UNIVERSIDADE FEDERAL DE GOIÁS
ESCOLA DE ENGENHARIA ELÉTRICA, MECÂNICA E DE COMPUTAÇÃO

TIAGO DO CARMO NOGUEIRA

Modelo Baseado em Redes Neurais Profundas com Unidades Recorrentes Bloqueadas para Legendagem de Imagens por Referências

Goiânia
2020

UNIVERSIDADE FEDERAL DE GOIÁS
ESCOLA DE ENGENHARIA ELÉTRICA, MECÂNICA E DE COMPUTAÇÃO

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO (TECA) PARA DISPONIBILIZAR VERSÕES ELETRÔNICAS DE TESES

E DISSERTAÇÕES NA BIBLIOTECA DIGITAL DA UFG

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio da Biblioteca Digital de Teses e Dissertações (BDTD/UFG), regulamentada pela Resolução CEPEC nº 832/2007, sem ressarcimento dos direitos autorais, de acordo com a Lei 9.610/98, o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou download, a título de divulgação da produção científica brasileira, a partir desta data.

O conteúdo das Teses e Dissertações disponibilizado na BDTD/UFG é de responsabilidade exclusiva do autor. Ao encaminhar o produto final, o autor(a) e o(a) orientador(a) firmam o compromisso de que o trabalho não contém nenhuma violação de quaisquer direitos autorais ou outro direito de terceiros.

1. Identificação do material bibliográfico

[ ] Dissertação [X] Tese

2. Nome completo do autor

Tiago do Carmo Nogueira

3. Título do trabalho

“Modelo Baseado em Redes Neurais Profundas com Unidades Recorrentes Bloqueadas para Legendagem de Imagens por Referências”

4. Informações de acesso ao documento (este campo deve ser preenchido pelo orientador)

Concorda com a liberação total do documento [X] SIM [ ] NÃO[1]

[1] Neste caso o documento será embargado por até um ano a partir da data de defesa. Após esse período, a possível disponibilização ocorrerá apenas mediante:
a) consulta ao(à) autor(a) e ao(à) orientador(a);
b) novo Termo de Ciência e de Autorização (TECA) assinado e inserido no arquivo da tese ou dissertação.
O documento não será disponibilizado durante o período de embargo.
Casos de embargo:
- Solicitação de registro de patente;
- Submissão de artigo em revista científica;
- Publicação como capítulo de livro;
- Publicação da dissertação/tese em livro.

Obs. Este termo deverá ser assinado no SEI pelo orientador e pelo autor.

Documento assinado eletronicamente por **Gelson Da Cruz Junior**, **Professor do Magistério Superior**, em 21/10/2020, às 12:29, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **TIAGO DO CARMO NOGUEIRA**, **Discente**, em 21/10/2020, às 13:15, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **1628899** e o código CRC **95ADD9DA**.

Referência: Processo nº 23070.047495/2020-64 SEI nº 1628899

TIAGO DO CARMO NOGUEIRA

Modelo Baseado em Redes Neurais Profundas com Unidades Recorrentes Bloqueadas para Legendagem de Imagens por Referências

Tese apresentada ao Programa de Pós-Graduação da Escola de Engenharia Elétrica, Mecânica e de Computação da Universidade Federal de Goiás, como requisito parcial para obtenção do título de Doutor em Engenharia Elétrica e de Computação.

Área de concentração: Engenharia de Computação.

Orientador: Prof. Dr. Gélson da Cruz Júnior

Co-Orientador: Prof. Dr. Cássio Dener Noronha Vinhal

Goiânia
2020

Ficha de identificação da obra elaborada pelo autor, através do
Programa de Geração Automática do Sistema de Bibliotecas da UFG.

Carmo Nogueira, Tiago do
Modelo Baseado em Redes Neurais Profundas com Unidades Recorrentes Bloqueadas para Legendagem de Imagens por Referências [manuscrito] / Tiago do Carmo Nogueira. - 2020.
cxxi, 121 f.

Orientador: Prof. Dr. Gélson da Cruz Júnior; co-orientador Dr. Cássio Dener Noronha Vinhal.
Tese (Doutorado) - Universidade Federal de Goiás, Escola de Engenharia Elétrica, Mecânica e de Computação (EMC), Programa de Pós-Graduação em Engenharia Elétrica e de Computação, Goiânia, 2020.

Inclui gráfico, tabelas, algoritmos, lista de figuras, lista de tabelas.

1. Aprendizado profundo. 2. Rede neural convolucional. 3. Unidades recorrentes bloqueadas. 4. Legendagem de imagens. 5. Parte do discurso. I. Cruz Júnior, Gélson da, orient. II. Título.

CDU 004

UNIVERSIDADE FEDERAL DE GOIÁS

ESCOLA DE ENGENHARIA ELÉTRICA, MECÂNICA E DE COMPUTAÇÃO

ATA DE DEFESA DE TESE

Ata Nº 06/2020 da sessão de Defesa de Tese de Tiago do Carmo Nogueira que confere o título de Doutor em **Engenharia Elétrica e de Computação**, na área de concentração em **Engenharia de Computação.**

Aos **vinte e oito dias do mês de setembro de dois mil e vinte**, a partir das **14h00min**, através de vídeo conferência, realizou-se a sessão pública de Defesa de Tese intitulada **"Modelo Baseado em Redes Neurais Profundas com Unidades Recorrentes Bloqueadas para Legendagem de Imagens por Referências"**. Os trabalhos foram instalados pelo Orientador, Professor Doutor **Gelson da Cruz Júnior (EMC/UFG)** com a participação dos demais membros da Banca Examinadora: Professora Doutora **Deller James Ferreira (InF-UFG)**, membro titular externo, Doutor **Gilberto Antonio Marcon dos Santos (CoRIS-OSU)**, membro titular externo; Professor Doutor **Cássio Dener Noronha Vinhal (EMC/UFG)**, membro titular externo; Professor Doutor **Rodrigo Pinto Lemos (EMC/UFG)**, membro titular interno. Durante a argüição os membros da banca **não fizeram** sugestão de alteração do título do trabalho. A Banca Examinadora reuniu-se em sessão secreta a fim de concluir o julgamento da Tese tendo sido o candidato **aprovado** pelos seus membros. Proclamados os resultados pelo Professor Doutor **Gelson da Cruz Júnior,** Presidente da Banca Examinadora, foram encerrados os trabalhos e, para constar, lavrou-se a presente ata que é assinada pelos Membros da Banca Examinadora, aos **vinte e oito dias do mês de setembro de dois mil e vinte**.

TÍTULO SUGERIDO PELA BANCA

Documento assinado eletronicamente por **Gilberto Antonio Marcon Dos Santos**, **Usuário Externo**, em 28/09/2020, às 16:00, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Rodrigo Pinto Lemos**, **Professor do Magistério Superior**, em 28/09/2020, às 16:06, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Gelson Da Cruz Junior**, **Professor do Magistério Superior**, em 28/09/2020, às 16:36, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Cassio Dener Noronha Vinhal**, **Professor do Magistério Superior**, em 28/09/2020, às 17:17, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Deller James Ferreira**, **Professor do Magistério Superior**, em 29/09/2020, às 15:47, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **TIAGO DO CARMO NOGUEIRA**, **Discente**, em 01/10/2020, às 20:43, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **1569497** e o código CRC **C01717E1**.

Referência: Processo nº 23070.040904/2020-00 SEI nº 1569497

Tiago do Carmo Nogueira

Graduado em Computação (UEG) e Redes de Computadores (Faesgo) com especialização em Engenharia de Sistemas (ESAB), Gestão Pública (UFMT), Redes e Computação Distribuída (IFMT). Mestre em Ciência da Computação pelo Instituto de Informática (INF – UFG) e Doutorando em Engenharia Elétrica e de Computação pela Escola de Engenharia Elétrica, Mecânica e de Computação (EMC – UFG). Atualmente é Professor de Ensino Básico, Técnico e Tecnológico (EBTT) do Instituto Federal de Educação, Ciência e Tecnologia Baiano (IFBaiano).

Dedico este trabalho à minha família, principalmente à minha mãe, Teodora Maria Nogueira, por estar ao meu lado em todos os momentos.

Agradecimentos

A Deus por essa conquista alcançada e por estar ao meu lado nesta longa caminhada. Foram muitas dificuldades enfrentadas durante estes quase quatro anos, mas todas elas foram superadas com a graça de Deus.

À minha família, em especial a minha mãe, pelas orações e o apoio nos árduos momentos em minha vida. Agradeço aos meus irmãos Lucas do Carmo Nogueira, Ailton do Carmo Nogueira, Conceição do Carmo Nogueira, Antônio do Carmo Nogueira, Maria Aparecida Nogueira, José do Carmo Nogueira, Nilda Dourado Nogueira e Hilda Dourado Nogueira por acreditarem em mim, me apoiando e incentivando a continuar nos meus estudos, tanto no mestrado quanto no doutorado.

Aos meus queridos orientadores, Dr. Gélson da Cruz Júnior e Dr. Cássio Dener Noronha Vinhal pela compreensão, supervisão e paciência. Agradeço toda a dedicação oferecida a mim durante este período de doutorado.

Aos meus amigos, Matheus Rudolfo Diedrich Ullmann, Raimundo Nonato de Araújo Soares Neto, Joaquim Clinton Rosa, Pedro Paulo da Silva Pereira, Ricardo de Oliveira Ferreira, João Melchior Junior, Jacinto José Franco e Gildásio Mesquita da Silva que me apoiaram em toda essa jornada.

Ao Instituto Federal Baiano, Instituto Federal do Tocantins e Instituto Federal do Mato Grosso pelo apoio e compreensão. Nestes últimos quatro anos, foram viagens longas realizadas por mim, desses estados para o estado de Goiás, horas e mais horas de viagens para a realização deste objetivo.

Agradeço aos meus amigos de trabalho, George Gabriel Mendes Dourado, Cleyton Fábio Leite Batista, Antonio Queiroz da Silva Neto, Maxwell Francisco da Silva e Luciano Bertollo Rusciolelli pelo apoio de sempre.

Aqui, gostaria de fazer um agradecimento especial. Agradeço e dedico este trabalho em memória do meu pai, Abílio do Carmo Nogueira. Apesar do mesmo não ter tido a oportunidade de estudar, me ensinou o caminho a seguir para conquistar meus objetivos. Tenho orgulho das minhas origens, sendo filho de um pedreiro e de uma simples dona de casa. Essa vitória dedico a vocês. Tenho certeza que o senhor está orgulhoso daí de cima (céu).

"...automatically describing the visual content of images is paint a picture in your mind's eye" [184].

Yang *et al*. (2019),
Image captioning by incorporating affective concepts learned from both visual and textual components.

Resumo

Nogueira, Tiago C. **Modelo Baseado em Redes Neurais Profundas com Unidades Recorrentes Bloqueadas para Legendagem de Imagens por Referências**. Goiânia, 2020. 121p. Tese de Doutorado. Escola de Engenharia Elétrica, Mecânica e de Computação, Universidade Federal de Goiás.

Descrever imagens por meio da linguagem natural se tornou uma tarefa desafiadora para a visão computacional. A legendagem de imagem é capaz de criar descrições de forma automática, pelas arquiteturas de aprendizado profundo, que utilizam redes neurais convolucionais (CNNs) e redes neurais recorrentes (RNNs). Dessa forma, a tarefa de legendagem de imagem possui várias aplicabilidades, como, por exemplo, nas descrições de objetos em cenas para a locomoção de pessoas cegas em ambientes desconhecidos e nas descrições de imagens médicas para o diagnóstico precoce de doenças. No entanto, arquiteturas apoiadas em RNNs tradicionais, além de ter problemas com o gradiente explodindo e desvanecendo, sofrem com as gerações de sentenças não descritivas. Para solucionar tais dificuldades, propõe-se, neste trabalho, um modelo baseado na estrutura codificador-decodificador, utilizando CNNs para extrair as características das imagens e as unidades recorrentes bloqueadas (GRU) multimodais para realizar as descrições. Além disso, aplicou-se a parte-do-discurso (PoS) e a função de verossimilhança para a geração dos pesos na GRU. O método proposto realiza a transferência de conhecimento na fase de validação pela técnica k-vizinhos mais próximos (kNN). Assim, os resultados experimentais nos conjuntos de dados Flickr30k e MS-COCO demonstram que o modelo proposto baseado na PoS apresenta pontuações significativas quando comparados aos modelos de ponta, prevendo legendas mais descritivas, aproximando-se das legendas esperadas, tanto na legenda de predição quanto nas selecionadas pela kNN. Esses resultados corroboram para a melhoria nas descrições das imagens de forma automática, podendo beneficiar várias aplicações, como, por exemplo, os modelos de legendagem de imagens médicas para o diagnóstico precoce de doenças.

Palavras–chave

Aprendizado Profundo, Rede Neural Convolucional, Unidades Recorrentes Bloqueadas, Legendagem de Imagens, Parte do Discurso, Verossimilhança.

Abstract

Nogueira, Tiago C. **Based-Model on Deep Neural Networks Using Gated Recurrent Units for Image Captioning by References**. Goiânia, 2020. 121p. PhD. Thesis. Escola de Engenharia Elétrica, Mecânica e de Computação, Universidade Federal de Goiás.

Describing images using natural language has become a challenging task for computer vision. Image captioning can automatically create descriptions through deep learning architectures that use convolutional neural networks (CNNs) and recurrent neural networks (RNNs). Image captioning has several applications, such as object descriptions in scenes to help blind people walk in unknown environments, and medical image descriptions for early diagnosis of diseases. However, architectures supported by traditional RNNs, in addition to problems of exploding and fading gradients, can generate non-descriptive sentences. To solve these difficulties, this study proposes a model based on the encoder-decoder structure using CNNs to extract the image characteristics and multimodal gated recurrent units (GRU) to generate the descriptions. The part-of-speech (PoS) and the likelihood function are used to generate weights in the GRU. The proposed method performs knowledge transfer in the validation phase using the k-nearest neighbors (kNN) technique. The experimental results in the Flickr30k and MS-COCO data sets demonstrate that the proposed PoS-based model is statistically superior to the leading models. It provides more descriptive captions that are similar to the expected captions, both in the predicted and kNN-selected captions. These results indicate an automatic improvement of the image descriptions, benefitting several applications, such as medical image captioning for early diagnosis of diseases.

Keywords

Deep Learning, Convolutional Neural Network, Gated Recurrent Units, Image Captioning, Part-of-Speech, Likelihood.

Sumário

Lista de Figuras

Lista de Tabelas

CAPÍTULO 1

Introdução

Observando-se uma cena em uma imagem, os seres humanos são capazes, facilmente, de identificar pessoas ou objetos ali contidos, detectando até mesmo as emoções retratadas pelas expressões faciais. Os seres humanos percebem facilmente as estruturas tridimensionais dos objetos presentes em uma imagem. Em paralelo, técnicas matemáticas têm sido desenvolvidas por pesquisadores em visão computacional para a recuperação das formas tridimensionais e das aparências dos objetos contidos nas imagens.

No início da década de 70, a área da visão computacional possuía uma agenda ambiciosa de imitar em robôs a inteligência humana. A visão computacional recuperava as estruturas tridimensionais das imagens, utilizando-as, futuramente, como um caminho para o entendimento completo nas cenas das imagens. Nessa época, alguns dos pioneiros da inteligência artificial acreditavam que a solução do problema da entrada visual em um robô seria um passo fácil no caminho para solucionar questões mais difíceis, como, por exemplo, as soluções de problemas relacionadas ao raciocínio lógico complexo.

A inteligência artificial resolvia/resolve rapidamente problemas considerados intelectualmente difíceis para os seres humanos, que podem ser descritos por listas de regras formais e matemáticas. Um desses problemas, no campo da visão computacional, está relacionado ao reconhecimento de palavras ou de rostos em uma determinada imagem [57].

A visão computacional, nas últimas décadas, concentrou-se nas técnicas matemáticas sofisticadas para a realização de análises quantitativas das imagens e das cenas. Assim, as aplicações dessas técnicas possibilitaram a rastreabilidade de objetos em determinadas imagens, identificando-as e nomeando-as, pelas combinações entre a detecção e o reconhecimento dos objetos.

Entretanto, apesar dos sucessos nas aplicações dessas técnicas, ainda é distante que os computadores tenham a capacidade de interpretação de uma determinada imagem no mesmo nível humano. Esse fato ocorre em razão de que a visão computacional trata problemas inversos, que buscam recuperar algumas incógnitas com informações insuficientes para especificar completamente uma solução. Para solucionar tal problema, pode-se

recorrer a modelos matemáticos probabilísticos, capazes de desambiguar possíveis soluções.

Atualmente, a visão computacional realiza a tarefa de descrever a perspectiva do mundo de uma determinada imagem, construindo as suas propriedades pelas formas, da iluminação e das distribuições de cores nas cenas [165]. Nesse contexto, um dos benefícios dessa área está ligada à sua ampla aplicabilidade no mundo real, como, por exemplo, no reconhecimento óptico de caracteres; na inspeção de máquinas; no reconhecimento automatizado de objetos; na construção de modelos 3D (fotogrametria); nas imagens médicas; na segurança automobilística; na captura de movimentos de objetos; no reconhecimento de impressões digitais e na biometria.

Muitas tarefas dessas áreas podem ser resolvidas extraindo um conjunto de características das imagens, fornecendo-as para um algoritmo simples de aprendizado de máquina [57]. A tecnologia de aprendizado de máquina potencializa muitos aspectos da sociedade moderna, como, por exemplo, busca na Web, filtragem de conteúdo em redes sociais e as recomendações em sites de comércio eletrônico. O aprendizado de máquina está cada vez mais presente em produtos de consumo, como câmeras e smartphones [97].

A inserção do aprendizado de máquina permitiu que os computadores resolvessem problemas que envolviam o conhecimento do mundo real e tomassem decisões subjetivas [57]. Dessa forma, esses sistemas podem ser usados para identificar objetos nas imagens; transcrever fala em texto; combinar itens de notícias, postagens ou produtos com os interesses dos usuários e selecionar resultados relevantes de uma busca.

O aprendizado de máquina pode ser definido como programas de computadores que otimizam as soluções de determinados problemas, sob critérios de desempenho, utilizando dados de exemplo ou experiências anteriores (conhecimentos prévios). Assim, os modelos gerados pelo aprendizado de máquina podem ser preditivos (previsões futuras) ou descritivos (conhecimento de dados). Para tal, o aprendizado de máquina utiliza estatística para a construção de modelos matemáticos, realizando, dessa forma, inferências em determinados conjuntos de dados [5] nas etapas de treinamento e de validação.

A etapa de treinamento consiste em um processo de aplicação de um conjunto ordenado de passos, que ajustam os pesos da rede neural, utilizando um conjunto de dados para o seu treinamento. Esse processo, conhecido também como algoritmo de aprendizagem, possui o intuito de ajustar os resultados das saídas das redes neurais com os valores esperados [38]. Nessa etapa, alguns aspectos importantes são considerados para o treinamento da rede neural, dentre eles, a seleção do modelo de treinamento, a inicialização da rede e o tempo de treinamento. Na etapa de validação são realizadas as estimativas dos erros de predições do modelo, utilizando um conjunto de dados para a validação [159]. Esse processo fornece informações que auxiliam nos ajustes dos parâmetros de treinamento da rede neural, como, por exemplo, a inicialização dos pesos.

Para a realização de inferências, na etapa de treinamento, o aprendizado de máquina busca algoritmos eficientes capazes de solucionar os problemas de otimização, armazenando e processando enormes quantidades de dados. Na etapa de validação, depois que o modelo é aprendido, a representação da solução algorítmica de inferência precisa ser também eficiente [5].

Observa-se que, em alguns casos, a complexidade de espaço e tempo do algoritmo de aprendizado de máquina pode ser tão importante quanto a sua precisão preditiva [5]. Esse fator colabora para a diversidade de tarefas as quais o aprendizado de máquina pode ser aplicado, como nas tarefas de associação de aprendizado [52], na classificação [129], na regressão [53], no aprendizado não-supervisionado [72] e por reforço [80].

Nessa perspectiva, as tarefas de aprendizado por reforço possuem o intuito de aprender um mapeamento de entrada para uma determinada saída, cujos valores corretos são fornecidos por um supervisor, enquanto que nas tarefas de aprendizado não-supervisionado não há um supervisor, existindo, nesse caso, apenas os dados de entrada [5]. O aprendizado por reforço está ligado à capacidade que o algoritmo possui de avaliar a qualidade dos dados ou informações, aprendendo com as sequências de informações passadas, gerando, assim, novos dados ou informações [5].

Apesar do sucesso na aplicação de técnicas de aprendizado de máquina para a realização dessas tarefas, as técnicas convencionais eram limitadas na sua capacidade de processar dados naturais de forma bruta.

Por décadas, a construção desses sistemas exigiu engenharia cuidadosa e conhecimento considerável do domínio para projetar um extrator de características que transformasse dados brutos (como os valores de *pixel* de uma imagem) em uma representação interna adequada ou em um vetor de características.

Esse vetor de características pode ser gerado para que o subsistema de aprendizado, geralmente um classificador, seja capaz de detectar ou classificar padrões na entrada do sistema [97]. No entanto, o desempenho de um algoritmo de aprendizado de máquina depende das representações dos dados de entrada, fazendo com que as aplicações modernas utilizem cada vez mais das técnicas do aprendizado profundo para realizar o aprendizado dessas representações, extraindo recursos abstratos de alto nível dos dados brutos de entrada do sistema.

O aprendizado profundo resolve esse problema central no aprendizado das representações, introduzindo as que são expressas ou as simples, permitindo que o computador construa conceitos complexos, a partir de conceitos mais simples [57]. Os modelos do aprendizado profundo denotam os vários níveis de representações obtidos pela composição dos módulos simples, mas não lineares, que transformam a representação simples em um nível mais alto, incluindo as entradas brutas dos dados [97]. Assim, o aprendizado profundo se torna factível, à medida que são disponibilizadas quantidades exponenciais

de volume de dados brutos para o seu treinamento [57].

Nesse contexto, o aprendizado profundo vem fazendo grande diferença na solução de problemas que estavam ainda em aberto na área da inteligência artificial, mostrando-se ter bons modelos para a descoberta de estruturas complexas em dados de alta dimensão. Portanto, esses modelos podem ser aplicados a muitos domínios da ciência, negócio ou do governo [97].

Além de se obter resultados significativos no reconhecimento de imagens e no reconhecimento de falas, o aprendizado profundo promove resultados promissores para várias tarefas de entendimento da linguagem natural, particularmente na classificação de tópicos, na análise dos sentimentos, nos sistemas de respostas e perguntas, na tradução automática de idiomas e na descrição automática de imagens, sendo esta última o objeto central deste trabalho.

1.1 Visão Geral

Descrever imagens de uma forma automática por frases em linguagem natural tem atraído a atenção na área da visão computacional. A legendagem de imagens é uma tarefa que cria uma descrição para determinada imagem por uma sequência de palavras ou frases [150, 188, 104]. Para a realização dessa tarefa, são utilizados modelos de aprendizado de máquina capazes de codificar os objetos, os atributos semânticos e inferir as suas relações [64]. Embora seja fácil para os seres humanos entenderem como ocorrem essas relações, para as máquinas ainda é uma tarefa desafiadora, principalmente porque são necessárias, além de reconhecer os objetos que estão contidos em uma imagem, expressar as relações existentes entre eles [62].

Vários trabalhos na literatura propõem uma estrutura codificador-decodificador para a realização dessa tarefa [65, 62, 117, 105, 150, 64, 188] (Figura 1.1).

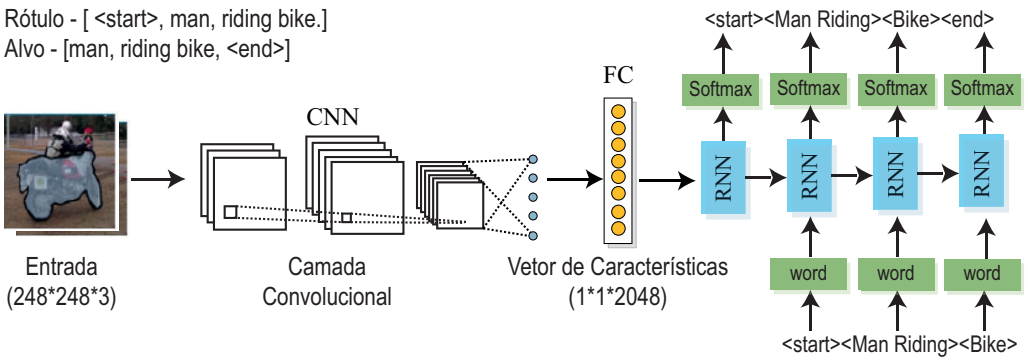


Figura 1.1: *Exemplo da estrutura codificador-decodificador para tarefas de legendagem de imagens.*

Essa estrutura utiliza uma Rede Neural Convolucional (CNN) como codificador, extraindo as características da imagem e uma Rede Neural Recorrente (RNN) como decodificador, gerando as descrições das imagens [150, 188]. Nesse sentido, de acordo com Seshadri e Srikanth [149], as abordagens baseadas nas estruturas codificador-decodificador para legendagem de imagens podem ser categorizadas em modelos baseadas em injeção e mesclagem.

Nos modelos baseados em injeção, o codificador primeiro realiza a codificação da imagem em uma representação vetorial com um comprimento fixo, combinando a forma codificada da imagem de entrada com cada palavra gerada da sequência da descrição. Nessa estrutura, o decodificador atua como um modelo de geração de textos, usando uma sequência de informações como entrada para a geração da próxima palavra. O modelo baseado em mesclagem, por sua vez, combina a forma codificada da imagem de entrada com a entrada da descrição textual, que são usadas por um modelo decodificador para a geração da próxima sequência de palavras. Esse formato separa a entrada da imagem, do texto e as interpretações das entradas codificadas.

Portanto, nas estruturas dos modelos aplicados à tarefa de legendar imagens, existem dois módulos principais: um codificador da imagem de entrada e um decodificador de saída. Nesse sentido, todos os modelos apoiados nessas estruturas utilizam as CNNs no processo de codificação das imagens [84, 25], por causa da sua capacidade de aprender padrões a partir dos valores dos *pixels* das imagens e RNNs para o processo de decodificação. Assim, a utilização de modelos que combinam as extrações de recursos globais e locais das imagens por CNN-RNN, podem melhorar a qualidade na geração dessas sentenças [188].

1.2 Problema e Motivação

Apesar dos avanços nas tarefas de legendagem de imagem pela utilização de CNN e RNN, a aplicação da RNN tradicional apresenta alguns problemas, como, por exemplo, o gradiente desvanecendo (do inglês *vanishing gradient*) [6] e gradiente explodindo (do inglês *exploding gradient*) [103, 172].

Esses problemas impossibilitam as alterações dos pesos de forma eficiente das redes neurais, que utilizam algoritmos *backpropagation*. Uma possível solução, além da utilização da Memória Longa de Curto Prazo (do inglês *Long-Short Memory* - LSTM) ou das Unidades Recorrentes Bloqueadas (do inglês *Gated Recurrent Units* - GRU) na decodificação e geração de sentenças, pode-se aplicar, também, o método RMSprop como otimizador ou a técnica de grampo no gradiente (do inglês *Clipping Gradient*) [128, 50].

Nesse sentido, Sharma et al. [150] aplicaram um modelo baseado na estrutura codificador-decodificador com duas etapas: a primeira etapa aplica um codificador neural

e a segunda etapa um decodificador. A primeira extrai as características das imagens por um modelo baseado no Grupo de Geometria Visual (do inglês *Visual Geometry Group* - VGG) e na segunda etapa são geradas as sentenças em linguagem natural pela utilização de LSTMs e GRUs.

Outro aspecto importante para a descrição de imagem são os mecanismos de atenção. A atenção semântica é uma das facetas mais curiosas do sistema humano, diferentemente das abordagens tradicionais, que comprimem uma imagem em uma representação estática. Os modelos de atenção permitem que características salientes possam desempenhar papeis importantes na descrição das imagens [23, 176, 65, 139]. Inspirando-se nesse aspecto, Peng et al. [136] propuseram um modelo de geração de legenda com LSTM dupla, utilizando as informações de cenas das imagens, introduzindo um mecanismo de atenção semântico para as descrições das imagens.

Entretanto, esses modelos, mesmo sendo eficientes em legendar imagens, ainda sofrem com a geração de sentenças não descritivas. Nesse segmento, Ding et al. [44] observam que, no processo de geração de sentenças descritas para uma imagem, pode-se avaliar a importância de cada palavra para a formação da frase correta, isso é, avaliar o conjunto de palavras que melhor descreve a imagem. Dessa forma, existem palavras geradas durante o processo de legendagem de imagem, que são pouco informativas, fazendo com que ocorra um acúmulo de erros na geração da descrição, com um grau de dificuldade significativa para corrigi-las. Assim, além dos problemas tradicionais ocasionados pelas redes recorrentes, ocorrem, ainda, problemas de informações inadequadas [126] e de reconhecimento incorreto dos objetos das imagens. Para solucionar tais dificuldades, este trabalho tem como objetivo:

- Propor um modelo utilizando Inception e VGG, para a extração das características visuais das imagens de treinamento e uma GRU multimodal para a geração de sentenças por referências;

- Atribuir pesos diferentes para cada conjunto de palavras, pela análise da parte-do-discurso e da aplicação da função de verossimilhança, possibilitando uma melhoria na alimentação dos pesos na RNN na fase de decodificação;

- Aplicar a GRU para a geração das legendas, auxiliadas pela aplicação de um método baseado em transferência de aprendizado, k-vizinhos mais próximos, realizando uma comparação entre as semelhanças do conjunto de frases k mais aproximadas e a frase de validação, fazendo com que o modelo vincule as frases em linguagem natural ao conteúdo das imagens mais próximas (modelo baseado em referências).

Para a experimentação, a validação e a robustez da solução proposta foram utilizados dois conjuntos de dados, MS-COCO e Flickr30k. O MS-COCO foi escolhido pelo fato de ser um dos maiores conjuntos de dados de imagens [182], possuindo cerca de cento e sessenta mil imagens rotuladas. Não obstante, o Flickr30k foi selecionado por ser um dos conjuntos de dados mais populares nas realizações de *benchmarks* [147], possuindo cerca de oito mil imagens devidamente rotuladas. Em conseguinte, pela realização dos experimentos nos dois conjuntos de dados, o modelo proposto é capaz de propiciar uma análise quantitativa e qualitativa, uma vez que o modelo foi exposto a conjuntos de imagens distintas, obtendo resultados com maior grau de confiabilidade.

1.3 Trabalhos Correlatos

A legendagem de imagem é uma tarefa que descreve o conteúdo semântico e visual da imagem, gerando um conjunto de sentenças descritivas [62]. Trabalhos recentes melhoram significativamente a qualidade na geração de legendas, aplicando abordagens pelas CNNs-RNNs [178, 45, 105, 150, 62, 188]. Tais abordagens, geralmente, utilizam a estrutura codificador-decodificador para a tradução da máquina neural [178, 136, 65, 64, 117, 55]. Nesse caso, a CNN codifica a imagem de entrada, transformando-a em um vetor de características [45, 150, 188]. A RNN decodifica o vetor de entrada em uma sequência de palavras, que representa a semântica do objeto [192, 104].

Os modelos baseados na estrutura codificador-decodificador são classificados em duas categorias, abordagens *top-down* e *bottom-up* [190, 45]. As abordagens *top-down* convertem diretamente as características visuais da imagem em descrições, enquanto que as abordagens *bottom-up* convertem os atributos de aspectos semânticos da imagem de entrada em uma sequência de palavras [178, 190, 45, 65].

Porém, o uso dessas abordagens de formas separadas resolve parcialmente os desafios, tanto em nível visual quanto semântico, sofrendo ainda com a falta de modelos de ponta capazes, de gerarem sentenças baseadas em aspectos específicos da imagem, bem como informações ambíguas [178]. Para solucionar tais problemas, esforços são realizados para a junção das duas abordagens, propiciando informações de níveis visuais e semânticos [178, 190].

Ainda sobre as estruturas codificador-decodificador, abordagens que utilizam RNNs tradicionais possuem problemas com gradiente explodindo ou desvanecendo, dificultando o treinamento desses modelos [105]. Alguns trabalhos utilizam a LSTM para solucionar esses problemas [150, 176]. Nesse aspecto, a LSTM realiza as transcrições das imagens, decodificando o vetor de características de entrada [104, 23, 176, 65, 139].

Nesse sentido, Sharma et al. [150] propuseram um modelo que utiliza redes neurais profundas, aplicando técnicas de aprendizado de máquina em uma estrutura

codificador-decodificador, divididas em duas etapas. A primeira etapa extrai as características da imagem por uma CNN. A segunda etapa gera as sentenças em linguagem natural por duas RNNs. Na primeira fase são detectados os objetos e extraídas as características da imagem, objetivando comparar as semelhanças entre elas. Esse modelo adota o VGG para realizar a extração das características da imagem. Para a geração das sentenças em linguagem natural, o modelo adota as redes recorrentes LSTM e GRU, comparando o desempenho entre as duas, pela pontuação Bleu [150].

Similar a esse modelo, Al-Muzaini, Al-Yahya e Benhidour [1] propuseram a implementação de uma abordagem para a geração de legendas semânticas de imagens em árabe. O modelo se baseia em uma RNN-LSTM para codificar sequências linguísticas de comprimento variável e um extrator de imagem baseado em CNN para a extração das características, gerando um vetor de comprimento fixo.

No entanto, modelos que utilizam abordagens CNN-LSTM podem ter perdas no controle do conteúdo da imagem original, sendo propenso a realização de àajustes das sentenças no conjunto de treinamento [82]. De acordo com He e Hu [62], uma solução para esse problema seria adicionar um vetor de características visuais da imagem como uma entrada extra para todas as unidades de bloco na LSTM. Entretanto, essa estratégia sobrecarrega ainda mais o conteúdo da imagem na LSTM, piorando o seu desempenho [82]. Adicionar um vetor de características semânticas como entrada extra na LSTM, pode superar esses problemas. Outra solução, seria utilizar-se das relações entre as regiões de objetos na imagem [23].

Dessa forma, Li et al. [104] propuseram a utilização de uma rede recorrente bidimensional (2DLSTM), contendo características de invariância na tradução, codificando as relações entre as regiões de uma determinada imagem, extraindo os mapas de características globais e locais por uma CNN, convertendo-os em atributos locais estruturais. A aplicação dessa abordagem indica uma forte influência do módulo de linguagem na precisão do modelo, obtendo resultados significativos nas avaliações linguísticas.

As perdas do conteúdo da imagem de entrada ao longo do processo de legendagem, mesmo em um pequeno espaço de tempo, ocorrem porque a LSTM guarda as informações das imagens apenas no processo de entrada. Para Zhou et al. [192], aplicar técnicas de atenção condicional ao texto, baseando-se na *g*-LSTM, interpretando as características das imagens com base no contexto textual, extraindo informações, pode ser uma possível solução. Assim, o modelo manteria as informações das imagens ao longo do tempo.

Não obstante, as informações locais e globais são atributos importantes para a geração das descrições das imagens. Com problemas de perda de informações globais e locais importantes nas descrições da imagem advindas das redes neurais multimodais e dos métodos baseados em recuperação, Yuan, Li e Lu [188] propuseram um novo

modelo de três portas (3 *gated*) de ponta-a-ponta, o qual combina as características locais e globais das imagens. Essa abordagem se utiliza de todas as informações contidas na imagem, melhorando a qualidade da sua descrição. Dessa forma, a primeira porta controla a quantidade necessária de características locais, que poderão ser utilizados pelo mecanismo de atenção. As demais portas são utilizadas para o módulo de inclusão multimodal e o módulo de linguagem, respectivamente. Na abordagem, a LSTM gera a probabilidade em cada etapa de tempo, utilizando a Frequência da Rede LSTM (*GF*-LSTM) para modelar a probabilidade condicional [188].

