UNIVERSIDADE FEDERAL DO AMAZONAS - UFAM

INSTITUTO DE COMPUTAÇÃO - ICOMP

PROGRAMA PÓS-GRADUAÇÃO EM INFORMÁTICA - PPGI

OneTrack - Modelos Baseados em Transformers e Eficientes em Tempo de Inferência para Rastreamento de Múltiplos Objetos.

Luiz Carlos Silva de Araújo Filho

Manaus - AM

Agosto 2024

Luiz Carlos Silva de Araújo Filho

OneTrack - Modelos Baseados em Transformers e Eficientes em Tempo de Inferência para Rastreamento de Múltiplos Objetos.

Dissertação de mestrado submetido à Coordenação de Pós-Graduação em Informática do Instituto de Computação (ICOMP), como requisito parcial para obtenção do Título de Mestre em Informática.

Orientador

Carlos Maurício Seródio Figueiredo, Prof. Dr.

Universidade Federal do Amazonas - UFAM

Instituto de Computação - IComp

Manaus - AM

Agosto 2024

Ficha Catalográfica

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

Araujo Filho, Luiz Carlos Silva de
A663o OneTrack - modelos baseados em transformers e eficientes em tempo de inferência para rastreamento de múltiplos objetos. / Luiz Carlos Silva de Araujo Filho . 2024
83 f.: il. color; 31 cm.

Orientador: Carlos Mauricio Seródio Figueiredo
Dissertação (Mestrado em Informática) - Universidade Federal do Amazonas.

1. Mot. 2. Transformers. 3. Rastreamento rápido. 4. Modelo unificado. 5. Rastreamento de múltiplos objetos. I. Figueiredo, Carlos Mauricio Seródio. II. Universidade Federal do Amazonas III. Título

Ministério da Educação
Universidade Federal do Amazonas
Coordenação do Programa de Pós-Graduação em Informática

FOLHA DE APROVAÇÃO

"ONETRACK - MODELOS BASEADOS EM TRANSFORMERS E EFICIENTES EM TEMPO DE INFERÊNCIA PARA RASTREAMENTO DE MÚLTIPLOS OBJETOS"

LUIZ CARLOS SILVA DE ARAÚJO FILHO

DISSERTAÇÃO DE MESTRADO DEFENDIDA E APROVADA PELA BANCA EXAMINADORA CONSTITUÍDA PELOS PROFESSORES:

Prof. Dr. Carlos Maurício Serodio Figueiredo - PRESIDENTE

Prof. Dr. Eduardo Freire Nakamura - MEMBRO INTERNO

Prof. Dr. Tiago Eugênio de Melo - MEMBRO EXTERNO

MANAUS, 18 de julho de 2024.

Documento assinado eletronicamente por **Carlos Maurício Seródio Figueiredo**, **Usuário Externo**, em 15/08/2024, às 17:16, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Tiago Eugenio de Melo**, **Usuário Externo**, em 22/08/2024, às 11:45, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Eduardo Freire Nakamura**, **Professor do Magistério Superior**, em 22/08/2024, às 12:02, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Maria do Perpétuo Socorro Vasconcelos Palheta**, **Secretária**, em 22/08/2024, às 19:08, conforme horário oficial de Manaus, 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.ufam.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **2192764** e o código CRC **1E9868E5**.

Avenida General Rodrigo Octávio, 6200 - Bairro Coroado I Campus Universitário Senador Arthur Virgílio Filho, Setor Norte - Telefone: (92) 3305-1181 / Ramal 1193
CEP 69080-900, Manaus/AM, coordenadorppgi@icomp.ufam.edu.br

Às pessoas que me ajudaram a trilhar este caminho e chegar onde cheguei, dedico este trabalho que marca mais uma etapa da minha vida.

Agradecimentos

Primeiramente, gostaria de agradecer às pessoas mais próximas de mim, especialmente à minha noiva, futura esposa e eterna companheira, Krishna Ribeiro. Seria impossível concluir este trabalho sem seu constante apoio e suporte. Aos meus pais, Rose Mary Silva e Luiz Carlos de Araújo que me ajudaram a chegar onde cheguei, sequer teria começado o percurso acadêmico que hoje sou capaz de trilhar sem eles. À minha irmã Cibele Silva que participou da minha criação como uma segunda mãe e certamente foi minha maior referência durante minha infância e adolescência. À minha sobrinha que espero poder influenciar a experimentar o mundo acadêmico futuramente.

Em seguida, gostaria de direcionar um agradecimento ao meu orientador Carlos Maurício Seródio que me acompanha em minha carreira acadêmica desde o início, e depositou sua confiança na minha capacidade o suficiente para me orientar em todas as principais frentes acadêmicas que participei.

Um agradecimento especial à minha terapeuta, Rosane Texeira, que em vários momentos me ajudou a passar por fases difíceis da minha vida, certamente teve um grande papel para a conclusão desta etapa.

Adicionalmente, direciono meus agradecimentos a Universidade Federal do Amazonas como instituição que proporcionou espaço e recursos que incentivam o desenvolvimento científico para a região Norte e para o Brasil. Extendo meus agradecimentos aos professores que tive contato nesta etapa de mestrado na UFAM, mas todos foram excelentes em suas funções de educadores e fomentadores do processo de pesquisa na universidade.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento

de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001. Este trabalho foi parcialmente financiado pela Fundação de Amparo à Pesquisa do Estado do Amazonas – FAPEAM – por meio do projeto POSGRAD.

OneTrack - Modelos Baseados em Transformers e Eficientes em Tempo de Inferência para Rastreamento de Múltiplos Objetos.

Autor: Luiz Carlos Silva de Araújo Filho

Orientador: Carlos Maurício Seródio Figueiredo, Prof. Dr.

Resumo

O Rastreamento de Múltiplos Objetos (MOT) é um problema crítico na visão computacional, essencial para entender como objetos se movem e interagem em vídeos. Este campo enfrenta desafios significativos, oclusões e dinâmicas ambientais complexas afetam a precisão e eficiência dos modelos. Enquanto abordagens tradicionais têm se apoiado em Redes Neurais Convolucionais (CNNs), este trabalho apresenta o OneTrack-M, um modelo MOT baseado em transformers, projetado para aumentar a eficiência computacional e a precisão do rastreamento. Nossa abordagem simplifica a arquitetura típica baseada em transformers, ao eliminar a necessidade de um modelo decodificador para detecção e rastreamento de objetos. Ao invés disso, apenas o codificador serve como base para a interpretação de dados temporais, reduzindo significativamente o tempo de processamento e aumentando a velocidade de inferência. Paralelamente, emprega-se técnicas inovadoras de pré-processamento de dados e treinamento multitarefa para abordar desafios diversos de objetivos dentro de um único conjunto de pesos. Resultados experimentais demonstram o OneTrack-M alcançando tempos de inferência pelo menos 25% mais rápidos em comparação com modelos de ponta na literatura, mantendo ou melhorando as métricas de precisão de rastreamento. Essas melhorias destacam o potencial da solução proposta para aplicações em tempo real,

como veículos autônomos e sistemas de vigilância, tais respostas rápidas são cruciais para a eficácia do sistema.

Palavras-chave: MOT, Rastreamento de Múltiplos Objetos, Transformers, Rastreamento Rápido, End2End, Modelo Unificado.

OneTrack - Modelos Baseados em Transformers e Eficientes em Tempo de Inferência para Rastreamento de Múltiplos Objetos.

Autor: Luiz Carlos Silva de Araújo Filho

Orientador: Carlos Maurício Seródio Figueiredo, Prof. Dr.

Abstract

Tracking Multiple Objects (MOT) is a critical problem in computer vision, essential for understanding how objects move and interact in videos. This field faces significant challenges, as occlusions and complex environmental dynamics affect the accuracy and efficiency of models. While traditional approaches have relied on Convolutional Neural Networks (CNNs), this work presents OneTrack-M, a MOT model based on transformers, designed to enhance computational efficiency and tracking accuracy. Our approach simplifies the typical transformer-based architecture by eliminating the need for a decoder model for object detection and tracking. Instead, only the encoder serves as the basis for interpreting temporal data, significantly reducing processing time and increasing inference speed. In parallel, innovative data preprocessing techniques and multitask training are employed to address various objectives within a single set of weights. Experimental results demonstrate that OneTrack-M achieves inference times at least 25% faster compared to state-of-the-art models in the literature, while maintaining or improving tracking accuracy metrics. These improvements highlight the proposed solution's potential for real-time applications, such as autonomous vehicles and surveillance systems, where quick responses are crucial for system effectiveness.

Keywords: MOT, Multiple Object Tracking, Transformers, Fast Tracking, End2End, Unified Model.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1

INTRODUÇÃO

A visão computacional simula a natureza complexa da percepção visual humana, permitindo que máquinas interpretem dados visuais, levando ao desenvolvimento de tecnologias avançadas. O progresso desta é impulsionado por melhorias algorítmicas e de hardwares, como CPUs e GPUs, facilitando o processamento de grandes volumes de dados, bem como a execução de modelos complexos de aprendizado de máquina (*Machine Learning* - ML) e aprendizado profundo (*Deep Learning* - DL).

Entre as especificações de visão computacional, o Rastreamento de Múltiplos Objetos (*Multiple Object Tracking* - MOT) destaca-se em aprender como os objetos se movem e interagem em vídeos. Sendo assim a tarefa se resume ao processo de encontrar objetos e manter seus trajetos ao longo do tempo em um cenário de vídeo. Este domínio enfrenta desafios como oclusões, dinâmicas ambientais e funcionamento em tempo real, tradicionalmente abordados por uma progressão de modelos, desde métodos baseados em características até algoritmos avançados de ML e DL.

As Redes Neurais Convolucionais (*Convolutional Neural Networks* - CNNs) têm sido utilizadas para detecção e classificação de objetos, por capturar padrões visuais, tanto simples quanto complexos, através do processo de convolução que fornece contexto espacial sobre as imagens. Por outro lado, o uso de transformers supera as CNNs em tarefas como: classificação de objetos (Dosovitskiy et al., 2021), e processamento de vídeo (Liu et al., 2021b), devido a capacidade destes de modelar relacionamentos de longo alcance nos dados, tornando-os ideais para entender contextos e relações temporais entre objetos. Ambos apresentam características essenciais para o MOT (Wojke

et al., 2017).

As soluções que empregam CNNs como *backbone* tentam utilizar esta capacidade de extração de características de imagens para realizar o rastreamento. Esta utilização é lógica, já que esta tecnologia foi desenvolvida para o tratamento de imagens. No entanto esta por si só não é capaz, inerentemente, de tratar vídeos, sendo necessário agregar outras estruturas à arquitetura. Já os modelos baseados em atenção, originalmente desenvolvidos para processamento de linguagem natural, apresentam potencial em aplicações de MOT, oferecendo novas possibilidades para analisar e interpretar sequências de imagens. No entanto, os transformers enfrentam desafios relacionados à eficiência, tanto computacionalmente quanto em precisão de rastreamento, impulsionando a busca por soluções inovadoras.

Logo, reduzir o tempo de inferência permanece um desafio comum no MOT. Alguns dos principais trabalhos que utilizam transformers neste cenário apresentam problemas de baixa velocidade, como Zeng et al. (2022), Sun et al. (2021), e Meinhardt et al. (2022). Além disso, outros modelos baseados em CNNs precisam de ferramentas como filtro de Kalman (Bewley et al., 2016) para realizar a associação de objetos ao longo do tempo, não sendo possível um modelo fim-a-fim.

Devido a isso, há necessidade em pesquisar formas de reduzir os tempos de inferência em sistemas de MOT, que é enfatizada por aplicações que dependem de processamento em tempo real. Tais como: veículos autônomos dependentes de respostas rápidas para evitar obstáculos e prevenir colisões, e sistemas de vigilância que requerem processamento rápido para detectar anomalias e rastrear indivíduos de forma eficaz.

Este trabalho apresenta uma abordagem para aumentar a eficiência de sistemas MOT com transformers, aproveitando-os para melhorar a acurácia enquanto mantém velocidades de inferência compatíveis com casos de uso reais.

A abordagem simplifica uma etapa comum em modelos transformers ao eliminar a necessidade de um modelo decodificador para gerar detecções e rastreamentos de objetos. Nesse caso, é utilizado somente o codificador, servindo como base para a interpretação de dados temporais. Em suma, reduz-se o tempo de processamento em comparação com modelos similares. Além disso, aborda-se um problema comum em

modelos de fim-a-fim — objetivos muito diferentes sendo treinados com um único conjunto de pesos — incorporando técnicas de treinamento multitarefa.

Dessa forma, as contribuições deste trabalho são:

- Os modelos OneTrack e OneTrack-M, que representam um método para construir e treinar modelos de MOT baseados em transformers, agora enfatizando a obtenção de tempos de inferência mais rápidos.
- A rotina de treinamento do modelo OneTrack-M integra conceitos de treinamento multitarefas para otimizar o aprendizado das tarefas de MOT.
- O método de codificação por canal usado pelo OneTrack-M foi desenvolvido para refletir como os dados de entrada foram modelados, sendo capaz de emular o espaço tridimensional de um vídeo a partir de uma pilha de imagens.

1.1 Objetivos

1.1.1 Objetivo Geral

Proposição de um método eficaz e eficiente baseado em transformers para rastrear múltiplos objetos simultâneos no contexto de vídeo, demonstrando sua eficácia através de avaliação comparativa de métricas estabelecidas utilizando para avaliação um *dataset* de *benchmark* MOT.

1.1.2 Objetivos Específicos

- Adaptar e avaliar arquitetura MOT baseada em transformers com foco em diminuir tempo de inferência para esse tipo de modelo.
- Proposição e validação de rotina de treinamento multitarefa para MOT que permita a obtenção de um melhor modelo em termos de eficiência de rastreamento.
- Proposição e validação de *embedding* com contexto espaço temporal para o modelo transformer utilizado no trabalho.

1.2 Organização do Trabalho

Este trabalho está organizado da seguinte forma:

No Capítulo 2, é estabelecida a base teórica, detalhando a metodologia que fundamenta este estudo. Esta parte é essencial para contextualizar as decisões metodológicas adotadas nas etapas subsequentes do trabalho.

No Capítulo 3, exploramos os trabalhos relacionados, enfatizando como influenciaram o projeto da arquitetura do nosso modelo. Esta análise é crucial para entender as escolhas e adaptações feitas no desenvolvimento do modelo.

No Capítulo 4, é dedicado à descrição detalhada da metodologia empregada. Aqui, discutimos os aspectos específicos utilizados, a arquitetura desenvolvida e os formatos de dados necessários para o funcionamento do modelo.

No Capítulo 5, são apresentados os resultados obtidos. Este capítulo inclui detalhes sobre o *dataset* usado para treino e validação, além dos resultados alcançados. Fazemos uma comparação contínua com os resultados de trabalhos anteriores, especialmente aqueles que serviram como referência para decisões de arquitetura do nosso modelo.

Por fim, o Capítulo 6 conclui o trabalho, fornecendo um resumo das descobertas e contribuições. Também discutimos possíveis direções para trabalhos futuros e próximos passos na pesquisa.

2

FUNDAMENTOS TEÓRICOS

Esta seção constrói a base teórica necessária para entender os conceitos específicos da abordagem tratada neste trabalho. Cada elemento abordado aqui será ressaltado posteriormente.

2.1 Detecção de Objetos

A detecção de objetos é uma tarefa fundamental em visão computacional que envolve a identificação e localização de objetos dentro de uma imagem. Diferentemente da classificação de imagens, que categoriza uma imagem inteira, a detecção de objetos visa reconhecer múltiplos objetos e suas posições exatas dentro da imagem. Este processo geralmente resulta na geração de um "*bounding box*"em torno de cada objeto detectado, acompanhado de uma etiqueta de classificação do que o objeto representa.

Formalmente, estas *bounding boxes* podem ser modeladas a partir de suas coordenadas na imagem, seus pontos centrais, suas dimensões (altura e largura) e suas extremidades. Estas possibilidades estão demonstradas nas Figura 1. A forma de representá-las variam de acordo com a abordagem utilizada, mas para se adequar ao formato esperado por redes neurais, estes valores são normalmente relativos à imagem - seus valores variam no intervalo de $[0, 1]$. Estes valores são especificamente importantes por conta de como é calculada a retro-propagação nas redes neurais artificiais.

