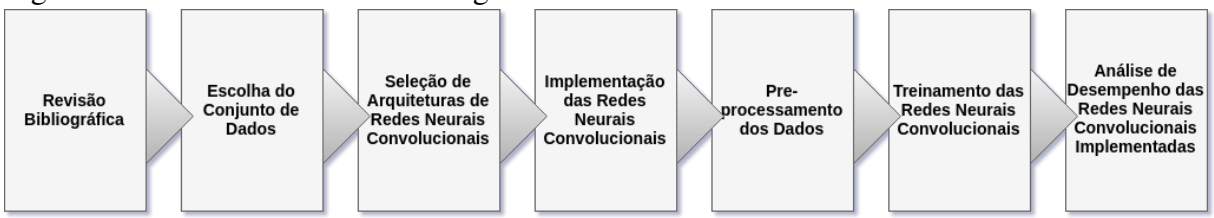
Trabalho	Técnicas Utilizadas	Tipo de Problema	Domínio do Problema
Kutila *et al.* (2007)	SVM	Classificação	Detecção de distração de motoristas baseado em imagens e informações de sensores
Aguiar e Guerra (2017)	VGG-16, VGG-19 e InceptionV3	Classificação	Detecção de câncer de mama baseado em imagens histopatológicas
Este trabalho	VGG-16, InceptionV3, ResNet-50 e ResNet-101	Classificação	Detecção de distração de motoristas baseado em imagens

Fonte: Elaborado pelo próprio autor.

4 METODOLOGIA

O desenvolvimento deste trabalho ocorre em 7 etapas, que estão dispostas nas seções a seguir, sendo uma seção para cada etapa: na Seção 4.1, é feita uma revisão bibliográfica para descobrir quais técnicas vem sendo mais utilizadas na literatura, sobre Classificação de imagens; na Seção 4.2, é feita a escolha do conjunto de dados utilizado para treinar os modelos de Aprendizado de Máquina; na Seção 4.3, é feita a seleção das Arquiteturas de Redes Neurais Convolucionais utilizadas neste trabalho; na Seção 4.4, é feita a implementação das Redes Neurais Convolucionais, de acordo com as arquiteturas selecionadas na Seção 4.3; na Seção 4.5, é feito o pre-processamento dos dados; na Seção 4.6, são feitos os treinamentos das redes neurais implementadas na Seção 4.4; por fim, na Seção 4.7, são feitas as análises de desempenho das redes neurais treinadas na Seção 4.6. A seguir, na Figura 6, é mostrada uma visão geral da metodologia adotada.

Figura 6 – Visão Geral da Metodologia



Fonte: Elaborado pelo Próprio Autor

4.1 Revisão Bibliográfica

A primeira etapa para a realização deste trabalho é a realização de uma revisão bibliográfica para descobrir quais as técnicas mais utilizadas para resolução de problemas semelhantes aos que este trabalho visa resolver, que é a classificação de imagens.

4.2 Escolha do Conjunto de Dados

Dado que o objetivo do presente trabalho é detectar distração de motoristas a partir de imagens, faz-se necessário um conjunto de dados com imagens de motoristas distraídos e não distraídos, de diversas formas. E uma vez que a identificação da distração do motorista pode ser usada para dar *feedbacks* ao motorista, é importante possuir um conjunto de dados com diversas classes e não apenas “distraído” e “não distraído”.

4.3 Seleção de Arquiteturas de Redes Neurais Convolucionais

Detectar se um motorista está distraído e como está distraído baseado em uma imagem é, em essência, um problema de classificação. Dessa forma, faz-se necessária uma etapa para a seleção das técnicas que serão utilizadas para essa tarefa de classificação.

Como discutido anteriormente na Seção 2.3, uma das técnicas mais utilizadas na literatura para classificação de imagens é Aprendizado Profundo, mais especificamente as Redes Neurais Convolucionais. Também é possível utilizar algumas arquiteturas de Redes Neurais, aproveitando treinamentos anteriores em outros conjuntos de dados e fazer alguns ajustes para aproveitar o aprendizado já adquirido para um outro problema. Dessa forma, esta etapa se resume a selecionar as arquiteturas de Redes Neurais Convolucionais que são implementadas e testadas.

4.4 Implementação das Redes Neurais Convolucionais

Uma vez que as arquiteturas de Redes Neurais Convolucionais são selecionadas, na etapa anterior, nesta etapa as arquiteturas de Redes Neurais Convolucionais são implementadas. As implementações são feitas utilizando a linguagem de programação *Python*[1], e a biblioteca de aprendizado profundo *Keras*[2] que permite a implementação em alto nível, de Redes Neurais. Além disso a biblioteca *Keras* permite facilmente importar pesos sinápticos de Redes Neurais treinadas previamente, uma vez que a arquitetura seja a mesma.

4.5 Pre-processamento dos Dados

Com as arquiteturas de Redes Neurais Convolucionais selecionadas, torna-se necessário fazer o pre-processamento dos dados que serão utilizados, que se tratando de Redes Neurais Convolucionais, torna-se basicamente ler as imagens e redimensioná-las para o tamanho esperado pela rede, uma vez que vários pré-processamentos são feitos pela própria Rede Neural Convolucional.

4.6 Treinamento das Redes Neurais Convolucionais

Após implementadas as Redes Neurais Convolucionais e com as imagens já redimensionadas, nessa etapa são feitos os treinamentos das redes. O processo de treinamento é feito

[1] https://www.python.org
[2] https://keras.io

dividindo o conjunto de dados em 60% para treinamento, 20% para validação e 20% para testes.

