UNIVERSIDADE
LUSÓFONA

IEquus- Raio-X

Trabalho Final de curso

2ª Entrega Intercalar

Matilde Rodrigues, a22306236, CoMA

Orientador: Daniel Fernandes

Co-orientador: João Carvalho

Departamento de Engenharia Informática e Sistemas de Informação

Universidade Lusófona, Centro Universitário de Lisboa

11/04/2026

Direitos de cópia

Resumo

Com os avanços tecnológicos na área da medicina veterinária, emergiu a oportunidade de desenvolver aplicações móveis destinadas a otimizar o processo de monitorização de animais. No seguimento do desenvolvimento da aplicação iEquus num Trabalho Final de Curso anterior, identificou-se a necessidade de integrar uma funcionalidade que, a partir de uma imagem de um membro de um cavalo, fosse capaz de entregar a correspondente imagem radiográfica do mesmo membro, mantendo o mesmo ângulo de captação. O presente trabalho propõe uma *pipeline* de visão computacional para a classificação de imagens de membros de cavalos, recorrendo ao *fine-tuning* de modelos de *deep learning* pré-treinados disponíveis nas bibliotecas PyTorch, timm e Transformers. Foram testadas 15 arquiteturas, e a seleção do modelo mais adequado baseou-se nas métricas de *accuracy*, *precision*, *recall*, *F1-score* e a matriz de confusão.

Palavras-chave: tecnologia; saúde animal; deep learning; saúde equina; fine tuning; PyTorch.

Abstract

With technological advances in veterinary medicine, the opportunity has emerged to develop mobile applications aimed at optimizing the process of monitoring animals. Following the development of the iEquus application in a previous Final Course Project, the need was identified to integrate a feature that, from an image of a horse's limb, would be capable of producing the corresponding radiographic image of the same limb, maintaining the same capture angle. The present work proposes a computer vision pipeline for classifying images of horse limbs, using fine-tuning of pre-trained deep learning models available in the PyTorch, timm and Transformers libraries. Fifteen architectures were tested, and the selection of the most suitable model was based on the metrics of accuracy, precision, recall, F1-score, and the confusion matrix.

Keywords: technology; animal health; deep learning; equine health; fine-tuning; PyTorch.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Equações

Lista de Siglas

CNN	Convolutional Neural Network
ResNet	Residual Network
ViT	Vision Transformer
MHSA	Multi-Head Self-Attention
SE	Squeeze-and-Excitation
AI	Artificial Intelligence
Timm	Pytorch Image Models

1 Introdução

Com o avanço contínuo das tecnologias digitais, a medicina veterinária tem vindo a adotar abordagens inovadoras que visam otimizar os processos de ensino e aprendizagem. Entre essas abordagens destaca-se a integração de ferramentas baseadas em *Deep Learning*, cuja aplicação no contexto académico tem demonstrado resultados promissores em termos de precisão, eficiência e acessibilidade da informação.

Neste âmbito, o desenvolvimento da presente solução surge como uma resposta a uma necessidade real de apoio ao ensino da anatomia equina, permitindo aos estudantes de Medicina Veterinária aceder de forma rápida, intuitiva e rigorosa a informações sobre a estrutura de elevada complexidade, o esqueleto do cavalo. Esta iniciativa contribui, assim, para um processo de aprendizagem mais interativo, dinâmico e aprofundado, representando um avanço significativo na modernização dos métodos pedagógicos aplicados à formação veterinária.

1.1 Enquadramento

A aplicação de técnicas de *deep learning*, em particular *Convolutional Neural Network* (CNN), tem ganho destaque crescente na medicina veterinária nos últimos anos. O *deep learning* refere-se a um conjunto de métodos baseados em redes neuronais profundas capazes de extrair automaticamente padrões complexos a partir de dados, sendo especialmente eficaz em tarefas de visão computacional. As CNNs, por sua vez, são arquiteturas projetadas para análise de imagens, utilizando camadas convolucionais que identificam características relevantes de forma hierárquica. Diversas revisões sistemáticas recentes demonstram a expansão destas técnicas em contexto veterinário, sobretudo em tarefas como classificação de radiografias, segmentação de estruturas anatómicas e deteção automática de patologias [1].

