UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
PROJETO DE GRADUAÇÃO

Kaio Giurizatto Utsch

USO DE REDES NEURAIS CONVOLUCIONAIS PARA CLASSIFICAÇÃO DE IMAGENS DIGITAIS DE LESÕES DE PELE

Vitória-ES

Janeiro/2018

Kaio Giurizatto Utsch

USO DE REDES NEURAIS CONVOLUCIONAIS PARA CLASSIFICAÇÃO DE IMAGENS DIGITAIS DE LESÕES DE PELE

Parte manuscrita do Projeto de Graduação do aluno Kaio Giurizatto Utsch, apresentado ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Engenheiro Eletricista.

Vitória-ES

Janeiro/2018

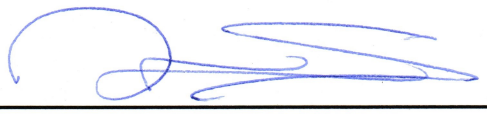
Kaio Giurizatto Utsch

USO DE REDES NEURAIS CONVOLUCIONAIS PARA CLASSIFICAÇÃO DE IMAGENS DIGITAIS DE LESÕES DE PELE

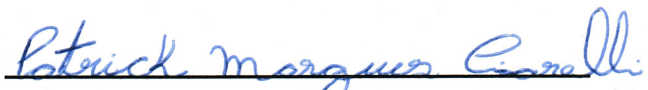
Parte manuscrita do Projeto de Graduação do aluno Kaio Giurizatto Utsch, apresentado ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Engenheiro Eletricista.

Aprovado em 5 de Janeiro de 2018.

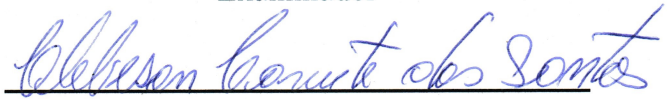
COMISSÃO EXAMINADORA:



Prof. Dr. Jorge Leonid Aching Samatelo
Universidade Federal do Espírito Santo
Orientador



Prof. Dr.Patrick Marques Ciarelli
Universidade Federal do Espírito Santo
Examinador



Msc. Clebeson Canuto dos Santos
Universidade Federal do Espírito Santo
Examinador

Vitória-ES

Janeiro/2018

Ao meu filho André, com quem aprendo mais do que ensino.

AGRADECIMENTOS

Aos meus pais, por todo o amor, carinho e educação ao longo dos anos, e por não me deixarem desistir.

Ao meu orientador pela indispensável ajuda e apoio, e por despertar meu interesse por esse tema fascinante.

À banca examinadora pela aceitação do convite e pelo tempo investido para leitura e avaliação desse trabalho.

RESUMO

O câncer de pele é o tipo de câncer mais comum no Brasil. O melanoma é seu subtipo mais mortal. Portanto, é essencial que seja detectado em seus estágios iniciais, quando a taxa de sobrevivência ainda é alta. Entretanto, ele é muito fácil de ser confundido por pintas comuns ou outros tipos de lesões de pele, até mesmo por especialistas. Assim, alguma ferramenta de diagnóstico automatizado se torna indispensável. Porém, as soluções automatizadas existentes não são muito superiores aos diagnósticos comuns. A partir de 2012 houveram grandes avanços na aplicação de redes neurais para a classificação de imagens. A aplicação de redes neurais convolucionais profundas está superando a performance humana em diversas tarefas. O presente projeto de graduação faz uso de uma rede convolucional para tentar solucionar o problema de classificação binária de imagens de melanomas ou ceratoses seborréicas contra nevos. A tarefa a solucionar foi obtida do desafio ISIC 2017, que provê um banco de dados para realizar o treino das redes e para validar a solução. Para auxiliar e facilitar a tarefa foram usadas as técnicas de pré-processamento de imagens e transferência de aprendizado. Um estudo aprofundado é feito sobre a arquitetura da rede usada, detalhando seu funcionamento interno. São treinados diversos classificadores para a tarefa dos quais o melhor obteve um desempenho equiparável a soluções de outras equipes. Especificamente, é obtido uma média de área sob a curva de característica de operação do receptor de 0.877 quando testado no banco de dados do desafio ISIC 2017, ficando situado entre os 10 melhores resultados.

Palavras-chave: Câncer; Melanoma; *Deep-learning*; Redes neurais convolucionais; *Transfer-learning*; *Inception*; Processamento digital de imagens.

ABSTRACT

Skin cancer is the most common type of cancer in Brazil. Melanoma is its deadliest subtype. Therefore it is essencial that it be detected in its early stages, when the survival rate is still high. However, it is easily mistaken with common spots, even by specialists. Thus, some form of automated diagnostic tool becomes indispensable. But existing automated tools are not very superior to common diagnoses. Since 2012 there have been great advances in the aplication of convolutional neural networks for image classification. The use of deep convolutional neural networks is surpassing human performance in a variety of tasks. This graduation project uses one of these networks to try to solve the problem of melanoma or seborreic keratosis versus nevi binary image classification. The task was taken from the 2017 ISIC challenge, which provided a dataset to train classifiers upon and to validate the solution. To help and ease the task, image pre-processing and transfer learning techniques were used. A deeper investigation is done on the network architecture that was used, detailing its inner workings. A group of different classifiers is trained on the task, the best of which obtained a performance equivalent to other teams' solutions. Specifically, a mean of the receiver operating characteristic areas under the curves of 0.877 is obtained, when tested on the ISIC 2017 challenge dataset, achieving a place in the top ten range of the results.

Keywords: Cancer; Melanoma; Deep Learning; Convolutional Neural Networks; Transfer Learning; Inception; Digital Image Processing.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AUC	*Area Under the Curve*
CNN	*Convolutional Neural Network*
CSV	*Comma Separated Values*
DL	*Deep Learning*
GFLOPS	*Giga Floating Point Operations Per Second*
GPU	*Graphics Processing Unit*
ILSVRC	*Imagenet Large Scale Visual Recognition Challenge*
ISBI	*International Symposium on Biomediacal Imaging*
ISIC	*International Skin Imaging Collaboration*
JPEG	*Joint Photographic Experts Group*
ML	*Machine Learning*
MLP	*Multilayer Perceptron*
NIN	*Network in Network*
ROC	*Receiver Operating Characteristic*
SGD	*Stochastic Gradient Descent*
UFES	*Universidade Federal do Espírito Santo*

SUMÁRIO

1 INTRODUÇÃO

1.1 Apresentação

O câncer de pele é o tipo de câncer mais comum no Brasil (1). Existem vários tipos de câncer de pele, dos quais podemos citar melanoma, carcinoma basocelular e carcinoma espinocelular como os mais comuns. Destes, o melanoma é considerado o mais perigoso (2) pois apesar de representar menos de 5% dos tumores de pele, é o responsável por mais de 70% das mortes (1) (3).

O diagnóstico é realizado primariamente por inspeção visual, podendo ser aprofundada por análise dermatoscópica. Uma biópsia geralmente é necessária para a confirmação. A distinção entre melanoma e nevos (pintas) melanocíticos não é trivial, especialmente nos estágios iniciais do melanoma. A acurácia de diagnóstico por inspeção visual não auxiliada é de aproximadamente 60% (4), aumentando para 75%-84% com o auxílio de dermatoscopias por profissionais treinados (4) (5) (6).

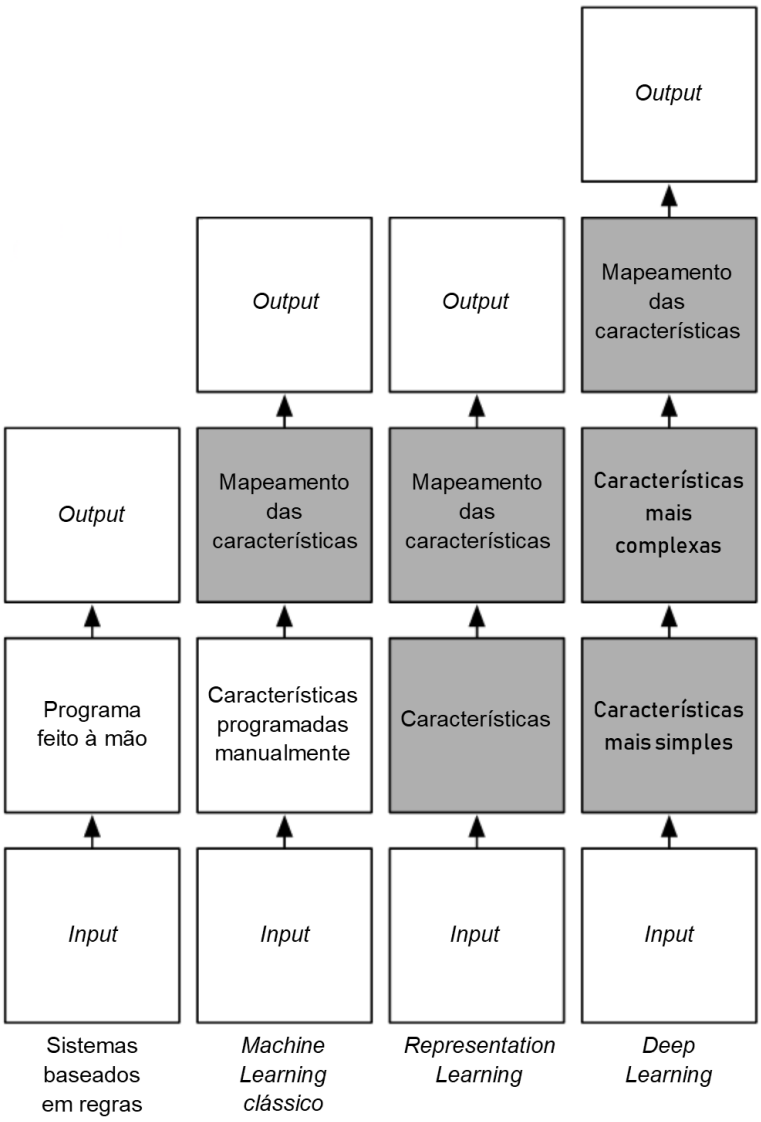
Detecção rápida é essencial para a cura, já que a taxa de sobrevivência de mais de 95% nos estágios iniciais da doença cai para menos de 15% nos estágios avançados (1) (3).

Dessa forma, ferramentas de diagnóstico automático são essenciais aos médicos. O diagnóstico auxiliado por computador consegue aumentar a precisão do diagnóstico, assim como sua velocidade. Computadores podem conseguir extrair algumas informações, como variações na coloração, assimetrias ou características da textura que podem não ser facilmente percebidos por olhos humanos.

Diferentes procedimentos foram propostos para melhorar o diagnóstico de câncer de pele melanoma de forma manual, dentre eles: (*i*) a lista de sete pontos, (*ii*) regra ABCD, e (*iii*) o método Menzies (2). Alguns desses procedimentos são usados por métodos de diagnóstico automatizado, mas são baseados em características do câncer de pele que são difíceis de extrair de maneira automatizada, como por exemplo, características morfológicas que o câncer de pele apresenta. Por isso, em geral, torna-se necessário criar manualmente extratores de características baseados em técnicas de visão computacional. Esse trabalho é laborioso e é limitado pelo método em que se baseia.

Em geral, métodos de diagnóstico automatizados presentes na literatura têm a seguinte estrutural geral: (*i*) aquisição da imagem de lesão de pele; (*ii*) segmentação da lesão em si da região de pele; (*iii*) extração de características da mancha da lesão e (*iv*) classificação das características.

Figura 1 – Comparação dos métodos de solução computacional.



Fonte: (7).

Tal sequência de passos tem como etapa crítica a extração de características, que envolve um esforço importante na implementação do sistema automatizado, já que o rendimento do classificador depende fortemente da capacidade discriminativa das características extraídas. Essa sequência está demonstrada nas duas primeiras colunas da Figura 1. Na primeira, todo o classificador é codificado manualmente. Já na segunda, é codificado um extrator de características das imagens e o computador aprende por si como usar essas características para classificar de forma ideal, fazendo uso de *Machine Learning* - ML (tradução, aprendizado de máquinas).

Na atualidade, existe uma área de pesquisa relacionada a ML denominada *Deep Learning* - DL (tradução livre, aprendizado profundo) que permite implementar um sistema de

classificação *end-to-end* (7), ou seja, máquinas de aprendizagem que têm a capacidade de extrair características de importância dos dados e simultaneamente efetuar a tarefa de classificação com alto nível de precisão. É semelhante à técnica chamada *representation learning* (terceira coluna), na qual as características e a classificação são aprendidas. No entanto, essas características aprendidas não conseguiam obter resultados satisfatórios, e eram por via de regra inferiores aos métodos clássicos de ML. O diferencial de DL é o uso de arquiteturas profundas, com muitas camadas, que permitem uma extração de características hierárquicas. Corresponde dessa forma à quarta coluna da Figura 1. Especificamente, *Deep learning* surgiu de um conjunto de algoritmos inspirados no córtex visual de mamíferos (8), que tenta modelar abstrações de alto nível de dados usando um grafo profundo com várias camadas de processamento hierárquico, em que as computações de uma camada são compostas de várias transformações lineares e não lineares de outra inferior (9).

Entre as diferentes arquiteturas de redes neurais profundas presentes na literatura, as de maior importância em aplicações de processamento de imagens e visão computacional são as *Convolutional Neural Networks* – CNN (tradução, redes neurais convolucionais), propostas em 1998 por Yann LeCun (10) sendo aplicadas ao problema de classificação de dígitos manuscritos, obtendo uma taxa de classificação de 99.05% sobre o banco de dados MNIST. Em 2012 com o advento de maior poder computacional, melhores algoritmos e novas ideias, este tipo de arquitetura obteve um resultado surpreendente ao ser aplicada no desafio *Imagenet Large Scale Visual Recognition Challenge* - ILSVRC (11), obtendo uma taxa de erro de classificação 37% menor que o segundo qualificado.

Redes neurais estão superando a performance humana em muitas áreas onde são aplicadas, seja a tarefa uma classificação de imagens (12) ou jogos altamente subjetivos (13) (14). Depois do desenvolvimento ou escolha de uma arquitetura, os únicos pré-requisitos são um banco de dados suficientemente grande (quantidade de experiência) e poder de processamento computacional (velocidade de aprendizado) adequado. Porém, como a tecnologia está maturando e soluções estáveis estão sendo implementadas com sucesso em outras áreas, estão ocorrendo tentativas de aplicar essa tecnologia para a área médica, e mais uma vez, com sucesso, a performance está atingindo níveis superiores à de especialistas humanos (15) (16).

Para problemas da área médica, redes neurais estão sendo aplicadas com sucesso desde ao menos os meados dos anos 90 (17), mas apenas os avanços recentes das CNN possibilitaram aplicar essas redes ao diagnóstico médico por imagens com desempenhos satisfatórios. Recentemente, esse tipo de aplicação chegou a superar o desempenho de médicos especialistas na detecção de retinopatia diabética em imagens da retina (15).

Devido a esses promissores resultados, vindos de um campo que apenas recentemente saiu de um longo período de hibernação (18), as possibilidades de desenvolvimento parecem ser grandes. A maioria das aplicações é feita em áreas que estão diretamente conectadas com tecnologia. A área médica ainda não dispõe de muitas soluções que façam uso dessa tecnologia. Aplicações que façam uso de DL têm um grande potencial para mudar para melhor a vida humana com estudos e desenvolvimentos que apliquem DL para problemas médicos.

1.2 Objetivos

Este trabalho tem como objetivo geral o desenvolvimento de um classificador de imagens de lesões de pele baseado em uma arquitetura de rede neural convolucional. Para isso, pretende-se aplicar técnicas de processamento de imagens que permitam normalizar as imagens a tratar e técnicas de *transfer learning* que permitam usar arquiteturas de redes neurais já consolidadas da literatura e aplicá-las ao domínio em estudo.

1.2.1 Objetivos Específicos

- Realizar uma revisão bibliográfica de trabalhos que estudam o problema de classificação de câncer de pele;
- Estudar o estado da arte em classificação de imagens com CNN;
- Obter um banco de imagens para estudo e análise do problema;
- Implementar um sistema que permita a classificação automatizada de câncer de pele usando imagens digitais;
- Validar e testar o sistema implementado;

1.3 Métodologia Adotada

A pesquisa realizada no presente trabalho enquadra-se no campo de pesquisa Teórico-Aplicada Quantitativa. Em sentido amplo de pesquisa, a investigação científica tratada neste trabalho tem a metodologia descrita pelas características na Tabela 1.

Tabela 1 – Características da metodologia científica aplicada ao trabalho.

Quanto à natureza	Pesquisa aplicada
Quanto aos objetivos	Exploratórios e descritivos
Quanto às abordagens	Quantitativas
Quanto aos procedimentos	Experimentais

Baseado nestas questões, o trabalho foi desenvolvido conforme as etapas apresentadas a seguir. Essas etapas englobam fases, processos, ações principais e produção de artefatos.

Na primeira fase, "Planejamento", é definido o escopo da pesquisa e os objetivos. Na segunda fase, "Execução", é efetuada a pesquisa bibliográfica, definidas as questões norteadoras e o projeto é construído. Na terceira fase, o protótipo é finalizado e os resultados são apresentados.

1.4 Trabalhos relacionados

Os trabalhos lidos para a elaboração deste projeto podem ser divididos em duas classes:

- Diagnósticos automatizados de câncer de pele;
- Métodos e arquiteturas em DL de ponta.

A primeira classe compreende uma série de métodos convencionais de classificação automatizada de lesões de pele ou outras doenças, e servem para contextualização histórica e comparação.

A segunda compreende os trabalhos que descrevem métodos ou arquiteturas de DL que estão próximos do estado da arte, mesmo que não relacionados à área médica. Estes serviram como ferramentas básicas de aprendizado e de auxílio na hora de elaboração da proposta.

Alguns trabalhos que se enquadram em ambas as classes se assemelham muito a esse presente trabalho e serviram de fonte de ideias e para posterior comparação de resultados.

Dentre os diversos trabalhos lidos para a elaboração deste trabalho pode-se citar como de importância especial:

- ***Computer Aided Melanoma Skin Cancer Detection Using Image Processing*** (2) é um trabalho recente no qual os autores fazem uso de processamento de imagens para criar manualmente extratores de características. As características extraídas são baseadas nos métodos por diagnóstico visual, ensinados nas escolas de medicina.