Além disso, como as Redes Neurais são utilizadas com pesos sinápticos pré-treinados, esta etapa de treinamento tem como objetivo apenas ajustar o conhecimento obtido em outros problemas para resolver o problema que este trabalho se propõe a resolver, então o processo de treinamento não precisa de um número grande de iterações.

4.7 Análise de Desempenho das Redes Neurais Convolucionais Implementadas

Após o treinamento, são realizadas as classificações das imagens do conjunto de teste para a coleta das métricas apresentadas na Seção 2.4 para então avaliar o desempenho de cada rede. A partir dessa análise de desempenho, é possível determinar qual a arquitetura de Rede Neural Convolucional que melhor classifica distração de motoristas.

5 EXPERIMENTOS E RESULTADOS

Nesse capítulo, são apresentados os experimentos realizados além dos resultados obtidos.

5.1 Revisão Bibliográfica

A primeira etapa deste trabalho se dá a partir de uma revisão bibliográfica para descobrir quais as principais técnicas utilizadas na literatura recente para resolver problemas de classificação de imagens. Onde foi descoberto que a principal técnica utilizada é Aprendizado Profundo, mais especificamente com a utilização de Redes Neurais Convolucionais. Além disso foi descoberto que existem diversas arquiteturas de Redes Neurais Convolucionais, comumente utilizadas e que é possível utilizar o aprendizado adquirido na resolução de outros problemas para ajudar na resolução de outros problemas.

A seguir, estão os trabalhos que forneceram a base teórica necessária necessária para o desenvolvimento deste trabalho: (GRUS, 2015; PATTERSON; GIBSON, 2017; CHOLLET, 2017; HAYKIN, 2009; MONARD; BARANAUSKAS, 2003)

5.2 Escolha do Conjunto de Dados

O Kaggle[1] é uma plataforma muito completa de Ciência de Dados e Aprendizado de Máquina. Nela estão presentes diversos conjuntos de dados disponibilizados pela comunidade, que é bastante ativa, além de diversas competições. Algumas dessas competições são publicadas pela própria comunidade com o objetivo de fomentar discussões. Outras são publicadas por empresas que necessitam resolver algum problema, inclusive dando premiações para as melhores soluções.

O conjunto de dados utilizado neste trabalho foi disponibilizado em um desafio da plataforma Kaggle financiado pela *State Farm*, uma empresa de seguros. O desafio consiste, basicamente em classificar, dada uma imagem, se o motorista está dirigindo atentamente ou distraído de alguma forma. Para isso, é disponibilizado um conjunto de dados com imagens 2D de motoristas dirigindo. O conjunto de dados está dividido da seguinte forma:

a) **Imagens de treino:** 22.424 imagens classificadas

b) **Imagens de teste:** 79.726 imagens não classificadas

[1] <https://www.kaggle.com>

O conjunto de dados possui 10 classes, que são:

- c0: Condução normal
- c1: Digitando com a mão direita
- c2: Falando ao telefone com a mão direita
- c3: Digitando com a mão esquerda
- c4: Falando ao telefone com a mão esquerda
- c5: Operando o rádio
- c6: Bebendo
- c7: Alcançando atrás
- c8: Cabelo e Maquiagem
- c9: Conversando com passageiro

Como o conjunto de dados foi disponibilizado para um desafio, faz parte do desafio classificar todas as imagens do conjunto de teste com o intuito de descobrir se o modelo utilizado na classificação consegue generalizar de forma satisfatória as classificações feitas baseadas no conjunto de treino. Por conta disso, o conjunto de teste é significativamente maior.

Como mencionado anteriormente, classificar todas as imagens do conjunto de teste, faz parte do desafio. Então, deve-se gerar um arquivo descrevendo, para cada imagem, as probabilidades da imagem pertencer à uma determinada classe. Esse arquivo deve ser submetido na página do desafio para, só então, descobrir o erro obtido. E como o desafio busca descobrir apenas o menor erro, o erro é a única métrica que pode ser extraída da submissão das classificações do conjunto de teste.

Então, para que fosse possível avaliar outras métricas, o conjunto de testes utilizado neste trabalho é uma amostra do conjunto de treino. Além disso, do conjunto de treino também é retirada uma amostra para utilização como conjunto de validação. Dessa forma, do conjunto de treino disponibilizado, serão utilizadas 60% (13.456) das imagens para treino, 20% (4.484) das imagens para validação e 20% (4.484) para teste.

5.3 Seleção de Arquiteturas de Redes Neurais Convolucionais

Na literatura existem diversas arquiteturas de Redes Neurais Convolucionais que são comumente utilizadas. Geralmente, arquiteturas que atingem bons resultados no ILSVRC, como visto na Seção 2.3.

Dada as arquiteturas disponíveis, para este trabalho foram buscadas arquiteturas que

obtiveram bons resultados e que são utilizadas com frequência aproveitando-se os conhecimentos adquiridos anteriormente. Além disso, outro fator considerado com grande importância é a demanda recursos computacionais necessários para a execução dos treinamentos e testes. Baseado em experimentos prévios, foi percebido que demanda de recursos computacionais necessários tende a ser proporcional à quantidade de camadas.

Dessa forma, as arquiteturas selecionadas foram as arquiteturas descritas na Seção 2.3, que são: VGG-16, InceptionV3, ResNet-50 e ResNet-101. Todas são frequentemente citadas na literatura, todas possuem pesos sinápticos pré-treinados disponíveis para download[2] e todas possuem um número relativamente razoável de camadas, o que permite que os treinamentos e testes possam ser executados em um tempo hábil.

