Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Uma avaliação elaborada dos principais modelos de referência para classificação de imagens

Matheus Oliveira Braga

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Orientador
Prof. Dr. Ricardo Lopes de Queiroz

Brasília
2021

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Uma avaliação elaborada dos principais modelos de referência para classificação de imagens

Matheus Oliveira Braga

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Prof. Dr. Ricardo Lopes de Queiroz (Orientador)
CIC/UnB

Prof. Dr. Bruno Luiggi Macchiavello Espinoza
CIC/UnB

Prof. Dr. Renan Utida Barbosa Ferreira
FGA/UnB

Prof. Dr. João José Costa Gondim
Coordenador do Curso de Engenharia da Computação

Brasília, 20 de maio de 2021

Dedicatória

Dedico este trabalho aos meus pais que sempre fizeram de tudo para que eu pudesse ter o maior conforto possível para me dedicar totalmente aos meus objetivos e sempre me apoiaram nos momentos mais difíceis.

Agradecimentos

Primeiramente agradeço à minha família, em especial meus pais por proporcionarem a minha educação e exemplos de vida, nunca poderia ter chegado onde estou sem o apoio incondicional deles.

Agradeço à Universidade de Brasília, por fornecer os professores capacitados e ambiente acadêmico no qual passei todos esses anos e usufrui para me tornar o indivíduo que sou hoje.

Agradeço aos meus colegas de curso, que foram companhia essencial durante essa caminhada, com companheirismo, risadas, descontrações e seriedade nos momentos necessários.

Agradeço a todos os professores que fizeram parte dessa minha jornada, transmitindo o conhecimento necessário para a minha evolução.

Ao meu colega, Marcos Tonin, pela proposta interessante do tema em um momento delicado do trabalho.

Ao meu orientador, professor Dr. Ricardo Lopes de Queiroz, que esteve sempre disponível a ajudar de todas as formas possíveis neste projeto.

Por último, agradeço a todos que não foram mencionados mas que contribuíram de alguma forma nesta caminhada.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES), por meio do Acesso ao Portal de Periódicos.

Resumo

Ultimamente, a área da aprendizagem profunda vem sendo um objeto de pesquisa muito comum no campo da visão computacional. Por conta deste interesse de desenvolvimento na área, uma vasta gama de modelos de redes neurais propostos para classificação e reconhecimento de imagens e objetos vem surgindo, cada um com suas qualidades e defeitos. Consequentemente, pesquisadores que desejam apenas usufruir de tais modelos para auxiliar em suas pesquisas de áreas semelhantes, acabam necessitando se aprofundar no assunto apenas para poder decidir quais modelos melhor se encaixariam em seus trabalhos.

Este trabalho consiste em fazer uma análise comparativa dos principais modelos de referência para reconhecimento de imagens e detecção de objetos. O foco principal é conseguir apontar quais modelos se sobressaem em aspectos de precisão e tempo de execução, para que se possa obter uma noção de quais são indicados para certos tipos de situações. Por exemplo, uma boa precisão com um modelo mais leve, para máquinas menos robustas, ou a melhor eficiência possível, descartando a preocupação com a utilização de recursos.

Para tal, cada um dos modelos apresentados neste trabalho foi avaliado dentro do mesmo ambiente de *software* e *hardware*, e utilizando as mesmas bases de dados, sendo um deles composto por 50000 imagens do banco de dados da *ImageNet*, comumente utilizado em desafios de reconhecimento de imagens, e o outro, um banco composto por 100 imagens coletadas pelo autor deste trabalho.

Os resultados mostraram que o Inception-V3, DenseNet e ResNet, empregaram as melhores estratégias para precisão na classificação, enquanto o ShuffleNet-V2 mostrou ser o modelo mais econômico.

Palavras-chave: aprendizagem profunda, redes neurais convolucionais, modelos de referência, desempenho de redes neurais, reconhecimento de imagens, detecção de objetos, visão computacional

Abstract

Deep learning has been a common object of research in computer vision. Because of this large interest, a considerable amount of neural network models for classification and recognition of images and objects have been appearing, each with their own strengths and weaknesses. Consequently, researchers that wish to simply use these models as aid to their work in similar areas end up having to hard commit to studying these concepts just for the simplicity of deciding which models are best suited to develop their research.

This work consists in an analysis and comparison of the most well known resource models for image recognition and object detection. The main goal is to appoint which model better performs in aspects such as accuracy and execution time, to have a notion of which are suited for certain situations. For example, a less resource consuming model with decent accuracy for less powerful machines, or maybe, the best efficiency without any concern regarding resource consumption.

With that in mind, each one of the models presented in this work have been evaluated on the same software and hardware environments, and with the same datasets, one of them being the well known Imagenet set database, which includes 50000 images, commonly used in image recognition challenges, and the other one being a dataset built by the author of this document himself, containing 100 pieces.

The results display that Inception-V3, DenseNet and ResNet applied the best strategies for precision, while ShuffleNet-V2 was the most economic model.

Keywords: deep learning, convolutional neural networks, resource models, neural network performances, image recognition, object detection, computer vision

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

ANN Artificial Neural Network.

CNN Convolutional Neural Network.

CPU Central Process Unit.

CUDA Compute Unified Device Architecture.

FLOPs Floating-point Operations per second.

GB gigabyte.

GHz Giga hertz.

GPU Graphics processing unit.

IA Inteligência Artificial.

ILSVRC ImageNet Large Scale Visual Recognition Challenge.

LSR Label-smoothing Regularization.

MB megabyte.

RAM Random access memory.

ReLU Rectified Linear Unit.

Capítulo 1

Introdução

1.1 Contextualização

A inteligência artificial, termo que oficialmente surgiu em 1956 para denominar o comportamento semelhante de uma máquina ao de um ser humano em termos de raciocínio e ação [9], é uma importante área de pesquisa da Ciência da Computação. As maiores empresas de tecnologia mundiais, como a Google e o Facebook, investem grande parte de seus recursos em pesquisas relacionadas a IA, e acredita-se que até 2024, a indústria da robótica, fortemente relacionada, tenha um valor estimado de 80 bilhões de dólares [10].

Dentro do campo da IA, um nicho muito popular de pesquisa é o da aprendizagem profunda, que vem tendo um crescente interesse nos últimos anos (Figura 1.1), principalmente por conta do surgimento das redes neurais convolucionais [11]. As CNNs elevaram o nível dos resultados obtidos em pesquisas de reconhecimento de imagens e detecção de objetos. Não à toa, surgiram na área competições para avaliar a qualidade de algoritmos e arquiteturas de redes neurais para estes tipos de tarefas, como o ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [12], que fornece base de dados para treinamento e avaliação destes tipos de redes.

1.2 Problema de pesquisa

Consequentemente, com o surgimento de tantos algoritmos e modelos de redes diferentes, navegar por tantos estudos para tentar encontrar um bom modelo pode se tornar uma tarefa massante.

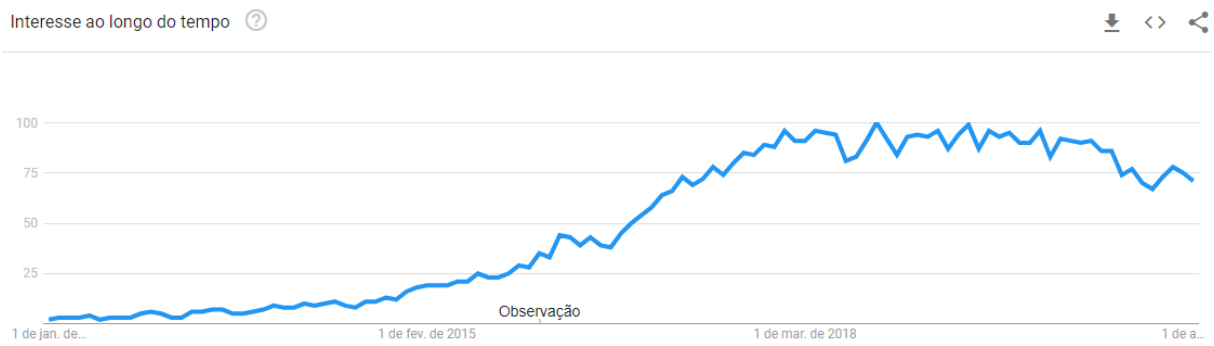

Figura 1.1: Incidência do termo *deep learning* em pesquisas Google a partir de 2012 (Fonte: [13]).

1.3 Objetivos

1.3.1 Objetivo Geral

O objetivo deste trabalho, é conseguir avaliar os modelos de reconhecimento de imagens mais conhecidos, para que se possa apontar quais modelos melhor se adaptam para ambientes ou objetivos específicos.

1.3.2 Objetivos específicos

- Identificar quais modelos alcançam maiores precisões na classificação de imagens.
- Identificar quais modelos executam a classificação de forma mais rápida e/ou menos custosa.
- Identificar se existe a influência do número de parâmetros totais da rede sobre a precisão.
- Identificar se existe a influência do tempo de execução da rede sobre a precisão.
- Identificar se existe a influência do número de camadas da rede sobre a precisão.

1.4 Estruturação do trabalho

Os seguintes capítulos deste trabalho se apresentam conforme: Capítulo 2, apresentando os principais conceitos teóricos utilizados pelos modelos contidos neste projeto, necessários para se tentar entender os resultados obtidos; Capítulo 3, que demonstra os procedimentos feitos para se testar os modelos e obter os dados de interesse para o desenvolvimento do projeto; Capítulo 4, onde são apresentados os resultados e feitas comparações entre

o desempenho de cada modelo em diferentes aspectos; e o capítulo 5, que levanta as conclusões a respeito dos resultados e comparações entre os modelos apontando quais se destacaram, além de levantar sugestões para trabalhos futuros.

Capítulo 2

Revisão Bibliográfica

Neste capítulo, será feita uma revisão bibliográfica dos principais conceitos que suportam este trabalho, e que foram essenciais para a conclusão dos resultados.

2.1 Neurônio Artificial e Redes Neurais Artificiais

Rede Neural Artificial (ANN) é um conceito base no aprendizado de máquina que simula o funcionamento das redes neurais biológicas de seres vivos. Uma rede neural biológica é composta por neurônios, que por sua vez são conectados entre si por axônios e dendritos, conexão esta denominada de sinapse [14].