Outro fator importante é a inclusão de estruturas, que se utilizam da atenção temporal e espacial para selecionar regiões nas imagens, prevendo palavras relacionadas por LSTMs hierárquicas, podendo aplicar abordagens de atenção adaptativas. Essas abordagens permitem a representação de dados visuais complexos, com informações em diferentes escalas, considerando as visuais de baixo nível e as contextuais de alto nível, melhorando, assim, o desempenho dos sistemas de legendagem de imagens [54].

Apesar do bom desempenho[1], nas descrições de imagens utilizando LSTM, implementá-las requer um esforço maior, precisando de mais parâmetros e de tempo para a sua execução [105]. Para solucionar esse problema, pode-se incorporar uma rede multimodal baseado em GRU para descrições de tamanho variáveis. Esse modelo possui duas partes importantes, um codificador CNN e um decodificador GRU. Esse modelo multimodal utiliza a arquitetura VGG como codificador, excluindo as últimas camadas totalmente conectada e softmax, utilizando-se a última saída como um representador das características da imagem. Na fase de treinamento do modelo, todos os parâmetros são aprendidos pela maximização da função de verossimilhança, usando o algoritmo *backpropagation* [105].

Outra direção promissora é a utilização de Redes Adversárias Generativas (GAN) para solucionar problemas na descrição de imagens advindos da utilização da LSTM. Assim, Song et al. [154] propuseram um modelo unificado baseado na GAN para solucionar problemas na recuperação e compactação de imagens. Esse modelo converte imagens em códigos binários para o aprendizado não-supervisionado de multitarefas. Porém, para Dognin et al. [46], apesar da viabilidade da aplicação da GAN na legendagem de imagens, existem barreiras significativas a serem quebradas, justamente porque a GAN possui uma natureza discreta para essa tarefa, dificultando o treinamento desses sistemas, os quais produzem resultados ainda insatisfatório.

Portanto, há vários trabalhos na literatura que endereçam a aplicação da estrutura codificador-decodificador para descrever imagens, utilizando as abordagens CNN e

[1] O desempenho nas tarefas de legendagem de imagens pode ser mensurado pelas aplicações de métricas capazes de avaliar a qualidade dos descritores neurais, calculando a precisão entre as frases candidatas e as frases de referências.

LSTM. No entanto, alguns desses modelos ainda possuem problemas, como, por exemplo, a explosão do gradiente e a geração de sentenças inadequadas, prejudicando, de forma significativa, na compressão e na descrição da imagem.

Sendo assim, este trabalho propõem um modelo capaz de solucionar problemas relacionados com a geração de sentenças inadequadas, aplicando um decodificador multimodal GRU e um módulo para a geração de pesos. Nessa proposta, o módulo de geração de pesos é capaz de gerar diferentes pesos, de acordo com a relevância de cada palavra para a formação da sentença, tornando-se um mecanismo importante para o treinamento de decodificadores neurais.

1.4 Organização da Tese

Além deste Capítulo, este trabalho está organizado da seguinte forma:

- O Capítulo 2 aborda os princípios básicos sobre o processamento de linguagem natural em relação às estruturas linguísticas; as tarefas da parte-do-discurso; da análise; do reconhecimento de entidade mencionadas; da rotulagem de funções semânticas; da detecção automática de sinônimos e das aplicabilidades.

- O Capítulo 3 apresenta os conceitos sobre as redes neurais artificiais, redes neurais convolucionais e recorrentes, bem como as arquiteturas Inception e VGG.

- O Capítulo 4 apresenta as arquiteturas dos modelos propostos e as descrições sobre as técnicas para as gerações de pesos por parte-do-discurso e verossimilhança, bem como as técnicas empregadas para a geração das legendas por referências, utilizando-se da *K*NN.

- O Capítulo 5 apresenta os conjuntos de dados utilizados durante a pesquisa, as configurações dos experimentos, as métricas usadas para a avaliação do modelo e as abordagens selecionadas para a realização de *benchmarks*.

- O Capítulo 6 apresenta os resultados das avaliações dos modelos baseado nas arquiteturas Inception-V3 e VGG19; as comparações com os resultados do estado da arte e as análises das descrições geradas pelos modelos.

- Por fim, o Capítulo 7 apresenta as conclusões das aplicações dos modelos propostos baseados na estrutura codificador-decodificador.

CAPÍTULO 2

Processamento de Linguagem Natural

O Processamento de Linguagem Natural (PLN) se iniciou na década de 1950, com a inserção da inteligência artificial e a linguística. Inicialmente, o PLN se distinguiu da área de recuperação de informações, que aplicava técnicas estatísticas escalonáveis, indexando e pesquisando grandes volumes de informações. Atualmente, o PLN e a recuperação de informação se convergiram, apoiadas em diversas áreas do conhecimento [124].

O PLN permite que as máquinas processem linguagens humanas de forma inteligente, tornando-se um campo hiperativo e interdisciplinar, unindo as áreas da inteligência artificial, a ciência cognitiva, a ciência da computação, o processamento de informação e a linguística [40]. O PLN, portanto, tornou-se uma técnica auxiliadora para a compreensão de informações geradas pelos humanos, utilizando-se de uma sequência lógica de palavras dependentes de contextos [169], simplificando a comunicação entre humanos e as máquinas [160]. Outrossim, o PLN modela os mecanismos cognitivos implícitos à produção e à compreensão da linguagem humana, visando o desenvolvimento de novas aplicações práticas [40].

Com a capacidade do PLN de entender o contexto o qual as palavras estão sendo inseridas, as informações se tornam mais significativas, facilitando, assim, a análise textual pelos padrões advindos das estruturas da comunicação humana. Para tal, o PLN se utiliza de um conjunto de técnicas computacionais, capazes de analisar e representar textos que advêm naturalmente dos níveis das análises linguísticas [108]. Assim, o PLN fornece uma ponte entre o computador e a linguagem natural, fazendo com que as máquinas tenham a capacidade humana de entender, processar e analisar [169, 93].

Pela aplicabilidade dessas habilidades, o PLN oferece uma gama de oportunidades para solucionar problemas de interesse no campo da inteligência artificial, tornando-a uma das fronteiras mais recentes para o desenvolvimento de *softwares* inteligentes [93]. Não obstante, um dos principais desafios dessa área, em comparação às demais, como, por exemplo, a visão computacional, é a complexidade de realizar a representação profunda da linguagem, por modelos estatísticos [169]. Ademais, a tarefa principal do PLN fornece uma representação textual, envolvendo a extração dos dados significativos, processando e

analisando os dados brutos das informações.

Para a realização dessa tarefa, pode-se aplicar duas abordagens: a) abordagens tradicionais, que realizam a análise cuidadosa dos recursos, implementando algoritmos capazes de extrair informações por instâncias dos recursos (imagens e/ou textos) ou b) abordagens que aplicam recursos da aprendizagem profunda, de forma supervisionada e orientada a dados, representando um modo mais robusto das informações.

2.1 Estruturas Linguísticas

Um sistema com PLN requer conhecimento considerável das estruturas linguísticas de um determinado idioma, entendendo as relações entre palavras, frases e sentenças, compreendendo, também, o significado dos léxicos e o contexto o qual estão inseridas [160]. Para tanto, habilidades sobre as estruturas fonológicas, morfológicas, sintáticas, lexicais, pragmáticas e do discurso são essenciais para a construção de uma sentença no PLN.

- **Estruturas fonológicas** - as estruturas fonológicas lidam com a interpretação dos sons da fala ou das palavras, utilizando as regras fonéticas (sons das palavras), as regras fonêmicas (variações da pronúncia) e as regras prosódicas (flutuações e entonações) [108]. Em um sistema PLN, por exemplo, de entrada de voz, analisam-se as ondas sonoras codificando-as em sinais digitalizados para interpretação, utilizando regras de comparação de modelos de idiomas específicos.

- **Estruturas morfológicas** - as estruturas morfológicas estão relacionadas às estruturas de composição das palavras, isso é, as menores unidades das palavras, os morfemas. Assim, uma palavra pode ser analisada morfologicamente pelo seu prefixo, raiz e sufixo. Um sistema PLN reconhece o significado de cada morfema, a fim de entender o que designa cada palavra.

- **Estruturas sintáticas** - as estruturas sintáticas se concentram nas análises das palavras em uma determinada frase, descobrindo a estrutura gramatical da frase. Para a realização dessa tarefa, as estruturas sintáticas necessitam de uma gramática e de um analisador. Geralmente, a saída desse processo representa a sentença, revelando os relacionamentos entre as dependências estruturais das palavras na mesma frase. Nesse sentido, a sintaxe transmite um significado na maioria dos idiomas, porque a ordem e dependência das palavras contribui para o significado das sentenças. Por exemplo, as frases a) o gato correu atrás do rato e b) o rato correu atrás do gato, possuem sintaxe parecidas, mas transmitem significados diferentes.

- **Estruturas lexicais** - as estruturas lexicais nesse nível, tanto para os seres humanos quanto os assistentes PLN[1], que se preocupam em interpretar o significado das palavras de forma individual. Desse modo, o processamento contribui para o nível de entendimento da palavra, atribuindo uma etiqueta na parte-do-discurso de cada palavra. Nesse sentido, os léxicos que funcionam em mais de uma parte-do-discurso recebem a etiqueta mais provável da parte-da-fala, baseando-se no contexto os quais ocorreram.

- **Estruturas pragmáticas** - as estruturas pragmáticas preocupam-se com o uso da linguagem em determinadas situações, utilizando o contexto para a compreensão da sentença. O principal objetivo é de se interpretar o sentido pragmático frasal e essa tarefa requer um conhecimento extra sobre o mundo, incluindo, assim, o conhecimento sobre as intenções, os planos e objetivos da frase. Nesse sentido, o PLN pode utilizar-se de técnicas de inferências ou outras bases de conhecimentos.

- **Discurso** - O discurso no PLN trabalha com unidades de textos maiores, diferentemente da sintaxe e da semântica as quais trabalham com unidades de comprimento de frases curtas. Assim, o discurso se concentra nas propriedades do texto na frase, transmitindo um significado pelas conexões entre os componentes da frase, realizando o processamento para a resolução da anáfora (substituição das palavras) e o reconhecimento do discurso (sentenças do texto).

2.2 Parte-do-Discurso

A marcação da Parte-do-Discurso (do inglês *Part-Of-Speech* - PoS) é uma tarefa que consiste em rotular cada palavra com uma etiqueta única, capaz de categorizá-la sintaticamente [160, 34], por exemplo, verbo, advérbios, substantivo, dentre outras classes morfológicas. Essa tarefa introduz ambiguações (análise lexical) e desambiguações (eliminação de alternativas ilegítimas) pelos *tokenizadores* (sistemas baseado em regras capazes de identificar palavras, marcadores de documentos, sinais de pontuações e sintagmas fixos[2]) [173]. Tal tarefa é crucial em diversas aplicações, como, por exemplo, no reconhecimento de entidades mencionadas. Dessa forma, os melhores classificadores da PoS são baseados em técnicas de treinamento de janelas de textos, por algoritmos de decodificação bidirecionais durante o processo de inferência.

[1] São assistentes virtuais inteligentes que identificam padrões na fala pelo reconhecimento da voz. Alguns exemplos desses assistentes são as aplicações Siri da Apple e a Alexa da Amazon.

[2] Os sintagmas são formados pela constituição de unidades significativas de vários elementos dentro da sentença, mantendo a ordem e a dependência entre as suas unidades [35].

2.3 Análise

A tarefa de análise visa rotular segmentos de uma frase com constituintes sintáticos – frases substantivas (NP) ou verbais (VP) [160]. Outrossim, a cada palavra se atribui uma etiqueta exclusiva, codificada com uma marcação de início ou com uma marcação interna. Outro tipo de análise, conhecida como análise de dependência, destina-se a mostrar os relacionamentos entre as palavras em uma sentença.

2.4 Reconhecimento de Entidades Mencionadas

O reconhecimento de entidades mencionadas visa rotular os elementos atômicos de uma determinada sentença, categorizando-as como “pessoa” ou “localização” [34]. Assim, a cada palavra se pode atribuir uma etiqueta prefixada, indicando o léxico de início ou o anterior de uma entidade.

2.5 Rotulagem de Funções Semânticas

A rotulagem de funções semânticas é um processo de identificação e classificação de argumentos de um texto, visando atribuir um papel semântico ao constituinte sintático da frase [111]. Por esse processo, as tarefas de rotulagem são capazes de produzir uma análise em forma de uma árvore, identificando os nós da árvore, representando os argumentos de um determinado verbo, classificando-o para a realização de cálculos das etiquetas correspondentes. Em outras palavras, a rotulagem de funções semânticas identifica a estrutura de argumentos dos predicados de uma sentença, extraindo as relações semânticas entre o predicado e os argumentos relacionados.

2.6 Detecção Automática de Sinônimos

A detecção automática de sinônimos se baseia na teoria da semântica distributiva, realizando a correlação entre as palavras que ocorrem no mesmo contexto, com significados semelhantes. Como uma das relações semânticas mais conhecidas, a sinonímia tem sido objeto de numerosos estudos. Pela definição, sinônimos são palavras com significados idênticos ou semelhantes. A descoberta de relações de sinônimo pode ajudar a abordar várias aplicações do PLN, como, por exemplo, a recuperação de informações, respostas às perguntas, resumos automáticos de textos, gerações de idiomas, tarefas de substituições e as aquisições de vinculações lexicais [186].

2.7 Aplicabilidade do Processamento de Linguagem Natural

Os recentes avanços computacionais, especialmente na disponibilidade de grandes volumes de dados e no aumento exponencial da capacidade computacional, vêm possibilitando as aplicações das melhores técnicas do aprendizado profundo [169], corroborando com o PLN nas áreas da visão computacional e do reconhecimento de fala. Nesse sentido, há várias aplicações do PLN que contribuem nas comunicações entre máquinas e seres humanos de forma prática, como, por exemplo, máquinas que recebem instruções por voz para a execução de determinadas tarefas [160]. No entanto, o desenvolvimento de métodos de PLN dependem de abordagens orientadas a dados, implementando modelos mais robustos.

A Tabela 2.1 apresenta alguns dos trabalhos disponíveis na literatura que se utilizam das técnicas do PLN para a realização de tarefas específicas.

Tabela 2.1: *Aplicabilidade do Processamento de Linguagem Natural.*

Tarefa	Trabalhos
Reconhecimento da fala	[115] [177] [127] [87] [15] [8] [145] [14]
Compreensão da linguagem	[56] [47] [141] [174] [102] [148] [94] [107] [121] [66] [81] [185] [131] [28]
Análise lexical	[83] [122] [167] [120] [39] [138] [86] [189] [16] [153] [146] [140] [17]
Sistemas de diálogos	[179] [90] [99]
Gráficos de conhecimentos	[74] [191] [130] [67]
Recuperação de informações	[100] [101]
Análise de sentimentos	[10] [112] [21] [187]
Perguntas e respostas	[7] [2] [118]
Computação social	[119] [37] [51]
Aquisição de conhecimento	[180] [183]
Legendagem de imagens	[178] [45] [105] [150] [62] [188] [136] [65] [64] [117] [55]

Portanto, as aplicações do PLN incluem o reconhecimento da fala, a compreensão da linguagem falada, a análise lexical, os sistemas de diálogos, os gráficos de conhecimentos, a tradução automática, a própria análise, a recuperação de informações, a análise de sentimentos, as aplicações de perguntas à respostas, a computação social, a aquisição de conhecimento e a geração de linguagem natural em processos de legendagem de imagens [40].

Nesse sentido, a tarefa de legendar imagens se utiliza das técnicas do PLN para gerar frases descritivas para determinadas imagens. Para isso, a utilização da rede neural artificial e do aprendizado profundo obtiveram grandes sucessos nas soluções de uma variedade de problemas, sendo capazes de gerarem sentenças mais descritivas. Sendo

assim, no próximo capítulo são apresentados os conceitos teóricos sobre as redes neurais artificiais, bem como as arquiteturas aplicadas para a realização dessas tarefas, apoiadas pelas técnicas do aprendizado profundo .

CAPÍTULO 3

Redes Neurais Artificiais

O cérebro humano é conhecido como uma máquina, que possui uma capacidade enorme de desenvolver regras complexas, sendo uma inspiração para as Redes Neurais Artificiais (RNAs) [76]. Assim, as RNAs adquirem a capacidade de se modelar, simulando tarefas executadas pelo cérebro humano [91]. As aplicações de RNAs para solucionar problemas reais ganham visibilidade por causa da sua habilidade em aprender e generalizar conhecimentos, baseando-se em exemplos.

Umas das principais características das RNAs são as realizações de tarefas com processadores distribuídos paralelamente, com o poder de processar grandes volumes de dados por unidades de processamento simples, armazenando conhecimentos exponenciais disponíveis naturalmente [36, 76]. Outras características que se destacam nas RNAs são: i) a capacidade de aprendizado adaptativo e generalizado; ii) as habilidades com tolerância a falhas; iii) a consideração dos fenômenos físicos não-lineares; iv) a aplicação da natureza contínua no mapeamento de entradas/saídas; v) a capacidade inserida de contextualização das informações e a dispensa de conhecimento estatísticos avançados em relação ao ambiente.

Nos últimos anos, há uma percepção enorme de aplicações baseadas nas RNAs em diferentes áreas do conhecimento [4, 9, 73], como, por exemplo, nas áreas da saúde, agricultura, segurança pública, transporte, dentre outros setores.

A abordagem convencional da computação se baseia em um conjunto explícito de instruções programadas [32]. No entanto, as RNAs apresentam paradigmas diferentes em relação à abordagem convencional e a solução para um determinado problema é inspirada no aprendizado que se baseia em exemplos, em um conjunto de referências. Essa inspiração se origina de estudos dos mecanismos de processamento de informações do sistema nervoso biológico, especificamente, do cérebro humano.

Uma das percepções baseadas nos estudos atuais sobre RNAs é que grande parte desses trabalhos estão focados em obter uma compreensão mais profunda do processamento de informações nos sistemas biológicos. Entretanto, os conceitos básicos também podem ser entendidos a partir de uma abordagem restritiva - a abstração do processamento de informações [157].

Nesse sentido, umas das RNAs que se destacam são as redes *feed-forward*. Uma rede neural *feed-forward* pode ser considerada como uma função matemática não-linear, transformando um conjunto de variáveis de entradas em um conjunto de variáveis de saída [49]. Essas transformações entre o conjunto de entradas e saídas são governadas por um conjunto de parâmetros conhecidos, como pesos. Os valores desses pesos podem ser determinados com base em um conjunto mapeado de exemplos. O processo que determina esses valores dos parâmetros é chamado de "aprendizado" ou "treino", podendo ser uma tarefa computacionalmente intensiva [134].

3.1 Rede Neural Padrão

Para entender a estrutura de uma Rede Neural Convolucional (CNN), necessita-se da compreensão básica da estrutura de uma RNA, visto que as redes convolucionais são baseadas nessas mesmas estruturas. Dessa forma, um neurônio artificial pode ser definido como uma unidade de processamento simples, que possui um número fixo de entradas x_n, conectadas aos neurônios por *links* ponderados (w_n) (Figura 3.1).

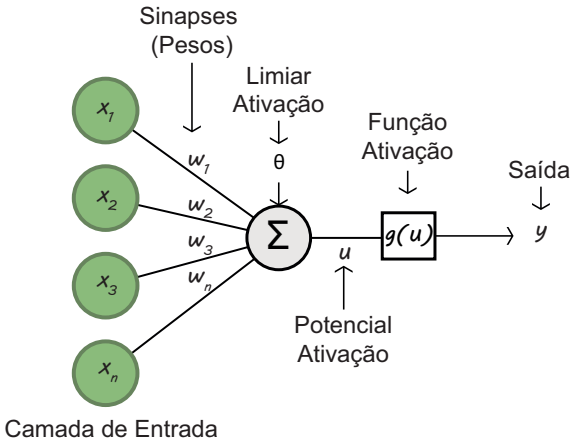


Figura 3.1: *Exemplo de um neurônio artificial simples.*

onde θ representa o limiar de ativação, especificando o patamar apropriado para que o resultado produzido pelo combinador linear $\sum$ possa gerar valores de disparos em direção à saída do neurônio. O potencial de ativação u representa a diferença do valor produzido entre o limiar de ativação θ e o combinador linear $\sum$.

Assim, o neurônio artificial resume a entrada, de acordo com a seguinte equação:

$$\mathbf{u} = \sum_{i=1}^{n} x_i \cdot w_i - \theta \tag{3-1}$$

onde $\sum_{i=1}^{n}$ é o somatório de todas as entradas (x_i) multiplicadas pelos seus pesos (w_i).

Para a realização do cálculo da saída de um determinado neurônio, pode-se aplicar uma função de ativação (g), atualizando-se uma função linear ou não-linear:

$$\mathbf{g}(\mathbf{u}) = \begin{cases} if(u < 0), & 0 \\ if(u \geq 0), & 1 \end{cases} \tag{3-2}$$

$$\mathbf{g}(\mathbf{u}) = \frac{1}{1 + e^{-\beta u}} \tag{3-3}$$

onde u é o potencial de ativação e β é uma constante real, que representa o nível de inclinação da função logística.

Como dito anteriormente, o neurônio artificial é um modelo abstrato do neurônio biológico. Analogicamente, a força de uma conexão neural é codificada nos pesos. A intensidade do sinal de entrada pode ser modelada, utilizando-se o número real ao invés de um somatório temporal de picos.

Nesse sentido, o neurônio artificial trabalha em intervalos de tempo discretos, sendo que as entradas são lidas e processadas em um momento no tempo. Assim, existem muitos métodos de aprendizado diferentes possíveis para um único neurônio, como, por exemplo, métodos supervisionados e semi-supervisionados [114]. A maioria dos métodos supervisionados se baseiam na ideia da alteração dos pesos na direção, em que a diferença entre a saída calculada e a saída esperada diminui, aplicando o algoritmo de retropropagação.

A retropropagação se tornou um dos algoritmos mais populares para o treinamento de redes neurais artificiais. Essa popularidade ocorre devido a sua capacidade de otimizar os pesos, para que a rede neural possa aprender, de forma correta, mapeando as suas entradas para suas saídas (multidimensional) [24].

3.1.1 Decida do Gradiente, Delta e Derivada

A regra da decida do gradiente está relacionada ao aprendizado que opera em uma relação diferenciada de ativação [18]. Dessa forma, a descida do gradiente pode ser definida como um algoritmo otimizado, capaz de encontrar os parâmetros w (pesos) e β (bias) de uma determinada função, com o intuito de minimizar a função de custo (Figura 3.2).

Para minimizar a função de custo, pode-se inserir valores iniciais para os coeficientes w (pesos) e β, aplicando pequenos valores aleatórios, a fim de chegar ao mínimo global. Observa-se que cada alteração nos valores dos parâmetros equivale a uma interação no gradiente ou a um passo em direção ao mínimo. Nesse sentido, a inicialização dos pesos possui grande impacto no processo da decida do gradiente, possibilitando a aplicação de variados métodos para essa finalidade [13].

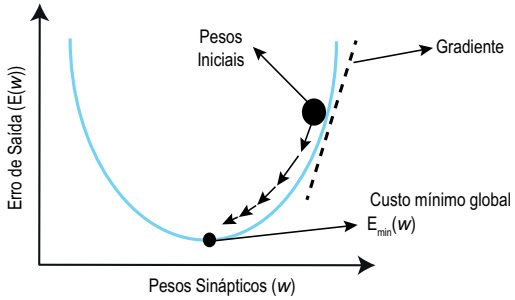


Figura 3.2: *Exemplo da descida do gradiente com um mínimo global.*

As atualizações dos pesos são funções dos vetores de entradas (x), das saídas calculadas f(líquida), da derivada da saída calculada f'(líquida), da saída esperada d e da constante de aprendizado η.

$$\mathbf{net} = x^t \cdot w \tag{3-4}$$

$$\Delta \mathbf{w} = \eta f'(net)(d - f(net))x \tag{3-5}$$

A regra delta ($\Delta \mathbf{w}$) possui a função de alteração dos pesos, com o intuito de minimizar os erros das redes neurais. O erro pode ser definido pela diferença entre a saída calculada e a saída desejada. Desse modo, os pesos são ajustados para um padrão em uma determinada etapa do aprendizado. Esse processo é repetido, com o objetivo de identificar um vetor de pesos, que possibilite a minimização do erro para todo o conjunto de treinamento da rede neural.

Não obstante, um conjunto de pesos só pode ser identificado se o conjunto de treinamento for separável linearmente. Essa limitação independe do algoritmo de aprendizado adotado, podendo ser, simplesmente, derivada de uma estrutura de neurônio único (Figura 3.3).

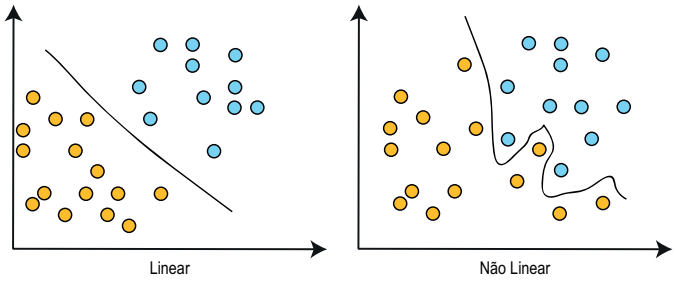


Figura 3.3: *Exemplos de conjuntos de treinamentos separáveis linearmente e não-linearmente.*

A separabilidade também é uma propriedade que pode implicar em problemas de otimização, pois funções com variáveis interrelacionadas (chamadas de não-separáveis), geralmente apresentam problemas para a convergência dos algoritmos [170].

3.1.2 Camadas de Entrada, Oculta e Saída

Uma rede neural pode ser implementada por um conjunto com vários neurônios simples, possibilitando que a saída de um determinado neurônio seja a entrada de outro neurônio (Figura 3.4).

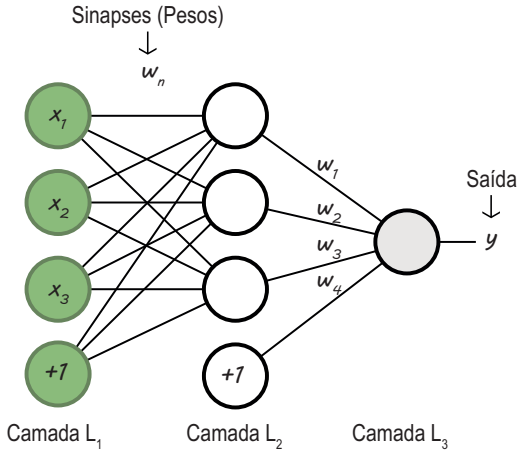


Figura 3.4: *Exemplo do modelo de uma rede neural complexa.*

A Figura 3.4 apresenta uma rede neural composta por um conjunto de neurônios de entrada, também chamada de camada de entrada (camada L_1), a camada oculta, cujos valores não são observados no conjunto de treinamento (camada L_2) e a camada de saída (camada L_3). Os círculos com a marcação "+1" representam as unidades de polarizações (bias ou viés) e correspondem ao termo de interceptação.

3.1.3 Função de Ativação

A função de ativação objetiva a processar em uma unidade de neurônio um limitador de saída. A escolha dessa função se torna importante pelo fato de que ela possibilita as definições dos dados de saída [151].

Nesse contexto, em sistemas de redes neurais, existem vários tipos de funções de ativação, dentre elas:

- **Função linear –** a função linear pode ser considerada a mais básica entre todas as funções. Essa função não realiza as alterações de saída de um neurônio. Geralmente,

é aplicada na camada de saída nas redes neurais de regressão, tendo a sua expressão matemática definida por:

$$\mathbf{g}(\mathbf{u}) = u \tag{3-6}$$

A representação gráfica da função linear é ilustrada na Figura 3.5.

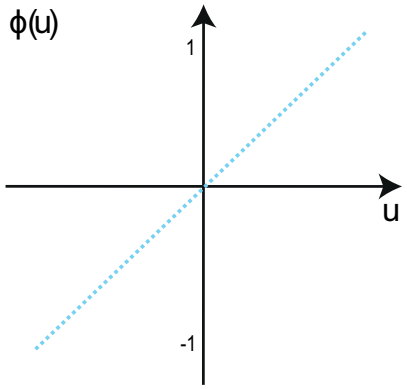


Figura 3.5: *Exemplo da função linear.*

- **Função limiar –** a função de ativação com limite também é considerada uma função simples ou função degrau, resumindo-se em definir uma saída de 1 ou 0, seguindo um limite previamente estabelecido, tendo a sua expressão matemática definida por:

$$\mathbf{g}(\mathbf{u}) = \begin{cases} 1, & se \quad u \geq 0 \\ 0, & se \quad u < 0 \end{cases} \tag{3-7}$$

A representação gráfica da função limiar é ilustrada na Figura 3.6.

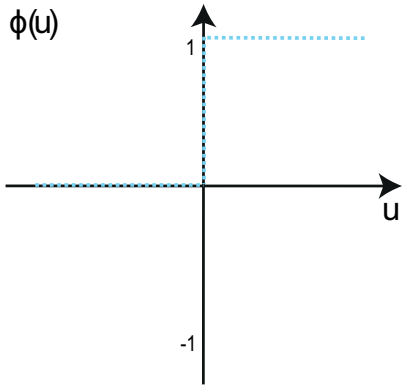


Figura 3.6: *Exemplo da função limiar.*

- **Função *sigmoide* –** a função *sigmoide* é amplamente utilizada em redes neurais *feed-forward* e em redes multicamadas, restringindo as suas saídas em números

positivos, com sinais contínuos, tendo a sua expressão matemática definida por:

$$\mathbf{g}(\mathbf{u}) = \frac{1}{1+e^{-u}} \tag{3-8}$$

A representação gráfica da função *sigmoide* é ilustrada na Figura 3.7.

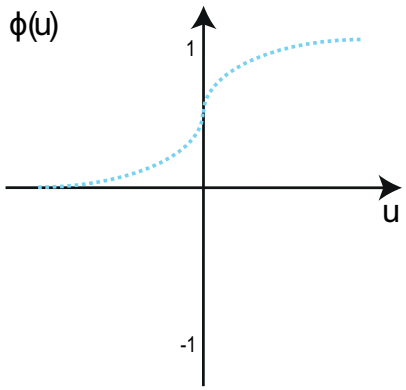


Figura 3.7: *Exemplo da função sigmoide.*

- **Função tangente hiperbólica –** a função tangente hiperbólica se caracteriza por ser uma das escolhas mais adequadas para as redes neurais, além de possuir um grande uso geral, a sua aplicabilidade permite saídas de valores entre -1 e 1, tendo a sua expressão matemática definida por:

$$\mathbf{g}(\mathbf{u}) = \frac{1-e^{-u}}{1+e^{-u}} \tag{3-9}$$

A representação gráfica da função tangente hiperbólica é ilustrada na Figura 3.8.

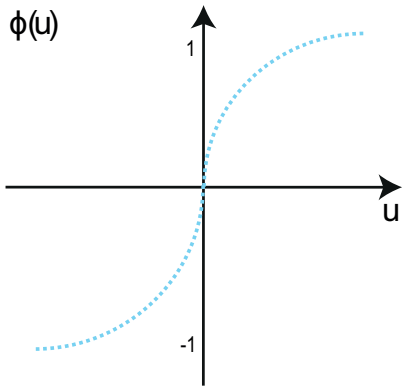


Figura 3.8: *Exemplo da função tangente hiperbólica.*

- **Função logística –** a função logística se caracteriza por produzir uma saída que assumirá sempre valores entre 0 e 1 (similar a função *sigmoide*), tendo a sua expressão matemática definida por:

$$\mathbf{g}(\mathbf{u}) = \frac{1}{1+e^{-u}} \tag{3-10}$$

A representação gráfica da função logística é ilustrada na 3.9.

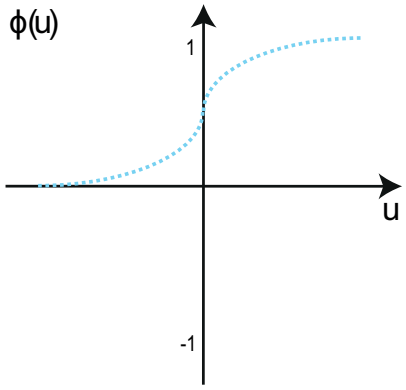


Figura 3.9: *Exemplo da função logística.*

- **Função *softmax* –** a função *softmax* é amplamente utilizada em redes neurais de classificação, fazendo com que a saída seja uma representação probabilística. Dessa forma, a *softmax* possibilita, pelas suas saídas, a probabilidade de um determinado dado ser de uma classe pré-definida, de forma que a sua saída não passe de apenas valores numéricos, cuja classe vencedora é a que possui maior probabilidade, tendo a sua expressão matemática definida por:

$$\mathbf{g}(\mathbf{u}) = \frac{e^{u_j}}{\sum_{j=1}^{k} e^{u_j}} \tag{3-11}$$

A representação gráfica da função *softmax* é ilustrada na Figura 3.10.

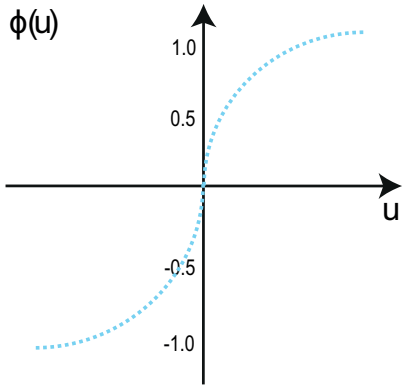


Figura 3.10: *Exemplo da função softmax.*

De acordo com Nair e Hinton [125], a função *softmax* não é recomendada para as camadas ocultas totalmente conectadas. Não obstante, em redes neurais é comum a aplicação de funções *sigmoidais*, como, por exemplo, a função logística e a tangente hiperbólica.

Entretanto, no aprendizado profundo, a função retificadora linear (do inglês *rectified linear function*, ReLU) tem sido mais utilizada por facilitar o aprendizado no

processo de treinamento. Isso ocorre porque as funções *sigmoidais* saturam a partir de um determinado ponto, enquanto a função ReLU se torna, simplesmente, uma função de identidade para os valores positivos. A função retificadora linear tem a sua expressão matemática definida por:

$$\mathbf{g}(\mathbf{u}) = \begin{cases} 0 & se \quad u < 0 \\ u & se \quad u \geq 0 \end{cases} \tag{3-12}$$

A representação gráfica da função retificadora linear é ilustrada na Figura 3.11.

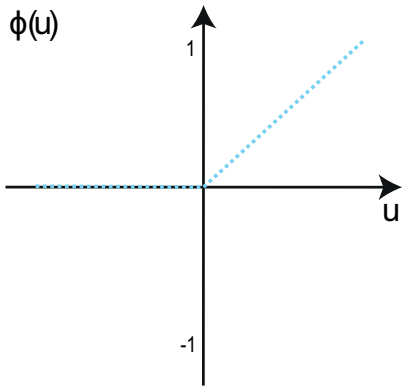


Figura 3.11: *Exemplo da função retificadora linear.*

3.1.4 Problemas com Underfitting e Overfitting

Durante o treinamento de uma rede neural podem ocorrer dois problemas: *overfitting* ou *underfitting* [96]. No primeiro, a rede neural é treinada para um conjunto específico de casos, mas não é capaz de avaliar outros conjuntos de situações. O segundo caso é caracterizado quando o conjunto de testes realizados na rede neural não é suficiente para se desenvolver um modelo de abstração de problemas de forma adequada, apresentando baixo desempenho nas classificações.