Antes do advento do aprendizado profundo, as técnicas de detecção de objetos dependiam em grande parte de métodos baseados em características e descritores locais,

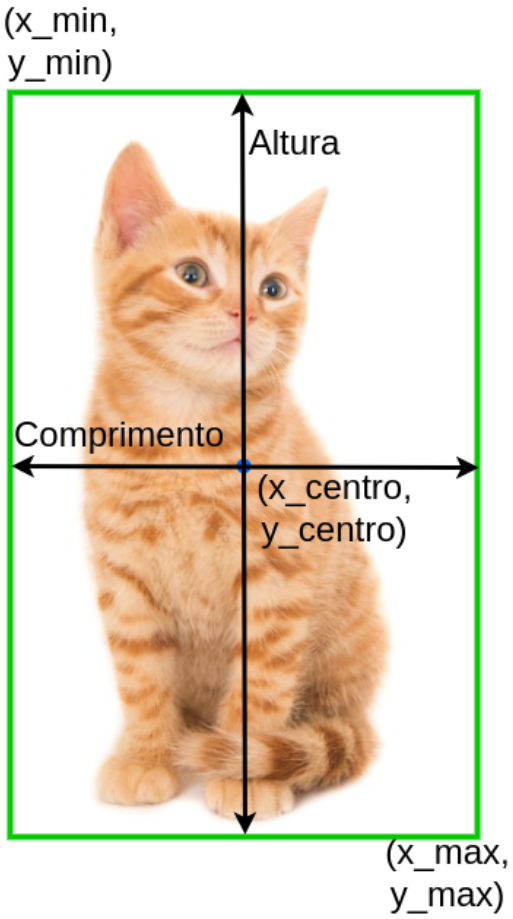

Figura 1 – Imagem de exemplo de coordenadas de uma *bounding box*. Autoria Própria.

como SIFT (*Scale-Invariant Feature Transform*)(Nguyen et al., 2014) e HOG (*Histogram of Oriented Gradients*) (Dalal and Triggs, 2005). Estes métodos envolviam extração manual de características seguida de um classificador como uma SVM (*Support Vector Machine*) para a detecção de objetos.

Com o advento do aprendizado de máquina, especialmente as redes neurais convolucionais (CNNs), houve uma significativa evolução nas técnicas de detecção de objetos. Modelos como R-CNN (*Regions with CNN features*) (Girshick et al., 2014) e suas variantes, como o Fast R-CNN (Girshick, 2015), demonstraram um desempenho notavelmente superior ao integrar CNNs para a extração automática de características e aprimorar a eficiência e precisão da detecção. Esses modelos usam uma abordagem de duas etapas, onde primeiro propõem regiões candidatas e, em seguida, classificam essas regiões usando CNNs.

Atualmente, a detecção de objetos continua a evoluir, com a adoção de aborda-

gens ainda mais sofisticadas e eficientes. Modelos como YOLO (*You Only Look Once*) (Redmon et al., 2016) e SSD (*Single Shot MultiBox Detector*)(Liu et al., 2016) oferecem métodos de detecção em tempo real, unindo proposta de regiões e classificação em uma única etapa, o que aumenta significativamente a velocidade de processamento mantendo alta precisão . Além disso, a introdução de transformers no campo da detecção de objetos promete uma nova onda de inovações, aproveitando a capacidade dos transformers de entender contextos globais em imagens (Carion et al., 2020).

2.2 Rastreamento de Objetos

O rastreamento de objetos consiste em localizar um objeto de interesse ao longo do tempo em sequências de imagens ou vídeos. Matematicamente, isso pode ser representado como a previsão de uma sequência de estados $s_1, s_2, ..., s_i$ representa o estado (por exemplo, posição e escala) do objeto no i-ésimo quadro. Para os efeitos deste trabalho, o ato de "rastrear"um objeto implica a pura associação de objetos detectados a trajetos em quadros anteriores.

Uma das formas de diferenciar abordagens para resolver este problema é se a solução é online ou offline. No rastreamento online, as decisões são tomadas quadro a quadro, ou seja, s_t é previsto apenas com a informação disponível de s_0 até s_t, onde t é o instante atual. No modo offline, toda a sequência de quadros é conhecida antecipadamente, permitindo otimizações globais. Pela definição, modelos offline performam melhor que os online, porém são limitados à não funcionar em tempo real, requisito esse que comumente é relevante para várias aplicações práticas de rastreamento de objetos. Sendo assim as aplicações online são bem mais exploradas no geral.

Outra forma de diferenciar soluções trata do escopo a ser resolvido. No SOT (*Single Object Tracking*), o objetivo é rastrear um único objeto, representado por um estado s_t. Enquanto que no MOT, vários objetos são rastreados simultaneamente, exigindo a gestão de múltiplas sequências de estados e lidando com as interações entre os objetos.

Métodos clássicos, como o filtro de Kalman e o rastreamento baseado em contornos, dependem da modelagem matemática de movimento e aparência. O filtro de

Kalman, por exemplo, é um método preditivo que estima a posição futura do objeto com base em seu estado atual e um modelo de movimento linear. Esta abordagem é bem comum em modelos derivados do SORT (Bewley et al., 2016). Outras formas de processamento de movimento, como OpticalFlow, também já foram empregados, como em Xing et al. (2021).

Com a incorporação de ML no rastreamento de objetos, surgiram métodos que combinam poderosas técnicas de aprendizado de máquina com algoritmos tradicionais de rastreamento, resultando em sistemas mais precisos e robustos.

DeepSORT (Wojke et al., 2017) é um método que combina detecção baseada em *deep learning* com um algoritmo de rastreamento baseado em Kalman e associação de dados usando métricas de similaridade aprendidas. Isso permite rastrear objetos de forma eficiente em tempo real, mesmo em situações de oclusão.

MDNet (Nam and Han, 2016) representa uma abordagem de rastreamento de objetos baseado em redes neurais profundas que aprende a distinguir o objeto de interesse do fundo em vídeos. Este modelo é treinado usando uma coleção de vídeos e depois adaptado para novos objetos no momento do teste.

Redes Siamesas, utilizadas em sistemas como SiamFC (Bertinetto et al., 2021), representam outra abordagem significativa. Estas redes comparam a semelhança entre a região de interesse inicial e candidatos em novos quadros, permitindo que o sistema rastreie o objeto ao longo do tempo. A abordagem se destaca pela eficiência e precisão, sendo adequada para rastreamento em tempo real.

Recentemente, os transformers, conhecidos por seu sucesso em processamento de linguagem natural, começaram a ser adaptados para visão computacional e rastreamento de objetos. Seu mecanismo de atenção permite modelar dependências de longo alcance, oferecendo uma abordagem inovadora para entender a relação entre objetos e o contexto em que se encontram. A integração de transformers em sistemas de MOT está abrindo novas possibilidades para rastreamento preciso e eficiente em ambientes complexos (Zeng et al., 2022).

2.3 *Deep Learning* e Redes Neurais Artificiais

Deep Learning, um ramo avançado do *Machine Learning*, utiliza redes neurais artificiais com várias camadas ocultas para aprender representações de dados em múltiplos níveis de abstração. Essas camadas adicionais permitem que o modelo capture relações complexas e padrões intricados nos dados, uma capacidade que os métodos de aprendizado raso não conseguem igualar. A profundidade dessas redes, que pode variar de algumas a centenas de camadas, confere a elas uma flexibilidade e poder de modelagem excepcionais, tornando-as adequadas para uma ampla gama de aplicações, desde o reconhecimento de fala até a análise de imagens complexas (LeCun et al., 2015).

A principal distinção entre aprendizado profundo e raso reside no número de camadas ocultas e na complexidade das representações que podem ser aprendidas. Enquanto modelos rasos, como perceptrons de uma única camada, são limitados na sua capacidade de processar dados não lineares e complexos, redes de *deep learning*, com múltiplas camadas, podem extrair características de alto e baixo nível, aprendendo representações mais sofisticadas. Esta habilidade é crucial para tarefas complexas de classificação e regressão, onde nuances e padrões sutis nos dados precisam ser capturados (Goodfellow et al., 2016).

Esse tipo de modelo apresenta a vantagem de aprender características diretamente de grandes conjuntos de dados brutos, sem a necessidade de intervenção manual. Isto é particularmente benéfico em campos como visão computacional e processamento de linguagem natural, onde a extração manual de características é desafiadora e ineficiente. Além disso, a capacidade de processar e aprender de grandes volumes de dados faz com que sejam ideais para aplicações que envolvem Big Data.

Em uma rede neural totalmente conectada (ou *dense layer*), cada neurônio de uma camada está conectado a todos os neurônios na camada seguinte, permitindo que a rede aprenda combinações complexas de características. Matematicamente, a saída de uma camada totalmente conectada pode ser descrita como $y = f(Wx + b)$, onde x é a entrada, W é a matriz de pesos, b é o vetor de viés, e f é a função de ativação. Este processo está demonstrado, bem como um exemplo de rede totalmente conectada, na Figura 2

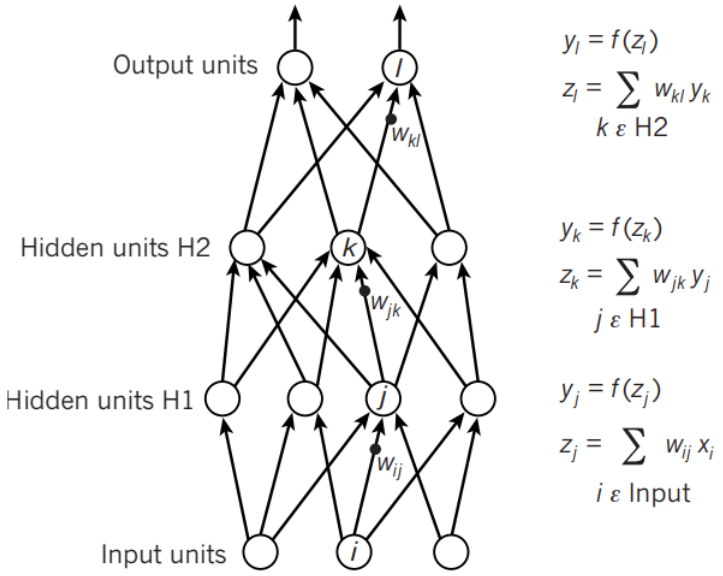

Figura 2 – Imagem representando o processo de cálculo do *backpropagation* em uma rede neural totalmente conectada. Fonte LeCun et al. (2015).

O treinamento eficaz de modelos de *deep learning* requer:

- **Dados**: Grandes conjuntos de dados são fundamentais para capturar a variedade de características necessárias para um aprendizado eficaz.
- **Função de Perda**: Uma função de perda, como a entropia cruzada para classificação ou o erro quadrático médio para regressão, quantifica o erro do modelo.
- **Otimizador**: Algoritmos como SGD ou Adam são utilizados para atualizar os pesos da rede na direção que minimiza a função de perda.
- **Backpropagation**: O *backpropagation* é o método usado para calcular os gradientes da função de perda. Para um peso w, o gradiente é $\frac{\partial L}{\partial w}$, sendo L a função de perda. Este gradiente é utilizado para atualizar os pesos na direção que minimiza a perda.

O *backpropagation* é um mecanismo fundamental no treinamento de redes neurais, permitindo a atualização eficiente dos pesos. Matematicamente, é baseado na regra da cadeia da diferenciação. Por exemplo, se um neurônio tem saída $o = f(z)$ e $z = wx + b$, o gradiente do erro E em relação a w é $\frac{\partial E}{\partial w} = \frac{\partial E}{\partial o} \cdot \frac{\partial o}{\partial z} \cdot \frac{\partial z}{\partial w}$.

Apesar de sua eficácia, o *deep learning* enfrenta limitações, como a necessidade de vastas quantidades de dados e o desafio do sobre-ajuste em conjuntos menores. A

interpretabilidade dos modelos também é uma área de pesquisa ativa, dada a natureza "caixa-preta"das redes profundas. Além disso, o custo computacional para treinar e inferir modelos profundos pode ser substancial, exigindo hardware especializado como GPUs, ou otimizações específicas para cada modelo.

2.3.1 Extração de Características

A extração de características é um procedimento crucial na transformação de dados brutos em representações mais concisas e informativas, enfatizando atributos essenciais para tarefas específicas. Tal processo é exemplificado pelo uso de redes neurais para gerar vetores de características. Esses vetores, aprendidos pela rede, buscam encapsular de forma eficiente as informações dos dados de treinamento, facilitando a inferência por etapas subsequentes, frequentemente compostas por uma rede de Perceptron Multicamada (*Multilayer Perceptron* - MLP).

Além de sua aplicação direta, a extração de características revela um potencial significativo em outro aspecto: a capacidade de desenvolver modelos genéricos para representar, por exemplo, imagens. Uma vez treinado para identificar objetos em imagens com precisão, tal modelo pode ser adaptado para variadas finalidades. Em cenários onde os dados de treinamento são limitados, a utilização de extratores de características de uma rede pré-treinada, como aquelas treinadas no conjunto de dados ImageNet (Murad and Pyun, 2017), apresenta-se como uma estratégia eficaz. Nesse contexto, apenas a etapa final do modelo, responsável por determinar a saída e interpretar a representação da imagem, necessita de treinamento adicional. Este método é conhecido como Aprendizado por Transferência (*Transfer Learning*).

Adicionalmente, mesmo em situações com disponibilidade razoável de dados, a adoção desses pesos pré-treinados, em substituição à inicialização aleatória dos pesos, pode ser vantajosa. Posteriormente, um treinamento completo do modelo nos dados específicos da tarefa é realizado, um processo denominado Ajuste Fino (*Fine Tuning*). Essas abordagens demonstram a versatilidade e eficácia da extração de características, tanto para aprimoramento de modelos específicos quanto para a generalização em

diferentes aplicações.

2.4 Aprendizado Multitarefas

Aprendizado Multitarefas (Multi-Task Learning - MTL) é uma abordagem de aprendizado de máquina que treina um modelo em várias tarefas relacionadas simultaneamente, usando uma representação compartilhada. Este método baseia-se na premissa de que aprender tarefas em conjunto pode melhorar o desempenho do modelo em cada tarefa individual, em comparação ao treinamento de cada tarefa isoladamente. A ideia central é que as tarefas compartilham informações úteis que podem ser utilizadas para melhorar a generalização (Caruana, 1997b).

No aprendizado multitarefas, um modelo comum é treinado em várias tarefas. Essas tarefas são geralmente relacionadas, mas distintas. A estrutura do modelo pode variar, mas frequentemente inclui partes específicas para cada tarefa e uma parte compartilhada que aprende representações comuns. Por exemplo, em uma rede neural, as primeiras camadas podem ser compartilhadas entre todas as tarefas, enquanto as camadas posteriores são especializadas para tarefas específicas.

Existem várias maneiras de implementar MTL, cada uma com seus próprios prós e contras:

- Arquitetura Compartilhada: A mais comum, onde as camadas iniciais são compartilhadas e as finais são específicas para cada tarefa. Esta abordagem é eficiente em termos de parâmetros, mas pode sofrer se as tarefas forem muito diferentes.

- Arquitetura com Adaptação de Tarefas: Utiliza módulos de adaptação para ajustar as representações aprendidas para cada tarefa. Isso permite uma maior flexibilidade, mas aumenta a complexidade do modelo.

- Aprendizado com Base em Relação entre Tarefas: Identifica e explora explicitamente as relações entre tarefas, o que pode melhorar a aprendizagem se as relações forem bem compreendidas.

Algumas vantagens e desvantagens desta metodologia de treinamento de modelos estão dispostas em seguida.

Vantagens:

- Melhora da Generalização: Ao compartilhar representações, o modelo pode generalizar melhor para novos dados.
- Eficiência de Dados: MTL pode ser particularmente útil em situações onde os dados para algumas tarefas são limitados.
- Eficiência Computacional: Compartilhar parâmetros reduz a necessidade de recursos computacionais.

Desvantagens:

- Conflito de Tarefas: Se as tarefas são muito diferentes ou contraditórias, o desempenho pode ser prejudicado.
- Complexidade de Gerenciamento: O treinamento de modelos MTL pode ser mais complexo, exigindo balanceamento entre tarefas.

No treinamento multitarefas, múltiplas funções de perda, cada uma representando uma tarefa diferente, são combinadas de alguma forma para formar uma função de perda composta. O modelo é então treinado para minimizar esta função de perda composta. A ideia central é que, ao fazer isso, o modelo aprende representações que são úteis para múltiplas tarefas.