As ANNs simulam o mecanismo biológico por meio de unidades computacionais (neurônios) que são conectados por meio de pesos (sinapses) [15]. Cada neurônio é alimentado por pelo menos uma entrada e seu peso correspondente (Figura 2.1), a entrada podendo ser um somatório caso haja mais de um dendrito. Na saída (axônio) ainda é aplicada uma função de ativação (abordado em 2.2). A Equação 2.1 representa a saída de um neurônio k com relação a suas x entradas e seus p pesos correspondentes, além da função de ativação (φ).

$$y_k = \varphi(\sum_{j=0}^{m} p_{kj} x_j) \tag{2.1}$$

Vários desses neurônios são agrupados em camadas, formando uma ANN com camadas sucessivas alimentando informação adiante em direção à saída, processo conhecido na aprendizagem de máquina como *feed-forward* [15]. Uma estrutura comum vista nos modelos estudados é a camada totalmente conectada (*fully-connected layer*), que seria uma camada em que todos os neurônios estão conectados com cada um dos neurônios da camada anterior.

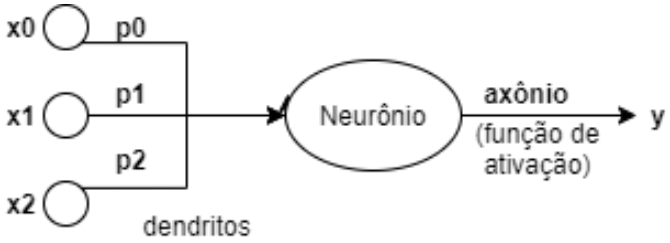

Figura 2.1: Neurônio artificial de uma ANN em detalhe com 3 entradas de peso p (dendritos) passando as informações adiante por meio do axônio.

2.2 Funções de ativação

Nesta seção é abordada as principais funções de ativação encontradas nas arquiteturas estudadas, bem como em redes neurais para classificação de imagens em geral.

Uma função de ativação é basicamente uma função aplicada na saída de cada um dos neurônios de forma a se atingir algum comportamento específico na rede.

2.2.1 Rectified Linear Units - ReLUs

Um tipo de função de ativação muita comum nas redes estudadas é a *Rectified Linear Unit* (ReLU) [1]. A ReLU é descrita pela Equação 2.2 (Figura 2.2), ou seja, para valores menores do que zero, o retorno é zero, caso contrário, o retorno será o próprio valor. Essa abordagem resulta em tempos menores na fase de treinamento para CNNs comparado ao uso de outras funções [1].

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

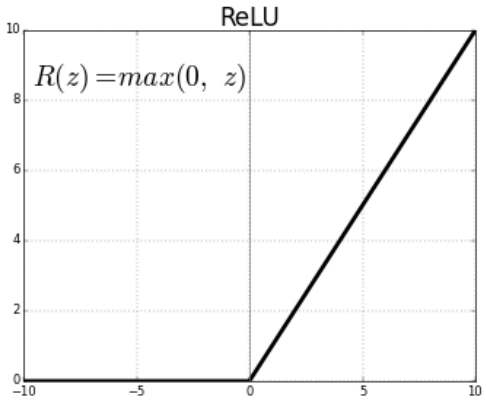

Figura 2.2: Demonstração gráfica da função ReLU, valores menores que zero são descartados (Fonte: [16]).

2.2.2 Softmax

O Softmax é outra função de ativação muito comum, em especial em redes neurais para classificação, e é aplicada na maioria das vezes na saída da rede [1, 6, 3]. Descrita pela Equação 2.3, basicamente, a função transforma cada um dos elementos i em uma distribuição probabilística, no intervalo (0, 1), com a soma de todos os valores totalizando 1.

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{2.3}$$

2.3 Backpropagation e Função de custo

A apredizagem efetiva da rede é feito por meio de um processo denominado *backpropagation*. Ao ser realizada a propagação das informações pela rede até sua saída (*feed-forward*), ao final, por meio da função de custo (*loss*), é calculado um erro baseado na resposta da rede em comparação com o resultado esperado [15]. Este erro é então propagado de volta ao início, atualizando os pesos e fazendo com que a rede aprenda baseado no quão errado estava a resposta.

2.4 Redes Neurais Convolucionais

As Redes Neurais Convolucionais (ou do inglês, Convolutional Neural Networks - CNN), estudo inicialmente apresentado por Yann LeCun em 1989 [11], deu início a um grande avanço nas pesquisas sobre reconhecimento de imagens e detecção de objetos.

Para uma rede neural ser considerada convolucional, é preciso que a mesma contenha pelo menos uma camada de operações de convolução, ao invés da tradicional multiplicação de matrizes [17]. A Equação 2.4 descreve a teoria por trás de uma operação de convolução, sendo definida como a integral do produto de duas funções f e g, sendo g oposto e deslocado de τ [18]. Na prática, o que ocorre nesta camada é um cálculo de convolução entre um filtro (ou *kernel*) de tamanho N x N sobre uma fração de mesmo tamanho da imagem, como na Figura 2.3, até que a operação seja feita sobre todos os *pixels*.

$$(f * g)(t) := \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau \tag{2.4}$$

Estes filtros, aplicados sobre frações da imagem, geram como resultado uma série de características que são armazenadas em um mapa, que contém os resultados de todas as operações de convolução daquele filtro específico. Além disso, a image é percorrida de acordo com o valor definido por um salto (*stride*). Se salto = 1, por exemplo, a janela

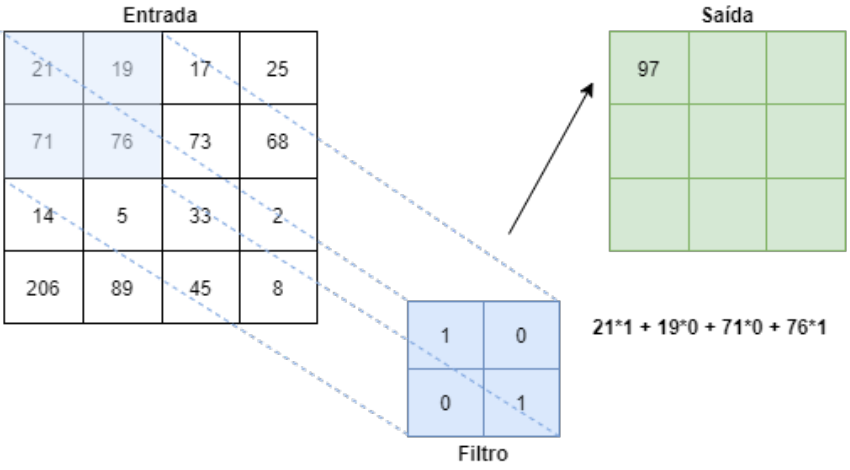

Figura 2.3: Exemplo de cálculo de convolução com filtro 2 x 2 e saltos de 1.

irá percorrer a imagem um *pixel* por vez, em outro caso, se salto = 2, isso significa que a janela do filtro se movimentará por dois *pixels* de cada vez, tanto horizontal quanto vertical.

2.4.1 Pooling

Além da camada de convolução, em uma CNN é comum, porém não obrigatório, existir uma camada de *pooling* aplicada logo após os mapas de características. O *pooling* é uma função que gera como resultado uma síntese dos valores de uma região do mapa de acordo com a característica daquele *pooling*. Por exemplo, um *max pooling* seria o *pooling* retornando o valor máximo dentro daquela região, como mostra na Figura 2.4.

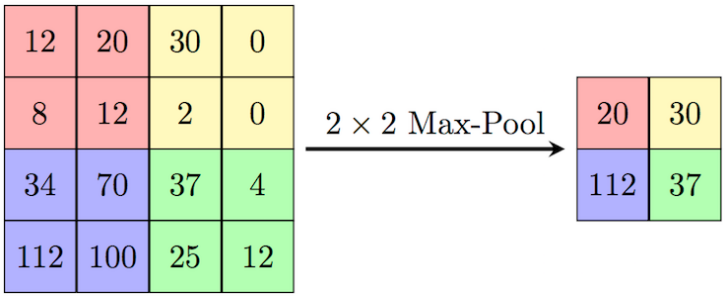

Figura 2.4: Max-pooling 2 x 2 com stride = 2 (Fonte: [19]).

O *pooling* em geral reduz o número de parâmetros totais da rede, que por consequência, reduz o consumo computacional tanto para treinar quanto para utilizá-la [20]. Este é um dos principais motivos pelos quais CNNs consomem menos poder computacional comparada a redes sem esta característica. Além disso, é por conta do *pooling* que CNNs, mesmo

que de forma limitada, são capazes de identificar características em imagens de forma independente de localização específica, propriedade denominada invariância à translação local [17].

2.5 Modelos de referência

Nesta seção será apresentado cada um dos modelos que foram avaliados no projeto, fazendo uma abordagem de suas principais características, para que se possa tentar entender e explicar os resultados obtidos.

2.5.1 Alexnet

O modelo Alexnet, que foi o vencedor do *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC) 2012 para o *top-5*, foi proposto no mesmo ano por Alex Krizhevsky [1], contendo cerca de 61 milhões de parâmetros totais, número menor apenas que o VGG19 [2] dentre os modelos estudados.

A arquitetura do Alexnet é estruturada de cinco camadas convolucionais seguidas de mais três camadas totalmente conectadas (vide Figura 2.5), em que as duas primeiras camadas e a quinta, são cada uma, sucedidas por uma camada de *max-pooling*. A terceira e quarta camadas são diretamente conectadas sem nenhuma outra estrutura entre as mesmas, e a última camada da rede é alimentada em um *softmax* de 1000 unidades (um para cada rótulo).

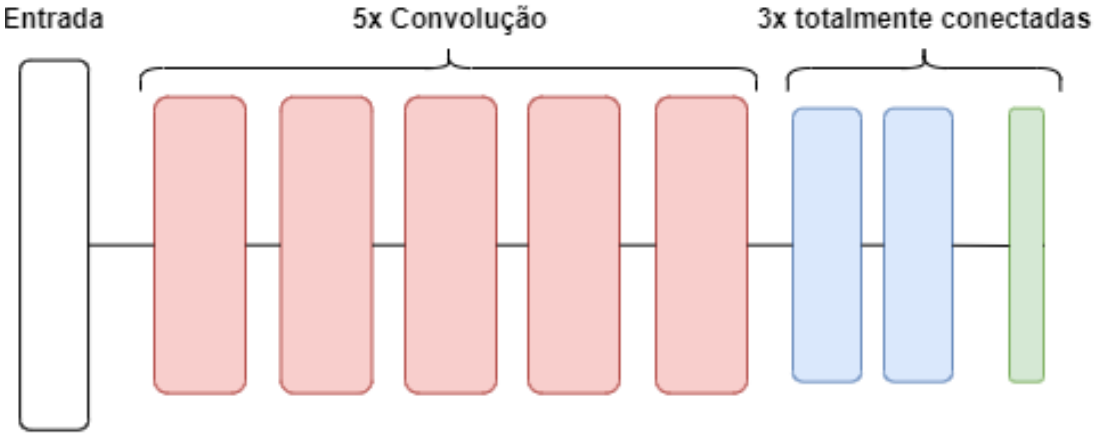

Figura 2.5: Arquitetura Alexnet como descrito em [1]. 5 CNNs em sequência seguidas de 3 camadas totalmente ligadas (a terceira em verde representa a saída de 1000 unidades por *softmax*).

