

Deep Learning Aplicado à Detecção de Falhas em Pavimentações Urbanas

Rafael Barbosa de Carvalho, Elloá B. Guedes

¹Laboratório de Sistemas Inteligentes
Grupo de Pesquisa em Sistemas Inteligentes
Universidade do Estado do Amazonas (UEA)
Av. Darcy Vargas, 1200 – Manaus – Amazonas
`{rbc.eng, ebgcosta}@uea.edu.br`

Resumo. Esta proposta de trabalho de conclusão de curso contempla a tarefa de Visão Computacional de detecção de quatro tipos de falhas na pavimentação asfáltica a partir de exemplos oriundos da base de dados RDD2020, contendo mais de 18 mil exemplos da Índia, Japão e República Checa. A partir de uma validação cruzada holdout foram avaliadas diferentes configurações de treinamento da arquitetura YOLOv4, uma rede neural convolucional moderna voltada para tarefas de detecção. Os resultados obtidos até o momento evidenciam boas métricas desta arquitetura no contexto em questão ($mAP@0.5 = 31,43\%$) e motivam a avaliação de mais configurações e arquiteturas para o problema, que pode vir colaborar no monitoramento automático da qualidade da pavimentação asfáltica.

1. Introdução

Entende-se por *Cidade Inteligente* (CI) uma área urbana que utiliza diferentes métodos eletrônicos e computacionais integrados para coleta de dados, utilizando-os para melhorar e otimizar os serviços, recursos e operações existentes [Janani et al. 2021]. Uma CI é um lugar no qual as redes e serviços tradicionais se tornam mais flexíveis, eficientes e sustentáveis com o uso de tecnologias da informação, digitais e de telecomunicações para melhorar o funcionamento da cidade em benefício de seus habitantes. Os diferentes componentes de uma CI incluem infraestrutura, transporte, energia, assistência médica e tecnologias inteligentes. Esses componentes fazem de tal cidade não apenas inteligente, mas também eficiente. [Mohanty et al. 2016].

Um exemplo pioneiro de CI é Nova Iorque (NYC, do inglês *New York City*) nos Estados Unidos da América, em que diversos serviços e dispositivos inteligentes estão disponíveis para a população, tais como: *LinkNYC*, uma estação física que provê acesso à internet via *WiFi*, ligações telefônicas gratuitas para todo o país, plugues para carregamento elétrico de dispositivos *USB*, etc.; *Automated Meter Reading* (AMR), dispositivos físicos acoplados à medidores de água e conectados por uma rede de sensores à um sistema computacional que monitora o consumo até quatro vezes ao dia, permitindo um melhor controle dos gastos e agilizando reparos que geram desperdício de água; monitoramento automatizado da qualidade do ar; disponibilização de bases de dados abertas com dados de concursos, cargos de oficiais eleitos, motoristas habilitados, gastos com folha de pagamento da prefeitura, dentre outros, permitindo uma ampla transparência e monitoramento nos gastos públicos; iluminação inteligente das ruas; sistemas de monitoramento de tráfego urbano, etc. [NYC 2021].

Em diversas soluções presentes em ecossistemas de CIs é comum que técnicas de *Visão Computacional* (VC) figurem como protagonistas. Essa área da Computação busca desenvolver métodos computacionais que permitam reproduzir as capacidades da visão humana em tarefas, tais como, de classificação, detecção e segmentação de objetos em imagens. Em especial, as técnicas de *Deep Learning* compõem atualmente o estado da arte de diversas tarefas de VC, nas quais o aprendizado é efetuado por meio de múltiplos níveis de representação e abstração, resultando em um nível quase humano de desempenho em tarefas de classificação massiva de imagens, de detecção de objetos para veículos autônomos, dentre outras [Chollet 2017].

No tocante às demandas de soluções em CIs, um dos problemas que pode ser abordado com VC diz respeito à supervisão e monitoramento da infraestrutura de estradas, especialmente no que diz respeito à análise da qualidade da pavimentação. A infraestrutura das estradas é um bem público de extrema importância, pois contribui para o desenvolvimento e crescimento econômico, trazendo benefícios sociais essenciais, permitindo o deslocamento de pessoas, o transporte de produtos, dentre outros. As estradas conectam comunidades e empresas, fornecem acesso à serviços de educação, emprego, serviços sociais e de saúde. No entanto, é comum que a pavimentação se desgaste e se deteriore devido à fatores relacionados à localização, tempo, volume de tráfego, clima, soluções de Engenharia, dentre outros, o que demanda uma extensa análise de sua qualidade para uma manutenção eficiente e econômica visando preservar seu estado e, por conseguinte, a segurança dos que nela transitam [Miller and Zaloshnja 2009].

No Brasil, o transporte rodoviário dispõe de 120.767,3 km de malha rodoviária federal, dos quais apenas apenas 64.000 km são pavimentados. O transporte rodoviário é o principal sistema logístico do País, correspondendo a cerca de 65 % de tudo o que circula, com grande influência tanto no abastecimento de cidades quanto na circulação de tudo o que é produzido [CNT 2019]. Muitos municípios e autoridades rodoviárias procuram implementar uma avaliação automatizada dos danos nas estradas. No entanto, estes municípios e autoridades geralmente carecem de tecnologia, *know-how* e recursos financeiros para pagar equipamentos de última geração para coleta de dados e análise de danos em estradas. [Arya et al. 2021a]. No Brasil, o Departamento Nacional de Infraestrutura de Transportes (DNIT), através da Diretoria de Infraestrutura Rodoviária (DIR), é responsável pela manutenção, recuperação e construção de vias de transportes interurbanas federais, [DNIT 2021] com programas como o PATO (Plano Atual de Trabalho e Orçamento), responsável por serviços para reparar ou sanar defeitos e reestabelecer o funcionamento dos componentes da rodovia, o CREMA (Contrato de Recuperação e Manutenção), responsável por intervenções para evitar o surgimento ou agravamento de defeitos e, de forma geral, realizar intervenções destinadas a reestabelecer o perfeito funcionamento da infraestrutura viária, devolvendo-a suas características técnicas originais [Federal 2021].

Considerando a importância do transporte rodoviário no Brasil, as dificuldades para implementar avaliações automatizadas e a importância da constante manutenção desta via de transporte para prevenir acidentes e para colaborar em aspectos econômicos para o País, esta proposta de trabalho de conclusão de curso em Engenharia de Computação visa contribuir neste contexto tentando responder à seguinte questão: *Os modelos de Deep Learning para detecção de objetos podem colaborar na detecção au-*

tomática de falhas em pavimentações urbanas?

Para responder esta pergunta será considerada uma base de dados realística disponível na literatura, denominada RDD2020, composta de mais de 20 mil instâncias de imagens coloridas bidimensionais de pavimentações urbanas contendo quatro tipos diferentes de danos, compreendendo três categorias de rachaduras e buracos [Arya 2021]. Para a proposição de soluções inteligentes para tal detecção serão considerados modelos de redes neurais artificiais convolucionais (CNNs, do inglês *Convolutional Neural Networks*) disponíveis no estado da arte a serem treinados mediante o paradigma do Aprendizado Supervisionado e implementados com *frameworks* de *Deep Learning* disponíveis na linguagem Python. A execução dos scripts propostos será feita em um servidor do Laboratório de Sistemas Inteligentes (LSI) da Universidade do Estado do Amazonas (UEA).

1.1. Objetivos

O objetivo geral deste trabalho consiste em avaliar modelos de *Deep Learning* aplicados ao problema de detecção de falhas em pavimentações urbanas. Para alcançar esta meta, alguns objetivos específicos precisam ser contemplados, a citar:

1. Formular um referencial teórico sobre CNNs aplicadas à problemas de detecção;
2. Coletar, analisar e organizar uma base de dados de imagens de pavimentações urbanas disponível na literatura;
3. Identificar tecnologias para implementação das CNNs em ambiente computacional;
4. Identificar, propor e/ou modificar arquiteturas de CNNs para a tarefa considerada;
5. Treinar e testar os diferentes modelos propostos; e
6. Avaliar os resultados obtidos.

1.2. Justificativa

Contribuir para o monitoramento e a detecção de falhas na pavimentação asfáltica para o transporte rodoviário significa evitar acidentes, salvar vidas e diminuir gastos, especialmente àqueles ligados ao custo do transporte logístico.

No caso dos Estados Unidos, mais da metade das mortes em rodovias estão relacionadas às condições deficientes das estradas, fator substancialmente mais letal do que dirigir embriagado, com excesso de velocidade ou sem cintos de segurança. Neste país, dez acidentes rodoviários acontecem por minuto, totalizando 5,3 milhões por ano, dos quais 38 % resultam em ferimentos não letais. As más condições na pavimentação já contribuíram para mais de 22.000 mortes e causam impactos financeiros da ordem de US\$ 217 bilhões anuais [Miller and Zaloshnja 2009]. No Brasil, conforme dados da Confederação Nacional do Transporte (CNT), em pesquisa realizada no ano de 2019, foi constatado que 52,9 % das rodovias apresentavam problema em sua pavimentação e que as condições das rodovias impactavam diretamente nos custos do transporte. No referido ano, estimou-se que, na média nacional, as inadequações do pavimento resultaram em uma elevação do custo operacional do transporte em torno de 28,5 %, sendo que o maior índice foi registrado na região Norte, com aumento percentual de 38,5 % [CNT 2019].

Pavimentos deficientes reduzem a segurança viária e aumentam o custo de manutenção dos veículos, do consumo de combustível, do uso de lubrificantes e do desgaste de pneus e freios. Em rodovias com pavimento em péssimo estado de conservação,

o acréscimo no custo operacional chega a ser de 91,5 %. A qualidade da pavimentação também está diretamente ligada à segurança. No ano de 2020, registrou-se 63.447 acidentes nas rodovias federais brasileiras, que culminaram em custos da ordem de R\$ 10,22 bilhões, índices que revelam a necessidade e a importância de investimentos efetivos em infraestrutura rodoviária no Brasil [CNT 2020].

Considerando que as técnicas de *Deep Learning* são emergentes, é importante propor trabalhos que possam ajudar a verificar a adequação destas soluções, indicando vantagens e limitações, bem como comparações com o estado da arte. Em particular, neste trabalho será considerado um problema com relevância prática e com dados reais.

No mais, do ponto de vista do bacharel em Engenharia de Computação em formação, a proposta de trabalho de conclusão de curso corrobora para a prática de conceitos, tecnologias e métodos de uma área emergente. Por fim, deve-se mencionar a importância da realização deste trabalho com vistas a colaborar com as atividades desenvolvidas pelo LSI, uma iniciativa do *Grupo de Pesquisas em Sistemas Inteligentes* da Escola Superior de Tecnologia (EST) da UEA.

1.3. Metodologia

Para alcançar os objetivos propostos no escopo deste trabalho, a condução das atividades realizadas obedeceu à metodologia escrita a seguir:

1. Estudo dos conceitos relacionados às CNNs perante tarefas de detecção;
2. Levantamento do ferramental tecnológico para implementação das CNNs;
3. Consolidação de uma base de dados representativa de imagens de pavimentações urbanas, contendo imagens de pavimentações com falhas;
4. Identificação de arquiteturas canônicas de CNNs do estado da arte para problemas de detecção;
5. Treinar as redes identificadas para a tarefa de aprendizado considerada;
6. Testar as redes previamente treinadas com vistas a coletar métricas de desempenho;
7. Analisar os resultados e elencar os modelos mais adequados para o problema considerado;
8. Escrita da proposta de Trabalho de Conclusão de Curso;
9. Defesa da proposta de Trabalho de Conclusão de Curso;
10. Escrita do Trabalho de Conclusão de Curso; e
11. Defesa do Trabalho de Conclusão de Curso;

1.4. Cronograma

Uma visão geral do cronograma das atividades a serem desenvolvidas ao longo deste trabalho podem ser vistas na Tabela 1. Elas possuem relação com as atividades listadas na Seção 1.3, a qual detalha a metodologia que o projeto de pesquisa deverá seguir.