- ***Visualizing and Understanding Convolutional Networks*** (19) Os autroes fazem uso da melhor arquitetura de classificação de imagens de 2012 (20) para aplicar técnicas de visualização das características aprendidas pela rede. Com a nova intuição adquirida fazem pequenas modificações na rede original e conseguem um aumento de desempenho considerável. Torna-se o melhor classificador de 2013 sem aumentar os recursos necessários. Fazem uso da informação aprendida pela rede em outras tarefas e mostram que essa informação é altamente transferível a novas tarefas.

- ***Going deeper with convolutions*** (21) Os autores aplicam ideias inovadoras para a estrutura de uma rede neural de forma a produzir o melhor e mais eficiente classificador dentre todos os competidores do desafio ILSVRC 2014. Aplicam mecanismos complexos para melhorar a capacidade de seu classificador em vez de apenas aumentar o tamanho do classificador em uma tentativa de força bruta.

- ***Dermatologist-level classification of skin cancer with deep neural networks*** (3) Os autores fazem uso de *transfer learning* para classificar imagens de câncer de pele. Passaram pela laboriosa tarefa de construir um banco de dados grande formado de imagens de diversos bancos de dados menores, dentro do domínio público ou não. A partir daí criaram uma estrutura taxonômica para auxiliar na classificação em distinções mais específicas de tipos de lesão. Treinaram com base nessas classes e obtiveram um resultado superior quando avaliaram o classificador nas classes com distinção mais grosseira. Compararam os resultados com um conjunto de dermatologistas e obtiveram resultados equivalentes ou superiores aos classificadores humanos. Realizaram diversas técnicas de visualização de características para melhor entender o que foi aprendido pela rede.

1.5 Questões norteadoras

- Técnicas de constância de cores e realce de bordas podem ser combinadas em uma etapa de pré-processamento, com o intuito de diminuir a variabilidade intraclasse das imagens das classes melanoma, ceratose seborréica e nevo melanocítico?

- Como se comparam as redes neurais, especificamente as CNNs, ao desempenho de métodos de inspeção visual de lesões de pele e de aplicações computadorizadas com características desenvolvidas manualmente?
- O quão transferíveis são as características aprendidas de uma CNN treinada em uma tarefa de classificação de imagens comuns à tarefa de diagnóstico médico por imagem quando transferidas por meio de *transfer learning*?

1.6 Estrutura do Texto

O presente trabalho está estruturado da seguinte maneira:

- **Introdução**: este capítulo inicial tem como objetivo contextualiar o trabalho e o problema aqui estudado, apresentando o problema e ideias iniciais sobre a solução do mesmo. Além disso, são apresentados os objetivos da realização deste projeto de graduação;
- **Referencial Teórico**: aqui serão apresentados os assuntos tratados neste trabalho, tais como: redes neurais convolucionais, arquitetura *Inception* e *transfer learning*;
- **Solução Proposta**: neste capítulo é apresentado o sistema sugerido para a resolução do problema em estudo, os experimentos, os resultados obtidos, as comparações com outros trabalhos e uma breve análise dos resultados;
- **Conclusão**: no capítulo final deste trabalho são apresentadas as conclusões e os trabalhos futuros.

2 EMBASAMENTO TEÓRICO

2.1 Introdução

Este capítulo tem por finalidade estabelecer os conceitos teóricos necessários usados no trabalho. Portanto, o capítulo inicia-se com a definição de uma Rede Neural Convolucional e como ela é caracterizada, em continuação é descrita a rede convolucional *GoogLeNet*, começando por uma resenha histórica e logo uma exposição de sua estrutura. Finalmente é explicado a técnica de ML *transfer learning*, iniciando com uma explicação geral sobre o assunto, sua definição matemática e finalizando com sua aplicação em redes neurais convolucionais.

2.2 Redes Neurais Convolucionais

CNN é uma conhecida arquitetura de aprendizado profundo inspirada no mecanismo de percepção visual dos seres vivos. Sua concepção foi possibilitada por meio de uma série de desenvolvimentos em diferentes campos de estudo datando desde o final da década de 1960.

Em 1968, após estudos sobre o córtex visual de primatas, foi postulada a ideia de uma estrutura hierárquica nos neurônios do sistema visual, em que determinadas células em um nível do sistema visual formam um "campo receptivo" para células em um nível superior. Dessa forma, campos receptivos simples podem ser combinados para formar campos receptivos grandes e complexos (8).

Em 1980 é proposta a rede neural *neocognitron* que faz uso das ideias de (8) na forma de uma rede baseada em camadas de neurônios que utilizam conexões esparsas, semelhantes ao que hoje é chamado de convolução. O *neocognitron* foi usado para o reconhecimento de dígitos manuscritos (22), apresentando como características principais aprender de maneira não supervisionada, e ser invariante ao deslocamento das entradas.

Em 1989 LeCunn (23) propôs uma arquitetura de rede baseada em camadas convolucionais e camadas totalmente conectadas. Tal rede foi treinada por meio do algoritmo de *backpropagation*, sendo aplicada também para o reconhecimento dígitos manuscritos atingindo um erro de classificação de 5%, usando 2 camadas convolucionais e 2 totalmente conectadas.

Em 1994 uma rede com camadas convolucionais (17) é aplicada no problema de detecção

de microcalcificações de mamografias digitais. Os autores também haviam desenvolvido uma aplicação de reconhecimento de dígitos paralelamente a (23).

Em 1998 é estabelecido o modelo padrão das CNNs como redes de camadas convolucionais intercaladas de operações de *max pooling* para redução da dimensionalidade e algumas camadas totalmente conectadas para a classificação final (10). Também em (10) é descrita a famosa rede neural LeNet-5. Ela é comparada a outros sistemas de reconhecimento de dígitos manuscritos apresentando um desempenho superior.

Em 2005, com um maior desenvolvimento dos *Graphics Processing Units* - GPUs devido à demanda por habilidades de processamento gráfico para jogos 3D, é proposto usar as GPUs para realizar cálculos computacionais para aplicações em ML, já que as GPUs implementam de forma altamente otimizada multiplicações de matrizes para computações de geometria 3D. Com algumas alterações, algoritmos de ML foram implementados de forma altamente efetiva em GPUs (24).

Em 2012, foi proposta a CNN AlexNet (20). Esta rede participou no desafio *ImageNet Large-Scale Visual Recognition Challenge* (ILSVRC) ganhando o desafio por uma margem ampla, reduzindo a taxa de erro do estado da arte de 25.8% para 16.4% (11). Tal resultado popularizou o uso das redes CNNs na área de visão computacional.

Em 2014, foi apresentada a VGG-19 (25) composta por 19 camadas. A principal contribuição da VGG-19 foi demonstrar empiricamente que o número de camadas (a profundidade da rede) é um componente crítico para um bom desempenho.

Em 2015, foi proposta a CNN *GoogLeNet* (21). Esta rede teve como objetivo principal usar o menor número de parâmetros sem perder profundidade, propondo-se o modulo *Inception* que reduziu significativamente o número de parâmetros da rede.

Resumidamente, pode-se apreciar que as CNNs foram aplicadas a tarefas de visão computacional desde o final da década de 1980. No entanto, houveram somente algumas aplicações dispersas, e a pesquisa ficou efetivamente inativa até meados dos anos 2000, quando houve um aumento na quantidade de dados rotulados, complementados por algoritmos aprimorados e o crescente poder computacional das GPUs que contribuíram para o seu avanço e um rápido progresso desde 2012. Atualmente, as CNNs continuam sendo um assunto relevante de pesquisa, procurando-se novas arquiteturas e aplicações. Um resumo atual sobre o tema pode ser encontrado em (26) e (27).

2.2.1 Componentes Básicos de uma CNN

Existem inúmeras variantes de arquiteturas da CNN na literatura. No entanto, seus componentes básicos são muito parecidos. Tomando o famoso LeNet-5 como exemplo, ele consiste em três tipos básicos de camadas: (*i*) convolucionais, (*ii*) de *pooling*, e (*iii*) totalmente conectadas. Na continuação é descrita brevemente cada uma delas.

2.2.1.1 A camada de Convolução

A camada convolucional tem como objetivo aprender características representativas das entradas. A camada de convolução é composta de vários *kernels* de convolução, também conhecidos como filtros, que são usados para calcular diferentes mapas de características.

Especificamente, cada neurônio de um mapa de características é conectado a uma região de neurônios vizinhos na camada anterior. Tal vizinhança é referida como campo receptivo do neurônio na camada anterior.

Os mapas de características intermediários são obtidos usando diferentes filtros. Matematicamente, o valor da característica na localização (i, j) referente ao k-ésimo mapa de características da l-ésima camada, $z^l_{i,j,k}$ é calculado como:

$$z^l_{i,j,k} = {\mathbf{w}^l_k}^T \mathbf{x}^l_{i,j} + b^l_k. \tag{2.1}$$

onde $\mathbf{x}^l_{i,j}$ é o fragmento da entrada centrada na posição (i, j) referente à l-ésima camada; $\mathbf{w}^l_k$ e b^l_k são o vetor de pesos e o termo de bias do k-ésimo filtro da l-ésima camada, respectivamente. Observe que o filtro $\mathbf{w}^l_k$ que gera o mapa de característica $\mathbf{z}^l_{:,:,k}$ é compartilhado. Tal mecanismo de compartilhamento de pesos tem várias vantagens, como a redução da complexidade do modelo e facilitar a etapa de treinamento da rede.

Em geral, um mapa de características final é calculado em duas etapas. Primeiro, a entrada é convoluída com um filtro já aprendido e, em seguida, é aplicada uma função de ativação não-linear aos resultados da convolução. Essa não-linearidade não é parte integral da camada convolucional, mas na prática, quase todas camadas convolucionais são seguidas de uma não-linearidade. Essas não linearidades são desejáveis, já que permitem a detecção de características não-lineares.

Seja $f(\bullet)$ a função de ativação não linear, então, o valor que retorna a função de ativação

quando a entrada é a característica convolucional $z^l_{i,j,k}$ pode ser calculado como:

$$a^l_{i,j,k} = f(z^l_{i,j,k}). \tag{2.2}$$

Funções de ativação típicas são as funções sigmoide, tangente hiperbólico (28) e mais recentemente, ReLU (20) (29). A mudança das funções de ativação tradicionais para a ReLU marcou um avanço pois acelerou significantemente o tempo de treino e de inferência das redes neurais. Dependendo da aplicação, tempos de treino até 6 vezes menores foram reportados, se comparados à ativação tangente hiperbólico (20).

2.2.1.2 A camada de *Pooling*

A camada de *pooling* (tradução livre, agrupamento) visa alcançar a invariância aos deslocamentos, reduzindo a resolução espacial dos mapas de características. Essa redução espacial é alcançada por meio de um *stride* (tradução livre, passo) não unitário. Dessa forma, apenas uma ativação a cada n (tamanho do stride) posições da entrada é calculada em cada dimensão espacial. O tamanho de *stride* mais comum é 2, de forma que as dimensões espaciais são reduzidas pela metade. As operações de *pooling* geralmente são colocadas entre duas camadas convolucionais.

Cada mapa de características de uma camada de *pooling* está conectada ao correspondente mapa de características da camada convolucional anterior. Matematicamente, seja a função de *pooling* denotada como $pool(\bullet)$, então, a saída gerada pela função de *pooling* para o mapa de características $\mathbf{a}^l_{:,:,k}$, é representada pela expressão:

$$y^l_{i,j,k} = pool(a^l_{m,n,k}), \forall\ (m,n) \in \mathcal{R}_{i,j} \tag{2.3}$$

onde $\mathcal{R}_{i,j}$ é uma vizinhança local centrada na posição (i,j). As operações típicas de *pooling* são:

- *max pooling*, que retorna apenas o maior valor dentro de seu campo receptivo (30);
- *average pooling*, que retorna a média dos valores do seu campo receptivo (31).

2.2.1.3 A camada Totalmente conectada

Uma camada totalmente conectada toma todos os neurônios na camada anterior e os conectam a cada neurônio da camada atual. Em geral, os filtros da primeira camada

convolucional são projetados para detectar características de baixo nível, enquanto os filtros das camadas superiores detectam características mais abstratas. Então, ao empilhar várias camadas convolucionais e de *pooling*, pode-se extrair gradualmente características com um maior nível de abstração. As camadas totalmente conectadas que seguem as camadas convolucionais e de *pooling* têm por finalidade interpretar essas características de alto nível de abstração efetuando funções do raciocínio complexo. Por exemplo, a classificação de objetos em uma imagem, a detecção de pedestres em uma sequência de vídeo, etc.

Finalmente, depois das camadas totalmente conectadas, vem a camada de saída. Para o problema de classificação é comum usar tantos neurônios quanto existam classes a prever e a saída de cada neurônio tem uma função de ativação de tipo *softmax* (11). A *softmax* é responsável por transformar as saídas dos neurônios em um formato de probabilidades. Isto é, todos os valores são não negativos, menores que um, e o somatório da saída de todos os softmax é igual a 1. Esse formato é necessário para a função de perda *cross entropy* (tradução, entropia cruzada), que é a mais utilizada para nos casos de regressão logística como o é a classificação de imagens.

2.2.2 Treinamento de uma CNN

CNNs em geral fazem uso de algoritmos de aprendizado para ajustar seus parâmetros livres (isto é, as bias e os pesos) de modo que se obtenha os valores de saída esperados. O algoritmo comumente usado para este propósito é o algoritmo de *backpropagation* (32) (23). Tal algoritmo calcula o gradiente de uma função objetivo (também conhecida como função custo/perda/desempenho) para determinar como ajustar os parâmetros da rede a fim de minimizar os erros que afetam o desempenho na tarefa de predição.

Formalmente, seja $\boldsymbol{\theta}$ a concatenação de todos os parâmetros livres da CNN (as bias e os pesos). O parâmetro ótimo, $\boldsymbol{\theta}_{opt}$, para uma tarefa específica pode ser obtida ao minimizar uma função de perda apropriada. Suponha=se que se tem um conjunto de treinamento de N pares entrada-saída $\{\mathbf{x}_k, \mathbf{y}_k\}_{k=1}^{N}$, onde, $\mathbf{x}_k$ é o k-ésimo valor de entrada e $\mathbf{y}_k$-ésimo valor de saída esperada e $\mathbf{o}^k$ é a saída gerada pela CNN para a entrada $\mathbf{x}_k$, então uma função de perda a minimizar é a média das perdas sobre o conjunto de treinamento, ou seja:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N} \sum_{k=1}^{N} l(\boldsymbol{\theta}, \mathbf{y}_k, \mathbf{o}_k) \tag{2.4}$$

Treinar uma CNN é um problema de otimização global. Ao minimizar a função de perda,

pode-se encontrar o melhor conjunto de parâmetros. Ou seja:

$$\boldsymbol{\theta}_{opt} = min_{\boldsymbol{\theta} \in \mathcal{A}} \{\mathcal{L}(\boldsymbol{\theta})\} \tag{2.5}$$

onde $\mathcal{A}$ é o espaço de parâmetros.

O algoritmo *Stochastic Gradient Descent* (SGD) é a solução comum para otimizar os parâmetros de uma rede CNN ao solucionar a expressão (2.5) (33)(34). Ele é uma aproximação iterativa do método de otimização de gradiente descendente, no qual o gradiente descendente é realizado apenas sobre uma fração do conjunto de treino a cada iteração. O tamanho dessa fração é denominado *batch size* (tradução livre, tamanho do lote), e é o número de exemplos de treino usado a cada iteração. O uso de tamanhos de *batch* menores serve dois propósitos: (*i*) permite que todos os dados de treino caibam na memória, mesmo para redes maiores; (*ii*) evita mínimos locais da função de perda devido ao ruído de uma seleção aleatória de exemplos de treino.

2.2.3 Sobre a Natureza dos dados de entrada de uma CNN

No contexto de ML, cada instância de uma entrada de uma CNN pode ser vista como um tensor (uma generalização de vetores e matrizes para espaços de dimensões mais altas) de ordem 3. Se são adicionadas várias dessas instâncias em um *batch* para treino, é gerado um tensor de ordem 4. Um *voxel* (de *volumetric pixel*) denota aqui um elemento designado pelos três (ou quatro) índices necessários para endereçá-lo de maneira única. Isso é equivalente a como é endereçado um vetor com um índice, ou uma matriz com dois índices.

Supondo um tensor de entrada $(N_{in}, A_{in}, L_{in}, C_{in})$, onde:

- N_{in} Tamanho do *batch*
- A_{in} Altura
- L_{in} Largura
- C_{in} Número de canais

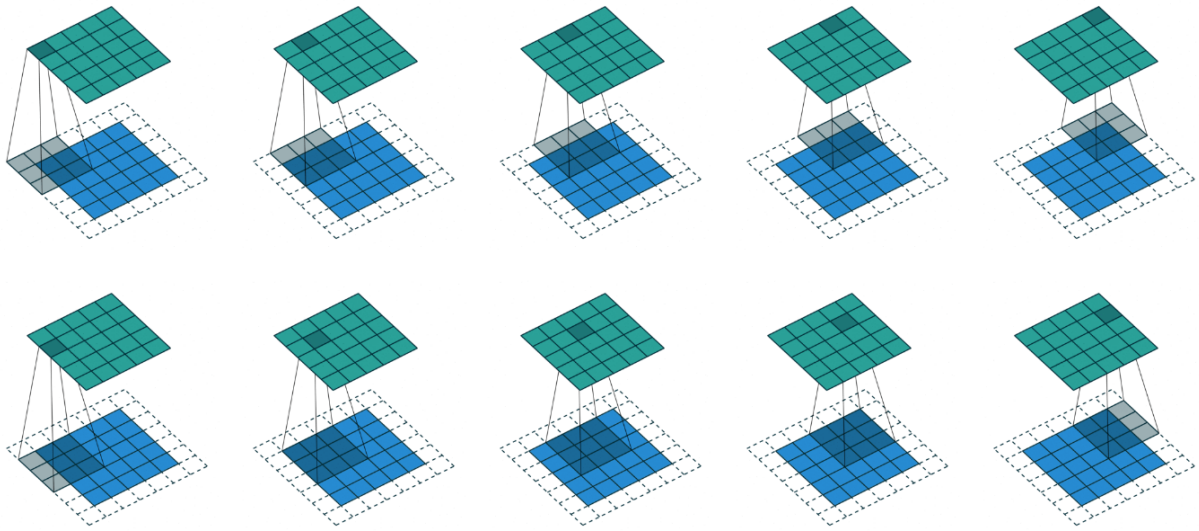
No caso de uma imagem em cores, N_{in} é o tamanho de *batch* escolhido (o número de imagens de treino), A_{in} é a altura da imagem em pixels, L_{in} é a largura da imagem em pixels e C_{in} é o número de canais, 3, para imagens RGB.