Um exemplo concreto da eficácia destes modelos é a aplicação Pet-X [2], que utiliza uma arquitetura de CNN para analisar imagens de raio-X da caixa torácica de cães e gatos, permitindo a identificação de alterações pulmonares com precisão clínica. Este tipo de solução evidencia o potencial das CNNs no apoio ao diagnóstico veterinário.

Neste enquadramento, o presente trabalho propõe uma abordagem complementar, aplicando modelos de *deep learning* em imagens de membros de equinos com o objetivo de mais tarde estabelecer correspondência com radiografias do mesmo membro.

1.2 Motivação e Identificação do Problema

O presente trabalho surge do problema identificado entre os alunos do curso de Medicina Veterinária na identificação das estruturas ósseas equinas, particularmente durante a sua experiência prática no Hospital Veterinário de Equinos de Santo Estêvão, pertencente à Universidade Lusófona. Neste contexto, os alunos participam ativamente nos cuidados prestados aos equinos, o que requer um conhecimento aprofundado da anatomia e fisiologia destes animais.

Reconhecendo a importância desta etapa formativa, a instituição demonstra interesse em promover a melhoria contínua da qualidade do ensino e da aprendizagem dos seus estudantes. Assim, o presente projeto visa responder a essa necessidade através da integração de uma nova funcionalidade (*feature*) numa ferramenta previamente desenvolvida, com o objetivo de facilitar a compreensão de estruturas anatómicas complexas e, consequentemente, contribuir para uma formação mais sólida e eficaz.

1.3 Objetivos

À semelhança do TFC anterior a este [3], o principal objetivo deste TFC é oferecer uma solução aos estudantes de medicina veterinária que os auxilie durante sua prática clínica, facilitando o acompanhamento da saúde equina. Para tal será desenvolvido uma *feature* na aplicação criada no TFC mencionado, que consiga identificar a imagem do membro de um cavalo e retornar uma imagem de raio-x do mesmo membro na mesma posição.

1.4 Estrutura do Documento

Este documento encontra-se organizado pelos seguintes capítulos:

Capítulo 1 - Introdução: apresenta uma contextualização do tema do Trabalho Final de Curso, enquadrando-o nas áreas de estudo relevantes e definindo os objetivos a atingir.

Capítulo 2 - Pertinência e Viabilidade: discute a pertinência do projeto e analisa a sua viabilidade social, económica e técnica.

Capítulo 3 - Conceitos Fundamentais: expõe e explica os principais conceitos utilizados ao longo do trabalho, incluindo métricas, ferramentas e algoritmos essenciais para o desenvolvimento da solução proposta.

Capítulo 4 - Estado da Arte: descreve soluções existentes relacionadas com o problema em estudo e destaca as diferenças e inovações que distinguem a solução aqui apresentada.

Capítulo 5 - Solução Proposta: detalha a abordagem desenvolvida para resolver o problema identificado, apresentando a metodologia adotada, a fundamentação das escolhas efetuadas e a comparação entre os modelos de *deep learning* considerados.

Capítulo 6 - Método e Planeamento: explica o planeamento inicial do projeto, descreve o modo como o trabalho foi conduzido e inclui uma reflexão crítica sobre o processo de planeamento.

Capítulo 7 – Resultados e Discussão: explica o processo de obtenção dos dados, bem como compara e analisa os melhores modelos em estudo.

Capítulo 8 – Implementação: indica a infraestrutura na qual o trabalho foi desenvolvido e descreve, passo a passo, a implementação do mesmo.

Capítulo 9 - Conclusão: apresenta uma síntese dos principais resultados alcançados nesta primeira fase do projeto.

2 Pertinência e Viabilidade

2.1 Pertinência

A pertinência deste TFC assenta no problema que o Doutor José Prazeres, docente e diretor do Hospital Veterinário de Equinos de Santo Estêvão, identificou entre os alunos do curso de Medicina Veterinária, que estes tinham recorrentemente problemas na identificação óssea equina, no que o Doutor sugeriu uma funcionalidade de dicionário de bolso para ajudar os alunos neste problema. A formação anatómica nesta área exige um conhecimento detalhado dos múltiplos componentes do esqueleto do cavalo, o que, tradicionalmente, implica o recurso a modelos físicos, materiais didáticos limitados e imagens bidimensionais.