2. Fundamentação Teórica

Nas seções a seguir encontra-se uma breve apresentação dos conceitos que fundamentaram a realização deste trabalho. Uma visão geral sobre *Deep Learning* encontra-se disponível na Seção 2.1; os conceitos e características gerais das CNNs são abordados na Seção 2.2; a Seção 2.3 considera as CNNs quando aplicadas no contexto da detecção de objetos, incluindo as métricas de desempenho e as principais arquiteturas propostas na literatura; e, por fim, a Seção 2.4 apresenta os trabalhos relacionados na literatura.

Tabela 1. Cronograma de atividades levando em consideração os nove meses para a realização do TCC.

	2021				2022			
	09	10	11	12	01	02	03	04
Atividade 1	X	X						
Atividade 2		X	X					
Atividade 3		X	X	X				
Atividade 4		X	X	X				
Atividade 5					X	X		
Atividade 6						X	X	
Atividade 7							X	X
Atividade 8	X	X	X	X				
Atividade 9					X			
Atividade 10					X	X	X	X
Atividade 11								X

2.1. Deep Learning

Inteligência Artificial (IA) é uma área da Computação que pode ser definida como a automatização de tarefas intelectuais normalmente feitas por humanos [Chollet 2017]. Algoritmos e modelos da IA demonstram capacidades para o aprendizado de padrões, processamento de linguagem natural, localização de objetos, representação de conhecimento, reconhecimento de fala, etc.

Machine Learning (ML) é uma subárea da IA que permite com que modelos computacionais aprendam a partir de dados sem terem sido explicitamente programados para este fim. Em outras palavras, o objetivo da ML é desenvolver métodos que aprendam de forma automática usando observações do mundo real, os chamados dados de treinamento, sem a definição explícita de regras ou lógicas por seres humanos, ou seja, sem um supervisor do treinamento. Neste sentido, pode-se definir ML como programação por amostra de dados. [Khan et al. 2018].

Deep Learning (DL) ou Aprendizado Profundo, por sua vez, é um conjunto de técnicas de ML capazes de representar e reconhecer características automaticamente através da exposição à um conjunto de dados, utilizando múltiplas camadas sucessivas de representações hierárquicas [Chollet 2017]. Aprender características automaticamente em vários níveis de abstração permite que modelos aprendam funções complexas, sem depender completamente de características criadas por humanos. Isso é especialmente importante para abstrações em alto nível, nas quais muitas vezes sequer os especialistas humanos não são capazes de descrever de forma explícita as características de uma dada entrada que permitem a obtenção de uma determinada saída [Bengio 2009].

Mediante a grande quantidade de imagens com rótulos disponíveis para treinamento e o avanço do poder computacional, soluções de DL tem se destacado em problemas de VC, sendo utilizada para classificação e localização de objetos [Uijlings et al. 2013], aprendizado e transferência de estilos [Gatys et al. 2015], colorização de imagens em preto e branco [Zhang et al. 2016], detecção de patologias em exames médicos [Evangelista and Guedes 2018], etc.

2.2. Redes Neurais Convolucionais

Redes Neurais Convolucionais (CNNs, do inglês *Convolutional Neural Networks*) são uma categoria de redes neurais artificiais profundas que tem sido aplicadas com sucesso no aprendizado de características em imagens, com desempenho superior quando comparadas à outras técnicas disponíveis na literatura [Chollet 2017].

No Aprendizado Supervisionado, um paradigma em que diversas tarefa de ML são realizadas, tem-se como objetivo conceber modelos capazes de inferir uma saída dada uma certa entrada e, para tanto, tem-se o aprendizado baseado em exemplos de entrada e saída previamente utilizados em uma etapa de ajuste de parâmetros denominada treinamento. Neste processo, tem-se como objetivo ajustar os pesos dos neurônios artificiais componentes da CNN a partir dos exemplos disponíveis para a tarefa.

Dado um neurônio artificial, componente elementar de uma CNN, tem-se que os pesos são inicializados com valores aleatórios uniformemente distribuídos dentro de um intervalo definido. Ao fornecer um vetor de entrada (\mathbf{x}) a um neurônio será feita uma soma ponderada desta entrada em relação a um vetor de pesos (\mathbf{w}), resultando em um potencial de ativação:

$$u = \sum_{i=1}^n w_i x_i. \quad (1)$$

Após esta etapa, o valor u é fornecido como parâmetro para uma função de ativação $f(\cdot)$, a qual precisa ser contínua e derivável, produzindo a saída $y = f(u)$.

Considerando que os neurônios estão dispostos em múltiplas camadas hierárquicas, o erro ocorre quando o valor produzido por um neurônio na camada de saída é diferente do esperado, suscitando uma atualização no vetor de pesos (\mathbf{w}). O algoritmo mais popular para a correção de erros e aprendizado de características é o *backpropagation*, composto essencialmente de duas etapas:

- **Fase forward.** Uma entrada é fornecida aos neurônios da camada de entrada e o resultado do produto dos respectivos pesos é propagado pela rede, camada a camada de forma subsequente até chegar à camada de saída. O valor obtido é comparado com o valor esperado e o erro é verificado ao examinar a diferença entre esses valores;
- **Fase backwards.** Nesta etapa, o valor do erro é utilizado para o ajuste dos pesos através da derivada parcial da função de ativação, calculando o gradiente da função de custo para os pesos de cada neurônio. Prossegue-se da camada de saída em direção à camada de entrada até que minimize-se o erro inicialmente observado para a respectiva entrada fornecida [Goodfellow et al. 2016, Haykin 2009]

As CNNs possuem múltiplas camadas conectadas de forma hierárquica, capazes de extrair características dos dados de entrada e produzir um resultado desejado, tal como uma classificação. As camadas convolucionais, cujos neurônios possuem filtros, também chamados de *kernels*, efetuam uma convolução com a entrada para gerar um mapa de características, o qual é fornecido para a camada subsequente. As camadas de *pooling* são tipicamente utilizadas para reduzir a dimensão dos mapas de características obtidos. As camadas densas, por sua vez, são do tipo totalmente conectadas, e colaboraram para a obtenção da saída do problema. Diante do que foi exposto, quando uma

entrada é fornecida à uma CNN, tem-se que a mesma passa por um processo de sucessivas transformações de diversos tipos com vistas à extração de características para então obter um determinado resultado [Khan et al. 2018].

No caso de uma imagem ser fornecida como entrada à uma CNN, tem-se que os dados desta são representados por meio de *pixels*, os quais, respeitando o esquema de cores RGB, possuem três componentes inteiros distintos. Assim, para este contexto, tem-se que uma imagem é uma matriz tridimensional com comprimento, largura e três canais de cor, conforme ilustrado na Figura 1.

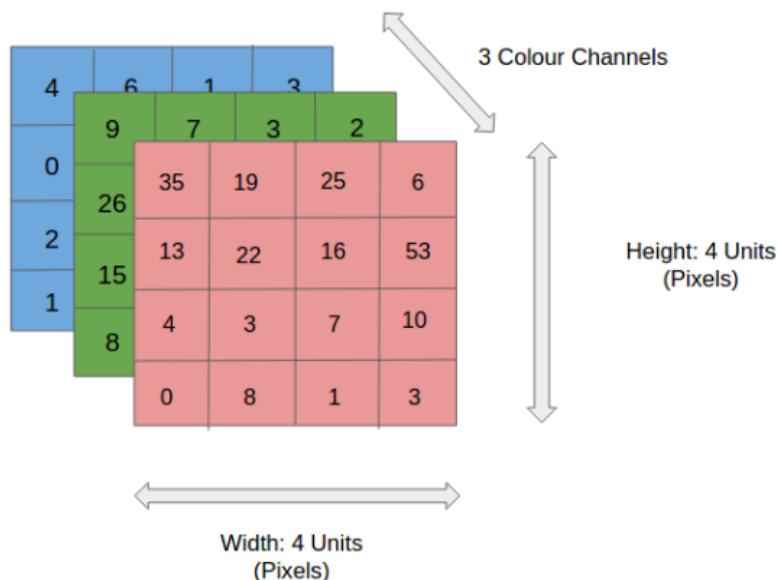


Figura 1. Representação de uma imagem em uma matriz tri-dimensional. Fonte: <https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/>.

As camadas convolucionais são as camadas mais particulares das CNNs. A convolução é uma operação entre duas funções f e g cujo objetivo é produzir uma terceira função, denotada por $f * g$ que expressa como a forma da primeira função é modificada pela segunda. No contexto de redes neurais, a primeira função é referenciada como entrada, a segunda como *kernel* (ou filtro) e o resultado da convolução é um mapa de características [Goodfellow et al. 2016]. Os filtros componentes destas camadas são matrizes de números discretos, que são ajustados durante o processo de treinamento de uma CNN visando a melhor extração de características da entrada. Para exemplificar, conforme Fig. 2, tem-se uma entrada de dimensões 4×4 e um filtro de dimensões 2×2 . Ao aplicar este filtro na entrada por meio de uma convolução, será produzido um mapa de características de dimensões 3×3 .

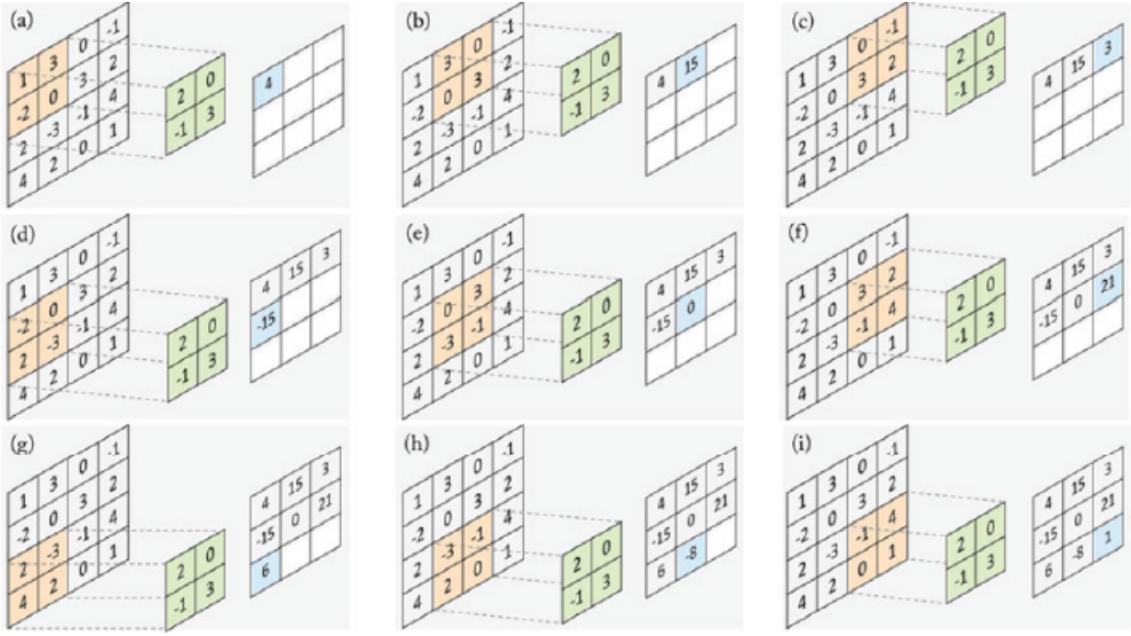


Figura 2. Exemplo da operação de convolução. Fonte: [Khan et al. 2018].