Também no contexto de ML, a operação de convolução é similar ao conceito matemático de convolução discreta. Aqui, uma camada convolucional é composta por um número $N_{filtros}$ de filtros. Um filtro convolucional simples tem seu formato definido por (A_{filtro}, L_{filtro}, C_{in}), onde:

- A_{filtro} Altura do filtro
- L_{filtro} Largura do filtro
- C_{in} Número de canais do tensor de entrada

Para efeito de simplificação, dá-se primeiro um exemplo com apenas um canal, tanto para a entrada quanto para a saída. Dessa forma torna-se possível uma representação gráfica da operação de convolução. Parte da operação descrita a seguir está demonstrada na Figura 2.

Figura 2 – Passos de uma convolução 2D.



Fonte: <https://github.com/vdumoulin/conv_arithmetic>.

Uma parte da imagem de entrada (em azul na imagem) é sobreposta com o filtro (sombreamento na camada azul). Seus valores são multiplicados ponto a ponto e é obtido um somatório dessas multiplicações. Esse é o valor de um pixel (região sombreada na camada verde) da imagem de saída (camada verde). Note que a imagem azul teve valores adicionados em seu contorno (em branco, usualmente zero). Após calcular um valor da imagem de saída do filtro é sobreposto a um outro conjunto de pixels da imagem de entrada e a operação é repetida, até se obter todos os pixels da imagem de saída.

Quando se tem mais de um canal a visualização não é tão trivial, porém, pode-se imaginar um paralelepípedo sendo sobreposto a uma fração de um outro, da mesma forma que no caso bidimensional se sobrepôs formas retangulares. O somatório da multiplicação ponto a ponto desses paralelepípedos dá o valor de um voxel (um pixel de um canal de uma instância). Deslizando-se esse filtro e repetindo a operação, obtém-se um canal (que tem altura e largura apenas).

O número de canais na saída depende de quantos filtros se tem. Para gerar os outros canais de saída, a operação é repetida com outros filtros (mesmas dimensões, mas valores diferentes). Após repetir essa operação com $N_{filtros}$ filtros tem-se a saída para uma instância ou imagem de treino.

Esse procedimento todo será repetido N_{in} vezes para formar a saída da camada convolucional para um *batch* de treino. No algoritmo *backpropagation*, são os pesos e as bias dos filtros que são modificados e aprendidos.

2.3 Rede Neural *GoogLeNet*

2.3.1 Desafio ILSVRC

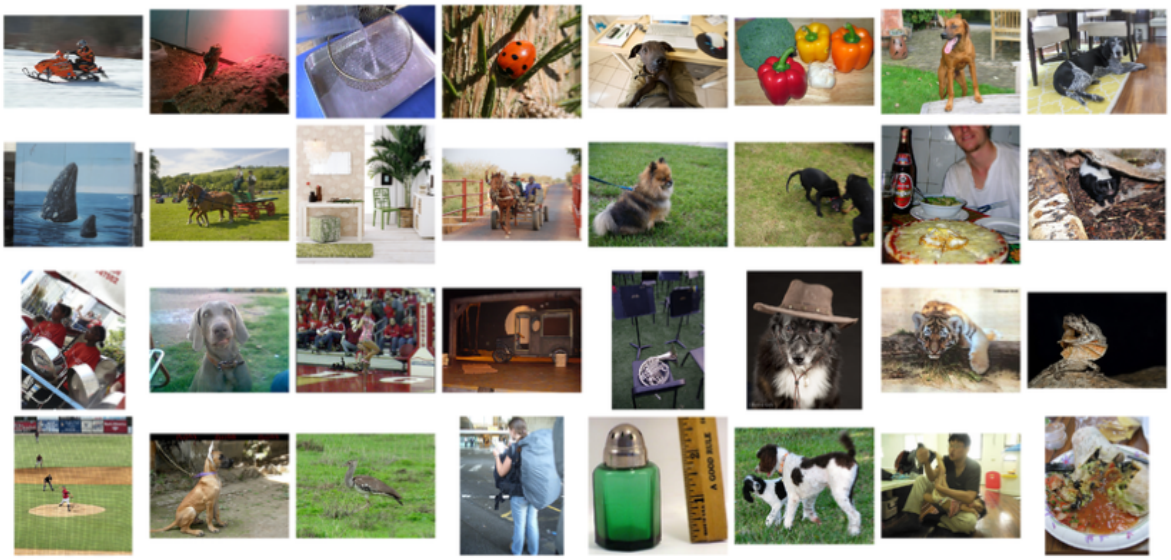
O desafio de classificação de imagens ILSVRC ocorre desde 2010 e tornou-se a referência padrão para reconhecimento e classificação de imagens em larga escala (11). Ele consiste de três tarefas: (*i*) classificação de imagens, (*ii*) localização de objeto único e (*iii*) detecção de objetos.

Destes, apenas a classificação de imagens está presente no desafio desde sua concepção e é a única que será tratada neste trabalho. Nela, é provido um banco de dados com mais de um milhão de imagens de treino rotuladas, e conjuntos menores de validação e teste, também rotulados, mas não divulgados aos participantes durante a competição. As imagens estão distribuídas em 1000 classes diferentes, variando de objetos naturais como diversas raças de cachorros e gatos, a objetos feitos pelo homem como automóveis e ferramentas. Uma amostra das imagens de diversas classes do conjunto de validação do desafio de 2012 é apresentado na Figura 3.

A tarefa de classificação tem por objetivo treinar modelos de ML (classificadores) que permitam classificar as imagens de entrada em uma determinada classe. A saída do classificador deve ser um vetor com a probabilidade da imagem entrada de pertencer a

cada uma das mil classes possíveis. Para comparar o rendimento dos classificadores é utilizado o erro top-5. Para esta métrica de erro, uma classificação é considerada válida se a classe correta estiver dentre as 5 maiores probabilidades atribuídas pelo classificador.

Figura 3 – Imagens do conjunto de validação do ILSVRC 2012.



Fonte: (11).

No transcurso dos últimos anos as soluções baseadas em DL dominaram o desafio ILSVRC, apresentando uma grande melhora da taxa de erro do desafio entre os anos 2012 e 2015. Em detalhe, foram:

- Em 2012, foi aplicada pela primeira vez uma CNN profunda. A rede vencedora, *AlexNet* (20), fez uso de 7 camadas (5 convolucionais e 2 totalmente conectadas) e a técnica de *dropout* (35) para regularizar . A AlexNet atingiu um erro top-5 de 16.4% no ILSVRC 2012. Comparado ao erro top-5 do segundo colocado daquele ano, isso são quase 10 pontos percentuais a menos o que significa uma diminuição relativa de 37.4%.
- Em 2013, o vencedor do desafio (19) usou uma estrutura semelhante à *AlexNet*, mas modificou o tamanho dos filtros convolucionais da primeira camada de 11×11 para 7×7 e diminui o *stride* de 4 para 2. Isso permitiu um erro top-5 de 11.7%. Os autores também notaram que retirar a camada totalmente conectada não aumenta o erro significativamente.
- Em 2014 a equipe VGG (25), segunda colocada no desafio, limitou-se ao uso das camadas convolucionais de filtros 3×3. Além disso aumentou o número de camadas para 19 (16 convolucionais e 3 totalmente conectadas).

- Com uma inspiração nas tendências dos anos anteriores, e outras novidades descritas nessa seção, a rede denominada *GoogLeNet* venceu o desafio de classificação do ILSVRC de 2014 com um erro top-5 de 6.7%, marginalmente melhor que o da equipe VGG, de 7.3%. A equipe do *Google*, entretanto mudou consideravelmente a estrutura da rede com diversas inovações que acima de tudo permitiram uma eficiência muito maior do uso de recursos computacionais. Em relação à rede VGG, a *GoogLeNet* obteve um resultado comparável, mas usando três vezes menos parâmetros. Comparada à *AlexNet*, são 12 vezes menos parâmetros e ao mesmo tempo aproximadamente metade do erro top-5. Isso se torna particularmente importante quando o objetivo é tornar as soluções de DL realizáveis em dispositivos embarcados.
- Vale notar ainda a rede *ResNet*, vencedora do ILSVRC 2015, que é uma arquitetura muito profunda (152 camadas), se limita a convoluções simples 3×3 e faz uso de conexões residuais. Nessas conexões residuais a entrada de uma camada é passada sem modificações adiante, e então apenas uma diferença sobre este valor é aprendido pelas camadas convolucionais e sua soma passada para a saída (36).

2.3.2 Arquitetura da *GoogLeNet*

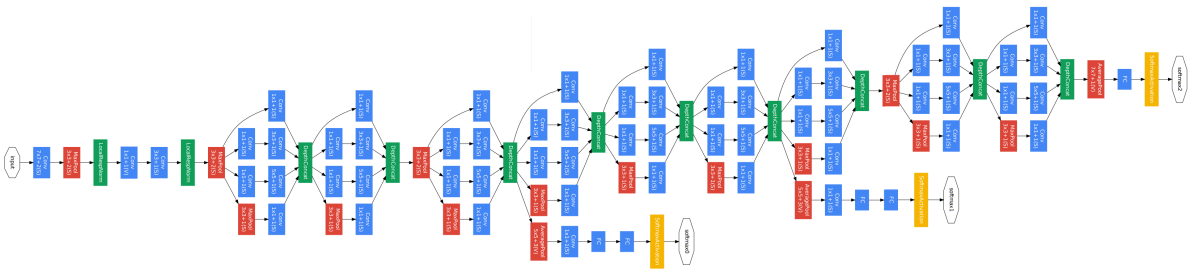
A arquitetura da rede *GoogLeNet* está dada na Figura 4. A rede pode ser dividida em três seções para descrever seus blocos:

1. **Rede de entrada**: é a parte da rede que vai até a primeira paralelização de caminhos e serve para adaptar a entrada para o corpo principal da rede. Seus blocos são:
   - Blocos azuis: operações de convolução.
   - Blocos vermelhos: *max pooling* para redução das dimensões espaciais.
   - Blocos verdes: regularizadores *Local Response Normalization*
2. **Corpo da rede**: é a parte central da rede, composta de vários blocos *Inception* (descritos na próxima seção) conectados em cascata. Seus blocos são:
   - Blocos azuis: operações de convolução.
   - Blocos vermelhos: *max pooling* sem redução de dimensão espacial.
   - Blocos verdes: concatenadores dos tensores provindos dos caminhos paralelos.
3. **Classificadores auxiliares**: são os dois ramos curtos que "brotam" do corpo da rede.

- Blocos azuis: o primeiro é uma convolução 1×1 e os dois seguintes são camadas totalmente conectadas.
- Blocos vermelhos: *max pooling* para redução das dimensão espaciais.
- Blocos amarelos: são blocos *softmax* para computação das funções de perda.

4. **Classificador final**: é a parte final da rede.
   - Bloco azul: camada totalmente conectadas.
   - Blocos vermelhos: *average pooling* de todos os neurônios de cada canal.
   - Blocos amarelos: são blocos *softmax* para computação da função de perda.

Figura 4 – Arquitetura da rede *GoogLeNet*

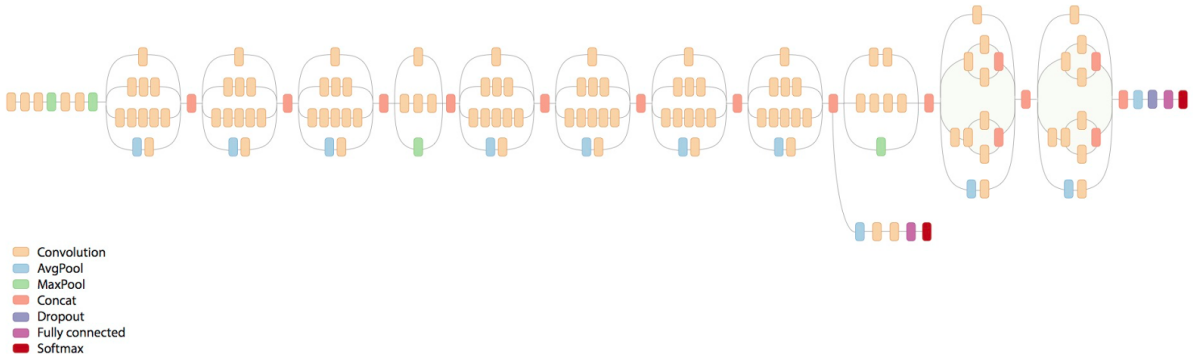


Fonte: (21)

Essa é a primeira rede divulgada que faz uso dos módulos que foram denomidados *Inception* pelos autores (21), e a vencedora do ILSVRC 2014. Algumas melhorias foram feitas a essa rede pelos mesmos autores, primariamente na composição dos blocos *Inception* e na profundidade da rede (37). Essas redes foram chamadas pela equipe de *Inception-v2* e *Inception-v3*. A arquitetura usada para este trabalho é a *Inception-v3*, para a qual foram disponibilizados os modelos já treinados ao público. Após isso foram divulgadas mais melhorias (38), as quais foram chamadas *Inception-v4* e *Inception-ResNet*, levando em conta as novidades do ILSVRC 2015 (36).

Na Tabela 2 está descrita a estrutura da rede *Inception-v3*. Ela é ligeiramente maior que a *GoogLeNet* e uma representação gráfica dela é dada na Figura 5. Entretanto, a representação tabular é mais completa. Ela referencia as imagens dos módulos individuais quando necessário para compreensão.

Figura 5 – Arquitetura da rede *Inception-v3*



Fonte: <https://research.googleblog.com/2016/03/train-your-own-image-classifier-with.html>

Tabela 2 – Composição da arquitetura *Inception-v3*

Tipo	**Tamanho do filtro/*stride*** ou comentário	**Tamanho da entrada**
convolução	3×3 / 2	$299 \times 299 \times 3$
convolução	3×3 / 1	$149 \times 149 \times 32$
convolução com *padding*	3×3 / 1	$147 \times 147 \times 64$
max pooling	3×3 / 2	$147 \times 147 \times 64$
convolução	3×3 / 1	$73 \times 73 \times 64$
convolução	3×3 / 2	$71 \times 71 \times 80$
convolução	3×3 / 1	$35 \times 35 \times 288$
$3\times$*Inception*	Como na Figura 6b	$35 \times 35 \times 288$
$5\times$*Inception*	Como na Figura 6c	$17 \times 17 \times 768$
$2\times$*Inception*	Como na Figura 6d	$8 \times 8 \times 1280$
average pooling	8×8	$8 \times 8 \times 2048$
linear	*logits*	$1 \times 1 \times 2048$
softmax	classificador	$1 \times 1 \times 1000$

Fonte – (37)

2.3.3 Módulo *Inception*

O módulo *Inception* pode ser considerado o bloco padrão usado para construir a rede *GoogLeNet*, ao ponto de que a rede inteira também é conhecida por esse nome. O módulo *Inception* é inspirado no conceito *Network in Network* - NIN (39) (tradução livre, Rede dentro de Rede). Segundo (39), os filtros convolucionais, por serem modelos linearizados gerais, não tem capacidade de abstração suficiente para algumas tarefas altamente não lineares, e portanto sugerem substituir os filtros simples por "micro redes", com maior poder de abstração e não-linearidade. Essas "micro redes" seriam então concatenadas para formar uma rede grande. Em (39) é proposto usar um Perceptron Multicamadas

(*Multilayer Perceptron* - MLP) com suas camadas conectadas por ativações não lineares. Assim, a rede total seria composta de redes menores.

A "micro rede" construída pela equipe do Google é o módulo *Inception* mostrado na Figura 6a. Pode-se observar quatro blocos principais:

- **O bloco *Base***. Corresponde à entrada ao módulo *Inception*. Esta composta dos valores das saídas das funções de ativação da camada anterior (ou do módulo anterior). Esses valores são as entradas para os diferentes blocos convolucionais do módulo.
- **Os blocos convolucionais**. São filtros convolucionais, de tamanho $n \times m$, onde n e m representam as dimensões espaciais de cada filtro convolucional. Cada filtro convolucional vem seguido de uma função de ativação do tipo ReLU.
- **O bloco *Filter Concat***. É o responsável por juntar as saídas provindos dos caminhos paralelos internos do módulo, fazendo uma operação de concatenação das saídas na dimensão dos canais.
- **O bloco *Pool***. Realiza um *max pooling* sem reduzir a dimensão espacial da entrada.

A seguir, serão explicados os conceitos chaves do modulo *Inception*: (*i*) os caminhos paralelos; (*ii*) a composição das operações de convolução; (*iii*) os filtros de convolução 1×1; (*iv*) o uso de classificadores auxiliares.

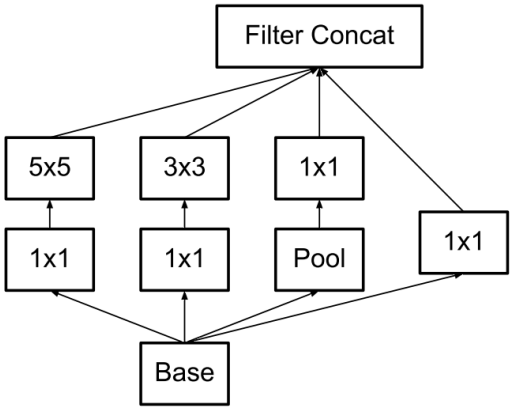
2.3.4 Caminhos paralelos

Como se pode ver na Figura 6a, a entrada que é recebida pelo bloco *Base*, serve de entrada para os primeiros blocos convolucionais via os caminhos paralelos. Isso permite que o que seja passado adiante não sejam apenas características diferentes como numa convolução simples, onde cada filtro aprende uma característica específica. Cada caminho paralelo permite extrair características de tipos diferentes. A convolução 5×5 extrai características em uma escala e complexidade diferentes da convolução 3×3; o bloco de *max pooling* produz um efeito completamente diferente, ampliando na prática o tamanho ocupado pelas ativações de maior valor e tornando o bloco mais invariante a deslocamentos; e a convolução 1×1 tem efeito duplo explicado na Seção 2.3.6.