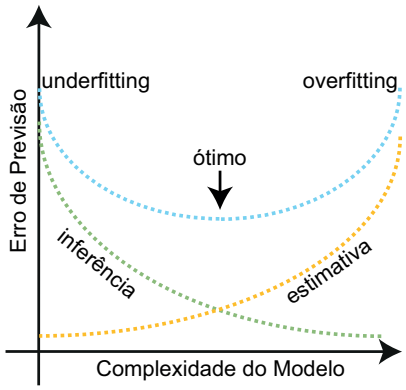


Figura 3.12: *Problemas de overfitting e underfitting.*

Quando ocorre o *overfitting*, a fase de treinamento da rede neural tem bons resultados, contudo, nos testes os resultados são ruins [78]. Por outro lado, quando ocorre o *underfitting*, a rede neural obtém resultados ruins tanto no treinamento quanto nos testes.

O *overfitting* normalmente ocorre quando uma rede neural muito adaptável é treinada, com um conjunto de dados que contém ruídos do sistema. Nesses casos, a rede estrutura o seu modelo, de acordo com o conjunto de dados que representa o erro, mas não um modelo geral. A literatura apresenta diversas formas de se combater esse problema, como interromper o treinamento assim que a rede neural começa a ter resultados piores ou adotando penalidades aos pesos [155, 78]. Em um estudo mais recente, Srivastava et al. [155] apresentaram uma solução adicional chamada *Dropout*.

O *Dropout* é um método para regularizar algoritmos de aprendizado de máquina (Figura 3.13).

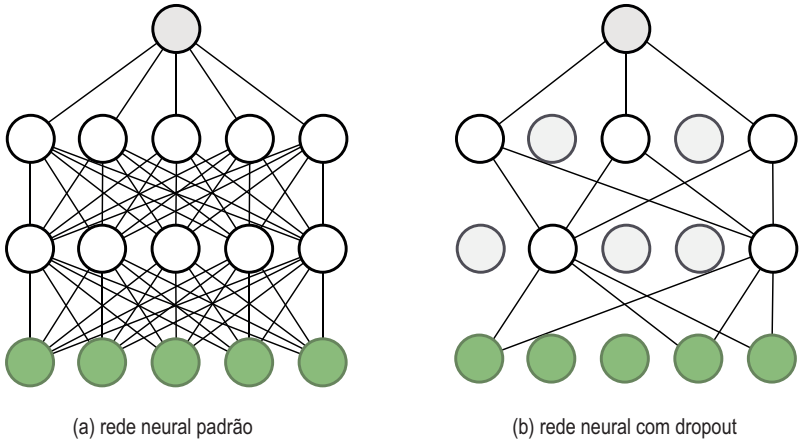


Figura 3.13: *Exemplo do modelo de rede neural modelada com base no Dropout.*

A ideia principal de uma camada *dropout* é desabilitar, aleatoriamente, as unidades de entrada após cada iteração do treinamento. Uma rede neural com n unidades que emprega *dropout* pode ser vista como uma coleção de 2^n redes possíveis. Essas redes possíveis têm um número menor de unidades, mas ainda compartilham pesos, de forma que o número total de parâmetros não seja alterado [155].

Segundo Srivastava et al. [155], o termo *dropout* se refere ao abandono de unidades ocultas e visíveis em uma rede neural. Assim, ao retirar uma dessas unidades, elas são removidas temporariamente, juntamente com todas as suas conexões de entradas e saídas.

A seleção de quais unidades deverão ser removidas é realizada de forma aleatória. Por exemplo, considere uma rede neural de camada única com unidades lineares:

$$\mathbf{net} = \sum w_j I_j \tag{3-13}$$

Observa-se que essa rede neural é chamada de linear, justamente por causa da sua função de ativação linear, ou seja, $f(x) = x$. Portanto, é fácil perceber que a saída da rede neural é um somatório ponderado linear de todas as entradas. Assim, aplicando-se a perda mínima do quadrado comum, obtém-se:

$$\mathbf{E_n} = \frac{1}{2}\left(t - \sum_{i=1}^{n} w_i^{'} I_i\right)^2 \tag{3-14}$$

$$\mathbf{E_d} = \frac{1}{2}\left(t - \sum_{i=1}^{n} \delta_i w_i I_i\right)^2 \tag{3-15}$$

A equação 3-14 apresenta a função de perda para uma rede neural padrão e a equação 3-15 para uma rede com *dropout*. Nesse sentido, as redes de retropropagação se utilizam da abordagem da descida do gradiente durante o seu treinamento. Dessa forma, aplicando-se essa abordagem na rede com *dropout* se tem:

$$\frac{\partial \mathbf{E_d}}{\partial \mathbf{w_i}} = -t\delta_i I_i + w_i \delta_i^2 I_i^2 + \sum_{j=1, j\neq i}^{n} w_i \delta_i \delta_j I_i I_j \tag{3-16}$$

A fim de identificar uma relação entre os gradientes da rede neural com *dropout* e a rede neural padrão, aplica-se $w^{'} = p * w$.

$$\mathbf{E_n} = \frac{1}{2}\left(t - \sum_{i=1}^{n} p_i w_i I_i\right)^2 \tag{3-17}$$

$$\frac{\partial \mathbf{E_n}}{\partial \mathbf{w_i}} = -t p_i I_i + w_i p_i^2 I_i^2 + \sum_{j=1, j\neq i}^{n} w_j p_i p_j I_i I_j \tag{3-18}$$

Assim, obtém-se a expectativa do gradiente da rede neural com *dropout*.

$$\mathbf{E}\left[\frac{\partial \mathbf{E_d}}{\partial \mathbf{w_i}}\right] = \frac{\partial E_n}{\partial w_i} + w_i p_i (1 - p_i) I_i^2 \tag{3-19}$$

De forma mais simples, cada unidade deve ser mantida com uma probabilidade fixa, independentemente de outras unidades, em que p pode ser selecionado, utilizando-se de um conjunto de validação ou definindo-se em 0.5 a taxa de *dropout* (probabilidade de se retirar um neurônio da camada oculta ou das camadas vissíveis). Entretanto, para as unidades de entrada, a probabilidade ideal de abandono é, geralmente, mais próxima de 1.0 [155].

Contudo, outros estudos mais recentes comprovam que interromper o treinamento previamente é o método mais adequado para combater tanto o problema de *underfitting* quanto o *overfitting* [79].

3.2 Rede Neural Convolucional

A partir da disponibilização do código fonte, que implementa a Rede Neural Convolucional (CNN) denominada AlexNet, os seus resultados significativos que apontam para uma indiscutível performance estatística para a classificação de imagens, aplicando o aprendizado profundo, as CNNs vêm atraindo a atenção da comunidade científica. Nesse sentido, o aprendizado profundo passou a ser bastante aplicado, principalmente no processamento de imagens digitais associados às CNNs [92].

O aprendizado profundo é o método mais utilizado para guiar o aprendizado de um modelo, como, por exemplo, modelos de regras e parâmetros, por um conjunto de dados [137]. O processo do aprendizado profundo finaliza com uma função, que possui a habilidade de receber dados brutos na sua entrada e fornecer uma representação adequada para um determinado problema.

Nesse contexto, as CNNs são uma das principais categorias de redes neurais aplicadas para o reconhecimento, classificação e detecção de imagens [58]. Desse modo, a tarefa de classificação de imagem consiste em realizar o processamento de uma imagem de entrada e classificá-la de acordo com os parâmetros já definidos no modelo. Os computadores enxergam essa imagem de entrada como uma matriz de *pixels* de tamanhos variados, dependendo da sua resolução. Com base nessa resolução é que os computadores, utilizando-se de um aprendizado profundo, mensuram a altura, a largura e a dimensão de uma imagem [58].

Uma CNN é muito semelhante a uma rede neural comum. Ambas são constituídas por neurônios que possuem pesos e vieses aprendíveis. Cada neurônio recebe um conjunto de entradas, executando um produto escalar e, opcionalmente, o segue com uma não-linearidade. Tanto na CNN quanto nas redes neurais comuns, ainda expressam uma função de perda na última camada (camada totalmente conectada). Assim, de acordo com Srivastava et al. [155], as características aplicáveis para o entendimento sobre as redes neurais comuns, aplicam-se às CNNs.

Uma das diferenças entre essas duas redes, redes neurais comuns e CNN, é o fato de que a CNN aproveita que as entradas das imagens são consistentes, restringindo a arquitetura de forma mais adequada, por camadas de convoluções [92]. Ainda, diferentemente das redes neurais comuns, a CNN possui neurônios organizados em três dimensões (largura, altura e profundidade) [133].

Assim, observa-se que uma CNN organiza os seus neurônios em três dimensões, conforme visualizado em uma das camadas. Cada camada de uma CNN transforma o volume de entrada 3D em um volume de saída 2D de ativações de neurônios [175]. No exemplo demonstrado pela Figura 3.14, a camada de entrada (a) representa a largura da imagem e (b) representa a altura. A profundidade da imagem é classificada em três canais

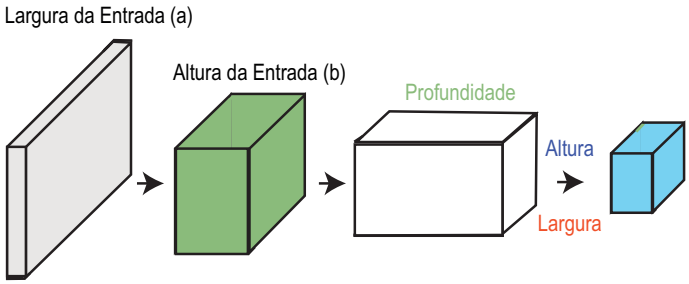


Figura 3.14: *Exemplo das três dimensões de uma CNN.*

(vermelho, verde e azul)[1].

Tecnicamente, em um modelo CNN, cada imagem de entrada passa por uma série de camadas: as camadas de convoluções com filtros (*kernels*), as camadas de *pooling* e as camadas profundas totalmente conectadas (FC). Após esse processo, aplica-se a função *softmax*, a fim de classificar um objeto com valores probabilísticos entre 0 e 1 [27]. A Figura 3.15 apresenta um modelo de CNN completo, aplicado no processamento de imagens, classificando os objetos, de acordo com uma base de valores.

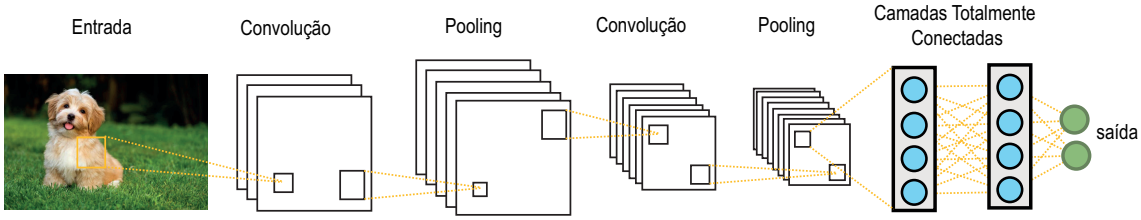


Figura 3.15: *Exemplo completo de uma CNN para a classificação de objetos em imagens.*

Uma CNN aproveita o fato de que uma imagem é composta por vários recursos[2] e cria um mecanismo para analisar cada recurso de forma isolada [137]. Para a realização dessa tarefa, a CNN conta com vários tipos de camadas:

- **Camada de convolução –** em cada camada de convolução se aplica um filtro na imagem de entrada, em que cada *pixel* da imagem é digitalizado, criando um mapa de características. Dessa forma, a camada de convolução é capaz de predizer a classe referente de cada recurso filtrado na imagem de entrada.
- **Camada de agrupamento (*pooling*) –** a camada de agrupamento diminui a quantidade de informações por recursos, guardando apenas as informações mais relevantes, advindas da camada de convolução.

[1]Pode ser aplicada aos padrões matiz, saturação e valor - HSV ou matiz, saturação e luminosidade - HSL.

[2]São atributos extraídos das imagens de entrada, como, por exemplo, as bordas das imagens, cores e medidas.

- **Camada de entrada –** a camada de entrada recebe a saída das camadas anteriores, deixando-as nos mesmos níveis, transformando-as em um vetor. Esse vetor possui a função, em um próximo estágio, de ser a entrada para outra camada. Nesta primeira camada de entrada são analisados os recursos, aplicando os pesos adequadamente.
- **Camada de saída –** a camada de saída fornece, para o conjunto de entradas, saídas com as probabilidades finais. Tanto a camada de entrada quanto a camada de saída são totalmente conectadas.

3.2.1 Camada de Convolução

A camada de convolução é a primeira a extrair recursos de uma imagem de entrada. Dessa forma, a convolução preserva o relacionamento entre os *pixels*, aprendendo os recursos de uma determinada imagem, utilizando-se pequenos quadrados de dados de entrada [97]. Essa tarefa é uma operação matemática, que recebe duas entradas: a matriz da imagem e o filtro.

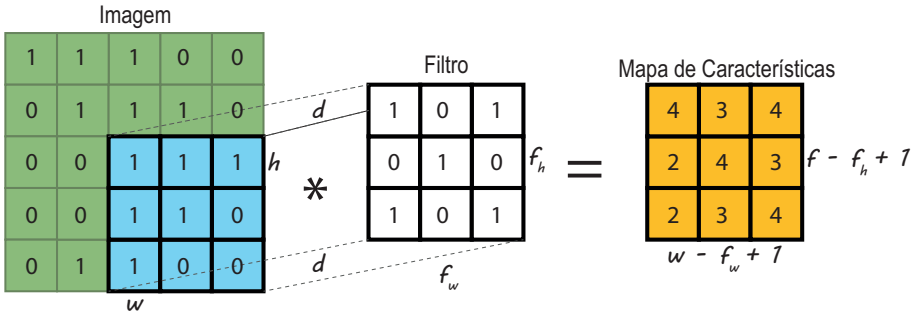


Figura 3.16: *Exemplo da entrada da camada de convolução com a matriz (f) e filtro (g).*

Em outras palavras, a convolução é uma operação entre duas funções (f e g), gerando uma terceira função, que pode ser interpretada como uma função modificadora de f. Nas tarefas de processamento de imagens, em que uma imagem pode ser definida como uma função bidimensional, a convolução se torna útil para a detecção e a suavização de bordas das imagens, para a extração de atributos, dentre outros aspectos [92]. Assim, dadas duas entradas $f(x)$ e $g(x)$ para uma variável continua x, a convolução pode ser definida como:

$$\mathbf{f}(\mathbf{x}) * \mathbf{g}(\mathbf{x}) = \int_{-\infty}^{\infty} f(t)g(x-t)dt \tag{3-20}$$

onde $(*)$ representa um operador de convolução. Para $f(x)$ e $g(x)$ quando x é definido em um conjunto de $\mathbb{Z}$ de inteiros, a equação de convolução discreta pode ser definida como:

$$\mathbf{f}(\mathbf{x}) * \mathbf{g}(\mathbf{x}) = \sum_{n=-\infty}^{\infty} f(n)g(x-n) \tag{3-21}$$

Ampliando-se a equação da convolução discreta para funções com duas variáveis x e y, obtém-se as seguintes equações:

$$\mathbf{f}(\mathbf{x},\mathbf{y}) * \mathbf{g}(\mathbf{x},\mathbf{y}) = \int_{t_1=-\infty}^{\infty} \int_{t_2=-\infty}^{\infty} f(t_1,t_2) g(x-t_1, y-t_2) dt_1 dt_2 \tag{3-22}$$

$$\mathbf{f}(\mathbf{x},\mathbf{y}) * \mathbf{g}(\mathbf{x},\mathbf{y}) = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f(n_1,n_2) g(x-n_1, y-n_2) \tag{3-23}$$

3.2.2 Camada de Pooling

Atualmente, a arquitetura de uma CNN é dividida em uma série de estágios e os primeiros estágios são compostos por dois tipos de camadas: a camada de convolução e a camada de *pooling*. Nesse sentido, a camada de *pooling* é uma forma de *downsampling*, computando o máximo ou médio local de uma determinada região no mapa de características de uma imagem [158].

A função da camada de *pooling* é reduzir, progressivamente, o tamanho da representação, diminuindo a quantidade de parâmetros e o custo computacional de uma rede neural. Portanto, a camada de *pooling* controla também o excesso de ajustes, operando de forma independente a cada fatia de profundidade da imagem de entrada. A camada *pooling* redimensiona a entrada, utilizando-se a operação de *pool* [158].

Uma das formas mais comum em se utilizar a camada *pooling* é a aplicação de filtros de tamanho $2x2$, ou seja, dois *pixels* de largura por dois de altura, descartando, assim, 75% das ativações. Nesse caso, cada operação de *pooling* máximo levaria no máximo quatro números de $2x2$ [97].

De forma geral, a camada de *pooling* é responsável por receber as saídas de um mapa de características das camadas de convoluções, transformando-as em um mapa condessado com essas características. Assim, para cada unidade da camada de *pooling* são reservados na região espacial $2x2$ neurônios da camada anterior (Figura 3.17).

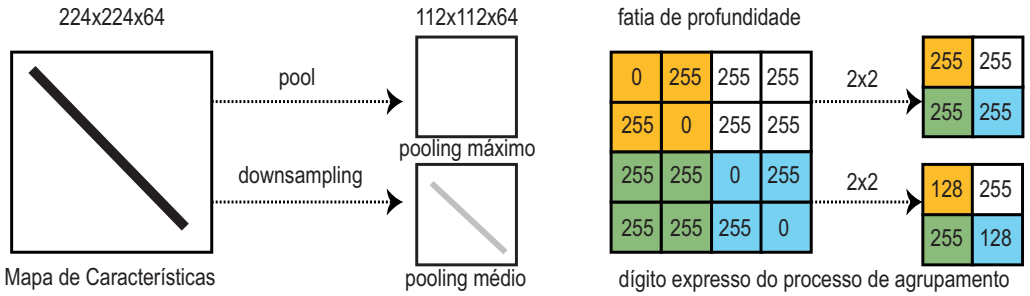


Figura 3.17: *Exemplo da comparação entre pool médio e pool máximo.*

Assim, percebe-se que a camada de *pooling* reduz, significativamente, a amostra do volume espacial, independentemente da fatia de profundidade da imagem de entrada. Verifica-se, também, que o volume de entrada de tamanho $224x224x64$ é agrupado com um filtro de tamanho $2x2$, com um volume de saída de tamanho $112x112x64$. Ademais, observa-se que, o volume é preservado.

Neste contexto, pode-se aplicar três formatos de *pooling* para condensar o mapa de características das camadas de convoluções:

- **Pooling máximo -** responsável por gerar a ativação máxima de uma determinada região (entrada $2x2$). Dessa forma, pode-se aplicar o *pooling máximo* em cada mapa de característica gerada pelas as camadas de convoluções, de forma separada, uma por vez.

$$\mathbf{h_i}(\mathbf{n}) = \max_{i \in N(n)} \tilde{h}[i] \tag{3-24}$$

- **Pooling médio -** responsável por gerar o valor médio dos mapas de características, realizando cálculos de todas as somas dos quadrados das ativações, ocorridas em uma determinada região.

$$\mathbf{h_i}(\mathbf{n}) = \frac{1}{n} \sum_{i \in N(n)} \tilde{h}[i] \tag{3-25}$$

- ***Pooling L2* -** reduz as informações advindas das camadas de convoluções, realizando cálculos da raiz quadrada de todas as somas dos quadrados das ativações ocorridas em uma determinada região de $2x2$. Desse modo, o *pooling* L2 realiza o agrupamento máximo das regiões.

$$\mathbf{h_i}(\mathbf{n}) = \frac{1}{n} \sqrt{\sum_{i \in N(n)} \tilde{h}^2[i]} \tag{3-26}$$

Outro aspecto importante sobre a camada de *pooling* é que em dois ou três estágios convolucionais outras funções não-lineares e de *pooling* são empilhadas, seguidos de mais camadas totalmente conectadas. Tanto as camadas convolucionais quanto as camadas de *pooling* são inspiradas pelos modelos clássicos de células simples ou de células complexas da neurociência visual [168].

3.2.3 Camada Totalmente Conectada

A Camada Totalmente Conectada (do inglês *Fully Connected* - FC) comprime uma matriz em um vetor (Figura 3.18) [3].

Observa-se que, pela Figura 3.18, a matriz do mapa de características é convertida como um vetor (x_1, x_2, x_3, x_4) e, assim, as FCs combinam esses recursos, criando

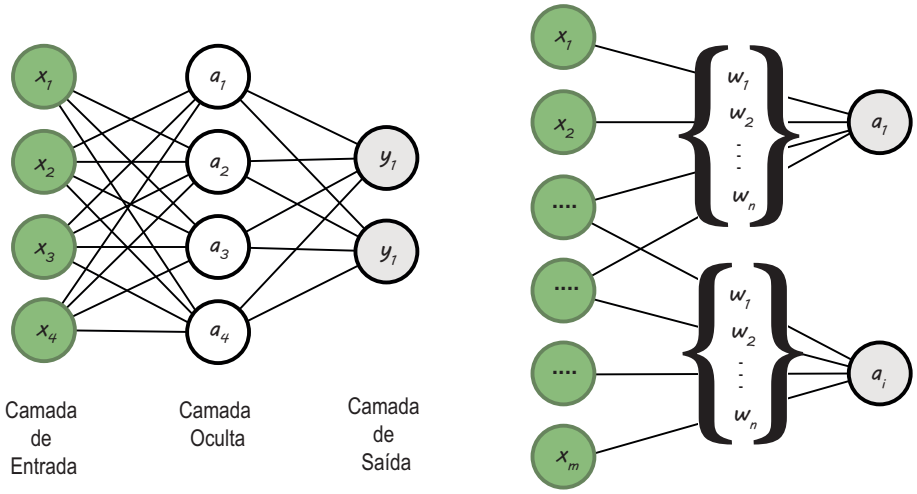


Figura 3.18: *Exemplo do modelo de uma camada totalmente conectada - FC.*

um modelo. Por fim, aplica-se a função de ativação, *softmax* ou *sigmoide*, classificando as saídas.

As FCs são componentes essenciais das CNNs, que têm sido comprovadamente bem-sucedidas em tarefas de reconhecimento e classificação de imagens na área da visão computacional [43]. O processo da CNN começa com a convolução e agrupamento, dividindo a imagem de entrada em recursos e analisando-as de forma independente (Figura 3.19). O resultado desse processo é alimentado por uma estrutura de Rede Neural Totalmente Conectada (FCNN), que conduz a decisão final de classificação [61].

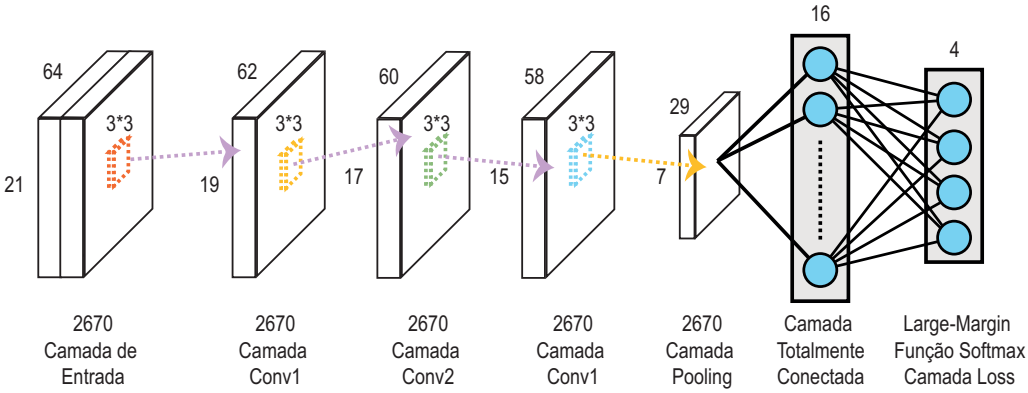


Figura 3.19: *Exemplo da arquitetura completa de uma CNN com FCs e softmax.*

De forma resumida, o objetivo de uma FC é extrair os resultados do processo de convolução/*pool* e usá-los, classificando imagens em rótulos [75]. A saída da convolução/*pool* é achatada em um único vetor de valores, cada um representando uma probabilidade de que um determinado recurso pertença a uma determinada classe, categoria ou rótulo.

3.2.4 Otimizador RMSProp e a Técnica Grampo do Gradiente

A estrutura do treinamento de uma rede neural é determinada por uma série de parâmetros que se ajustam, por exemplo, à taxa de aprendizagem da respectiva rede. Esses parâmetros devem ser configurados antes do treinamento. O algoritmo do treinamento tem o seu comportamento definido por tais parâmetros.

Nesse sentido, a taxa e a aprendizagem é uma constante global, identificada por métodos tradicionais. Entretanto, uma taxa alta de aprendizado ocasiona um aumento nos erros de perdas, resultando em alterações excessivas dos pesos da rede neural. Uma taxa baixa pode evitar esse problema, mas aumenta o tempo de convergência da rede neural [60].

Sendo assim, a escolha adequada dos parâmetros provém de uma busca eficiente no espaço definido do problema e melhora o gerenciamento para um conjunto grande de experimentos.

De acordo com Mantovani [116], a definição dos parâmetros deve ser feita como uma otimização de caixa preta, objetivando a otimização do algoritmo de treinamento. A otimização desses critérios deve ser realizada, de acordo com a medição do desempenho de uma ou mais medidas, para atingir, respectivamente, um ou vários objetivos específicos. Entretanto, deve-se evitar uma definição de solução geral para o treinamento, tornando a otimização do algoritmo uma tarefa difícil. Assim, cada problema deve ser analisado cuidadosamente, para que a melhor solução seja adotada. As classificações dos diferentes tipos de soluções são apresentadas por Cala [20]: a descida estocástica do gradiente (SGD); momentos; estratégias de parâmetros inicializados e algoritmos com taxas de aprendizado adaptáveis.

Nesse contexto, pode-se aplicar um otimizador, como, por exemplo, o RMSProp ou AdaGrad, sendo algoritmos de otimização que aplicam taxas de aprendizados adaptáveis.

O algoritmo Adagrad adapta, individualmente, as taxas de aprendizado de todos os parâmetros do modelo, escalando-as como inversamente proporcionais à raiz quadrada da soma de todos os valores quadrados dos históricos do gradiente [48], tendo a sua expressão matemática definida por:

$$\mathbf{v_t^w} = v_{t-1}^{w} + (\bigtriangledown w_t)^2 \tag{3-27}$$

$$\mathbf{w_{t+1}} = w_t - \frac{\eta}{\sqrt{v_t^w + \varepsilon}} * \bigtriangledown w_t \tag{3-28}$$

$$\mathbf{v_t^b} = v_{t-1}^{b} + (\bigtriangledown b_t)^2 \tag{3-29}$$

$$\mathbf{b_{t+1}} = b_t - \frac{\eta}{\sqrt{v_t^b + \varepsilon}} * \bigtriangledown b_t \tag{3-30}$$

onde v representa o histórico de atualização acumulativo do gradiente e η a taxa de aprendizado. Dessa forma, quanto menor for o gradiente acumulado, maior será a taxa de aprendizado.

No Adagrad, os parâmetros com a maior derivada parcial da função de perda têm uma diminuição rápida correspondente na sua taxa de aprendizado, enquanto os parâmetros com pequenas derivadas parciais têm uma diminuição relativamente pequena (Algoritmo 1.1).

Algoritmo 3.1: Pseudocódigo do algoritmo AdaGrad

Requerer: Taxa de aprendizagem global ε;
Requerer: Inicialização de parâmetro Θ;
Requerer: Constante baixa δ, valor intermediário 10^{-7}, para estabilidade numérica;
Inicializar variáveis de acumulação $r = 0$;
while *critério de parada não atendido* **do**
  Experimente um minilote de exemplos do conjunto de treinamento $\{x^{(1)},...,x^{(m)}\}$ com alvos correspondentes $y^{(i)}$;
  Gradiente:$g \leftarrow \frac{1}{n} \bigtriangledown \Theta \sum_i L(f(x^{(i)};\Theta), y^{(i)})$;
  Quadrado acumulado do gradiente: $r \leftarrow r + g \bigodot g$;
  Atualização: $\triangle\Theta \leftarrow -\frac{\varepsilon}{\delta+\sqrt{r}} \bigodot g$. (Divisão e raiz quadrada aplicada a elementos);
  Aplicar atualização: $\Theta \leftarrow \Theta + \triangle\Theta$;
end

Nesse contexto, o algoritmo RMSProp modifica o AdaGrad para ter um desempenho melhor na configuração não-convexa, alterando a acumulação de gradiente em uma média móvel ponderada exponencialmente [123, 194]. Isso é projetado para convergir rapidamente quando aplicado a uma função convexa.

Dessa forma, no RMSProp cada atualização é realizada separadamente para cada parâmetro (w^j), tendo as suas expressões matemáticas definidas por:

$$\mathbf{v_t^w} = \beta * v_{t-1}^w + (1-\beta)(\bigtriangledown w_t)^2 \tag{3-31}$$

$$\mathbf{w_{t+1}} = w_t - \frac{\eta}{v_t^w + \varepsilon} * \bigtriangledown w_t \tag{3-32}$$

$$\mathbf{v_t^b} = \beta * v_{t-1}^b + (1-\beta)(\bigtriangledown b_t)^2 \tag{3-33}$$

$$\mathbf{b_{t+1}} = b_t - \frac{\eta}{\sqrt{v_t^b + \varepsilon}} * \bigtriangledown b_t \tag{3-34}$$

onde η é a taxa inicial de aprendizado, v_t é a média exponencial dos quadrados do gradiente e $\bigtriangledown w_t$ representa o valor do gradiente no instante t do peso w^j.

Algoritmo 3.2: Pseudocódigo do algoritmo RMSProp

Requerer: Taxa de aprendizagem global ε, taxa de decaimento ρ;
Requerer: Inicialização de parâmetro Θ;
Requerer: Constante baixa δ, valor intermediário 10^{-6}, usado para estabilizar a divisão por pequenos números;
Inicializar variáveis de acumulação $r = 0$;
while *critério de parada não atendido* **do**
  Experimente um minilote de exemplos do conjunto de treinamento $\{x^{(1)},...,x^{(m)}\}$ com alvos correspondentes $y^{(i)}$;
  Gradiente:$g \leftarrow \frac{1}{n} \bigtriangledown \Theta \sum_i L(f(x^{(i)};\Theta), y^{(i)})$;
  Quadrado acumulado do gradiente: $r \leftarrow \rho r + (1-\rho) g \bigodot g$;
  Atualização: $\triangle\Theta \leftarrow -\frac{\varepsilon}{\sqrt{\delta+r}} \bigodot g$($\frac{1}{\sqrt{\delta+r}}$ aplicado a elementos);
  Aplicar atualização: $\Theta \leftarrow \Theta + \triangle\Theta$;
end

Esses algoritmos são muito utilizados e variações deles são desenvolvidas com frequência. Dessa forma, Mukkamala e Hein [123] realizam uma variação, que utiliza funções logarítmicas dos algoritmos. Tais variações mostraram ser especialmente eficientes em algumas situações, como, por exemplo, em espaços muito convexos.

Não obstante, o grampo do gradiente é uma técnica que define parâmetros capazes de limitar o gradiente. Esta limitação pode ser feita para auxiliar a fase de treinamento do modelo. Aplicando essa técnica, também é possível evitar problemas com o *overfitting* [144]. Assim, seja g o gradiente de $\frac{\partial E}{\partial w}$, se $||g|| \geq$ que o limite, então g pode ser definido como:

$$\mathbf{g} \leftarrow \frac{limite}{||g||} g \tag{3-35}$$

3.2.5 Arquitetura Inception

A arquitetura profunda Inception foi introduzida em 2014, por grupos de pesquisas da Google, em colaboração com várias universidades. Inicialmente denominada

GoogLeNet[3], a sua primeira versão (Inception-V1) foi a vencedora do desafio de classificação de imagens (ILSVRC 2014).Essa arquitetura propiciou uma diminuição significativa na taxa de erros quando comparadas com as demais arquiteturas, como, por exemplo, a arquitetura AlexNet e a ZF-Net, ambas vencedoras dos desafios de 2012 e 2013, respectivamente [163].

Na sua primeira versão, o Inception-V1 conta com módulos em que a entrada é processada por vários módulos paralelos de convoluções, mesclando as saídas em um único tensor [88], utilizando-se de técnicas convolucionais $1x1$ no meio da arquitetura e do *pool* médio global [163]. Posteriormente, o Inception-V2 introduziu a normalização de lotes e o Inception-V3 inseriu ideias adicionais de fatoração [162].

A arquitetura Inception se baseia em descobrir como uma estrutura local, idealmente esparsa de uma rede convolucional, pode ser coberta ou aproximada por meio de componentes densos, prontamente disponíveis (blocos convolucionais).

Dessa forma, um dos principais objetivos é identificar a estrutura local idealmente esparsa e repeti-la espacialmente [106, 164]. Para tal, é necessário construir camada por camada, analisando as estatísticas das últimas. Agrupá-las em unidades de correlação é uma possível solução.

Entretanto, com a aplicação dessa técnica pode haver um número menor de aglomerações localizadas espacialmente espalhadas, ocasionando problemas, com um número decrescente de caminhos em regiões maiores. Para evitar problema como esse, as versões do Inception estão restritas aos tamanhos de filtros convolucionais de $1x1$, $3x3$ e $5x5$. Assim, a proposta do Inception é a combinação das camadas convolucionais com todos os filtros, concatenados em um único vetor de saída.

Além da aplicação e junção dos filtros convolucionais concatenados, o Inception aplica operações de *pool* máximo paralelo alternativo em cada estágio (filtros $3x3$), trazendo um efeito benéfico para a estrutura convolucional (Figura 3.20).

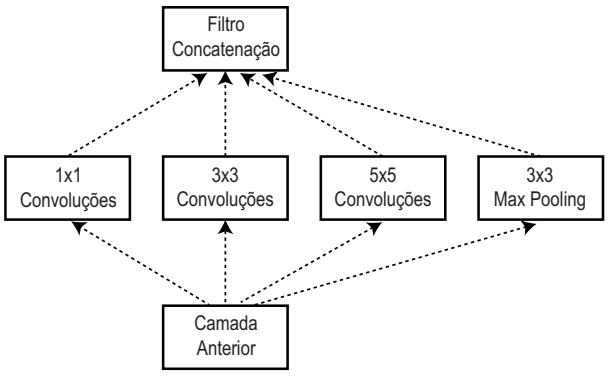


Figura 3.20: *Exemplo do módulo Inception (Adaptada de [161]).*

[3]Esse nome é uma homenagem a Yann LeCuns, pioneiro da rede LeNet 5 [98]

Observa-se que, por causa do seu empilhamento, as saídas com as estatísticas de correlação no módulo Inception tendem a variar. Isso ocorre porque, de acordo com que os recursos da camada de abstração mais alta são capturados e a concentração espacial tende a diminuir, podendo haver um aumento nas proporções de 3x3 e nas camadas de convoluções, à medida em que se avança para as camadas mais altas dessa estrutura.