A Alexnet também utiliza ReLUs como função de ativação nas primeiras sete camadas, e aplica algumas técnicas em seu treinamento para reduzir *overfitting*, um problema comum no treinamento de redes neurais, que é quando a rede funciona muito bem em situações específicas, que geralmente envolvem os dados utilizados para treino, mas falha para dados nunca apresentados [15].

Uma das técnicas aplicadas para combater o *overfitting* é o *dropout* [21], que consiste em tornar zero a resposta de um neurônio, a uma probabilidade de 0.5. Isso faz com que este neurônio não contribua para o *feedforward* da rede e nem para a *backpropagation*, e efetivamente muda a arquitetura da rede para cada entrada (vide Figura 2.6).

Além disso, também é aplicado na Alexnet um conceito denominado *overlapping pooling* (vide Figura 2.7), que é basicamente quando a janela de cada *pooling* se sobrepõe com a janela anterior, o que significa dizer que o *stride* é menor do que o tamanho de uma dimensão da janela do *pooling*.

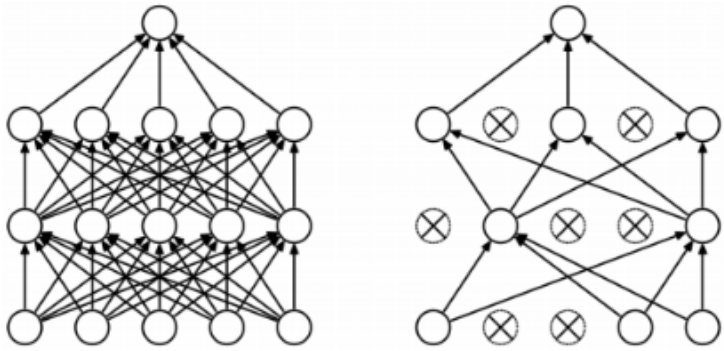

Figura 2.6: À esquerda o esquema usual de uma rede. À direita o esquema com *dropout*, com alguns neurônios inativos, que não contribuirão na propagação das informações (Fonte: [22]).

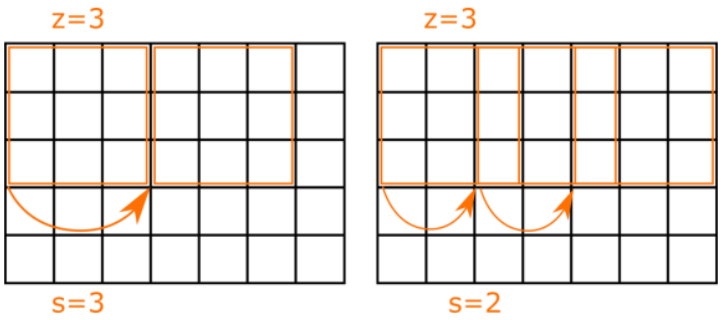

Figura 2.7: À esquerda o *pooling* normal. À direita o *overlapping pooling*, com janelas 3x3 e saltos de 2, fazendo com que cada passo se sobreponha com o passo anterior (Fonte: [22]).

2.5.2 VGG

O VGG é um modelo baseado nos variados estudos que relacionam precisão de rede com a sua profundidade, foi o modelo vencedor do ILSVRC 2014 em localização e tem como destaque a simplicidade da arquitetura e dos conceitos aplicados em sua construção [2].

Basicamente, o VGG utiliza filtros pequenos 3 x 3 com 1 *pixel* de *stride*, menor do que o AlexNet no qual foi baseado, que utilizava filtros 11 x 11 e *stride* 4 [1], além de convoluções 1 x 1 na entrada da rede para a linearidade dos canais, resultando em uma rede extensa, que favorece a precisão. O modelo ainda aplica ReLUs para cada convolução seguidos de *max-pooling*. Por fim, são utilizadas três camadas totalmente ligadas, duas de 4096 e uma de 1000 canais, que alimentam o *softmax* na saída (Figura 2.8).

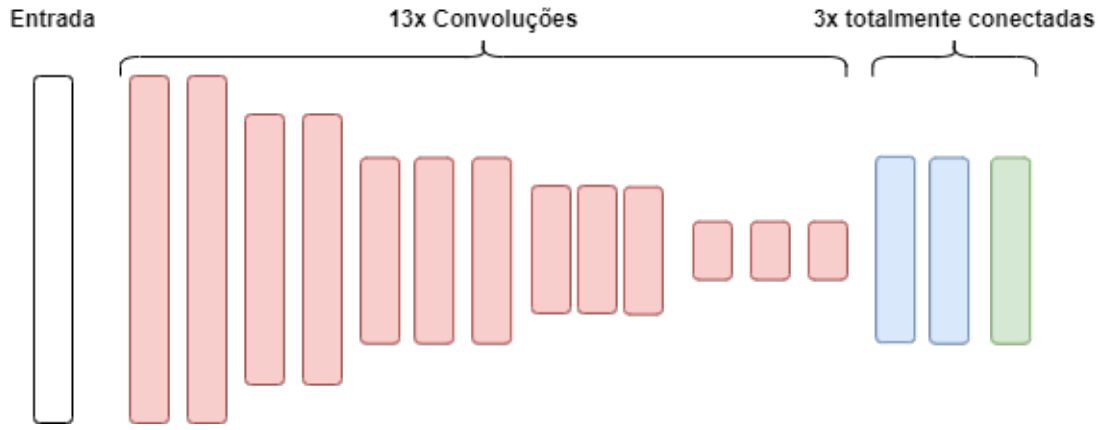

Figura 2.8: Esquema de uma VGG16 (16 camadas) [2]. As CNNs em sequência causam este aspecto de afunilamento dos mapas de características. As 3 últimas camadas em sequência são totalmente ligadas, com a última em verde representando as 1000 unidades para o *softmax*.

2.5.3 GoogLenet

A Googlenet foi o modelo vencedor do ILSVRC 2014. O modelo é composto por cerca de sete milhões de parâmetros e é estruturado por nove módulos *inception* (descritos a seguir), quatro camadas convolucionais, quatro camadas de *max-pooling*, três de *average-pooling*, cinco camadas totalmente ligadas e três camadas de saída aplicadas por função de ativação *softmax* para o classificador [3].

A peculiaridade da Googlenet está em seus módulos *inception* [23] Figura 2.9. Estes módulos permitem que sejam aplicados vários filtros de tamanhos diferentes em um mesmo bloco da imagem de uma só vez, antes de ser passado para a próxima camada da rede, de modo a tentar detectar melhores correlações entre os *pixels* mais próximos.

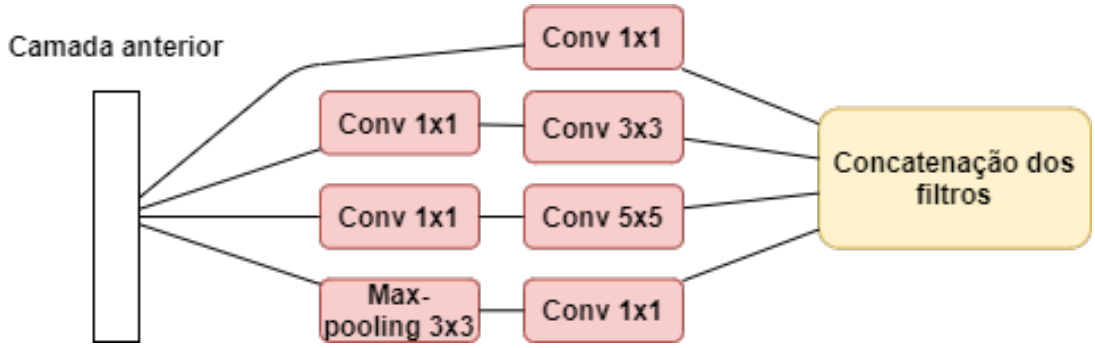

Figura 2.9: Ilustração de um módulo *inception* presente no GoogLeNet, com duas camadas de convoluções paralelas. Os mapas de características gerados são concatenadas em sequência como uma pilha e passados à próxima camada.

Além disso, outras características da Googlenet incluem o uso de *dropout* [21] nas camadas totalmente conectadas e ativação ReLU nas camadas convolucionais. Apesar de ser uma arquitetura mais profunda, com 22 camadas (Figura 2.10), a Googlenet tem menos parâmetros que arquiteturas menores como a Alexnet e a VGG16 [2].

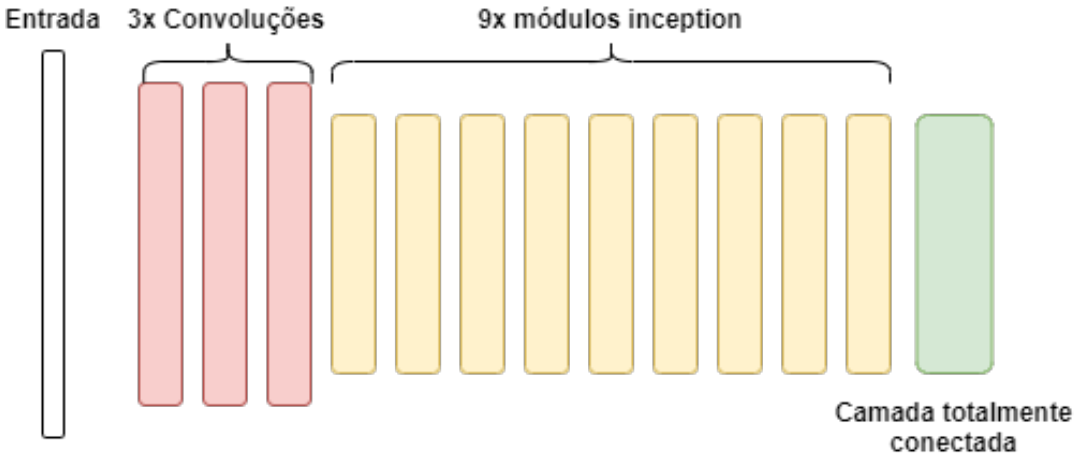

Figura 2.10: Esquema do GoogLeNet conforme [3]. 3 convoluções em sequência seguida de 9 módulos *inception* (cada módulo contém 2 camadas) e uma camada totalmente conectada de 1000 unidades aplicada por um *softmax*.