Formalmente, podemos representar a função de perda composta como:

$$L_{total} = \sum_{i=1}^{N} \alpha_i L_i \tag{2.1}$$

onde L_{total} é a função de perda total, N é o número de tarefas, L_i é a função de perda para a tarefa i, e α_i são pesos que definem a importância relativa de cada tarefa.

2.5 Redes Neurais Convolucionais

Redes Neurais Convolucionais (CNNs) revolucionaram o campo da visão computacional. Elas são projetadas para processar dados com uma estrutura de grade, como imagens, e são eficazes na captura de características espaciais e temporais. A inspiração para CNNs vem da organização do córtex visual humano, que processa informações visuais em termos de hierarquias de características.

A convolução é a operação principal em uma CNN e é matematicamente definida como:

$$(I * F)(i, j) = \sum_{m} \sum_{n} I(i + m, j + n) \times F(m, n) \tag{2.2}$$

onde I é a imagem de entrada, F é o filtro, e (i, j) são as coordenadas espaciais. Esta operação cria um mapa de características que captura informações importantes, como bordas e texturas.

Os principais hiper-parâmetros para este tipo de camada são os seguintes:

- Tamanho do Filtro: Define a área de entrada que o filtro considera para produzir uma única unidade de saída. Tamanhos comuns incluem 3x3 e 5x5.
- *Stride*: Refere-se ao número de pixels pelo qual o filtro se move durante a convolução. Strides maiores resultam em mapas de características menores.
- *Padding*: Adiciona pixels ao redor da borda da imagem de entrada, permitindo que a camada convolucional mantenha a resolução espacial após a aplicação do filtro.

E para criar uma rede neural convolucional, de forma genérica, é necessário pelo menos as seguintes camadas:

- Camadas Convolucionais: Responsáveis pela extração de características. Utilizam filtros para transformar os dados de entrada em uma forma mais útil.
- Camadas de Pooling: Reduzem a dimensionalidade dos mapas de características, aumentando a eficiência computacional e ajudando na redução de sobre-ajuste. Exemplo comum é o max pooling.

- Camadas Totalmente Conectadas: Seguem as camadas convolucionais e de pooling, e são usadas para a classificação ou regressão com base nas características extraídas.

Uma CNN típica tem uma arquitetura em camadas, começando com várias camadas convolucionais e de pooling que extraem características, seguidas por camadas totalmente conectadas para a tomada de decisões. A primeira camada convolucional aprende características de baixo nível (como bordas), e as camadas subsequentes combinam essas características de baixo nível para aprender características de alto nível (como formas ou objetos inteiros).

As CNNs são capazes de aprender um conjunto hierárquico de características. Inicialmente, elas detectam características de baixo nível. À medida que avançamos na rede, as características de alto nível são construídas usando as de baixo nível, permitindo que a rede identifique objetos complexos.

2.6 Redes Transformers

Os transformers, introduzidos por Vaswani et al. (2023) no artigo "*Attention is All You Need*"em 2017, representam um marco na modelagem de sequências, particularmente em NLP. Eles surgiram como uma alternativa aos modelos RNN e LSTM, que processam dados sequencialmente. Diferentemente desses modelos anteriores, os transformers permitem processamento paralelo e capturam dependências de longo alcance, superando desafios relacionados à memória e eficiência.

O transformer é composto por uma série de blocos que trabalham em conjunto para processar sequências de dados:

- Multi-Head Attention: Esta é a espinha dorsal do transformer. Ela permite que o modelo foque em diferentes partes da sequência simultaneamente, uma inovação crucial para entender contextos complexos.

- Normalização em Camadas e Conexões Residuais: Garantem a estabilidade durante o treinamento e ajudam a evitar o problema do desaparecimento do gradi-

ente.

- Camadas Totalmente Conectadas: Cada bloco de atenção é seguido por uma camada totalmente conectada que transforma a saída da atenção antes de passá-la para o próximo bloco.

A atenção em um transformer é quantificada usando a operação de "*Scaled Dot-Product Attention*", que é computacionalmente expressa por:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.3}$$

onde Q, K, e V representam matrizes de consulta, chave e valor, respectivamente. O termo $\sqrt{d_k}$ é um fator de escala que ajuda na estabilidade do gradiente.

A arquitetura do transformer é composta por duas partes principais: o codificador e o decodificador. Cada um consiste em uma série de blocos idênticos que contêm *Multi-Head Attention* e redes *feed-forward*. O codificador processa a sequência de entrada, enquanto o decodificador gera a saída, passo a passo, usando a saída do codificador e as saídas geradas até o momento. Modelos transformers foram inicialmente concebidos a partir dessa arquitetura, porém autores alteram essa abordagem de acordo com o problema: para geração de texto, o GPT utiliza apenas o decodificador (Brown et al., 2020), enquanto que o Vision Transformer utiliza apenas o codificador (Dosovitskiy et al., 2021).

Apesar de originalmente projetados para NLP, os transformers estão sendo adaptados para visão computacional. Em modelos como o Vision Transformer (ViT) e DETR (Carion et al., 2020), os transformers tratam imagens como sequências de recortes e aplicam técnicas similares às usadas em textos, permitindo uma abordagem inovadora para tarefas como classificação, detecção de objetos, entre outros.

Transformers requerem grandes quantidades de dados e recursos computacionais para treinamento. Além disso, sua natureza altamente flexível pode levar a desafios em termos de interpretação e generalização. Questões como a atenção dispersa e a dificuldade em capturar relações locais em dados também são áreas de pesquisa ativa.

3

TRABALHOS RELACIONADOS

Este capítulo abrange os principais trabalhos que influenciaram as decisões tomadas no design do OneTrack-M.

3.1 Abordagem Clássica de ML para MOT

Chama-se de Machine Learning Clássico o conjunto de técnicas de aprendizado de máquina que não são consideradas profundas, como Máquinas de Vetores de Suporte (SVMs), K-vizinhos mais próximos, K-means, entre outras. Embora essas técnicas careçam das capacidades de generalização de métodos profundos, elas ainda têm valor para resolver problemas mais simples, cenários de escassez de dados e em situações com pouco poder computacional disponível.

No caso de MOT, que é um problema complexo, as técnicas clássicas que mostram bons resultados são principalmente aquelas baseadas no filtro de Kalman, como (Bewley et al., 2016). Nesse caso, o filtro de Kalman fornece contexto temporal para o sistema, funcionando sequencialmente com qualquer modelo de detecção de objetos. A ideia é prever a posição futura a partir de detecções anteriores em uma janela de quadros. A partir desses valores, as detecções atuais são associadas às anteriores. De forma mais geral, filtros de partículas também são uma opção para inferir a trajetória dos objetos ao longo do tempo, como feito em Jaward et al. (2006).

Outra abordagem é o fluxo óptico para modelar o movimento de cada objeto ao longo do tempo, como em Xing et al. (2021). Técnicas de fluxo óptico calculam o movi-

mento dos objetos analisando as diferenças de intensidade entre quadros consecutivos. Esse método ajuda no rastreamento fornecendo vetores de movimento, que podem ser usados para prever as próximas posições dos objetos, auxiliando na associação de detecções entre quadros. O fluxo óptico é particularmente útil em cenários onde os objetos se movem de forma contínua e suave, tornando-se uma ferramenta valiosa no conjunto de ferramentas do MOT.

3.2 Deep Learning em MOT

Avanços recentes em técnicas de deep learning revolucionaram o campo do Rastreamento de Múltiplos Objetos (MOT), superando as limitações das abordagens tradicionais e clássicas, especialmente em cenários desafiadores. Modelos como o DeepSORT (Wojke et al., 2017), que combina o filtro de Kalman com características extraídas por redes profundas, exemplificam o estado da arte em MOT, mostrando as capacidades superiores de generalização desses modelos.

A introdução do deep learning no MOT derrubou barreiras anteriores, permitindo várias aplicações e inovações. Nesse contexto, Redes Neurais Convolucionais (CNNs) se destacam por sua capacidade de extrair informações locais de imagens, amplamente adotadas em modelos de deep learning, como mostrado em trabalhos como Du et al. (2023), Zhang et al. (2022), e Meinhardt et al. (2022). Nesses trabalhos, a parte convolucional da arquitetura é responsável por extrair características da imagem que são processadas em uma etapa subsequente.

Além das características espaciais, o processamento de sequências temporais é essencial para rastrear objetos em vídeos. Redes Neurais Recorrentes (RNNs) são usadas para manter informações relevantes ao longo do tempo, enquanto transformers, com suas camadas de atenção, estão emergindo como uma abordagem robusta para relacionar sequências de imagens, como ilustrado em Zeng et al. (2022) e Meinhardt et al. (2022).

Além das características temporais, métodos baseados em reidentificação usam aspectos visuais dos objetos para associar identidades, como visto em Aharon et al.

(2022) e Mostafa et al. (2022). Embora eficientes, esses métodos podem perder informações temporais cruciais, pois dependem apenas da similaridade dos objetos em diferentes quadros, em vez de modelar seu movimento ou até ambos simultaneamente.

O CenterTrack (Zhou et al., 2020) introduziu uma nova maneira de resolver o MOT. Ele inova ao repensar como os dados são modelados, tornando o processo de associação entre quadros uma questão de rastrear cada deslocamento de centróide entre os quadros. Essa abordagem torna a inferência mais direta e muda a forma como os objetos são detectados, usando mapas de calor. Assim, o modelo pode aprender todas essas partes de maneira unificada.

Finalmente, é essencial destacar que o progresso do MOT está intimamente ligado aos avanços na detecção de objetos trazidos pelo *deep learning*. Essa sinergia entre detecção e rastreamento melhora os modelos existentes e abre caminho para o desenvolvimento de novas técnicas e abordagens em MOT.

3.3 Modelos de Imagem com Transformers

Os transformers melhoraram o processamento de linguagem natural e agora desempenham um papel crucial na visão computacional, fornecendo abordagens inovadoras para tarefas complexas. O Vision Transformer (ViT) Dosovitskiy et al. (2021) adota uma estratégia nova, dividindo imagens em *patches* de tamanho fixo, tratando-os como sequências de *tokens* e aplicando camadas de atenção para capturar dependências globais. Essa abordagem contrasta com as convoluções locais das CNNs, permitindo que o ViT, quando treinado com grandes volumes de dados e parâmetros, supere modelos de CNN em classificação de imagens.

No entanto, novas variantes de transformers, como o Swin Transformer (Liu et al., 2021a), avançam ainda mais na visão computacional. O Swin Transformer destaca-se por sua capacidade de modelar hierarquias de imagem de forma eficiente, adaptando-se a diferentes escalas e contextos, tornando-o particularmente adequado para tarefas de detecção e segmentação de objetos. Este modelo até serve como base para outro modelo de ponta em classificação de vídeo: o Video Swin Transformer (Liu et al., 2021b).

Em paralelo, o DETR (Detection Transformer) (Carion et al., 2020) representa outra inovação significativa, simplificando o processo de detecção de objetos ao eliminar etapas de pós-processamento complexas, como a Supressão de Máximos Não Relevantes (NMS). O DETR modela a detecção de objetos como um problema de predição de conjuntos, usando uma combinação de codificador e decodificador transformer para gerar detecções diretas sem âncoras predefinidas.

Essas inovações em modelos baseados em transformers demonstram um avanço técnico significativo e expandem as possibilidades para o desenvolvimento de sistemas de MOT mais eficientes e precisos. O uso de transformers em MOT promete uma nova onda de abordagens que podem lidar melhor com desafios como oclusões e variações no tamanho dos objetos, aprimorando a robustez e a eficácia dos sistemas de rastreamento.

Assim, os transformers trazem novas técnicas e metodologias para o MOT, estabelecendo um novo paradigma no processamento e interpretação de dados visuais, pavimentando o caminho para inovações contínuas no rastreamento de múltiplos objetos.

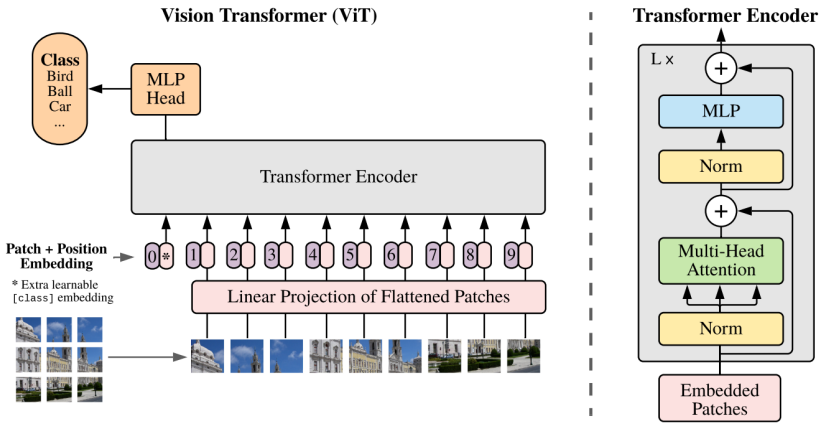

Figura 3 – Arquitetura do modelo Vision Transformer - Imagem retirada de Dosovitskiy et al. (2021)

3.4 Transformers no Rastreamento de Múltiplos Objetos (MOT)

A incorporação de modelos baseados em transformers no Rastreamento de Múltiplos Objetos (MOT) representa uma mudança de paradigma, oferecendo abordagens inovadoras e sofisticadas para desafios tradicionais neste campo. Um exemplo pioneiro é o modelo TrackFormer descrito em Meinhardt et al. (2022). Este modelo emprega uma arquitetura transformer de codificador-decodificador, semelhante às usadas em tarefas de Processamento de Linguagem Natural (NLP), para rastrear objetos em sequências de vídeo. O aspecto distintivo do TrackFormer reside em sua metodologia de rastreamento: ele gera e atualiza continuamente consultas de rastreamento para manter a identificação dos objetos entre os quadros. Essa abordagem demonstra a capacidade dos transformers de gerenciar informações espaciais e temporais complexas, mantendo a consistência do rastreamento sem a necessidade de um modelo dedicado de detecção de objetos.

O modelo MOTR introduzido por Zeng et al. (2022) representa outro avanço significativo. O MOTR expande o conceito de DETR para o contexto de MOT, adotando uma estrutura de codificador-decodificador para detectar e rastrear objetos simultaneamente. Ele usa consultas de rastreamento que se atualizam ao longo do tempo, semelhante ao TrackFormer, mas com uma implementação distinta. Essa técnica permite que o MOTR lide com a detecção de objetos como um problema de predição de conjuntos, melhorando a precisão do rastreamento e a eficiência computacional. Essa inovação marca um marco importante na simplificação do processo de MOT e no aprimoramento de sua eficácia.

Além disso, o desenvolvimento do TransTrack, conforme relatado por Sun et al. (2021), também contribuiu significativamente para o MOT. O TransTrack adota uma abordagem híbrida, integrando convoluções para extração de características e um transformer para associação de detecções. Este modelo híbrido ilustra o potencial de combinar CNNs e transformers no MOT, abordando de forma eficaz os desafios de rastreamento em contextos complexos e sugerindo uma direção promissora para futuras pesquisas.

Os modelos mencionados indicam um progresso substancial nos métodos de

MOT, demonstrando como os transformers podem reformular as estratégias de rastreamento, levando a sistemas mais precisos, eficientes e robustos. A integração bem-sucedida de transformers no MOT sinaliza um avanço considerável na visão computacional, apontando para novas possibilidades de pesquisa e desenvolvimento. No entanto, dois problemas comuns existem em todos esses modelos: o uso de uma arquitetura completa de transformer (ou seja, codificador e decodificador) e tempos de inferência muito altos, tornando o uso em tempo real impraticável.

3.5 Limitações das Soluções Atuais

A principal limitação encontrada na maioria dos trabalhos recentes, que têm sido foco de desenvolvimento no campo, é o tempo de inferência. Trabalhos como Mostafa et al. (2022) e Zhang et al. (2021) simplificaram o sistema em um único modelo que realiza tanto a detecção quanto o rastreamento de ponta a ponta. Ambos empregam uma CNN para extrair características das imagens e então realizam o MOT. No entanto, eles não se destacam na avaliação de métricas quando comparados a outros métodos mais lentos.

Como mencionado anteriormente, essas abordagens mais rápidas usam CNNs como base de sua solução. Esses métodos carecem em ambos os campos—métricas e velocidade—problemas que são abordados pela solução proposta no OneTrack-M. Essa melhoria é amplamente possibilitada por uma observação feita pelos autores de Zhang et al. (2021), que descrevem uma injustiça em relação ao passo de treinamento de tais modelos de ponta a ponta.