Um dos problemas apresentados por essa estrutura está relacionado ao alto custo computacional, mesmo em estruturas com camadas de convoluções modestas, gerando um número expressivamente grande de filtros convolucionais. Esse problema é ainda mais evidenciado quando se inserem operações de *pooling* paralelo alternativo, fazendo com que o número de filtros de saída seja igual ao número de filtros anteriores. Essa junção das camadas convolucionais e das operações de *pooling* levará a um aumento inevitável de saídas em cada estágio. Assim, sob esses aspectos, mesmo que essa estrutura descubra a estrutura local idealmente esparsa, mostra-se ainda ineficiente, elevando a uma explosão computacional entre os estágios.

Aplicar reduções e projeções de dimensões nos locais em que requisitos computacionais aumentam exponencialmente, pode ser uma solução viável para esse problema. Essas projeções de dimensões ocorrem pelas aplicações de convoluções 1x1, calculando as reduções antes das onerosas convoluções 3x3 e 5x5, respectivamente (Figura 3.21).

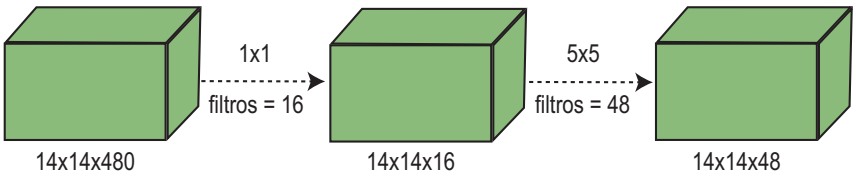


Figura 3.21: *Exemplo da redução das dimensões das camadas do Inception-V1.*

Além das aplicações das convoluções como redutores, essa estrutura utiliza a função de ativação linear retificadora, obtendo uma finalidade dupla (Figura 3.22).

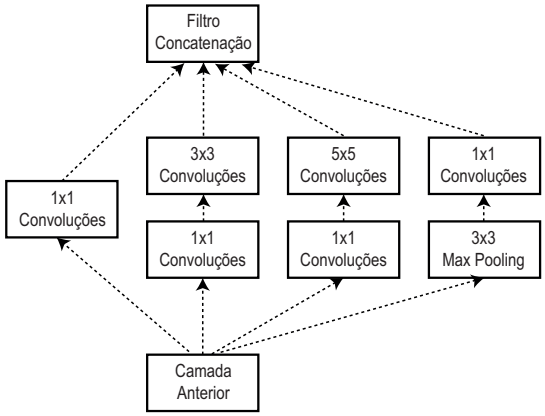


Figura 3.22: *Exemplo do módulo Inception com redução de dimensão (Adaptado de [161]).*

De forma generalizada, uma rede Inception consiste em um conjunto de módulos empilhados uns sobre os outros, adicionadas a esse empilhamento mais camadas de *pool* máximo, reduzindo a resolução da grade computacional (9 módulos).

A estrutura completa da rede Inception-V1 possui 27 camadas, incluindo as camadas profundas e as camadas de operações de *pooling*. O número geral de blocos de construções independentes utilizado para construção da rede pode se aproximar de 100 blocos. Além da camada de ativação linear, a estrutura usa, ainda, um pool médio, antes de cada classificador (Tabela 3.1).

Tabela 3.1: *Configurações da arquitetura Inception-V1 (Adaptado de [163]).*

tipo	caminho/passo	saída	profundidade	pool	params	ops
convolution	7x7=2	112x112x64	1		2.7K	34M
max pool	3x3=2	56x56x64	0			
convolution	3x3=1	56x56x192	2		112K	360M
max pool	3x3=2	28x28x192	0			
inception (3a)		28x28x256	2	32	159K	128M
inception (3b)		28x28x480	2	64	380K	304M
max pool	3x3=2	14x14x480	0			
inception (4a)		14x14x512	2	64	364K	73M
inception (4b)		14x14x512	2	64	437K	88M
inception (4c)		14x14x512	2	64	463K	100M
inception (4d)		14x14x528	2	64	580K	119M
inception (4e)		14x14x832	2	128	840K	170M
max pool	3x3=2	7x7x832	0			
inception (5a)		7x7x832	2	128	1072K	54M
inception (5b)		7x7x1024	2	128	1388K	71M
avg pool	7x7=1	1x1x1024	0			
dropout (40%)		1x1x1024	0			
linear		1x1x1000	1		1000K	1M
softmax		1x1x1000	0			

Uma das vantagens dessa arquitetura é o aumento significativo da inserção de unidades a cada estágio, sem a complexidade computacional. Sendo assim, a utilização da redução de dimensão permite realizar a proteção dos números de filtros de entrada entre os estágios, em cada camada. Esse fato faz com que se reduza a sua própria dimensão, antes de aplicar uma convolução.

Portanto, a utilização dos recursos computacionais, de forma aprimorada, permite um aumento, tanto na largura de cada estágio, quanto no número de estágios. Uma das possibilidades da utilização dessa arquitetura é a de se criar versões inferiores, mas computacionalmente mais econômicas.

Uma das características do Inception-V1 decorre da redução de dimensões. Esse fator pode ser definido pelo uso de fatorações de convoluções, de forma computacional-

mente eficiente. Um exemplo disso é uma camada convolucional 1x1, seguida de outra camada convolucional 3x3.

Em uma rede padrão, espera-se que os resultados das ativações sejam significativamente próximos e correlacionados. Assim, almeja-se que as ativações possam ser reduzidas antes mesmo das suas agregações, resultando em representações locais igualmente expressivas.

No Inception-V2 são realizadas propostas de fatorações das convoluções em várias configurações, com o intuito de aumentar a eficiência computacional da solução. Como as próprias redes Inceptions são totalmente convolucionais, cada peso corresponde a uma multiplicação a cada ativação. Dessa forma, qualquer redução do custo computacional está intrinsecamente ligada a um número reduzido de parâmetros, fazendo com que o treinamento dessas redes seja mais rápido, aumentando o tamanho dos filtros.

Pode-se, também, realizar fatorações em convoluções menores. Esse fato é importante nessa estrutura, porque em convoluções com filtros espaciais maiores, como, por exemplo, nas convoluções 5x5 ou 7x7, há uma tendência a ser exponencialmente caras, isso é, ter um custo computacional mais alto. Em outras palavras, uma convolução 3x3 possui um custo computacional menor quando se compara a convolução 5x5 ou 7x7, com o mesmo número de filtros. A ideia principal do Inception-V2 é substituir uma convolução 5x5 por uma rede de várias camadas com menos parâmetros, mantendo-se o tamanho tanto da entrada quanto da saída.

Resumidamente, o Inception-V2 permite realizar a fatoração/substituição da camada convolucional 5x5 por duas camadas convolucionais 3x3, melhorando o custo e o desempenho computacional (Figura 3.23).

Os resultados das aplicações da arquitetura Inception-V2 sugerem que convoluções maiores que 3x3 não são úteis, pois pode ser sempre fatorada em tamanhos menores de filtros.

Outra proposta dessa arquitetura é que se pode diminuir ainda mais o tamanho dos filtros, isso é, as convoluções de 3x3 podem ser fatoradas por convoluções de 2x2. Entretanto, o Inception-V2 aplica uma técnica de fatoração, que melhora ainda mais os seus resultados por convoluções assimétricas, isso é, utilizando uma convolução de 3x1 seguida de outra convolução de 1x3.

Portanto, a redução de uma camada convolucional de 3x3 por duas camadas de 3x1 e 1x3 é 33% mais barata para o mesmo número de filtros, tanto de entrada quanto de saída (Figura 3.24).

Entretanto, de acordo com Ketkar e Santana [88], na prática, o emprego dessa fatoração possui resultados significativos apenas nas camadas intermediárias ou nas últimas camadas, ou seja, nas camadas iniciais não se tem resultados satisfatórios. Dessa forma, resultados muitos bons podem ser alcançados, utilizando convoluções 1x7,

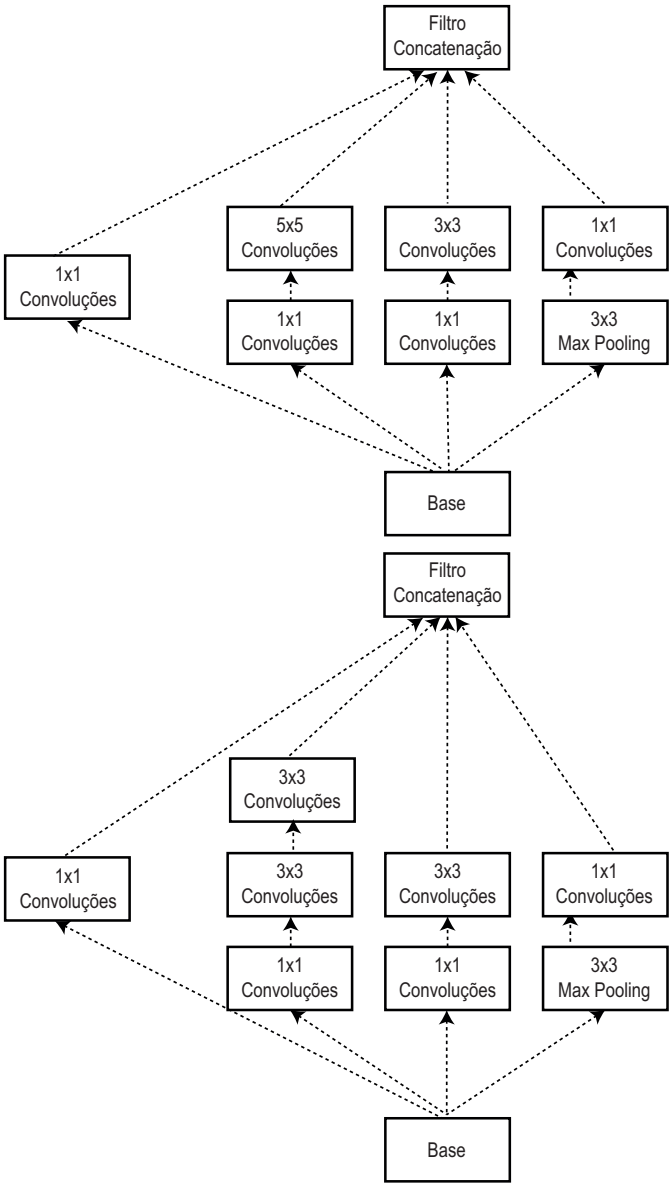


Figura 3.23: *Exemplo da estrutura Inception substituindo uma convolução 5x5 por duas convoluções 3x3 (Adaptado de [161]).*

seguidas de convoluções $7x1$ (Figura 3.25).

Nesse sentido, a proposta final do Inception-V2 fatora a tradicional convolução $7x7$ em três camadas convolucionais de $3x3$. A parte inicial da rede possui três módulos Inception tradicional de $35x35$ com 288 filtros cada. Esses filtros são reduzidos para uma grade de $17x17$, com 768 filtros, reduzindo, ainda, para $8x8x1280$. A estrutura conta com dois módulos Inception, com tamanho de filtros de saídas concatenados de 2048.

A arquitetura Inception-V3 possui características similares à arquitetura anterior (Figura 3.26).

Na arquitetura V2 se inserem classificadores auxiliares, com o intuito de combater os problemas com o gradiente explodindo, promovendo um aprendizado mais es-

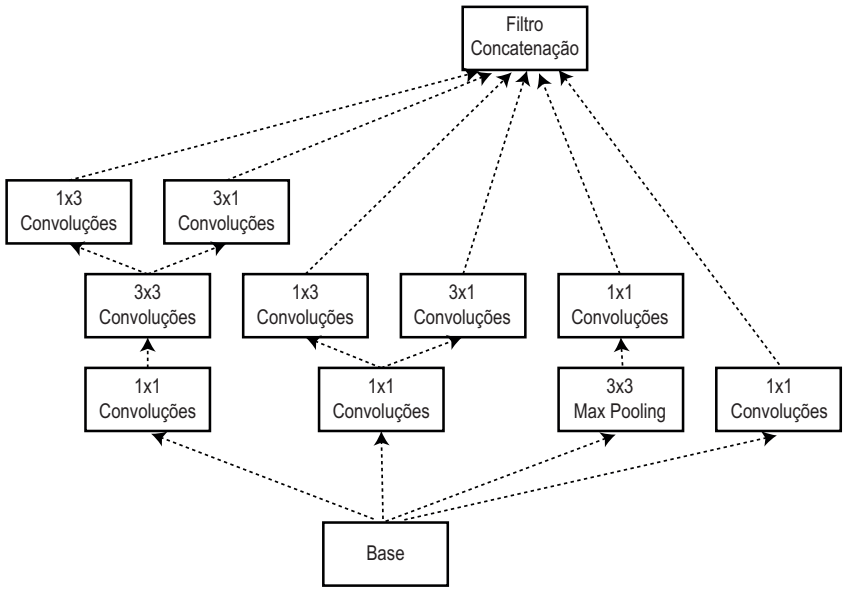


Figura 3.24: *Exemplo da estrutura Inception substituindo uma convolução* 5*x*5 *por duas convoluções* 3*x*3*, representada por convoluções* 1*x*3 *e* 3*x*1.

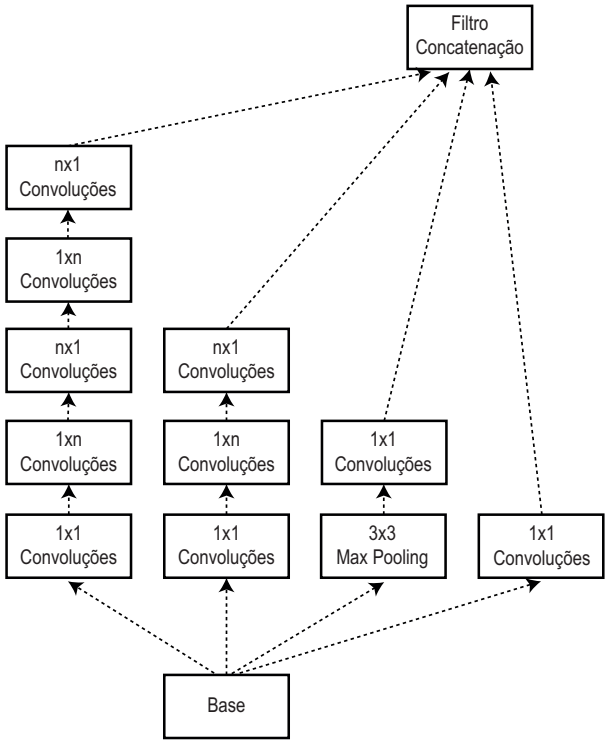


Figura 3.25: *Exemplo de módulos de criação após a fatoração das nxn convoluções (Adaptado de [161]).*

tável, com resultados convergentes. No entanto, apesar da introdução de classificadores auxiliares na arquitetura anterior, com a finalidade de melhorar o desempenho das redes profundas, especificamente, na boa convergência dos resultados, os classificadores não contribuem de forma significativa quando as precisões estão próximas da saturação na

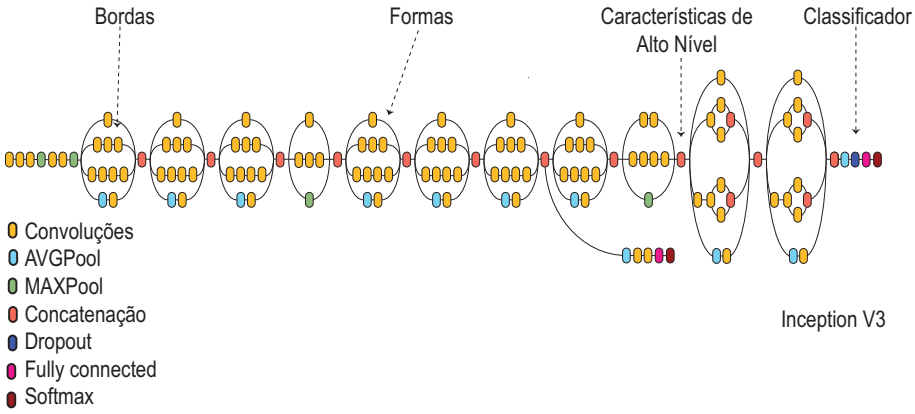


Figura 3.26: *Exemplo da arquitetura Inception-V3.*

fase de treinamento do modelo.

De acordo com Szegedy et al. [164], os classificadores auxiliares possuem um papel de regularizadores, obtendo um bom desempenho se as ramificações laterais da rede Inception forem normalizadas em lote (*batch normalization*). Dessa forma, a arquitetura do Inception-V3 propõe a aplicação da normalização em lotes nos classificadores auxiliares entre as ramificações laterais.

Observa-se que, tradicionalmente, as redes Inceptions anteriores aplicam operações de *pooling* para diminuir o tamanho do mapa de características, evitando gargalos representacionais. Nesse sentido, o Inception-V3 propõe uma variante, reduzindo ainda mais o custo computacional, removendo gargalos representacionais.

Além dessas características, a nova arquitetura Inception aplica otimizadores estocásticos, por exemplo, o RMSProp e a aplicação de suavização de rótulos. A suavização de etiquetas se caracteriza como um componente regularizador, que possui o objetivo de impedir que a rede seja excessivamente ajustada, aplicando uma taxa de *dropout* satisfatória para o aprendizado do modelo.

De acordo com Szegedy et al. [162], os modelos Inceptions anteriores são treinados de forma particionada. Assim, cada rede é dividida em várias sub-redes, ajustando integralmente o modelo. Porém, as arquiteturas baseadas em Inception são bastante ajustáveis, fazendo com que ocorram diversas alterações no número de filtros em cada camada do modelo. Algumas dessas alterações não afetam diretamente a qualidade da rede treinada.

Assim, para a otimização dessa rede, as arquiteturas anteriores ajustavam os tamanhos das camadas, equilibrando os cálculos entre as sub-redes dos modelos. No entanto, modelos mais recentes podem realizar os seus treinamentos, sem a necessidade de particionar as suas réplicas. Isso ocorre devido ao acesso exponencial às memorias de retropropagação, selecionando tensores necessários para o cálculo do gradiente, reduzindo o número de tensores.

Nesse sentido, o Inception-V4 propõe retirar complexidades extras das arquiteturas anteriores, realizando escolhas uniformes para os blocos em cada tamanho do mapa de características. A configuração completa da arquitetura Inception-V4 é apresentada na Figura 3.27, contendo o esquema geral do modelo.

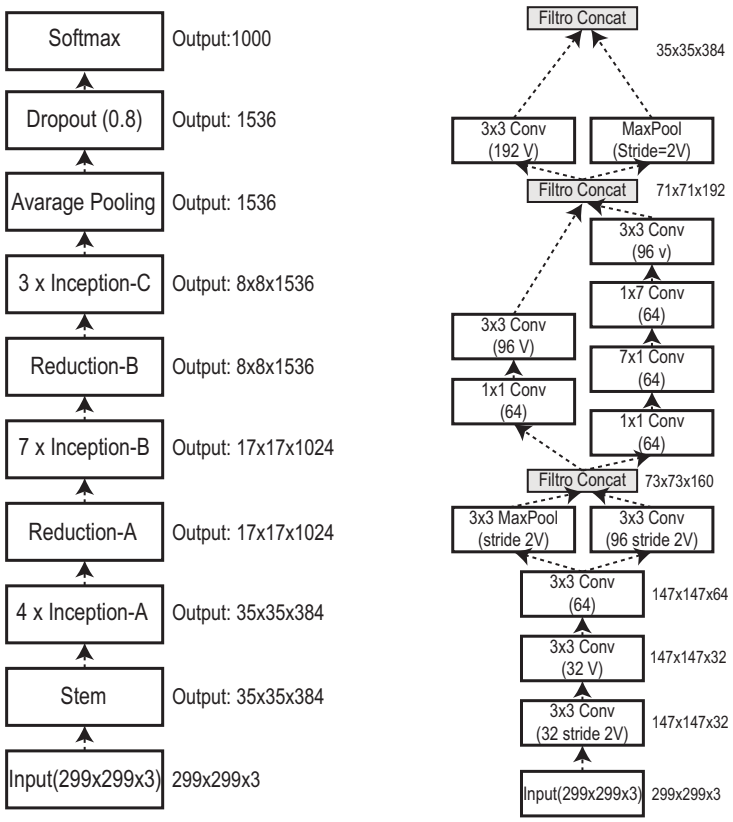


Figura 3.27: *Exemplo detalhado do Inception-V4 (Adaptado de [161]).*

Observa-se que houve uma mudança no tronco da arquitetura Inception. Na versão anterior, as arquiteturas possuíam apenas três módulos Inceptions (A, B e C). Nessa nova versão, além da descrição desses módulos, foram inseridos módulos de redução de dimensionalidades entre os módulos intermediários, alterando tanto a largura quanto a altura do mapa de características do modelo.

3.2.6 Arquitetura VGG

A arquitetura do Grupo de Geometria Visual (do inglês *Visual Geometry Group* - VGG) foi desenvolvida pelo grupo do departamento de ciência e engenharia da Universidade de Oxford. A arquitetura VGG se caracteriza pelas propostas de melhorias causadas pelo aumento da profundidade nas redes neurais convolucionais, utilizando os mesmos princípios das redes neurais AlexNet [152]. Nesse modelo são propostas quatro arquite-

turas, a VGG11 (8 conv e 3 FC), a VGG13, a VGG16 e a VGG19 (16 conv e 3 FC), essas duas últimas são as mais utilizadas atualmente (Figura 3.28).

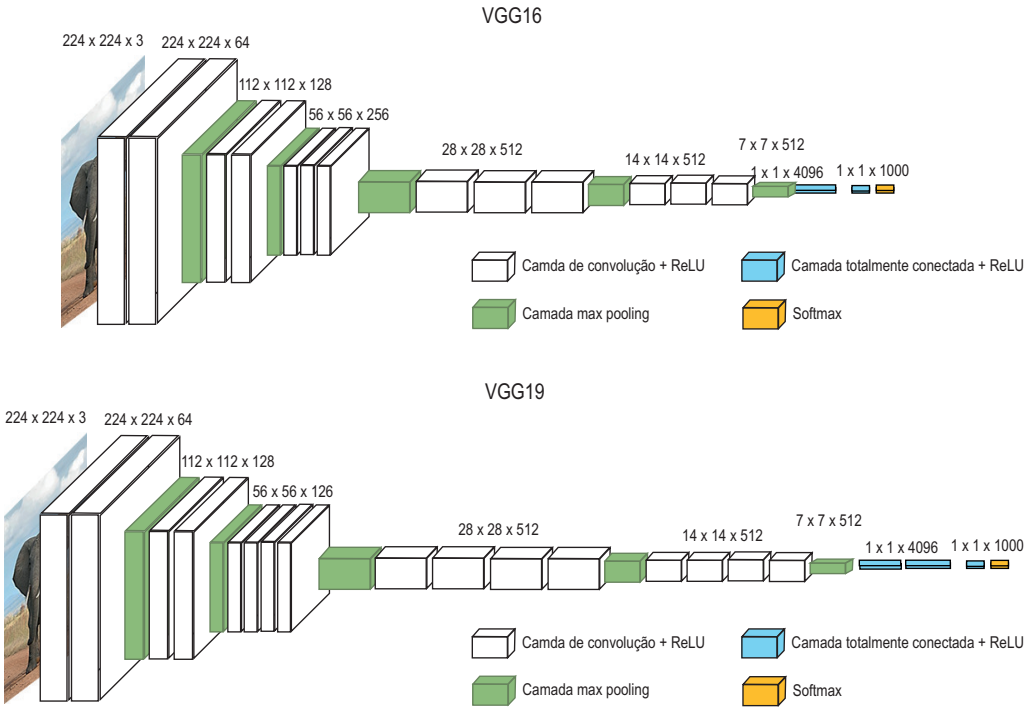


Figura 3.28: *Exemplo do esquema geral das arquieturas VGG16 e VGG19.*

Inicialmente, a configuração padrão da VGG conta com uma entrada com canais RGB de tamanho fixo 244x244, substituindo apenas o valor médio RGB de cada *pixel* no conjunto de treinamento. Dessa forma, a imagem de entrada pode ser passada pela pilha de camadas convolucionais (conv), utilizando-se de filtros pequenos, por exemplo, 1x1 ou 3x3.

Uma das características principais dessa arquitetura é que a resolução espacial da camada de entrada é preservada, mesmo após as realizações das convoluções [152]. Esse agrupamento espacial pode ser realizado aplicando cinco camadas de *pooling* máximo, seguidas de algumas camadas convolucionais. No entanto, nem todas as camadas convolucionais são seguidas de camadas de *pooling* máximo. Assim, a camada de *pooling* máximo pode ser realizada pela aplicação de janelas de tamanho 2x2 com 2 passos. Para cada pilha de camadas convolucionais são adicionadas mais três camadas totalmente conectadas (FC). As duas primeiras FC possuem 4096 canais e a terceira com 1000 canais (um canal para cada classe). As configurações das FCs são padronizadas em todas as camadas. A última camada é composta por uma função de ativação (*softmax*).

Umas das vantagens nessa arquitetura é a utilização da função de ativação não retificadora. Nesse sentido, em todas as camadas ocultas nessa arquitetura são adicionadas funções de ativação ReLU. Outra observação é a não utilização de normalização local.

Essa normalização, além de não melhorar o desempenho da rede em alguns conjuntos de dados, por exemplo, no conjunto ILSVRC, ainda leva a um aumento no consumo de memória, elevando o tempo computacional da sua execução.

Tabela 3.2: *Configurações das arquiteturas VGG11, VGG13, VGG16 e VGG19 [152].*

Configuração ConvNet					
A	**A-LRN**	**B**	**C**	**D**	**E**
11 camadas	**11 camadas**	**13 camadas**	**16 camadas**	**16 camadas**	**19 camadas**
Entrada (224 x 224 RGB image)					
conv3-64	conv3-64	conv3-64	conv3-64	conv3-64	conv3-64
	LRN	conv3-64	conv3-64	conv3-64	conv3-64
maxpool					
conv3-128	conv3-128	conv3-128	conv3-128	conv3-128	conv3-128
		conv3-128	conv3-128	conv3-128	conv3-128
maxpool					
conv3-256	conv3-256	conv3-256	conv3-256	conv3-256	conv3-256
conv3-256	conv3-256	conv3-256	conv3-256	conv3-256	conv3-256
			conv1-256	conv3-256	conv3-256
					conv3-256
maxpool					
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
			conv3-512	conv3-512	conv3-512
					conv3-512
maxpool					
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
			conv3-512	conv3-512	conv3-512
					conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Outra característica importante está relacionada à largura das camadas convolucionais. Essas, por sua vez, são extremamente pequenas, iniciando-se com um tamanho de 64 até chegar em 512. A tabela 3.2 apresenta um resumo com as configurações de cada versão da arquitetura VGG.

Observa-se que, nessa arquitetura, aplica-se campos receptivos[4] relativamente pequenos nas primeiras camadas convolucionais, ou seja, a proposta aplica campos receptivos de tamanho $3x3$. Para os autores, é perceptível que uma pilha com duas

[4]Tamanho dos blocos de *pixels*.

camadas convolucionais de tamanho 3x3 possa ser equivalente a um campo receptível de tamanho 5x5. Consequentemente, três camadas convolucionais do mesmo tamanho (3x3) são equivalentes a um campo receptível de tamanho 7x7.

No entanto, o ganho com a aplicação da arquitetura VGG está relacionado à incorporação de três camadas retificadoras não lineares, diferente das propostas anteriores, que incorporavam apenas uma camada não linear. Outra vantagem é a diminuição de parâmetros e a incorporação convolucional de tamanho 1x1. Essa incorporação é uma maneira de aumentar ainda mais a não linearidade, sem afetar os campos receptíveis das camadas convolucionais, adicionada pela função retificadora.

3.3 Rede Neural Recorrente

A Rede Neural Recorrente (RNN) se caracteriza por receber uma série de entradas sem limite predeterminado de tamanho [19]. A RNN pode receber um ou mais vetores de entrada e produzir um ou mais vetores de saída. A saída é influenciada não apenas pelos pesos aplicados nas entradas, mas, também, por um vetor de estado oculto, que representa o contexto com base nas informações anteriores. Assim, a mesma entrada é capaz de produzir uma saída diferente, dependendo das entradas anteriores da série [181] (Figura 3.29).

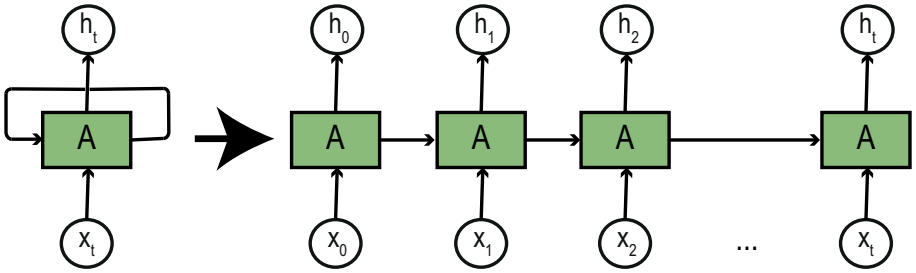


Figura 3.29: *Exemplo da rede neural recorrente desenrolada.*

A RNN aprende por exemplos anteriores, tomando decisões baseados nesse aprendizado [22]. Para a realização dessa tarefa, a RNN possui um processamento temporal, com a capacidade de implementar memórias adaptativas, extraindo informações usadas anteriormente e prevendo saídas posteriores [156].

A RNN é chamada de recorrente, porque executa a mesma tarefa para cada elemento de uma sequência, com a saída, sendo dependente dos cálculos anteriores. Para tal, a RNN possui uma memória que captura informações sobre o que foi calculado até o momento, podendo usar essas informações em sequências arbitrariamente longas [19].

As RNNs, geralmente, são projetadas em uma estrutura semelhante a uma cadeia, retornando às camadas anteriores. Na RNN padrão, esse módulo de *loop* possui uma estrutura muito simples. Essa estrutura pode ser uma camada simples, que controla

todo o fluxo [68]. Para a realização de melhorias nessas estruturas, as RNNs podem ser classificadas em Memória Longa de Curto Prazo (LSTM) e Unidades Recorrentes Bloqueadas (GRU).

3.3.1 Memória Longa de Curto Prazo (LSTM)

A LSTM é um tipo especial de RNN capaz de aprender dependências em longo prazo [30]. Inicialmente, a LSTM foi proposta por Hochreiter e Schmidhuber [71] e, desde então, sofreu várias modificações. Esse tipo de unidade recorrente trabalha significativamente bem, em uma variedade de problemas, justificando a sua utilização e popularização [30].

Dessa forma, a LSTM foi projetada exclusivamente para evitar problemas de dependência em longo prazo. Diferente da RNN padrão, a qual possui um módulo de repetição, com uma única camada de função de tangente hiperbólica (*tanh*), a LSTM possui quatro camadas interagindo de maneira integrada (Figura 3.30).

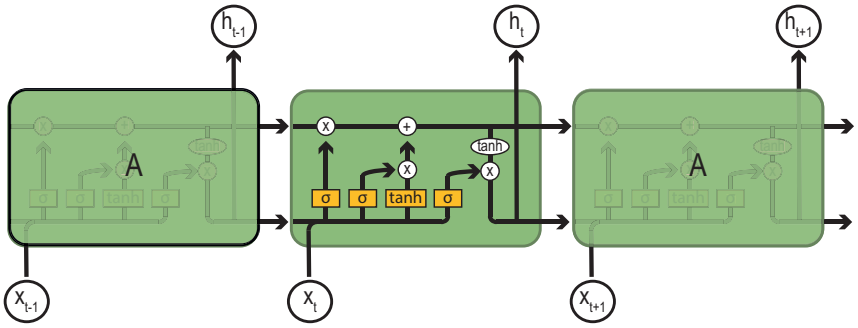


Figura 3.30: *Exemplo da arquitetura da Memória Longa de Curto Prazo (LSTM).*

Nesse sentido, a LSTM possui a capacidade de remover ou adicionar informações no estado da célula, regulada por estruturas denominadas de portas [132]. Essas portas podem aprender quais dados em uma sequência são importantes para manter ou ser descartados.

Neste sentido, a LSTM possui a capacidade de remover ou adicionar informações no estado da célula, regulada por estruturas denominadas de portas [132]. Essas portas podem aprender quais dados em uma sequência são importantes para manter ou ser descartados.

Ao fazer isso, eles podem transmitir informações relevantes ao longo da cadeia de sequências, realizando previsões. Para cada porta há uma rede neural *sigmoide* (σ) e uma operação de multiplicação ($\odot$). Assim, a camada *sigmoide* é responsável por gerar números de zero a um, descrevendo a quantidade de componentes que devem ser liberados. Ao controlar essas portas, os erros podem ser retropropagados por qualquer quanti-

dade de camadas. Esse mecanismo permite que a rede aprenda tarefas que dependem de eventos que ocorreram diversas vezes [69].

Diferentemente das unidades recorrentes simples que realizam a soma ponderada dos sinais de entradas, aplicando a função linear, cada i-ésimo da unidade LSTM mantém uma memória C_t no tempo t, gerando uma saída h_t ou uma função de ativação (Figura 3.31).

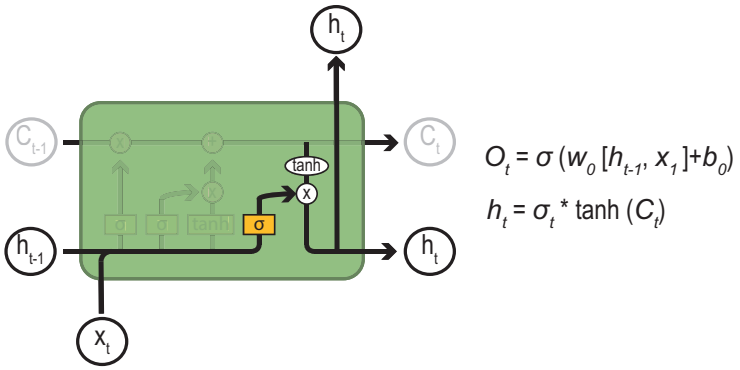


Figura 3.31: *Exemplo da geração da saída ou da função de ativação na Memória Longa de Curto Prazo (LSTM).*

onde o_t representa uma porta de saída, que visa modular a quantidade de exposição do conteúdo da memória. Desse modo, a porta de saída é calculada da seguinte forma:

$$\mathbf{o_t^j} = \sigma(W_0 x_t + U_0 h_{t-1} + V_0 c_t)^j \tag{3-36}$$

onde σ é uma função *sigmoide* logística e V_0 é uma matriz diagonal. A célula de memória c_t^j é atualizada esquecendo, parcialmente, a memória existente e adicionando um novo conteúdo na memória c_t^j:

$$\mathbf{c_t^j} = f_t^j c_{t-1}^j + i_t^j \tilde{c}_t^j \tag{3-37}$$

Em que o novo conteúdo na memória é:

$$\mathbf{\tilde{c}_t^j} = tanh(W_c x_t + U_c H_{t-1})^j \tag{3-38}$$

A modulação da memória existente é esquecida pela porta f_t^j, e o grau a qual o novo conteúdo é adicionado na célula de memória é definido pela a porta i_t^j. Assim, as portas são calculadas por:

$$\mathbf{f_t^j} = \sigma(W_f x_t + U_f h_{t-1} + V_f c_{t-1})^j \tag{3-39}$$

$$\mathbf{i_t^j} = \sigma(W_i x_t + U_i h_{t-1} + V_i c_{t-1})^j \tag{3-40}$$

Portanto, diferentemente das unidades recorrentes padrões, as quais sobrescrevem o seu conteúdo a cada etapa de tempo, a LSTM decide se deseja manter as informações na memória existentes, pelas suas portas. Intuitivamente, se a LSTM detecta um novo recurso importante em uma sequência de entradas no estágio inicial, ela pode carregar facilmente essas informações por um longo período, capturando, assim, possíveis dependências dos recursos, com longas distâncias nas informações.