Normalmente subsequentes às camadas convolucionais, tem-se nas CNNs as camada de *pooling* cujo objetivo é a redução da dimensão da entrada. Existem dois tipos principais de camadas deste tipo, a de *Max Pooling*, conforme ilustrado na Fig. 3, que recebe como entrada um conjunto de valores e que produz como saída o maior valor fornecido entrada, e a de *Mean Pooling* que realiza a média aritmética dos valores fornecidos.

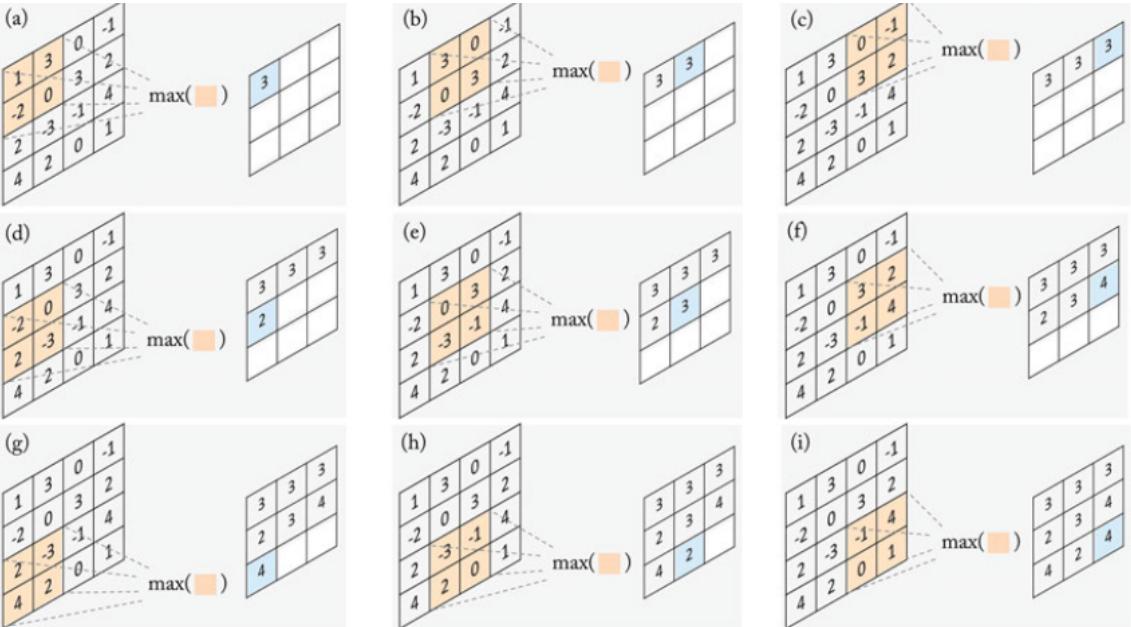


Figura 3. Processo de *Max Pooling*. Fonte: [Khan et al. 2018].

As camadas densas são camadas totalmente conectadas, normalmente localizadas no final das CNNs com vistas a atuar de modo análogo às redes neurais artificiais *multi-*

layer perceptron (MLP), produzindo uma saída específica para a tarefa em consideração [Khan et al. 2018].

A quantidade, o tipo e a disposição das camadas previamente mencionadas compõem a arquitetura de uma CNN. Porém, propor boas arquiteturas não é uma tarefa fácil. Nas tarefas de classificação, por exemplo, muitos problemas fazem uso de arquiteturas que destacaram no *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC), uma competição que avaliou algoritmos para detecção e classificação de imagens em larga escala. Esta competição contemplava uma base de dados com 1.281.167 imagens de treinamento, 50.000 imagens de validação e 100.000 imagens de teste, agrupadas em 1.000 categorias [ImageNet 2021]. Algumas das arquiteturas de CNNs para tarefas de classificação, com destaque para aquelas oriundas do ILSVRC, são mencionadas a seguir:

- **LeNet.** Conhecida por sua relevância histórica por ser a primeira arquitetura de CNN, foi concebida por LeCun no ano de 1990, proposta para reconhecimento de dígitos manuscritos com desempenho em estado da arte para o problema [LeCun et al. 1995]. Consiste em 2 camadas de convolução, cada uma seguida de uma camada de *Max Pooling* para extração de características. Após estas camadas, possui uma camada de convolução, seguida de duas camadas densas para classificar as características extraídas [Khan et al. 2018];
- **AlexNet.** Solução vencedora do desafio ILSVRC 2012, é considerada a primeira solução de CNNs profundas [Krizhevsky et al. 2012]. A sua principal diferença para as arquiteturas anteriores é o aumento da profundidade da rede, de 5 camadas na *LeNet* para 8 camadas, o que resultou em um número显著mente maior de parâmetros, e utilização de técnicas para regularização como *dropout* e *data augmentation*. Sua arquitetura consiste em 5 camadas convolucionais e 3 camadas densas, além da utilização de camadas de *max pooling* e *dropout*;
- **VGGNet.** Extremamente popular desde sua criação em 2014 [Simonyan and Zisserman 2014], utiliza filtros de tamanho reduzido (3×3) e foi concebida originalmente com 19 camadas. Devido à menor quantidade de parâmetros em razão do tamanho menor dos filtros, possui menor complexidade computacional. Sua arquitetura possui variações em tamanho de camadas, como a VGG-7 ou VGGnet-16;
- **Inception.** Vencedora do ILSVRC em 2014, possui diferenças significativas em relação às suas antecessoras. Destaca-se o “módulo Inception,” no qual o processamento ocorre de forma paralela, permitindo uma aceleração em *hardware* de sua execução. Nesta arquitetura, camadas convolucionais comuns são substituídas por pequenos blocos com um conjunto de camadas convolucionais de diferentes tamanhos (1×1 , 3×3 e 5×5) e de *pooling* para capturar informações espaciais em diferentes escalas [Khan et al. 2020, Szegedy et al. 2015, Krizhevsky et al. 2012].

2.3. Detecção de Objetos com Redes Neurais Convolucionais

Na tarefa de detecção de objetos, além da tarefa de classificação mencionada anteriormente, é necessário que seja indicada a localização de um ou mais objetos de interesse na imagem. O termo localização se refere à casos onde há apenas uma instância do objeto em uma imagem, enquanto detecção é empregado quando há várias instâncias de objetos a serem identificadas [Michelucci 2019], conforme ilustrado na Fig. 4.

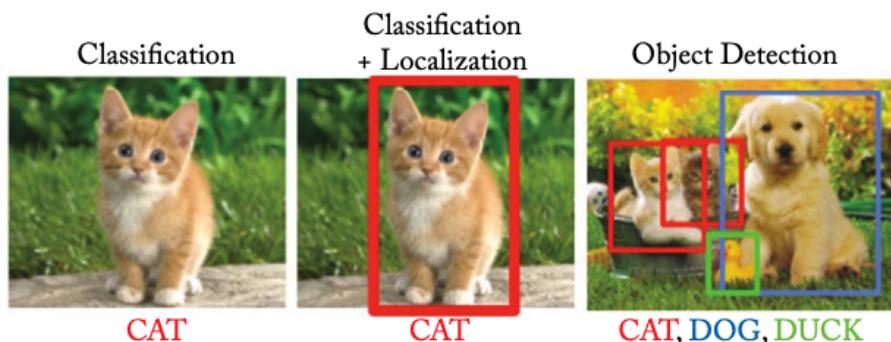


Figura 4. Contraste entre as tarefas de VC de classificação, localização e detecção. Fonte: [Khan et al. 2020].

De maneira geral, conforme ilustrado na Fig. 5, a área de Visão Computacional considera que há um *pipeline* típico para sistemas de reconhecimento de objetos, a citar:

- **Base de Objetos.** Esta base contém exemplo dos objetos do domínio do problema a serem identificados, podendo ser um descrição qualitativa ou quantitativa, superfície geométrica, um vetor de características ou exemplos rotulados em imagens;
- **Detector de características.** Responsável por aplicar operadores às imagens que localizam as características que auxiliam na formação de hipóteses;
- **Formação de hipótese.** Utiliza as características das imagens na base de objetos e atribui probabilidades de classificação às hipóteses formuladas;
- **Verificação da hipótese.** Estabelece limiares de probabilidade para confirmar uma dada classificação.

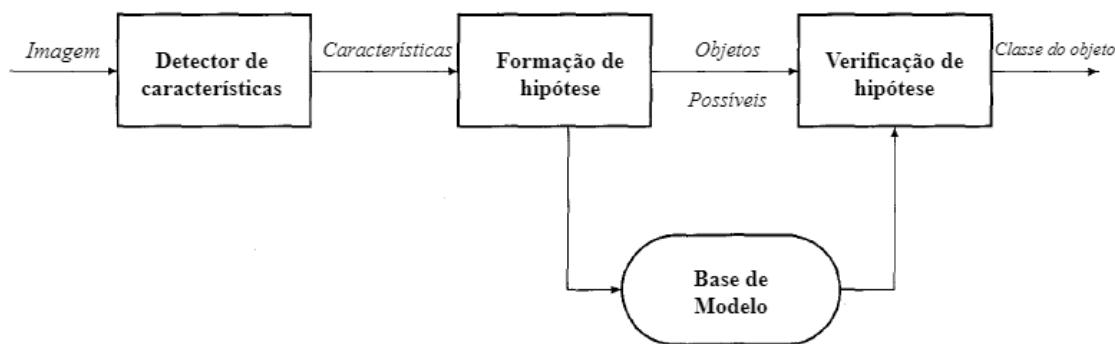


Figura 5. Pipeline para sistema de detecção de objetos. Fonte: [Jain et al. 1995].

Todos os sistemas de reconhecimento utilizam detectores de características a partir da base de objetos. Os componentes de formação e verificação de hipóteses variam conforme diferentes abordagens para solucionar o problema. Alguns sistemas usam apenas a formação de hipóteses e então selecionam o objeto com maior probabilidade como o objeto correto, como as abordagens de classificação de padrões, enquanto alguns sistemas de VC baseados em IA realizam a maior parte do trabalho na fase de verificação de hipóteses, confiando menos na etapa de formação [Jain et al. 1995].

2.3.1. Métricas de Desempenho em Detecção de Objetos

Na aferição de soluções para detecção de objetos são contempladas múltiplas métricas as quais avaliam a qualidade da localização e também da classificação. A apresentação das métricas a seguir é baseada em um *survey* disponível na literatura [Padilla et al. 2020].

A primeira métrica considerada é a *Intersection Over Union* (IoU), cuja compreensão intuitiva é ilustrada na Figura 6, a qual afere a sobreposição entre uma *bounding box* prevista B_p e a *bounding box* desejada B_d , calculada como segue:

$$\text{IoU} = \frac{\text{área}(B_p \cap B_d)}{\text{área}(B_p \cup B_d)}. \quad (2)$$

Ao comparar o IoU com um *threshold* t , tem-se que quando $\text{IoU} \geq t$ a detecção é correta, e incorreta em caso contrário. Esta métrica varia no intervalo $[0, 1]$, alcançando o valor máximo quando a sobreposição das áreas é perfeita [Michelucci 2019].