5.4 Implementação das Redes Neurais Convolucionais

Após selecionadas, as arquiteturas a serem utilizadas, são feitas as implementações. Todas as arquiteturas foram implementadas em Python utilizando-se Keras tendo o Tensorflow [3] – um *framework* de Aprendizado de Máquina *open source* muito utilizado para criação de Redes Neurais – como *framework* de aprendizado de máquina. Todos os códigos foram feitos na ferramenta *Jupyter*[4], uma ferramenta que permite criar e compartilhar códigos com execução iterativa, que é usado via *browser*. Os arquivos criados usando o *Jupyter*, são chamados de *Jupyter Notebooks*.

Como todas as arquiteturas selecionadas foram projetadas para uso do conjunto de dados do *Imagenet*, todas possuem saídas do tamanho da quantidade de classes do *Imagenet*, que são 1000 classes. Porém, o conjunto de dados utilizado neste trabalho possui apenas 10 classes. Logo, um dos primeiros problemas resolvidos foi adaptar os modelos, já existentes e treinados, para classificar 10 classes ao invés de 1000.

Uma boa solução é remover a última camada do modelo original, que é responsável por fazer o mapeamento para as classes e possui 1000 neurônios, e adicionar uma nova camada com 10 neurônios, fazendo assim, o mapeamento para as classes do conjunto de dados utilizado [5]. Infelizmente não foi possível fazer essa adaptação nos modelos de todas arquiteturas, uma vez que apenas uma das quatro arquiteturas selecionadas foi originalmente implementada utilizando

[2] <https://github.com/fchollet/deep-learning-models/releases>
[3] <https://www.tensorflow.org>
[4] <http://jupyter.org>
[5] <https://flyyufelix.github.io/2016/10/03/fine-tuning-in-keras-part1.html>

um modelo do tipo *Sequential* [6] do Keras, que permite, entre outras operações, remover camadas. Essa arquitetura foi a VGG-16.

Nas demais arquiteturas, a solução encontrada foi adicionar uma nova camada com 10 neurônios ao fim da última camada do modelo original, conseguindo dessa forma, o mapeamento para as classes do conjunto de dados utilizado. Essa solução funciona tanto quanto a solução descrita acima. Por outro lado, essa solução aumenta a quantidade de sinapses treináveis, tornando o treinamento mais custoso. Esta solução foi aplicada para as arquiteturas InceptionV3, ResNet-50 e ResNet-101.

Nas Redes Neurais Convolucionais, as camadas de convolução, basicamente, detectam padrões. De forma que camadas mais "rasas" tendem à detectar padrões mais simples – como cantos e bordas – e camadas mais "profundas" tendem a detectar padrões mais complexos (PATTERSON; GIBSON, 2017). Dessa forma, quando se utiliza um modelo pre-treinado, faz sentido que as primeiras camadas não sejam treinadas novamente, uma vez que padrões simples tendem a terem sido suficientemente treinados anteriormente. Essa operação de determinar que determinadas camadas podem ou não serem treinadas, também é uma exclusividade do tipo de modelo *Sequential* do Keras. Logo, essa técnica pôde ser utilizada apenas na arquitetura VGG-16, onde foi definido que as 10 primeiras camadas não são treináveis.

Para o desenvolvimento de cada arquitetura, foi criado um *Jupyter Notebook* contendo um arcabouço de código bem semelhante. O que difere um *Jupyter Notebook* de outro são 2 métodos que são implementados de acordo com cada arquitetura. Um método, é o que é responsável pela leitura de uma imagem e outro que é responsável por criar o modelo, baseado na arquitetura.

5.5 Pré-Processamento dos Dados

Como discutido anteriormente, as convoluções das Redes Neurais Convolucionais funcionam, também, como uma espécie de pre-processamento, onde são aplicados diversos tipos de filtros. Assim, além do pré-processamento "natural" feito dentro das próprias redes, é feita uma pequena etapa de pré-processamento antes da imagem ser dada como entrada à rede.

Como cada arquitetura de Rede Neural Convolucional é diferente, cada uma pode esperar que a imagem de entrada esteja de uma determinada forma. Assim, nesta etapa são feita duas operações após a imagem ser lida na memória: redimensionamento da imagem

[6] <https://keras.io/getting-started/sequential-model-guide/>

para as dimensões esperadas pela rede; e uma normalização nos valores dos pixeis. Esse pre-processamento é feito no método que é responsável pela leitura de uma imagem, descrito na seção anterior.

5.6 Treinamento das Redes Neurais Convolucionais

Como foi dito anteriormente, para cada arquitetura, foi criado um *script*. Todos possuem 4 etapas principais: 1 - carregar as imagens do disco, na memória; 2 - carregar o modelo da arquitetura com pesos sinápticos pré-treinados; 3 - realizar o treinamento; 4 - executar a avaliação das predições feitas no conjunto de testes.

A etapa de treinamento é executada em 10 iterações de uma época. Assim, o treinamento total consiste de 10 épocas – como se trata de redes pre-treinadas, não se faz necessário um grande número de épocas. Dessa forma, em cada iteração ao fim to treinamento de 1 época, é possível avaliar o modelo. Assim, após o treinamento de uma época é feita a predição dos dados do conjunto de teste e em seguida a avaliação das predições feitas. Dessa forma, é possível avaliar a evolução do modelo de acordo com o avanço dos treinamentos. Após o fim do treinamento das 10 épocas, cada modelo é salvo para que possa ser reutilizado futuramente.