2.5.4 InceptionV3

O modelo Inception V3 [4] usou como base os conceitos de *inception* do Googlenet além de outras técnicas desenvolvidas após o ILSVRC 2014. O Inception V3 inclui várias melhorias comparado aos modelos *inception* anteriores, como *label-smoothing regularization* (LSR) juntamente com um classificador auxiliar.

O LSR é aplicado com o intuito de reduzir *overfitting* tornando o modelo "menos confiante". Basicamente, o LSR identifica quando existe predições fora da realidade para rótulos específicos em pontos relativamente ainda no início da rede, fazendo uma regularização dos valores e distribuindo melhor as probabilidades entre os outros rótulos. Isto, aliado a um *auxiliary classifier* [3], permite a uma rede extensa como a Inception V3 ser treinada com bancos de dados de tamanho considerável (da ordem de dezenas de milhões) sem sofrer *overfitting*.

A estrutura da Inception V3 (Figura 2.11) é composta por uma série de blocos que incluem, camadas convolucionais, *average pooling*, *max-pooling*, *dropout*, além de camadas totalmente ligadas e *softmax*, extendendo-se por 42 camadas contendo quase 30 milhões de parâmetros.

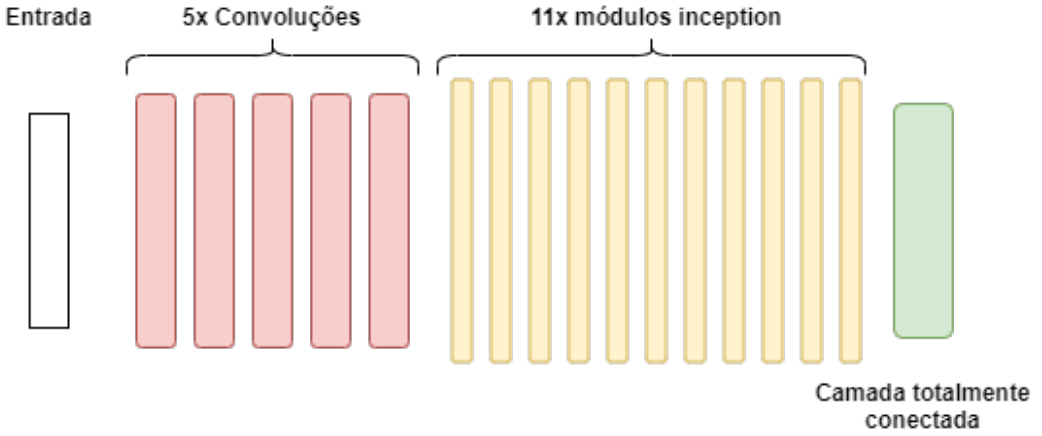

Figura 2.11: Esboço do esquema do InceptionV3 abstraído de detalhes. Composto por 5 CNNs em sequência seguido por 11 módulos *inception*, terminando com uma camada totalmente conectada de 1000 unidades com *softmax* pra classificação dentre os 1000 rótulos (a arquitetura pode ser vista de forma detalhada em [4]).

2.5.5 ResNet

A ResNet foi o modelo vencedor do ILSVRC 2015 e é uma arquitetura que introduziu soluções para redes muito extensas utilizando um conceito chamado aprendizagem por resíduo [5].

Na construção de redes neurais, caso a estrutura seja muito extensa, em certo ponto, o gradiente que é utilizado para calcular a função de custo da rede converge para zero, impedindo que os pesos de cada neurônio sejam atualizados, interrompendo a apredizagem da rede neural, este problema é conhecido como desaparecimento do gradiente.

Na arquitetura da ResNet, são incluídas conexões entre a entrada e a saída entre camadas, em que algumas são puladas (Figura 2.12). Por estas conexões passam os dados

residuais, preservando os valores dos gradientes até o fim da rede, que então são usados no *back-propagation* para o processo de aprendizagem da rede neural.

A quantidade de camadas e parâmetros da ResNet varia de acordo com a versão, abrangendo de 18 até 1202 camadas e de 270 mil até cerca de 19 milhões de parâmetros, incluindo camadas convolucionais, *average-pooling*, camadas totalmente conectadas e *softmax*.

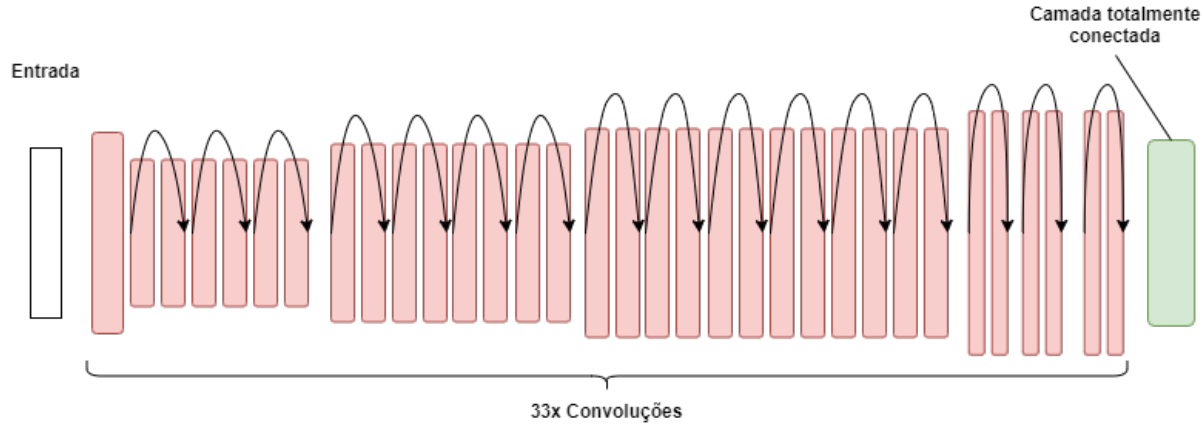

Figura 2.12: Esquema de uma ResNet34 conforme [5]. Contém 33 CNNs em sequência. O tamanho dos filtros aplicados é o que gera esse formato afunilado dos mapas de características. As setas entre as camadas representam as conexões residuais. Assim como nos modelos vistos até aqui, ao final, há uma camada totalmente conectada de 1000 unidades ativada por *softmax*.

2.5.6 DenseNet

O modelo DenseNet surgiu com o objetivo de solucionar o problema do desaparecimento de gradiente (mencionado em 2.5.5) na aprendizagem para arquiteturas muito profundas [6], assim como o ResNet visto anteriormente.

A ideia chave empregada pela DenseNet é, em cada camada, os mapas de características são passados como entrada não só para a camada subsequente, mas para todas as camadas até o final da rede, em uma estrutura denominada bloco *dense* (Figura 2.13). Esta abordagem ajuda a preservar o gradiente pela extensão da rede, além de poupar parâmetros por não ser mais necessário para a rede reaprender mapas redundantes. Não à toa, a DenseNet, apesar de ser uma rede que ultrapassa as 100 camadas, possui uma quantidade consideravelmente menor de parâmetros totais do que a VGG16 [2] por exemplo, com apenas 16 camadas.

A DenseNet reduz o número de parâmetros ainda mais ao dividir a rede em agrupamentos de camadas *dense*, cada agrupamento separado por uma camada de transição

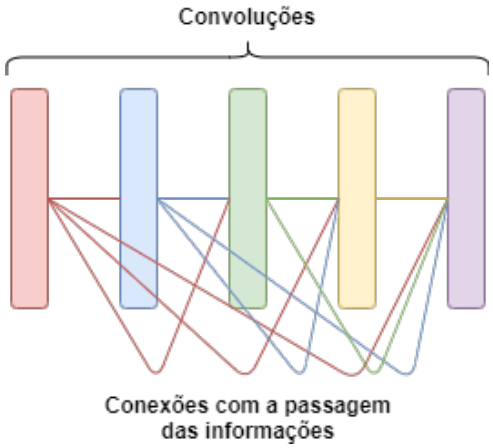

Figura 2.13: Ilustração de um bloco *dense*. Os dados de uma camada são propagados diretamente para todas as camadas subsequentes..

composta por uma convolução 1 x 1 e por um *average pooling* (Figura 2.14), efetivamente reduzindo também o tamanho dos mapas de características. Um esboço geral da DenseNet-121 pode ser visto na Figura 2.15.

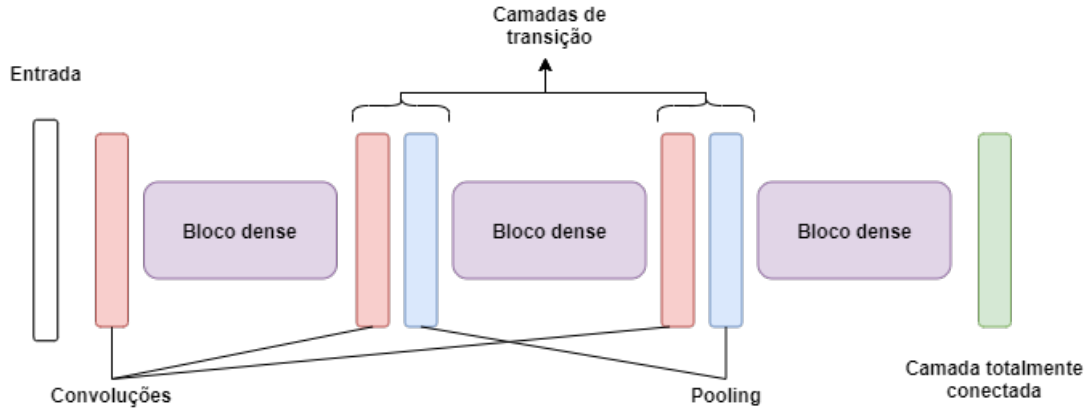

Figura 2.14: Camadas de transição da DenseNet. Ficam localizadas entre blocos *dense* e são compostas por uma convolução 1x1 e um *average pooling*.