3.3.2 Unidade Recorrente Bloqueada (GRU)

A GRU foi introduzida por Cho et al. [29], simplificando, de maneira eficiente, a LSTM (Figure 3.32). Nesse sentido, a GRU captura adaptativamente as dependências em diferentes escalas de tempo, baseando-se apenas em duas portas, atualizando e redefinindo o seu conteúdo na unidade de memória recorrente [143, 30, 31].

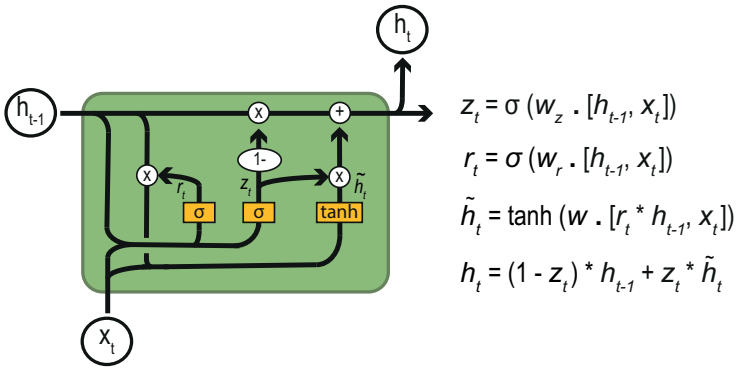


Figura 3.32: *Exemplo da arquitetura da Unidade Recorrente Bloqueada (GRU).*

O principal objeto da GRU, além de manter as informações de recurso em um longo período, é evitar o gradiente desvanecendo [105]. De maneira similar a LSTM, a GRU possui unidades de conexão capazes de modular as informações, sem necessariamente realizar a separação das células da memória. Dessa forma, cada unidade possui, então, uma porta de atualização (z_t) e uma de reset (r_t), expondo o seu conteúdo de memória, realizando o equilíbrio entre o conteúdo atual e o anterior, com o tempo de adaptação controlado pela porta de atualização. Assim, a GRU pode ser definida pelas seguintes equações:

$$\mathbf{z_t} = \sigma(X_t W_z + U_z H_{t-1} + b_z) \tag{3-41}$$

$$\mathbf{r_t} = \sigma(X_t W_t + U_r H_{t-1} + b_r) \tag{3-42}$$

$$\tilde{\mathbf{h}}_\mathbf{t} = \varphi[X_t W_h + (R_t \bigodot H_{t-1}) U_h + b_h] \tag{3-43}$$

$$\mathbf{h_t} = (1 - Z_t) \bigodot H_{t-1} + Z_t \bigodot \tilde{H}_t \tag{3-44}$$

Além das portas z_t e r_t, a estrutura conta com a ativação candidata ($\tilde{h}_t$) e o bloco de saída (h_t), para o período t. Nessa estrutura, x_t representa as instâncias, isso é, o conjunto contendo as frases e imagens de treinamento. O w e u são os pesos representantes de x_t e x_{t-1}, respectivamente, enquanto b representa as bases.

Não obstante, as multiplicações entre os elementos são representadas por $\bigodot$ e as funções de ativação, de ambos os portões são representadas por *sigmoides* logísticas $\sigma(\cdot)$, obrigando r_t e z_t a assumirem valores que variam de 0 a 1.

3.3.3 Problemas com Gradiente Desvanecendo ou Explodindo

Devido ao problema do gradiente desvanecendo as RNNs são difíceis de serem treinadas. Esse problema se refere à dificuldade que existe no treinamento de uma RNN por métodos como a descida de gradiente e retropropagação.

Nesse sentido, o treinamento dessas redes neurais recorrentes se utilizam de algoritmos de retropropagação [70]. Nesse tipo de algoritmo, o erro geral do gradiente é igual à soma dos erros individuais, em cada etapa. Essa aplicação é conhecida como algoritmo de retropropagação pelo tempo (do inglês *backpropagation through time* - BPTT). Assim, o erro no tempo T pode ser definido pela seguinte expressão:

$$\frac{\partial \mathbf{E}}{\partial \mathbf{w}} = \sum_{t=1}^{T} \frac{\partial E_t}{\partial w} \tag{3-45}$$

Também, pode-se aplicar a regra das cadeias para calcular o erro geral do gradiente:

$$\frac{\partial \mathbf{E}}{\partial \mathbf{w}} = \sum_{t=1}^{T} \frac{\partial E}{\partial y_t} \frac{\partial y_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial w} \tag{3-46}$$

onde o termo $\frac{\partial h_t}{\partial h_k}$ é definido pela derivada da relação entre o estado oculto k e t, sendo um produto das matrizes jacobianas. Dessa forma, a medida em que a sequência fica mais longa, ou seja, a distância entre t e k aumentam, o valor de γ determinará se o gradiente ficará pequeno ou grande, em outras palavras, desvanecerá ou explodirá.

$$\left\| \frac{\partial \mathbf{h_t}}{\partial \mathbf{h_k}} \right\| = \left\| \prod_{i=k+1}^{t} \frac{\partial h_i}{\partial h_{i-1}} \right\| \leq (\gamma w \gamma h)^{t-k} \tag{3-47}$$

O gradiente desvanecendo (γw e $\gamma h < 1$) ocorre devido à natureza recursiva das RNNs. Essa falha pode ser resolvida aplicando a arquitetura da LSTM/GRU ou pela alteração da função de ativação.

Não obstante, nos modelos de aprendizado de máquina, o objetivo é encontrar os parâmetros que minimizam uma função de custo. Nas redes neurais, é utilizado o algoritmo da descida de gradiente, que executa iterações nos parâmetros proporcionais ao valor do gradiente. Em uma rede neural com muitas camadas, os pesos são calculados, de acordo com a regra das cadeias, quando se utiliza o método retropropagação. Normalmente, os gradientes utilizam valores que vão de -1 a 1 (função hiperbólica). Essa regra multiplica o valor dos erros ao longo das camadas, o que faz com que ele diminua exponencialmente. Como consequência, a rede neural é treinada muito lentamente.

Para solucionar esse problema, pode-se alterar as funções de ativação *sigmoide* e tangente hiperbólica pelas funções de ativações que gerem valores mais altos, como, por exemplo, a função de ativação ReLU [77] proposta por Vinod e Geoffrey [125].

O gradiente explodindo (γw e $\gamma h > 1$) se refere ao aumento exponencial dele, que também inviabiliza o treinamento da rede neural. Esse problema pode ser resolvido de diversas formas. Uma delas, a mais simples, é estabelecendo um limiar máximo para os gradientes. Caso esse valor ultrapasse o limiar, o algoritmo irá defini-lo como o valor do gradiente [135].

Em um trabalho mais recente foi ilustrado como resolver o problema do gradiente explodindo, utilizando-se a GRU [85]. A GRU se destaca por utilizar menos parâmetros, sendo uma solução para ambos os problemas, tanto o gradiente explodindo quanto o gradiente desvanecendo.

Outra possível solução para os problemas da explosão do gradiente ou gradiente desvanecendo é manter-se os valores dos gradientes próximos de 1, por um processo de regularização [59].

Portanto, introduzir modelos baseados na arquitetura GRU possibilita a solução de problemas, como o gradiente desvanecendo e explodindo. No entanto, ainda assim, como visto nos capítulos anteriores, esses modelos sofrem com problemas de geração de informações inadequadas (sentenças não descritivas). Dessa maneira, o próximo capítulo apresenta um modelo baseado na arquitetura codificador-decodificador. O modelo proposto utiliza CNN-GRU, introduzindo um módulo de geração de pesos capaz de resolver o problema, com as informações inadequadas, treinando a rede recorrente com diferentes pesos, de acordo com a sua importância para geração das informações.

CAPÍTULO 4

Proposta Metodológica

Neste capítulo são propostas duas abordagens baseadas na arquitetura codificador-decodificador, utilizando CNN e RNN para a descrição de imagens (Figura 4.1). Para a extração das características, utilizou-se os modelos Inception-V3 e VGG19 como codificadores, tanto no treinamento quanto na validação.

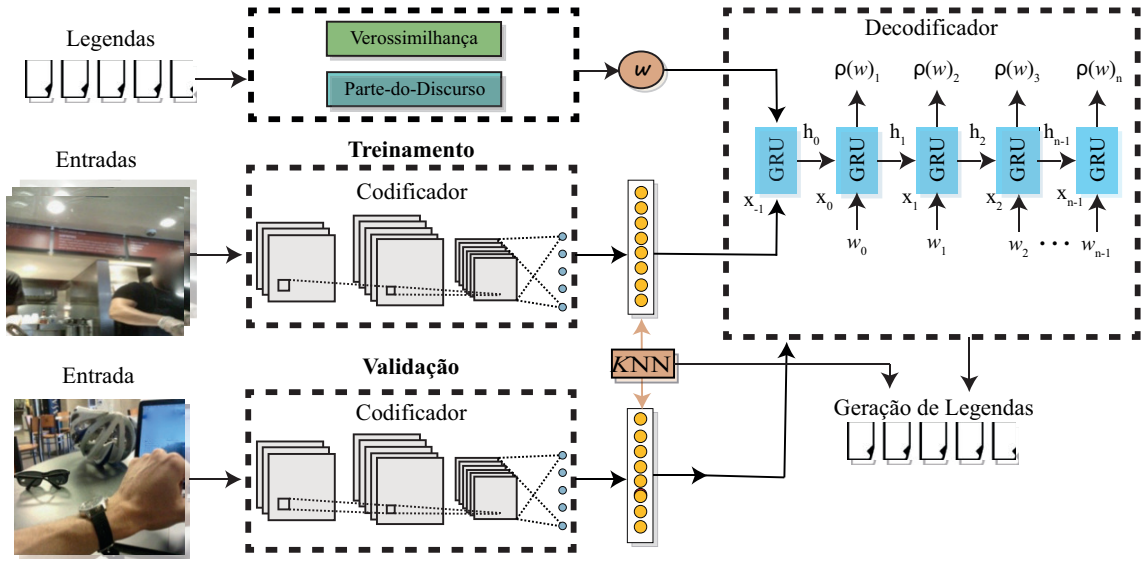


Figura 4.1: *Modelo proposto baseado na estrutura codificador-decodificador - r-GRU (PoS e Verossimilhança).*

A seleção do modelo Inception-V3 justifica-se pelo fato de que, o mesmo possui um custo computacional menor quando comparado aos demais modelos [164, 63]; além de ser uma das redes profundas mais utilizadas para transferência de aprendizado [12]. Ao mesmo tempo, o VGG19 é um dos modelos convolucionais mais populares, o qual possui como vantagens a simplicidade e praticidade, além de desempenho significativo para tarefas de classificação, detecção e legendagem de imagens [152].

4.1 Arquitetura dos Modelos Propostos

Na arquitetura que utiliza o Inception-V3, tradicionalmente, o modelo possui entrada de 299*x*299*x*3, gerando uma *pool* de saída de 8*x*8*x*2048, reduzindo assim, o número de conexões e parâmetros, ganhando eficiência (Figura 4.2).

Inception-V3

Camada	Entrada	Filtros
conv	299x299	3
conv	149x149	32
conv padded	147x147	32
pool	147x147	64
conv	73x73	64
conv	71x71	80
conv	35x35	192
3 Inception	35x35	288
5 Inception	17x17	768
2 Inception	8x8	1280
pool	8x8	2048
linear	1x1	2048
softmax	1x1	1000
Codificador		

Figura 4.2: *Arquitetura completa do Inception-V3.*

Este modelo realiza cálculos dos pesos no conjunto de legendas de entrada por meio da análise da PoS ou da função de verossimilhança, alimentando a GRU com os pesos e as características das imagens da CNN (Figura 4.3).

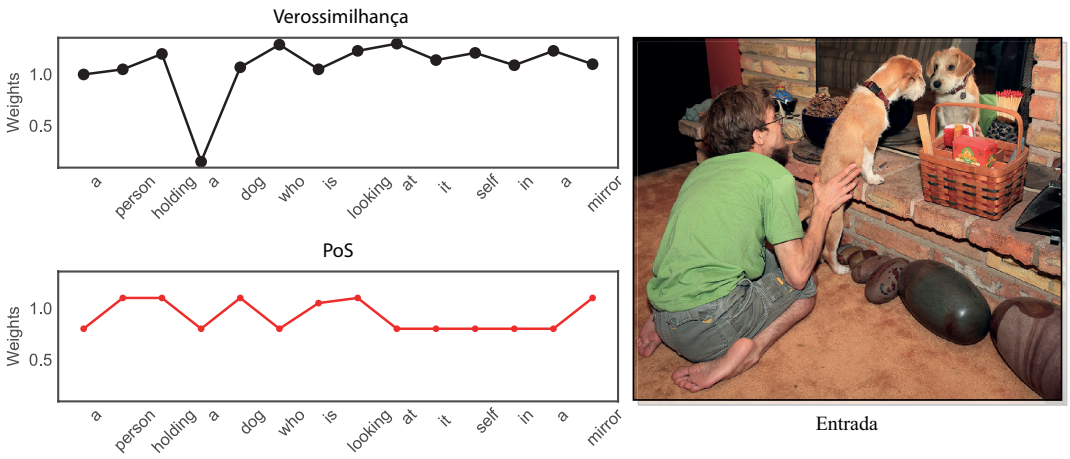


Figura 4.3: *Geração de pesos por meio da análise da PoS ou da função de verossimilhança.*

Na fase de validação, utilizou-se *k*NN para identificar as legendas mais próximas da imagem de entrada, selecionando a legenda com maior similaridade. Dessa forma, são

comparadas as similaridades entre as legendas de referência (melhor legenda da *k*NN) e a legenda de predição (legenda gerada pelo modelo).

Para verificar a eficiência do modelo, inseriu-se uma legenda descritiva da imagem de entrada, fornecida por um especialista linguístico. Para essa avaliação utilizou-se a métrica Meteor [11] entre as legendas fornecidas pelo modelo.

Não obstante, o modelo baseado na arquitetura VGG19 possui dezenove camadas profundas, utilizando filtros de 3x3 para captura detalhada das características das imagens, contendo cinco estágios de camadas de convolucões, cinco camadas *pool* e três camadas *fully connected* (Figura 4.4).

VGG19

Camada	Entrada	Filtros
input	224x224	3
2 conv	244x244	64
maxpool	112x112	128
2 conv	112x112	128
maxpool	56x56	256
4 conv	56x56	256
maxpoll	28x28	512
4 conv	28x28	512
maxpool	14x14	512
4 conv	14x14	512
maxpool	7x7	512
2 fc	1x1	4096
1 fc	1x1	1000
softmax	classifier	
Codificador		

Figura 4.4: *Arquitetura completa do VGG19.*

O VGG19 possui entrada de 224x224 com a última camada como um classificador, utilizando a regressão por meio da função softmax, classificando as imagens de entrada por probabilidade. Este modelo também realiza cálculos dos pesos no conjunto de legendas de entrada por meio da análise da PoS ou da função de verossimilhança, alimentando a GRU com os pesos e as características das imagens extraídas pela CNN.

Similar ao modelo anterior, na fase de validação, utilizou-se também a *k*NN para realização da identificação das legendas mais próximas da imagem de entrada - legenda com maior similaridade – comparando as similaridades entre as legendas de referência e a legenda de predição. Para verificar a eficiência deste modelo, forneceu-se uma legenda descritiva da imagem de entrada (especialista linguístico), mensurando as pontuações Meteor entre as legendas fornecidas pelo modelo (melhor legenda da *k*NN, a legenda de predição e a legenda fornecida pelo especialista linguístico).

Nas duas abordagens, adotou-se um decodificador baseando-se em GRUs:

$$\mathbf{z_t} = \sigma(X_t W_z + U_z h_{t-1} + b_z) \tag{4-1}$$

$$\mathbf{r_t} = \sigma(X_t W_t + U_r h_{t-1} + b_r) \tag{4-2}$$

$$\tilde{\mathbf{h}}_\mathbf{t} = \varphi[X_t W_h + (R_t \bigodot h_{t-1}) U_h + b_h] \tag{4-3}$$

$$\mathbf{h_t} = (1 - Z_t) \bigodot h_{t-1} + Z_t \bigodot \tilde{h}_t \tag{4-4}$$

A GRU é treinada com os pesos advindos da aplicação da análise da PoS ou da função de verossimilhança, além das características das imagens extraídas pela CNN. Neste sentido, a GRU adotada neste trabalho possui unidades de conexão capazes de modular as informações, sem necessariamente realizar a separação das células de memória utilizando a porta de atualização (z_t) e de redefinição (r_t).

4.2 Geração de Pesos por meio da Parte-do-Discurso

Sabe-se que, na geração da descrição de uma determinada imagem existem palavras que são mais representativas, ou seja, há ocorrência de palavras que possuem maior significância quando comparadas às demais. Assim, gerar pesos diferentes durante o treinamento da rede neural, de acordo com o seu grau de significância para a legenda pode ser uma boa estratégia.

Não obstante, alguns dos argumentos que um decodificador de palavras utiliza são necessários apenas em algumas etapas, não necessitando persistir essas informações ao longo do processo da legendagem [64].

Considerando este fator e motivados pela proposta de Deshpande *et al*. [41], onde aplicaram a análise da PoS na geração significativa de sentenças de imagens, propõem-se neste trabalho uma abordagem onde calcula-se os pesos das palavras antes do treinamento da rede recorrente.

Neste sentido, as etiquetas PoS podem ser definidas como sendo um conjunto sintático de classes de palavras PoS, por exemplo, substantivo, verbos, adjetivos e etc. Neste caso, o analisador seleciona um subconjunto de etiquetas (θ), procurando pela PoS de w_{n-1} neste subconjunto (θ), caso encontre, o comutador linguístico é ativado.

Por exemplo, suponhamos que existe um conjunto de frases definido por $S(I) = \{w_0, w_1, w_2, ..., w_{n-1}\}$, onde w seja a palavra rotulada da imagem I, w_0 será a palavra inicial, w_{n-1} a parada final da sequencia de rótulos e n é o tamanho da sentença para I.

No instante de n, a probabilidade da palavra é analisada pela a entrada da sentença da imagem e pelas etiquetas geradas no conjunto $S(I)$. Assim, considerando a frequência de uma determinada palavra dentro de uma sentença de entrada, o peso desta palavra pode ser calculado da seguinte forma:

$$\mathbf{P}(\mathbf{S_n}) = \frac{c(w_n)}{N_{token}(S)} \tag{4-5}$$

onde $c(w_n)$ é o número de vezes que a palavra w_n aparece na sentença e $N_{token}(S)$ representa o número total de palavras do *token*[1] da sentença de entrada S.

No entanto, mesmo considerando a frequência de uma determinada palavra em uma sentença, é razoável prever que palavras diferentes não possuem a mesma relevância para o entendimento ou para a formação de uma determinada frase [193].

Dessa forma, percebe-se que verbos, substantivos e conjunções possuem pesos maiores quando comparadas as demais classes de palavras. Assim, este modelo considera a importância da palavra para a descrição da imagem, atribuindo pesos mais altos para o conjunto de palavras mais importantes. A Tabela 4.1 apresenta os pesos adotados por este modelo para diferentes classes de palavras.

Tabela 4.1: *Geração de pesos de acordo com a classe de palavra.*

Etiqueta	**Classe**	W_{weight}
NOUM	Substantivo	1.10
VERB	Verbo	1.05
ADJ	Adjetivo	0.90
ADV	Advérbio	0.90
SCONJ	Conjução Subordinativa	0.80
INTJ	Interjeição	0.80
DET	Artigo	0.00
CCONJ	Conjução Coordenativa	0.80
CONJ	Conjunção	0.80
AUX	Verbo Auxiliar	0.80
PREP	Preposição	0.80

De maneira simples, os pesos para diferentes classes são dados da seguinte forma:

$$\eta_{\mathbf{w}} = \left\{ w_{weight}, if(PoS(S_n) = classe) \right\} \tag{4-6}$$

onde PoS (S_n) é a parte-do-discurso da sentença S e w_{weight} é o peso da palavra de acordo com a categorização de sua classe. Dessa forma, aplicou-se o cálculo dos pesos em uma fase de pré-processamento do modelo, analisando a PoS e inferindo diferentes pesos de

[1] Unidade semântica de processamento, por exemplo, uma palavra, frase ou parágrafo.

acordo com sua classe de palavra e frequência, isto é, a sua ocorrência dentro da sentença (Eq. 4-7).

$$\mathbf{PoS_{weight}}(\mathbf{S}) = \sum_{n=0}^{N_{token}(S)-1} (\eta_{w_n} * h_{n+1}) \tag{4-7}$$

onde h representa o estado oculto no instante de $n+1$ e η_{w_n} é o parâmetro com o peso da palavra n na sentença S, sendo este valor dependente da classe à qual a palavra é classificada. Por exemplo, dado a seguinte sentença:

$$\mathbf{S} = [\text{“}a\text{”}, \text{“}boy\text{”}, \text{“}holding\text{”}, \text{“}a\text{”}, \text{“}racket\text{”}, \text{“}is\text{”}, \text{“}standing\text{”}, \text{“}in\text{”}, \text{“}the\text{”}, \text{“}grass\text{”}] \tag{4-8}$$

Aplicando η_w obtem-se os seguintes pesos:

$$\eta_{\mathbf{w}} = [0, 1.10, 1.05, 0, 1.10, 1.05, 1.10, 0.80, 0, 1.10] \tag{4-9}$$

Observa-se que $N_{token(S)}$ é igual a 10, ou seja, o número total de palavras de S é igual a 10. Assim, a geração dos pesos finais será:

$$\mathbf{PoS_{weight(S)}} = [(0 * h_1) + (1.10 * h_2) + ... + (1.10 * h_{10})] \tag{4-10}$$

Após esta etapa, a rede recorrente é treinada aplicando os pesos advindos da $PoS_{weight(S)}$:

$$\tilde{\mathbf{h}}_{\mathbf{t}} = \varphi[X_t W_h + (R_t \bigodot h_{t-1}) U_h + b_h] \tag{4-11}$$

$$\mathbf{h_t} = (1 - Z_t) \bigodot h_{t-1} + Z_t \bigodot \tilde{h}_t \tag{4-12}$$

A GRU recebe uma sequência de entradas, atualizando tanto o estado de memória quanto o estado oculto de h_t. Neste caso, h_t é representado por h_{n+1} na equação 4-7.

4.3 Geração de Pesos por meio da Verossimilhança

Como mencionado anteriormente na geração de pesos por meio da PoS, dado uma sentença S de uma determinada imagem I composta por uma sequência de palavras de rótulos w_n, os modelos são capazes de gerar a probabilidade de cada palavra no instante de tempo n, onde n varia de 0 (inicial) a $n-1$ (final). Percebe-se que n denota a quantidade

de palavras composta em S na imagem I. Assim, as probabilidades das palavras w_n são dependentes de I:

$$\rho(\mathbf{w_n}|\mathbf{I}) = \rho(w_1|I) \cdot \rho(w_2|I) \cdots \rho(w_n|I) \tag{4-13}$$

Para geração das probabilidades das palavras, utilizou-se a função logarítmica de verossimilhança e o estimador de máxima verossimilhança (MLE) em I:

$$\mathcal{L}(\mathbf{I}|\mathbf{w_n}) = \sum_{n=1}^{n} \log \rho(w_n|I) \tag{4-14}$$

$$\mathcal{L}(\mathbf{I}|\mathbf{w_n})_{\mathbf{mle}} = \arg\max_{I \in I} \mathcal{L}(I|w_n) \tag{4-15}$$

onde $\mathcal{L}(I|w_n)$ denota a função logarítmica de verossimilhança e $\mathcal{L}_{mle}$ o Estimador de Máxima Verossimilhança (MLE) dos pesos da sentença S da imagem I. Não obstante, assim como no modelo PoS, considerou-se que as palavras possuem importância diferente para formação da sentença S.

Dessa forma, o modelo gera pesos maiores para as palavras que indicam maior importância para o entendimento da sentença, como por exemplo, palavras que descrevem o objeto principal de uma determinada cena da imagem I. Dessa forma, o peso baseado na importância da palavra em relação a sentença S é dada por $\phi(w|S)$:

$$\mathcal{L}(\mathbf{I}|\mathbf{w_n}) = \sum_{n=1}^{n} \phi(w|S) \log \rho(w_n|I) \tag{4-16}$$

Para calcular os pesos de cada palavra em relação a sentença S, utilizou-se a seguinte equação:

$$\phi(\mathbf{w_n}|\mathbf{S}) = \beta\rho(w_n|I) \tag{4-17}$$

onde $\rho(w_n|I)$ denota a probabilidade de w_n na sentença S da imagem I e β garante que a média de todos os pesos das palavras da sentença S não ultrapasse ao valor 1. Nesta etapa, para realizar o cálculo do peso em $\rho(w_n|I)$, aplicou-se a Estimativa Robusta de Densidade por *Kernel* (RKDE) [89]. O RKDE estima uma função de densidade de probabilidade para $\rho(w_n|I)$:

$$\beta\rho(\mathbf{w_n}|\mathbf{I}) = \frac{\beta}{|X_{w_n}|} \sum_{I_n \in X_{w_n}} K_\sigma(I - I_n) \tag{4-18}$$

onde $|X_{w_n}|$ representa o conjunto de imagens I onde as sentenças contem a palavra w_n e

K_σ representa a função Gaussiana de Densidade do *Kernel*.

$$\mathbf{K_\sigma(I-I_n)} = (\frac{1}{\sqrt{2\pi\sigma}})^d \; exp(-\frac{||I-I_n||^2}{2\sigma^2}) \tag{4-19}$$

O parâmetro σ em K representa a distância média entre duas imagens em um conjunto de imagens de treinamento. Dessa forma, por meio da aplicação do RKDE pode-se inferir pesos diferentes de acordo com a importância da palavra. Por exemplo, se um conjunto de imagens semelhantes contem palavras semelhantes em suas sentenças, o modelo infere pesos maiores para essas palavras, medindo a importância das palavras tanto nas legendas quanto para as imagens.

Após esta etapa, a GRU é treinada aplicando os pesos advindos da $\mathcal{L}(I|w_n)$. Dessa forma, a GRU recebe os pesos $\mathcal{L}(I|w_n)$ atualizando o estado oculto de memória de $h_t = \left\{(1-Z_t)\bigodot h_{t-1} + Z_t \bigodot \tilde{h}_t\right\}$.

4.4 Geração de Legendas por Referências

Após o treinamento na estrutura codificador-decodificar o modelo produz uma sequência de descrições $S = \left\{S_{pred}, S_{ref}, S_{ling}\right\}$ para uma determinada imagem de entrada I. Dessa forma, S é um conjunto com três sentenças: a sentença de predição, a sentença de referência e a sentença linguística. Esta última, fornecida como entrada no modelo por um especialista linguístico.

Observa-se que, além da legenda de predição gerada pela própria GRU, o modelo indica ainda a melhor legenda gerada por referências utilizando a pontuação de consenso. A pontuação de consenso representa as descrições das imagens semelhantes, tornando-se muito útil para geração de sentenças de referências [42]. Neste sentido, as sentenças de referências pode ser definida como um conjunto de frases de uma ou mais imagens de treinamento que possuem características similares com uma determinada imagem de entrada (validação).

Algumas abordagens utilizam as pontuações de consensos em um processo de substituição da legenda da imagem de validação pela legenda da imagem mais próxima [42] [117]. Por exemplo, no processo de validação, dado uma imagem I com $S = \{s_0, s_1, ..., s_n\}$, a pontuação de consenso identifica no conjunto de treinamento k mais próximos de I, realizando a junção de todas as legendas de k, criando um conjunto de n legendas candidatas para I. Após esta etapa, o modelo seleciona a melhor legenda deste conjunto, realizando a substituição da legenda de entrada.

Em nossa abordagem aplicou-se o kNN para recuperar as legendas de n imagens mais próximas da imagem de entrada I, criando um conjunto com C de legendas para I.

Neste contexto, de acordo com Devlin et al. [42], as abordagens que aplicam *k*NN para estes tipos de tarefas, superam as abordagens inovadoras de geração de legendas.

Ao contrário dos modelos tradicionais, o nosso modelo não realiza a substituição da legenda de predição pela a melhor legenda da pontuação de consenso, utilizando-se a melhor legenda de referência (S_{ref}) para comparar com a legenda de predição (S_{pred}) gerado pelo modelo e a legenda fornecida por um especialista linguístico (S_{ling}).

Neste sentido, para a identificação de similaridade entre as legendas da imagem de treinamento e de validação, adotou-se a seguinte equação:

$$\mathbf{s_{ref}}(\mathbf{s_a}|\mathbf{s_b}) = \arg\max_{s_a \in S} \sum_{s_b \in S} Sim(s_a, s_b) \tag{4-20}$$

onde $Sim(s_a, s_b)$ denota a pontuação de similaridade entre as legendas S_a (predição) e S_b (treinamento). Após a aplicação de $Sim(s_a, s_b)$ o modelo é capaz de selecionar a legenda com a maior semelhança lexical média no conjunto $S_{ref}(s_a|s_b)$. Para esta tarefa, utilizou-se a métrica de avaliação Meteor [11]:

$$\mathbf{Sim}(\mathbf{s_a}, \mathbf{s_b}) = (\frac{10PR}{R+9P}) \cdot (1 - (0.5\frac{C}{M_u})) \tag{4-21}$$

onde P representa a taxa de precisão e R a taxa de recuperação. M_u denota os números de palavras correspondentes e C o número de blocos da sentença que serão analisadas. Dessa forma, P pode ser encontrada pela a razão dos números de palavras mapeadas encontradas entre duas sentenças (m), por exemplo, S_a e S_b, e o número total de palavras da tradução (t).

$$\mathbf{P} = \frac{m(s_a, s_b)}{t} \tag{4-22}$$

A taxa de recuperação R por sua vez, pode ser encontrada pela a razão de o número de palavras mapeadas entre duas sentenças (S_a, S_b) e o número total de palavras de referência (r).

$$\mathbf{R} = \frac{m(s_a, s_b)}{r} \tag{4-23}$$

Com intuito de realizar as comparações entre as legendas de previsão (S_{pred}), a legenda selecionada pelo *K*NN (S_{ref}) e a legenda fornecida por uma especializada linguístico (S_{ling}), aplicou-se novamente a função verificadora de verossimilhança entre os pares das três legendas. Para tal, utilizou-se novamente a métrica de avaliação de pontuação de consenso Meteor:

$$\mathbf{Sim}(\mathbf{S_{pred}}, \mathbf{S_{ling}}) = (\frac{10PR}{R+9P}) \cdot (1 - (0.5\frac{C}{M_u})) \tag{4-24}$$

$$\mathbf{Sim}(\mathbf{S_{ref}}, \mathbf{S_{ling}}) = (\frac{10PR}{R+9P}) \cdot (1 - (0.5\frac{C}{M_u})) \tag{4-25}$$

Essa etapa objetiva-se a comparar a eficiência do sistema de geração de legendas, verificando quão próximas são as sentenças de predição e de referência em relação a sentença fornecida pelo especialista linguístico.

Não obstante, ainda na etapa de validação do modelo, para otimizar o processo de busca, utilizou-se o Beam Search (Tabela 4.2).

Tabela 4.2: *Pseudocódigo do algorítmo Beam Search.*

Algorithm Beam Search
Input: K_{Beam} B, input x, Parameters Θ
Output: Approx. -best summaries
$\pi[0] \leftarrow \{\varepsilon\}$
$S = V \; if \; abstractive \; else \; \{x_i \vert \forall_i\}$
for $i = 0 \; to \; N-1$ **do**
$\triangleright \; Generate \; Hypotheses$
$N \leftarrow \{[y, y_{i+1}] \vert y \in \pi[i], y_{i+1} \in S\}$
$\triangleright \; Hypotheses \; Recombination$
$H \leftarrow \left\{ \begin{matrix} y \in N \vert s(y,x) > s(y,x) \\ \forall y \in N \; s.t. y_c = y_c \end{matrix} \right\}$
$Filter \; B - Max$
end for
return $\pi[N]$

O Beam Search decide o número total de palavras que deverão ser mantidas na memória em um espaço de tempo, sendo um hiperparâmetro ajustável. Assim, para cada etapa, mantem-se as melhores frases das kNN candidatas. Neste sentido, o próximo capítulo apresenta os detalhes dos experimentos com os valores destes parâmetros.

CAPÍTULO 5

Experimentos

Neste capítulo são apresentados os conjuntos de dados utilizados na pesquisa, bem como, maiores detalhes sobre os parâmetros, técnicas e métricas de avaliação aplicadas durante os experimentos.

5.1 Conjunto de Dados

Realizou-se experimentos em dois conjuntos de dados publicamente disponíveis, o Flickr30K [142] e MS-COCO (*Microsoft Common Objects in Context*) [110]. O conjunto de dados do Flickr30k possui 8 000 imagens contendo 6 000 imagens para treinamento, 1 000 para teste e 1 000 para validação. O MS-COCO é um dos maiores conjuntos de dados de imagem rotuladas, contendo 164 062 imagens, sendo 82 783 imagens para o treinamento, 40 504 para testes e 40 504 para validação. Cada imagem, em ambos os conjuntos de dados, possui cinco legendas descritivas (Figura 5.1).

Flickr30k

S01 A boy holding a racket is standing in the grass.
S02 A boy in a purple shirt and blue jeans is standing in the grass holding a round toy with a handle.
S03 A boy playes with a stringless racket in his backyard.
S04 A boy walks through the lawn.
S05 A child in purple clothing is playing with a toy inside a yard with other toys and a fence.

MS-COCO

S01 This is two men in cleats on the grass.
S02 Two young men playing frisbee in a park.
S03 A couple of men playing frisbee against each other on a field.
S04 Two teenagers playing of game of Frisbee in a field.
S05 Two young men play with a Frisbee in a grassy area.

Figura 5.1: *Conjuntos de dados publicamente disponíveis, o Flickr30K e MS-COCO.*

5.2 Configurações dos Experimentos

Nos experimentos, utilizou-se a pesquisa heurística conhecido como Beam Search. O Beam Search é um algoritmo de inferência de maior popularidade quando se trata de modelos de decodificação neural. Este algoritmo permite a tomada de decisões locais não gananciosas, obtendo uma sequência com a probabilidade geral mais alta, reduzindo ao mesmo tempo, os requisitos de memória [33]. Neste sentido, para a otimização do Beam Search, utilizou-se dois parâmetros, k_{beam} e α. O parâmetro k_{beam} é a largura do feixe para a geração das sentenças, indicando o número de elementos principais a serem procurados. O parâmetro α indica o peso do modelo de linguagem.

Neste modelo, utilizou-se as arquiteturas padrões do Inception-V3 e VGG19 como codificadores e a GRU como decodificador. Assim, ajustou-se as iterações do codificador para 1 000 iterações. Objetivando-se solucionar um possível problema com explosão do gradiente, adotou-se o RMSProp como otimizador estocástico [194], definido pelas seguintes equações:

$$\mathbf{v_t^w} = \beta * v_{t-1}^w + (1-\beta)(\bigtriangledown w_t)^2 \tag{5-1}$$

$$\mathbf{w_{t+1}} = w_t - \frac{\eta}{v_t^w + \varepsilon} * \bigtriangledown w_t \tag{5-2}$$

$$\mathbf{v_t^b} = \beta * v_{t-1}^b + (1-\beta)(\bigtriangledown b_t)^2 \tag{5-3}$$

$$\mathbf{b_{t+1}} = b_t - \frac{\eta}{\sqrt{v_t^b + \varepsilon}} * \bigtriangledown b_t \tag{5-4}$$

Com intuito de evitar problemas com *overfitting* [113], adotou-se *dropout* com uma taxa de 0.1 e uma taxa de aprendizado de $1x10^{-5}$. O parâmetro k_{beam} recebeu os valores $\{1,3,5,10\}$ com $\alpha = 0.7$, otimizando os resultados obtidos pelo Beam Search no processo de validação do modelo.