O treinamento é feito utilizando como parâmetros, o conjunto de treinamento, o *batch size*, número de épocas e o conjunto de validação. A seguir, é descrito cada parâmetro, de acordo com a documentação do Keras[7]:

- **Conjunto de treino:** é composto pelo conjunto de dados de treino, incluindo as imagens e as classes de cada imagem;
- ***Batch size:*** número de exemplos de treino executados para que os pesos sinápticos sejam utilizados;
- **Número de épocas:** número de épocas que serão executadas no treinamento. Uma época é uma iteração passando por todos os dados do conjunto de treino;
- **Conjunto de Validação:** é composto pelo conjunto de dados de validação, incluindo as imagens e as classes de cada imagem.

Os treinamentos e avaliações foram executados em máquinas virtuais *n1-standard-8* na Google Cloud Plataform [8]. Cada máquina *n1-standard-8* possui 8 CPUs virtuais, 30GB de memória RAM e 45GB de disco SSD – tamanho personalizado pelo autor. As máquinas

[7] <https://keras.io/models/model/>
[8] <https://cloud.google.com>

utilizadas não possuem GPU. O que tornou as operações mais demoradas.

Todo o ambiente foi provisionado usando Docker. Foi criada uma imagem docker[9] contendo diversas ferramentas e bibliotecas necessárias para o uso de Aprendizado Profundo em geral, tais como: *Tensorflow*, *Keras*, *Jupyter*, *Scikit-learn*, entre outras. Essa imagem serviu como base para execução dos experimentos. Dessa forma, pode-se garantir que todas as arquiteturas são experimentadas no mesmo ambiente, embora em máquinas diferentes. Além de agilizar a replicação dos experimentos.

5.7 Resultados da Análise de Desempenho das Redes Neurais Convolucionais Implementadas

Como descrito na Seção 4.7, uma análise de desempenho se faz necessária, para permitir avaliar qual arquitetura consegue melhores resultados para o problema da classificação de distração de motoristas usando o conjunto de dados descrito na Seção 5.2.

Para avaliar quão bons foram os resultados obtidos pelas Redes Neurais, foram avaliadas as métricas descritas na Seção 2.4. Além disso, a Seção 5.6 descreveu que o processo de treinamento ocorreu em 10 iterações de treinamento – 10 épocas – intercaladas com avaliação do modelo e coleta das métricas.

Das métricas avaliadas, o *loss* é a métrica de maior relevância, uma vez que durante o treinamento, a Rede Neural busca sempre diminuir o *loss*. Assim, a rede que alcançar o menor valor de *loss*, será considerada a rede com o melhor desempenho. Como o *loss* não tem uma faixa de valores predefinida, é difícil avaliar o quão bons são os resultados apenas com ele. Para auxiliar nessa análise, a Taxa de Acerto é utilizada para obtenção de uma visão mais ampla do desempenho. Além disso, para fazer uma análise mais local e dar uma melhor noção do desempenho classe-a-classe, são utilizadas a Matriz de Confusão, *precision*, *recall* e *f1-score*.

A seguir, na Tabela 3, é possível ver os valores de *loss* obtidos em cada época por cada arquitetura, onde é possível verificar que as arquiteturas com melhores valores de *loss* foram VGG-16 e InceptionV3. O menor valor de *loss* de cada arquitetura está destacado em negrito. Além disso, o menor resultado geral é destacado com um asterisco.

A Figura 7 mostra uma visão geral dos desempenhos de cada arquitetura por época, em relação ao *loss*. É possível perceber que a VGG-16 e a InceptionV3 obtiveram valores de *loss* consideravelmente menores que ResNet-50 e ResNet-101. Outro detalhe importante, é

[9] <https://github.com/eduardoslopes/deep_learning_docker>

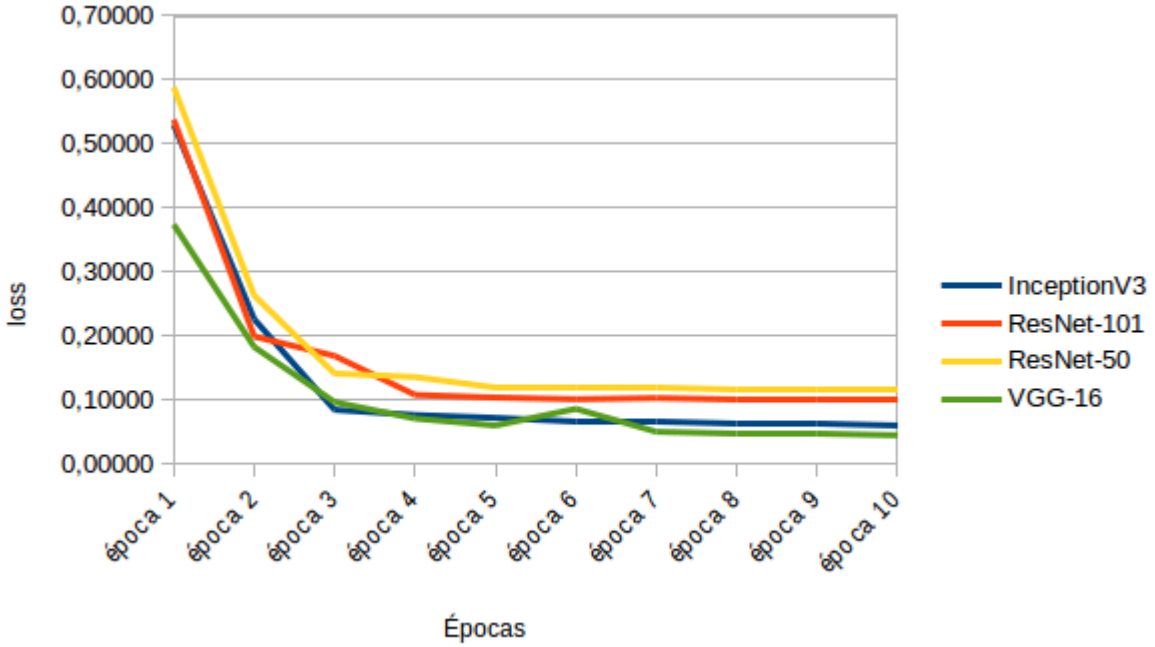
Tabela 3 – Valores de *loss* por época (epc) por arquitetura