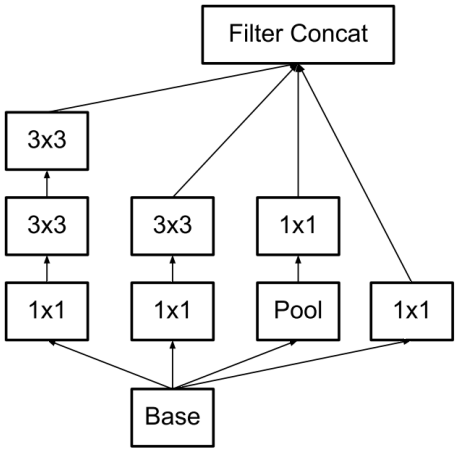
A parte central da arquitetura da *GoogleNet* consiste de dois módulos *Inception* ligados em série diretamente uns aos outros, aos quais segue então um bloco *max pooling* para

Figura 6 – Módulos *Inception*

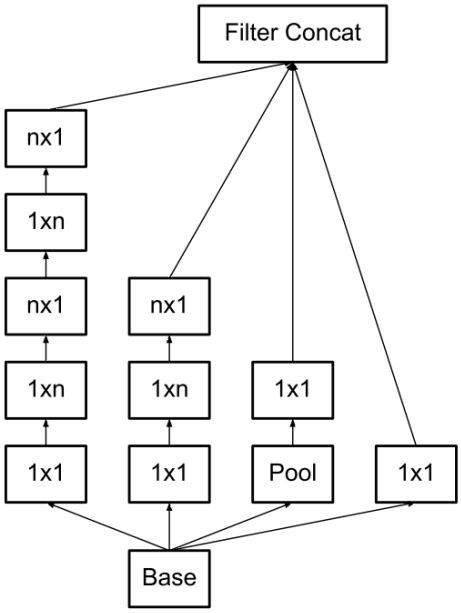
(a) Primeira versão do módulo *Inception*, usada na GoogLeNet



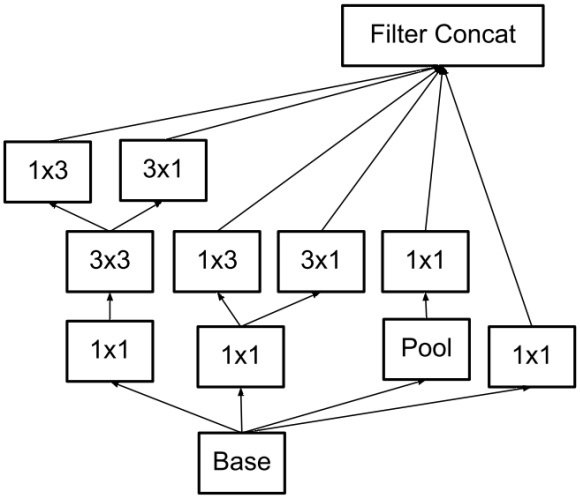
(b) Um dos módulos melhorados usados na *Inception-v3*



(c) Um dos módulos melhorados usados na *Inception-v3*



(d) Um dos módulos melhorados usados na *Inception-v3*



Fonte: (37).

reduzir cada dimensão espacial pela metade. Após isso, repete-se o padrão mais duas vezes, sendo a segunda vez com cinco módulos e a terceira com dois.

Supondo-se que as convoluções conservam o número de canais, teria-se após a concatenação dos quatro caminhos paralelos quatro vezes tantos canais quanto na entrada. Para as convoluções é possível diminuir o número de canais na saída, mas isso não é possível para a operação de *max pooling*, que conserva o número de canais. Teria-se assim no mínimo o mesmo número de canais que na entrada, se não houvesse nenhum canal de convolução.

Entretanto, a convolução é evidentemente a operação mais importante nas CNNs e assim torna-se necessário reduzir de alguma forma a dimensionalidade dos canais para se manter um limite de memória e tempo computacional aceitáveis.

2.3.5 Fatorização de convoluções

Numa camada convolucional, cada peso corresponde a uma multiplicação por ativação (37). Isso torna as convoluções com filtros de grande tamanho custosas computacionalmente. Para um filtro cujos lados são de igual valor, um quadrado, um aumento linear em seus lados acarreta num aumento quadrático do número de operações.

Para reduzir as computações necessárias, usa-se um método para fatorizar as convoluções com filtros grandes em duas ou mais convoluções com filtros menores, ligados em cascata. Uma convolução 5×5 tem o mesmo campo receptivo de duas convoluções 3×3 ligadas em série, como ilustrado na Figura 7a. Entretanto, tem apenas $2 \times 3^2 = 18$ pesos contra os $5^2 = 25$ do filtro maior. Isso acarreta num custo computacional de apenas uma fração $(18/25 = 0.72)$ do original. A mesma lógica pode ser aplicada para convoluções com filtros 7×7 com três convoluções 3×3 concatenadas se obtendo uma redução de 45% $(= 3 \times 3^2/7^2 = 0.55)$. Esse tipo de fatorização é observado em um dos módulo melhorados do *Inception-v3*, como na Figura 6a.

Entretanto, outra possibilidade é fatorizar as convoluções de filtros quadrados em filtros assimétricos $1 \times n$ e $n \times 1$, de forma que o custo computacional de um filtro retangular seja proporcional à soma das dimensões de seus lados. Esse tipo de fatorização é mostrada na Figura 7b e é usado nos módulos melhorados das Figuras 6c e 6d. Note que para um filtro quadrado, de dimensões espaciais $n \times n$, o número de parâmetros sofre uma redução de n^2 para $2n$. No caso de uma convolução 7×7, isso é uma redução por um fator de $7^2/(7 + 7) = 0.29$, menos de um terço do original.

2.3.6 Filtro Convolucional 1 x 1

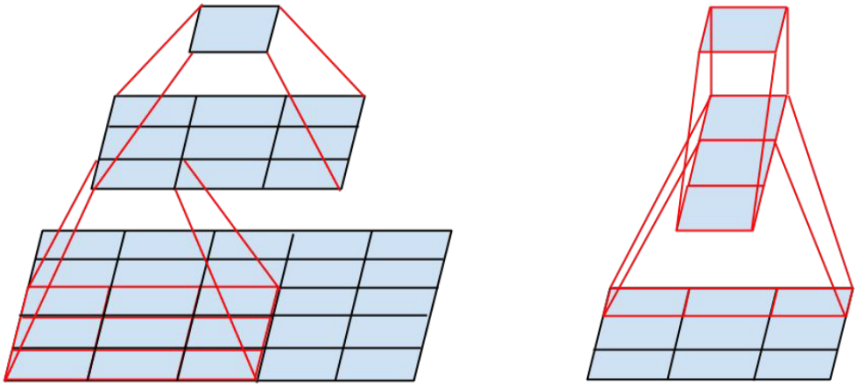
Os filtros convolucionais 1×1 no modulo *inception* tem dois propósitos principais:

- Servem como uma camada a mais após a qual pode ser adicionada uma não linearidade, fazendo um papel semelhante ao atribuído ao MLP nos NINs. Dessa forma, eles aumentam o poder representacional do modelo (39).

Figura 7 – Fatorização de convoluções

(a) Uma convolução 5×5 fatorizada em duas 3×3.

(b) Uma convolução 3×3 fatorizada em duas assimétricas



Fonte: (37).

- As operações de convolução 1×1 permitem reduzir o número de canais ou filtros que serão passados adiante, minimizando substantivamente o número de computações nas camadas que o seguem.

Considerando duas operações de convolução idênticas, ligadas em série, um aumento linear no número de filtros acarretaria num aumento quadrático do tempo de computação. Assim, como a ideia central dessa rede, baseado no princípio NIN, é precisamente ligar vários blocos de mesma estrutura (módulo *Inception*) em série, assim como uma CNN normal tem convoluções ligadas em série, a redução de dimensionalidade provida pelas convoluções 1×1 se torna essencial.

O efeito efetivo de uma convolução 1×1 é diferente das convoluções com filtros de dimensões espaciais maiores. Por analisar apenas um pixel por canal, esse filtro não extrai informação espacial. Entretanto, analisa todos os canais, e atua como um extrator das informações mais importantes. Mais explicativo seria denominá-lo um filtro $1 \times 1 \times C_{filtro}$, onde C_{filtro} representa o número de canais de entrada. Seu efeito matemático é equivalente a uma soma ponderada de todos os canais de entrada, para cada filtro 1×1. Alternativamente, a operação pode ser vista como uma projeção do espaço de características para um espaço de dimensões menores (21) pois sua saída (antes da função de ativação) é uma combinação linear das entradas ao nível dos canais.

Um exemplo extremamente simples é o de transformar uma imagem RGB em uma de escala de cinza. Essa operação pode ser vista como uma convolução 1×1 com cada um dos

três pesos (um para cada canal) fixado em valor de $\frac{1}{3}$. Dessa forma obtém-se a combinação das intensidades de cor de cada um dos três canais de entrada em uma única imagem com um canal de intensidade representando a escala de cinza da imagem original.

Na etapa de treinamento esse filtro aprende os pesos para extrair e fazer uma composição dos canais mais importantes para o filtro que o segue. O único motivo que faz o filtro 1×1 divergir da função de soma ponderada dos canais é a ativação não linear que sempre o segue no módulo *Inception*.

2.3.7 Classificadores Auxiliares

O problema de *vanishing gradient* (tradução livre, desvanecimento do gradiente), é um problema que atrasou o desenvolvimento de redes neurais por muitos anos (40). Ele acontece quando o gradiente propagado para as camadas inferiores durante a etapa de treinamento se torna tão pequeno a ponto de a rede não conseguir ser treinada de maneira efetiva. Um dos motivos que ele acontece é quando o gradiente é passado por sucessivas funções de ativação com valores cujos gradientes têm valores absolutos menores que um. Pela regra da cadeia usada no algoritmo *backpropagation* esses valores são multiplicados, resultando em um valor tanto menor quanto maior seja o número de camadas que ele tenha passado.

Esse problema foi parcialmente contornado com o uso de funções de ativação ReLU, cuja derivada é sempre unitária ou zero. Entretanto, para redes muito profundas o problema parece persistir (36).

Uma forma de fazer os gradientes chegarem à base da rede é com o uso de classificadores auxiliares. Esses classificadores são semelhantes ao classificador normal e estão conectados a saídas de alguns módulos *Inception* na parte intermediária da rede, em vez de no final. Eles aprendem a classificar as imagens com as características aprendidas pelos módulos aos que estão conectados, apenas com o intuito de se poder calcular uma função de perda sobre a qual se possa calcular os gradientes para os pesos. Efetivamente, funcionam como "injetores de gradiente" na rede, ajudando a treinar as camadas mais baixas e mitigando o problema de *vanishing gradient*.

No caso da rede *GoogLeNet*, os gradientes foram usados com um peso menor (0.3) em relação aos do classificador principal. Os classificadores auxiliares são usados somente no treino e são eliminados da rede na hora de fazer a inferência.

2.4 *Transfer Learning*

Transfer learning (tradução livre, aprendizado transferido) é uma técnica de ML que consiste em usar o conhecimento aprendido ao resolver um problema, denominado tarefa fonte, para resolver um problema diferente, mas relacionado, denominado tarefa destino (41).

Com a reutilização da informação aprendida na tarefa fonte, é possível economizar recursos na etapa de treinamento, pois os parâmetros aprendidos na tarefa fonte se aproximam dos parâmetros ideais da tarefa destino.

Pode-se fazer uma analogia interessante: suponha duas pessoas que querem aprender a tocar violão. Uma já tem considerável habilidade no piano, enquanto a outra nunca teve contato com instrumentos musicais. A primeira pessoa terá uma considerável vantagem ao aprender a tocar o violão, já que habilidades como leitura da notação musical e a destreza nos dedos adquiridas no aprendizado de tocar o piano (tarefa fonte), são transferidas e usadas de forma benéfica na aprendizagem de tocar o violão (tarefa destino) (41).

O uso de *transfer learning* permite reduzir drasticamente o tempo na etapa de treinamento para obter um resultado satisfatório. Treinar algumas redes muito profundas em tarefas complexas pode chegar a levar semanas ou meses para convergir, mesmo em dezenas de processadores especializados para esses tipos de processamento (42). Um outro caso em que *transfer learning* é de utilidade, ou mesmo necessário, é quando não é possível obter um banco de dados rotulado de grande tamanho para uma tarefa específica. Isso pode acontecer por diferentes causas como os dados serem raros, caros de coletar e rotular, ou simplesmente inacessíveis (41). Uma rede complexa, necessária para obter resultados satisfatórios em tarefas também complexas, aliada a um banco de dados pequeno causa um *overfitting* (tradução livre, sobre ajuste). Nesses casos, a rede aprende muito bem o conjunto de treinamento, e se especializa nele tanto a ponto de não generalizar bem com exemplos fora deste conjunto. Repositórios de *big data* estão se tornando prevalentes com uma série de banco de dados disponíveis sem custo na internet. Usar bancos de dados que são relacionados, porém não idênticos a uma tarefa objetivo torna soluções por *transfer learning* uma abordagem atrativa (41).

2.4.1 Definição matemática

Para obter uma noção mais exata do que consiste *transfer learning*, uma definição matemática da técnica e de seus componentes é dada a seguir.

Um domínio $\mathcal{D}$ consiste de dois componentes:

- um espaço de características $\mathcal{X}$;
- uma distribuição de probabilidade marginal $P(\mathbf{x})$, onde $\mathbf{x} = \{x_1, ..., x_n\} \in \mathcal{X}$.

Agora, considerando um domínio específico $\mathcal{D}$, uma tarefa $\mathcal{T}$ será definida por duas componentes:

- um espaço de rótulos $\mathcal{Y}$;
- uma função preditiva objetivo $f(\bullet)$, que não é observada, mas pode ser aprendida pelos dados de treinamento;
- O conjunto de treinamento consiste em um conjunto de pares $\{\mathbf{x}_k, \mathbf{y}_k\}$, onde $\mathbf{x}_k \in \mathcal{X}$ e $\mathbf{y}_k \in \mathcal{Y}$.

A função $f(\bullet)$ pode ser usada para predizer o rótulo correspondente $f(\mathbf{x})$ de uma nova instância $\mathbf{x} \in \mathcal{X}$. De um ponto de vista probabilístico $f(\mathbf{x})$ pode ser interpretada como $P(\mathbf{y}|\mathbf{x})$.

Para um melhor entendimento da notação definida acima, suponha-se o problema de classificação de imagens, então:

- Em relação ao domínio $\mathcal{D}$: $\mathcal{X}$ é o espaço de todas as imagens possíveis; $\mathbf{x}$ é uma imagem particular; x_i é o i-ésimo vetor ou tensor de características correspondente à i-ésima imagem; n é o número de pixels ou vetores de características em $\mathbf{x}$.
- Em relação à tarefa $\mathcal{T}$: $\mathcal{Y}$ é o conjunto de rótulos e nesse caso contém os rótulos pelos quais devem ser classificadas as imagens e $\mathbf{y}_k$ assume o valor de um desses rótulos.

Pelas definições acima, temos: um domínio $\mathcal{D} = \{\mathcal{X}, P(\mathbf{x})\}$ e uma tarefa $\mathcal{T} = \{\mathcal{Y}, f(\bullet)\}$. Ao considerar uma tarefa fonte e destino, tem-se: O *conjunto de dados do domínio fonte* é definido como o conjunto de pares $\{(\mathbf{x}_{S_1}, \mathbf{y}_{S_1}), ..., (\mathbf{x}_{S_{n_S}}, \mathbf{y}_{S_{n_S}})\}$, onde $\mathbf{x}_{S_k} \in \mathcal{X}_S$ e $\mathbf{y}_{S_k} \in \mathcal{Y}_S$; o *conjunto de dados do domínio destino* é definido como o conjunto de pares $\{(\mathbf{x}_{T_1}, \mathbf{y}_{T_1}), ..., (\mathbf{x}_{T_{n_T}}, \mathbf{y}_{T_{n_T}})\}$, onde, $\mathbf{x}_{T_k} \in \mathcal{X}_T$ e $\mathbf{y}_{T_k} \in \mathcal{Y}_T$. Como é indicado em (43), na maioria dos casos .

$$0 \leq n_T \ll n_S.$$

Definição 2.1 (*Transfer Learning*)**.** Seja: um domínio fonte $\mathcal{D}_S$ e a tarefa a aprender $\mathcal{T}_S$; um domínio destino $\mathcal{D}_T$ e tarefa a aprender $\mathcal{T}_T$. Então, o objetivo de *Transfer learning* é ajudar no aprendizado da função preditiva $f_T(\bullet)$ em $\mathcal{D}_T$ usando o conhecimento em $\mathcal{D}_S$ e $\mathcal{T}_S$, sendo $\mathcal{D}_S \neq \mathcal{D}_T$ e/ou $\mathcal{T}_S \neq \mathcal{T}_T$. —

2.4.2 Transfer Learning em CNNs

Nessa seção são dados os passos necessários para realizar um *transfer learning* em uma rede neural CNN.

- O primeiro passo consiste em escolher uma rede já treinada para uma tarefa semelhante à tarefa destino. Pela escolha da tarefa semelhante, essa rede terá uma estrutura (definição do grafo computacional) razoavelmente adequada para resolver a tarefa destino.
- O segundo passo é aproveitar a informação aprendida no processo de treinamento da rede na tarefa fonte. Essa informação está contida no grafo fonte na forma dos pesos da rede.

O grafo pode ser aproveitado quase integralmente, ou ter algumas de suas camadas mais altas (mais próximas da saída) retiradas. Em cima das camadas já aproveitadas, pode-se inserir camadas adicionais em número adequado para a tarefa destino. Mas, comumente é trocada a camada de classificação caso a tarefa seja diferente. Isso ocorre tanto quando as tarefas, os domínios ou ambos diferem para fonte e destino.

Uma orientação geral é: quanto mais dissimilares forem as tarefas fonte e destino, menos camadas da rede fonte devem ser usadas. Isso se deve à seguinte observação, as camadas baixas aprendem características gerais e camadas mais altas aprendem características mais específicas da tarefa do domínio em questão. Para o caso da tarefa de classificação de imagens, as primeiras camadas da rede apresentam padrões quase que universais na

forma de filtros de Gabor e filtros *blobs* (44). O uso de técnicas de *feature vizualization* demonstraram que camadas superiores estão relacionadas com padrões mais complexos e que nas intermediárias acontece uma transição gradual entre os padrões básicos e os de alto nível (19) (45).