3.6 Treinamento Multitarefa (MTL)

Em relação à eficácia do treinamento multitarefa, em Caruana (1997a), os autores mostram que redes neurais treinadas em múltiplas tarefas relacionadas podem alcançar melhor desempenho em cada tarefa em comparação com o treinamento isolado, provando que o método pode levar a uma melhor transferência de conhecimento e seleção ótima de pesos para resolver um problema multifacetado.

Alguns estudos sugerem que o MTL pode ser eficaz em cenários com escassez de dados, uma vez que o aprendizado conjunto permite ao modelo usar informações de todas as tarefas para melhorar sua capacidade de generalização. Em Zhang and Yang (2017), os autores exploram abordagens e discutem como o método pode superar a escassez de dados.

Essa abordagem se assemelha ao desenvolvimento e treinamento dos modelos de MOT pelos autores, onde um modelo de ponta a ponta é comumente treinado com todas as tarefas simultaneamente. No entanto, Zhang et al. (2021) argumenta sobre a dissonância entre o aprendizado dessas tarefas, o que prejudica o aprendizado geral do modelo, de modo que treinar uma tarefa prejudica o aprendizado de outra. A solução dos autores foi usar a abordagem de otimização conjunta para resolver esse problema, com funções de perda específicas para cada tarefa. No nosso caso, propomos uma abordagem nova, usando treinamento em múltiplas etapas, onde o modelo é treinado mais de uma vez, cada vez com uma tarefa alvo (primeiro detecção e depois rastreamento), mas tudo no mesmo modelo. Mais detalhes sobre essa abordagem são fornecidos na Seção 4.8.

3.7 Considerações Finais

Na Tabela 1 há uma distribuição destes modelos discutidos, onde há pontos relevantes sobre cada um deles, ressaltando suas inovações e suas dificuldades. Neste momento, apenas será tratado a qualidade conceitual de cada abordagem, sua qualidade prática, isto é, valores de métricas em *benchmark*, será tratada de forma comparativa com o modelo proposto neste trabalho no Capítulo 5 com os resultados obtidos.

Tabela 1 – Resumo dos Trabalhos Relacionados em MOT

Trabalhos	**Métodos**	**Arquitetura**	**Atenção**	**Limitações**
DeepSORT (Wojke et al., 2017)	Associação de aparência, filtro de Kalman	Detecção + Associação	Não	Limitado em cenários complexos
StrongSORT (Du et al., 2023)	DeepSORT aprimorado com modelos melhores	Detecção + Incorporação + Associação	Não	Limitado em cenários complexos
BotSORT (Aharon et al., 2022)	Rastreamento por detecção, fusão IoU-ReID	Detector + Associador	Não	Problemas com movimento da câmera em cenas densas
FairMOT (Zhang et al., 2021)	Detecção de objetos, características de re-ID	Treinamento fim-a-fim, requer pós-processamento	Não	Luta com oclusão e variações de escala
ByteTrack (Zhang et al., 2022)	Associação de dados via BYTE, YOLOX	Detector, Associador e ReID	Não	Problemas com movimento da câmera, oclusão, desfoque de movimento
CenterTrack (Zhou et al., 2020)	Objetos como pontos, dois quadros e mapa de calor	Fim-a-Fim	Não	Problemas de eficiência e eficácia
MOTR (Zeng et al., 2022)	Problema de previsão de conjuntos, consultas de rastreamento e detecção	Fim-a-Fim	Sim	Alto custo computacional
MOTRv2 (Zhang et al., 2023)	Semelhante ao MOTR, detector YOLOX separado	Detecção separada, rastreamento fim-a-fim	Sim	Alto custo computacional
TrackFormer (Meinhardt et al., 2022)	Consultas de objeto e rastreamento	Fim-a-Fim	Sim	Necessita de grandes conjuntos de dados, alto custo de memória e computacional
Transtrack (Sun et al., 2021)	Extração de características por CNN, decodificadores duplos	Fim-a-Fim	Sim	Problemas de robustez com consultas de objeto, falta de informações de longo prazo
LMOT (Mostafa et al., 2022)	Codificador DLA-34, transformador linear	Treinamento separado para rastreador e detector	Sim	Luta com variações de condições ambientais

4

Os modelos OneTrack e OneTrack-M

Este capítulo constitui o cerne do estudo, detalhando tanto as decisões adotadas quanto seus fundamentos teóricos. Em casos que há algumas opções possíveis para valores, essa decisão será deixada em aberto propositalmente, e tais valores são abordados na seção de resultados com os testes de ablação. É estruturado em três seções principais: a descrição da arquitetura do modelo OneTrack, a descrição do modelo OneTrack-M e os critérios adotados para o treinamento da última versão (OneTrack-M).

O modelo OneTrack foi construído usando como base os trabalhos apresentados por Zeng et al. (2022) e Carion et al. (2020). Em contrapartida, as mudanças trazidas no OneTrack-M, baseiam-se significativamente nos estudos de Zhou et al. (2020) e Zeng et al. (2022). O primeiro estabelece um modelo de referência que usa centros de objetos para rastreamento, enquanto o segundo serve como um *benchmark* para abordagens de rastreamento que integram transformers. Sendo assim, a principal mudança entre as versões está em como o modelo trata as posições de cada objeto e como o modelo realiza o rastreamento.

De modo geral, a arquitetura dos modelos OneTrack incorporam o codificador pré-treinado do modelo em Dosovitskiy et al. (2021), que é usado para extrair características de sequências de imagens e gerar mapas de atenção. Essa escolha metodológica aproveita a eficácia do Vision Transformer em aprender nuances visuais presentes em imagens, criando uma correlação entre cada um dos recortes dela.

4.1 Arquitetura OneTrack

A arquitetura do modelo traz uma nova abordagem para aplicar modelos Transformers à tarefa de MOT. A Figura 4 mostra uma visão geral da abordagem proposta neste trabalho. O fluxo de dados pode ser descrito da seguinte forma:

1. Dado um inteiro N, serão utilizados N quadros em cada inferência do modelo, onde um se refere ao quadro atual e $N - 1$ são os quadros anteriores que serão utilizados como contexto.

2. Uma etapa de pré-processamento necessária é recortar as detecções dos $N - 1$ quadros anteriores e construir um rastro para cada objeto ao longo dos quadros passados. Cada objeto recebe um id único (esse id será utilizado apenas para treinamento).

3. É feito um reshape nos recortes para que todos estejam com tamanho fixo de 64x64 pixels.

4. A entrada do modelo se torna esses recortes, suas *bounding boxes* e seus IDs correspondentes, bem como o quadro atual.

5. Cria-se os *embeddings* posicionais necessários para contextualizar o modelo na posição de cada recorte.

6. Cria-se *embeddings* de rastreamento utilizando os ids de cada objeto.

7. A sequência de entrada do encoder é formada por duas partes:

    a) Uma composta por projeções lineares dos recortes feitos a partir dos quadros anteriores, onde cada objeto individual é um elemento da sequência. É calculada a média dos $N - 1$ recortes encontrados para cada objeto. Essas imagens são achatadas, formando um vetor de 12288 posições (64 * 64 * 3), o qual passa por uma camada MLP que produz um vetor de 768 posições. O tamanho da sequência é completado com vetores com valor 0 até completar 100 posições. (passado)

b) A outra é composta pela projeção linear de um mapa de características extraído do quadro atual. O mapa de características tem formato [768, 10, 10] e é feito um reshape dele para que fique no formato [100, 768], ou seja, é uma sequência de 100 vetores de 768 posições, igual à primeira parte da entrada. (presente)

8. Os *embeddings* posicionais são somados às duas partes da entrada (passado e presente).

9. O *embedding* de rastreamento é somado apenas à parte do passado da entrada.

10. A parte do passado e do presente são concatenadas e alimentadas ao encoder.

11. A saída do encoder é do formato [200, 768].

12. Apenas as 100 primeiras posições são utilizadas.

13. Cada posição da sequência passa por duas MLPs:

    a) Uma referente à cabeça de detecção, dando as coordenadas das *bounding boxes* para cada objeto encontrado (ou nenhum).

    b) Outra referente à cabeça de rastreamento, prevendo a qual id este objeto pertence (ou nenhum, no caso de ser um objeto novo na sequência).

4.1.1 Criador de Trajetos para Janela

Nesta abordagem, foi criado um histórico de recortes para cada objeto individual. Ou seja, para cada objeto previamente identificado nos $N - 1$ quadros anteriores, será construída uma lista de imagens desse mesmo objeto ao longo desses quadros. Esses recortes compõem metade das entradas do encoder, sendo responsáveis por fornecer ao modelo o contexto de quais objetos já foram vistos anteriormente. Para isso, são necessárias tanto as *bounding boxes* quanto os IDs de cada objeto em cada quadro.

Esse ID é um inteiro que corresponde ao trajeto de cada objeto detectado ao longo dos quadros. O ID não tem um limite de valor, mas, neste caso, o modelo é

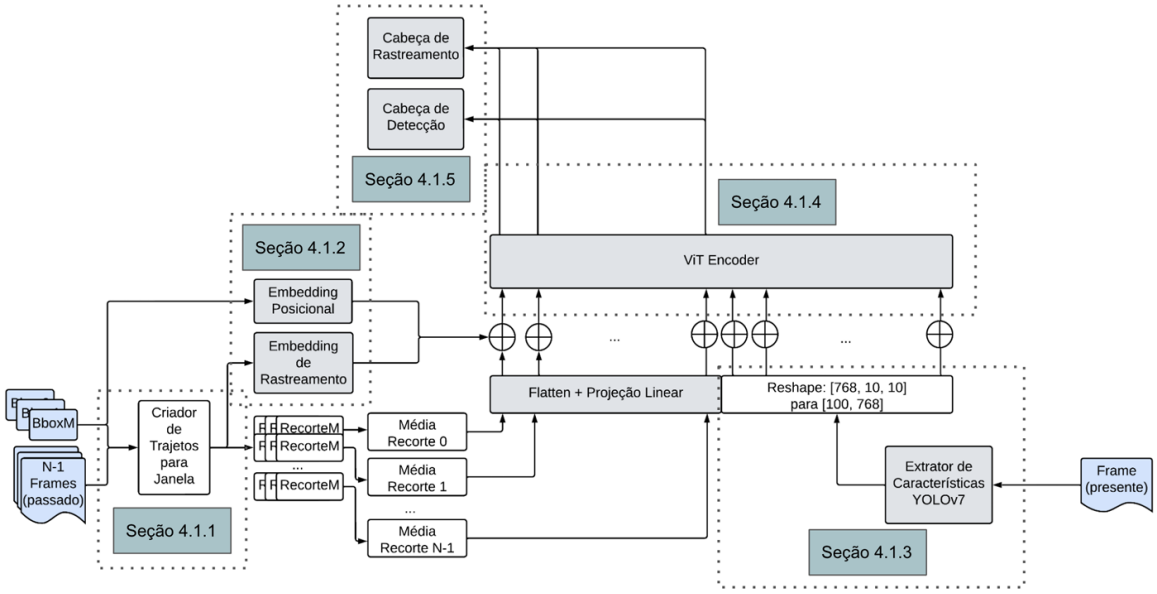

Figura 4 – Arquitetura desenvolvida para o modelo OneTrack.

limitado a um contexto de 100 objetos, isto é, 100 objetos diferentes encontrados em N quadros. Portanto, é realizada uma normalização, onde cada objeto recebe um novo ID, limitado ao intervalo de 0 a 99. Como resultado desta etapa, considerando que haviam M objetos diferentes detectados nos $N-1$ quadros, haverá M vetores, cada um contendo $N-1$ recortes de objetos (caso um objeto não exista em algum dos quadros, será feito um preenchimento com vetores de valor 0).

Para cada um dos M vetores, será calculada a média (desconsiderando os preenchimentos com 0). Esta redundância é necessária apenas por questões técnicas de manter a forma dos tensores ao longo do processamento.

Esta abordagem determina também a forma como os dados foram modelados no escopo geral deste modelo. Isto provou-se uma limitação que foi superada na próxima versão do modelo.

4.1.2 Embeddings do modelo

No trabalho de Dosovitskiy et al. (2021), e para modelos transformers em geral, é necessário fornecer um conjunto de embeddings que permite ao modelo manter co-

nhecimento espacial dos tokens de entrada que ele recebe. Para o caso de imagens, comumente estes embeddings se referem à posição relativa de cada recorte na imagem original. No caso deste trabalho, além das duas dimensões da imagem - altura e largura - também há a dimensão temporal. Sendo assim para manter estas características foram adicionados aos embeddings posicionais um conjunto de embeddings que são gerados a partir dos valores de ID do objeto e coordenadas relativas à imagem original, onde cada token desta sequência se trata de uma pilha de recortes de cada objeto já detectado durante a janela de tempo em questão.

4.1.3 Tratamento do quadro Atual

Para o quadro atual, a abordagem adotada é significativamente mais simples. Conforme descrito no artigo do DETR (Carion et al., 2020), um modelo Transformer pode ser capaz de aprender a detectar objetos utilizando características extraídas da imagem por outro modelo pré-treinado. Nesse contexto, o modelo escolhido para extrair essas características foi o YOLOv7 (Wang et al., 2022), especificamente seu *backbone* pré-treinado no *dataset* COCO Lin et al. (2015).

Entende-se que essas características são suficientemente robustas para não somente fornecer informações úteis ao modelo para a funcionalidade de detecção dos objetos, mas também para representar como cada um desses objetos se apresenta. O mapa de características obtido é submetido a um reshape para que fique em um formato análogo a uma sequência, onde os canais gerados se tornam vetores de características com 768 valores ao longo de uma sequência de 100 posições.

A escolha deste extrator de características foi motivada pelos resultados obtidos pelo modelo YOLOv7 na detecção de objetos. Como exposto no modelo MOTR, um dos pontos fracos dos modelos unificados baseados em transformers é justamente a habilidade de detecção dos objetos, sendo assim utilizar um detector de objetos para extração de características pode mostrar um conjunto de características que facilite ao modelo aprender essa tarefa.

4.1.4 Codificador ViT

Nesta etapa, a metodologia foi fortemente inspirada pelo Vision Transformer (Dosovitskiy et al., 2021). No Vision Transformer, eram feitos recortes inteiros da imagem, e seu *embedding* posicional correspondia à posição desse recorte na imagem original. Neste caso, ao invés de realizar um recorte direto na imagem, o recorte é feito em partes específicas dela, de acordo com as *bounding boxes*.

Portanto, o contexto para criar o *embedding* posicional são as *bounding boxes* de cada objeto em cada quadro. Vale ressaltar que este é aprendido durante o treinamento. Além deste, também há o contexto dos IDs, que se refere aos trajetos de cada objeto. Isto é feito utilizando mais um *embedding* aprendido, que é somado às projeções dos recortes obtidos dos quadros anteriores.

Essas escolhas buscam guiar o transformer a entender onde os objetos estavam nos quadros passados e como são esses objetos. A projeção linear utiliza os recortes achatados para gerar vetores com 768 valores, onde cada um deles ocupa uma posição da sequência.

4.1.5 Cabeças de saída do modelo

Seguindo um conceito apresentado no artigo do Vision Transformer (Dosovitskiy et al., 2021), onde os autores desconsideram as saídas do encoder correspondentes às partes da imagem que foram dadas como sequência de entrada para o modelo e utilizam apenas uma posição definida como responsável por classificar as imagens, serão consideradas apenas as 100 primeiras posições da sequência para gerar as saídas correspondentes aos objetos encontrados e rastreados.

As cabeças de saída do modelo são compostas por camadas MLP; especificamente, são redes com 3 camadas totalmente conectadas, cada uma com 768 nós. A cabeça de detecção possui quatro saídas, correspondendo às posições centrais, à largura e à altura de cada *bounding box*. A cabeça de rastreamento possui uma saída softmax com 100 posições, correspondentes à limitação de 100 objetos rastreados por vez.

4.2 Configurações do Modelo OneTrack

Para treinar este modelo, existem alguns parâmetros que podem ser ajustados conforme as necessidades específicas. Um desses parâmetros é a quantidade de quadros considerados para contextualizar o quadro atual. O valor utilizado para os testes deste trabalho foi de 4 quadros. Deve-se ter cuidado para que a janela de análise não seja muito grande, pois se houver muitos objetos únicos nessa janela, isso prejudicará a capacidade do modelo de prever novos objetos. Isso ocorre devido a outro fator: o tamanho da sequência determina quantos objetos podem ser rastreados por quadro.