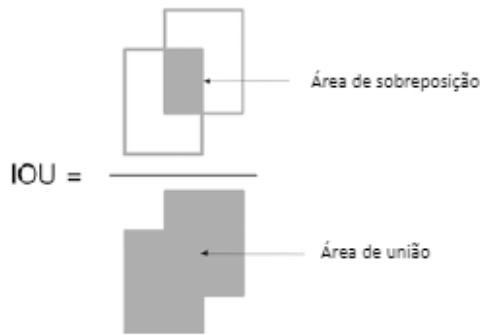


Figura 6. Interpretação visual para a métrica IoU. Fonte: [Michelucci 2019].

A partir do IoU é possível então ter três tipos de resultados:

- **Verdadeiro Positivo (TP, do inglês *True Positive*)**. Detecção correta de uma *bounding box* desejada;
- **Falso Positivo (FP, do inglês *False Positive*)**. Uma detecção incorreta de um objeto inexistente ou uma detecção mal posicionada de um objeto existente;
- **Falso Negativo (FN, do inglês *False Negative*)**. A não detecção de uma *bounding box* desejada.

A partir da quantificação dos valores dos três tipos de resultados, calculam-se então as métricas de Precisão (P) e Revocação (R), dadas como segue:

$$P = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad (3) \quad R = \frac{\text{TP}}{\text{TP} + \text{FN}}. \quad (4)$$

A precisão e a revocação costumam ser sintetizadas em uma única métrica denominada *F₁-Score*, a qual contempla a média harmônica dos dois valores previamente mencionados, calculada como segue:

$$F_1\text{-Score} = \frac{2 \times P \times R}{P + R}. \quad (5)$$

A curva da precisão *versus* revocação, denominada curva ROC, pode ser compreendida como o balanço entre a precisão e a revocação associadas às *bounding boxes* geradas por um detector. Um detector de objetos é considerado bom quando a precisão se mantém alta à medida que a revocação aumenta, o que significa que, à medida que o limiar de confiança varia, tanto a precisão quanto a revocação se mantêm altas. Assim, a área sob a curva ROC (AUC ROC) tende a refletir tais características de um detector.

Em muitas situações, porém, a curva ROC possui um formato serrilhado (zig-zag), o que dificulta o cálculo do AUC ROC. Para contornar esta dificuldade, tem-se a métrica *Average Precision* (AP) que é obtida a partir da média dos valores de uma interpolação de 11 pontos igualmente distribuídos na ($\{0, 0.1, \dots, 0.9, 1\}$) para diferentes níveis de revocação, calculada como segue:

$$AP = \frac{1}{11} \sum_{R \in \{0, 0.1, \dots, 0.9, 1\}} P_{\text{interp}}(R), \quad (6)$$

em que $P_{\text{interp}}(R) = \max_{\tilde{R}: \tilde{R} > R} P(\tilde{R})$. A média da AP (mAP, do inglês *Mean Average Precision*) denota a média aritmética ponderada do AP de todas as classes presentes:

$$mAP = \frac{1}{N} \sum_{i=1}^N AP_i, \quad (7)$$

em que AP_i denota o valor do AP da i -éssima classe e N é o número total de classes. O AP e o mAP são amplamente utilizadas na avaliação de sistemas de detecção de objetos, inclusive tendo sido adotados em competições relevantes da área, tais como o *Open Image Dataset*, *Pascal VOC Challenge* e *MS COCO Challenge* [Kuznetsova et al. 2020, Everingham et al. 2010].

2.3.2. Arquiteturas de CNNs para Tarefas de Detecção

Ao longo do tempo, várias melhorias na metodologia e estratégia de aprendizagem das CNNs foram realizadas visando torná-las escaláveis e eficazes para problemas de VC mais complexos, especialmente envolvendo múltiplas classes. Tais melhorias contemplam modificações de unidades de processamento, estratégias de otimização de parâmetros e hiperparâmetros, padrões de projeto e conectividade de camadas, etc. Estas melhorias impactam na capacidade de representação das CNNs, especialmente em decorrência das inovações arquiteturais, com a reestruturação das unidades de processamento e a concepção de novos blocos [Khan et al. 2020].

Reconhecer objetos e localizá-los em imagens são consideradas tarefas desafiadoras de VC e, para resolvê-los, há diferentes abordagens, conjuntos de técnicas e mudanças arquiteturais disponíveis na literatura. A *Region-Based CNN* (R-CNN) [Girshick et al. 2014], por exemplo, é uma arquitetura para tarefas de detecção cuja abordagem para resolver este problema é dividida em três módulos, conforme ilustrados na Fig. 7, os quais encontram-se listados a seguir:

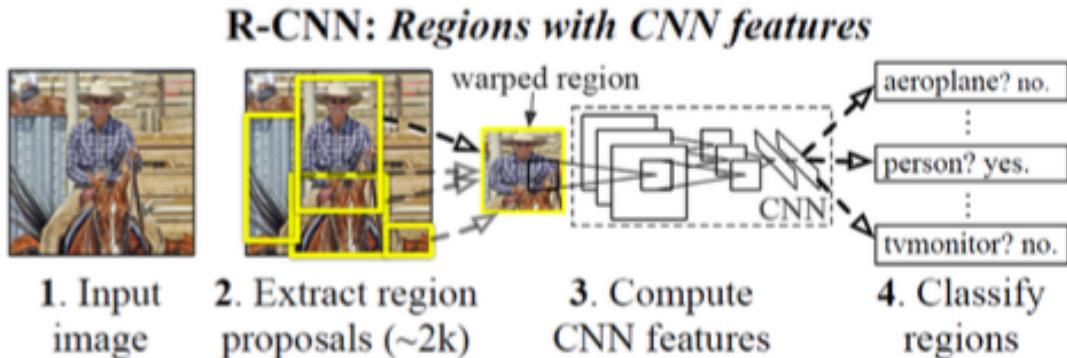


Figura 7. Ilustração dos módulos componentes de uma R-CNN. Fonte: [Girshick et al. 2014].

1. Em seu primeiro módulo, utiliza uma técnica de divisão para produzir regiões independentes, inicialmente pré-configurando com 2.000 regiões, por meio de um algoritmo chamado *selective search*, que busca segmentar uma imagem, utilizando a intensidade dos *pixels* e métodos baseados em grafos. Em seguida, regiões adjacentes são agrupadas mediante critérios de similaridades de textura e cor, passando a constituir as chamadas de *region of proposals* e, por fim, a última etapa classifica as regiões geradas a partir dos passos anteriores;
2. O segundo módulo de uma R-CNN é uma CNN profunda, que é usada para extrair um vetor de características 4096-dimensional de cada região;
3. O terceira e último módulo recebe as características extraídas da CNN e utilizada SVMs treinadas para cada classe para que então seja realizada a classificação dos objetos previamente localizados [Michelucci 2019].

Segundo os autores, na base de dados PASCAL VOC 2012 esta rede alcançou *mAP* igual a 53.3% [Girshick et al. 2014]. Apesar deste desempenho, uma das desvantagens das R-CNNs é o seu custo computacional, pois cada região proposta é sujeita à extração de características de uma CNN e, em seguida, são armazenadas em disco, o que demanda espaço em disco e causa baixo desempenho na detecção de objetos em tempo de teste.

Para contornar os problemas das R-CNNs, uma contrapartida elaborada pela literatura foram as redes *Fast R-CNNs*. Ao invés de fornecer um conjunto de regiões para o módulo de CNN tal como nas R-CNNs, esta arquitetura fornece a imagem original para que a CNN produza então um mapa de características. A partir deste mapa, são identificadas as *region of proposals*, as quais são transformadas em matrizes quadradas por meio de uma camada de *pooling* de *Region Of Interest* (RoI), para que sejam então alimentadas para classificação por meio de uma camada densa, conforme ilustrado na Fig. 8. As *Fast R-CNNs* são mais rápidas que a proposta anterior pois não há a necessidade de fornecer 2.000 *region of proposals* como entrada para uma CNN a cada iteração, contornando este processo custoso com apenas uma convolução por imagem, o que culminou em maior rapidez e maior *mAP* quando aplicada ao conjunto de dados PASCAL VOC 2012 [Girshick 2015].

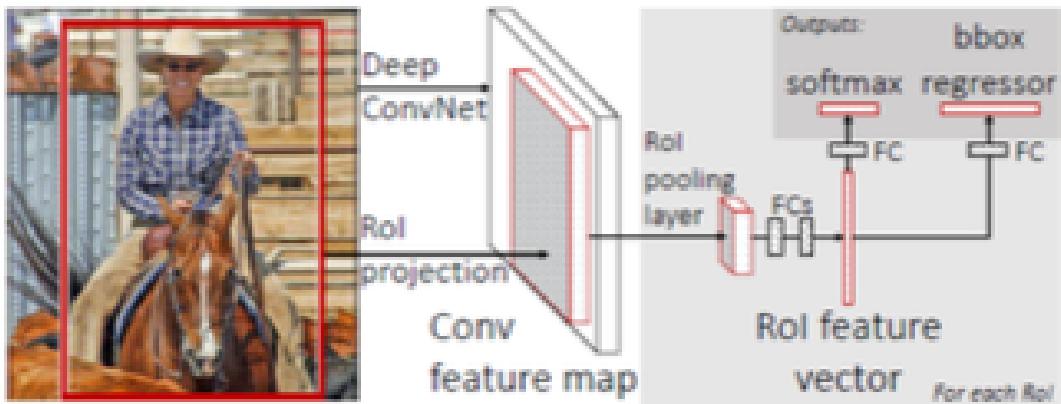


Figura 8. Ilustração dos passos para detecção com uma *Fast R-CNN*. Fonte: [Girshick 2015].

A rede *Regional Proposal Network* (RPN) foi concebida para simultaneamente prever o *bounding box* dos objetos e os respectivos *scores* de classificação a cada posição. Este modelo consiste em uma rede completamente convolucional a qual é treinada com vistas a produzir *region proposals* de alta qualidade para detecção de objetos. Ao mesclar uma RPN com uma *Fast R-CNN* tem-se a proposta da arquitetura *Faster R-CNN*, ilustrada na Figura 9, na qual a RPN compartilha camadas com uma rede de classificação, diminuindo drasticamente o custo computacional ao passo que aumenta a eficiência na detecção [Ren et al. 2017, Michelucci 2019]. Segundo seus autores, quando avaliada no *dataset* PASCAL VOC 2007 alcançou *mAP* igual a 59,9%, ao passo que no *dataset* PASCAL VOC 2012 alcançou resultados comparáveis aos estado da arte.