Quanto aos pesos das camadas que serão aproveitadas, tem-se duas opções: (*i*) pode-se evitar que sejam modificados na etapa de treinamento, comumente referido em DL como congelamento dos pesos da rede ou (*ii*) fazer uma sintonia fina para adaptá-los para a tarefa destino. Em ambos os casos é necessário treinar as novas camadas, sendo que estas são inicializadas com valores constantes ou aleatórios, assim como é feito ao se treinar uma rede do princípio. A diferença é que estas novas camadas terão como entrada as características extraídas pelas camadas aproveitadas da rede fonte em vez dos dados de entrada originais. Embora pareça intuitivo sempre fazer a sintonia fina, isso pode gerar problemas quando a rede está sobredimensionada para um banco de dados de destino pequeno. Assim, neste caso, não modificar as camadas baixas funciona como uma regularização sem custo algum.

Quando é optado por "congelar" algumas camadas, existem dois fatos que possibilitam uma diminuição significativa do tempo de treinamento.

- O primeiro fato é que as ativações das camadas congeladas são constantes ao longo do treino para uma mesmo dado de entrada, ao contrário das camadas que estão sendo treinadas, cujas ativações mudam a cada passo de otimização, quando seus pesos são alterados. Dessa forma, essas ativações podem ser calculadas apenas uma vez antes de começar o treino, e seus valores são salvos em arquivos que são informalmente chamados de *bottlenecks* (tradução livre, gargalos). O nome referencia a última camada da rede, onde a representação da informação é muito mais compacta e não tem relação alguma com o uso do termo para designar a etapa mais lenta de uma linha de produção. Numa iteração da etapa de treinamento, a primeira camada a ser treinada é então alimentada com os valores salvos nos *bottlenecks*.
- O segundo fato é que como as camadas iniciais não serão treinadas, não é necessário calcular os gradientes para elas, e o algoritmo *backpropagation* pode efetivamente parar na primeira camada treinável.

Esses dois fatos em conjunto permitem que com um tempo de treinamento mínimo, obtenha-se um resultado para o qual normalmente seria necessário um tempo de algumas ordens de grandeza maior, considerando uma estrutura de rede de várias camadas de profundidade.

3 SOLUÇÃO PROPOSTA

3.1 Introdução

Neste capítulo é apresentado o problema específico a solucionar, o sistema proposto para a solução do problema em estudo e os resultados experimentais obtidos na avaliação quantitativa, usando um banco de dados especializado. Sendo assim, o capítulo divide-se em três seções além desta introdução. Na segunda seção, realiza-se uma exposição do desafio ISIC, o banco de dados disponibilizado no desafio para as duas tarefas de classificação binária de imagens de lesões de pele. Na terceira seção, é descrito o sistema de classificação *end-to-end* proposto baseado no uso de uma rede CNN e a técnica de *transfer learning*, além de uma descrição do ambiente de desenvolvimento e da métrica de validação usada. Na última seção, é feita uma exposição dos resultados obtidos e finalmente uma comparação com outros participantes do desafio.

3.2 Desafio ISIC

O desafio no ISIC 2017 *Skin Lesion Analysis Towards Melanoma Detection* (46) (tradução livre, análise de lesões de pele para detecção de melanomas), foi proposto pela ISIC no simpósio ISBI com o objetivo de apoiar o desenvolvimento de algoritmos de diagnóstico automático de melanoma.

O desafio ISIC 2017 provê um banco de dados com imagens de lesões de pele rotuladas, e propõe três problemas a serem solucionados pelos participantes: (*i*) Segmentação de imagens; (*ii*) Extração de características de imagens de dermoscopia; (*iii*) Classificação de lesões.

Neste trabalho, foi proposta uma solução apenas para o terceiro problema, a classificação de lesões de pele. O problema consiste em duas tarefas de classificação binária independentes, que envolvem três diagnósticos de lesão de pele exclusivos (apenas uma ocorrência por imagem). As lesões são:

- **Melanomas**, tumores de pele malignos, (melanocíticos) (ver Figura 8).

- **Ceratoses Seborréicas**, tumores de pele benignos, (não melanocíticos) (ver Figura 9).
- **Nevos**, tumores de pele benignos, também conhecidos coloquialmente como pintas (melanocíticos) (ver Figura 10).

As duas tarefas binárias a serem realizadas são as de identificação positiva de melanomas, e ceratoses seborréicas, considerando-se as duas outras classes como exemplos negativos em cada tarefa. Explicitamente, as tarefas são a classificação binária entre as seguintes divisões de classes:

1. Melanoma Versus Nevo e Ceratose Seborréica;
2. Ceratose Seborréica Versus Melanoma e Nevo.

Quando começou-se a trabalhar no problema citado, o desafio já havia sido concluído e portanto não foi possível competir com os outros grupos participantes. Entretanto, como todos os resultados já foram divulgados, é factível comparar o resultado obtido neste trabalho com o resultado final oficial do desafio.

3.2.1 Banco de dados

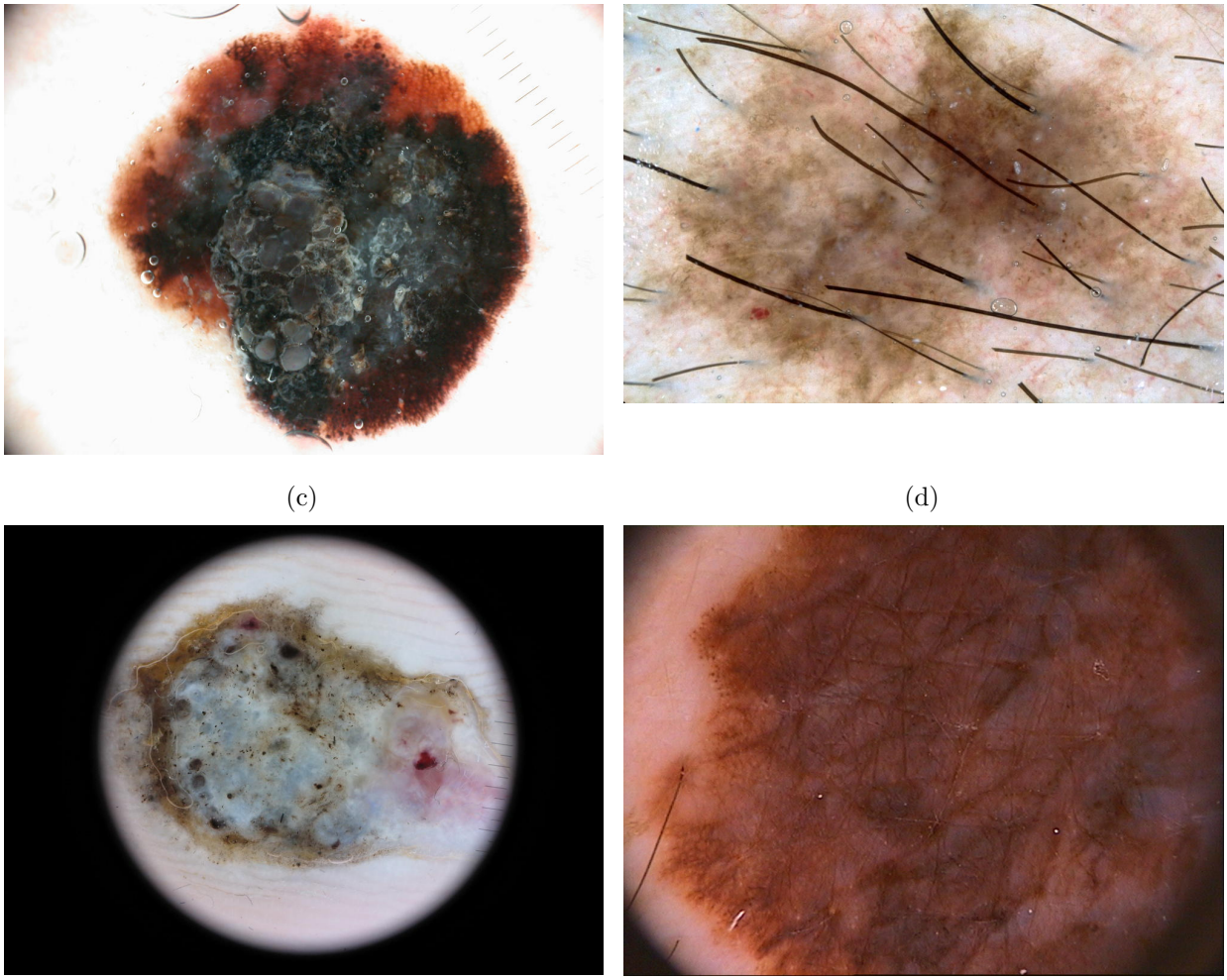
O banco de dados do desafio 2017 consiste de 2000 imagens para a etapa de treinamento, 150 imagens para a etapa de validação e 600 imagens para a etapa de teste. Todas as imagens vêm em formato JPEG em diferentes tamanhos. Exemplos de imagens do banco de dados são apresentados nas Figuras 8, 9 e 10. A primeira imagem de cada tipo foi escolhida para demonstrar um exemplo típico do conjunto, enquanto as outras três foram escolhidas para demonstrar a diversidade do intraclasse do conjunto.

Além das imagens de treino, validação e teste, também são disponibilizados três arquivos em formato CSV contendo os rótulos das imagens (*ground truth*), um arquivo CSV para cada divisão do banco de dados.

Enquanto o desafio estava aberto, o desafio disponibilizava todas as imagens para *download*, mas os rótulos de validação e de teste não eram disponibilizados. Para realizar as submissões de teste e validação, os participantes deveriam mandar um arquivo no mesmo formato que o arquivo do *ground truth* contendo a probabilidade estimada de cada imagem ser da

Figura 8 – Imagens do conjunto de treino de melanoma.

(a) (b)



(c) (d)

Fonte: Desafio ISIC 2017.

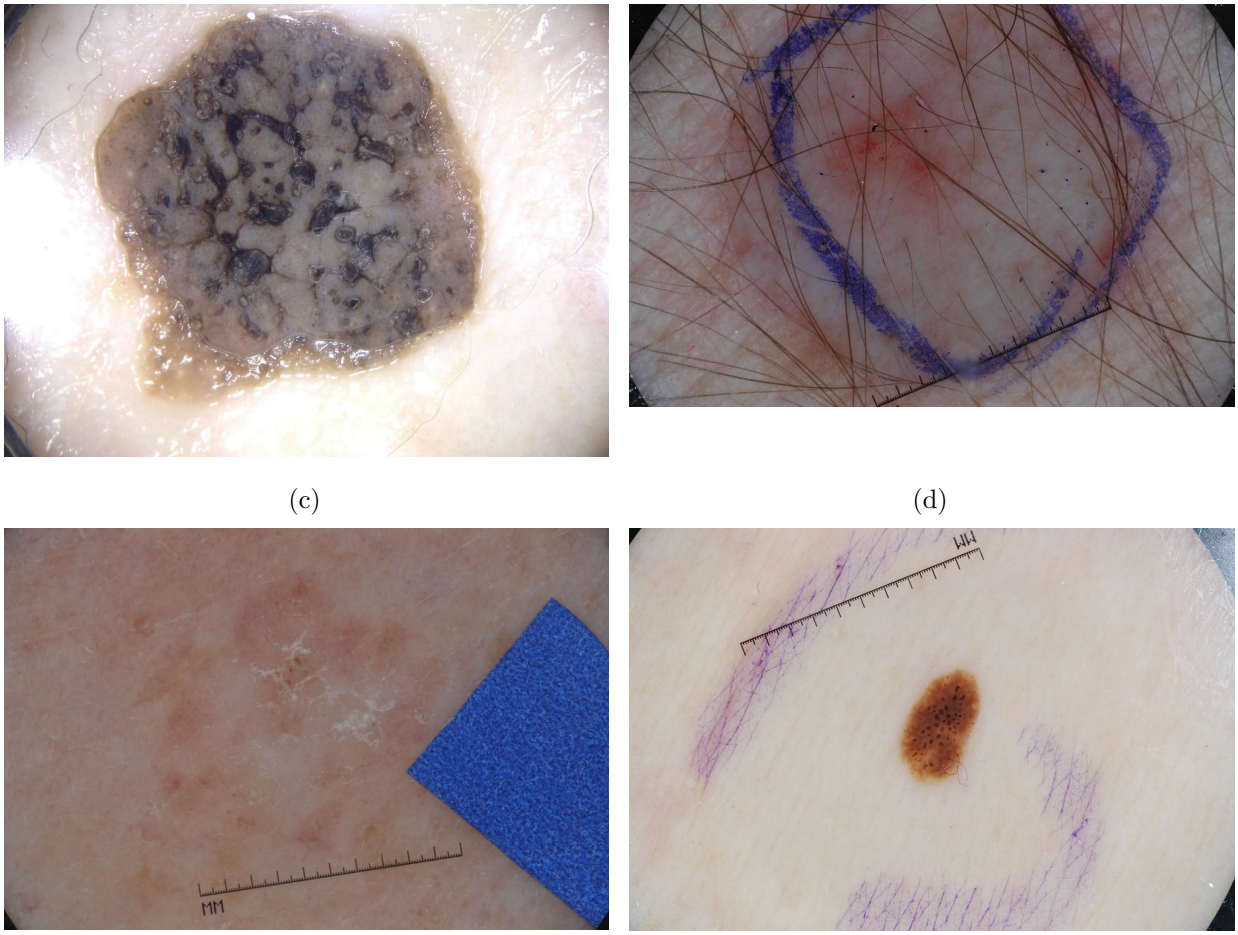
classe alvo para cada uma das duas tarefas. O resultado das submissões de validação eram divulgados imediatamente, sem um limite de tentativas. Para o conjunto de teste, somente a última submissão foi considerada e divulgada, e apenas após a conclusão do desafio. Podiam ser feitas múltiplas submissões, mas apenas a última submissão era considerada válida e divulgada.

Das imagens do banco de dados para a etapa de treinamento, 374 são de melanomas, 254 são ceratose, e 1372 são nevos, caracterízando um *dataset* desbalanceado. As distribuições relativas interclasse são semelhantemente desbalanceadas para o conjunto de validação (42, 39 e 78) e para o de teste (117, 90 e 393).

Ao observar as imagens do conjunto de treino, pode-se notar algumas características que ocorrem de forma frequente:

Figura 9 – Imagens do conjunto de treino de ceratose seborréica.

(a) (b)



(c) (d)

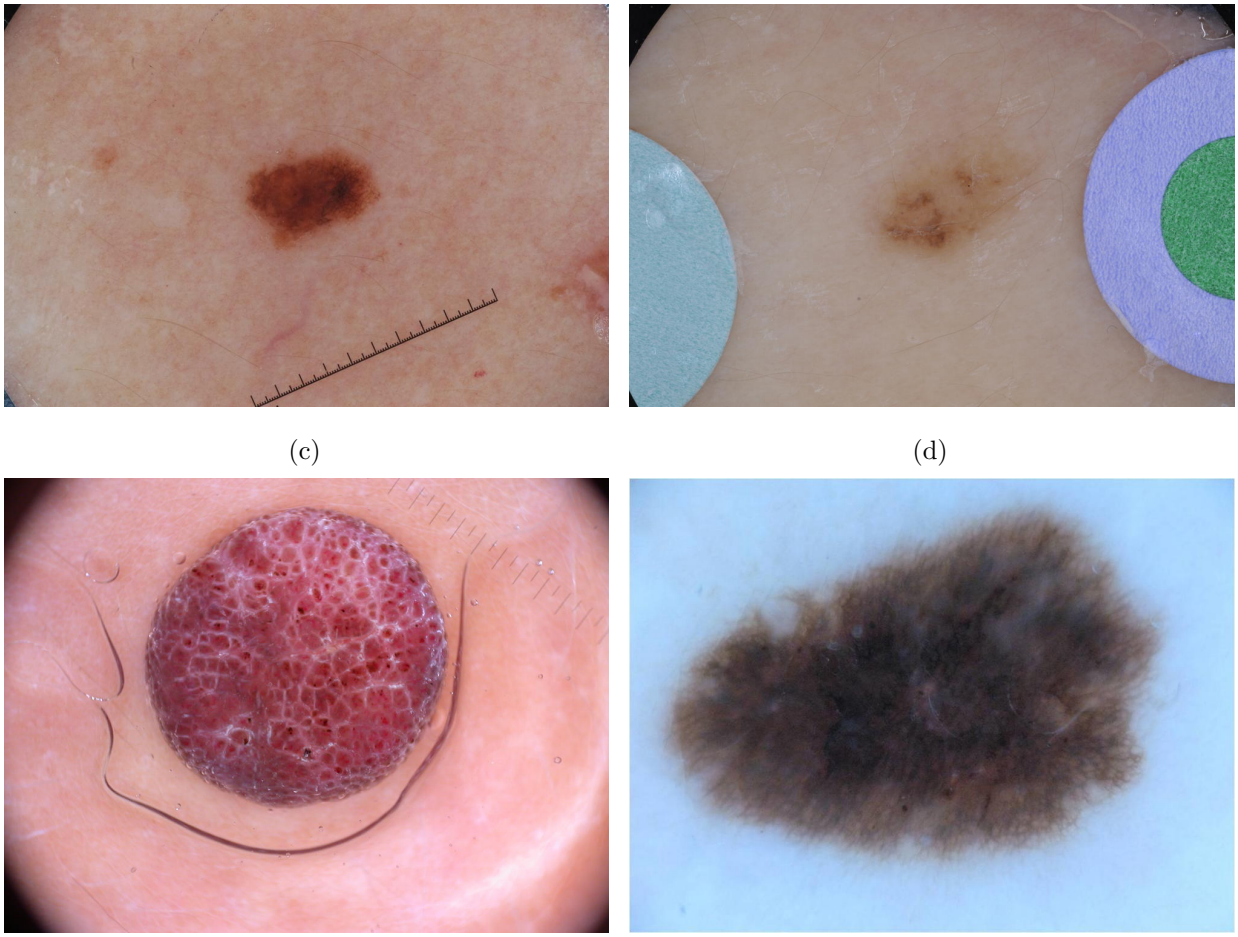
Fonte: Desafio ISIC 2017.

- Semelhança interclasse de imagens (9d e 10a).
- Anotações sobre a pele (9b e 9d).
- Contorno preto nas imagens (em todas, com exceção de 8b e 10d; especialmente pronunciado em 8c).
- Pelos em grande quantidades (8b e 9b).
- Adesivos coloridos sobre a pele (9c e 10b).
- Réguas para mostrar escala (8a, 9c, 9d, 10a e 10c)
- Óleo transparente usada em algumas dermoscopias (8b e 10c)

Se houver um desbalanço dessas características em um nível interclasse (e.g. se houver muito mais réguas em imagens de melanoma do que em nevos ou ceratoses), o classificador

Figura 10 – Imagens do conjunto de treino de nevo.