Seguindo a definição apresentada por Carion et al. (2020), foi escolhido um limite de 100 objetos únicos. Este é um valor razoável, pois dificilmente haverá tantos objetos diferentes ao longo da janela de análise e dificilmente haverá tantos objetos sendo encontrados de uma só vez por quadro.

A escolha do modelo extrator de características poderia ter sido diferente. Poderiam ter sido utilizados modelos que não necessariamente foram projetados para detecção de objetos, ou ainda, como no Vision Transformer, poderia ser utilizada a imagem pura, obtendo uma projeção linear de cada recorte da imagem e utilizando isso como sequência de entrada para o encoder.

A parte do encoder foi exatamente a mesma proposta por Dosovitskiy et al. (2021) para o modelo base com recortes de tamanho 16, treinado no *dataset* ImageNet (Murad and Pyun, 2017). Utilizar o modelo com pesos iniciados aleatoriamente não gerou resultados satisfatórios, mas isso já era esperado, pois para treinar transformers é necessária uma grande quantidade de dados, a qual não estava disponível para os testes conduzidos neste trabalho.

O treinamento foi realizado utilizando o otimizador AdamW, com taxa de aprendizado fixa em 0.0002, a mesma configuração utilizada no treinamento do MOTRv2 (Zhang et al., 2023).

4.2.1 Funções de Perda

Para ajustar os pesos do modelo durante o treinamento, foi utilizada uma combinação ponderada de funções de perda, exatamente como proposto por Carion et al. (2020). Primeiramente, é realizado o chamado *bipartite matching*, que visa fazer correspondências entre as previsões e os rótulos antes do cálculo das perdas. As três funções de perda utilizadas foram:

1. **Huber Loss:** Esta função de perda tenta relacionar as funções de erro absoluto médio e erro quadrático médio para avaliar as *bounding boxes* inferidas a cada época. Ela foi essencial para permitir que o modelo aprendesse as relações entre as coordenadas e os objetos encontrados. Esta função está descrita mais detalhadamente abaixo:

$$L(a) = \begin{cases} \frac{1}{2}a^2 & \text{para } |a| \leq \delta \\ \delta(|a| - \frac{1}{2}\delta) & \text{caso contrário} \end{cases} \tag{4.1}$$

   Onde:

   - a é o erro, isto é, a diferença entre o valor previsto e o valor real.
   - δ é um valor de limiar. Para erros menores que δ, a perda Huber é quadrática. Para erros maiores que δ, a perda é linear.

2. **Sigmoid Focal Loss:** Esta função tenta penalizar os IDs errados, considerando tanto casos de falta de IDs, IDs relacionados erroneamente ou até mesmo sobrecarga de IDs. Ela foi inicialmente proposta para abordar o problema do desequilíbrio de classes em tarefas de detecção de objetos. A Sigmoid Focal Loss é definida como:

$$FL(p_t) = -\alpha_t(1 - p_t)^\gamma \log(p_t) \tag{4.2}$$

   Onde:

   - p_t é a probabilidade da classe verdadeira. Se a classe verdadeira é 1, p_t é igual a p (a probabilidade prevista de 1), caso contrário é $1 - p$.
   - α_t é um coeficiente de balanceamento. Geralmente, é definido como α se a classe verdadeira é 1 e $1 - \alpha$ caso contrário.

- γ é o parâmetro de foco que regula o peso das classificações erradas. Valores maiores de γ reduzem o peso relativo das classificações fáceis e aumentam o peso das classificações difíceis.

A principal vantagem da Focal Loss é sua capacidade de focar no treinamento de exemplos difíceis, permitindo que modelos de detecção treinem de forma mais eficaz em cenários desequilibrados.

3. **Generalized Box IoU Loss:** Esta função também foi utilizada para ajustar o cálculo das bounding boxes, convertendo os valores do formato de treinamento de centroides e altura/largura para topo, esquerda, baixo, direita.

$$GIoU = IoU - \frac{A_c - A_{union}}{A_c} \tag{4.3}$$

Onde:

- IoU é a métrica Intersection over Union tradicional.
- A_c é a área da caixa delimitadora que cobre ambas as caixas delimitadoras previstas e verdadeiras.
- A_{union} é a área da união das caixas delimitadoras previstas e verdadeiras.

A perda GIoU é, então, a diferença entre a métrica IoU e a proporção da área que está fora das caixas delimitadoras previstas e verdadeiras, mas dentro de A_c. A perda GIoU tem um valor entre -1 e 1, e um valor menor indica uma melhor correspondência entre as caixas delimitadoras previstas e verdadeiras. A principal vantagem da perda GIoU é sua capacidade de fornecer uma penalização para caixas delimitadoras que estão próximas, mas não se sobrepõem, permitindo uma localização mais precisa dos objetos.

Por fim, é calculada a média ponderada entre essas perdas, com peso 5 para o erro Huber, e peso 2 para a Sigmoid Focal Loss e para a Generalized Box IoU Loss.

4.2.2 Arquitetura OneTrack-M

Semelhantemente ao OneTrack original, camadas específicas são adicionadas a este modelo para funções chave de rastreamento de múltiplos objetos, detalhadas abaixo:

- **Cabeça de Heatmap**: Identifica centros de objetos em cada imagem.
- **Cabeça de Dimensão**: Estima as dimensões (altura e largura) dos objetos detectados.
- **Cabeça de Deslocamento de Centro**: Calcula as variações nas posições dos centros dos objetos entre quadros.

A Figura 5 ilustra a arquitetura completa, mostrando a interconexão desses componentes. As seções seguintes descrevem melhor cada elemento, esclarecendo suas funções e o papel que desempenham no contexto geral do modelo OneTrack-M.

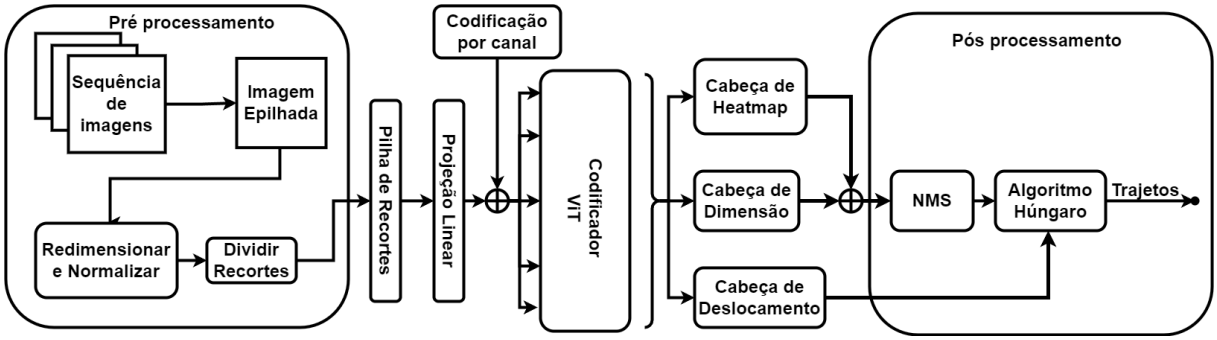

Figura 5 – Arquitetura implementada para o modelo OneTrack-M.

4.3 Preprocessamento

A etapa inicial do nosso modelo envolve um tratamento específico dos dados para apresentá-los adequadamente à rede neural. O principal desafio aqui é fornecer contexto temporal entre quadros selecionados. Isso é feito empilhando todas as imagens de uma janela temporal ao longo da dimensão do canal. Para uma imagem padrão com dimensões $[3, Largura, Altura]$, considerando um tamanho de janela W, o formato de

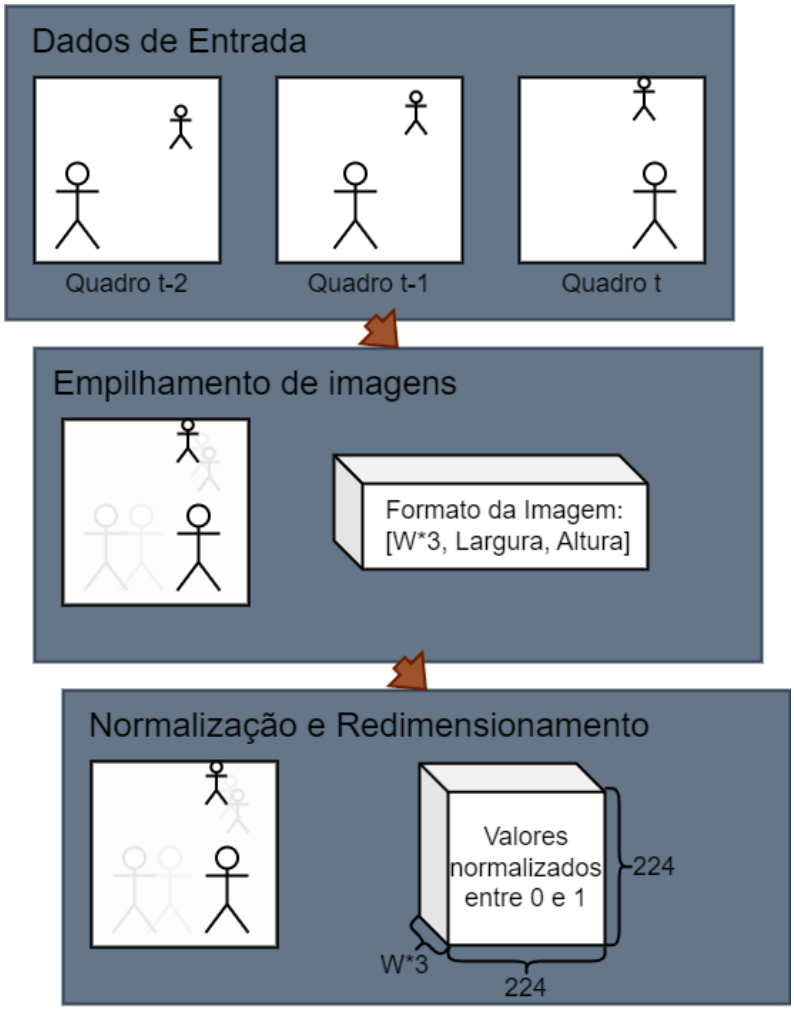

Figura 6 – Representação detalhada da etapa de preprocessamento do modelo.

entrada torna-se $[3 * W, Largura, Altura]$. A partir dessas imagens, são gerados recortes que posteriormente são convertidos em tokens através de uma camada linear.

Essa metodologia permite que a rede neural absorva completamente o contexto temporal, alinhando-se às estratégias de preprocessamento adotadas em trabalhos anteriores como Dosovitskiy et al. (2021), mas adaptando-as para análise de vídeo neste projeto. Os componentes desta seção do modelo são ilustrados na Figura 6. Vale notar que não é necessário nenhum input adicional para a rede além das imagens, normalizadas e redimensionadas para uma matriz quadrada de tamanho 224, que é o mesmo utilizado no modelo ViT de referência.

4.3.1 Empilhamento de Imagens e Geração de Recortes

De acordo com o procedimento descrito pelos autores do Vision Transformer, esta etapa consiste em, dado um tamanho quadrado (16x16 ou 32x32), dividir igualmente a imagem em recortes. Este trabalho manteve o tamanho de imagem 224x224, novamente compatível com o procedimento experimental do trabalho base mencionado anteriormente. Assim, para recortes de tamanho 16x16, obtêm-se 196 pequenas imagens $(\frac{224}{16})^2$. Cada uma passa por uma camada linear para gerar uma projeção vetorial de cada recorte, com tamanho 768.

Essa etapa torna-se eficiente porque usa uma camada convolucional com saída de 768 canais, um kernel e um stride correspondendo ao tamanho do recorte. Assim, cada etapa da convolução processa um recorte, seja 16x16 ou 32x32, sem sobreposição, e então projeta as camadas para o tamanho desejado, ou seja, esses recortes quadrados tornam-se um vetor de tamanho 768. Na Figura 5 essas etapas são ilustradas nos blocos Imagens Empilhadas, Empilhamento de recortes e Projeção Linear.

4.3.2 Normalizações

No contexto de rastreamento, que depende da análise de deslocamentos dos centros dos objetos, e considerando um fluxo de vídeo a 60 quadros por segundo, uma janela temporal de 5 quadros representa apenas 83 milissegundos. Os deslocamentos observados entre quadros consecutivos tendem a mostrar magnitudes relativamente pequenas nesse curto intervalo. Para aprimorar a capacidade preditiva do modelo, dado essa característica, os valores de deslocamento foram normalizados para o intervalo $[-1, 1]$. Os parâmetros de normalização foram determinados por meio de uma análise exploratória do benchmark no conjunto de dados MOT17. Se o modelo for adaptado para diferentes contextos ou conjuntos de dados, é necessário revisar e ajustar os valores de normalização considerando as peculiaridades dos deslocamentos dos novos dados.

Os valores específicos usados na normalização são detalhados na Tabela 2, fornecendo uma referência quantitativa para esta etapa crucial de preprocessamento. Esta prática não só facilita a interpretação dos dados pelo modelo, mas também assegura

uma adaptação mais eficiente às variações sutis de movimento, melhorando a precisão do rastreamento.

Tabela 2 – Deslocamentos observados no conjunto de treinamento MOT17 (Milan et al., 2016).

Deslocamento Min. X	Deslocamento Máx. X	Deslocamento Min. Y	Deslocamento Máx. Y
-0.0174	0.0057	-0.0157	0.0166

4.4 Codificação por Canal

Apesar do esforço para incorporar contexto temporal, o modelo pode enfrentar dificuldades devido à falta de uma indicação clara da ordem sequencial dos recortes de imagem. Portanto, fornecer um contexto posicional explícito é crucial para otimizar o desempenho em modelos baseados em atenção.

Neste cenário, optamos por uma abordagem inovadora através da implementação de uma técnica chamada Codificação por Canal. Este método não se limita a codificações posicionais simples, fixas ou aprendidas, mas busca uma representação mais abrangente da estrutura espacial e temporal dos dados. Especificamente, adiciona *embeddings* únicos para cada canal de recorte de imagem, refletindo a janela temporal no processamento de vídeo.

A Codificação por Canal inova ao atribuir um vetor de *embedding* distinto para cada quadro dentro da janela temporal, distinguindo não apenas as características espaciais, mas também a posição temporal de cada recorte dentro da sequência. Essa abordagem é vital para que o modelo entenda o movimento e a ordem dos objetos, uma vez que a sequência de imagens fornece informações cruciais para o rastreamento.

Inicialmente, essas codificações são um tensor de parâmetros treináveis com dimensões $[W, n_d]$, onde W é o tamanho da janela temporal e n_d é a dimensão do *embedding*. Este vetor é então adicionado ao vetor derivado dos recortes de imagem. Consequentemente, essa estratégia não só facilita a diferenciação entre os quadros, mas também melhora a capacidade do modelo de capturar dinâmicas temporais. Representa uma solução eficaz para integrar informações temporais em modelos baseados

em transformers projetados para rastreamento de múltiplos objetos, com o benefício adicional de minimizar o impacto no tempo de inferência.

Esses vetores são adicionados às projeções lineares de cada recorte obtidas na etapa de preprocessamento anterior.

4.5 O Codificador ViT

A estratégia adotada pelo modelo envolve o uso do codificador do Vision Transformer pré-treinado no ImageNet (Murad and Pyun, 2017) para extrair mapas de atenção, convertendo cada recorte de imagem em um elemento da sequência de entrada. Diferente da aplicação convencional do ViT para classificação, o *token* de classificação é omitido neste contexto. Esta modificação adapta o ViT para funcionar como um extrator de características focado em entender o contexto espacial e temporal, mantendo as configurações originais da rede pré-treinada.

Essa abordagem é significativamente diferente dos métodos tradicionais descritos na literatura, que frequentemente empregam uma arquitetura de codificador-decodificador, como evidenciado em trabalhos como Sun et al. (2021), Meinhardt et al. (2022), Zeng et al. (2022), e Zhang et al. (2023). Embora robustas, essas soluções enfrentam o desafio de longos períodos de inferência. Simplificando para apenas o codificador e transferindo a responsabilidade da saída final para um componente mais eficiente, o OneTrack-M conseguiu reduzir significativamente o tempo de processamento.