Tabela 5.1: *Parâmetros usados durantes os experimentos.*

Parâmetro	Valor
Interações	1 000
Otimizador	RMSProp
Taxa de Dropout	0.1
Taxa de Aprendizado	$1X10^{-5}$
k_{Beam}	1, 3, 5 e 10
α	0.7

Os experimentos foram realizados com uma NVIDIA DGX-1, possuindo GPUs de 8 x Tesla V100, com uma memória total de 512 GB e 2.133 MHz, DDR-4 RDIMM.

O treinamento da abordagem utilizando a função de verossimilhança leva cerca de 1.5 dias no Flickr30k e 3.5 dias no MS-COCO. Utilizando a PoS, leva 2.5 e 4.5 no Flickr30k e MS-COCO, respectivamente.

5.3 Métricas para Avaliação do Modelo

Realizou-se as avaliações dos modelos propostos nos conjuntos de dados Flickr30k e MS-COCO, com as métricas de avaliação Bleu, Meteor, cider e Rouge.

5.3.1 Bleu

A métrica de estudo básico de avaliação bilíngue (do inglês *Bilingual Evaluation Understudy* - Bleu), foi utilizada extensivamente na avaliação de algoritmos de tradução de textos automáticos de uma linguagem para a outra [64]. O Bleu é implementado para avaliar a qualidade dos descritores, calculando a precisão de n-gramas[1] entre as frases de referência e as frases candidatas.

Existem várias variações da métrica Bleu. No entanto, a métrica padrão requer um cálculo de penalidade de brevidade P_b calculada da seguinte forma:

$$\mathbf{B_p} = \begin{cases} 1 & if c > r \\ e^{(1-r/c)} & if c \leq r \end{cases} \tag{5-5}$$

onde r torna-se o comprimento do conjunto de dados de referência e c o comprimento da tradução candidata. Assim, a métrica Bleu pode ser determinada por:

$$\mathbf{BLEU} = B_p \cdot exp\left(\sum_{n=1}^{N} w_n log p_n\right) \tag{5-6}$$

onde B_p representa o valor de brevidade da sentença candidata e p_n é a média geométrica da precisão modificada de n-gramas. N é o comprimento máximo dos n-gramas da sentença candidata.

5.3.2 Meteor

A métrica para avaliação de tradução com ordenação explícita (do inglês *Metric for Evaluation of Translation with Explicit ORdering* - Meteor) é uma métrica que avalia

[1] É uma sequência contígua de n itens (sílabas, fonemas, palavras, letras) da amostra de uma determinada fala ou de um texto.

o alinhamento entre as palavras nas frases candidatas e nas frases de referências de objetos [95]. Essa métrica também avalia a tradução automática baseada em um conceito de generalização de correspondência de *unigramas* entre as traduções de referências realizadas por humanos e as traduções realizadas por máquinas. A métrica Meteor pode ser calculada da seguinte forma:

$$\mathbf{METEOR} = (\frac{10PR}{R+9P})(1-P_m) \tag{5-7}$$

onde a recuperação e a precisão de *unigramas* são dadas por R e P, respectivamente. A penalidade de concisão P_m é determinada por:

$$\mathbf{P_m} = 0.5(\frac{C}{M_u}) \tag{5-8}$$

onde M_u é o número de *unigramas* correspondentes e C é o número mínimo de frases necessárias para corresponder os *unigramas* de saídas das traduções de referência.

5.3.3 Cider

A métrica avaliação de descrição de imagem baseada em consenso (do inglês *Consensus-based Image Description Evaluation* - Cider), também, é uma métrica para a avaliação de algoritmos de legendagem de imagens. O Cider consiste em três partes: um método baseado em coletas de anotações humanas, mensurando o consenso; uma métrica automatizada para captura do consenso; e dois conjuntos de dados que descrevem as imagens [171]. Assim, a pontuação Cider para n-gramas de comprimento n pode ser calculado utilizando a semelhança média de consenso entre as sentenças candidatas e as sentenças de referências, as quais são necessariamente responsáveis pela precisão e recuperação:

$$\mathbf{CIDEr_n}(\mathbf{c_i}, \mathbf{S_i}) = \frac{1}{m}\sum_j \frac{g^n(c_i \cdot g^n(s_{ij}))}{\|g^n(c_i\| \left\|g^n(s_{ij})\right\|} \tag{5-9}$$

onde $g^n(c_i)$ é um vetor formado por $g_k(c_i)$ correspondente a todos os n-gramas de comprimento n e $\|g^n(c_i)\|$ é a magnitude do vetor $g^n(c_i)$. Da mesma forma para $g^n(s_{ij})$, utilizando n-gramas de ordem mais alta para capturar propriedades semânticas e gramaticais mais ricas. Assim, pode-se combinar as pontuações de n-gramas de comprimentos variados da seguinte forma:

$$\mathbf{CIDEr}(\mathbf{c_i}, \mathbf{S_i}) = \sum_{n=1}^{N} w_n CIDEr_n(c_i, S_i)) \tag{5-10}$$

5.3.4 Rouge

A métrica de estudo básico orientado à lembrança para avaliação de resumos (do inglês *Recall-Oriented Understudy for Gisting Evaluation* - Rouge) é uma métrica composta por medidas que determinam automaticamente a qualidade de um resumo, comparando-o com outros resumos criados por seres humanos [109]. Assim, a métrica Rouge conta o número de palavras entre o resumo gerado automaticamente pelo gerador e o resumo ideal realizado pelo ser humano. Formalmente, a métrica Rouge é uma recuperação de *n*-gramas entre um resumo candidato e um conjunto de resumos de referências. A métrica Rouge é calculada da seguinte forma:

$$\textbf{ROUGE} = \frac{\sum_{S \in S_H} \sum_{g_n \in S} C_m(g_n)}{\sum_{S \in S_H} \sum_{g_n \in S} C(g_n)} \tag{5-11}$$

onde n representa o comprimento de n-grama, g_n e $C_m(g_n)$ é o número máximo de n-gramas que co-ocorrem em um resumo candidato e em um conjunto de resumos de referências.

5.3.5 API de avaliação MS-COCO

Para a realização dessas avaliações, neste trabalho, adotou-se a Interface de Programação de Aplicações (do inglês *Application Programming Interface* - API) de avaliação do MS-COCO. Essa API garante a consistência na avaliação de algoritmos de geração de legendas utilizando um serviço de avaliação [26]. Este serviço de avaliação possui a função de receber as legendas candidatas, realizando a classificação dessas legendas por meio de métricas populares, incluindo as métricas Bleu, Meteor, Cider e Rouge.

5.4 Abordagens para os Benchmarks

Nesta seção, são apresentados de forma breve, as similaridades e diferenças entre alguns modelos do estado da arte baseados na arquitetura codificador-decodificar e o método aqui proposto. Estes modelos são utilizados como *benchmarks*, realizando a comparação do desempenho ora proposto em relação aos resultados de ponta.

- **phi-LSTM [166]** - a arquitetura hierárquica de Memória Longa de Curto Prazo (phi-LSTM), diferentemente das arquiteturas tradicionais, introduz um mecanismo de decodificação de legendas de frase-em-frase, gerando legendas de maneira não sequencial. A estrutura baseia-se em LSTM dupla, composta pelo decodificador de frase e o decodificador de sentenças abreviadas. Dessa forma, dada uma imagem

qualquer decodificada pela CNN, o modelo decodifica primeiramente as frases substantivas descrevendo as entidades dominantes da imagem. O decodificador também codifica cada uma das frases substantivas em um vetor de representações. Este vetor servirá como entrada no decodificador das sentenças abreviadas. Assim, o modelo LSTM gera em cada etapa de decodificação duas saídas, um indicador binário determinando a próxima entrada e uma função softmax com a previsão da sequência da próxima frase. Similar ao método aqui proposto, o modelo phi-LSTM utiliza a função de verossimilhança para realizar os cálculos dos pesos das sentenças baseando-se nas previsões das palavras e nas perdas das indicações das frases. No entanto, neste modelo, diferentemente do modelo phi-LSTM, considerou-se ainda, a aplicação da análise PoS como uma alternativa para geração de pesos para o decodificador, além de introduzir um decodificador mais simples e um codificador que utiliza uma estrutura diferente. Enquanto o modelo phi-LSTM baseia-se na estrutura VGG para a codificação, a nossa proposta utiliza-se a estrutura Inception-V3 como codificador.

- **He *et al.* [64]** - tal modelo é baseado na arquitetura codificador-decodificador, empregando uma CNN como decodificador e uma RNN como decodificador. Além da estrutura popularmente utilizada, o modelo introduz a avaliação das marcas da parte-do-discurso das sentenças. Dessa forma, o modelo combina os vetores de recursos visuais à incorporação de palavras, superando alguns problemas de ajustes excessivos nas entradas da LSTM. Diferentemente deste modelo, a nossa abordagem utiliza a LSTM para a geração das sentenças.

- **Ding *et al.* [45]** - tal modelo baseia-se em uma arquitetura baixo-para-cima com aplicação de uma Fast-RCNN para a extração de recursos de imagem de alto nível, combinando as informações de baixo nível com os recursos de alto nível, detectando regiões de atenção de uma determinada imagem. Primeiro, o modelo realiza o rastreamento de alvos e regiões nas imagens para a descrição semântica. Depois dessa etapa, o modelo cria um vetor de características correspondente a CNN, considerando a projeção convolucional e os cálculos das matrizes de pesos. Na última etapa, essas informações são inseridas no modelo de geração de linguagem, obtendo sentenças das descrições das imagens correspondentes. Diferentemente do modelo aqui proposto, tal modelo utiliza-se da LSTM para geração das descrições das imagens.

- **3G [188]** - tal modelo combina as informações locais e globais das imagens para a geração de legendas, utilizando-se de três portas para os recursos globais e locais,

decidindo quais recursos globais deverão ser incorporados. O modelo propõe a utilização de três portas, utilizando-se das Portas de Feedbacks da LSTM (GF-LSTM) para geração das sentenças. Dessa forma, a primeira porta controla a quantidade de recursos globais que devem ser incorporados pelo modelo multimodal, e as demais portas são utilizadas para o módulo de idioma e para o módulo de incorporação multimodal. Assim, as últimas portas são responsáveis pela a ligação entre a imagem e os textos. A diferença entre o modelo 3G e o modelo aqui proposto, é a utilização da análise PoS e a função de verossimilhança no processo de geração de peso junto às GRUs multimodais para geração de sentenças. Além disso, uma das vantagens do modelo aqui proposto, em comparação com o modelo 3G, é a simplicidade de implementar o decodificador para gerar sentenças e a aplicação do *k*NN para geração de sentenças por referência.

- **m-GRU [105]** - tal modelo apresenta uma abordagem baseado em encapsulamento multimodais, por exemplo, GRU. Diferentemente dos modelos do estado-da-arte que são baseados em LSTM e similar ao método aqui proposto, tal modelo gera descrições de imagens de forma variável por meio de uma GRU multimodal. Assim, a CNN extrai os recursos das imagens enquanto a GRU multimodal incorpora tais recursos. A diferença entre esse modelo e o modelo aqui proposto, está no processo de geraração de pesos e na utilização da *k*NN na fase de validação para geração de descrições mais próximas, ou seja, geração das descrições através de referências.

- **Google-NIC [172]** - o NIC é um modelo neural baseado na LSTM e CNN para a descrição de imagens, utilizando-se um maximizador de probabilidade para a descrição da imagem alvo. Diferentemente deste modelo, o modelo aqui proposto utiliza a GRU multimodal para a geração das descrições das imagens, além de utilizar-se da transferência de aprendizado a fim de buscar as legendas do conjunto de treinamento mais próximas da imagem alvo.

CAPÍTULO 6

Resultados e Discussões

Nesta seção são apresentados os estudos de ablações dos modelos baseados nas arquiteturas Inception-V3 e VGG19; os estudos de benchmarks comparativos com os modelos de ponta; e as análises aprofundadas das gerações de legendas para as imagens de entradas na fase de validação do modelo.

Para a avaliação do desempenho o modelo r-GRU-Inception e r-GRU-VGG durante os experimentos, percebeu-se que, quanto maior a quantidade de imagens selecionadas para os testes, maiores os desafios para as duas abordagens. Isso ocorre devido à necessidade de empregar recursos computacionais maiores para o processamento massivo desses dados.

Sendo assim, nos experimentos, configurou-se o Flickr30k com 1 000 imagens para treinamento, 1 000 para teste e 4 000 para a validação. No MS-COCO foram separadas 1 000 imagens para treinamento, 1 000 para teste e 4 000 para a validação.

6.1 Estudo de Ablação do Modelo Baseado na Arquitetura Inception

Inicialmente, configurou-se os parâmetros k_{Beam} e α a fim de identificar a influência da largura do feixe e o peso do modelo de linguagem nas pontuações finais entre os modelos aqui propostos. No entanto, durante os experimentos percebeu-se que, o peso do modelo de linguagem não alterava de forma significativa os resultados das pontuações em ambos os modelos, tanto no modelo r-GRU-Inception baseado na geração de pesos por meio da verossimilhança, quanto no modelo baseado na PoS. Assim, inferiu-se diferentes valores para o parâmetro k_{Beam}, resultando em diferentes pontuações em ambos os modelos (Tabela 6.1).

Observa-se que, o parâmetro k_{Beam} assumiu quatro valores com pequenos intervalos, inferindo os seguintes valores para este parâmetro: 1, 3, 5 e 10. No entanto, quando se analisa as diferenças entre as pontuações alcançadas pelos modelos nos intervalos de largura do feixe entre 1 e 10 para geração de sentenças, nota-se a existência de ocorrên-

Tabela 6.1: *Estudo de ablação com parâmetro k_{Beam} no conjunto de dados Flickr30k e MS-COCO, entre os modelos r-GRU-Inception (verossimilhança) e r-GRU-Inception (PoS).*

Flickr30k								
Métricas	$k_{Beam} = 1$		$k_{Beam} = 3$		$k_{Beam} = 5$		$k_{Beam} = 10$	
	Verossimilhança	**PoS**	**Verossimilhança**	**PoS**	**Verossimilhança**	**PoS**	**Verossimilhança**	**PoS**
B-1	0.651	0.658	0.690	0.689	0.694	**0.695**	0.691	0.693
B-2	0.435	0.450	0.454	0.458	0.458	**0.463**	0.456	0.461
B-3	0.320	0.328	0.332	0.336	0.336	**0.341**	0.334	0.338
B-4	0.222	0.227	0.229	0.230	0.231	**0.232**	0.230	0.231
METEOR	0.281	0.287	0.294	0.298	0.298	**0.302**	0.295	0.300
CIDEr	0.463	0.470	0.476	0.482	0.479	**0.486**	0.477	0.484
ROUGE	0.426	0.432	0.437	0.438	0.442	**0.451**	0.439	0.450
MS-COCO								
Métricas	$k_{Beam} = 1$		$k_{Beam} = 3$		$k_{Beam} = 5$		$k_{Beam} = 10$	
	Verossimilhança	**PoS**	**Verossimilhança**	**PoS**	**Verossimilhança**	**PoS**	**Verossimilhança**	**PoS**
B-1	0.748	0.760	0.751	0.763	0.754	**0.766**	0.753	0.764
B-2	0.558	0.565	0.560	0.568	0.563	**0.571**	0.561	0.570
B-3	0.428	0.431	0.430	0.434	0.431	**0.438**	0.431	0.438
B-4	0.309	0.316	0.311	0.319	0.315	**0.322**	0.313	0.322
METEOR	0.270	0.268	0.273	0.271	**0.276**	0.275	0.274	0.274
CIDEr	1.076	1.097	1.081	1.102	1.090	1.109	1.085	**1.110**
ROUGE	0.532	0.538	0.534	0.540	0.539	**0.543**	0.539	**0.543**

cias de pontuações com diferenças significativas. Assim, no conjunto de dados Flickr30k o modelo baseado em verossimilhança obteve a pontuação Bleu-1 (0.651) com o parâmetro $k_{Beam} = 1$, enquanto que o mesmo modelo obteve a pontuação Bleu-1 (0.691) com o parâmetro $k_{Beam} = 10$.

Este aumento de pontuação foi verificado em todas as métricas, em ambos os modelos, tanto no modelo baseado em verossimilhança quando no modelo com PoS. Nota-se também que, estes resultados corroboram com as pontuações obtidas no conjunto de dados MS-COCO. Sendo assim, em ambos os conjuntos de dados, os modelos baseados na arquitetura Inception-V3 obtiveram diferenças significativas entre os intervalos na largura do feixe para geração de sentenças.

Analisando apenas as melhores pontuações, percebe-se que, ambos os modelos, tanto na abordagem com verossimilhança, quanto na abordagem com PoS, obtiveram pontuações maiores quando inferiu-se a largura do feixe igual a 5 (Figura 6.1).

Entretanto, em alguns casos os modelos tiveram pontuações maiores nas métricas de avaliações com uma largura do feixe maior, isto é, com o parâmetro $k_{Beam} = 10$ (por exemplo, Cider e Rouge no conjunto de dados MS-COCO). No entanto, as maiores pontuações foram obtidas com o parâmetro $k_{Beam} = 5$ (por exemplo, Bleu1-4, Meteor, Cider e Rouge no conjunto de dados Flickr30k).

Dessa forma, selecionou-se as melhores pontuações obtidas em ambos os modelos para a realização das comparações de desempenho entre as abordagens nos conjuntos de dados Flickr30k e MS-COCO. Assim, a Figura 6.2 apresenta as comparações entre as pontuações obtidas no conjunto de dados Flickr30k entre os modelos baseados na geração

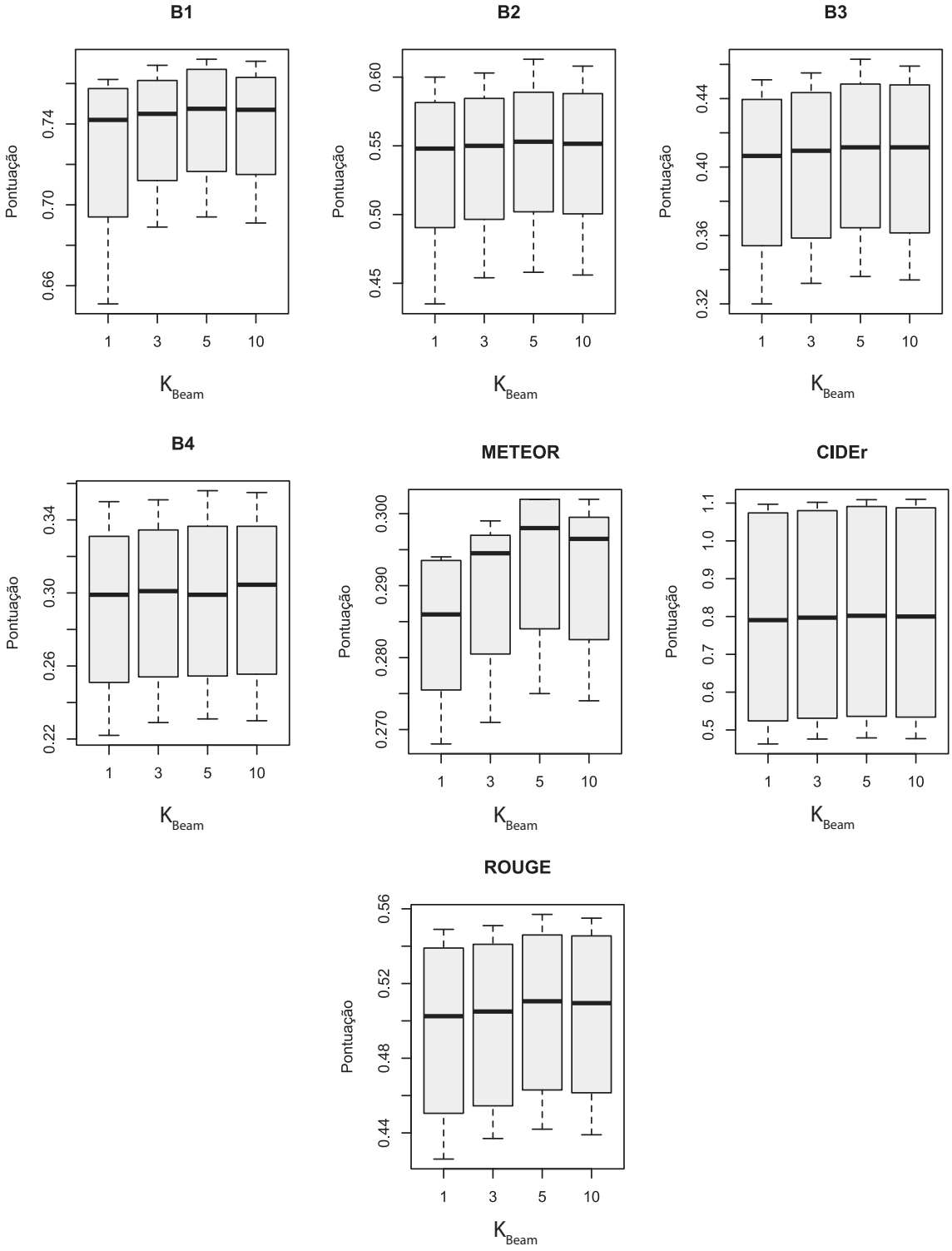


Figura 6.1: *Média de pontuações obtidos pelo k_{Beam} entre os modelos r-GRU-Inception e r-GRU-VGG, nos conjuntos de dados Flickr30k e MS-COCO.*

de pesos por meio da função de verossimilhança - r-GRU-Inception (verossimilhança); e por meio da análise da PoS - r-GRU-Inception (PoS).

Observa-se que, o modelo r-GRU-Inception (PoS) alcançou pontuações melho-

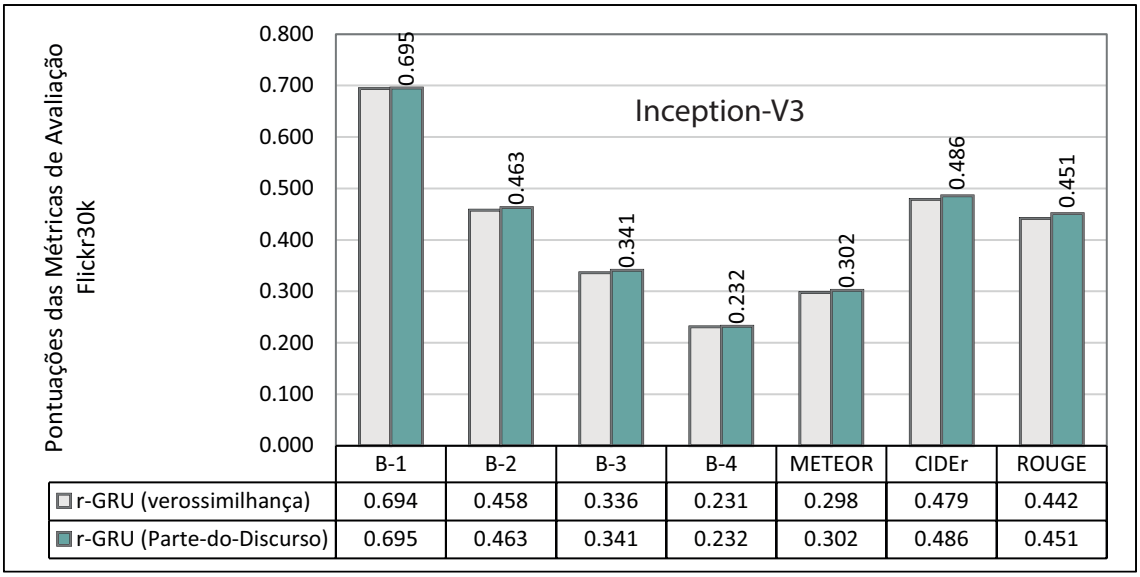


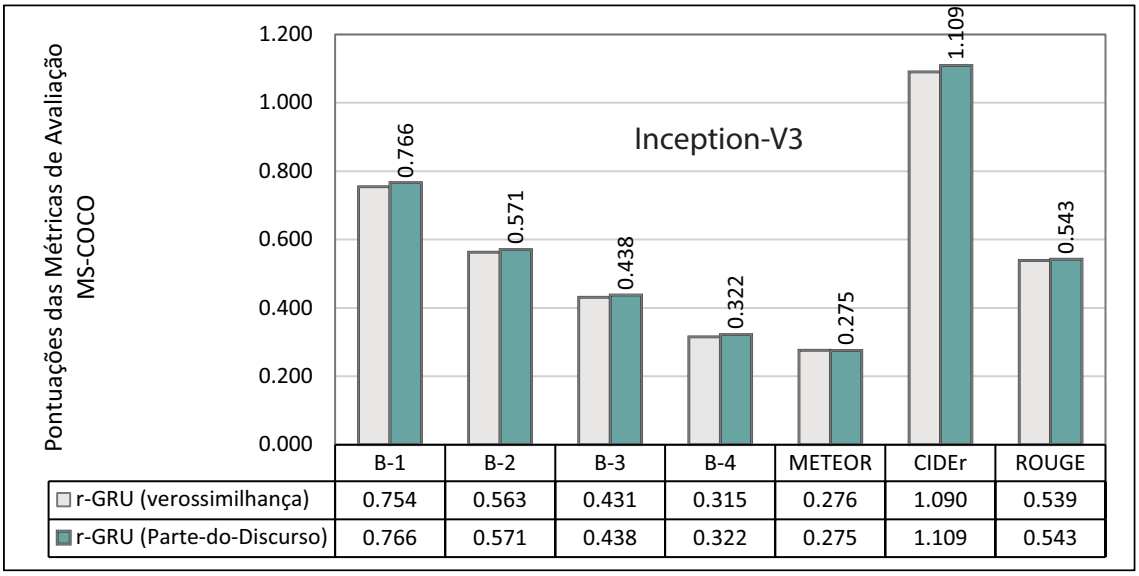
Figura 6.2: *Resultados das avaliações de desempenho dos modelos r-GRU-Inception (verossimilhança) e r-GRU-Inception (PoS) no conjunto de dados Flickr30k.*

res em todas as métricas no conjunto de dados Flickr30k quando comparadas aos resultados obtidos no r-GRU-Inception (verossimilhança). Dessa forma, obtendo as pontuações Bleu-1 (0.695), Bleu-2 (0.463), Bleu-3 (0.341), Bleu-4 (0.232), Meteor (0.302), Cider (0.486) e Rouge (0.451). Enquanto que as pontuações obtidas pelo modelo r-GRU-Inception (verossimilhança) foram: Bleu-1 (0.694), Bleu-2 (0.458), Bleu-3 (0.336), Bleu-4 (0.231), Meteor (0.298), Cider (0.479) e Rouge (0.442).

Portanto, todas as métricas alcançadas pelo modelo com verossimilhança obtiveram pontuações menores quando comparadas aos resultados obtidos pelo o modelo baseado na PoS. Entretanto, ambos os modelos obtiveram boas pontuações com a largura do feixe para geração de sentenças igual a 5, quando comparados com os demais valores para o mesmo parâmetro. Ou seja, as pontuações obtidas com o parâmetro $k_{Beam} = 5$ foram superiores aos demais, em ambos os modelos.

Analisou-se também, o desempenho dos dois modelos no conjunto de dados MS-COCO. Assim, a Figura 6.3 apresenta as comparações das pontuações obtidas no conjunto de dados MS-COCO entre os modelos r-GRU-Inception (verossimilhança) e r-GRU-Inception (PoS).

Observa-se que, o modelo r-GRU-Inception (PoS) obteve resultados maiores em quase todas as métricas no MS-COCO, exceto na pontuação alcançada pela métrica Meteor, obtendo uma pontuação de 0.275, enquanto que a r-GRU-Inception (verossimilhança) obteve uma pontuação Meteor (0.276) e Rouge (0.539). A r-GRU-Inception (PoS) alcançou as seguintes pontuações: Bleu-1 (0.766), Bleu-2 (0.571), Bleu-3 (0.438), Bleu-4 (0.322) e Cider (1.109). O modelo r-GRU-Inception (verossimilhança) obteve as pontuações Bleu-1 (0.754), Bleu-2 (0.563), Bleu-3 (0.431), Bleu-4 (0.315) e Cider (1.090).



	B-1	B-2	B-3	B-4	METEOR	CIDEr	ROUGE
r-GRU (verossimilhança)	0.754	0.563	0.431	0.315	0.276	1.090	0.539
r-GRU (Parte-do-Discurso)	0.766	0.571	0.438	0.322	0.275	1.109	0.543

Figura 6.3: *Resultados das avaliações de desempenho dos modelos r-GRU-Inception (verossimilhança) e r-GRU-Inception (PoS) no conjunto de dados MS-COCO.*

Estes resultados obtidos pelos experimentos corroboram que, o modelo proposto baseada na PoS possui resultados maiores quando comparados aos resultados obtidos pelo modelo de verossimilhança nos dois conjuntos de dados (Flickr30k e MS-COCO) (Figura 6.4).

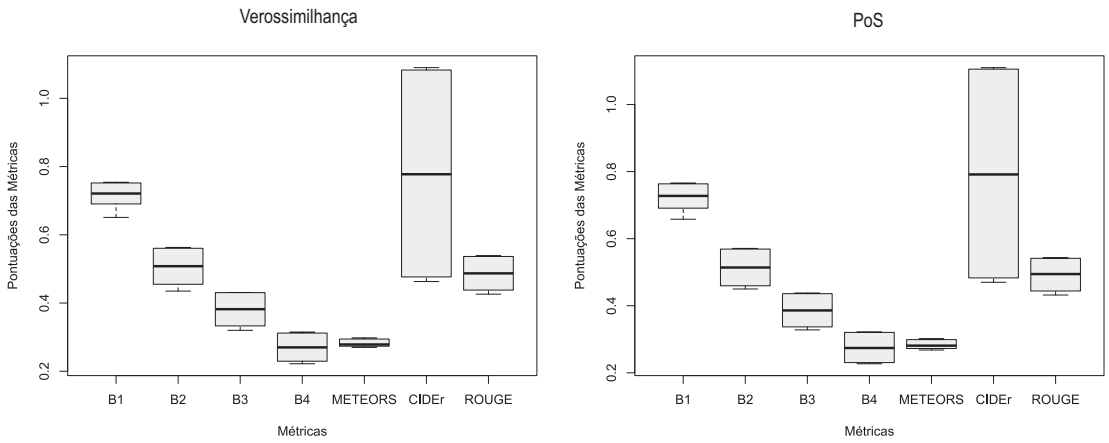


Figura 6.4: *Resultados das avaliações de desempenho dos modelos r-GRU-Inception.*

6.2 Estudo de Ablação do Modelo Baseado na Arquitetura VGG

Similar à abordagem anterior, para a realização do estudo de ablação na arquitetura VGG, configurou-se os parâmetros k_{Beam} e α, identificando a influência da largura

do feixe e o peso do modelo de linguagem nas pontuações obtidas entre os dois modelos. Percebeu-se também que o peso do modelo de linguagem não influenciou de forma significativa nas pontuações entre os modelos r-GRU-VGG (verossimilhança) e r-GRU-VGG (PoS). Dessa forma, a Tabela 6.2 apresenta as pontuações obtidas em ambos os modelos baseados na arquitetura VGG no conjunto de dados Flickr30k e MS-COCO.

Tabela 6.2: *Estudo de ablação com parâmetro k_{Beam} no conjunto de dados Flickr30k e MS-COCO, entre os modelos r-GRU-VGG (verossimilhança) e r-GRU-VGG (PoS).*

Flickr30k								
Métricas	$k_{Beam} = 1$		$k_{Beam} = 3$		$k_{Beam} = 5$		$k_{Beam} = 10$	
	Verossimilhança	**PoS**	**Verossimilhança**	**PoS**	**Verossimilhança**	**PoS**	**Verossimilhança**	**PoS**
B-1	0.730	0.736	0.734	0.739	0.738	**0.741**	0.737	**0.741**
B-2	0.531	0.538	0.535	0.540	0.541	**0.543**	0.540	0.542
B-3	0.380	0.385	0.381	0.389	0.388	**0.392**	0.385	**0.392**
B-4	0.289	0.275	0.291	0.278	0.277	**0.283**	0.296	0.280
METEOR	0.294	0.293	0.296	0.295	0.298	**0.302**	0.298	0.299
CIDEr	0.578	0.580	0.580	0.584	0.586	**0.591**	0.584	0.588
ROUGE	0.469	0.473	0.471	0.476	0.475	**0.482**	0.473	0.480
MS-COCO								
Métricas	$k_{Beam} = 1$		$k_{Beam} = 3$		$k_{Beam} = 5$		$k_{Beam} = 10$	
	Verossimilhança	**PoS**	**Verossimilhança**	**PoS**	**Verossimilhança**	**PoS**	**Verossimilhança**	**PoS**
B-1	0.755	0.762	0.760	0.769	0.768	**0.772**	0.762	0.771
B-2	0.598	0.600	0.601	0.603	0.607	**0.613**	0.606	0.608
B-3	0.448	0.451	0.453	0.455	0.459	**0.463**	0.458	0.459
B-4	0.346	0.350	0.350	0.351	0.351	**0.356**	0.351	0.355
METEOR	0.285	0.294	0.288	0.299	0.292	**0.302**	0.291	**0.302**
CIDEr	1.001	1.072	1.010	1.079	1.013	**1.092**	1.012	1.090
ROUGE	0.540	0.549	0.542	0.551	0.549	**0.557**	0.548	0.555

Nota-se que, o modelo r-GRU-VGG (PoS) alcançou pontuações melhores em todas as métricas no conjunto de dados Flickr30k e MS-COCO quando comparadas aos resultados obtidos com o modelo r-GRU-VGG (verossimilhança). No entanto, os dois modelos têm bons resultados com a largura do feixe para geração de sentenças igual a 5, isto é, com o parâmetro $k_{Beam} = 5$.

Nesta abordagem, também é constado que, houveram diferenças significativas nas pontuações dos dois modelos entre os valores inferidos para a largura do feixe para geração de sentenças.

Não obstante, selecionou-se as melhores pontuações a fim de compará-las em ambas as abordagens. Dessa forma, a Figura 6.5 apresenta as comparações entre as pontuações obtidas no conjunto de dados Flickr30k entre os modelos r-GRU-VGG (verossimilhança) e r-GRU-VGG (PoS).