(a) (b)



(c) (d)

Fonte: Desafio ISIC 2017.

pode aprender a classificar essas características, em vez de classificar a lesão em si. Dessa forma, o classificador apresentaria melhores resultados na fase de treino, porém não generalizaria bem (47).

3.3 Sistema de Classificação Proposto

Aqui é apresentado o ambiente em que ocorreu o desenvolvimento e a métrica de validação e comparação usada, para então expor o sistema de classificação desenvolvido. A solução aqui proposta para os problema de classificação binária de lesões de pele está baseada em duas etapas:

- **Etapa de Pré-processamento da imagem**. Com o intuito de minimizar a variabilidade entre as características de captura das imagens, são aplicados algoritmos de

constância de cores e realce de bordas.

- **Etapa de extração de características e classificação**. Aqui é aplicada a técnica de *transfer learning* usando a rede *GoogLeNet*. Algumas variações são feitas sobre a rede e sobre os hiper-parâmetros que a definem.

3.3.1 O Ambiente de Desenvolvimento

Toda a parte de programação foi desenvolvida na linguagem de programação *Python*, na versão 3.7. A seleção da linguagem foi feita tomando em conta as facilidades de implementação por ela proporcionadas, assim como a disponibilidade de diversas bibliotecas para aplicações de ML e DL.

Dentre as diversas bibliotecas usadas, destaca-se *TensorFlow* (versão 1.3.0). *TensorFlow* foi escolhida por sua extensa documentação, ter apoio e desenvolvimento da *Google*, suporte para computação com GPUs, e ser atualmente uma das mais populares bibliotecas para DL, com diversos guias e tutoriais disponíveis na internet (48).

O desenvolvimento aconteceu primariamente em um *notebook* do autor, rodando o sistema operacional *Ubuntu Linux* (versão 16.04 LTS). O computador está equipado com um processador *Intel CORE i5-5200U*, 8 GB DDR3 de memória RAM, e um GPU *Nvidia GeForce 930m*. Na GPU foram executadas a maior parte das operações. As especificações da placa são as seguintes:

- ***Pipelines*** 384;
- **velocidade de *clock***: 941 MHz;
- **memória**: 2 GB DDR3;
- **poder de precessamento**: 713 $GFLOPS$.

Apesar de ser uma placa de nível médio baixo (está classificada em oitava de nove de sua geração pelo fabricante) e já ter mais de dois anos, conseguiu realizar todas as computações necessárias em tempo hábil, com nenhuma instância de treino durando mais que duas horas. Isso se deve a três fatores:

- A arquitetura *Inception-v3* utilizada é altamente eficiente em comparação com outras de desempenho semelhante;

- A técnica de *transfer learning* possibilitou utilizar a informação aprendida para outra tarefa, e assim poupar muito tempo de treinamento.
- O uso dos *bottlenecks*, que apesar de levar muito tempo para computá-los e salvá-los, uma vez obtidos, possibilizaram um tempo de treinamento equivalente ao de uma rede rasa.

3.3.2 Métrica de Validação

Em problemas de ML, especificamente em problemas de classificação é muito comum usar a acurácia preditiva como a métrica para comparar o desempenho de diferentes classificadores, ou até mesmo para medir a evolução de seu desempenho ao longo do treino. Ela é definida como a taxa de classificações corretas em dados não vistos antes.

Entretanto, em muitos casos a acurácia preditiva se torna inadequada para capturar o desempenho dos classificadores. Tome como exemplo o próprio caso tratado neste trabalho. A distribuição das imagens entre as classes de lesões de pele é muito desbalanceada, como evidenciado pelo banco de dados do desafio, no qual a proporção de casos positivos para a tarefa 1 (melanoma) é de apenas 18%.

Para exemplificar, suponha que a distribuição na população seja muito menor, com uma ocorrência de lesões malignas na proporção de 1% de todas as lesões de pele. Quando se tem um classificador muito otimista, que classifica todas as lesões como não malignas, ele terá uma acurácia preditiva de 99%, um valor muito alto. Do ponto de vista de acurácia preditiva, esse é um classificador muito bom. Infelizmente, ele não ajuda em nada a detectar as lesões malignas.

Por outro lado se for projetado um classificador que sempre fizesse a predição “maligno”, na maioria dos casos criaria uma preocupação desnecessária. Fica claro desses exemplos que nenhum dos dois classificadores é realmente útil na prática. O classificador ideal, que atribui o rótulo certo a cada classe é muito difícil de se encontrar para casos não triviais. Geralmente, tem-se fazer um compromisso entre os dois casos problemáticos descritos acima. Especialmente num banco de dados no qual as classes estão severamente desbalanceadas, a acurácia preditiva não é um indicador confiável da efetividade do classificador.

Para usar métricas que não estejam sujeitas aos defeitos descritos acima, é necessário definir algumas variáveis auxiliares:

- **Verdadeiros Positivos** (N_{VP}), é o número de instâncias positivas classificados como positivas.
- **Falsos Positivos** (N_{FP}), é o número de instâncias negativas classificados como positivas.
- **Verdadeiros Negativos** (N_{VN}), é o número de instâncias negativas classificados como negativas.
- **Falsos Negativos** (N_{FN}), é o número de instâncias positivas classificados como negativas.
- **Número de instâncias Positivas** ($N_P = N_{VP} + N_{FN}$), é o número total de instâncias positivas.
- **Número de instâncias Negativas** ($N_N = N_{FP} + N_{VN}$), é o número total de instâncias negativas.

Em aplicações de alto risco, como em alguns casos médicos, os falsos negativos têm importância especial pois são casos em que uma doença, possivelmente fatal, passa despercebida pelo médico.

Das quantidades descritas acima são calculadas outras métricas que serão de ajuda na obtenção de uma métrica mais abrangente.

- **Taxa de verdadeiros positivos** ($m_{TVP} = N_{VP}/N_P$), é a razão entre verdadeiros positivos e o total de instâncias positivas. Coloquialmente, é a probabilidade de detecção.
- **Taxa de falsos positivos** ($m_{TPF} = N_{FP}/N_N$), é a razão entre falsos positivos e o total de instâncias negativas. Coloquialmente, é a probabilidade de alarme falso.

3.3.2.1 A Característica de Operação de Receptor

A curva da característica de operação de receptor, mais conhecida no meio de ML pelo seu acrônimo em inglês ROC (*Receiver Operating Characteristic*), é uma representação gráfica do desempenho de um classificador binário que consegue remediar os problemas da métrica de acurácia preditiva já descritos. Especificamente, a curva ROC é uma representação gráfica que ilustra o desempenho (ou performance) de um sistema classificador binário a medida que seu limiar de discriminação é variado. Nesse sentido, cada ponto da ROC

correspondente à taxa de falsos positivos (abcissa) e à taxa de verdadeiros positivos (ordenada) para um determinado valor do limiar de discriminação. Um ponto na curva que está ao "noroeste" de outro pode ser considerado estritamente melhor que este, pois é melhor nas duas métricas.

A saída de uma CNN usada em regressão logística geralmente é dada como a probabilidade ou confiança de que cada exemplo pertence a alguma classe. No caso da classificação binária durante o treino, quando da confiança é maior que o limiar 0.5, considera-se que o que a amostra pertence à classe. Entretanto, esse limiar pode ser variado continuamente, mudando-se assim a qual classe é atribuída cada amostra. Isso torna as taxas de verdadeiros positivos e de falsos positivos funções desse limiar. Embora a mudança na classificação cause uma mudança descontínua em m_{TVP} e m_{TFP}, a descontinuidade pode ser tornada tanto menor quanto mais amostras se tiver.

Plotando-se os valores de m_{TVP} e m_{TFP} à medida que se varia o limiar, obtém-se a curva ROC. Essa curva contém todos os pontos que o classificador pode assumir, à medida que se varia o limiar de 0 a 1.

Alguns pontos são de importância especial na curva ROC. Como o classificador retorna probabilidades, usualmente fica no intervalo aberto $(0, 1)$, isto é, não atinge os valores 0 ou 1 pois não tem certeza absoluta. Particularmente:

- **O ponto** $(0, 0)$ corresponde a um classificador com um limiar de discriminação tão próximo de um que todas as probabilidades são menores que este limiar e portanto todos os exemplos são considerados falsos. Dessa forma $m_{TVP} = 0$ e $m_{TFP} = 0$.
- **O ponto** $(1, 1)$ corresponde a um classificador com um limiar de discriminação tão próximo de zero que todas as probabilidades são maiores que este limiar e portanto todos os exemplos são considerados verdadeiros. Dessa forma $m_{TVP} = 1$ e $m_{TFP} = 1$.

A linha diagonal $m_{TVP} = m_{TFP}$ corresponde a um classificador aleatório.

Essa forma de ver o desempenho do classificador tem a desvantagem de necessitar de um gráfico, ao contrário das outras métricas vistas, que somente necessitam de um valor escalar. Com a curva ROC, raramente é possível determinar definitivamente se um classificador é melhor que outro pois em um valor de limiar um classificador pode se sobressair enquanto para outro limiar, outro classificador pode ter a vantagem.

Área Sob a Curva

Para contornar esses dois problemas, há a possibilidade de se calcular a área sob a curva ROC, o que resulta em um valor escalar que pode ser ordenado (maior é melhor) para comparar classificadores. Esse valor também é conhecido pelo seu valor em inglês, ROC AUC (ROC *Area Under the Curve*).

3.3.3 Etapa de Pré - processamento da imagem

A implementação da etapa de pré-processamento consta de duas partes:

- **Normalização da cor**. A cor da imagem de entrada é normalizada usando o algoritmo de constância de cor *Shades of Gray* (49), uma vez que não precisa de conhecimento prévio sobre a configuração de aquisição da imagem, usando apenas a imagem como fonte de informação na estimação da cor da fonte de luz. Além disso, como é indicado em (50), este método demonstrou alcançar desempenhos semelhantes a algoritmos mais complexos que requerem uma etapa de treinamento. O efeito da normalização da cor pode ser apreciado ao comparar as Figuras 11a e 11b.
- **Realce de bordas**. Com o intuito de melhorar o desempenho da CNN, foi aplicado também um filtro de *Sharpening* (tradução livre, filtro de Nitidez) na imagem normalizada, com o objetivo de realçar informações espaciais importantes como observado na Figura 11c.

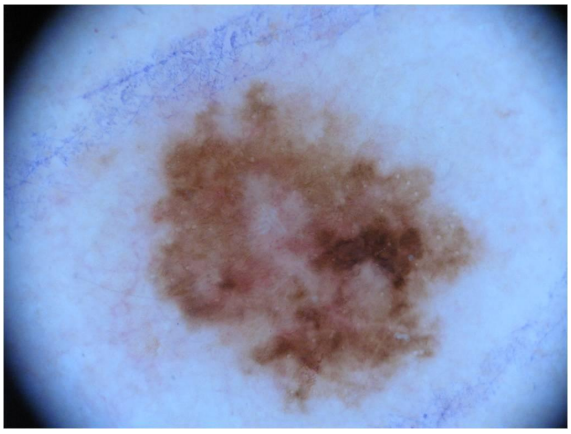
3.3.4 Etapa de extração de características e classificação

No caso desse trabalho é usada uma rede neural CNN, especificamente a rede *Inception-v3* como já descrito. Este tipo de rede pode efetuar simultaneamente as tarefas de extração de características e classificação. Além disso, aqui é usada a técnica de ML *transfer learning*, implicando que será usada uma rede já treinada para uma outra tarefa de classificação de imagens.

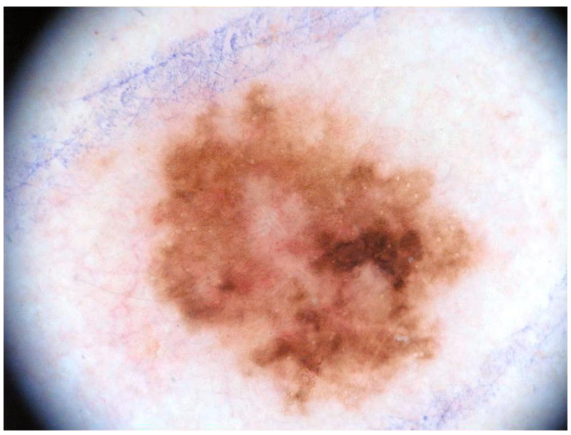
Baixou-se um arquivo contendo a definição da rede *Inception-v3*, como descrito na Tabela 2, e com os pesos pré-treinados sobre o *dataset* do ILSVRC. Foram adicionada a essa rede

Figura 11 – Pré processamento aplicado às imagens.

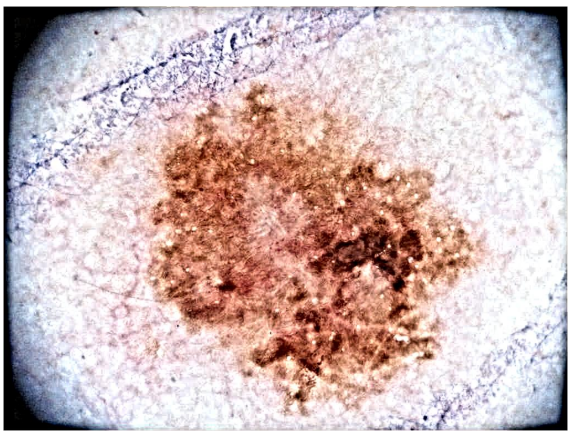
(a) Imagem original.



(b) Imagem normalizada via o algoritmo *Shades of Gray*.



(c) Imagem depois de aplicar o filtro de *Sharpening*.



Fonte: Produção do próprio autor.

algumas operações para adaptar as imagens do *dataset* do desafio ISIC 2017 (composto de imagens JPEG de diversos tamanhos) à entrada esperada pela primeira camada do *Inception-v3* (um tensor de *floats* de tamanho $299 \times 299 \times 3$) . Essas operações são responsáveis por:

- decodificar as imagens JPEG;
- subtrair a média aproximada dos valores (escolheu-se 128)
- transformar seus valores para *floats*;
- dividir os valores para tornar o desvio padrão próximo de um (escolheu-se 128)
- recortar (*crop*) o maior quadrado central da imagem;
- redimensionar a imagem para 299×299.

Após isso foram adicionadas as operações necessárias para as *bottlenecks*, o classificador, as operações de treino e as funções para calcular as métricas. Originalmente, essas operações estariam conectadas à saída do último módulo *Inception* da rede. Porém objetivou-se treinar classificadores sobre alguns diferentes módulos *Inception*, em diferentes profundidades da rede, para possivelmente obter resultados melhores com características aprendidas diferentes.

Cada classificador é construído da mesma forma que o classificador original da rede *Inception-v3*. Esse procedimento é descrito a seguir:

1. É tomada a saída do módulo em questão, e é tirada a média das ativações de cada canal. Isso é feito na prática com uma camada *average pooling* cujo filtro tem uma área receptiva de mesmas dimensões que a dimensões espaciais do canal. Nesse processo toda a informação espacial é perdida pois um tensor de entrada tem seu formato mudado de (N_{in}, A_{in}, L_{in}, C_{in}) para (N_{in}, 1, 1, C_{in}). Esse processo é diferente do usual processo de "achatamento" das dimensões espaciais em um vetor comprido para cada canal.
2. Os valores retornados pelo *average pooling* são passados para as funções que salvam esses valores em disco nos arquivos *bottlenecks*.
3. São adicionadas funções para ler *bottlenecks* em disco e carregá-los na forma de tensores.
4. Os valores lidos nos *bottlenecks* são passados para uma camada totalmente conectada com dois neurônios (um neurônio para cada classe).
5. Os *logits* (valores obtidos das camadas totalmente conectadas) são passadas por uma função *Softmax* para transformar seus valores num formato de probabilidades.

6. A função de perda (entropia cruzada) é calculada comparando-se a saída da operação *Softmax* com os rótulos dados (*ground truth*)

Então são adicionadas as operações de treino, para minimizar a função de perda (entropia cruzada), treinando-se os pesos e bias da camada totalmente conectada. Apenas os pesos da camada totalmente conectada são treinados, todos os outros parâmetros da rede original são mantidos fixos. Também são adicionadas as operações responsáveis por calcular as métricas de treino e avaliação.

Para as descrições a seguir, vale tomar nota dos seguintes hiper-parâmetros:

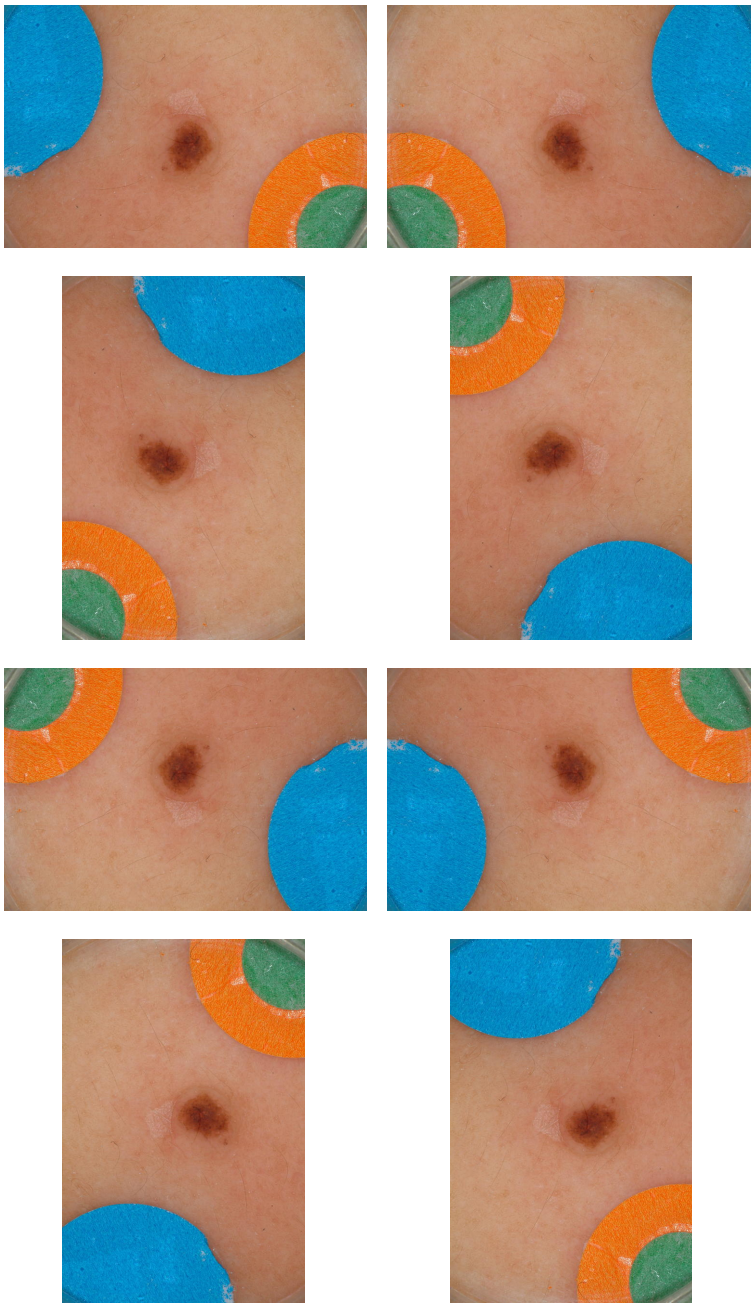
- **tamanho do *batch***: é a quantidade de amostras que são usadas para cada iteração de treino;
- **taxa de aprendizado**: valor escalar pelo qual são multiplicados os gradientes antes de aplicá-los aos pesos;
- **número de iterações de treino**: por quantas iterações foi treinada a rede.
- **número de épocas de treino**: medida aproximada de quantas vezes a rede "viu" o conjunto de treino inteiro durante o treino. Pode ser obtido pelo número de imagens visto durante o treino (tamanho do *batch* multiplicado pelo número de iterações de treino) dividido pelo número de imagens de treino no banco de dados.