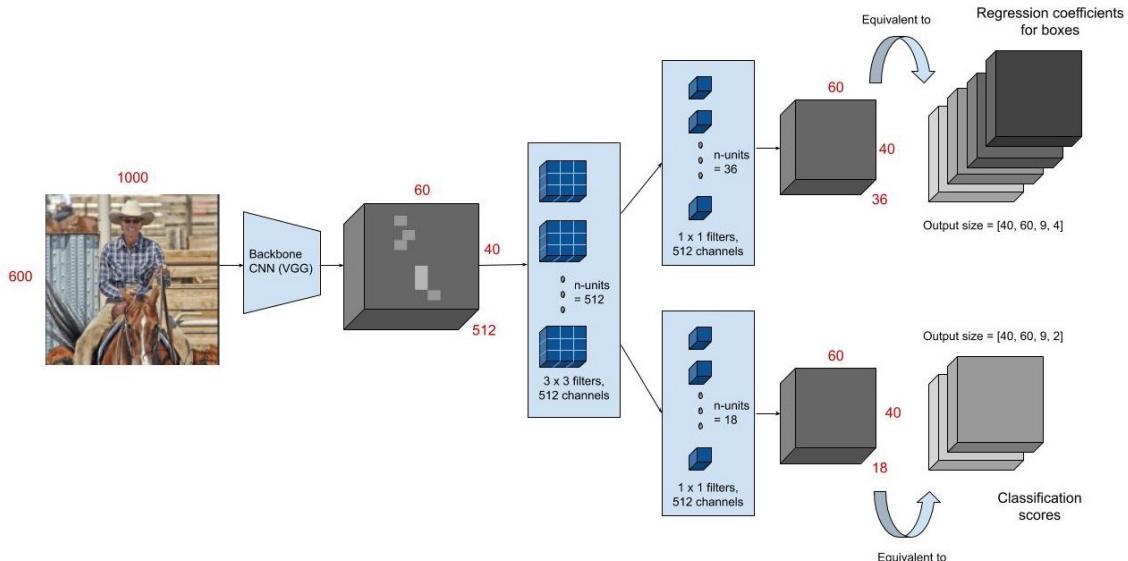


Figura 9. Ilustração dos passos para detecção com uma *Faster R-CNN*. Fonte: [Ren et al. 2017].

Na proposta *You Only Look Once* (YOLO) foi desenvolvida uma abordagem na qual uma rede consegue realizar todas as tarefas necessárias (detecção e classificação de múltiplos objetos) em uma única etapa (*single shot*). Para tanto, aborda a detecção de objetos como um problema de regressão, utilizando uma única rede neural para prever as *bounding boxes* e as classes dos objetos [Redmon et al. 2016]. Em decorrência dessa estratégia, a YOLO classifica de forma rápida e é amplamente utilizado em aplicações em tempo real [Michelucci 2019], superando o desempenho das arquiteturas previamente mencionadas nesse nicho. Na ocasião de sua criação, representava o estado da arte em detecção de objetos em tempo real, sendo capaz de detectar objetos em até 45 quadros por segundo [Redmon et al. 2016].

Conforme ilustrado na Fig.10, o *pipeline* da YOLO primeiramente divide uma imagem em uma grade de células com dimensões $S \times S$, em que cada célula é responsável por detectar o objeto cujo centro esta nela localizado. Um coeficiente de confiança é obtido multiplicando dois elementos: a probabilidade da célula conter o objeto e a métrica IoU (*intersection over union*), a qual mostra quão acurada está uma *bounding box* que contém tal objeto. Para célula são previstas B *bounding boxes* e um vetor C -dimensional com as probabilidades associadas às C classes do problema [Reis 2019, Redmon et al. 2016].

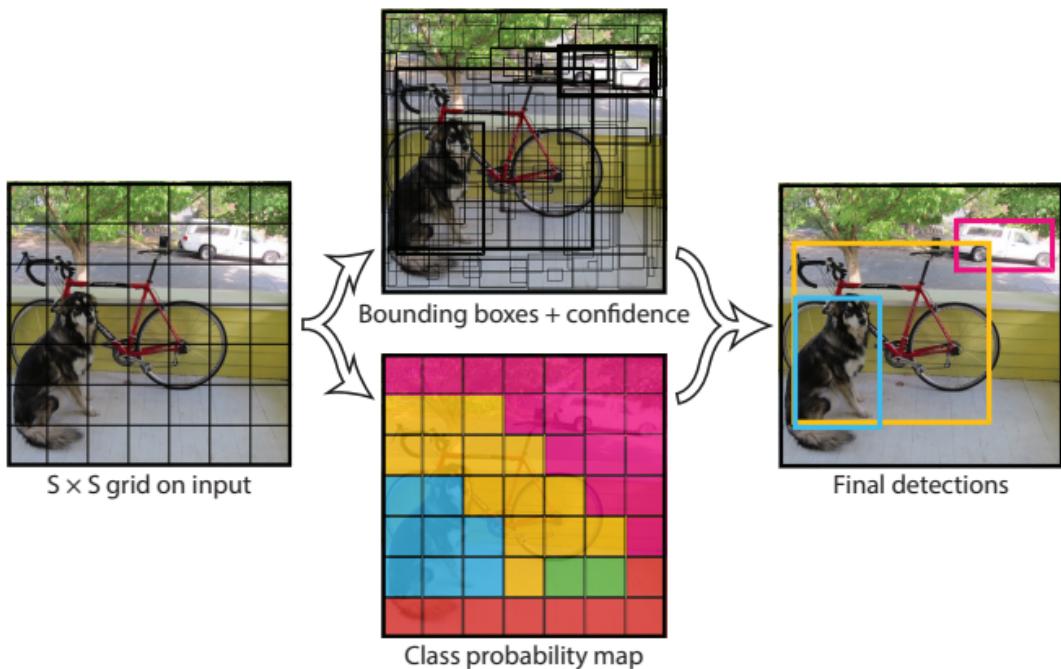


Figura 10. Divisão da imagem em células e previsão de *bounding boxes* na YOLO. Fonte: [Redmon et al. 2016].

Em sua proposta original, a extração de características na YOLO é feita com uma CNN inspirada na arquitetura Inception pré-treinada no *dataset* ImageNet. As 20 primeiras camadas convolucionais são consideradas, mas introduz-se uma camada de *pooling* com filtros 1×1 para redução de dimensionalidade e então camadas densas

[Jiao et al. 2019], conforme ilustrado na Fig. 11.

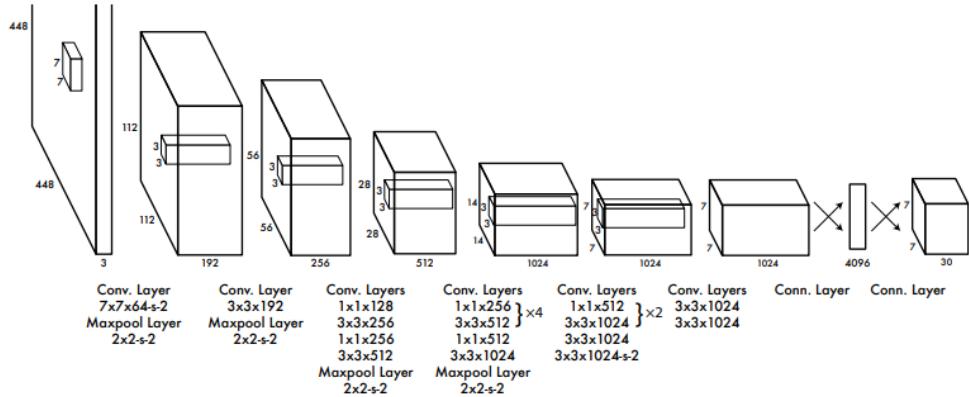


Figura 11. Arquitetura da rede neural presente na YOLO. Fonte:
[Redmon et al. 2016].

A YOLO destacou-se bastante na literatura em especial para detecção em tempo real, mas experimentos mostraram que suas principais fragilidades residiam na baixa precisão da localização [Jiao et al. 2019]. Assim, decisões de projeto foram consideradas visando melhorar o projeto da YOLO, resultando em novas versões desta arquitetura.

A YOLOv2, proposta em 2016, considera várias melhorias em relação à sua primeira versão, a citar: adição de camadas de *batch normalization*, as quais facilitam a convergência da rede e ajudam a regularizar o modelo, prevenindo *overfitting*; utilização de um classificador de alta resolução, com dimensões 448×448 que é treinado por 10 épocas no *dataset* ImageNet para ajuste fino de parâmetros; enquanto a YOLO utiliza as coordenadas previstas das *bounding boxes* para classificação do objeto, a YOLOv2 utiliza convoluções com *anchor boxes* nesta classificação, o que provoca uma melhora na revocação do modelo; e, por fim, a combinação de características de alta e baixa resolução extraídas da imagem [Redmon and Farhadi 2017]. Além disso, a YOLOv2 propõe um novo *backbone* de classificação: uma rede denominada Darknet-19, com 19 camadas convolucionais e 5 camadas de *max-pooling* que requerem menos operações para processar uma imagem, mas ainda atingindo alta precisão [Jiao et al. 2019]. O autor proponente da YOLOv2 realizou experimentos comparativos desta arquitetura com outras previamente existentes utilizando para treinamento uma junção das bases de dados PASCAL VOC 2007 e 2012 e com aferição de desempenho na porção de testes da base PASCAL VOC 2007. Os resultados obtidos, reproduzidos na Tabela 2, comprovam uma melhoria significativa de desempenho perante as alternativas existentes e a possibilidade de lidar com diferentes resoluções das imagens de entrada.

A YOLOv3, uma melhoria posterior na YOLOv2, contempla a utilização da DarkNet-53, a qual é inspirada na ResNet, contendo 53 camadas convolucionais e camadas residuais, considerado um extrator de características mais robusto que o DarkNet-19 [Redmon and Farhadi]. A última camada convolucional da YOLOv3 prevê um tensor tridimensional que codifica as previsões das classes, as probabilidade de haver objetos e as *bounding boxes* [Jiao et al. 2019]. Uma das vantagens da YOLOv3 é que esta arqui-

Tabela 2. Comparativo de desempenho de detecção de objetos no PASCAL VOC 2007. Fonte: [Redmon and Farhadi 2017].

Arquitetura	mAP	FPS
Fast R-CNN	70,0	0,5
Faster R-CNN VGG16	73,2	7
Faster R-CNN ResNet	76,4	5
YOLO	63,4	45
YOLOv2 288 × 288	69,0	91
YOLOv2 352 × 352	73,7	81
YOLOv2 416 × 416	76,8	67
YOLOv2 480 × 480	77,8	59
YOLOv2 544 × 544	78,6	40

tetura demonstrou melhorias de desempenho em relação à versão anterior, mas sem ônus significativo no tempo de treinamento.

A YOLOv4, desenvolvida a partir de melhorias na YOLOv3, foca em melhorias na velocidade de inferência e na acurácia. Para tanto, foi projetada para treino exclusivamente com aceleração em *hardware* via GPU, considera uma nova abordagem de aumento artificial de dados (*data augmentation*) chamada *Mosaic* e também novas estratégias para extração de características e regularização [Bochkovskiy et al. 2020]. Segundo seus proponentes, quando comparada no *dataset* MS COCO, os resultados experimentais são significativamente melhores que as contrapartidas avaliadas quando considerado um cenário de detecção em tempo real, com ganho significativo em relação à sua antecedente YOLOv3, conforme mostrado na Figura 12.