A escolha de uma rede pré-treinada é justificada pelo grande volume de dados necessário para o treinamento eficaz de redes transformer. A limitação dos dados disponíveis no conjunto de dados de benchmark usado, combinada com preocupações de que incluir dados externos poderia comprometer a comparabilidade entre diferentes modelos, reforça a relevância dessa decisão. Assim, usar o ViT pré-treinado surge como uma solução pragmática, permitindo aproveitar aprendizados anteriores sem exigir conjuntos de treinamento extensivos, alinhando-se aos objetivos de eficiência e eficácia do OneTrack-M.

4.6 Cabeças de Saída

A estrutura de saída do OneTrack-M consiste em três componentes principais: heatmaps, dimensões e deslocamentos. Como mostrado na Figura 7, cada uma dessas cabeças compreende uma sequência de duas camadas totalmente conectadas seguidas por duas camadas convolucionais, especificamente ativadas para cada tipo de saída. Para heatmaps e dimensões de objetos, é usada a ativação sigmoide, enquanto para deslocamentos, a ativação é tangente hiperbólica para representar a faixa de valores $[-1, 1]$ possível para essa saída.

A etapa de processamento de heatmaps que ajusta seu formato para o esperado pelas cabeças de saída é ilustrada na Figura 8. Para esta etapa, processamos os mapas de atenção em uma dimensão menor usando uma série de camadas totalmente conectadas, resultando em um tensor de forma [batch, 196*W, 196], que é reformatado para [batch, W, 196, 196]. Este tensor final é a entrada usada para realizar a inferência nas três cabeças implementadas.

Esta configuração visa detectar grades de informação, alinhando-se ao método de saída proposto pelo CenterTrack (Zhou et al., 2020). A diferenciação reside no tratamento da oclusão: ao contrário da linha de base, que é limitada a duas imagens consecutivas, nossa abordagem empilha múltiplos quadros para entrada. Isso permite que o modelo preserve o contexto de deslocamento dos objetos por um período mais longo, mesmo na presença de oclusões.

4.6.1 Cabeça de Heatmap

A saída de heatmap identifica centros de objetos, permitindo o rastreamento de objetos ilimitados—uma vantagem significativa sobre outras abordagens de MOT com transformers, como Zeng et al. (2022) e Zhang et al. (2023). Para gerenciar a complexidade, a dimensão da imagem é reduzida antes da inferência e depois expandida de volta às suas dimensões originais. Esse procedimento é consistente em todas as cabeças do modelo. A ativação para esta saída é feita por uma função sigmoide, limitando os valores entre 0 e 1.

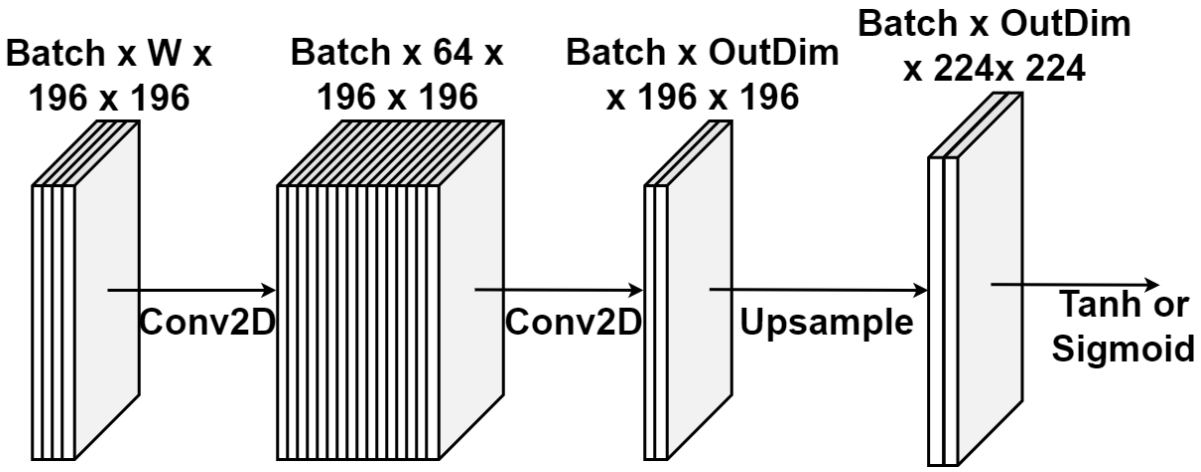

Figura 7 – Esquema detalhado da rede responsável pela última parte das cabeças do modelo. Cada cabeça tem a mesma estrutura, variando apenas na função de ativação e na dimensão de saída. Para a de *heatmaps*, $OutDim = 1$, enquanto para as outras duas, $OutDim = 2$.

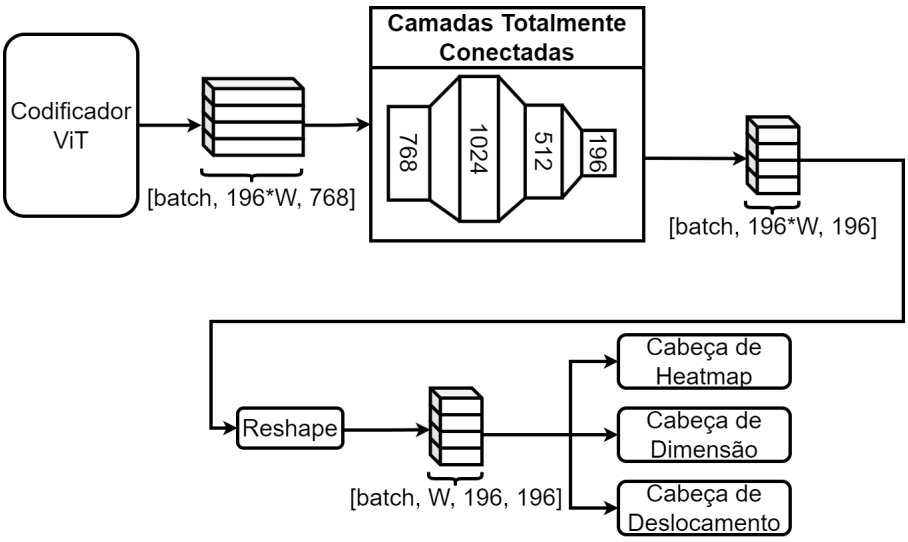

Figura 8 – Esquema da etapa de saída após os dados serem processados pelo Vision Transformer. O primeiro bloco de tensor representa os mapas de atenção extraídos pelo modelo. Cada bloco é uma sub-etapa desta etapa final do modelo.

A Figura 9 compara uma amostra do conjunto de dados com uma previsão do modelo, onde cada objeto é marcado por um pico no heatmap, suavizado por um filtro gaussiano conforme mostrado por Zhou et al. (2020) e Zhou et al. (2019).

Figura 9 – Exemplo de uma imagem com detecção de mapa de calor e seus picos. Uma concentração de pessoas é notável na região central, com áreas vazias nas regiões superior e inferior, demonstrando boa capacidade do modelo.

4.6.2 Cabeça de Dimensão

Esta cabeça estima as dimensões (largura e altura) de cada objeto usando uma grade de forma $[2, Largura, Altura]$. Cada elemento da matriz corresponde às dimensões de um objeto em sua localização central. Embora as saídas sejam processadas simultaneamente, ajustes finos são necessários para integrar esses resultados na inferência final. Esta saída é ativada por uma função sigmoide para manter a faixa entre 0 e 1. Esta cabeça é semelhante à implementada no CenterTrack (Zhou et al., 2020).

4.6.3 Cabeça de Deslocamento

Funcionando de forma semelhante à Cabeça de Dimensão, mas com pesos diferentes, esta parte do modelo calcula os deslocamentos dos centros dos objetos nas duas dimensões da imagem, considerando o quadro anterior. Mesmo com dados de instâncias anteriores, a regressão é feita apenas entre o momento atual e o imediatamente anterior. Considerando que o deslocamento pode ser em qualquer direção, esta cabeça é ativada por uma função tangente hiperbólica, tornando a faixa de valores de saída -1 a 1.

A Figura 10 apresenta a saída completa do modelo, incluindo centros de objetos, dimensões e deslocamentos. Esses deslocamentos são essenciais para associar objetos a trajetórias específicas, permitindo um rastreamento eficaz ao longo do tempo.

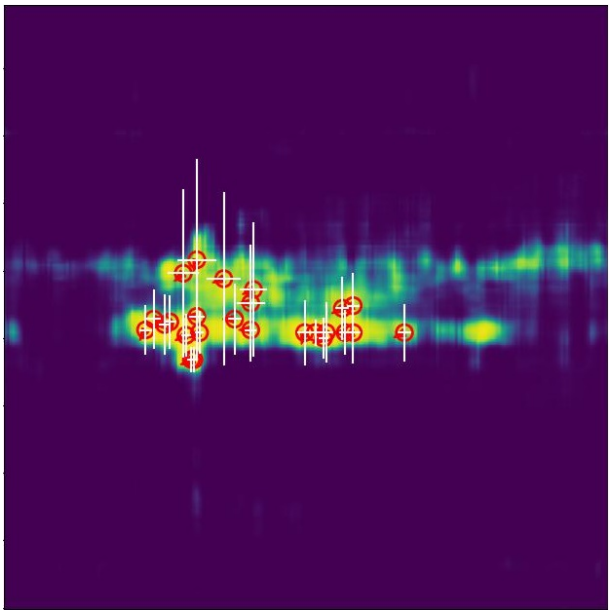

Figura 10 – Imagem completa de saída do modelo. Aqui você pode ver o tamanho de cada objeto detectado (barras brancas) e a direção do movimento (setas vermelhas) a partir de seus centros.

4.7 Pós-Processamento e Associação de Dados

Cada cabeça do modelo representa uma peça da saída completa do MOT. No entanto, duas etapas adicionais são necessárias para construir totalmente a rotina de rastreamento. A primeira envolve processar e unir todas as saídas do modelo, enquanto a segunda emprega uma abordagem comum para associar as inferências atuais com o histórico de rastreamento dos objetos.

4.7.1 Combinando Saídas

Para integrar as saídas do modelo, um procedimento específico é seguido:

1. **Aplicar um Limite no Heatmap**: Esta etapa identifica os centros de objetos detectados aplicando um valor de corte considerando que o valor no pixel representa a proabilidade de ser um centroide.

2. **Determinar Dimensões e Deslocamentos dos Objetos**: Usando os centros identificados, determinam-se as dimensões e os deslocamentos dos objetos.

Na Figura 9 vemos as saídas brutas de cada detecção encontrada pelo modelo em um determinado quadro. É possível ver que algumas dessas detecções estão sobrepostas ou não são precisas. Isso é resolvido aplicando o algoritmo de Supressão de Máximos Não Relevantes (NMS) para mitigar falsos positivos, suprimindo detecções sobrepostas e de baixa confiança. Este método é amplamente utilizado em sistemas de detecção de objetos, incluindo variantes do YOLO, como Wang et al. (2022).

Dado um conjunto de caixas delimitadoras B com seus respectivos escores de confiança S e um limite de sobreposição T, o algoritmo NMS prossegue da seguinte forma:

1. **Selecionar a Caixa com Maior Escore de Confiança**: Seleciona a caixa delimitadora b_{max} com o maior escore de confiança s_{max} de S.

2. **Comparar com Outras Caixas**: Compara b_{max} com cada caixa b_i em BB, exceto ela mesma.

3. **Calcular a Interseção sobre União (IoU)**: Para cada comparação, calcula a Interseção sobre União (IoU) entre b_{max} e b_i, denotada por $IoU(b_{max}, b_i)$.

4. **Supressão**: Se $IoU(b_{max}, b_i) > T$, a caixa b_i é considerada redundante e removida de B.

5. **Repetir**: Repete os passos 1 a 4 até que todas as caixas em B sejam avaliadas ou removidas.

A Figura 11 ilustra as previsões antes e depois do NMS, demonstrando a eficácia do algoritmo em reduzir falsos positivos. A imagem (a) mostra objetos detectados imediatamente após a imagem ser processada pelo modelo, enquanto a imagem (b) mostra após o algoritmo de redução de redundância.

Para rastrear objetos detectados, as detecções atuais devem ser associadas às anteriores. O algoritmo húngaro é um método de otimização para resolver problemas de atribuição. Seu objetivo é encontrar a correspondência perfeita com o custo mínimo entre elementos de dois conjuntos, minimizando o custo total de associação. No contexto do rastreamento de múltiplos objetos, ele é usado para associar detecções atuais com

Figura 11 – Imagem (a) mostra as detecções sobre a imagem antes de aplicar o passo NMS. Imagem (b) mostra como as detecções ficam após aplicar o NMS.

trajetórias existentes com base em uma métrica de distância ou similaridade (neste caso, o IoU entre caixas delimitadoras).

Formalmente, o algoritmo húngaro resolve o problema de atribuição da seguinte forma: Deixe C ser uma matriz de custo $n \times m$, onde n é o número de objetos detectados e m o número de trajetórias. O algoritmo busca uma permutação P de elementos m tal que a soma dos custos de atribuição $\sum_{i=1}^{n} C_{i,P(i)}$ seja minimizada, onde $P(i)$ é a trajetória atribuída ao objeto detectado i.

Dado um conjunto de objetos detectados D e um conjunto de trajetórias existentes T, o algoritmo prossegue da seguinte forma:

1. **Construir a Matriz de Custo**: Uma matriz de custo C é construída, onde cada elemento $C_{i,j}$ representa o custo de atribuir o objeto detectado d_i à trajetória t_j. O custo pode ser definido com base na distância euclidiana, IoU inverso ou qualquer outra métrica de dissimilaridade relevante.

2. **Aplicar o Algoritmo Húngaro**: O algoritmo é aplicado à matriz de custo para encontrar a atribuição de custo mínimo. Isso resulta em uma correspondência um-para-um entre objetos detectados e trajetórias, onde cada objeto é associado a no máximo uma trajetória e vice-versa.

3. **Atualizar Trajetórias**: Com base nas atribuições feitas, as trajetórias são atualizadas com novas detecções. Trajetórias sem detecções correspondentes podem ser marcadas como "perdidas"ou terminadas. Da mesma forma, detecções sem trajetórias correspondentes são marcadas como novos objetos.

4.8 Treinamento Multitarefa Baseado em Partes

Conforme discutido no Capítulo 3, os modelos de Rastreamento de Múltiplos Objetos (MOT) são frequentemente treinados de ponta a ponta usando uma estratégia de aprendizado multitarefa. No entanto, os resultados apresentados neste estudo, consistentes com as observações do FairMOT (Zhang et al., 2021), indicam que simplesmente atribuir uma função de perda específica para cada tarefa não garante o desempenho ótimo do modelo. Em resposta a essa limitação, foi adotada uma metodologia de treinamento que alterna entre fases de aprendizado individualizado de tarefas e fases de treinamento conjunto, conhecida como Treinamento Multitarefa Baseado em Partes (TMP).

A rotina de treinamento segue este processo, chamado Treinamento Multitarefa Baseado em Partes (TMP):

1. **Fase de Treinamento da Cabeça de Heatmap**: Inicialmente, a tarefa de geração de mapas de calor é isolada, congelando os pesos das cabeças de dimensão e deslocamento e anulando seus respectivos pesos na combinação ponderada das funções de perda. Esta etapa foca exclusivamente em aprender centros de objetos.

2. **Período de Treinamento**: Esta fase dura 50 épocas ou continua até que não se observe redução na função de perda por 10 épocas.

3. **Treinamento Individual de Outras Cabeças**: O processo é repetido para as cabeças de dimensão e deslocamento, treinando cada uma individualmente com os mesmos critérios de congelamento de pesos e duração.

4. **Treinamento Conjunto Final**: Após completar 150 épocas de treinamento focado, é conduzida uma fase final de treinamento conjunto de 50 épocas, onde todas as

camadas são ativadas e contribuem com seus pesos para a média ponderada das funções de perda.

Esta abordagem sequencial permitiu que o modelo aprendesse os detalhes específicos de cada tarefa em isolamento antes de integrar esse conhecimento em um aprendizado conjunto. Esse processo facilitou a compreensão inter-tarefa do modelo, resultando em desempenho superior ao final do treinamento.