Como descrito anteriormente, fez-se uso dos *bottlenecks* para acelerar o processo de treino. Antes do processo de treino do classificador sobre um determinado módulo *Inception*, foi calculada a ativação de cada imagem apenas uma vez, e esse valor foi salvo em disco. Na etapa de treino, esse valor é inserido diretamente na camada totalmente conectada, de forma que o treino do classificador se tornasse tão rápido quanto treinar uma rede de apenas uma camada.

Para o treino, foram testados alguns otimizadores adaptativos (Adam, Adagrad), mas percebeu-se que quando se fez uso deles, ocorria o problema de *overfitting*. Com uma mudança para SGD ou então *momentum*, o problema de *overfitting* não acontecia, ou acontecia de forma muito reduzida. Resultados semelhantes foram recentemente descritos (51), quando se mostrou que o uso de otimizadores adaptativos causa *overfitting* quando usados em sistemas sobreparametrizados.

Após algumas experimentações iniciais, decidiu-se por um tamanho de *batch* igual a 32. Este valor teve desempenho semelhante a um tamanho de *batch* de 64, mas optou-se pelo

Figura 12 – Permutações de rotação e espelhamento aplicadas.



Fonte: Produção do próprio autor.

número menor por este levar a um aprendizado mais ruidoso, conseguindo assim evitar melhor mínimos locais da função de perda.

Notou-se que mesmo utilizando-se o otimizador SGD, uma taxa de aprendizado que funciona muito bem para a saída de um módulo, causa *overfitting* quando usada para treinar um classificador sobre a saída de outro módulo. Portanto, experimentou-se com diversas taxas de aprendizado. Quando ocorria *overfitting* a taxa era dividida por um fator de aproximadamente 3.0. Inversamente, se um valor causasse um treino muito lento, a

taxa era multiplicada pelo mesmo fator. Foi selecionado apenas o classificador treinado com a melhor taxa encontrada (a que convergiu num valor bom mais rápido, sem causar *overfitting*) para cada módulo. Notou-se que módulos mais baixos (mais próximos da entrada) tenderam a necessitar de taxas de aprendizado mais altas para obter resultados semelhantes.

Dentre o conjunto dos melhores classificadores para cada camada, foram escolhidos os dois melhores para serem treinados por mais épocas, e com um conjunto de dados aumentado artificialmente.

A partir de cada imagem de treino foram geradas mais 7, por meio de espelhamentos e rotações. Inicialmente cada imagem foi espelhada em torno de seu eixo vertical (teve seus lados direito e esquerdo trocados). Com isso dobrou-se o número de imagens. Em seguida, cada uma dessas imagens recebeu três tipos de rotação (90°, 180° e 270°), de forma que todas as possibilidades de manipular a imagem sem distorção foram feitas. O *dataset* foi ampliado de 2000 imagens para 16000.

O processo descrito acima está ilustrado na Figura 12. A primeira coluna corresponde à imagem original e suas três rotações, enquanto a segunda coluna corresponde à imagem espelhada e suas rotações. A imagem foi escolhida para se poder perceber facilmente as diferenças entre cada permutação de rotação e espelhamento.

Não foram aplicados outros tipos de distorção sobre as imagens pois a assimetria é uma das características usadas para classificar melanomas. Dessa forma, essa estratégia para aumentar o *dataset* seria inadequada para essa tarefa específica.

Na Tabela 3 estão dados os valores de ROC AUC para os classificadores em cada um dos módulos. Esses valores foram obtidos para o conjunto de treino não modificado. A escolha das melhores taxas de aprendizado foi feita levando-se em conta apenas os resultados para o conjunto de validação. Esses resultados estão dados após 20 épocas de treino, com um tamanho de *batch* de 32 imagens.

Esse experimento foi repetido com o conjunto pré-processado e os resultados podem ser vistos na Tabela 4.

A partir dos dados das Tabela 3 e 4, foram escolhidos os classificadores sobre os módulos 8 e 9 com as imagens originais não processadas, por apresentarem o melhor desempenho sobre o conjunto de validação. Com a determinação dos melhores módulos e suas respectivas taxas de aprendizado, foram realizados treinamentos completos, com o banco de dados

Tabela 3 – Desempenho para classificadores em diferentes camadas.

Módulo	**ROC AUC**	**Taxa de aprendizado**
10	0.929	0.001
9	0.935	0.03
8	0.936	0.03
7	0.932	0.3
6	0.923	1.0

Fonte – Produção do próprio autor.

Tabela 4 – Desempenho para classificadores em diferentes camadas para imagens pré processadas.

Módulo	**ROC AUC**	**Taxa de aprendizado**
10	0.866	0.001
9	0.909	0.03
8	0.904	0.03

Fonte – Produção do próprio autor.

aumentado de 16 mil imagens. Essas fases de treinamento foram realizadas por 100 épocas.

3.4 Resultados experimentais

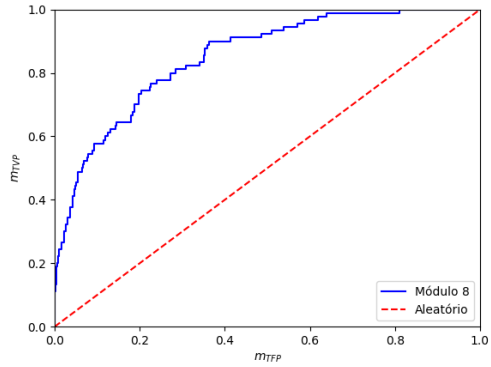
3.4.1 Resultados

Na Figura 13 está dada a ROC AUC depois do treinamento completo para a tarefa de ceratose seborréica versus melanoma e nevos para os dois classificadores escolhidos. Na Figura 14 está dado o mesmo resultado para a tarefa de melanoma versus ceratose seborréica e nevos. A partir dessas figuras vê-se que o classificador aprendeu informação útil, e está claramente acima de um classificador aleatório, denotado pelas linhas tracejadas vermelhas no gráfico.

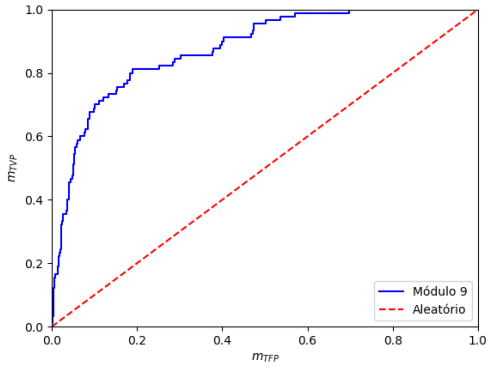
Tanto a biblioteca *TensorFlow*, usada para treinar e fazer as predições da rede, quanto a biblioteca *scikit-learn*, usada para traçar os gráficos das curvas ROC, provêm funções para calcular a ROC AUC. Entretanto, na suas configurações padrão elas provêm resultados significantemente diferentes. Os gráficos estão acompanhados do valor calculado com a função de *scikit-learn*. A função do *TensorFlow* provê uma forma de mudar o número de pontos usados para fazer o cálculo da ROC AUC e notou-se que quando se diminuía este número, o valor decrescia, ficando mais próximo do calculado pelo *scikit-learn*. Quanto

Figura 13 – Curvas ROC sobre o conjunto de teste para classificação de ceratose seborréica.

(a) Curva ROC para o módulo 8 com uma AUC = 0.850



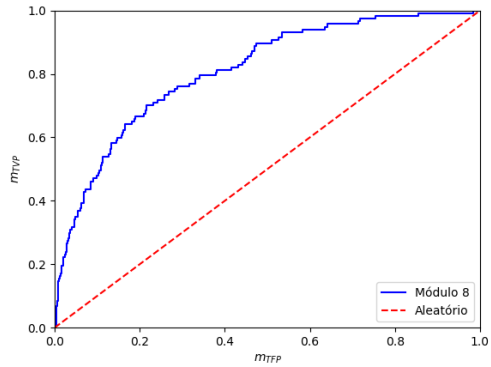
(b) Curva ROC para o módulo 9, com uma AUC = 0.877



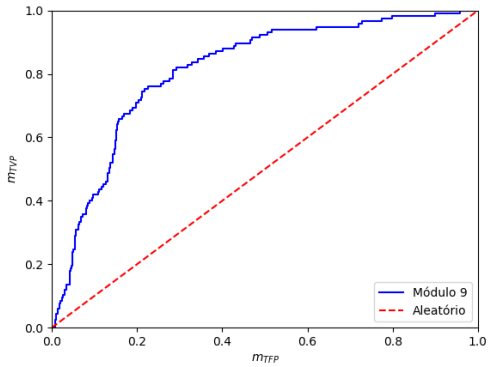
Fonte: Produção do próprio autor.

Figura 14 – Curvas ROC sobre o conjunto de teste para classificação de melanoma .

(a) Curva ROC para o módulo 8 com uma AUC = 0.810



(b) Curva ROC para o módulo 9, com uma AUC = 0.813



Fonte: Produção do próprio autor.

se aumentava o número de pontos, o valor estabilizava próximo do que considerou-se por final. Como a quantidade de pontos usados para fazer uma aproximação de área por meio de integral está diretamente relacionado com a precisão do cálculo, isso aponta para que o valor dado pelo *TensorFlow* seja o mais próximo da realidade. A função dada pelo *scikit-learn* não possibilita um ajuste semelhante.

Na Tabela 5 são dados as ROC AUC para cada tarefa, módulo e biblioteca usada. Nota-se que os valores dados pela função do *TensorFlow* são substancialmente mais altos. Também fica claro que o classificador sobre o módulo 9 se sobressaiu ao outro. Para efeitos de comparação somente será considerado o classificador sobre o módulo 9.

Tabela 5 – Comparação das ROC AUC obtidas por métodos diferentes.

Tarefa	**Módulo 8 (TensorFlow)**	**Módulo 8 (scikit-learn)**	**Módulo 9 (TensorFlow)**	**Módulo 9 (scikit-learn)**
Melanoma	0.852	0.810	0.862	0.813
Ceratose Seborréica	0.876	0.850	0.912	0.877

Fonte – Produção do próprio autor

3.4.2 Comparação

Para obter uma noção da qualidade dos classificadores treinados, eles foram comparados com as submissões ao desafio ISIC 2017. Apenas o classificador sobre o módulo 9 é usado para comparação em cada tarefa. O desempenho dos classificadores é medido por meio da ROC AUC. Os classificadores obtidos nesse trabalho tiveram um desempenho como dado na Tabela 6.

Tabela 6 – Valores obtidos para comparação no ranking.

fonte	**Melanoma**	**Ceratose Seborréica**	**Média**
TensorFlow	0.862	0.912	0.887
scikit-learn	0.813	0.877	0.845

Fonte – Produção do próprio autor

No total, foram 23 equipes para cada uma das tarefas. São dadas três tabelas de classificação com os 10 primeiros colocados de cada tarefa:

- Classificação de melanoma (Tabela 7)
- Classificação de ceratose seborréica (Tabela 8)
- Classificação geral (Tabela 9)

A classificação geral é medida pela média simples dos valores de ROC AUC para cada uma das tarefas específicas.

Comparando os desempenhos obtidos com os dos participantes do desafio, a classificação que teria sido obtida caso fosse possível participar do desafio está dada na Tabela 10.

Uma breve descrição dos métodos usados pelos 4 primeiros colocados no *ranking* geral permite uma comparação ao método deste trabalho. Estes são os trabalhos que obtiveram

Tabela 7 – *Ranking* para o problema de **classificação de melanoma** do desafio ISIC 2017.

Posição	**Equipe**	**desempenho**
1	RECOD Titans (52)	0.874
2	Lei Bi (53)	0.870
3	Kazuhisa Matsunaga (54)	0.868
4	monty python (55)	0.856
5	T D	0.836
6	Xulei Yang	0.830
7	Rafael Souza	0.805
8	x j	0.804
9	Cristina Vasconcelos	0.791
10	CV	0.789

Fonte – Produção do próprio autor

Tabela 8 – *Ranking* para o problema de **classificação de ceratose seborréica** do desafio ISIC 2017.

Posição	**Equipe**	**desempenho**
1	monty python (55)	0.965
2	Kazuhisa Matsunaga (54)	0.953
3	RECOD Titans (52)	0.943
4	Xulei Yang	0.942
5	T D	0.935
6	Lei Bi (53)	0.921
7	CV	0.911
8	Cristina Vasconcelos	0.911
9	Masih Mahbod	0.908
10	Dylan Shen	0.886

Fonte – Produção do próprio autor

um resultado geral superior ao deste trabalho, se considerada o valor de ROC AUC provido pelo *TensorFlow*. São eles:

1. ***Image Classification of Melanoma, Nevus and Seborrheic Keratosis by Deep Neural Network Ensemble*** (54) Primeiro colocado geral, terceiro para melanoma e segundo para ceratose seborréica. Fizeram uso de:
   - pré-processamento: normalização da iluminância e da cor;
   - arquitetura do classificador: *ResNet-50*
   - treino: *transfer learning* com ajuste fino dos pesos;
   - dados externos: 1444 imagens;
   - *data augmentation*: rotação, translação, mudança de escala e espelhamento;
   - avaliação: *ensemble*

Tabela 9 – *Ranking* geral do desafio ISIC 2017.

Posição	**Equipe**	**desempenho**
1	Kazuhisa Matsunaga (54)	0.911
2	monty python (55)	0.910
3	RECOD Titans (52)	0.908
4	popleyi . (53)	0.896
5	Xulei Yang	0.886
6	T D	0.886
7	Cristina Vasconcelos	0.851
8	Cristina Vasconcelos	0.850
9	Euijoon Ahn	0.836
10	x j	0.829

Fonte – Produção do próprio autor

Tabela 10 – Classificação hipotética no desafio.

fonte	**Melanoma**	**Ceratose Seborréica**	**Média**
TensorFlow	4	7	5
scikit-learn	7	13	9

Fonte – Produção do próprio autor

2. ***Incorporating the Knowledge of Dermatologists to Convolutional Neural Networks for the Diagnosis of Skin Lesions*** (55) Segundo colocado geral, quarto para melanoma e primeiro para ceratose seborréica. Fez uso de:

   - segmentação: usa uma rede VGG para fazer uma segmentação da lesão;
   - anotações externas: anotações adicionais foram feitas por um conjunto de médicos sobre a presença de indicadores tradicionais de lesões e a partir delas foi feito um extrator para essa características com uma rede VGG;
   - arquitetura do classificador: *ResNet-50*;
   - treino: *transfer learning*;
   - dados externos: dataset de treino do ISIC 2016 e as anotações;
   - *data augmentation*: rotação e mudança de escala;
   - avaliação: *ensemble* do classificador normal com um classificador para as características e um para as imagens em coordenadas polares.

3. ***RECOD Titans at ISIC Challenge 2017*** (52) Quarto colocado geral, primeiro para melanoma e terceiro para ceratose seborréica. Fizeram uso de:

   - arquitetura do classificador: *ResNet-101* e *Inception-v4*;
   - treino: *transfer learning* com ajuste fino;
   - dados externos: 7544 imagens de diversos *datasets* do domínio público;

- *data augmentation*: rotação e mudança de escala e deslocamento;
- avaliação: *ensemble* de 7 redes.

4. ***Automatic Skin Lesion Analysis using Large-scale Dermoscopy Images and Deep Residual Networks*** (52) Terceiro colocado geral, segundo para melanoma e sexto para ceratose seborréica. Fizeram uso de:

   - segmentação: usa uma rede VGG para fazer uma segmentação da lesão;
   - arquitetura do classificador: *ResNet*;
   - treino: *transfer learning* com ajuste fino;
   - dados externos: aproximadamente 1600 imagens do domínio público;
   - *data augmentation*: rotação, espelhamentos e mudança de escala;
   - avaliação: *ensemble* de 2 redes.

Vale notar que dentre estes listados estão os três vencedores (melanoma, ceratose e geral) e que todos os listados fizeram uso de *ensembles* de classificadores, *transfer learning*, *data augmentation*, além de usar mais imagens para treino do que provido pelo desafio (46).

Enquanto a técnica de *ensembles* é fácil de implementar, a tarefa de coletar e rotular mais imagens para treino é algo muito trabalhoso e requer um tempo considerável. Embora o uso de imagens externas seja permitido pelo desafio, não mede o desempenho da arquitetura em si, e qualquer participante que tivesse acesso a esse banco de dados aumentado teria visto ganhos no desempenho de seu classificador.

O uso de *data augmentation* foi semelhante ao deste trabalho, com alguns destes realizando mais e outros ligeiramente menos variações.

No entanto, uma escolha notável e comum a todos os exemplos acima listados foi o uso de *transfer learning* com ajuste fino dos pesos. Isso acarretou num menor tempo de treino reportado entre os trabalhos listados de 12 horas em duas GPUs de ponta (Nvidia Titan X).