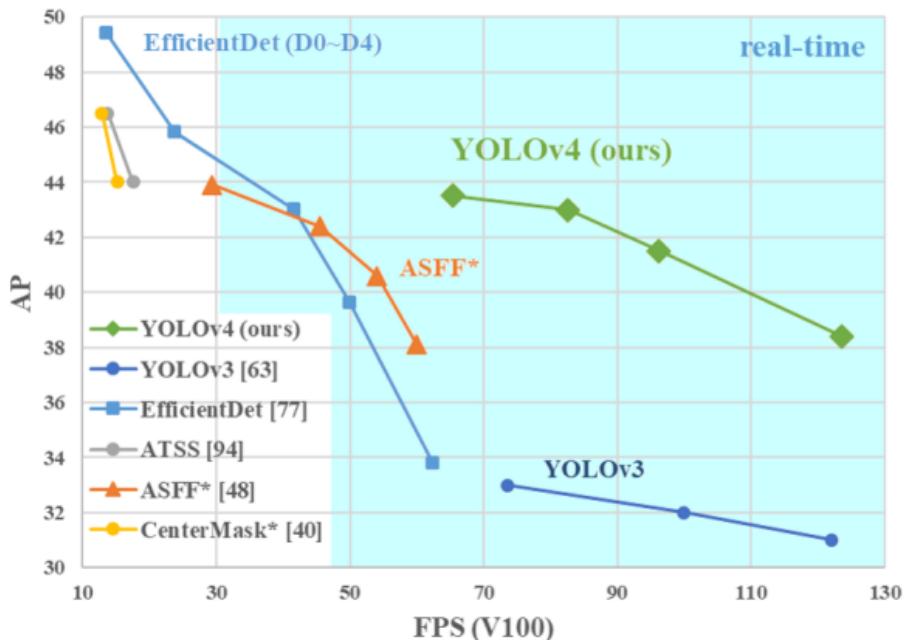


Figura 12. Comparação da YOLOv4 com outras arquiteturas na tarefa de detecção no *dataset* MS COCO. Fonte: [Bochkovskiy et al. 2020].

2.4. Trabalhos Relacionados

A base de dados RDD2020 [Arya 2021, Arya et al. 2021b] foi utilizada em uma competição de Visão Computacional promovida pelo IEEE, denominada *Global Road Damage Detection Challenge* (GRDDC), a qual contou com a participação de 121 equipes e cujas 12 melhores soluções foram sintetizadas em um artigo que reporta o estado da arte para este problema [Arya et al. 2020]. O critério para ranqueamento das equipes foi a métrica F_1 -Score avaliada em dois conjuntos de testes cujos rótulos não estavam publicamente disponíveis para os participantes.

Analizando os resultados reportados por Arya *et al.*, é possível verificar que não houve nenhum time da América Latina dentre as melhores soluções e que as equipes envolveram múltiplos competidores tanto do meio acadêmico quanto profissional. Cerca de 67 % das equipes utilizaram ou consideraram YOLO (YOLOv3, YOLOv4, etc.) na elaboração das soluções. Além disso, as 3 melhores soluções fizeram uso de comitês de modelos. A equipe vencedora, composta por integrantes de universidades dos Estados Unidos e da Jordânia, utilizou um comitê de modelos, aumento artificial de dados e a arquitetura YOLOv5, a qual ainda não foi documentada em publicações técnico-científicas, alcançando F_1 -Score igual a 0,674 e 0,666 nas duas etapas de testes da competição [Arya et al. 2020].

3. Materiais e Métodos

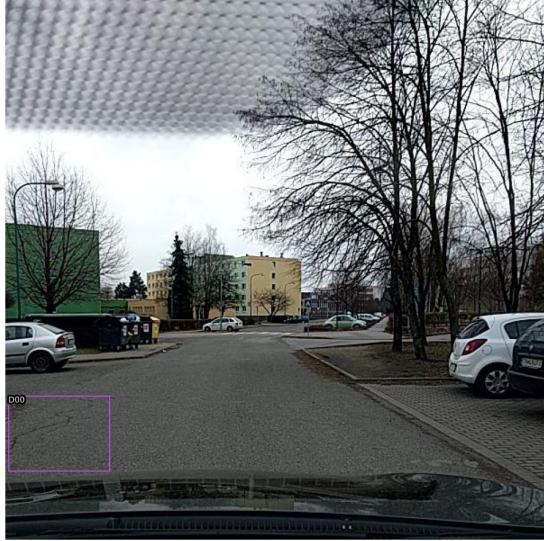
O problema de Visão Computacional considerado no escopo deste trabalho será abordado como uma tarefa de detecção de objetos mediante Aprendizado Supervisionado. Para o desenvolvimento de uma solução para a mesma será utilizada uma estratégia de validação cruzada *holdout* a partir dos dados experimentais e modelos descritos nas seções a seguir. As métricas para aferição de desempenho também são apresentadas ao longo dessa seção.

3.1. Dados Experimentais

Como fonte de experiência para a tarefa considerada foi utilizado o *dataset* RDD2020 [Arya 2021, Arya et al. 2021b], o qual possui originalmente 26.336 imagens de estradas de três diferentes países, a citar: República Tcheca, Índia e China. Neste *dataset* há mais de 31.000 instâncias de falhas de quatro diferentes classes:

1. **Classe D00.** Rótulo associado à trincas longitudinais (*longitudinal cracks*), as quais são predominantemente paralelas ao eixo da via, sendo consideradas um dano funcional e estrutural no pavimento;
2. **Classe D10.** Rótulo associado à trincas transversais (*transverse cracks*), as quais são perpendiculares ao eixo da via e causam danos funcionais e estruturais no pavimento;
3. **Classe D20.** Rótulo associado à trincas em malha tipo “couro de jacaré” (*alligator cracks*), as quais são múltiplas, interligadas e sem direções definidas. Representam um dano estrutural tipicamente característico do fim da vida útil do asfalto;
4. **Classe D40.** Rótulo associado à buracos (*potholes*), isto é, cavidades de tamanhos variados no revestimento do pavimento [CNT 2018].

A Figura 15 ilustra exemplos de detecções das quatro classes presentes em imagens da base de dados.



(a) D00



(b) D10



(c) D20



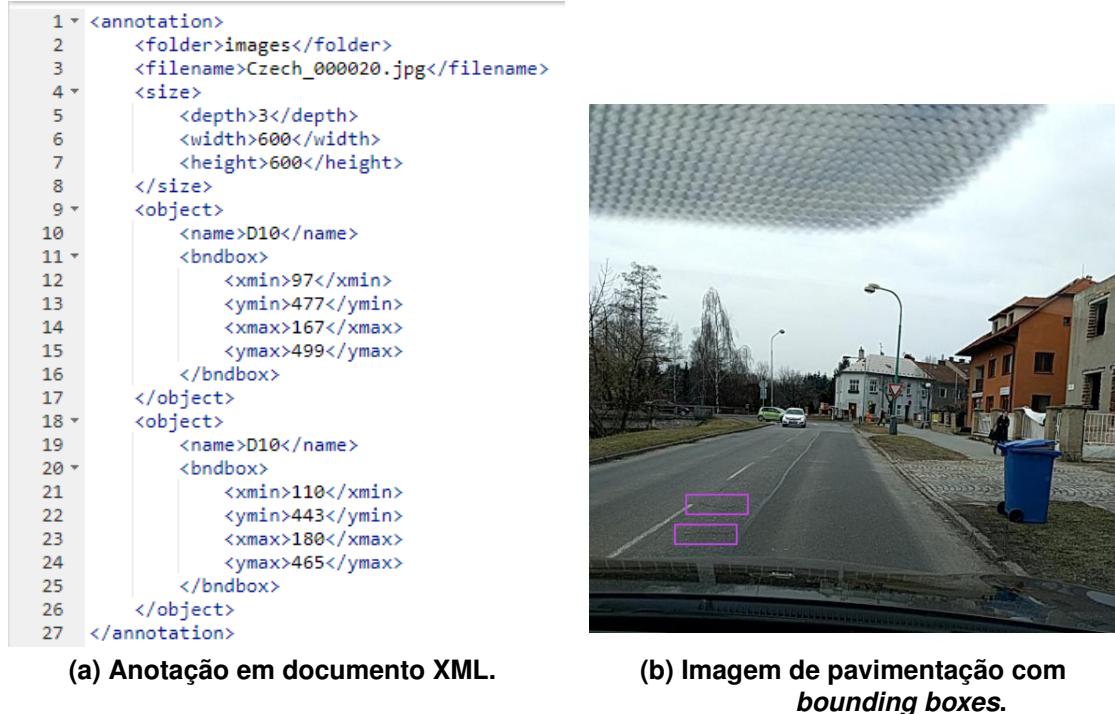
(d) D40

Figura 13. Exemplos de imagens da base de dados RD2020 ilustrando as quatro classes a serem detectadas.

A base de dados RD2020 é disponibilizada segundo uma estrutura de pastas cujos exemplos estão divididos por país, conforme ilustrado na Figura 14, em que as imagens são do tipo JPEG coloridas e em diferentes dimensões. As informações para detecção estão organizadas segundo o padrão PASCAL VOC [Everingham et al. 2010], em que cada imagem possui uma anotação correspondente no formato XML, conforme ilustra a Figura 15a, a qual informa as classes das instâncias presentes na imagem e a respectiva localização das *bounding boxes* que, quando inseridas na imagem correspondente, resultam no artefato ilustrado na Figura 15b.



Figura 14. Estrutura de arquivos do dataset RD2020.



processamento e análise exploratória da mesma visando identificar a quantidade de classes e sua distribuição, cujos resultados são ilustrados no histograma da Figura 16. Em seguida, foram consideradas apenas as imagens com rótulos disponíveis, contabilizando 21.041 itens. Desse conjunto, foram desconsideradas ocorrências de classes não pertencentes ao conjunto indicado anteriormente (D00, D10, D20 e D40), por não ser possível determinar seu impacto nos resultados subsequentes devido à falta de informação sobre as mesmas. Além disso, foi adicionada uma nova classe, rotulada como “0”, que indica que a respectiva imagem não possui instâncias de falhas na pavimentação. Ao final deste pré-processamento, conforme ilustra o histograma da Fig. 17, tem-se 18.667 imagens com 25.046 ocorrências de falhas. Essas imagens e rótulos serão utilizados nas próximas etapas do trabalho.

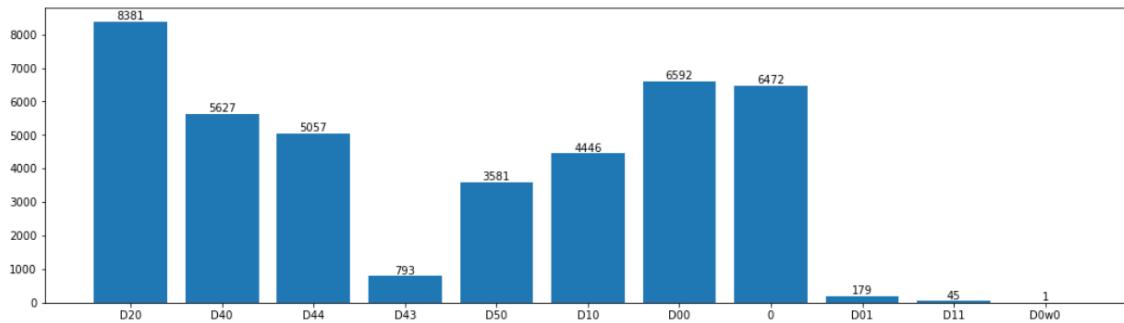


Figura 16. Histograma com a quantidade de classes e sua distribuição no dataset RD2020.