2.5.7 MobileNetV2

O MobileNet-V2 é um modelo construído com o intuito de executar em dispositivos móveis e sistemas embarcados, sendo assim, um modelo preocupado principalmente com a utilização de recursos computacionais [24]. O modelo apresenta duas propriedades principais, a utilização da técnica de *depthwise separable convolution* [25] e uma estrutura residual invertida contendo a primeira técnica.

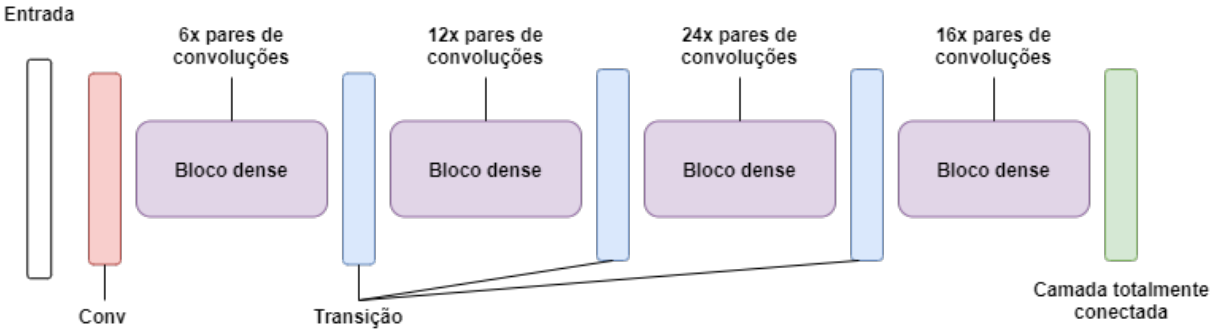

Figura 2.15: Esboço da estrutura da DenseNet-121 conforme [6]. Contém 4 blocos *dense*, cada um composto por conjuntos sucessivos de 2 convoluções (1x1 e 3x3), cada bloco intercalado por uma região de transição. A rede possui uma CNN logo na entrada e uma camada totalmente conectada na saída com *softmax*.

A *depthwise separable convolution* é uma convolução que acontece em duas etapas. Na primeira, é utilizado um filtro para cada canal da imagem, ao invés de um filtro sendo aplicado por todos os canais de uma só vez (Figura 2.16). Na segunda etapa, é utilizado um filtro 1 x 1 com a profundidade relativa à quantidade de canais da imagem (Figura 2.17). Esta técnica reduz consideravelmente a quantidade de cálculos realizados na convolução pela rede, reduzindo o custo computacional.

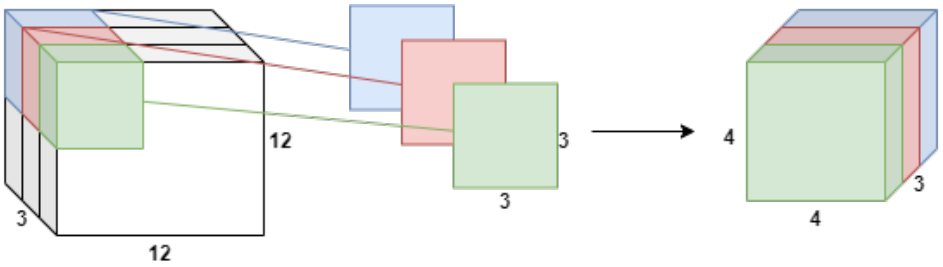

Figura 2.16: *Depthwise separable convolution* Etapa 1: 1 filtro para cada canal da imagem.

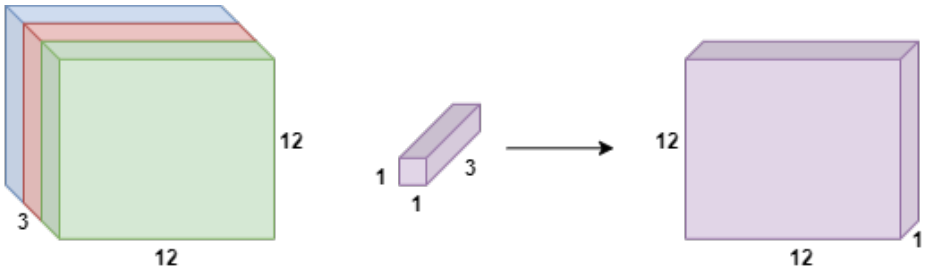

Figura 2.17: *Depthwise separable convolution* Etapa 2: Filtro 1 x 1 que é aplicado sobre todos os canais de uma só vez.

A estrutura residual invertida do MobileNet-V2, faz o processo de *depthwise separable convolution* invertendo a ordem das duas etapas, e insere a conexão do gradiente entre as camadas de *bottleneck* da rede. A estratégia é tentar obter uma rede profunda o suficiente para garantir bons resultados com baixo custo computacional. (Nota: O estudo referente ao MobileNetV2 não deixa claro exatamente como é organizado certas estruturas e por isso o esboço da arquitetura foi deixada de fora.)

2.5.8 ShuffleNetV2

O ShuffleNet-V2, como o modelo anterior, também possui como alvo dispositivos móveis e sistemas embarcados. O foco deste modelo no entanto é o tempo de execução, que acredita-se ser minimizado pela redução do custo de acesso à memória [7]. O modelo se baseia em quatro princípios para se atingir tal objetivo: o primeiro é tornar o número de canais da entrada igual a quantidade de canais da saída, o que reduz o custo de acesso à memória; o segundo, é reduzir o uso de convoluções em grupo [26]; o terceiro, é evitar fragmentação da rede para que não haja redução do grau de paralelismo computacional, que reduziria a eficiência da rede; e o quarto é evitar operações elementares, que aumentam o tempo de acesso à memória.

O modelo se estrutura da seguinte forma (de acordo com a Figura 2.18). É realizada uma operação de separação dos canais [26], que divide os canais da imagem entre dois caminhos. O primeiro caminho atua como identidade dos canais e é simplesmente ligado ao final do módulo, satisfazendo o terceiro princípio. O segundo caminho, realiza três operações de convolução, de modo que a quantidade de canais é preservada da entrada até a saída ao final dessas operações, satisfazendo o primeiro princípio. As duas convoluções 1 x 1 do segundo ramo não são convoluções em grupo, o que satisfaz o princípio dois, e operações elementares como ReLU e concatenação de mapas são minimizadas apenas para o segundo caminho, satisfazendo o quarto princípio. Ao final do módulo, é aplicada uma operação de embaralhamento dos canais [26] para auxiliar o fluxo de informações entre os canais das características.

(Nota: Assim como no caso do MobileNetV2, o estudo referente ao ShuffleNetV2 não deixa claro exatamente como é organizado certas estruturas e por isso o esboço da arquitetura foi deixada de fora.)

2.5.9 SqueezeNet

O SqueezeNet é um modelo baseado no princípio de se manter uma rede compacta para a minimização de custos computacionais [8]. A arquitetura é baseada em três estratégias: a substituição do padrão de filtros 3 x 3 por filtros 1 x 1; a redução da quantidade de canais

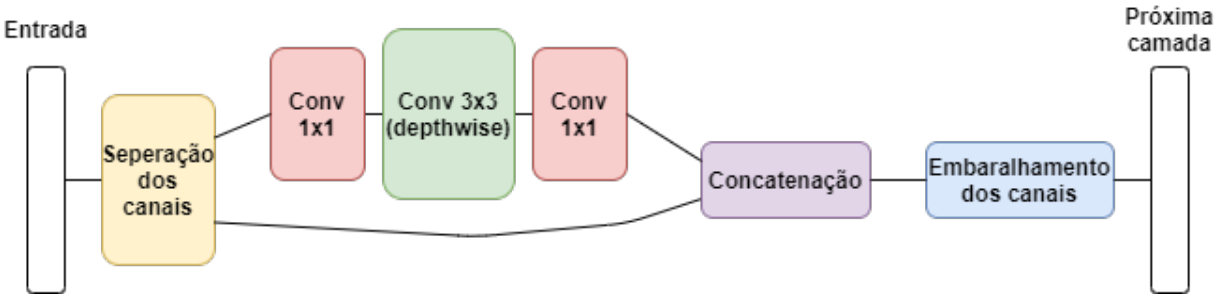

Figura 2.18: O bloco com as sequências de operações de separação e emparalhamento de canais do ShuffleNetV2 conforme [7].

de entrada para filtros 3 x 3 usando camadas de compressão (*squeeze layers*); e reservar a redução de amostragem mais para o fim da rede para que camadas convolucionais tenham maiores mapas de ativação.

A ideia das estratégias 1 e 2 é de reduzir o número de parâmetros da rede e ao mesmo tempo preservar a precisão, enquanto a estratégia 3 tenta maximizar a precisão com uma quantidade de parâmetros limitado.

Estas três estratégias são implementadas por meio de um módulo especial do SqueezeNet denominado módulo de fogo (*Fire Module*) (Figura 2.19 e Figura 2.20). Este módulo aplica as camadas de compressão de filtros 1 x 1 passando o conteúdo para as camadas subsequentes e discretamente expandindo a quantidade de canais de uma camada para outra.

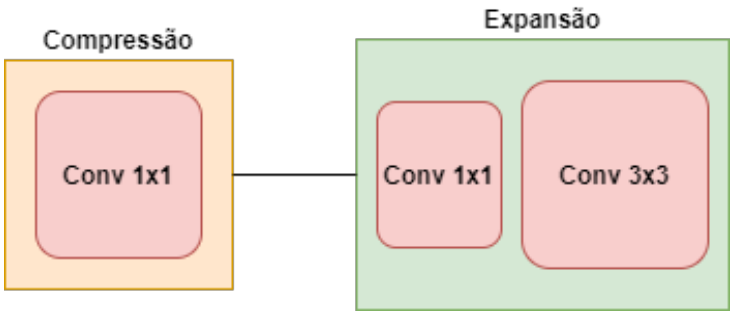

Figura 2.19: Esboço do módulo de fogo conforme [8]. O bloco da esquerda realiza a compressão por meio de convoluções 1x1, enquanto no bloco de expansão é realizada convoluções 1x1 e 3x3.