Percebeu-se que, o modelo r-GRU-VGG (PoS), alcançou pontuações melhores em todas as métricas no conjunto de dados Flickr30k quando comparadas aos resultados obtidos no r-GRU-VGG (verossimilhança), obtendo assim, as pontuações: Bleu-1 (0.741), Bleu-2 (0.543), Bleu-3 (0.392), Bleu-4 (0.283), Meteor (0.302), Cider (0.591) e Rouge (0.482). Enquanto que o modelo r-GRU-VGG (verossimilhança) obteve pontuações Bleu-

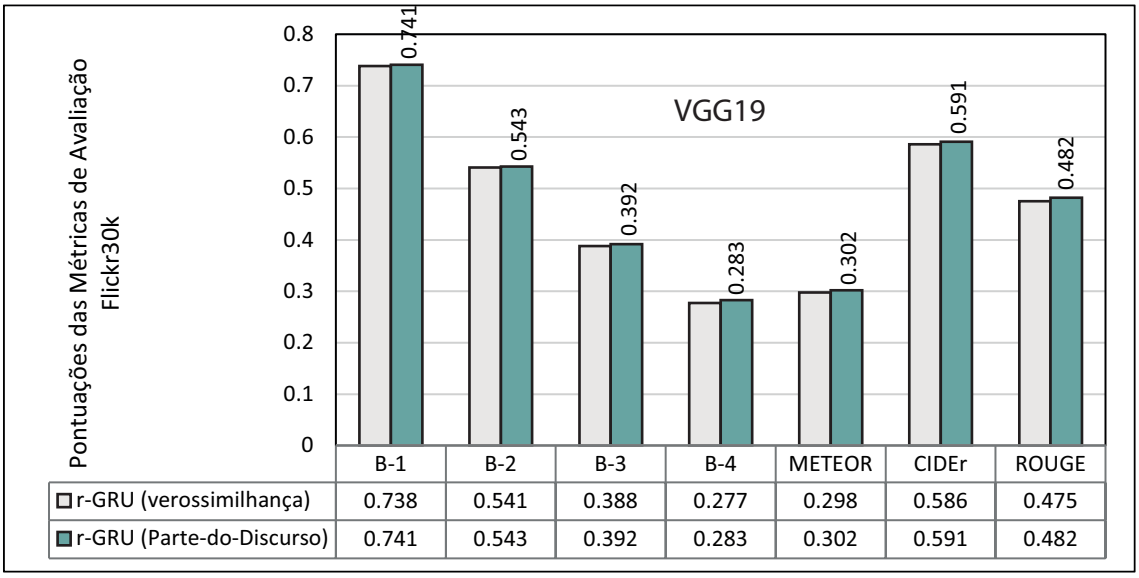


Figura 6.5: *Resultados das avaliações de desempenho dos modelos r-GRU-VGG (verossimilhança) e r-GRU-VGG (PoS) no conjunto de dados Flickr30k.*

1 (0.738), Bleu-2 (0.541), Bleu-3 (0.388), Bleu-4 (0.277), Meteor (0.298), Cider (0.586) e Rouge (0.475).

Apesar das pontuações resultantes do modelo r-GRU-VGG baseado em verossimilhança serem menores quando comparadas as pontuações do modelo baseado na PoS, ambos os modelos obtiveram bons resultados quando a largura do feixe para geração de sentenças foi igual a 5, ou seja, as pontuações obtidas com o parâmetro $k_{Beam} = 5$ foram superiores aos demais, em ambos os modelos. Essas pontuações corroboram com os resultados obtidos pela abordagem baseada na arquitetura anterior (Inception).

Analisando os resultados do desempenho dos modelos no conjunto de dados MS-COCO (Figura 6.6), percebe-se que, os resultados seguiram o mesmo padrão dos resultados obtidos no conjunto de dados Flickr30k. Ou seja, o modelo baseado na PoS utilizando a arquitetura VGG19 obteve resultados melhores quando comparado ao modelo baseado em verossimilhança. Dessa forma, o modelo r-GRU-VGG (PoS) obteve as seguintes pontuações: Bleu-1 (0.772), Bleu-2 (0.613), Bleu-3 (0.463), Bleu-4 (0.356), Meteor (0.302), Cider (1.092) e Rouge (0.557). Enquanto que o modelo r-GRU-VGG (verossimilhança) obteve pontuações Bleu-1 (0.768), Bleu-2 (0.607), Bleu-3 (0.459), Bleu-4 (0.351), Meteor (0.292), Cider (1.013) e Rouge (0.549).

Portanto, por meio dos estudos de ablações, percebe-se que, os modelos propostos pontuam melhor quando a largura do feixe para legendagem de imagens recebe o valor 5, obtendo pontuações superiores em relação aos demais valores neste mesmo parâmetro. Outra observação importante é que o modelo baseado na arquitetura VGG obteve pontuações melhores quando comparadas as demais abordagens, tanto no modelo de verossimilhança quando no modelo da PoS (Figura 6.8).

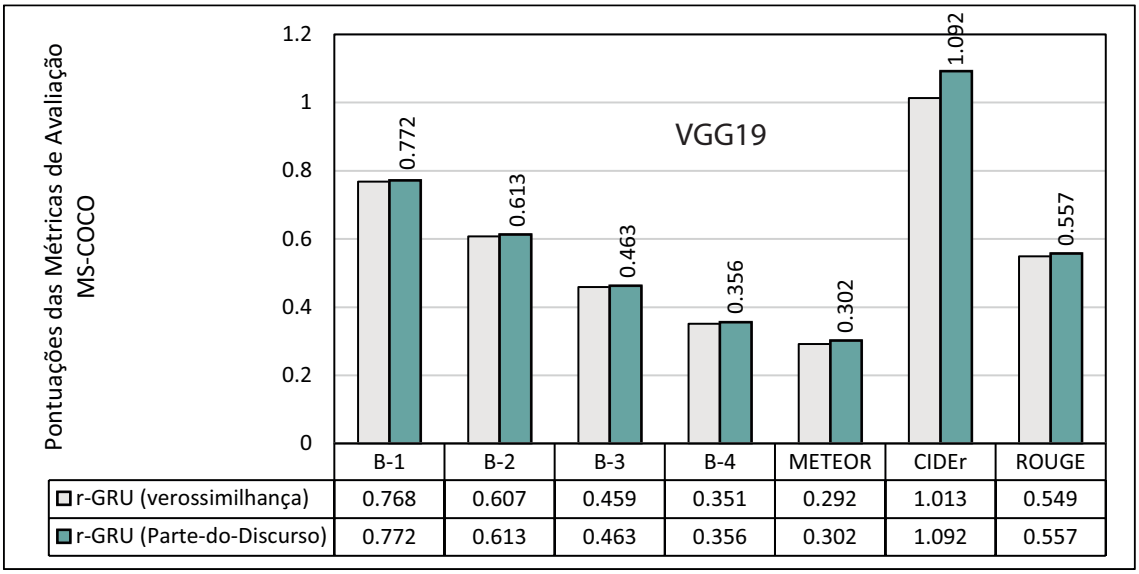


Figura 6.6: *Resultados das avaliações de desempenho dos modelos r-GRU-VGG (verossimilhança) e r-GRU-VGG (PoS) no conjunto de dados MS-COCO.*

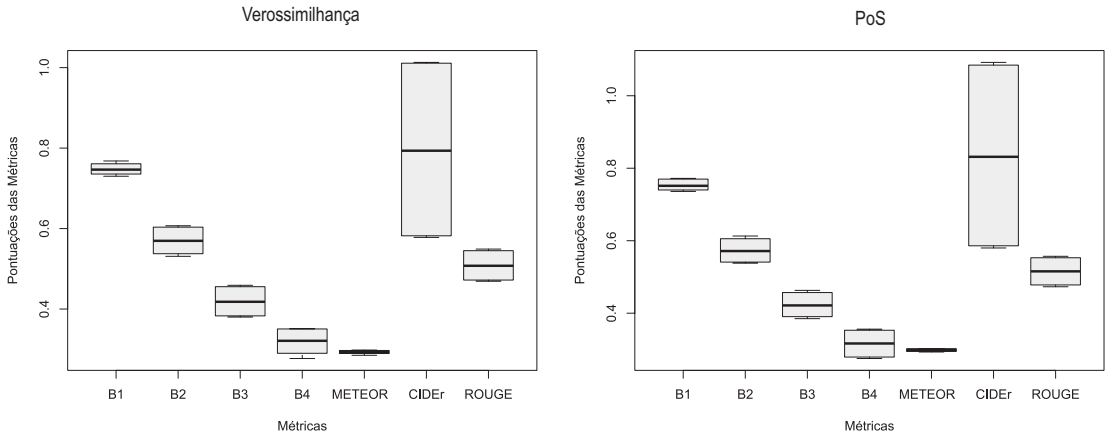


Figura 6.7: *Resultados das avaliações de desempenho dos modelos r-GRU-VGG.*

Dessa forma, para realização de *benchmark*, selecionou-se este modelo a fim de comparar estes resultados com as pontuações obtidas pelos modelos de ponta. Assim, na seção a seguir serão realizadas as comparações entre o modelo r-GRU-VGG e os demais modelos selecionados para os *benchmarks*.

6.3 Estudo de Benchmarks com Modelos de Ponta

Para a realização deste estudo de benchmarks, selecionaram-se os modelos de ponta phi-LSTM [166], He et al. [64], Ding et al. [45], 3G [188], m-GRU [105] e Google-NIC [172]. Dessa forma, comparamos os melhores resultados do nosso modelo r-

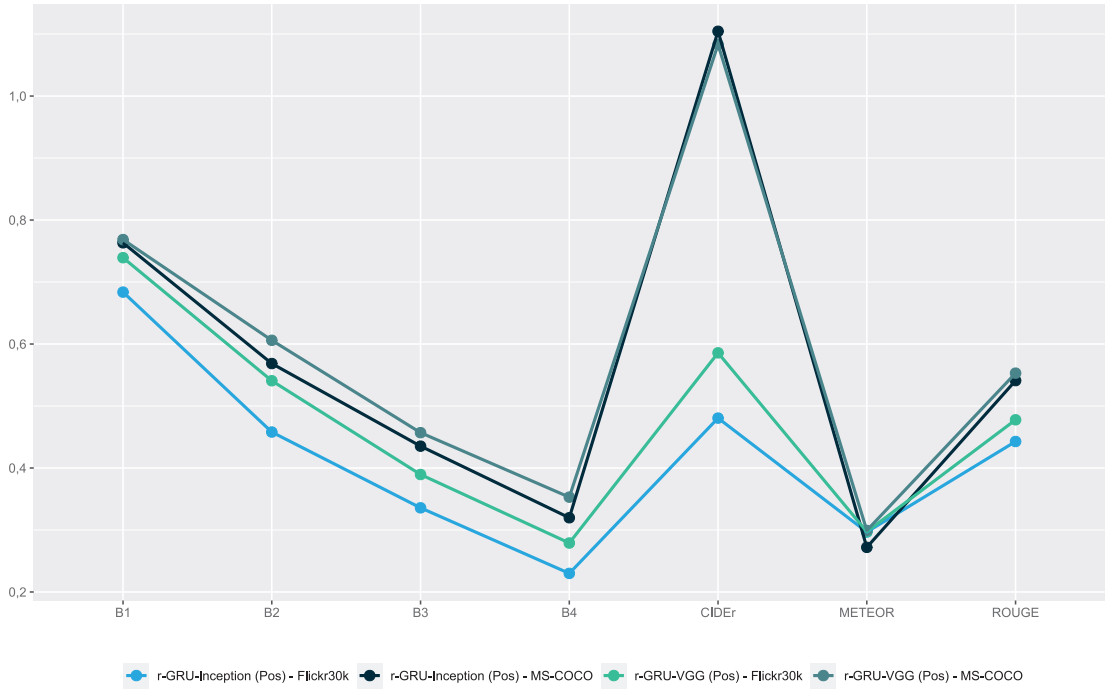


Figura 6.8: *Melhores pontuações entre os modelos r-GRU-Inception e r-GRU-VGG, nos conjuntos de dados Flickr30k e MS-COCO.*

GRU com os modelos do estado da arte. Como mostrado na seção anterior, o modelo baseado na arquitetura VGG19 obteve pontuações superiores quando comparadas aos resultados obtidos pelo modelo baseado na arquitetura Inception. Assim, a Tabela 6.3 apresenta o desempenho do modelo r-GRU-VGG no conjunto de dados Flickr30k, nas duas abordagens, tanto no modelo de geração de pesos baseado em verossimilhança quanto na PoS.

Tabela 6.3: *Desempenho do modelo r-GRU-VGG comparado com outros modelos de ponta no conjunto de dados Flickr30k. Valores em negrito representam as melhores pontuações. Valores em vermelho representam as melhores pontuações dos modelos do estado-da-arte.*

Flickr30k - VGG19							
Modelo	**BLEU**				**METEOR**	**CIDEr**	**ROUGE**
	B-1	**B-2**	**B-3**	**B-4**			
phi-LSTM [166]	0.642	0.454	0.317	0.218	0.190	0.452	0.446
He at al. [64]	0.638	0.446	0.307	0.211			
Ding et al. [45]	0.623	0.427	0.282	0.181			
3G [188]	0.694	0.457	0.332	0.226	0.230		
m-GRU [105]	0.669	0.405	0.289	0.200	0.295	0.483	
NIC [172]	0.663	0.423	0.277	0.183			
r-GRU (verossimilhança)	**0.738**	**0.541**	**0.388**	**0.277**	**0.298**	**0.586**	**0.475**
r-GRU (PoS)	**0.741**	**0.543**	**0.392**	**0.283**	**0.302**	**0.591**	**0.482**

Analisando o desempenho do modelo r-GRU-VGG em relação aos demais modelos de ponta no conjunto de dados Flickr30k, percebe-se que, o modelo baseado na

PoS supera os demais modelos de ponta selecionados para estes benchmarks. Observa-se também que, em todos os casos, o modelo baseado em verossimilhança alcança pontuações superiores em relação aos demais modelos.

Não obstante, o modelo aqui proposto supera em alguns casos até 25% (vinte e cinco por cento) em relação aos melhores resultados entre os modelos de ponta. Portanto, o modelo r-GRU (PoS) obteve uma pontuação Bleu-4 (0.283) maior quando comparada ao modelo de ponta 3G (0.226) no conjunto de dados Flickr30k. Dessa forma, obtendo um aumento de 25,22% (vinte e cinco vírgula vinte e dois por cento) na pontuação Bleu-4 em relação ao resultado obtido pelo modelo 3G.

Os demais percentuais de aumento de pontuações alcançados pelo modelo r-GRU no conjunto de dados Flickr30k foram: Bleu-1 (6,77%), Bleu-2 (18,81%) e Bleu-3 (18,07%). Observa-se também que, o modelo r-GRU baseado na PoS obteve aumento nas pontuações Meteor (2,37%) e Cider (22,36%) em relação ao modelo m-GRU [105].

Tabela 6.4: *Desempenho do modelo r-GRU-VGG comparado com outros modelos de ponta no conjunto de dados MS-COCO. Valores em negrito representam as melhores pontuações. Valores em vermelho representam as melhores pontuações dos modelos do estado-da-arte.*

MS-COCO - VGG19							
Modelo	**BLEU**				**METEOR**	**CIDEr**	**ROUGE**
	B-1	**B-2**	**B-3**	**B-4**			
phi-LSTM [166]	0.695	0.523	0.385	0.282	0.243	0.905	0.517
He at al. [64]	0.711	0.535	0.388	0.279	0.239	0.882	–
Ding et al. [45]	0.764	0.563	0.436	0.317	0.265	**1.103**	0.535
3G [188]	0.719	0.529	0.387	0.284	0.243	–	–
m-GRU [105]	0.668	0.459	0.331	0.229	0.255	0.733	–
NIC [5]	0.640	0.457	0.327	0.239	0.211	0.763	–
r-GRU (verossimilhança)	**0.768**	**0.607**	**0.459**	**0.351**	**0.292**	1.013	**0.549**
r-GRU (PoS)	**0.772**	**0.613**	**0.463**	**0.356**	**0.302**	**1.092**	**0.557**

Analisando o desempenho do modelo r-GRU-VGG em relação aos demais modelos de ponta no conjunto de dados MS-COCO (Tabela 6.4), percebe-se também que, o modelo baseado na PoS supera os demais modelos de ponta selecionados para estes benchmarks, com exceção da métrica de pontuação Cider. Observa-se que, em quase todos os casos, o modelo baseado em verossimilhança também alcança pontuações superiores em relação aos demais modelos.

Considerando os ganhos percentuais em relação aos demais modelos no conjunto de dados MS-COCO, o modelo r-GRU-VGG obteve aumento percentuais em relação ao melhor modelo de ponta seleciona para este benchmarks, nas seguintes pontuações: Bleu-1(1,04%), Bleu-2 (8,88%), Bleu-3 (6,19%), Bleu-4 (12,30%), Meteor (13,96%) e Rouge (7,73%).

Estes resultados corroboram que, o modelo r-GRU (PoS) supera em alguns casos até 13% (treze por cento) em relação aos melhores resultados entre os modelos de ponta.

Assim, o modelo baseado na PoS demonstra resultados promissores quando comparados aos modelos do estado da arte.

Uma observação importante em relação a comparação estrutural do modelo r-GRU-VGG e o modelo que apresentou o segundo melhor resultado [45], é a aplicação da Fast-RCNN na extração dos recursos da imagem; e a função de cálculos das matrizes de pesos que alimenta o decodificador. Em nossa abordagem, utilizamos uma técnica mais simples e sofisticada para geração das matrizes de pesos, gerando pesos diferentes de acordo com a importância de cada palavra para formação da sentença.

Dessa forma, modelos que utilizam a PoS para geração das matrizes de pesos nas redes recorrentes podem alcançar melhores resultados quando comparadas aos modelos que utilizam outras formas, como por exemplo, a distância euclidiana. Outra característica que a nossa abordagem se destaca em relação a Fast-RCNN está relacionada ao decodificador de palavras. Neste sentido, a nossa abordagem utiliza um decodificar (GRU). Este decodificador possui uma menor complexidade de implementação quando comparada ao Fast-RCNN. De acordo com Li et al. [105], o decodificador GRU é tão bom quanto a LSTM para memória de longo prazo, além do mais, modelos baseados em GRUs são bem mais simples, tornando estes modelos mais fáceis de treinar, sendo um fator importante para dados em grande escala.

6.4 Análises Aprofundadas das Legendas de Imagens Geradas Pelos Modelos

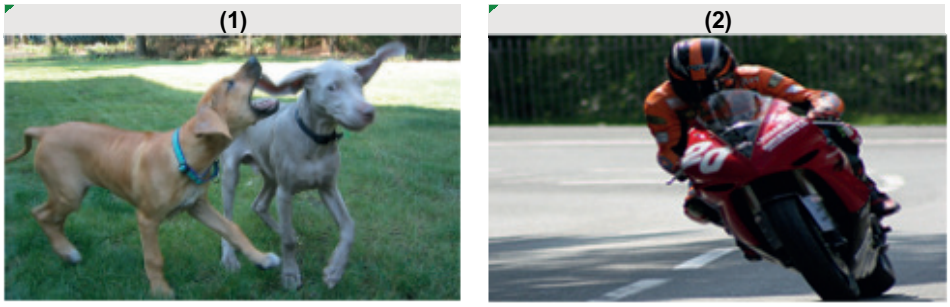
Na fase de validação, o modelo r-GRU é capaz de gerar três sentenças descritivas para uma determina imagem. Por exemplo, dada uma imagem alvo I, o modelo neural gera uma sequência de sentenças.

$$\mathbf{S}(\mathbf{I}) = \left\{S_{pred}, S_{ref}, S_{ling}\right\} \tag{6-1}$$

onde S_{pred} representa a sentença de predição gerada pelo modelo, S_{ref} a melhor legenda selecionada pela knn (similaridade entre as características da imagem de entrada e as imagens de treinamento) e S_{ling} uma legenda fornecida por uma especialista humano para a imagem de entrada. Assim, a Figura 6.9 apresenta exemplos de legendas geradas pelo modelo r-GRU com suas respectivas pontuações (Meteor [11]).

Para a realização das análises profundas das legendas, selecionou-se três grupos de imagens com as melhores pontuações identificadas durante as validações.

Sendo assim, percebe-se que, no primeiro conjunto de imagens, o modelo r-GRU gerou também três sentenças para cada imagem de entrada (I), gerando as pontuações comparativas entre a imagem de predição e de referência com a legenda de entrada do



S_{pred}	Two dogs are playing on the green grass; the brown dog is trying to to biting the white dog's ear (0.504).	A man dressed in orange clothes riding on a red motorcycle, making a right sharp turn (0.715).
S_{ref}	Two dogs are playing in a grassy field; one dog is biting the ear of the other dog (0.561).	A motorcycle driver dressed in orange gear swerves to the right (0.340).
S_{ling}	A brown and a white dog are playing on the green grass; the brown dog is biting the ear of the white dog.	A man dressed in orange clothes driving on a red motorcycle.

S_{pred}	A group of people made up of three men and two women are facing the beach on a sunny day (0.833).	A group of race cars are competing against each other on a dirt track; the green car leads the race (0.839).
S_{ref}	Three men and two women stand facing the ocean from the shore on a sunny day (0.496).	Race cars are racing each other on a dirt racetrack and the green one is in the lead (0.369).
S_{ling}	A group of people composed of three men and two women are facing the beach.	A group of race cars is competing against each other; the green car is leading the race.

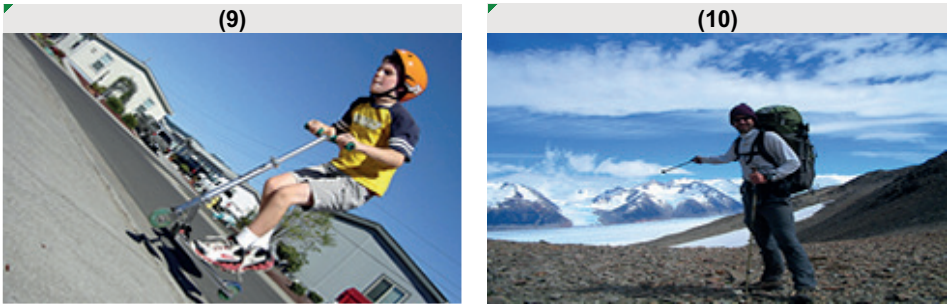
Figura 6.9: *Análise profunda das pontuações obtidas pelas legendas de predição e por referências geradas pelo modelo r-GRU – Conjunto de exemplares (1) com quatro imagens de entrada $S(I_0)$, $S(I_1)$, $S(I_2)$ e $S(I_3)$.*

sistema (fornecida por um especialista linguístico) (Figura 6.10). Assim, percebe-se que, neste primeiro conjunto de exemplares avaliado pelo modelo, as legendas de predição geraram pontuações maiores quando comparadas com as legendas de referência. Dessa forma, o modelo r-GRU obteve as seguintes pontuações: imagem 2 (0.715); imagem 3

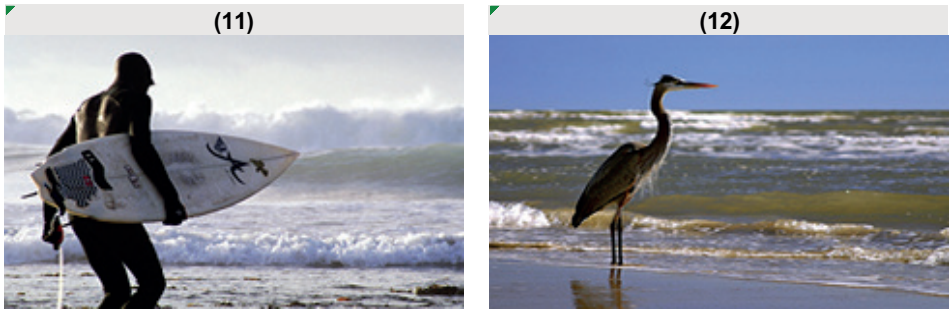
(0.833); e imagem 4 (0.839).

Figura 6.10: *Análise profunda das pontuações obtidas pelas legendas de predição e por referências geradas pelo modelo r-GRU – Conjunto de exemplares (2) com quatro imagens de entrada $S(I_0)$, $S(I_1)$, $S(I_2)$ e $S(I_3)$.*

Observa-se que, neste primeiro conjunto de exemplares de imagens de validação, há ocorrência de percentuais significativos entre as legendas geradas pelo modelo. Sendo assim, as legendas preditivas, em alguns casos chegaram a ser 127% (cento e vinte e sete



S_{pred}	A boy in gray shorts with a yellow and blue shirt riding a scooter on the street in a neighborhood (0.661).	A backpacker in the mountains as he stands using his hiking stick to point at a glacier (0.435).
S_{ref}	A small child rides down the sidewalk on a scooter in a neighborhood (0.286).	A hiker with a backpack is walking with a snow-covered mountain in the background (0.229).
S_{ling}	A boy in a yellow and blue shirt, gray shorts, riding a scooter on the street.	A backpacker with a green backpack pointing a hiking stick at the glaciers.



S_{pred}	A surfer in a dark wetsuit carries a white surfboard towards the ocean into the waves (0.383).	A tall gray bird walks along the beach as the waves roll (0.988).
S_{ref}	A Person In a black wetsuit carrying a surfboard into the waves (0.339).	A water large bird standing at the ocean 's edge (0.138).
S_{ling}	A surfer man carrying a board on the beach.	A tall bird walks along the beach as the waves roll.

Figura 6.11: *Análise profunda das pontuações obtidas pelas legendas de predição e por referências geradas pelo modelo r-GRU – Conjunto de exemplares (3) com quatro imagens de entrada $S(I_0)$, $S(I_1)$, $S(I_2)$ e $S(I_3)$.*

por cento) maior quando comparado com a pontuação da legenda de referência para a mesma imagem de entrada.

Apesar do modelo r-GRU apresentar uma pontuação menor na imagem 1, observa-se que a sentença preditiva gerada pelo modelo descreve a imagem de forma mais detalhada. Isso ocorre pelo fato de que este modelo baseado na PoS geram os pesos na fase de treinamento considerando a importância de cada palavra para formação das frases das imagens. Portanto, gerando sentenças mais descritivas quando comparadas às sentenças de referência. A Figura 6.9 apresenta os resultados das análises profundas das legendas geradas pelo modelo r-GRU com suas respectivas pontuações. Assim, observa-se que, neste conjunto de exemplares as legendas de predições obtiveram uma média de pontuações maiores em todas as imagens de entrada.

Dessa forma, o modelo de predição do modelo r-GRU alcançou uma pontuação média maior quando comparadas as legendas de referências (Tabela 6.5).

Tabela 6.5: *Média das Pontuações obtidas pelas legendas de predição e por referência geradas pelo modelo r-GRU.*

Legendas	N	Média	E.P.[1]	1º Q[2]	2º Q[3]	3º Q[4]	Valor-p[5]
Predição	12.00	0.72	0.06	0.49	0.77	0.90	<0.001
Referência		0.37	0.06	0.21	0.34	0.53	

Os resultados das análises profundas das legendas geradas pelo modelo r-GRU no terceiro conjunto de exemplares corroboram com as análises anteriores (Figura 6.10). Dessa forma, as legendas de predição geradas pelo modelo r-GRU foram: imagem 9 (0.661); imagem 10 (0.435); imagem 11 (0.383); e imagem 12 (0.988). Assim, percebe-se um aumento em média de 110% (cento e dez por cento) nestas pontuações quando comparadas as pontuações obtidas pelas legendas de referências.

Observou-se durante as análises das validações que, quanto maior eram os números de objetos contidos nas cenas das imagens, maiores eram os desafios para descrevê-las. No entanto, os resultados demonstraram que o modelo r-GRU se adaptou bem à esse desafio, gerando descrições longas e mais descritivas, os quais descrevem de forma simples a correlação entre os objetos das cenas por meio de sentenças naturais. Percebe-se também que, tanto as legendas de predição quanto as legendas de referência se aproximam das legendas de entrada do sistema.

De modo geral, na fase de validação, o modelo r-GRU gera legendas mais detalhadas, as quais se aproximam das legendas esperadas, alcançando pontuações significativas nas duas legendas, tanto de predição quanto a legenda fornecida pela transferência de conhecimento, neste caso, a legenda selecionada pela *k*NN.

[1]Medida da precisão da média amostral. O erro padrão é obtido dividindo o desvio padrão pela raiz quadrada do tamanho da amostra.

[2]Medida de posição que representa que pelo menos 25% das respostas são menores que ele.

[3]Medida de posição que representa que pelo menos 50% das respostas são menores que ele.

[4]Medida de posição que representa que pelo menos 75% das respostas são menores que ele.

[5]Uma p-valor menor que 0.05, gera evidências para rejeição da hipótese nula do teste.

Portanto, o modelo pode ser utilizado para verificação e identificação das similaridades entre as legendas de predição e de referência, sendo uma técnica auxiliadora no processo de seleção das melhores legendas para uma determinada imagem de entrada.

CAPÍTULO 7

Conclusões

Este trabalho propôs um modelo codificador-decodificador baseado na análise da PoS e na função de verossimilhança, gerando diferentes pesos para o treinamento da rede recorrente multimodal. O modelo proposto obteve resultados promissores quando comparados aos demais modelos de ponta nos dois conjuntos de dados, no Flickr30k e MS-COCO. Além da aplicação dos pesos na fase de pré-processamento, o modelo utiliza uma técnica de transferência de conhecimento pela *k*NN, selecionando as melhores descrições na fase de validação.

Dessa forma, os resultados demonstram que a aplicação da PoS, gerando pesos significativos para diferentes classes de palavras, obtém um desempenho melhor do que o modelo que utiliza a função de verossimilhança, em ambos os conjuntos de dados. No entanto, os dois modelos alcançaram resultados melhores quando comparados aos modelos do estado da arte, gerando legendas mais descritivas, próximas às legendas esperadas, obtendo pontuações significativas, tanto nas legendas fornecidas pela transferência de conhecimento quanto nas de predição.

Assim, o modelo proposto corrobora para a melhoria na legendagem de imagens de forma automática, possuindo várias aplicabilidades, como, por exemplo, nas descrições de imagens médicas, para o diagnóstico precoce de doenças e descrições de objetos em cenas, para a locomoção de pessoas cegas em ambientes desconhecidos.

Portanto, a aplicação de técnicas capazes de inferir diferentes pesos, levando em consideração a importância de cada palavra para a formação da legenda, pode alcançar resultados significativos, sendo um caminho promissor para os modelos de legendagem de imagens. Não obstante, usar-se de técnicas de transferência de conhecimento, buscando legendas das imagens mais próximas pode ser uma alternativa significativa, permitindo realizar-se comparações com as legendas preditas, auxiliando nas seleções das legendas mais descritivas para as imagens.

7.1 Contribuições e Trabalhos Futuros

Uma das vantagens do modelo proposto em relação às outras abordagens é a simplicidade da implementação do decodificador multimodal e a capacidade de interconectar as frases de referência às imagens de entrada, facilitando a comparação entre as legendas de previsão e as legendas de referências fornecidas pelos modelos. Assim, as principais contribuições deste trabalho foram:

- A utilização da PoS e da função de verossimilhança para treinar um gerador de legenda de imagem, permitindo a interligação entre as frases de linguagem natural aos conteúdos de imagens mais próximas, considerando diferentes escores, de acordo com as classes gramaticais das palavras;

- A aplicação da GRU na geração de legendas por referência, utilizando o método *k*NN, comparando as sentenças geradas pela GRU com as sentenças do conjunto de treinamento pelas métricas de avaliação.

Não obstante, como possíveis trabalhos futuros, destacam-se a:

- Realização de um estudo de ablação, ampliando os valores inferidos nos parâmetros da largura do feixe (K_{beam}) da heurística Beam Search utilizada pelo modelo r-GRU, baseado na arquitetura Inception-V3 e VGG19;

- Comparação dos resultados obtidos pelo modelo r-GRU, que possui a introdução do módulo de geração de pesos PoS e a verossimilhança com um modelo r-GRU puro (sem a introdução do módulo de geração de pesos);

- Alteração dos parâmetros no módulo de transferência de conhecimento no processo de geração de legendas por referências, inferindo diferentes valores para o *k*NN;

- Aplicação do modelo r-GRU nas descrições de objetos em cenas em tempo real, para auxiliar a locomoção de pessoas cegas em ambientes desconhecidos;

- Inserção do modelo r-GRU em sistemas de legendagem de imagens para o diagnóstico precoce de doenças nas áreas da saúde e da agricultura.

Referências Bibliográficas

[1] AL-MUZAINI, H. A.; AL-YAHYA, T. N.; BENHIDOUR, H. **Automatic arabic image captioning using rnn-lstm-based language model and cnn**. *database*, 9(6), 2018.

[2] ALASMARI, A. **Toward quality multimorbid health questions for online information seeking in q&a sites**. In: *Proceedings of the 2020 Conference on Human Information Interaction and Retrieval*, p. 511–514, 2020.

[3] ALBAWI, S.; MOHAMMED, T. A.; AL-ZAWI, S. **Understanding of a convolutional neural network**. In: *2017 International Conference on Engineering and Technology (ICET)*, p. 1–6. IEEE, 2017.

[4] ALBU, A.; UNGUREANU, L. **Artificial neural network in medicine**. *Telemed. JE Health*, 18(6):446–453, 2012.

[5] ALPAYDIN, E. **Introduction to machine learning**. MIT press, 2020.

[6] ANEJA, J.; DESHPANDE, A.; SCHWING, A. G. **Convolutional image captioning**. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 5561–5570, 2018.

[7] ANTOL, S.; AGRAWAL, A.; LU, J.; MITCHELL, M.; BATRA, D.; LAWRENCE ZITNICK, C.; PARIKH, D. **Vqa: Visual question answering**. In: *Proceedings of the IEEE international conference on computer vision*, p. 2425–2433, 2015.

[8] ASHRAF, J.; IQBAL, N.; KHATTAK, N. S.; ZAIDI, A. M. **Speaker independent urdu speech recognition using hmm**. In: *2010 The 7th International Conference on Informatics and Systems (INFOS)*, p. 1–5. IEEE, 2010.

[9] BABAKHANI, P.; BRIDGE, J.; DOONG, R.-A.; PHENRAT, T. **Parameterization and prediction of nanoparticle transport in porous media: A reanalysis using artificial neural network**. *Water Resources Research*, 53(6):4564–4585, 2017.

[10] BAKSHI, R. K.; KAUR, N.; KAUR, R.; KAUR, G. **Opinion mining and sentiment analysis**. In: *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, p. 452–455. IEEE, 2016.

[11] BANERJEE, S.; LAVIE, A. **Meteor: An automatic metric for mt evaluation with improved correlation with human judgments**. In: *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, p. 65–72, 2005.

[12] BARRATT, S.; SHARMA, R. **A note on the inception score**. *arXiv preprint arXiv:1801.01973*, 2018.

[13] BENGIO, Y. **Practical recommendations for gradient-based training of deep architectures**. In: *Neural networks: Tricks of the trade*, p. 437–478. Springer, 2012.

[14] BENNETT, I. M. **Network based interactive speech recognition system**, Nov. 9 2010. US Patent 7,831,426.

[15] BENNETT, I. M.; BABU, B. R.; MORKHANDIKAR, K.; GURURAJ, P. **Distributed real time speech recognition system**, July 7 2015. US Patent 9,076,448.

[16] BJERVA, J. **One model to rule them all: Multitask and multilingual modelling for lexical analysis**. *arXiv preprint arXiv:1711.01100*, 2017.

[17] BJERVA, J. **Multitask and multilingual modelling for lexical analysis**. *KI-Künstliche Intelligenz*, 32(4):287–290, 2018.

[18] BOOIJ, O.; TAT NGUYEN, H. **A gradient descent rule for spiking neurons emitting multiple spikes**. *Information Processing Letters*, 95(6):552–558, 2005.

[19] BRITZ, D.; GOLDIE, A.; LUONG, M.-T.; LE, Q. **Massive exploration of neural machine translation architectures**. *arXiv preprint arXiv:1703.03906*, 2017.

[20] CALA, L. L. **Recognition and Tracking of Vehicles in Highways using Deep Learning**. PhD thesis, Universidade de São Paulo, 2018.