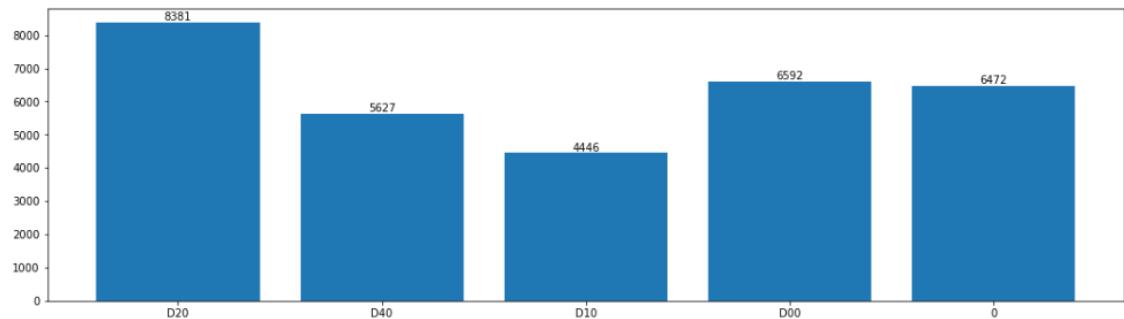


Figura 17. Histograma com a quantidade de classes e sua distribuição após pré-processamento do dataset.

3.2. Arquiteturas, Parâmetros e Hiperparâmetros

A arquitetura considerada nesta proposta de trabalho de conclusão de curso foi a YOLOv4 [Bochkovskiy et al. 2020], por ser a mais recente dentre as atualizações da YOLO abordadas na fundamentação teórica. Conforme a documentação oficial [Redmon 2016], foi utilizada a implementação desenvolvida por seus proponentes com o *framework* de código aberto Darknet para redes neurais, desenvolvido em C e CUDA, promovendo assim rapidez e suporte ao uso de GPUs.

Para evitar problemas como *underfitting* e *overfitting*, foi utilizada a técnica de regularização *Early Stopping* que, ao final de cada época de treinamento, verifica se houve

melhora na generalização no conjunto de validação. Em caso positivo, o treino continua normalmente e, em caso contrário, desconta-se de um parâmetro denominado paciência até que este seja igual a zero, o que causa o encerramento do treino. Nesta arquitetura, foram utilizados pesos pré-treinados na base de dados MS COCO, disponibilizados pelos autores da arquitetura. A estratégia de utilização de pesos pré-treinados é amplamente sugerida por considerar o aproveitamento de características já aprendidas pela arquitetura na base de dados de origem.

No tocante aos hiperparâmetros, considerou-se quatro configurações diferentes, as quais encontram-se listadas a seguir:

1. **Configuração 1.** Tamanho do *batch*: 64; Tamanho máximo do *batch*: 4.000; Taxa de aprendizado: 10^{-2} ; *Steps*: 3.200;
2. **Configuração 2.** Tamanho do *batch*: 64; Tamanho máximo do *batch*: 4.000; Taxa de aprendizado: 10^{-2} ; *Steps*: 6.400;
3. **Configuração 3.** Tamanho do *batch*: 64; Tamanho máximo do *batch*: 4.000; Taxa de aprendizado: 10^{-3} ; *Steps*: 6.400;
4. **Configuração 4.** Tamanho do *batch*: 64; Tamanho máximo do *batch*: 8.000; Taxa de aprendizado: 10^{-3} ; *Steps*: 10.000;

As diferentes configurações basicamente consideram um treino mais curto e outro mais longo com duas taxas de aprendizados distintas.

3.3. Avaliação de Desempenho

Para avaliação de desempenho, como mencionado anteriormente, foi utilizada a técnica de validação cruzada *holdout* com 70% dos dados destinados ao treinamento, 10% para validação e 20% para testes. O quantitativo e a distribuição dos exemplos nestas partições encontram-se ilustrados nas Figuras 18, 19 e 20.

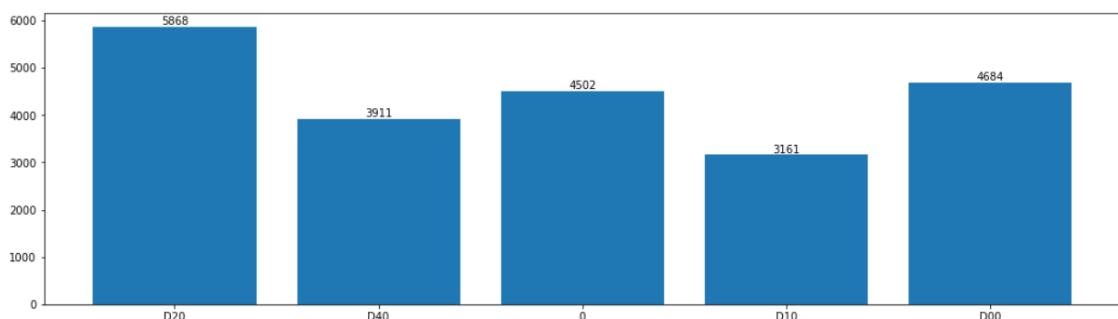


Figura 18. Partição de Treinamento.

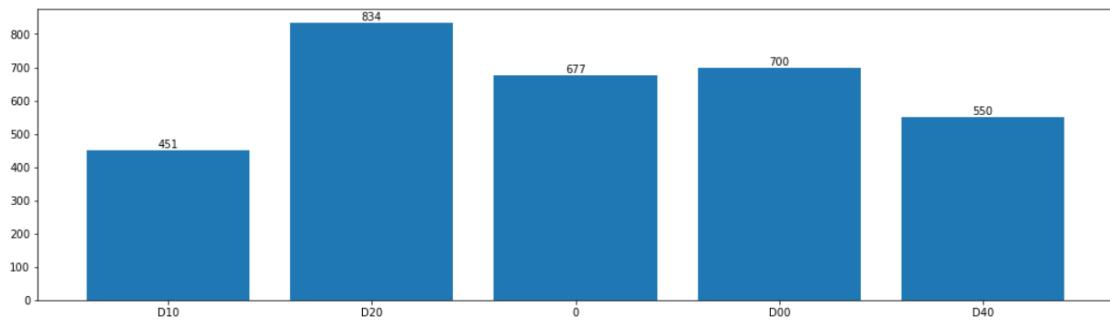


Figura 19. Partição de Validação.

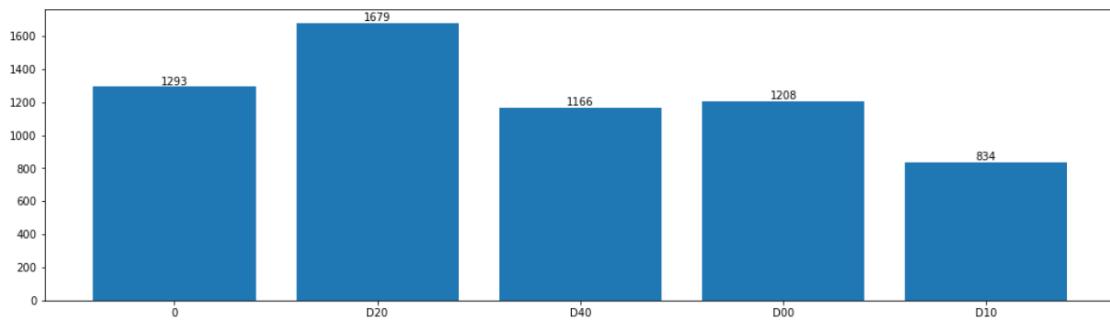


Figura 20. Partição de Teste.

As métricas de desempenho serão aferidas na partição de testes e contemplam: AP para as quatro classes, Precisão, Revocação, IoU e mAP com $threshold t \geq 0.5$ para o IoU, a qual será denotada como mAP@0.5.

4. Resultados Parciais

Para a execução dos *scripts* de treinamento foi utilizado um servidor com a seguinte configuração: processador Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz, 32 GB de memória principal, 960 GB de memória secundária e 2 placas de vídeo NVIDIA GTX 1080 Ti com 11 GB de VRAM para aceleração em *hardware* do treinamento. Os resultados obtidos do treinamento para as diferentes configurações propostas encontram-se dispostos na Tabela 3.

Em face das múltiplas métricas de desempenho aferidas, considerou-se como referência o melhor resultado na métrica mAP@0.5, por unificar aspectos da localização e da classificação. Desta forma, observa-se que a Configuração 3 foi aquela com melhor desempenho no conjunto de testes com $mAP@0.5 = 31,34\%$. É interessante ressaltar que o tempo de treinamento deste modelo foi aproximadamente de 16 h.

Para melhor analisar a capacidade de detecção da YOLOv4 perante a Configuração 3 foram selecionadas algumas imagens da base de dados da partição de testes e os rótulos desejados (*ground truth*) foram comparados com aqueles previstos pela arquitetura após o treino. Ao analisar os exemplos disponíveis nas Figuras 21 e 22 percebe-se que houve uma boa detecção das Classes D00 e D10, o que é reforçado pelos respectivos APs. Embora o AP_{D20} tenha sido o maior dentre as classes consideradas,

Tabela 3. Síntese dos resultados experimentais.

Métrica	Configuração 1	Configuração 2	Configuração 3	Configuração 4
AP_{D00}	30,88 %	22,62 %	31,11 %	26,16 %
AP_{D10}	24,83 %	17,93 %	20,04 %	21,27 %
AP_{D20}	39,51 %	36,35 %	44,93 %	40,39 %
AP_{D40}	29,87 %	18,27 %	29,62 %	25,82 %
Precisão	0,66	0,57	0,59	0,56
Revocação	0,18	0,12	0,22	0,23
F_1 -Score	0,28	0,19	0,32	0,33
Average IoU	46,62 %	38,73 %	41,09 %	39,04 %
mAP@0.5	31,27 %	23,79 %	31,43 %	28,41 %

os desafios na detecção desta classe foram mais evidentes. Na Figura 23, por exemplo, o rótulo desejado para a classe D20 contempla uma área à margem do asfalto com vegetação e que, muito provavelmente por esse motivo, não foi detectada corretamente pelo modelo (falso negativo), apesar da instância D40 interna à D20 ter sido corretamente identificada. Na Figura 24, por sua vez, há divergência entre o desejado e o previsto no tocante à área da falha na pavimentação.