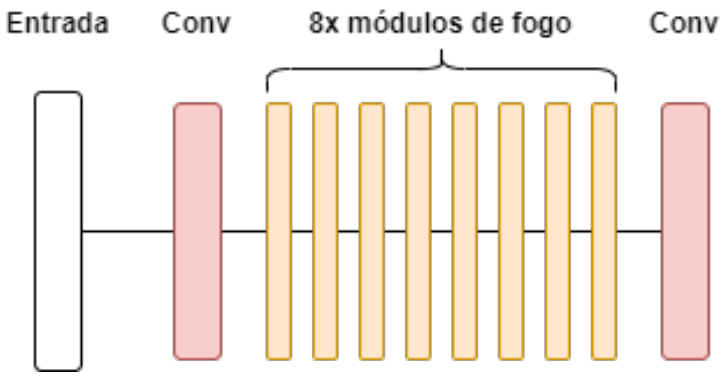

Figura 2.20: Esquema da arquitetura do SqueezeNet conforme [8]. Contém uma CNN no início da rede seguido de uma sequência de 8 módulos de fogo. Ao final existe uma CNN sem a presença usual de uma camada totalmente conectada como visto nos outros modelos. Esta CNN já gera diretamente a saída de 1000 componentes com a aplicação do *softmax* para os 1000 rótulos.

Capítulo 3

Metodologia e Desenvolvimento

Neste capítulo será apresentado os procedimentos adotados para a realização do trabalho, bem como as decisões tomadas e outros detalhes gerais do desenvolvimento.

3.1 Trabalhos relacionados

Existem algumas pesquisas na área que realizaram comparações entre rede neurais para classificação e detecção de objetos, porém com objetivos um pouco mais específicos. Os trabalhos [27, 28], por exemplo, compararam os desempenhos de diversas redes presentes neste estudo na detecção de doenças em plantas. Os dados utilizados são de folhas de, principalmente, pés de tomate com presença de fungos, infecção por bactérias ou pragas.

3.2 Bases de imagens utilizados na avaliação

Duas bases de dados foram utilizadas para avaliar os modelos, o banco de validação da *ImageNet* [12] utilizado no ILSVRC de 2012 composto por 50000 imagens e um pequeno banco de 100 imagens manualmente coletadas da *web*.

Foi utilizado o banco de imagens do ILSVRC de 2012, tendo em vista que todos os modelos, com exceção do GoogLeNet [3], foram treinados e validados utilizando esta versão. O GoogLeNet foi treinado e validado utilizando o banco do ILSVRC de 2014, mas as diferentes versões todas se baseiam na original, de 2010 [12].

3.2.1 ImageNet

A base de dados da *Imagenet* [12] é a direção padrão que pesquisadores tomam quando buscam uma quantidade grande de imagens tanto para treinar quanto para validar seus modelos de visão computacional [29]. O banco de dados consiste de mais de 15 milhões

de imagens classificadas entre mais de 22 mil rótulos e foi construída por meio da coleta extensiva de imagens da *web* com o auxílio de *crowd-sourcing*.

Os administradores da *ImageNet* tem o costume de realizar competições relacionadas a visão computacional, o ILSVRC. Nela, os pesquisadores testam seus modelos quanto à precisão na classificação de imagens e detecção de objetos.

3.2.2 Amostras manualmente coletadas

Foram coletadas manualmente 100 imagens da *web* aleatoriamente, cada uma de um rótulo diferente, para testar os modelos em uma aplicação mais prática, tomando o cuidado de verificar que nenhuma das 100 imagens já estivesse na base de dados utilizada da *ImageNet*.

Esta verificação foi feita manualmente, imagem por imagem, puxando da base de dados todas as imagens daquele rótulo específico.

3.2.3 Tipos das classes (rótulos)

Cada modelo classifica cada uma das imagens em 1 das 1000 classes diferentes. As classes variam bastante quanto ao tipo, abrangendo de seres vivos até objetos isolados, como uma roda de carro, por exemplo.

Para animais, as classes se subdividem em espécies diferentes para um mesmo ser vivo, como várias espécies de cobras, aves, peixes, raças de cachorro, etc.

Para objetos em geral e outras coisas inanimadas, existem alguns agrupamentos, como frutas, pratos de comida, bebidas, instrumentos musicais, dispositivos eletrônicos, veículos automotivos e não-automotivos, artigos esportivos, construções, paisagens, etc.

3.3 Escolha dos modelos de referência

Existe uma quantidade considerável de modelos de referência que poderiam ser escolhidos para se avaliar, mas por limitações da biblioteca, os modelos escolhidos foram os que são disponibilizados por tal. Foi considerada a hipótese de se utilizar mais de uma biblioteca, mas foi constatado que bibliotecas diferentes geravam resultados distintos para o mesmo modelo, sendo que o objetivo é que todos os modelos sejam executados sob iguais condições, logo, optou-se por utilizar apenas uma biblioteca.

3.4 Escolha da biblioteca

Como dito na seção anterior, a escolha da biblioteca impacta diretamente nos modelos que foram escolhidos. As duas opções eram utilizar o Pytorch [30] ou o Keras [31] (por meio do Tensorflow).

O Pytorch foi optado por conter uma maior quantidade de modelos, além de modelos mais conhecidos, e ainda, por disponibilizar uma API mais simplificada, incluindo rotinas de pré-processamento.

3.5 Especificação da máquina e utilização de recursos computacionais

As especificações da máquina utilizada incluem uma CPU Intel Core i7-8700 de 6 núcleos e 12 *threads*, com *clock* base de 3.2 GHz e memória *cache* de 12 MB [32], além de memória RAM de 24 GB.

Uma observação importante é que a utilização de GPU (usual em projetos envolvendo CNNs) foi descartada na execução do projeto por um problema com o *software* de gerenciamento de núcleos CUDA da máquina.

3.6 Treinamento das redes

Todas as redes já vêm pré-treinadas pelo Pytorch. De acordo com a documentação [33], todos os modelos fornecidos pela API foram treinados e validados com as imagens do banco da *ImageNet*.

3.7 Pré-processamento das imagens e verificação dos resultados

Os modelos foram construídos para serem alimentados com imagens em um formato padrão, que não precisa ser obrigatoriamente empregado, porém, para o melhor desempenho, é recomendado que seja feita o pré-processamento de cada imagem antes de inserí-las.

Todos os modelos utilizados recomendam o mesmo processo de formatação conforme a documentação do Pytorch [33]. Espera-se que as imagens tenham tamanho 224 x 224, padrão RGB e sejam normalizadas conforme descrito. As imagens são carregadas em um intervalo [0, 1] e normalizadas utilizando-se dois parâmetros ($mean = [0.485, 0.456, 0.406]$

e *std* = [0.229, 0.224, 0.225]). Para o Inception-V3, a única distinção é que as imagens sejam de tamanho 299 x 299.

Como dito na Seção 3.4, a API do Pytorch simplifica bastante este processo, sendo necessário apenas a chamada da função *transform.Compose()* com os parâmetros sugeridos.

Os resultados da classificação de cada imagem por cada modelo é verificado comparando a saída da rede. Esta saída indica os 5 rótulos mais prováveis para aquela imagem. Se o *groundtruth* daquela imagem estiver no rótulo mais provável, o *top-1* é satisfeito. Se o *groundtruth* estiver dentre qualquer um dos 5 rótulos mais prováveis, o *top-5* é satisfeito.

3.8 Dados coletados

Foram coletados os dados de precisão para o top-1 e top-5, o tempo total de execução, o tempo médio por imagem, o número de camadas da arquitetura e o número total de parâmetros de cada modelo para uso nas análises e levantamento de conclusões.

Vale ressaltar que foi levantada a hipótese de se utilizar FLOPs como métrica de análise juntamente com as outras, porém, vários modelos não fornecem essas dados em seus respectivos estudos.

3.9 Análise dos dados coletados

Em geral, o objetivo é fazer várias correlações entre as métricas envolvidas entre os modelos:

- **Precisão x Número total de parâmetros:** Tentar observar a existência ou não da influência do número de parâmetros sobre a precisão do modelo.
- **Precisão x número de camadas:** A influência do número de camadas sobre a precisão da rede.
- **Precisão x tempo de execução:** Observar a relação entre a precisão das redes e o tempo que elas levam para classificar as imagens.

Capítulo 4

Resultados

Neste capítulo será apresentado os resultados obtidos pelos testes feitos com as bases de dados da *ImageNet* e coletados manualmente.

Vale notar que, para buscar uma avaliação mais equilibrada entre os modelos, dentre os modelos que possuem várias versões de arquiteturas (por exemplo, ResNet18, ResNet34, ResNet50 e etc.), foram escolhidas as que apresentam resultados mais próximos dos outros modelos, para uma comparação mais justa. Ao final deste capítulo, será feita uma observação sobre a hipótese de se usar versões melhores destas mesmas redes. Estes modelos foram: ResNet50, VGG16 e o DenseNet121.

4.1 Precisões na classificação top-1 e top-5

4.1.1 ImageNet

Cada modelo obteve o seguinte desempenho (Tabela 4.1) para as 50000 imagens de validação da *ImageNet*, sendo o Inception-V3 o modelo mais preciso, em contraste ao AlexNet:

Tabela 4.1: Resultados para a ImageNet (ordenado pelo top-1).

Modelo	**top-1 acc**	**top-5 acc**	**tempo(s)**	**params(M)**
Inception-V3	77.2%	93.5%	3660	27
ResNet50	76.1%	92.8%	3032	25
DenseNet121	74.4%	91.9%	2693	7.9
MobileNet-V2	71.8%	90.2%	4495	3.5
VGG16	71.5%	90.3%	4802	138
GoogLeNet	69.7%	89.5%	2223	6.6
ShuffleNet-V2	69.3%	88.3%	538	2.2
SqueezeNet	58%	80.4%	1030	1.2
AlexNet	56.5%	79%	663	61

Pela Tabela 4.1 é possível observar que um número maior de parâmetros não necessariamente traduz em uma rede mais precisa, isso fica ainda mais claro na Figura 4.1, apesar da VGG16 possuir uma quantidade consideravelmente maior de parâmetros, o modelo ainda fica atrás de quatro outros. Isto é ainda mais evidente se observarmos o AlexNet, sendo o modelo menos preciso, apesar do segungo maior em parâmetros.