Arquitetura	**epc 1**	**epc 2**	**epc 3**	**epc 4**	**epc 5**	**epc 6**	**epc 7**	**epc 8**	**epc 9**	**epc 10**
VGG-16	0,3744	0,1833	0,0972	0,0715	0,0607	0,0866	0,0509	0,0487	0,0466	**0,0455***
InceptionV3	0,5302	0,2267	0,0851	0,0770	0,0726	0,0655	0,0659	0,0640	0,0637	**0,0607**
ResNet-101	0,5382	0,1997	0,1693	0,1084	0,1043	0,1017	0,1035	0,1009	0,1008	**0,1002**
ResNet-50	0,5888	0,2635	0,1419	0,1363	0,1200	0,1192	0,1184	0,1178	0,1168	**0,1159**

Fonte: Elaborado pelo próprio autor.

que com as arquiteturas ResNet-50 e ResNet-101, a partir da época 5, não houveram reduções significativas de *loss*, tendo-se retas perfeitas. De forma semelhante, com as arquiteturas VGG-16 e InceptionV3, a partir da época 7, também não houveram reduções tão significativas nos valores de *loss*, embora seja possível verificar que houve uma pequena redução, de acordo com a Tabela 3.

Figura 7 – Visão Geral do *loss* de cada arquitetura por época



Fonte: Elaborado pelo Próprio Autor

A seguir, na Tabela 4, é possível ver os valores de Taxa de Acerto obtidos em cada época por cada arquitetura, onde é possível verificar que as arquiteturas com maiores valores de Taxa de Acerto também foram VGG-16 e InceptionV3, com taxas de acerto 98,48% e 98,31%, respectivamente, na época 10. Semelhante à Tabela 3, o maior valor de Taxa de Acerto de cada arquitetura está destacado em negrito. Além disso, o maior resultado geral, também, é destacado com um asterisco.

Na figura 8 é possível analisar a evolução das Taxas de Acerto de cada arquitetura por época. Nesta figura, é possível perceber uma relação entre o *loss* e a Taxa de Acerto. A

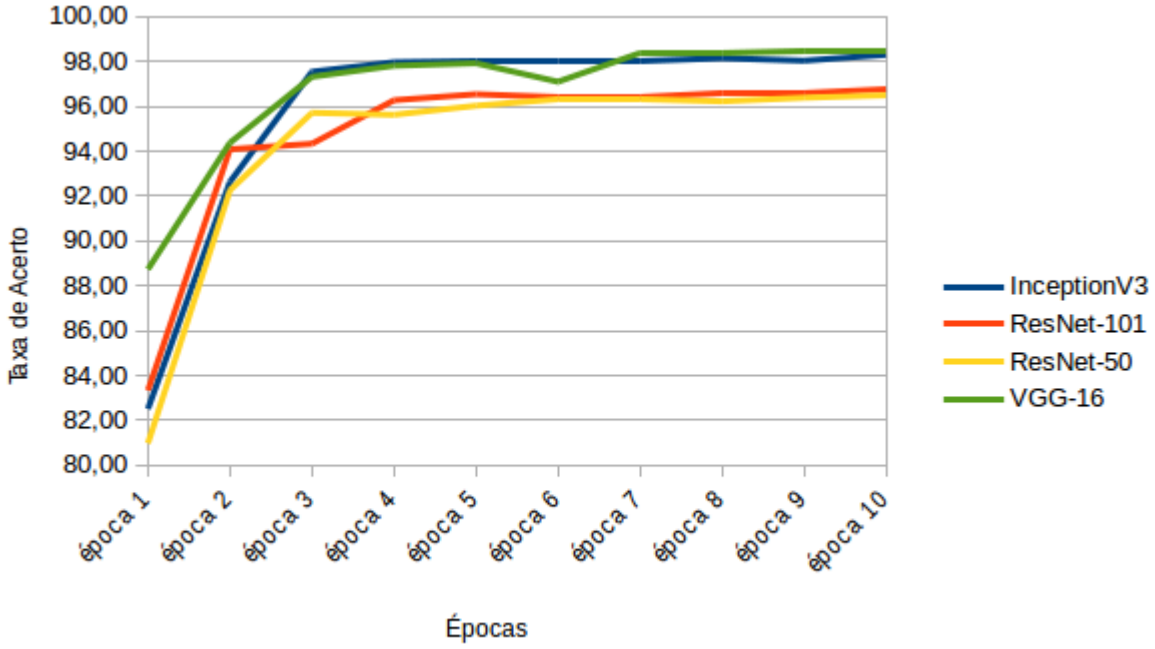
Tabela 4 – Valores de Taxa de Acerto em porcentagem por época (epc) por arquitetura