Neste trabalho, ao não fazer o ajuste fino, foi possível fazer uso da técnica dos *bottlenecks* e assim possibilitar um tempo de treino centenas de vezes menor, sem considerar a diferença de poder de processamento. Tudo isso sem acarretar em grandes diminuições do desempenho do classificador obtido. Outro apecto de *transfer learning* do qual não foi lançado mão pelos trabalhos listados foi o de não usar as camadas superiores. Isso tende a melhorar o desempenho e é um trabalho relativamente fácil. Vale ressaltar novamente que

neste trabalho não foram usadas imagens externas ao *dataset* do desafio e ainda assim em alguns casos foi obtido um desempenho próximo, com um tempo de treino muito menor.

4 CONCLUSÕES E PROJETOS FUTUROS

4.1 Conclusões

Diagnósticos auxiliados por computador são recursos essenciais para ajudar médicos na detecção e prevenção de câncer de pele, uma vez que conseguem detectar características que muitas vezes passam despercebidas até mesmo pelo olhar de especialistas. Entretanto, os métodos atuais estão fundamentados em extração de características baseadas em métodos de inspeção visual tradicionais, e portanto não diferem tanto do que pode ser diagnosticado por um médico.

Os métodos de DL conseguem extrair características otimizadas da distribuição relacionada à tarefa de classificação. Com um pouco de inventividade humana na hora de criar as arquiteturas de DL, nasceram classificadores com tanta habilidade de abstração que apenas com imagens e rótulos tem a capacidade de aprender por si mesmos as características que os permitem superar o desempenho humano em algumas tarefas altamente complexas.

Uma dessas redes é a *Inception-v3*, que em algumas de suas variações conseguiu repetidamente atingir a posição de melhor classificador de imagens. Por meio de técnicas como *transfer learning*, tornou-se possível transferir todo o conhecimento codificado nos pesos aprendidos da rede. Dessa forma esse conhecimento se tornou acessível para qualquer pessoa equipada de um computador a nível consumidor, o que torna possíveis trabalhos como este.

Na questão desempenho, o classificador treinado ficou com um nível satisfatório e comparável a outros do desafio que dispunham do mesmo banco de dados. Entretanto, não foi comparado o desempenho destes a métodos tradicionais de diagnóstico auxiliado por computador, ou mesmo a médicos dermatologistas. Entretanto, o fato de que todos os trabalhos que obtiveram um resultado superior ao deste fizeram uso de CNNs profundas, com apenas um eventual auxílio de um método tradicional, indica uma superioridade das CNNs.

A aplicação de *transfer learning* possibilitou o treinamento do classificador em um computador comum. Notou-se que, como descrito na literatura, o desempenho ficou melhor quando se retirou algumas camadas da rede original e se treinou o classificador sobre camadas mais baixas. Entretanto, ao contrário do esperado devido à dissimilaridade das

tarefas, as camadas ideais para a tarefa de classificação de câncer estavam bem próximas do classificador final. Um motivo para isso pode ter sido o uso do *average pooling* em camadas nas quais a informação espacial ainda é importante. Dessa forma, adicionar uma camada totalmente conectada no lugar do *average pooling* pode levar a um desempenho melhor das camadas baixas.

Surpreendentemente, ao contrário do que indicava a literatura, o uso de imagens pré processadas diminuiu o desempenho do classificador. Isso pode ser devido ao uso de *transfer learning* já que a rede original foi treinada para um espaço de cores de imagens naturais. Particularmente, a operação de *sharpening* parece ter modificado as imagens a ponto de perderem informação importante. É provável que o uso das imagens apenas normalizadas em cor leve a uma melhora em relação às imagens normais.

O presente trabalho se mostrou extremamente satisfatório em nível de aprendizado em CNNs, *transfer learning*, e processamento de imagens.

4.2 Temas a serem pesquisados

No processo de leitura teórica para o trabalho, surgiram diversos temas e ideias que por diferentes motivos não foram estudados a fundo. Aqui são indicadas algumas dessas ideias para que se possa continuar a pesquisa deste fascinante tema:

- Analisar o motivo do pré-processamento não ter funcionado mais a fundo, já que a literatura indica ganhos substanciais com essa técnica.
- Coletar mais dados de treino usando banco de dados disponíveis na internet e verificar o impacto de mais dados sobre o desempenho do classificador.
- Repetir o experimento de *transfer learning* com as redes que são o atual estado da arte.
- Experimentar com a técnica de *ensemble learning* (56) na qual as predições de vários classificadores são consideradas e a predição final é com base em uma combinação dessas.
- Usar técnicas de visualização da informação aprendida para descobrir onde estão falhas no classificador, ou no banco de dados, e com base nisso fazer melhoras na arquitetura ou no pré processamento das imagens.

- Ao longo do treino, especialmente quando foi treinado um classificador sobre cada um dos módulos, tornou-se extremamente repetitivo o trabalho de achar os hiper-parâmetros ideais para cada módulo. Essa tarefa acontece na base da experimentação e com muitas tentativas. Segundo um recente estudo por Golovin et al.(57):

  > Qualquer sistema suficientemente complexo acaba se tornando efetivamente uma caixa preta quando se torna mais fácil de experimentar com ele do que entendê-lo.

  Portanto, experimentar com o uso de ferramentas que automatizem o processo de ajustar hiper-parâmetros pode ser de grande proveito. Em ML esta área é conhecida como *black-box optimization*.

REFERÊNCIAS

1 DIMATOS, D. C. et al. Melanoma cutâneo no Brasil. *Arquivos Catarinenses de Medicina*, v. 38, n. Suplemento 01, p. 14, 2009. Citado na página 12.

2 JAIN, S.; JAGTAP, V.; PISE, N. Computer Aided Melanoma Skin Cancer Detection Using Image Processing. *Procedia Computer Science*, v. 48, p. 736–741, dec 2015. Citado 2 vezes nas páginas 12 e 17.

3 ESTEVA, A. et al. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, Macmillan Publishers Limited, part of Springer Nature. All rights reserved., v. 542, p. 115, jan 2017. Disponível em: <http://dx.doi.org/10.1038/nature2105610.1038/nature21056>. Citado 2 vezes nas páginas 12 e 17.

4 KITTLER, H. et al. Diagnostic accuracy of dermoscopy. *The Lancet Oncology*, Elsevier, v. 3, n. 3, p. 159–165, dec 2017. ISSN 1470-2045. Disponível em: <http://dx.doi.org/10.1016/S1470-2045(02)00679-4>. Citado na página 12.

5 VESTERGAARD, M. et al. Dermoscopy compared with naked eye examination for the diagnosis of primary melanoma: a meta-analysis of studies performed in a clinical setting. *British Journal of Dermatology*, Blackwell Publishing Ltd, v. 159, n. 3, p. ???–???, jun 2008. ISSN 00070963. Disponível em: <http://doi.wiley.com/10.1111/j.1365-2133.2008.08713.x>. Citado na página 12.

6 CARLI, P. et al. Pattern analysis, not simplified algorithms, is the most reliable method for teaching dermoscopy for melanoma diagnosis to residents in dermatology. *British Journal of Dermatology*, Blackwell Science Ltd, v. 148, n. 5, p. 981–984, may 2003. ISSN 00070963. Disponível em: <http://doi.wiley.com/10.1046/j.1365-2133.2003.05023.x>. Citado na página 12.

7 GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>. Citado 2 vezes nas páginas 13 e 14.

8 HUBEL, D. H.; WIESEL, T. N. Receptive fields and functional architecture of monkey striate cortex. *The Journal of Physiology*, v. 195, n. 1, p. 215–243, mar 1968. ISSN 0022-3751. Disponível em: <http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1557912/>. Citado 2 vezes nas páginas 14 e 19.

9 LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *Nature*, v. 521, n. 7553, p. 436–444, 2015. ISSN 14764687. Citado na página 14.

10 LECUN, Y. et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, v. 86, n. 11, p. 2278–2323, 1998. ISSN 00189219. Citado 2 vezes nas páginas 14 e 20.

11 RUSSAKOVSKY, O. et al. ImageNet Large Scale Visual Recognition Challenge. *CoRR*, abs/1409.0575, 2014. Disponível em: <http://arxiv.org/abs/1409.0575>. Citado 5 vezes nas páginas 14, 20, 23, 26 e 27.

12 HE, K. et al. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: *Proceedings of the IEEE International Conference on Computer Vision*. [S.l.: s.n.], 2015. ISBN 9781467383912. ISSN 15505499. Citado na página 14.

13 OPENAI. *Dota 2*. 2017. Disponível em: <https://blog.openai.com/dota-2/>. Citado na página 14.

14 SILVER, D. et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, Nature Publishing Group, a division of Macmillan Publishers Limited. All Rights Reserved., v. 529, p. 484, jan 2016. Citado na página 14.

15 GULSHAN, V.; PENG, L.; AL., E. Development and Validation of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs. *JAMA: Journal of the American Medical Association*, v. 316, n. 22, p. 2402–2410, dec 2016. ISSN 0098-7484. Disponível em: <http://dx.doi.org/10.1001/jama.2016.17216>. Citado na página 14.

16 WENG, S. F. et al. Can machine-learning improve cardiovascular risk prediction using routine clinical data? *PLOS ONE*, Public Library of Science, v. 12, n. 4, p. e0174944–, apr 2017. Disponível em: <https://doi.org/10.1371/journal.pone.0174944>. Citado na página 14.

17 ZHANG, W. et al. Computerized detection of clustered microcalcifications in digital mammograms using a shift-invariant artificial neural network. *Medical Physics*, American Association of Physicists in Medicine, v. 21, n. 4, p. 517–524, apr 1994. ISSN 00942405. Disponível em: <http://doi.wiley.com/10.1118/1.597177>. Citado 2 vezes nas páginas 14 e 19.

18 CREVIER, D. *AI: the tumultuous history of the search for artificial intelligence*. Basic Books, 1993. ISBN 9780465029976. Disponível em: <https://books.google.com.br/books?id=QJNQAAAAMAAJ>. Citado na página 15.

19 ZEILER, M. D.; FERGUS, R. Visualizing and Understanding Convolutional Networks. *CoRR*, abs/1311.2, 2013. Disponível em: <http://arxiv.org/abs/1311.2901>. Citado 3 vezes nas páginas 17, 27 e 39.

20 KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. ImageNet Classification with Deep Convolutional Neural Networks. *Advances In Neural Information Processing Systems*, p. 1–9, 2012. ISSN 10495258. Citado 4 vezes nas páginas 17, 20, 22 e 27.

21 SZEGEDY, C. et al. Going deeper with convolutions. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2015. v. 07-12-June, p. 1–9. ISBN 9781467369640. ISSN 10636919. Citado 4 vezes nas páginas 17, 20, 29 e 34.

22 FUKUSHIMA, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, v. 36, n. 4, p. 193–202, 1980. ISSN 03401200. Citado na página 19.

23 LECUN, Y. et al. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Comput.*, MIT Press, Cambridge, MA, USA, v. 1, n. 4, p. 541–551, 1989. ISSN 0899-7667. Disponível em: <http://dx.doi.org/10.1162/neco.1989.1.4.541>. Citado 3 vezes nas páginas 19, 20 e 23.

24 STEINKRAUS, D.; BUCK, I.; SIMARD, P. Y. Using GPUs for machine learning algorithms. In: IEEE. *Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on*. [S.l.], 2006. p. 1115–1120. ISBN 0769524206. Citado na página 20.

25 SIMONYAN, K.; ZISSERMAN, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *International Conference on Learning Representations (ICRL)*, abs/1409.1, p. 1–14, 2015. Disponível em: <http://arxiv.org/abs/1409.1556>. Citado 2 vezes nas páginas 20 e 27.

26 GU, J. et al. Recent advances in convolutional neural networks. *CoRR*, abs/1512.07108, 2015. Citado na página 20.

27 RAWAT, W.; WANG, Z. Deep convolutional neural networks for image classification: A comprehensive review. *Neural Computation*, v. 29, n. 9, p. 2352–2449, 2017. Citado na página 20.

28 LECUN, Y. et al. Efficient backprop. In: _____. *Neural Networks: Tricks of the Trade*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998. p. 9–50. Citado na página 22.

29 NAIR, V.; HINTON, G. E. Rectified linear units improve restricted boltzmann machines. In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. USA: Omnipress, 2010. (ICML'10), p. 807–814. Citado na página 22.

30 BOUREAU, Y.-L.; PONCE, J.; LECUN, Y. A theoretical analysis of feature pooling in visual recognition. In: *27TH INTERNATIONAL CONFERENCE ON MACHINE LEARNING, HAIFA, ISRAEL*. [S.l.: s.n.], 2010. Citado na página 22.

31 WANG, T. et al. End-to-end text recognition with convolutional neural networks. In: *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*. [S.l.: s.n.], 2012. p. 3304–3308. Citado na página 22.

32 RUMELHART, D. et al. *Learning Internal Representations by Error Propagation*. Institute for Cognitive Science, University of California, San Diego, 1985. (ICS report). Disponível em: <https://books.google.com.br/books?id=Ff9iHAAACAAJ>. Citado na página 23.

33 WIJNHOVEN, R. G. J.; WITH, P. H. N. de. Fast training of object detection using stochastic gradient descent. In: *2010 20th International Conference on Pattern Recognition*. [S.l.: s.n.], 2010. p. 424–427. Citado na página 24.

34 ZINKEVICH, M. A. et al. Parallelized stochastic gradient descent. In: *Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 2*. USA: Curran Associates Inc., 2010. (NIPS'10), p. 2595–2603. Citado na página 24.

35 E. Hinton, G. et al. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint*, arXiv, 2012. Citado na página 27.

36 HE, K. et al. Deep Residual Learning for Image Recognition. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [s.n.], 2016. p. 770–778. ISBN 978-1-4673-8851-1. ISSN 1664-1078. Disponível em: <http://ieeexplore.ieee.org/document/7780459/>. Citado 3 vezes nas páginas 28, 29 e 35.

37 SZEGEDY, C. et al. Rethinking the Inception Architecture for Computer Vision. *CoRR*, abs/1512.0, 2015. ISSN 08866236. Disponível em: <http://arxiv.org/abs/1512.00567>. Citado 5 vezes nas páginas 29, 30, 32, 33 e 34.

38 SZEGEDY, C. et al. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. *CoRR*, abs/1602.0, 2016. ISSN 01678655. Disponível em: <http://arxiv.org/abs/1602.07261>. Citado na página 29.

39 LIN, M.; CHEN, Q.; YAN, S. Network in network. *arXiv preprint*, p. 10, 2013. ISSN 03029743. Disponível em: <http://arxiv.org/abs/1312.4400>. Citado 2 vezes nas páginas 30 e 33.

40 KOLEN, J. F.; KREMER, S. C. Gradient flow in recurrent nets: The difficulty of learning longterm dependencies. In: _____. *A Field Guide to Dynamical Recurrent Networks*. Wiley-IEEE Press, 2001. p. 464–. ISBN 9780470544037. Disponível em: <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5264952>. Citado na página 35.

41 WEISS, K.; KHOSHGOFTAAR, T. M.; WANG, D. D. A survey of transfer learning. *Journal of Big Data*, v. 3, n. 1, p. 40, may 2016. ISSN 21961115. Disponível em: <https://doi.org/10.1186/s40537-016-0043-6>. Citado na página 36.

42 CHOLLET, F. Xception: Deep Learning with Depthwise Separable Convolutions. *CoRR*, abs/1610.02357, 2016. Disponível em: <http://arxiv.org/abs/1610.02357>. Citado na página 36.

43 PAN, S. et al. *A Survey on Transfer Learning*. 2010. Citado na página 38.

44 YOSINSKI, J. et al. How transferable are features in deep neural networks? 2014. ISSN 10495258. Disponível em: <https://arxiv.org/abs/1411.1792http://arxiv.org/abs/1411.1792>. Citado na página 39.

45 OLAH, C.; MORDVINTSEV, A.; SCHUBERT, L. Feature Visualization. *Distill*, 2017. Citado na página 39.

46 CODELLA, N. C. F. et al. Skin Lesion Analysis Toward Melanoma Detection: A Challenge at the 2017 International Symposium on Biomedical Imaging (ISBI), Hosted by the International Skin Imaging Collaboration (ISIC). *CoRR*, p. 2–6, 2017. Disponível em: <http://arxiv.org/abs/1710.05006>. Citado 2 vezes nas páginas 40 e 60.

47 YUKDOWSKY, E. Artificial intelligence as a positive and negative factor in global risk. 01 2008. Citado na página 44.

48 WARDEN, P. : tutorial. Citado na página 45.

49 FINLAYSON, G.; TREZZI, E. Shades of gray and colour constancy. p. 37–41, 01 2004. Citado na página 49.

50 GIJSENIJ, A.; GEVERS, T.; WEIJER, J. van de. Computational color constancy: Survey and experiments. *IEEE Transactions on Image Processing*, v. 20, n. 9, p. 2475–2489, Sept 2011. Citado na página 49.

51 WILSON, A. C. et al. The Marginal Value of Adaptive Gradient Methods in Machine Learning. p. 1–14, 2017. Disponível em: <http://arxiv.org/abs/1705.08292>. Citado na página 52.

52 MENEGOLA, A. et al. RECOD titans at ISIC challenge 2017. *CoRR*, abs/1703.04819, 2017. Disponível em: <http://arxiv.org/abs/1703.04819>. Citado 3 vezes nas páginas 58, 59 e 60.

53 BI, L. et al. Automatic skin lesion analysis using large-scale dermoscopy images and deep residual networks. *CoRR*, abs/1703.04197, 2017. Disponível em: <http://arxiv.org/abs/1703.04197>. Citado 2 vezes nas páginas 58 e 59.

54 MATSUNAGA, K. et al. Image classification of melanoma, nevus and seborrheic keratosis by deep neural network ensemble. *CoRR*, abs/1703.03108, 2017. Disponível em: <http://arxiv.org/abs/1703.03108>. Citado 2 vezes nas páginas 58 e 59.

55 GONZÁLEZ-DÍAZ, I. Incorporating the knowledge of dermatologists to convolutional neural networks for the diagnosis of skin lesions. *CoRR*, abs/1703.01976, 2017. Disponível em: <http://arxiv.org/abs/1703.01976>. Citado 2 vezes nas páginas 58 e 59.

56 MACLIN, R.; OPITZ, D. W. Popular ensemble methods: An empirical study. *CoRR*, abs/1106.0257, 2011. Disponível em: <http://arxiv.org/abs/1106.0257>. Citado na página 63.

57 GOLOVIN, D. et al. Google Vizier. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '17*, p. 1487–1495, 2017. Disponível em: <http://dl.acm.org/citation.cfm?doid=3097983.3098043>. Citado na página 64.