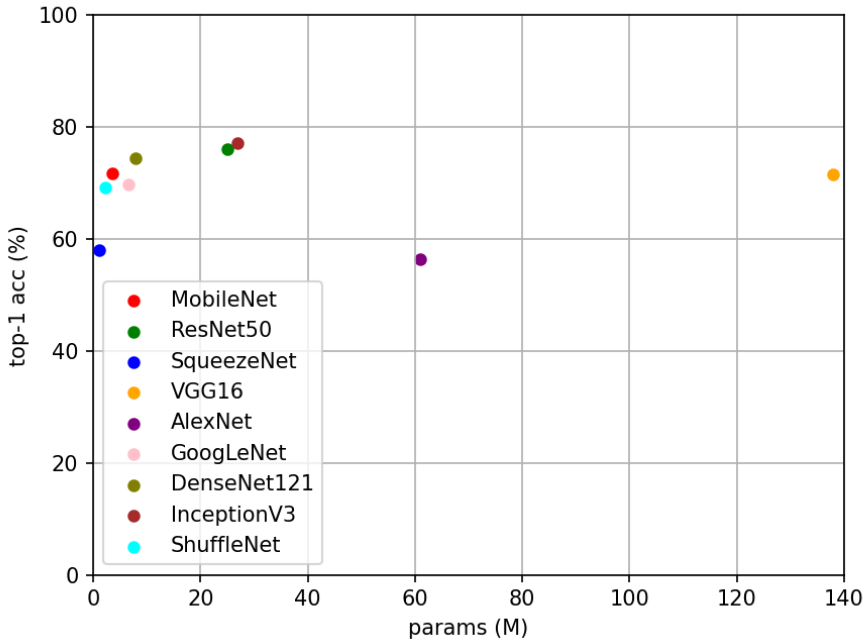

Figura 4.1: Precisão top-1 x params (ImageNet).

Pela Figura 4.2, podemos observar que o número de parâmetros não influencia diretamente no tempo de execução da rede. O AlexNet possui 61 milhões de parâmetros atrás apenas do VGG16, porém, só não é mais rápido que o ShuffleNet-V2. Em contra-partida, o MobileNet-V2, com apenas 3.5 milhões de parâmetros, é quase 7 vezes mais lento que o AlexNet.

A estratégia do ShuffleNet-V2 de focar no baixo tempo de execução, utilizando-se de sua arquitetura baseada nos 4 princípios (visto em 2.5.8), é comprovada aqui com sucesso. O modelo é capaz de entregar uma precisão próxima aos mais bem colocados, praticamente na casa dos 70%, com o menor tempo possível entre os concorrentes.

Podemos tentar relacionar ainda os resultados de precisão com a quantidade de camadas da rede. Na Figura 4.3, pode ser observado que a tendência de redes mais profundas gera bons resultados, mas não é obrigatório, como demonstra o VGG16 e o GoogLeNet, compondo muito menos camadas do que o DenseNet121 ou o ResNet50, por exemplo. Isto mostra a eficiência dos módulos *inception*, presentes não só no GoogLeNet, como também

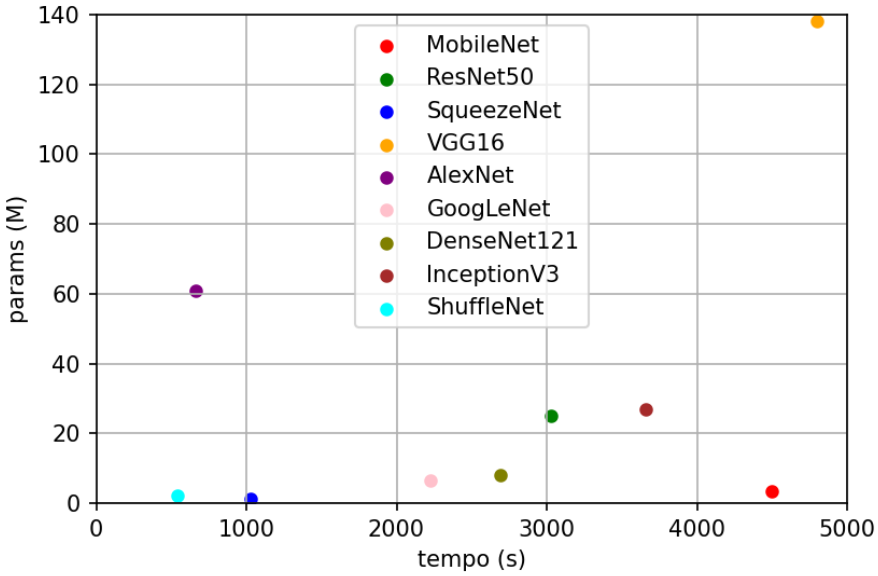

Figura 4.2: params x tempo (ImageNet).

no modelo mais preciso, Inception-V3, além de demonstrar a efetividade da arquitetura VGG, que emprega apenas conceitos simples de *deep learning* mas mostra ser um modelo eficiente com acertos acima dos 70%.

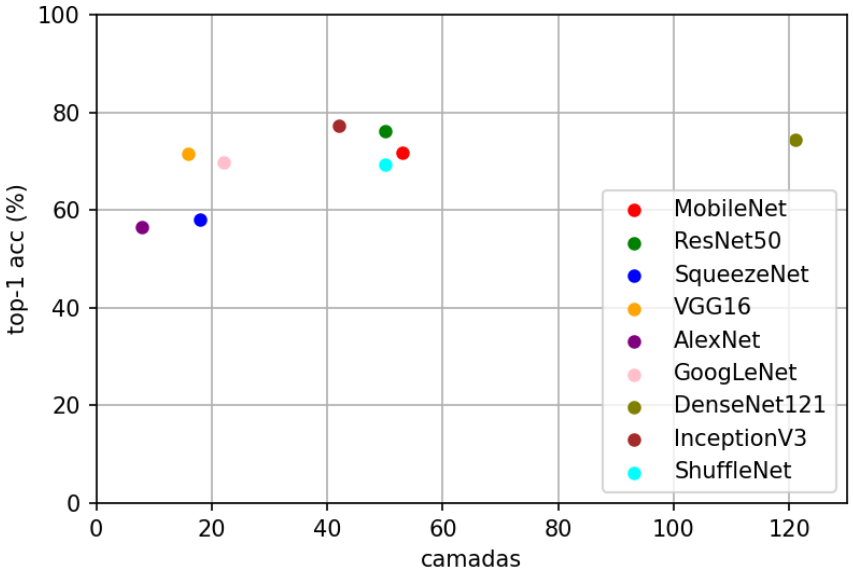

Figura 4.3: Precisão top-1 x camadas (ImageNet).

Quanto ao tempo de execução influenciar ou não a precisão, conforme a Figura 4.4 é

observado que um modelo que demora mais não necessariamente é mais preciso. Isto é evidente com o VGG, de maior tempo, mas quinto em precisão, enquanto o DenseNet121 tem tempo muito menor sendo mais efetivo.

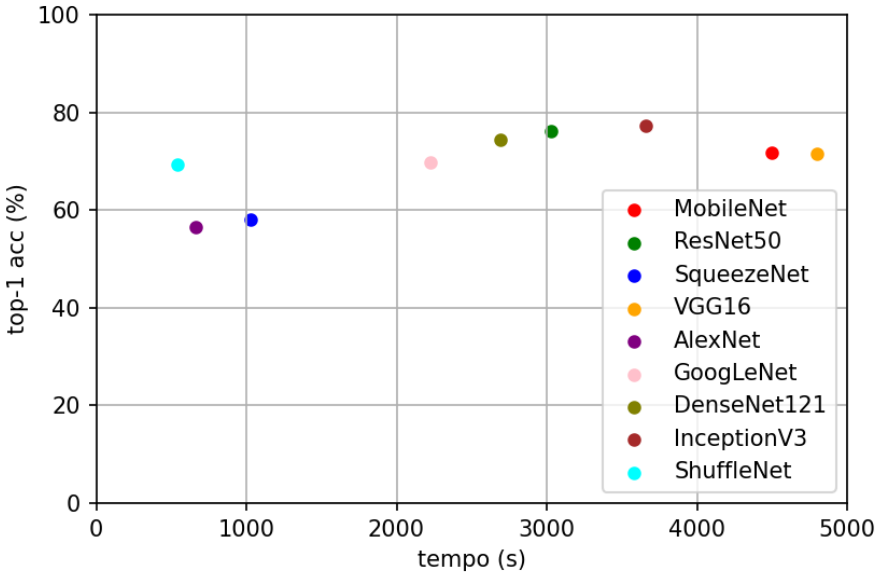

Figura 4.4: Precisão top-1 x tempo (ImageNet).

4.1.2 Imagens manualmente coletadas

Para as imagens manualmente coletadas da *web*, todos os modelos mostraram resultados satisfatórios (Tabela 4.2), o que leva a conclusão de que qualquer um poderia ser considerado para projeto de vida real.

Observa-se que, mais uma vez, o Inception-V3 foi o modelo mais efetivo, e o AlexNet o de menor precisão. O ShuffleNet-V2, desta vez, aparece como o segundo modelo mais preciso, mostrando assim excelente desempenho por ser também o modelo de menor tempo nesta classificação (é possível notar o salto do ShuffleNet-V2 na Figura 4.5).

Porém, é importante ressaltar que, a quantidade pequena de imagens desse banco pode influenciar em resultados precipitados, sendo natural que com mais imagens, os dados comecem a convergir para os resultados da seção anterior.

4.1.3 Consideração final

É importante lembrar que, vários dos modelos testados, como o ResNet, DenseNet e o VGG, possuem versões diferentes com mais ou menos camadas e propriedades diferentes.

Tabela 4.2: Resultados para as 100 imagens (ordenado pelo top-1).