4.8.1 Funções de Perda

Durante o treinamento do modelo de Rastreamento de Múltiplos Objetos (MOT), várias funções de perda específicas foram aplicadas para melhorar a precisão do modelo na detecção e rastreamento de objetos. Essas funções de perda são essenciais para orientar o modelo na detecção precisa dos centros dos objetos, bem como suas dimensões e deslocamentos. Para maior clareza, cada função de perda implementada é detalhada abaixo e expressa através de equações matemáticas.

4.8.1.1 Perda de Heatmap

A função de perda de heatmap é projetada para otimizar a detecção de centros de objetos no heatmap. Esta função consiste em duas partes principais: **Perda de Centro** e **Perda Focal**.

Perda de Centro é calculada da seguinte forma:

$$C_{loss} = -\frac{1}{N}\sum_{i=1}^{N}\left[P_i \cdot \log(\hat{P}_i) + (1 - P_i) \cdot \log(1 - \hat{P}_i)\right] \cdot W_i \tag{4.4}$$

onde P_i é a probabilidade real de um pixel ser um centro de objeto, $\hat{P}_i$ é a probabilidade prevista, N é o número total de pixels, e W_i representa pesos determinados pela importância de cada pixel. Esta função de perda é a mesma demonstrada pelo CenterTrack (Zhou et al., 2020).

Perda Focal é definida como:

$$\text{FocalLoss} = -\frac{1}{N}\sum_{i=1}^{N}(1-\hat{P}_i)^{\gamma}\cdot P_i\cdot\log(\hat{P}_i) \tag{4.5}$$

onde γ é o parâmetro que modula a perda para exemplos bem classificados, focando o aprendizado em exemplos mal classificados. Neste modelo, γ é definido como 4.

4.8.1.2 Perda de Grade

Esta função otimiza as grades que representam dimensões e deslocamentos dos objetos. Baseia-se na perda L1 entre previsões e valores reais, aplicada apenas onde os objetos estão presentes (máscara):

$$\text{GridLoss} = \frac{1}{M}\sum_{i=1}^{M}|G_i-\hat{G}_i| \tag{4.6}$$

onde G_i são os valores reais da grade, $\hat{G}_i$ são os valores previstos pela rede, e M é o número de elementos selecionados pela máscara.

Essas funções de perda permitem que o modelo aprenda com precisão a localização, dimensões e deslocamentos dos objetos, facilitando um rastreamento eficaz em sequências de imagens. Elas são combinadas usando uma média ponderada. Na equação abaixo, vemos como o valor final da perda é calculado para cada lote de imagens:

$$Loss_f = \frac{(CenterLoss + FocalLoss) * w1 + (GridLoss_{dim}) * w2 + (GridLoss_{dlc}) * w3}{w1 + w2 + w3} \tag{4.7}$$

onde $w1 = w2 = w3 = 1.0$ para o treinamento final. Como descrito no TMP, esses pesos variam para 0 dependendo da fase de treinamento. $GridLoss_{dim}$ refere-se à Perda de Grade aplicada à saída de dimensão, e $GridLoss_{dlc}$ à Perda de Grade aplicada à saída de deslocamento.

4.9 Considerações Finais

Este capítulo detalhou o desenvolvimento e a implementação dos modelos OneTrack e OneTrack-M, focado no rastreamento de múltiplos objetos, abrangendo desde a conceituação teórica até a aplicação prática das técnicas de aprendizado profundo. As arquiteturas foram cuidadosamente desenhadas para explorar os benefícios do *Vision Transformer*, empregando-o na análise de sequências de imagens, otimizando a detecção e o rastreamento de objetos através de um conjunto específico de cabeças de saída (especificamente no OneTrack-M): mapas de calor, grades de tamanhos e deslocamentos.

A introdução de uma rotina de treinamento multitarefa por partes (TMP) representou uma inovação significativa no processo de aprendizado do modelo, a fim de garantir uma estabilidade melhor ao modelo durante seu treinamento.

Os desafios enfrentados na implementação das codificações suplementares e na normalização dos dados destacaram a importância de um pré-processamento adequado, essencial para a eficácia do modelo em cenários de rastreamento realistas. Além disso, a abordagem adotada para a associação de dados, utilizando o algoritmo Húngaro, provou ser crucial para a manutenção das trajetórias dos objetos ao longo das sequências de imagens com pouco *overhead*.

5

RESULTADOS OBTIDOS

Esta seção detalha os experimentos conduzidos e os resultados obtidos, fornecendo detalhes sobre o desempenho dos modelos OneTrack e OneTrack-M. O foco desta seção está no segundo modelo pois este trouxe melhorias consideráveis sobre a versão anterior, decisões tomadas especificamente para solucionar alguns problemas conceituais levantados pela primeira iteração desta arquitetura.

5.1 Configuração Experimental

Os experimentos foram realizados em um computador equipado com um processador Intel Core i7-11700 a 3.60GHz e 32GB de RAM, rodando o sistema operacional Ubuntu 20.04.6 LTS 64 bits. Para o treinamento e inferência do modelo, foi utilizada uma placa gráfica Geforce RTX 2060 com suporte a CUDA 11.0. A escolha da placa gráfica foi baseada na pontuação de capacidade de computação fornecida pelo fabricante, garantindo uma capacidade de computação comparável à Nvidia V100, comumente usada em trabalhos relacionados. Essa aproximação assegura a validade da comparação dos tempos de inferência com resultados publicados anteriormente.

5.1.1 Pareamento de performance entre diferentes GPUs

Para garantir a integridade dos testes de performance conduzidos neste trabalho, corroborando com o tema previamente estabelecido, um dos modelos utilizados como *baseline*, cujos valores de FPS reportados em seus artigo por Zhang et al. (2021), foi executado sob as configurações utilizadas neste trabalho. Sendo assim, utilizando uma GPU Nvidia V100 para inferência, sob as condições apontadas pelos autores, espera-se taxa de quadros por segundo de 25.9. Ao executar no mesmo conjunto de dados, desta vez sob as condições deste trabalho, foi possível verificar um aumento na taxa de quadros por segundo deste modelo para 26.3. Isto indica uma diferença de 1.5%. Sendo assim os valores reportados neste trabalho serão ajustados por esta taxa para garantir uma comparação mais justa em relação aos valores reportados em seus respectivos trabalhos originais.

5.2 Conjunto de Dados

O *benchmark* MOT17 (Milan et al., 2016) foi o conjunto de dados utilizado para validar os resultados deste trabalho. Ele foi escolhido por ser amplamente adotado em pesquisas de MOT para manter uma linha de base comparativa consistente. O MOT17 inclui conjuntos de dados de treinamento e teste predefinidos. As imagens cobrem diversos cenários, variando em proximidade de objetos e movimento de câmera, focando no rastreamento de pessoas em ambientes moderadamente densos. A Figura 12 exemplifica um quadro do conjunto de dados, que contém 18 sequências de imagens anotadas com coordenadas de caixas delimitadoras e identificações de objetos.

O MOT17 contém um total de 18 sequências de imagens com anotações de coordenadas de caixas delimitadoras e IDs dentro da sequência. As anotações seguem um formato de topo, esquerda, altura e largura para cada objeto, bem como sua confiança de predição e o respectivo ID (novo ou não). A Figura 12 mostra um exemplo de uma amostra deste conjunto de dados, contendo uma sequência de 3 exemplos separados por 60 quadros.

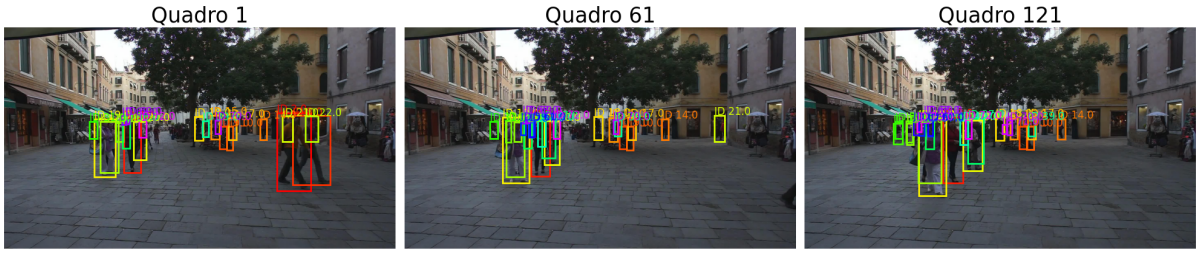

Figura 12 – Sequência de imagens e anotações de exemplo retiradas do conjunto de dados MOT17.

5.3 Métricas de Avaliação

Um conjunto de métricas baseado em dois padrões estabelecidos na literatura foi adotado: Clear MOT e HOTA. Clear MOT, detalhado em Bernardin and Stiefelhagen (2008), aproxima o desafio de rastreamento a conceitos como acurácia, precisão e recall, enquanto HOTA, introduzido em Luiten et al. (2020), foca em aspectos específicos de MOT, como manutenção consistente de trajetória. A biblioteca TrackEval, de Jonathon Luiten (2020), foi utilizada para avaliar as saídas do modelo de acordo com essas métricas.

As métricas utilizadas são:

- **MOTA (Multiple Object Tracking Accuracy)**: Uma medida agregada que considera falsos positivos, falsos negativos e erros de identificação para calcular a precisão geral do rastreamento. É expressa como:

$$\text{MOTA} = 1 - \frac{\sum_t (FN_t + FP_t + IDSW_t)}{\sum_t GT_t} \tag{5.1}$$

  onde FN_t, FP_t e $IDSW_t$ representam o número de falsos negativos, falsos positivos e trocas de identificação em cada passo de tempo t, respectivamente, e GT_t é o número total de objetos.

- **HOTA (Higher Order Tracking Accuracy)**: Combina capacidades de detecção e associação em uma única métrica, equilibrando desempenho em ambas as dimensões.

$$\text{HOTA} = \frac{1}{T} \sum_{t=1}^{T} \left(\frac{\sum_{(i,j) \in \mathcal{M}_t} \text{IOU}(i,j)}{|\mathcal{M}_t|} \cdot \frac{|\mathcal{M}_t|}{|\mathcal{G}_t| + |\mathcal{P}_t| - |\mathcal{M}_t|} \right) \tag{5.2}$$

Nesta equação, $\mathcal{M}_t$ representa o conjunto de pares correspondentes verdade-predição no tempo t, $\mathcal{G}_t$ é o conjunto de objetos verdadeiros, $\mathcal{P}_t$ é o conjunto de objetos previstos, e IOU(i, j) é a Interseção sobre União entre o objeto verdadeiro i e o objeto previsto j. T é o número total de passos de tempo.

- **IDS (ID Switches)**: Quantifica o número de vezes que a identificação de um objeto muda ao longo de sua trajetória. É um indicador da consistência na manutenção das identidades dos objetos.

- **IDF1**:

$$\text{IDF1} = \frac{2 \times \text{IDTP}}{2 \times \text{IDTP} + \text{IDFP} + \text{IDFN}} \tag{5.3}$$

- **FPS (quadros por segundo)**: Esta métrica relaciona-se a um termo usado para quantificar a velocidade de captura de vídeo. No contexto deste trabalho e medições de tempos de inferência para modelos de processamento de vídeo, esta métrica define quantos quadros por segundo o modelo pode inferir, ou seja, uma medida de frequência, sendo o inverso do tempo médio de inferência do modelo para uma dada sequência:

$$\text{FPS} = \frac{1}{I_t/N} \tag{5.4}$$

onde I_t é o tempo total de inferência de uma sequência em segundos e N é o número de quadros nesta sequência.

Essas métricas fornecem uma avaliação abrangente do desempenho do modelo, desde a precisão na detecção de objetos até a eficácia na manutenção de suas trajetórias e identidades ao longo do tempo. Além dessas métricas, o tempo médio de inferência do modelo neste conjunto de dados também foi avaliado, medido a partir do número de quadros processados por segundo, ou FPS. Esta medida inclui o preprocessamento, inferência, pós-processamento e processo de associação.

5.4 Resultados dos Testes de Benchmark - OneTrack-M

A Tabela 3 mostra os resultados obtidos pelo modelo em comparação com outros modelos estabelecidos na literatura. Em relação ao HOTA, nossa abordagem obteve 65.105% de desempenho, sendo superior aos demais. Isso mostra que tanto o processo de detecção quanto o de associação estão funcionando bem. Para IDF1, nossa configuração de modelo alcançou 79.608%, 0.108% superior aos segundos modelos com melhor desempenho, StrongSORT e BoTSORT, ambos com 79.5%. Um IDF1 alto indica um forte desempenho em associar objetos detectados às suas identidades reais. Um baixo número de trocas de ID (IDS) implica que o algoritmo de rastreamento é bom em manter a identidade dos objetos rastreados, significando que o rastreador é confiável na associação dos IDs corretos ao longo dos quadros. Neste teste, temos cerca de 100 ocorrências a menos de trocas de ID em comparação com o segundo maior, StrongSORT, com 1194. Finalmente, FPS representa a rapidez com que o modelo realiza suas inferências. Este é o ponto mais forte do nosso trabalho, pois os modelos que alcançam métricas mais altas tendem a ter um grande impacto na velocidade de execução. Com 35.7 FPS, nosso modelo seria capaz de rodar em tempo real considerando filmagens capturando 30 FPS em um cenário prático.

Falsos positivos e falsos negativos na etapa de detecção podem ter um grande impacto no MOTA, enquanto as outras métricas não são tão afetadas por falsos rastreamentos. Isso indica que o modelo pode efetivamente rastrear objetos que encontra, mas está vendo algumas detecções erradas ou redundantes, afetando a métrica MOTA, mas isso não é levado em consideração pelas outras métricas, que valorizam mais os rastreamentos corretos.

A configuração selecionada para este resultado foi considerada a mais equilibrada entre desempenho nas métricas e o menor tempo de inferência. Neste caso, 5 imagens por inferência, com TMP e usando o ViT-Base-16 como *backbone*.

Estes resultados destacam principalmente a capacidade do modelo de manter trajetórias, ou seja, reconhecer quais objetos estão correlacionados ao longo do tempo, demonstrando que os mecanismos desenvolvidos para isso durante a concepção do modelo foram relevantes para esses resultados. Outro ponto relevante é o tempo de

Tabela 3 – Comparação de Resultados entre modelos estabelecidos na literatura no conjunto de dados MOT17

Modelo	HOTA	MOTA	IDF1	IDS	FPS
OneTrack-M	**65.11**	67.95	**79.61**	**1090**	35.7
OneTrack	61.3	64.7	64.5	2994	**43.5**
Bytetrack	63.1	80.3	77.3	2196	11.97
CenterTrack	52.2	67.8	64.7	3039	17.76
MOTR	-	65.1	66.4	2049	7.6
MOTRV2	62.0	78.6	75.0	-	7,0
TrackFormer	-	74.1	68.0	2829	7.5
TransTrack	54.1	75.2	63.5	3603	10.15
BoTSORT	64.6	**80.6**	79.5	1257	6.7
StrongSORT	68.9	78.2	80.4	-	8.4
FairMOT	-	73.7	72.3	3303	26.28
LMOT	-	72.0	70.3	3071	28.92

processamento, que, apesar de utilizar um modelo transformer, ao realizar outras etapas de forma eficiente e simplificar o processo em uma única passagem, relatou o melhor tempo de inferência entre os modelos considerados para este trabalho. Esta versão do modelo apresenta as seguintes configurações: usando o método TMP, Vision Transformer versão Base-16 e uma janela de 5 imagens por vez. Isso provou ser a configuração mais estável, pois equilibra o tempo de processamento, desempenho qualitativo e a capacidade de correlacionar informações relevantes na janela de tempo para cada objeto.

Vale ressaltar a comparação dos resultados da versão -M do modelo em relação à versão original. Houve uma concessão na métrica de FPS, produzindo inferências aproximadamente 18% mais lentas, enquanto todas as outras métricas apresentaram melhorias. Observou-se um aumento de aproximadamente 6% na métrica HOTA, 5% na métrica MOTA e 23% na métrica IDF1, além de uma redução de 64% na quantidade de trocas de ID. Esses resultados demonstram a solução de um problema latente na abordagem inicial, onde era possível haver mais de um objeto com o mesmo ID, propiciando erros em vários quadros subsequentes. As qualidades de rastreamento do modelo demonstram melhora, embora de forma mais modesta, indicando que a abordagem baseada em transformers, extraindo características temporais a partir de um codificador pré-treinado, é eficaz em manter características espaço-temporais necessárias para correlacionar os mesmos objetos em movimento ao longo do tempo. As mudanças feitas

na modelagem dos dados para entrada e saída do modelo foram fundamentais para a melhora na identificação precisa de objetos com seus IDs únicos em uma mesma sequência de imagens.