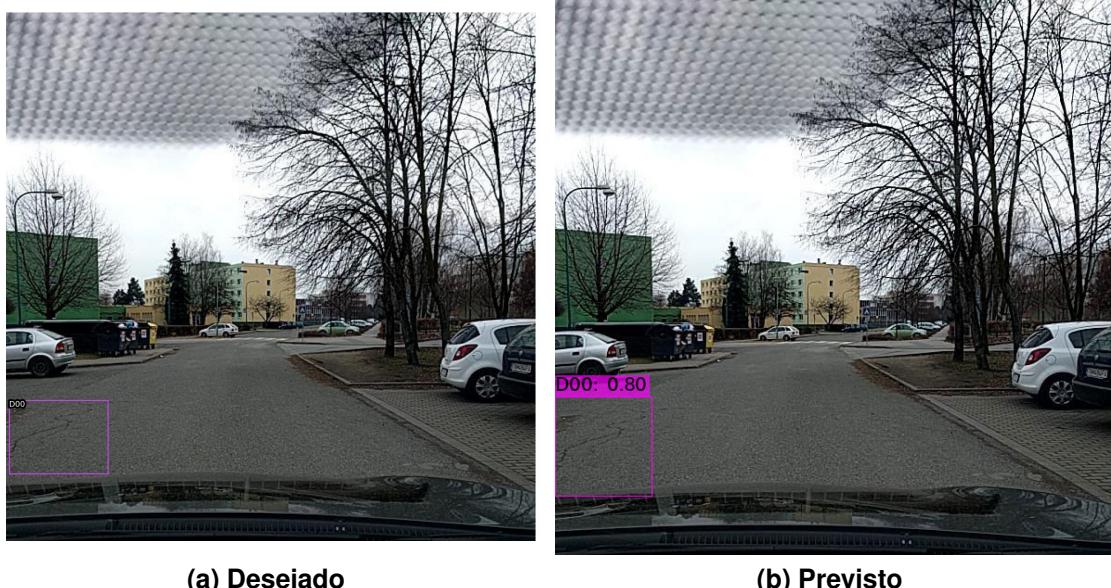


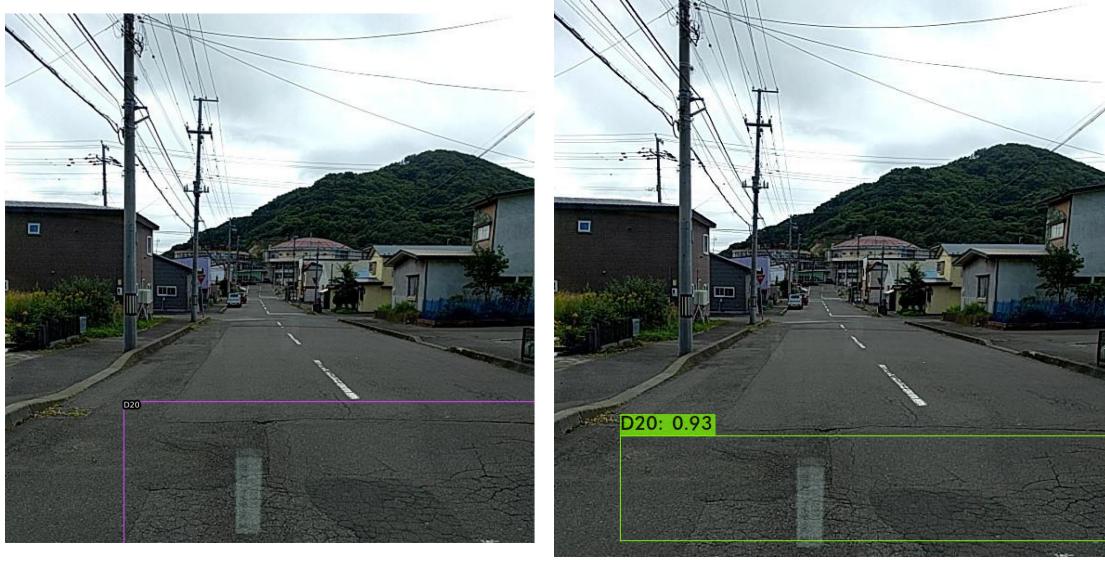
Figura 21. Imagem com instância de classe D00.



Figura 22. Imagem com instância de classe D10.



Figura 23. Imagem com instância de classes D20 e D40.



(a) Desejado

(b) Previsto

Figura 24. Imagem com instância de classe D20.

5. Considerações Parciais

O objetivo dessa proposta de trabalho de conclusão de curso consiste em avaliar o uso de redes neurais convolucionais profundas para a tarefa de detecção de quatro tipos de falhas em pavimentações asfálticas a partir de exemplos da base de dados RDD2020, que possui mais de 18 mil exemplos. Para isto, foi utilizada a arquitetura YOLOv4 [Bochkovskiy et al. 2020] com o *framework* Darknet, implementado na linguagem de programação C. Os melhores resultados obtidos perante uma configuração de treino resultaram em um $mAP@0.5 = 31,43\%$ e F_1 -Score igual a 0,32. Para a obtenção desses resultados, destacou-se como desafio o processo de treinamento utilizando o Darknet, pelo fato de sua implementação estar em uma linguagem de mais baixo nível, com menos abstrações e possibilidades de customização do que outras soluções disponíveis.

Como sequência deste trabalho, serão consideradas, implementadas e avaliadas outras arquiteturas de CNNs, tais como outras versões da YOLO, especialmente utilizando a linguagem *Python* e *frameworks* como Keras, Tensorflow e PyTorch, de forma a contornar as dificuldades enfrentadas nesta primeira etapa e também visando melhoria nas métricas de desempenho da tarefa.

Referências

- Arya, D. (2021). Rdd2020: An image dataset for smartphone-based road damage detection and classification.
- Arya, D., Maeda, H., Ghosh, S. K., Toshniwal, D., Mraz, A., Kashiyama, T., and Sekimoto, Y. (2021a). Deep learning-based road damage detection and classification for multiple countries. *Automation in Construction*, 132:103935.
- Arya, D., Maeda, H., Ghosh, S. K., Toshniwal, D., Omata, H., Kashiyama, T., and Sekimoto, Y. (2020). Global road damage detection: State-of-the-art solutions. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 5533–5539. IEEE.
- Arya, D., Maeda, H., Ghosh, S. K., Toshniwal, D., and Sekimoto, Y. (2021b). RDD2020: An annotated image dataset for automatic road damage detection using deep learning. *Data in Brief*, 36:107133.
- Bengio, Y. (2009). *Learning deep architectures for AI*. Now Publishers Inc.
- Bochkovskiy, A., Wang, C.-Y., and Liao, H.-Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*.
- Chollet, F. (2017). *Deep learning with Python*. Simon and Schuster.
- CNT (2018). Confederação Nacional do Transporte (cnt) – conheça os 13 principais defeitos do pavimento das rodovias. Disponível em <https://www.cnt.org.br/agencia-cnt/conheca-principais-defeitos-pavimento>. Acesso em 2 de agosto de 2023.
- CNT (2019). Confederação Nacional do Transporte (cnt – piora a qualidade das rodovias brasileiras. Disponível em <https://www.cnt.org.br/agencia-cnt/piora-a-qualidade-das-rodovias-brasileiras>. Acesso em 2 de agosto de 2023.
- CNT (2020). Confederação Nacional do Transporte (cnt) – CNT lança painel com dados do transporte rodoviário no Brasil. Disponível em <https://cnt.org.br/agencia-cnt/cnt-lanca-painel-com-dados-do-transporte-rodoviario-no-brasil>. Acesso em 2 de agosto de 2023.
- DNIT (2021). Infraestrutura rodoviária. Disponível em <https://www.gov.br/dnit/pt-br/assuntos/infraestrutura-rodoviaria>. Acesso em 2 de agosto de 2023.
- Evangelista, L. G. C. and Guedes, E. B. (2018). Computer-aided tuberculosis detection from chest x-ray images with convolutional neural networks. In *Anais do XV Encontro Nacional de Inteligência Artificial e Computacional*, pages 518–527. SBC.
- Everingham, M., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338.
- Federal, G. (2021). Rodovias federais - obras públicas - manutenção das rodovias. Disponível em <https://www.gov.br/infraestrutura/pt-br/assuntos/transporte-terrestre/rodovias-federais/rodovias-federais-obra-publicas-manutencao-das-rodovias>. Acesso em 2 de agosto de 2023.

- Gatys, L. A., Ecker, A. S., and Bethge, M. (2015). A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*.
- Girshick, R. (2015). Fast r-CNN. In *2015 IEEE International Conference on Computer Vision (ICCV)*, Santiago, Chile. IEEE.
- Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, Estados Unidos. IEEE.
- Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.
- Haykin, S. (2009). *Neural Networks and Learning Machines*. Pearson Prentice-Hall, Estados Unidos, 3 edition.
- ImageNet (2021). Imagenet large scale visual recognition challenge (ilsvrc). Disponível em <https://www.image-net.org/challenges/LSVRC/index.php>. Acesso em 2 de agosto de 2023.
- Jain, R., Kasturi, R., and Schunck, B. G. (1995). *Machine vision*, volume 5. McGraw-hill New York.
- Janani, R. P., Renuka, K., Aruna, A., and Lakshmi, N. K. (2021). IoT in smart cities: A contemporary survey. *Global Transitions Proceedings*. Disponível em <https://doi.org/10.1016/j.gltip.2021.08.069>. Acesso em 2 de agosto de 2023.
- Jiao, L., Zhang, F., Liu, F., Yang, S., Li, L., Feng, Z., and Qu, R. (2019). A survey of deep learning-based object detection. *IEEE Access*, 7:128837–128868.
- Khan, A., Sohail, A., Zahoor, U., and Qureshi, A. S. (2020). A survey of the recent architectures of deep convolutional neural networks. *Artificial Intelligence Review*, 53(8):5455–5516.
- Khan, S., Rahmani, H., Shah, S. A. A., and Bennamoun, M. (2018). A guide to convolutional neural networks for computer vision. *Synthesis Lectures on Computer Vision*, 8(1):1–207.
- Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105.
- Kuznetsova, A., Rom, H., Alldrin, N., Uijlings, J., Krasin, I., Pont-Tuset, J., Kamali, S., Popov, S., Mallochi, M., Kolesnikov, A., et al. (2020). The open images dataset v4. *International Journal of Computer Vision*, 128(7):1956–1981.
- LeCun, Y., Jackel, L. D., Bottou, L., Cortes, C., Denker, J. S., Drucker, H., Guyon, I., Muller, U. A., Sackinger, E., Simard, P., et al. (1995). Learning algorithms for classification: A comparison on handwritten digit recognition. *Neural networks: the statistical mechanics perspective*, 261(276):2.
- Michelucci, U. (2019). *Advanced applied deep learning: convolutional neural networks and object detection*. Springer.
- Miller, T. R. and Zaloshnja, E. (2009). On a crash course: The dangers and health costs of deficient roadways. Technical Report 01135610. Disponível em <http://www>.

- infrastructureusa.org/on-a-crash-course-the-dangers-and-health-costs-of-deficient-roadways/. Acesso em 2 de agosto de 2023.
- Mohanty, S. P., Choppali, U., and Kougianos, E. (2016). Everything you wanted to know about smart cities: The internet of things is the backbone. *IEEE Consumer Electronics Magazine*, 5(3):60–70.
- NYC (2021). New York City Department of Information Technology & Telecommunications DoITT – who we are – about DoITT. Disponível em <https://www1.nyc.gov/site/doitt/about/who-we-are.page>. Acesso em 2 de agosto de 2023.
- Padilla, R., Netto, S. L., and da Silva, E. A. B. (2020). A survey on performance metrics for object-detection algorithms. In *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, pages 237–242, Niterói, Brasil.
- Redmon, J. (2013–2016). Darknet: Open source neural networks in c. <http://pjreddie.com/darknet/>.
- Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, Estados Unidos. IEEE.
- Redmon, J. and Farhadi, A. Yolov3: An incremental improvement. arXiv:1804.02767. Disponível em <https://arxiv.org/pdf/1804.02767.pdf>; . Acesso em 2 de agosto de 2023.
- Redmon, J. and Farhadi, A. (2017). Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271.
- Reis, L. P. (2019). Detecção e contagem de caixas empilhadas para controle de estoque utilizando deep learning. Trabalho de Conclusão de Curso em Engenharia de Computação. Universidade do Estado do Amazonas.
- Ren, S., He, K., Girshick, R., and Sun, J. (2017). Faster r-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149.
- Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
- Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9.
- Uijlings, J. R., Van De Sande, K. E., Gevers, T., and Smeulders, A. W. (2013). Selective search for object recognition. *International journal of computer vision*, 104(2):154–171.
- Zhang, R., Isola, P., and Efros, A. A. (2016). Colorful image colorization. In *European conference on computer vision*, pages 649–666. Springer.