Modelo	**top-1 acc**	**top-5 acc**	**tempo(s)**
Inception-V3	90%	98%	10.7
ShuffleNet-V2	87%	96%	1.4
VGG16	86%	97%	15.9
MobileNet-V2	86%	96%	13.4
ResNet50	84%	96%	8.6
DenseNet121	83%	97%	7.5
SqueezeNet	83%	95%	2.8
GoogLeNet	82%	94%	6
AlexNet	72%	90%	1.8

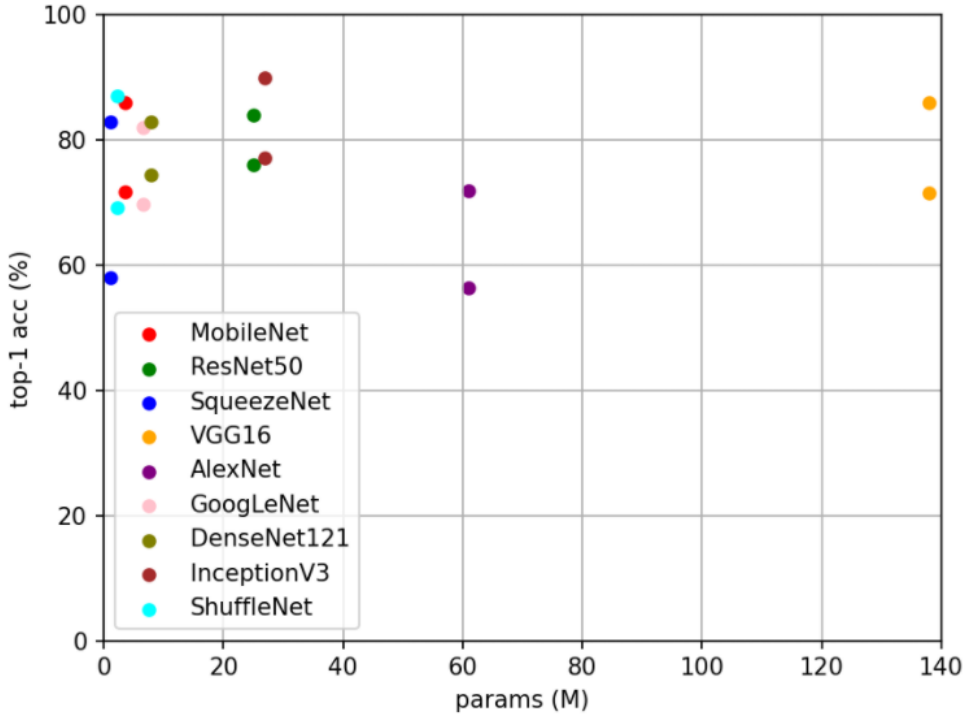

Figura 4.5: As diferenças de precisão dos modelos para os dois *datasets*.

Sendo assim, para estes modelos, pode-se optar por outras versões se o desejo for fazer um *trade-off* de melhor precisão por maior tempo de execução ou maior custo computacional, e vice-versa [6, 2, 5].

Capítulo 5

Conclusão

O desenvolvimento deste trabalho procurou esclarecer quais modelos de redes neurais para classificação de imagens e detecção de objetos são mais efetivos desempenhando seus objetivos. Para tal, foram utilizados dois bancos de dados de imagens, o *dataset* de validação da *ImageNet* utilizado no ILSVRC contendo 50000 imagens e um *dataset* de 100 imagens manualmente coletadas da *web*.

Os modelos foram avaliados quanto a precisão na classificação das imagens em seus rótulos respectivos com acertos para o top-1 e/ou top-5, além de avaliações de métricas de tempo de execução e consumo de recursos computacionais.

Os resultados obtidos estão de acordo com os apresentados por cada modelo em suas pesquisas e pelo Pytorch, *framework* utilizado para o projeto. Foram feitas correlações entre as métricas coletadas, procurando encontrar características que destacam alguma arquitetura das outras.

Dos resultados, foi constatado que o modelo mais eficiente quanto a precisão foi o Inception-V3, o que mostra que a implementação de módulos *inception* aliados a redes extensas é uma ótima estratégia a seguir como base para construções de modelos eficientes para classificação de imagens. Além do Inception-V3, vale destacar também os modelos ResNet e DenseNet, que obtiveram ótimos resultados, demonstrando que a passagem do gradiente das camadas iniciais para o final da rede a fim de evitar seu desaparecimento, é outra característica essencial para a construção de redes eficientes. Por fim, também é válido destacar o ShuffleNet-V2, que obteve uma precisão menor comparado aos modelos mais precisos, porém, para compensar, mostrou ser um modelo extremamente eficiente em questões de tempo de resposta e custo computacional, demonstrando ser um modelo de referência para dispositivos móveis e sistemas embarcados.

É importante notar também que, os modelos ResNet e DenseNet, possuem versões mais eficientes quanto a precisão, pagando com maiores tempos de resposta e gasto computacional, com arquiteturas mais densas e complexas. Dessa forma, para sistemas mais

robustos em que as limições computacionais são desprezíveis, ou projetos mais ambiciosos, estes dois modelos são podem ter maior prioridade.

Dentre as limitações da metodologia, é válido expor que o banco de 100 imagens possa ser muito pequeno para que se consiga levantar conclusões pertinentes. Outra limitação está no fato de diferentes bibliotecas apresentarem desempenhos e resultados distintos para os mesmos modelos, limitando o projeto à utilização de apenas uma destas bibliotecas e, consequentemente, limitando a disponibilidade de modelos para avaliar.

Para estudos futuros, sugere-se a utilização de uma base de dados maior para testes fora da bolha *ImageNet* e outros bancos tradicionais, e de preferência com imagens tiradas diretamente com a câmera de um dispositivo móvel pessoal. Além da comparação entre um número maior de redes conhecidas.

Referências

[1] Alex Krizhevsky, Ilya Sutskever e Geoffrey E. Hinton: *ImageNet Classification with Deep Convolutional Neural Networks*. NeurIPS, 2012. ix, 5, 6, 8, 10

[2] Karen Simonyan, Andrew Zisserman: *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2014. ix, 8, 10, 11, 13, 27

[3] Christian Szegedy, Wei Liu, Yangqing Jia Pierre Sermanet Scott Reed Dragomir Anguelov Dumitru Erhan Vincent Vanhoucke e Andrew Rabinovich: *Going deeper with convolutions*. CVPR, 2015. ix, 6, 10, 11, 12, 19

[4] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe Jonathon Shlens e Zbigniew Wojna: *Rethinking the Inception Architecture for Computer Vision*. CVPR, 2016. x, 11, 12

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren e Jian Sun: *Deep Residual Learning for Image Recognition*. CVPR, 2016. x, 12, 13, 27

[6] Gao Huang, Zhuang Liu, Laurens van der Maaten e Kilian Q. Weinberger: *Densely Connected Convolutional Networks*. CVPR, 2017. x, 6, 13, 15, 27

[7] Ningning Ma, Xiangyu Zhang, Hai Tao Zheng e Jian Sun: *ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design*. ECCV, 2018. x, 16, 17

[8] Forrest N. Iandola, Song Han, Matthew W. Moskewicz Khalid Ashraf William J. Dally Kurt Keutzer: *SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size*. 2016. x, 16, 17, 18

[9] Stuart Russell, Peter Norvig: *Artificial Intelligence: A Modern Approach*, volume 3. Prentice Hall, 2010. 1

[10] *5 Trends In Computer Science Research*. `https://www.topuniversities.com/courses/computer-science-information-systems/5-trends-computer-science-research`. 1

[11] Y. LeCun, B. Boser, J. S. Denker D. Henderson R. E. Howard W. Hubbard e L. D. Jackel: *Backpropagation applied to handwritten zip code recognition*. Neural Computation, 1989. 1, 6

[12] Olga Russakovsky, Jia Deng, Hao Su Jonathan Krause Sanjeev Satheesh Sean Ma Zhiheng Huang Andrej Karpathy Aditya Khosla Michael Bernstein Alexander

C. Berg e Li Fei-Fei: *Imagenet large scale visual recognition challenge*. https://image-net.org/challenges/LSVRC/2012/index.php. 1, 19

[13] *Google Trends*. https://trends.google.com.br/trends. 2

[14] *The neuron and nervous system*. https://www.khanacademy.org/science/biology/human-biology/neuron-nervous-system/a/the-synapse. 4

[15] Aggarwal, Charu C.: *Neural Networks and Deep Learning. A Textbook*, volume 1. Springer, 2018. 4, 6, 9

[16] *Gráfico ReLU*. https://bit.ly/3bsDUlc. 5

[17] Goodfellow, Ian, Yoshua Bengio e Aaron Courville: *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org. 6, 8

[18] Smith, Steven W.: *The Scientist and Engineer's Guide to Digital Signal Processing*, volume 1. California Technical Publishing, 1997. 6

[19] https://computersciencewiki.org/index.php/Max-pooling_/_Pooling. 7

[20] Murphy, John: *An Overview of Convolutional Neural Network Architectures for Deep Learning*. 2016. 7

[21] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky Ilya Sutskever e Ruslan R. Salakhutdinov: *Improving neural networks by preventing co-adaptation of feature detectors*. 2012. 9, 11

[22] Slijepcevic, Djordje: *Deep Learning with Tensorflow*. http://cvml.ist.ac.at/courses/DLWT_W17/material/AlexNet.pdf. 9

[23] Min Lin, Qiang Chen e Shuicheng Yan: *Network In Network*. 2013. 10

[24] Mark Sandler, Andrew Howard, Menglong Zhu Andrey Zhmoginov e Liang Chieh Chen: *MobileNetV2: Inverted Residuals and Linear Bottlenecks*. CVPR, 2018. 14

[25] Chollet, François: *Xception: Deep Learning with Depthwise Separable Convolutions*. CVPR, 2017. 14

[26] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin e Jian Sun: *ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices*. CVPR, 2018. 16

[27] Maeda-Gutiérrez, Valeria, Carlos E. Galván-Tejada, Laura A. Zanella-Calzada, José M. Celaya-Padilla, Jorge I. Galván-Tejada, Hamurabi Gamboa-Rosales, Huizilopoztli Luna-García, Rafael Magallanes-Quintanar, Carlos A. Guerrero Méndez e Carlos A. Olvera-Olvera: *Comparison of Convolutional Neural Network Architectures for Classification of Tomato Plant Diseases*. Applied Sciences, 10, 2020. 19

[28] Brahimi, Med, Marko Arsenovic, Sohaib Laraba, Srdjan Sladojevic, Boukhalfa Kamel e Abdelouahab Moussaoui: *Deep Learning for Plant Diseases: Detection and Saliency Map Visualisation*. 2018. 19

[29] *A Gentle Introduction to the ImageNet Challenge (ILSVRC).* https://bit.ly/3utlCad. 19

[30] *Torchvision Models.* https://pytorch.org/vision/stable/models.html. 21

[31] *Keras Models.* https://www.tensorflow.org/api_docs/python/tf/keras/applications. 21

[32] *Processador Intel® Core™ i7-8700.* https://ark.intel.com/content/www/br/pt/ark/products/126686/intel-core-i7-8700-processor-12m-cache-up-to-4-60-ghz.html. 21

[33] *PyTorch API Documentation.* https://pytorch.org/vision/stable/models.html. 21