[21] CAMBRIA, E. **Affective computing and sentiment analysis**. *IEEE intelligent systems*, 31(2):102–107, 2016.

[22] CARVALHO, E.; FERREIRA, B.; FERREIRA, M.; PEREIRA, G.; UEYAMA, J.; PESSIN, G. **Uso de redes neurais recorrentes para localização de agentes em ambientes internos**. *Symposium on Knowledge Discovery, Mining and Learning*, 2012.

[23] CHANG, Y.-S. **Fine-grained attention for image caption generation**. *Multimedia Tools and Applications*, 77(3):2959–2971, 2018.

[24] CHAUVIN, Y.; RUMELHART, D. E. **Backpropagation: theory, architectures, and applications**. Psychology press, 1995.

[25] CHEN, J.; ZHUGE, H. **A news image captioning approach based on multimodal pointer-generator network**. *Concurrency and Computation: Practice and Experience*, 2020.

[26] CHEN, X.; FANG, H.; LIN, T.-Y.; VEDANTAM, R.; GUPTA, S.; DOLLÁR, P.; ZITNICK, C. L. **Microsoft coco captions: Data collection and evaluation server**. *arXiv preprint arXiv:1504.00325*, 2015.

[27] CHEN, Y.; YANG, X.; ZHONG, B.; PAN, S.; CHEN, D.; ZHANG, H. **Cnntracker: Online discriminative object tracking via deep convolutional neural network**. *Applied Soft Computing*, 38:1088–1098, 2016.

[28] CHO, K. **Natural language understanding with distributed representation**. *arXiv preprint arXiv:1511.07916*, 2015.

[29] CHO, K.; VAN MERRIËNBOER, B.; GULCEHRE, C.; BAHDANAU, D.; BOUGARES, F.; SCHWENK, H.; BENGIO, Y. **Learning phrase representations using rnn encoder-decoder for statistical machine translation**. *arXiv preprint arXiv:1406.1078*, 2014.

[30] CHUNG, J.; GULCEHRE, C.; CHO, K.; BENGIO, Y. **Empirical evaluation of gated recurrent neural networks on sequence modeling**. *arXiv preprint arXiv:1412.3555*, 2014.

[31] CHUNG, J.; GULCEHRE, C.; CHO, K.; BENGIO, Y. **Gated feedback recurrent neural networks**. In: *International Conference on Machine Learning*, p. 2067–2075, 2015.

[32] CHUNG, M. H.; YANG, Y. K.; LEE, K. H.; LEE, J. H.; MOON, J. W. **Application of artificial neural networks for determining energy-efficient operating set-points of the vrf cooling system**. *Building and Environment*, 125:77–87, 2017.

[33] COHEN, E.; BECK, C. **Empirical analysis of beam search performance degradation in neural sequence models**. In: *International Conference on Machine Learning*, p. 1290–1299, 2019.

[34] COLLOBERT, R.; WESTON, J.; BOTTOU, L.; KARLEN, M.; KAVUKCUOGLU, K.; KUKSA, P. **Natural language processing (almost) from scratch**. *Journal of machine learning research*, 12(Aug):2493–2537, 2011.

[35] COLLOVINI, S.; GOULART, R.; VIEIRA, R. **Identificação de expressões anafóricas e não anafóricas com base na estrutura do sintagma**. In: *2. º Workshop em Tecnologia da Informação e da Linguagem Humana (TIL 2004)–Salvador, BA*, 2004.

[36] CONG, I.; CHOI, S.; LUKIN, M. D. **Quantum convolutional neural networks**. *Nature Physics*, 15(12):1273–1278, 2019.

[37] COPPERSMITH, G.; LEARY, R.; CRUTCHLEY, P.; FINE, A. **Natural language processing of social media as screening for suicide risk**. *Biomedical informatics insights*, 10:1178222618792860, 2018.

[38] DA SILVA, I. N.; SPATTI, D. H.; FLAUZINO, R. A. **Redes neurais artificiais para engenharia e ciências aplicadas curso prático**. *São Paulo: Artliber*, 2010.

[39] DAUD, S. P.; RIBEIRO, C. H. C. **Nlp–lexical analysis applied to requirements**. In: *Proceedings of the 9th Brazilian Conference on Dynamics Control and their Applications Serra Negra, SP-ISSN*, p. 2178–3667, 2010.

[40] DENG, L.; LIU, Y. **Deep learning in natural language processing**. Springer, 2018.

[41] DESHPANDE, A.; ANEJA, J.; WANG, L.; SCHWING, A. G.; FORSYTH, D. **Fast, diverse and accurate image captioning guided by part-of-speech**. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 10695–10704, 2019.

[42] DEVLIN, J.; GUPTA, S.; GIRSHICK, R.; MITCHELL, M.; ZITNICK, C. L. **Exploring nearest neighbor approaches for image captioning**. *arXiv preprint arXiv:1505.04467*, 2015.

[43] DIEHL, P. U.; NEIL, D.; BINAS, J.; COOK, M.; LIU, S.-C.; PFEIFFER, M. **Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing**. In: *2015 International Joint Conference on Neural Networks (IJCNN)*, p. 1–8. ieee, 2015.

[44] DING, G.; CHEN, M.; ZHAO, S.; CHEN, H.; HAN, J.; LIU, Q. **Neural image caption generation with weighted training and reference**. *Cognitive Computation*, p. 1–15, 2018.

[45] DING, S.; QU, S.; XI, Y.; SANGAIAH, A. K.; WAN, S. **Image caption generation with high-level image features**. *Pattern Recognition Letters*, 123:89–95, 2019.

[46] DOGNIN, P.; MELNYK, I.; MROUEH, Y.; ROSS, J.; SERCU, T. **Improved adversarial image captioning**. *ICLR 2019 Workshop DeepGenStruct*, 2019.

[47] DONG, L.; YANG, N.; WANG, W.; WEI, F.; LIU, X.; WANG, Y.; GAO, J.; ZHOU, M.; HON, H.-W. **Unified language model pre-training for natural language understanding and generation**. In: *Advances in Neural Information Processing Systems*, p. 13063–13075, 2019.

[48] DUCHI, J.; HAZAN, E.; SINGER, Y. **Adaptive subgradient methods for online learning and stochastic optimization**. *Journal of machine learning research*, 12(Jul):2121–2159, 2011.

[49] ERKAYMAZ, O.; OZER, M.; PERC, M. **Performance of small-world feedforward neural networks for the diagnosis of diabetes**. *Applied Mathematics and Computation*, 311:22–28, 2017.

[50] FAKOOR, R.; MOHAMED, A.-R.; MITCHELL, M.; KANG, S. B.; KOHLI, P. **Memory-augmented attention modelling for videos**. *arXiv preprint arXiv:1611.02261*, 2016.

[51] FARZINDAR, A.; INKPEN, D. **Natural language processing for social media**. *Synthesis Lectures on Human Language Technologies*, 10(2):1–195, 2017.

[52] FRANK, E.; HALL, M.; HOLMES, G.; KIRKBY, R.; PFAHRINGER, B.; WITTEN, I. H.; TRIGG, L. **Weka-a machine learning workbench for data mining**. In: *Data mining and knowledge discovery handbook*, p. 1269–1277. Springer, 2009.

[53] GAL, Y.; GHAHRAMANI, Z. **Dropout as a bayesian approximation: Representing model uncertainty in deep learning**. In: *international conference on machine learning*, p. 1050–1059, 2016.

[54] GAO, L.; LI, X.; SONG, J.; SHEN, H. T. **Hierarchical lstms with adaptive attention for visual captioning**. *IEEE transactions on pattern analysis and machine intelligence*, 42(5):1112–1131, 2019.

[55] GAO, L.; WANG, B.; WANG, W. **Image captioning with scene-graph based semantic concepts**. In: *Proceedings of the 2018 10th International Conference on Machine Learning and Computing*, p. 225–229. ACM, 2018.

[56] GARDNER, M.; GRUS, J.; NEUMANN, M.; TAFJORD, O.; DASIGI, P.; LIU, N.; PETERS, M.; SCHMITZ, M.; ZETTLEMOYER, L. **Allennlp: A deep semantic natural language processing platform**. *arXiv preprint arXiv:1803.07640*, 2018.

[57] GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep learning**. MIT press, 2016.

[58] GU, J.; WANG, Z.; KUEN, J.; MA, L.; SHAHROUDY, A.; SHUAI, B.; LIU, T.; WANG, X.; WANG, G.; CAI, J.; OTHERS. **Recent advances in convolutional neural networks**. *Pattern Recognition*, 77:354–377, 2018.

[59] GUO, P.-C. **A frobenius norm regularization method for convolutional kernels to avoid unstable gradient problem**. *arXiv preprint arXiv:1907.11235*, 2019.

[60] GUO, X.; CHEN, L.; SHEN, C. **Hierarchical adaptive deep convolution neural network and its application to bearing fault diagnosis**. *Measurement*, 93:490–502, 2016.

[61] HARLEY, A. W. **An interactive node-link visualization of convolutional neural networks**. In: *International Symposium on Visual Computing*, p. 867–877. Springer, 2015.

[62] HE, C.; HU, H. **Image captioning with text-based visual attention**. *Neural Processing Letters*, 49(1):177–185, 2019.

[63] HE, K.; ZHANG, X.; REN, S.; SUN, J. **Delving deep into rectifiers: Surpassing human-level performance on imagenet classification**. In: *Proceedings of the IEEE international conference on computer vision*, p. 1026–1034, 2015.

[64] HE, X.; SHI, B.; BAI, X.; XIA, G.-S.; ZHANG, Z.; DONG, W. **Image caption generation with part of speech guidance**. *Pattern Recognition Letters*, 2017.

[65] HE, X.; YANG, Y.; SHI, B.; BAI, X. **Vd-san: Visual-densely semantic attention network for image caption generation**. *Neurocomputing*, 328:48–55, 2019.

[66] HEBERT, M. **Natural language understanding (nlu) processing based on user-specified interests**, Apr. 7 2016. US Patent App. 14/503,469.

[67] HIXON, B.; CLARK, P.; HAJISHIRZI, H. **Learning knowledge graphs for question answering through conversational dialog**. In: *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, p. 851–861, 2015.

[68] HOCHREITER, S. **The vanishing gradient problem during learning recurrent neural nets and problem solutions**. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

[69] HOCHREITER, S. **The vanishing gradient problem during learning recurrent neural nets and problem solutions**. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

[70] HOCHREITER, S.; BENGIO, Y.; FRASCONI, P.; SCHMIDHUBER, J.; OTHERS. **Gradient flow in recurrent nets: the difficulty of learning long-term dependencies**, 2001.

[71] HOCHREITER, S.; SCHMIDHUBER, J. **Long short-term memory**. *Neural computation*, 9(8):1735–1780, 1997.

[72] HSU, K.; LEVINE, S.; FINN, C. **Unsupervised learning via meta-learning**. *arXiv preprint arXiv:1810.02334*, 2018.

[73] HU, B. **Teaching quality evaluation research based on neural network for university physical education**. In: *2017 International Conference on Smart Grid and Electrical Automation (ICSGEA)*, p. 290–293. IEEE, 2017.

[74] HU, S.; ZOU, L.; YU, J. X.; WANG, H.; ZHAO, D. **Answering natural language questions by subgraph matching over knowledge graphs**. *IEEE Transactions on Knowledge and Data Engineering*, 30(5):824–837, 2017.

[75] HWANG, R.-H.; PENG, M.-C.; HUANG, C.-W.; LIN, P.-C.; NGUYEN, V.-L. **An unsupervised deep learning model for early network traffic anomaly detection**. *IEEE Access*, 8:30387–30399, 2020.

[76] IBEH, G.; AGBO, G. **Estimation of mean monthly global solar radiation for warri-nigeria (using angstrom and mlp ann model)**. *Advances in applied science research*, 3:12–18, 2012.

[77] IDE, H.; KURITA, T. **Improvement of learning for cnn with relu activation by sparse regularization**. In: *2017 International Joint Conference on Neural Networks (IJCNN)*, p. 2684–2691. IEEE, 2017.

[78] JABBAR, H.; KHAN, R. Z. **Methods to avoid over-fitting and under-fitting in supervised machine learning (comparative study)**. *Computer Science, Communication and Instrumentation Devices*, 2015.

[79] JABBAR, H.; KHAN, R. Z. **Methods to avoid over-fitting and under-fitting in supervised machine learning (comparative study)**. *Computer Science, Communication and Instrumentation Devices*, 2015.

[80] JADERBERG, M.; MNIH, V.; CZARNECKI, W. M.; SCHAUL, T.; LEIBO, J. Z.; SILVER, D.; KAVUKCUOGLU, K. **Reinforcement learning with unsupervised auxiliary tasks**. *arXiv preprint arXiv:1611.05397*, 2016.

[81] JAECH, A.; HECK, L.; OSTENDORF, M. **Domain adaptation of recurrent neural networks for natural language understanding**. *arXiv preprint arXiv:1604.00117*, 2016.

[82] JIA, X.; GAVVES, E.; FERNANDO, B.; TUYTELAARS, T. **Guiding the long-short term memory model for image caption generation**. In: *Proceedings of the IEEE international conference on computer vision*, p. 2407–2415, 2015.

[83] JIAO, Z.; SUN, S.; SUN, K. **Chinese lexical analysis with deep bi-gru-crf network**. *arXiv preprint arXiv:1807.01882*, 2018.

[84] KALRA, S.; LEEKHA, A. **Survey of convolutional neural networks for image captioning**. *Journal of Information and Optimization Sciences*, 41(1):239–260, 2020.

[85] KANAI, S.; FUJIWARA, Y.; IWAMURA, S. **Preventing gradient explosions in gated recurrent units**. In: *Advances in neural information processing systems*, p. 435–444, 2017.

[86] KANG, N.; SINGH, B.; AFZAL, Z.; VAN MULLIGEN, E. M.; KORS, J. A. **Using rule-based natural language processing to improve disease normalization in biomedical text**. *Journal of the American Medical Informatics Association*, 20(5):876–881, 2013.

[87] KENNEWICK, R. A.; LOCKE, D.; KENNEWICK, M. R.; KENNEWICK, R.; FREEMAN, T.; OTHERS. **System and method for filtering and eliminating noise from natural language utterances to improve speech recognition and parsing**, Mar. 20 2012. US Patent 8,140,327.

[88] KETKAR, N.; SANTANA, E. **Deep Learning with Python**, volume 1. Springer, 2017.

[89] KIM, J.; SCOTT, C. D. **Robust kernel density estimation**. *Journal of Machine Learning Research*, 13(Sep):2529–2565, 2012.

[90] KLÜWER, T. **From chatbots to dialog systems**. In: *Conversational agents and natural language interaction: Techniques and Effective Practices*, p. 1–22. IGI Global, 2011.

[91] KOUSARRIZI, M. R. N.; GHANBARI, A. A.; TESHNEHLAB, M.; SHOREHDELI, M. A.; GHARAVIRI, A. **Feature extraction and classification of eeg signals using wavelet transform, svm and artificial neural networks for brain computer interfaces**. In: *2009 International Joint Conference on Bioinformatics, Systems Biology and Intelligent Computing*, p. 352–355. IEEE, 2009.

[92] KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. **Imagenet classification with deep convolutional neural networks**. In: *Advances in neural information processing systems*, p. 1097–1105, 2012.

[93] KULKARNI, A.; SHIVANANDA, A. **Natural language processing recipes**. Springer, 2019.

[94] LAVI, O.; AUERBACH, G.; PERSKY, E. **Dynamic natural language understanding**, Nov. 23 2010. US Patent 7,840,400.

[95] LAVIE, A.; AGARWAL, A. **Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments**. In: *Proceedings of the Second Workshop on Statistical Machine Translation*, p. 228–231. Association for Computational Linguistics, 2007.

[96] LAWRENCE, S.; GILES, C. L. **Overfitting and neural networks: conjugate gradient and backpropagation**. In: *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, volume 1, p. 114–119. IEEE, 2000.

[97] LECUN, Y.; BENGIO, Y.; HINTON, G. **Deep learning**. Nature, 2015.

[98] LECUN, Y.; BOSER, B.; DENKER, J. S.; HENDERSON, D.; HOWARD, R. E.; HUBBARD, W.; JACKEL, L. D. **Backpropagation applied to handwritten zip code recognition**. *Neural computation*, 1(4):541–551, 1989.

[99] LEE, C.; JUNG, S.; KIM, K.; LEE, D.; LEE, G. G. **Recent approaches to dialog management for spoken dialog systems**. *Journal of Computing Science and Engineering*, 4(1):1–22, 2010.

[100] LI, H. **Learning to rank for information retrieval and natural language processing**. *Synthesis Lectures on Human Language Technologies*, 4(1):1–113, 2011.

[101] LI, H. **Learning to rank for information retrieval and natural language processing**. *Synthesis Lectures on Human Language Technologies*, 7(3):1–121, 2014.

[102] LI, J.; JURAFSKY, D. **Do multi-sense embeddings improve natural language understanding?** *arXiv preprint arXiv:1506.01070*, 2015.

[103] LI, L.; TANG, S.; DENG, L.; ZHANG, Y.; TIAN, Q. **Image caption with global-local attention**. In: *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[104] LI, S.; ZHANG, J.; GUO, Q.; LEI, J.; TU, D. **Generating image descriptions with multidirectional 2d long short-term memory**. *IET Computer Vision*, 11(1):104–111, 2016.

[105] LI, X.; YUAN, A.; LU, X. **Multi-modal gated recurrent units for image description**. *Multimedia Tools and Applications*, 77(22):29847–29869, 2018.

[106] LI, X.; LI, X.; QU, Y.; HE, D. **Gear pitting level diagnosis using vibration signals with an improved inception structure**. *Vibroengineering Procedia*, 20:70–75, 2018.

[107] LIANG, P. **Learning executable semantic parsers for natural language understanding**. *Communications of the ACM*, 59(9):68–76, 2016.

[108] LIDDY, E. D. **Natural language processing**. *Syracuse University*, 2001.

[109] LIN, C.-Y. **Rouge: A package for automatic evaluation of summaries**. In: *Text summarization branches out*, p. 74–81, 2004.

[110] LIN, T.-Y.; MAIRE, M.; BELONGIE, S.; HAYS, J.; PERONA, P.; RAMANAN, D.; DOLLÁR, P.; ZITNICK, C. L. **Microsoft coco: Common objects in context**. In: *European conference on computer vision*, p. 740–755. Springer, 2014.

[111] LITKOWSKI, K. **Senseval-3 task: Automatic labeling of semantic roles**. In: *Proceedings of SENSEVAL-3, the Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, p. 9–12, 2004.

[112] LIU, B.; ZHANG, L. **A survey of opinion mining and sentiment analysis**. In: *Mining text data*, p. 415–463. Springer, 2012.

[113] LUO, R. C.; CHANG LIN, H. **Coping with overfitting problems of image caption models for service robotics applications**. In: *2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS)*, p. 815–820, May 2019.

[114] MA, J.; WEN, Y.; YANG, L. **Lagrangian supervised and semi-supervised extreme learning machine**. *Applied Intelligence*, 49(2):303–318, 2019.

[115] MALHOTRA, K.; BANSAL, S.; GANAPATHY, S. **Active learning methods for low resource end-to-end speech recognition.** In: *INTERSPEECH*, p. 2215–2219, 2019.

[116] MANTOVANI, R. G.; OTHERS. **Uso de meta-aprendizado para o ajuste de hiperparâmetros em problemas de classificação**. PhD thesis, Universidade de São Paulo (USP). Instituto de Ciências Matemáticas e de . . . , 2018.

[117] MAO, J.; XU, W.; YANG, Y.; WANG, J.; HUANG, Z.; YUILLE, A. **Deep captioning with multimodal recurrent neural networks (m-rnn)**. *arXiv preprint arXiv:1412.6632*, 2014.

[118] MARULLI, F.; VALLIFUOCO, L. **The imitation game to cultural heritage: a human-like interaction driven approach for supporting art recreation**. In: *Interactivity, Game Creation, Design, Learning, and Innovation*, p. 145–153. Springer, 2016.

[119] MASON, W.; VAUGHAN, J. W.; WALLACH, H. **Computational social science and social computing**, 2014.

[120] MESFAR, S. **Towards a cascade of morpho-syntactic tools for arabic natural language processing**. In: *International Conference on Intelligent Text Processing and Computational Linguistics*, p. 150–162. Springer, 2010.

[121] MEURERS, D. **Natural language processing and language learning**. *The Encyclopedia of Applied Linguistics*, 2012.

[122] MUKHERJEE, J. **Aviation field service report natural language processing**, Feb. 6 2018. US Patent 9,886,478.

[123] MUKKAMALA, M. C.; HEIN, M. **Variants of rmsprop and adagrad with logarithmic regret bounds**. In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, p. 2545–2553. JMLR. org, 2017.

[124] NADKARNI, P. M.; OHNO-MACHADO, L.; CHAPMAN, W. W. **Natural language processing: an introduction**. *Journal of the American Medical Informatics Association*, 18(5):544–551, 2011.

[125] NAIR, V.; HINTON, G. E. **Rectified linear units improve restricted boltzmann machines**. In: *Proceedings of the 27th international conference on machine learning (ICML-10)*, p. 807–814, 2010.

[126] NAKASHIMA, K.; IWASHITA, Y.; KAWAMURA, A.; KURAZUME, R. **Fourth-person captioning: Describing daily events by uni-supervised and tri-regularized training**. In: *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, p. 2122–2127. IEEE, 2018.

[127] NAMMOUS, M. K.; SAEED, K. **Natural language processing: Speaker, language, and gender identification with lstm**. In: *Advanced Computing and Systems for Security*, p. 143–156. Springer, 2019.

[128] NEZAMI, O. M.; DRAS, M.; WAN, S.; PARIS, C.; HAMEY, L. **Towards generating stylized image captions via adversarial training**. In: *Pacific Rim International Conference on Artificial Intelligence*, p. 270–284. Springer, 2019.

[129] NGUYEN, G. H.; BOUZERDOUM, A.; PHUNG, S. L. **Learning pattern classification tasks with imbalanced data sets**. *Pattern recognition*, p. 193–208, 2009.

[130] NICKEL, M.; ROSASCO, L.; POGGIO, T. **Holographic embeddings of knowledge graphs**. *arXiv preprint arXiv:1510.04935*, 2015.

[131] NIE, Y.; WILLIAMS, A.; DINAN, E.; BANSAL, M.; WESTON, J.; KIELA, D. **Adversarial nli: A new benchmark for natural language understanding**. *arXiv preprint arXiv:1910.14599*, 2019.

[132] OLAH, C. **Understanding lstm networks, 2015**. *URL http://colah. github. io/posts/2015-08-Understanding-LSTMs*, 2015.

[133] O'SHEA, K.; NASH, R. **An introduction to convolutional neural networks**. *arXiv preprint arXiv:1511.08458*, 2015.

[134] OZATAY, M.; AYGUN, L.; JIA, H.; KUMAR, P.; MEHLMAN, Y.; WU, C.; WAGNER, S.; STURM, J. C.; VERMA, N. **Artificial intelligence meets large-scale sensing: Using large-area electronics (lae) to enable intelligent spaces**. In: *2018 IEEE Custom Integrated Circuits Conference (CICC)*, p. 1–8. IEEE, 2018.

[135] PASCANU, R.; MIKOLOV, T.; BENGIO, Y. **On the difficulty of training recurrent neural networks**. In: *International conference on machine learning*, p. 1310–1318, 2013.

[136] PENG, Y.; LIU, X.; WANG, W.; ZHAO, X.; WEI, M. **Image caption model of double lstm with scene factors**. *Image and Vision Computing*, 86:38–44, 2019.

[137] PONTI, M. A.; DA COSTA, G. B. P. **Como funciona o deep learning**. *arXiv preprint arXiv:1806.07908*, 2018.

[138] POPOV, A. **Lexical modeling for natural language processing**. In: *Selected papers from the CLARIN Annual Conference 2018, Pisa, 8-10 October 2018*, p. 152–165. Linköping University Electronic Press, 2019.

[139] QU, Z.; CAO, B.; WANG, X.; LI, F.; XU, P.; ZHANG, L. **Feedback lstm network based on attention for image description generator**. *CMC-COMPUTERS MATERIALS & CONTINUA*, 59(2):575–589, 2019.

[140] QUESADA, L.; BERZAL, F.; CUBERO, J.-C. **A model-based multilingual natural language parser—implementing chomsky's x-bar theory in modelcc**. In: *International Conference on Flexible Query Answering Systems*, p. 293–304. Springer, 2013.

[141] RADFORD, A.; NARASIMHAN, K.; SALIMANS, T.; SUTSKEVER, I. **Improving language understanding by generative pre-training**, 2018.

[142] RASHTCHIAN, C.; YOUNG, P.; HODOSH, M.; HOCKENMAIER, J. **Collecting image annotations using amazon's mechanical turk**. In: *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, p. 139–147. Association for Computational Linguistics, 2010.

[143] RAVANELLI, M.; BRAKEL, P.; OMOLOGO, M.; BENGIO, Y. **Light gated recurrent units for speech recognition**. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2(2):92–102, 2018.

[144] RIBEIRO, L. C. F. **Análise de sentimento contextual em diálogos utilizando aprendizado de máquina**. PhD thesis, Universidade Estadual Paulista (UNESP), 2019.

[145] SAINATH, T. N.; RAMABHADRAN, B.; NAHAMOO, D.; KANEVSKY, D.; VAN COMPERNOLLE, D.; DEMUYNCK, K.; GEMMEKE, J. F.; BELLEGARDA, J. R.; SUNDARAM, S. **Exemplar-based processing for speech recognition: An overview**. *IEEE Signal Processing Magazine*, 29(6):98–113, 2012.

[146] SATTIKAR, A.; KULKARNI, R. **Natural language processing for content analysis in social networking**. *International Journal of Engineering Inventions*, 1(2012):06–09, 2012.

[147] SCHIFANELLA, R.; REDI, M.; AIELLO, L. M. **An image is worth more than a thousand favorites: Surfacing the hidden beauty of flickr pictures**. In: *Ninth International AAAI Conference on Web and Social Media*, 2015.

[148] SCHUSTER, S.; MANNING, C. D. **Enhanced english universal dependencies: An improved representation for natural language understanding tasks**. In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, p. 2371–2378, 2016.

[149] SESHADRI, M.; SRIKANTH, M.; BELOV, M. **Image to language understanding: Captioning approach**. *arXiv preprint arXiv:2002.09536*, 2020.

[150] SHARMA, G.; KALENA, P.; MALDE, N.; NAIR, A.; PARKAR, S. **Visual image caption generator using deep learning**. *Available at SSRN 3368837*, 2019.

[151] SHARMA, S. **Activation functions: Neural networks**. *Towards Data Science*, 6, 2017.

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

[153] SIT, M. A.; KOYLU, C.; DEMIR, I. **Identifying disaster-related tweets and their semantic, spatial and temporal context using deep learning, natural language processing and spatial analysis: a case study of hurricane irma**. *International Journal of Digital Earth*, 12(11):1205–1229, 2019.

[154] SONG, J.; HE, T.; GAO, L.; XU, X.; HANJALIC, A.; SHEN, H. T. **Unified binary generative adversarial network for image retrieval and compression**. *International Journal of Computer Vision*, p. 1–22, 2020.

[155] SRIVASTAVA, N.; HINTON, G.; KRIZHEVSKY, A.; SUTSKEVER, I.; SALAKHUTDINOV, R. **Dropout: a simple way to prevent neural networks from overfitting**. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[156] STAUDEMEYER, R. C.; MORRIS, E. R. **Understanding lstm–a tutorial into long short-term memory recurrent neural networks**. *arXiv preprint arXiv:1909.09586*, 2019.

[157] ŠTENCL, M.; LENDEL, V. **Application of selected artificial intelligence methods in terms of transport and intelligent transport systems**. *Periodica Polytechnica Transportation Engineering*, 40(1):11–16, 2012.

[158] SUN, Y.; XUE, B.; ZHANG, M.; YEN, G. G. **Evolving deep convolutional neural networks for image classification**. *IEEE Transactions on Evolutionary Computation*, 2019.

[159] SUNARYA, P. A.; REFIANTI, R.; MUTIARA, A. B.; OCTAVIANI, W. **Comparison of accuracy between convolutional neural networks and naïve bayes classifiers in sentiment analysis on twitter**. *International Journal of Advanced Computer Science and Applications*, 2019.

[160] SURABHI, M. C. **Natural language processing future**. In: *2013 International Conference on Optical Imaging Sensor and Security (ICOSS)*, p. 1–3. IEEE, 2013.

[161] SZEGEDY, C.; IOFFE, S.; VANHOUCKE, V.; ALEMI, A. **Inception-v4, inception-resnet and the impact of residual connections on learning**. *arXiv preprint arXiv:1602.07261*, 2016.

[162] SZEGEDY, C.; IOFFE, S.; VANHOUCKE, V.; ALEMI, A. A. **Inception-v4, inception-resnet and the impact of residual connections on learning**. In: *Thirty-first AAAI conference on artificial intelligence*, 2017.

[163] SZEGEDY, C.; LIU, W.; JIA, Y.; SERMANET, P.; REED, S.; ANGUELOV, D.; ERHAN, D.; VANHOUCKE, V.; RABINOVICH, A. **Going deeper with convolutions**. In:

Proceedings of the IEEE conference on computer vision and pattern recognition, p. 1–9, 2015.

[164] SZEGEDY, C.; VANHOUCKE, V.; IOFFE, S.; SHLENS, J.; WOJNA, Z. **Rethinking the inception architecture for computer vision**. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, p. 2818–2826, 2016.

[165] SZELISKI, R. **Computer vision: algorithms and applications**. Springer Science & Business Media, 2010.

[166] TAN, Y. H.; CHAN, C. S. **Phrase-based image caption generator with hierarchical lstm network**. *Neurocomputing*, 333:86–100, 2019.

[167] TAPASWI, N.; JAIN, S. **Morphological and lexical analysis of the sanskrit sentences**. *MIT International Journal of Computer Science & Information Technology*, 1(1):28–31, 2011.

[168] TIELEMAN, T.; HINTON, G. **Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude**. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[169] TORFI, A.; SHIRVANI, R. A.; KENESHLOO, Y.; TAVVAF, N.; FOX, E. A. **Natural language processing advancements by deep learning: A survey**. *arXiv preprint arXiv:2003.01200*, 2020.

[170] ULLMANN, M. R.; PIMENTEL, K. F.; DE MELO, L. A.; DA CRUZ, G.; VINHAL, C. **Comparison of pso variants applied to large scale optimization problems**. In: *2017 IEEE Latin American Conference on Computational Intelligence (LA-CCI)*, p. 1–6. IEEE, 2017.

[171] VEDANTAM, R.; LAWRENCE ZITNICK, C.; PARIKH, D. **Cider: Consensus-based image description evaluation**. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, p. 4566–4575, 2015.

[172] VINYALS, O.; TOSHEV, A.; BENGIO, S.; ERHAN, D. **Show and tell: A neural image caption generator**. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, p. 3156–3164, 2015.

[173] VOUTILAINEN, A. **A syntax-based part-of-speech analyser**. *arXiv preprint cmp-lg/9502012*, 1995.

[174] WANG, A.; SINGH, A.; MICHAEL, J.; HILL, F.; LEVY, O.; BOWMAN, S. R. **Glue: A multi-task benchmark and analysis platform for natural language understanding**. *arXiv preprint arXiv:1804.07461*, 2018.

[175] WANG, F.; HUANG, S.; SHI, L.; FAN, W. **The application of series multi-pooling convolutional neural networks for medical image segmentation**. *International Journal of Distributed Sensor Networks*, 13(12):1550147717748899, 2017.

[176] WANG, Y.; LIU, J.; WANG, X. **Image caption with synchronous cross-attention**. In: *Proceedings of the on Thematic Workshops of ACM Multimedia 2017*, p. 433–441. ACM, 2017.

[177] WEBER, D. **Object interactive user interface using speech recognition and natural language processing**, Aug. 13 2002. US Patent 6,434,524.

[178] WEI, W.; CHENG, L.; MAO, X.; ZHOU, G.; ZHU, F. **Stack-vs: Stacked visual-semantic attention for image caption generation**. *arXiv preprint arXiv:1909.02489*, 2019.

[179] WENG, F.; STOIA, L.; HU, J.; FENG, Z.; CAO, J. **System and method for generating natural language phrases from user utterances in dialog systems**, Oct. 28 2014. US Patent 8,874,443.

[180] WIEGAND, M.; ROTH, B.; KLAKOW, D. **Knowledge acquisition with natural language processing in the food domain: Potential and challenges**. In: *Proceedings of the Cooking with Computers workshop (CwC)*, 2012.

[181] WU, Z.; CHRISTOFIDES, P. D. **Economic machine-learning-based predictive control of nonlinear systems**. *Mathematics*, 7(6):494, 2019.

[182] XIE, L.; LIU, Y.; JIN, L.; XIE, Z. **Derpn: Taking a further step toward more general object detection**. In: *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, p. 9046–9053, 2019.

[183] XU, H.; HOFFMANN, A. **Rdrce: combining machine learning and knowledge acquisition**. In: *Pacific Rim Knowledge Acquisition Workshop*, p. 165–179. Springer, 2010.

[184] YANG, J.; SUN, Y.; LIANG, J.; REN, B.; LAI, S.-H. **Image captioning by incorporating affective concepts learned from both visual and textual components**. *Neurocomputing*, 328:56–68, 2019.

[185] YANG, X.; CHEN, Y.-N.; HAKKANI-TÜR, D.; CROOK, P.; LI, X.; GAO, J.; DENG, L. **End-to-end joint learning of natural language understanding and dialogue manager**. In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, p. 5690–5694. IEEE, 2017.

[186] YILDIZ, T.; YILDIRIM, S.; DIRI, B. **An integrated approach to automatic synonym detection in turkish corpus**. In: *International Conference on Natural Language Processing*, p. 116–127. Springer, 2014.

[187] YU, L.-C.; WANG, J.; LAI, K. R.; ZHANG, X. **Refining word embeddings for sentiment analysis**. In: *Proceedings of the 2017 conference on empirical methods in natural language processing*, p. 534–539, 2017.

[188] YUAN, A.; LI, X.; LU, X. **3g structure for image caption generation**. *Neurocomputing*, 330:17–28, 2019.

[189] ZHANG, Z.; BI, X. **Research and experiment of intelligent natural language processing algorithms**. *Wireless Personal Communications*, 102(4):2927–2939, 2018.

[190] ZHENG, J.; KRISHNAMURTHY, S.; CHEN, R.; CHEN, M.-H.; GE, Z.; LI, X. **Image captioning with integrated bottom-up and multi-level residual top-down attention for game scene understanding**. *arXiv preprint arXiv:1906.06632*, 2019.

[191] ZHENG, W.; CHENG, H.; YU, J. X.; ZOU, L.; ZHAO, K. **Interactive natural language question answering over knowledge graphs**. *Information Sciences*, 481:141–159, 2019.

[192] ZHOU, L.; XU, C.; KOCH, P.; CORSO, J. J. **Watch what you just said: Image captioning with text-conditional attention**. In: *Proceedings of the on Thematic Workshops of ACM Multimedia 2017*, p. 305–313. ACM, 2017.

[193] ZHU, W.; YAO, T.; ZHANG, W.; WEI, B. **Part-of-speech-based long short-term memory network for learning sentence representations**. *IEEE Access*, 7:51810–51816, 2019.

[194] ZOU, F.; SHEN, L.; JIE, Z.; ZHANG, W.; LIU, W. **A sufficient condition for convergences of adam and rmsprop**. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 11127–11135, 2019.