Arquitetura	epc 1	epc 2	epc 3	epc 4	epc 5	epc 6	epc 7	epc 8	epc 9	epc 10
VGG-16	88,74	94,38	97,32	97,81	97,93	97,10	98,37	98,37	98,44	**98,48***
InceptionV3	82,52	92,64	97,55	97,97	98,02	98,02	98,06	98,15	98,04	**98,31**
ResNet-101	83,34	94,09	94,34	96,28	96,54	96,45	96,41	96,61	96,59	**96,77**
ResNet-50	80,98	92,26	95,72	95,63	96,03	96,34	96,36	96,23	96,39	**96,50**

Fonte: Elaborado pelo próprio autor.

partir das figuras 6 e 7 é possível perceber, na arquitetura VGG-16, na época 6, o valor de *loss* teve um pequeno aumento e a Taxa de Acerto teve uma pequena diminuição. Embora não haja uma relação direta entre essas duas métricas, é intuitivo dizer que há uma relação, afinal, em geral, quando se erra menos, ocorrem mais acertos. Reforçando essa intuição, as arquiteturas que resultaram os menores valores de *loss*, resultaram também em maiores valores de Taxa de Acerto.

Figura 8 – Visão Geral dos valores das Taxas de Acerto de cada arquitetura por época

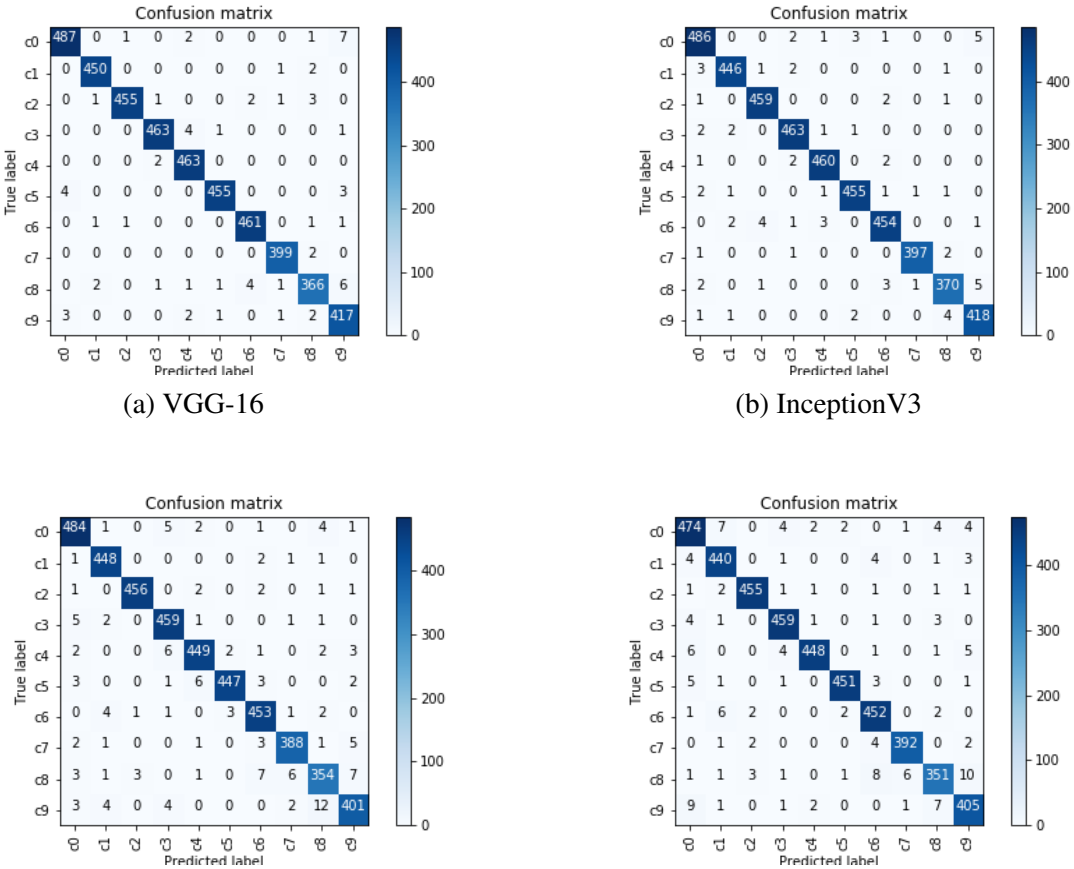


Fonte: Elaborado pelo Próprio Autor

Além das Taxas de Acerto e *loss*, também foram coletadas as matrizes de confusão em cada época. As matrizes de confusão geradas neste trabalho têm as colunas representando as classes reais e as linhas representando as classes que foram classificadas por cada Rede Neural, onde cada célula possui um tom de azul, e quanto maior o valor da célula, mais escuro o azul, possibilitando uma análise visual mais rápida.

Por motivos de organização, são mostradas nesta seção apenas as Matrizes de Confusão das Nedes Neurais na época 10, na Figura 9, que foi a época com os melhores

Figura 9 – Matriz de Confusão de cada arquitetura na época 10



(a) VGG-16

(b) InceptionV3

(c) ResNet-101

(d) ResNet-50

Fonte: Elaborado pelo Próprio Autor

resultados, tanto de *loss*, quanto de Taxa de Acerto.

Pela Figura 9 é possível perceber, em todas as arquiteturas, que as diagonais principais – que representam as classificações corretas – possuem valores mais altos. É possível perceber também, na Figura 9c, que a arquitetura ResNet-101 classificou 12 imagens *c9* (Conversando com passageiro) como *c8* (Cabelo e Maquiagem), que foi o caso de mais classificações erradas na época 10.