5.5 Resultados dos Testes das Características do Modelo

Esses testes visam validar algumas das decisões tomadas para a versão final do modelo. Novamente, estas mudanças de parâmetros estão diretamente relacionadas à versão OneTrack-M do modelo proposto. Nesta parte do trabalho, avaliamos diferentes tamanhos de janela, treinamento com e sem TMP, diferentes configurações para o extrator de características e a aplicação de um embedding posicional usual (como o usado no ViT (Dosovitskiy et al., 2021)) ou os Embeddings por Canal propostos para fornecer contexto para cada corte na sequência da rede transformer.

5.5.1 Teste de Tamanho de Janela

Na Tabela 4, pode-se ver que o tempo de processamento não é afetado pelo tamanho de janela adotado. Isso faz sentido e está em linha com a forma como o modelo processa suas entradas, empilhando todas as imagens da janela ao longo da dimensão do canal RGB, usando efetivamente mais memória para processar mais imagens por janela. No entanto, não é aconselhável aumentar esse parâmetro descontroladamente, pois mais quadros dificultam a previsão de deslocamento pelo modelo, principalmente porque esse deslocamento deve ser a partir da posição do quadro anterior. Ao receber contexto de instâncias temporais muito distantes, nota-se um aumento nos valores de ID. Assim, um valor que mantém essa taxa de erro aceitável foi 5.

5.5.2 Teste de Eficácia do TMP

Para avaliar a capacidade deste método de treinamento, duas versões do modelo foram treinadas, uma treinando todas as funções de perda de uma vez, como normalmente

Tabela 4 – Tabela de resultados sobre tamanho de janela.

Tamanho da Janela	HOTA	MOTA	IDF1	IDS	FPS
1	45.291	45.822	56.379	5145	35.7
2	56.159	58.944	74.388	1449	35.7
3	56.316	61.813	76.969	1365	35.7
4	62.05	64.305	78.41	1253	35.7
5	65.105	67.950	79.608	1090	35.7
10	52.525	56.821	64.061	3095	35.7
20	52.409	56.627	63.919	3193	35.7

é feito, por 200 épocas. A outra versão foi treinada seguindo a rotina de treinamento proposta pelo TMP, treinando cada cabeça ou tarefa do modelo individualmente, congelando os pesos das outras cabeças, uma de cada vez, seguido por um treinamento final com todas as cabeças de uma vez. A Tabela 5 mostra os resultados de cada teste. Podemos ver que o modelo se beneficia grandemente de uma rotina de treinamento isolada para cada tarefa. Para este teste, todos os outros hiperparâmetros do modelo foram mantidos constantes, ou seja, tamanho da janela fixo em 5 imagens por vez, e a etapa de codificação pelo ViT usa o modelo B16, ou seja, o tamanho base da rede, com imagens recortadas para 16x16 pixels.

Tabela 5 – Comparação de resultados entre um modelo treinado com e sem a técnica TMP.

Com TMP?	HOTA	MOTA	IDF1	IDS	FPS
Não	49.915	55.398	72.371	2312	37.7
Sim	65.105	67.950	79.608	1090	37.7

5.5.3 Teste de Diferentes Extratores de Características Transformer

Uma das partes essenciais para o bom desempenho deste modelo é a escolha da arquitetura responsável pela extração de características. Isso ocorre porque a arquitetura requer essa etapa para ser capaz de entender conceitos nos eixos espacial e temporal. Portanto, é essencial que essa parte seja robusta o suficiente para lidar com esses aspectos simultaneamente.

Isso também traz uma questão relevante para o tempo de inferência geral do

sistema. Esta parte do modelo é responsável por quase todo o consumo de tempo do pipeline. Assim, aplicar uma rede com muitos parâmetros impacta muito o tempo de processamento por segundo do modelo. Isso é bem refletido nos resultados obtidos, mostrados na Tabela 6.

Para mais detalhes sobre a configuração de cada versão do ViT, elas devem ser referenciadas a partir do trabalho original em Dosovitskiy et al. (2021), mas brevemente, a principal diferença para os modelos Large é um maior número de parâmetros e camadas, enquanto os números 16 e 32 nos modelos se referem ao tamanho em que as imagens são recortadas. Isso implica que os modelos 32 têm menos capacidade de representação de detalhes, pois precisam resumir mais informações em cada token de entrada para a sequência do codificador. Por outro lado, esse tamanho maior por recorte significa menos elementos por imagem, resultando em um tempo de inferência mais curto.

Tabela 6 – Resultados de comparação do modelo variando a versão do Vision Transformer usada.

Versão ViT	HOTA	MOTA	IDF1	IDS	FPS
ViT-Base-16	65.105	67.950	79.608	1090	37.7
ViT-Base-32	60.310	65.466	71.230	1322	60.18
ViT-Large-16	68.019	75.020	80.052	1108	8.11
ViT-Large-32	65.019	71.020	81.166	1243	21.88

5.5.4 Teste de Embeddings Contextuais

Este teste visa validar o método apresentado neste trabalho em relação ao *embedding* que incorpora contexto espacial e temporal. Durante o preprocessamento, a sequência de imagens é empilhada na dimensão do canal, e aplicar essas representações a cada canal promove a percepção espaço-temporal pelo modelo. Uma prática comum para esta parte dos modelos transformer inclui o uso de *embeddings* espaciais, seja através de funções fixas sinusoidais que definem valores para cada posição na imagem original ou através de representações aprendíveis com pesos ajustados durante o treinamento. Os resultados deste teste são apresentados na Tabela 7.

Tabela 7 – Resultados de comparação do modelo variando o método de embedding.

Método de Embedding	HOTA	MOTA	IDF1	IDS	FPS
Posicional	44.678	45.207	58.223	4866	39.5
Por Canal	65.105	67.950	79.608	1090	37.7

Os resultados indicam que, ao lidar com dados de entrada, o modelo não identifica características temporais, e apenas o *embedding* posicional é usado. Mesmo com uma janela de inferência de 5 imagens, há dificuldades em conectar corretamente o movimento dos objetos na cena, resultando em falhas de rastreamento, conforme mostrado pela alta incidência de trocas de identificação.

5.6 Resultados da Análise Qualitativa

Na Figura 13, há uma sequência de imagens com suas respectivas inferências. Ela demonstra um problema chamado roubo de ID. Devido ao método usado para associar IDs ao longo do tempo, esse fenômeno se torna presente nas inferências, onde um erro suficientemente grande no valor de deslocamento para dois objetos muito próximos pode fazer com que um deles receba o ID do outro, enquanto se torna necessário criar um novo ID para o que perdeu seu identificador.

Figura 13 – Exemplo de um caso de roubo de ID. Note que entre a segunda e a quarta imagem, o objeto com id=2 o perde e recebe o ID do objeto ao lado como um novo valor.

A seguir, a Figura 14 mostra uma demonstração de uma limitação do modelo para objetos a distâncias médias a longas da câmera. Aqui podemos ver uma pessoa em uma bicicleta que não é detectada por vários quadros, enquanto há outros elementos que parecem estar posicionados próximos à pessoa em uma noção bidimensional, no entanto, devido à perspectiva gerada pela imagem, o modelo não os detecta.

Figura 14 – Exemplo de um objeto muito distante para detecção e rastreamento.

Outro caso interessante, demonstrando a boa capacidade do modelo de manter a trajetória apesar das oclusões, está na Figura 15. Nesta imagem, há uma pessoa completamente atrás da pessoa à frente da imagem. No segundo quadro, o modelo até perde a detecção dessa pessoa, mas no último, o ID é recuperado, como o mesmo que havia sido perdido, apesar do caso mostrar uma oclusão severa.

Figura 15 – Demonstração da robustez do modelo em um cenário de oclusão de objetos.

Em resumo, enquanto o modelo pode mostrar alguns pontos fracos, que podem e devem ser considerados para trabalhos futuros, ele é compensado em métricas mais altas (além de MOTA) e especialmente em alta FPS, o que pode permitir um melhor uso

prático deste modelo, o que não estava disponível com alguns dos modelos de melhor desempenho, pontuando menos de 10 FPS, significando que em um vídeo de 30 FPS, estaria perdendo um terço dos quadros para processamento.

6

Considerações finais

Este trabalho traz contribuições relevantes para a área de rastreamento de objetos. Essas inovações focaram em melhorar dois aspectos principais: tempo de inferência e estabilidade de treinamento. Em relação ao tempo de inferência, o uso dos transformers permitiu a abstração do contexto temporal excluindo quaisquer etapas adicionais para contextualização temporal. Isto é possível a partir do empilhamento das imagens de entrada na dimensão dos seus canais de cores. E a partir da codificação implementada, que fornece a esses canais a informação temporal, o modelo pode considerar tudo como uma só imagem. Além de não precisar de um decodificador para inferir os objetos e seus trajetos, bastando apenas algumas camadas a mais na saída para processar os mapas de atenção gerados.

Em termos de estabilidade de treinamento, o método TMP se provou uma opção com potencial para melhorar o treinamento de modelos MOT no geral. Tornando o processo mais estável e melhorando os resultados obtidos.

A versão anterior deste trabalho, o modelo OneTrack apresentado no ato da qualificação, gerou um artigo que foi submetido e aceito para publicação no Journal of Internet Services and Applications (JISA) do SBCUP. O conteúdo produzido a partir da pesquisa do modelo OneTrack-M gerou um outro artigo, que está também em processo de revisão, no entanto pela banca revisora do Journal of the Brazilian Computer Society (JBCS).

Finalmente como trabalhos futuros, deixo a sugestão de investigação da viabilidade do TMP para outros modelos MOT que tenham processamento similar, isto é,

realizam ambas etapas de MOT - detecção e rastreamento - de forma unificada. Também em outros campos onde existam modelos que também realizam tarefas que são relacionadas, mas que podem acabar interferindo no aprendizado mútuo do modelo.

Complementarmente, em específico para o OneTrack-M, avaliar outros modelos extratores no lugar do Vision Transformer pode ser um caminho promissor, todavia demanda um trabalho a mais em alterar o funcionamento dos mecanismos da arquitetura proposta, como pré e pós processamentos, de forma que mantenha suas características e permita o treinamento efetivo do modelo.

Em suma, este trabalho foi capaz de trazer resultados relevantes para a comunidade científica, em específico das áreas de visão computacional e, especificamente, do MOT. O ciclo de desenvolvimento para modelos de IA seguiu designações comuns em outros trabalhos da área, fornecendo uma base comparativa sólida, bem como a análise crítica dos erros que o modelo apresenta, tentando explicar de forma lógica quais possíveis causas dos mesmos, determina um ciclo fechado para o resultado apresentado.

REFERÊNCIAS

Nir Aharon, Roy Orfaig, and Ben-Zion Bobrovsky. Bot-sort: Robust associations multi-pedestrian tracking, 2022. 32, 38

Keni Bernardin and Rainer Stiefelhagen. Evaluating multiple object tracking performance: The clear mot metrics. *EURASIP Journal on Image and Video Processing*, 2008, 01 2008. doi: 10.1155/2008/246309. 65

Luca Bertinetto, Jack Valmadre, João F. Henriques, Andrea Vedaldi, and Philip H. S. Torr. Fully-convolutional siamese networks for object tracking, 2021. 22

Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468, 2016. doi: 10.1109/ICIP.2016.7533003. 16, 22, 31

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. 30

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers, 2020. 21, 30, 34, 39, 43, 45, 46

Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997a. 36

Richard Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997b. 26

N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1, 2005. doi: 10.1109/CVPR.2005.177. 20

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. 9, 15, 30, 33, 34, 39, 42, 44, 45, 49, 69, 71

Yunhao Du, Zhicheng Zhao, Yang Song, Yanyun Zhao, Fei Su, Tao Gong, and Hongying Meng. Strongsort: Make deepsort great again, 2023. 32, 38

Ross Girshick. Fast r-cnn, 2015. 20

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation, 2014. 20

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org. 23

M. Jaward, L. Mihaylova, N. Canagarajah, and D. Bull. Multiple object tracking using particle filters. In *2006 IEEE Aerospace Conference*, pages 8 pp.–, 2006. doi: 10.1109/AERO.2006.1655926. 31

Arne Hoffhues Jonathon Luiten. Trackeval. https://github.com/JonathonLuiten/TrackEval, 2020. 65

Yann LeCun, Y. Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–44, 05 2015. doi: 10.1038/nature14539. 9, 23, 24

Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015. 43

Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. *SSD: Single Shot MultiBox Detector*, page 21–37.

Springer International Publishing, 2016. ISBN 9783319464480. doi: 10.1007/978-3-319-46448-0_2. URL http://dx.doi.org/10.1007/978-3-319-46448-0_2. 21

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows, 2021a. 33

Ze Liu, Jia Ning, Yue Cao, Yixuan Wei, Zheng Zhang, Stephen Lin, and Han Hu. Video swin transformer, 2021b. 15, 33

Jonathon Luiten, Aljosa Osep, Patrick Dendorfer, Philip Torr, Andreas Geiger, Laura Leal-Taixé, and Bastian Leibe. Hota: A higher order metric for evaluating multi-object tracking. *International Journal of Computer Vision*, 129(2):548–578, October 2020. ISSN 1573-1405. doi: 10.1007/s11263-020-01375-2. URL http://dx.doi.org/10.1007/s11263-020-01375-2. 65

Tim Meinhardt, Alexander Kirillov, Laura Leal-Taixe, and Christoph Feichtenhofer. Trackformer: Multi-object tracking with transformers, 2022. 16, 32, 35, 38, 52

Anton Milan, Laura Leal-Taixe, Ian Reid, Stefan Roth, and Konrad Schindler. Mot16: A benchmark for multi-object tracking, 2016. 11, 51, 64

Rana Mostafa, Hoda Baraka, and AbdElMoniem Bayoumi. Lmot: Efficient light-weight detection and tracking in crowds. *IEEE Access*, 10:83085–83095, 2022. doi: 10.1109/ACCESS.2022.3197157. 33, 36, 38

Abdulmajid Murad and Jae-Young Pyun. Deep recurrent neural networks for human activity recognition. *Sensors*, 17:2556, 11 2017. doi: 10.3390/s17112556. 25, 45, 52

Hyeonseob Nam and Bohyung Han. Learning multi-domain convolutional neural networks for visual tracking, 2016. 22

Thao Nguyen, Eun-Ae Park, Jiho Han, Dong-Chul Park, and Soo-Young Min. Object detection using scale invariant feature transform. In Jeng-Shyang Pan, Pavel Krömer, and Václav Snášel, editors, *Genetic and Evolutionary Computing*, pages 65–72, Cham, 2014. Springer International Publishing. 20

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2016. 21

Peize Sun, Jinkun Cao, Yi Jiang, Rufeng Zhang, Enze Xie, Zehuan Yuan, Changhu Wang, and Ping Luo. Transtrack: Multiple object tracking with transformer, 2021. 16, 35, 38, 52

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. 29

Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors, 2022. 43, 57

Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric, 2017. 15, 22, 32, 38

Xie Xing, Yang Yongjie, and Xinming Huang. Real-time object tracking based on optical flow. In *2021 International Conference on Computer, Control and Robotics (ICCCR)*, pages 315–318, 2021. doi: 10.1109/ICCCR49711.2021.9349376. 22, 31

Fangao Zeng, Bin Dong, Yuang Zhang, Tiancai Wang, Xiangyu Zhang, and Yichen Wei. Motr: End-to-end multiple-object tracking with transformer, 2022. 16, 22, 32, 35, 38, 39, 52, 53

Yifu Zhang, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. Fairmot: On the fairness of detection and re-identification in multiple object tracking. *International Journal of Computer Vision*, 129:3069–3087, 2021. 36, 37, 38, 59, 64

Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Fucheng Weng, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. Bytetrack: Multi-object tracking by associating every detection box, 2022. 32, 38

Yu Zhang and Qiang Yang. A survey on multi-task learning. *arXiv preprint arXiv:1707.08114*, 2017. 37

Yuang Zhang, Tiancai Wang, and Xiangyu Zhang. Motrv2: Bootstrapping end-to-end multi-object tracking by pretrained object detectors, 2023. 38, 45, 52, 53

Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. In *arXiv preprint arXiv:1904.07850*, 2019. 54

Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Tracking objects as points. *ECCV*, 2020. 33, 38, 39, 53, 54, 55, 60