Pela Figura 10, é possível ver, para cada arquitetura, para cada classe, os valores das métricas *precision*, *recall* e *f1-score*, além do *support*, que é apenas a quantidade de dados total da classe. A visualização dessas 3 métricas juntas, é comumente chamada de *Classification Report* – nome usado pela biblioteca *Scikit-learn*, uma das mais utilizadas para Aprendizado de Máquina.

Com a Matriz de Confusão é possível ver quantos casos foram classificados corretamente, quantos não foram e qual a classificação feita em caso de erro. Porém é difícil de comparar, e avaliar apenas pela Matriz de Confusão, uma vez que as informações não são

Figura 10 – *Classification Report* de cada arquitetura na época 10

	precision	recall	f1-score	support
c0	0.99	0.98	0.98	498
c1	0.99	0.99	0.99	453
c2	1.00	0.98	0.99	463
c3	0.99	0.99	0.99	469
c4	0.98	1.00	0.99	465
c5	0.99	0.98	0.99	462
c6	0.99	0.99	0.99	465
c7	0.99	1.00	0.99	401
c8	0.97	0.96	0.96	382
c9	0.96	0.98	0.97	426
avg / total	0.98	0.98	0.98	4484

(a) VGG-16

	precision	recall	f1-score	support
c0	0.97	0.98	0.97	498
c1	0.99	0.98	0.99	453
c2	0.99	0.99	0.99	463
c3	0.98	0.99	0.99	469
c4	0.99	0.99	0.99	465
c5	0.99	0.98	0.99	462
c6	0.98	0.98	0.98	465
c7	0.99	0.99	0.99	401
c8	0.98	0.97	0.97	382
c9	0.97	0.98	0.98	426
avg / total	0.98	0.98	0.98	4484

(b) InceptionV3

	precision	recall	f1-score	support
c0	0.96	0.97	0.97	498
c1	0.97	0.99	0.98	453
c2	0.99	0.98	0.99	463
c3	0.96	0.98	0.97	469
c4	0.97	0.97	0.97	465
c5	0.99	0.97	0.98	462
c6	0.96	0.97	0.97	465
c7	0.97	0.97	0.97	401
c8	0.94	0.93	0.93	382
c9	0.95	0.94	0.95	426
avg / total	0.97	0.97	0.97	4484

(c) ResNet-101

	precision	recall	f1-score	support
c0	0.94	0.95	0.95	498
c1	0.96	0.97	0.96	453
c2	0.98	0.98	0.98	463
c3	0.97	0.98	0.98	469
c4	0.99	0.96	0.97	465
c5	0.99	0.98	0.98	462
c6	0.95	0.97	0.96	465
c7	0.98	0.98	0.98	401
c8	0.95	0.92	0.93	382
c9	0.94	0.95	0.95	426
avg / total	0.97	0.96	0.96	4484

(d) ResNet-50

Fonte: Elaborado pelo Próprio Autor

normalizadas. A partir da Figura 10, é possível perceber que em cada arquitetura os valores das métricas estão bem próximos. O que significa que não há nenhuma classe onde o desempenho foi muito menor ou muito maior. Apenas na Arquitetura ResNet-50, têm-se um valor de *recall* de 0.92, mas que não difere muito dos padrões de desempenho vistos com a ResNet-50.

Analisadas as métricas, pode-se afirmar que a arquitetura que proporcionou os melhores resultados foi a VGG-16, que além do menor valor de *loss*, obteve também o maior valor de Taxa de Acerto e os menores números de classificação erradas de acordo com as Matrizes de Confusão. Além disso, a arquitetura InceptionV3 também obteve resultados muito próximos à VGG-16.

Como foi descrito na Seção 5.4, a arquitetura VGG-16 implementada neste trabalho passou por uma adaptação diferente das demais arquiteturas, que foi a especificação de que as 10 primeiras camadas da rede não são treináveis. Talvez essa configuração tenha ajudado a arquitetura VGG-16 a obter melhores resultados.

Já nas arquiteturas ResNet-50 e ResNet-101, talvez o conjunto de dados utilizado não fosse do tamanho adequado para extrair melhores resultados dessas arquiteturas. Um indício disso é que a partir da época 10, não houveram melhorias significativas nos valores de *loss*, como pode ser visto na Figura 7. Dessa forma, aparentemente, já na época 5 os modelos já atingiram

resultados muito próximos dos melhores possíveis para a arquitetura.

6 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho teve como objetivo descobrir a melhor técnica para classificação de imagens para o problema da detecção de distração de motoristas. Para isso foram apresentadas as implementações de modelos capazes de detectar a distração de motoristas baseado em imagens e apontado, ao final, o modelo com melhores resultados na classificação de distração de motoristas, que foi o modelo utilizando a arquitetura de Rede Neural Convolucional VGG-16.

Com os experimentos mostrados neste trabalho, é possível afirmar que o modelo implementado utilizando a arquitetura VGG-16, com algumas adaptações, é capaz de classificar satisfatoriamente imagens em algum tipo de distração ou sem distração, de acordo com o conjunto de dados utilizado.

Além disso, este trabalho teve como contribuição uma discussão sobre arquiteturas de Redes Neurais Convolucionais, ótimas para classificação de imagens. Além disso, foi desenvolvida uma imagem Docker com um ambiente com diversas ferramentas e bibliotecas para o uso de Aprendizado Profundo, que pode ser utilizada por qualquer pessoa.

Embora as máquinas utilizadas neste trabalho para a execução dos experimentos possuam configurações excelentes para aplicações comuns, para a execução de modelos de Aprendizado Profundo, são indicadas máquinas com GPU, o que permite a execução mais rápida de treinamento dos modelos. Com máquinas melhores teria sido possível a análise de outras arquiteturas, que talvez possibilitassem melhores resultados.

Dessa forma, um possível trabalho futuro é a implementação de outras arquiteturas, mais profundas – como *DenseNets* (HUANG *et al.*, 2017) –, utilizando melhores recursos computacionais. Outro possível trabalho futuro é a implementação de modelos mais leves que possam ser utilizados em sistemas embarcados – como *MobileNets* (HOWARD *et al.*, 2017). Além desses, outro possível trabalho futuro é fazer uma pesquisa em busca de descobrir técnicas que possam ser aplicados às arquiteturas InceptionV3, ResNet-50 e ResNet-101 com o objetivo de melhorar seus desempenhos – semelhante ao que foi feito com a arquitetura VGG-16, neste trabalho

REFERÊNCIAS

AGUIAR, D.; GUERRA, P. d. T. **Transferência de Conhecimento Utilizando Aprendizado Profundo Para Classificação de Imagens Histopatológicas**. [S.l.: s.n.], 2017.

BROWNLEE, J. **Machine Learning Mastery with Python: Understand Your Data, Create Accurate Models and Work Projects End-to-end**. [s.n.], 2016. Disponível em: <https://books.google.com.br/books?id=85V6nQAACAAJ>. Acesso em: 17 fev. 2018.

CHOLLET, F. **Deep learning with python**. [S.l.]: Manning Publications Co., 2017.

FRIEDMAN, J.; HASTIE, T.; TIBSHIRANI, R. **The elements of statistical learning**. [S.l.]: Springer series in statistics New York, 2001. v. 1.

GRUS, J. **Data science from scratch: first principles with python**. [S.l.]: O'Reilly Media, Inc., 2015.

HAN, J.; PEI, J.; KAMBER, M. **Data mining: concepts and techniques**. [S.l.]: Elsevier, 2011.

HAYKIN, S. S. **Neural networks and learning machines**. [S.l.]: Pearson Upper Saddle River, NJ, USA:, 2009. v. 3.

HOWARD, A. G. *et al.* Mobilenets: Efficient convolutional neural networks for mobile vision applications. **arXiv preprint arXiv:1704.04861**, 2017.

HUANG, G. *et al.* Densely connected convolutional networks. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2017.

JAN, W.-P.; LIU, X. **Deep Residual Learning for Image Recognition**. [S.l.]: IEEE, 2015.

KUTILA, M. *et al.* Driver distraction detection with a camera vision system. In: IEEE. **Image Processing, 2007. ICIP 2007. IEEE International Conference on**. [S.l.], 2007. v. 6, p. VI–201.

LECUN, Y. *et al.* Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, IEEE, v. 86, n. 11, p. 2278–2324, 1998.

MONARD, M. C.; BARANAUSKAS, J. A. Conceitos sobre aprendizado de máquina. **Sistemas inteligentes-Fundamentos e aplicações**, v. 1, n. 1, p. 32, 2003.

NICOLETTI, M. d. C. **Ampliando os limites do aprendizado indutivo de máquina através das abordagens construtiva e relacional.** Tese (Doutorado) — Universidade de São Paulo, 1994.

OCHI, L. S.; DIAS, C. R.; SOARES, S. S. F. Clusterização em mineração de dados. **Instituto de Computação-Universidade Federal Fluminense-Niterói**, 2004.

PATTERSON, J.; GIBSON, A. **Deep Learning: A Practitioner's Approach**. [S.l.]: "O'Reilly Media, Inc.", 2017.

PICKRELL, T. M. **Driver electronic device use in 2015**. [S.l.], 2016.

PRF Polícia Rodoviária Federal. Portal de Dados Abertos. **Acidentes de Trânsito**. *In*:_____, 2018. Disponível em: <https://www.prf.gov.br/portal/dados-abertos/acidentes>. Acesso em: 17 fev. 2018.

RASCHKA, S.; MIRJALILI, V. **Python Machine Learning, 2nd Ed.** 2. ed. Birmingham, UK: Packt Publishing, 2017. ISBN 978-1787125933.

REZENDE, S. O. **Sistemas inteligentes: fundamentos e aplicações**. [S.l.]: Editora Manole Ltda, 2003.

ROSENBLATT, F. **The perceptron, a perceiving and recognizing automaton Project Para**. [S.l.]: Cornell Aeronautical Laboratory, 1957.

RUSSAKOVSKY, O. *et al.* ImageNet Large Scale Visual Recognition Challenge. **International Journal of Computer Vision (IJCV)**, v. 115, n. 3, p. 211–252, 2015.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. **arXiv preprint arXiv:1409.1556**, 2014.

SPANHOL, F. A. *et al.* A dataset for breast cancer histopathological image classification. **IEEE Transactions on Biomedical Engineering**, IEEE, v. 63, n. 7, p. 1455–1462, 2016.

SZEGEDY, C. *et al.* Going deeper with convolutions. In: **The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2015.

SZEGEDY, C. *et al.* Rethinking the inception architecture for computer vision. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2016. p. 2818–2826.

VASCONCELOS, C. N.; GONZALEZ, E. W. Deep learning - teoria e prática, chapter 6. **Jornada de Atualização em Informática na Educação**, Sociedade Brasileira de Computação, 2017.

ZEILER, M. D.; FERGUS, R. Visualizing and understanding convolutional networks. In: SPRINGER. **European conference on computer vision**. [S.l.], 2014. p. 818–833.