O presente projeto visa resolver essas limitações através do desenvolvimento de uma ferramenta digital interativa capaz de apoiar o processo de aprendizagem, tornando-o mais dinâmico, acessível e eficiente. Ao permitir a visualização automatizada de imagens radiografias a partir de um *input* de uma fotografia convencional, esta aplicação contribui para a consolidação do conhecimento anatómico de forma mais intuitiva e prática.

Posteriormente quando esta *feature* estiver implementada na aplicação móvel pode ser criado um inquérito que avalie o grau de aceitação dos utilizadores ao utilizarem a *feature* disponibilizada.

2.2 Viabilidade

A viabilidade deste Trabalho Final de Curso assenta em três dimensões principais: técnica, económica e social, as quais em conjunto demonstram o potencial de continuidade e aplicabilidade da solução proposta para além do âmbito académico.

2.2.1 Viabilidade social

A viabilidade social do projeto é evidenciada pelo interesse e aceitação por parte do público alvo, os estudantes e docentes do curso de Medicina Veterinária, e por instituições associadas ao ensino e prática veterinária. A nova funcionalidade da aplicação iEquus representa um contributo direto para a melhoria do processo de ensino aprendizagem, promovendo o uso de tecnologias digitais no estudo anatómico.

2.2.2 Viabilidade económica

A viabilidade económica deste Trabalho Final de Curso encontra-se assegurada pelo facto de estar integrado num projeto previamente aprovado. Dado que a nova funcionalidade proposta não implica custos adicionais significativos, recorrendo maioritariamente a ferramentas e infraestruturas já existentes, a sua implementação constitui uma mais-valia. Além de apresentar um impacto financeiro reduzido, contribuirá diretamente para o enriquecimento da formação académica dos estudantes da Universidade Lusófona, reforçando o valor pedagógico da aplicação.

2.2.3 Viabilidade técnica

A viabilidade técnica é garantida pela utilização de tecnologias amplamente reconhecidas e consolidadas na área do *deep learning*. O modelo final a ser integrado na aplicação será baseado em arquiteturas disponibilizadas pela plataforma PyTorch [4], um ecossistema *open-source* especializado no desenvolvimento de redes neuronais, incluindo modelos do tipo *Convolutional Neural Networks*. Estas ferramentas oferecem um conjunto robusto de funcionalidades que asseguram a estabilidade, eficiência e escalabilidade necessárias para a incorporação da solução no sistema já existente.

3 Conceitos Fundamentais

Neste capítulo procura-se expor e explicar principalmente os conceitos utilizados ao longo deste trabalho, incluindo os modelos e algoritmos, as ferramentas e as métricas utilizadas para testar a eficiência de cada modelo, tudo para o desenvolvimento da solução proposta.

3.1 Modelos e Algoritmos Relevantes

Durante o desenvolvimento deste Trabalho Final de Curso foram analisadas e comparadas 15 arquiteturas de visão computacional: ResNet18 e 50, ViT, MobilenetV3 small, googlenet, dino-vit, vgg11, ConvneXt tiny, small e base, EfficientnetV2 -s e -m, densenet121, se-resnext50, nasnet large. Em seguida serão explicados abaixo os 5 algoritmos mais relevantes para este estudo, com base no melhor *f1-score* médio de cada arquitetura.

3.1.1 ResNet

Pioneiro na introdução de blocos residuais, o modelo ResNet [5] é uma CNN amplamente utilizada na classificação de imagens. Esta arquitetura destaca-se pelo uso de *skip connections*, que transmitem o mapa residual dentro de cada bloco, facilitando o fluxo do gradiente. A sua principal vantagem é mitigar o problema do *vanishing/exploding gradient*, embora possa apresentar tendência para *overfitting* em conjuntos de dados pequenos.

3.1.2 ViT

Ao contrário de uma CNN, o modelo ViT (*Vision Transformer*) [6] não utiliza convoluções para processar imagens. Em vez disso, divide cada imagem em patches, projeta-os em *embeddings* lineares e aplica mecanismos de *self-attention* para aprender dependências globais entre todos os patches. A sua principal vantagem é a capacidade de capturar o contexto global da imagem, embora seja computacionalmente exigente e normalmente necessite de grandes volumes de dados para treino.

3.1.3 GoogLenet

A GoogLeNet [7] é uma arquitetura CNN desenvolvida pela Google com o objetivo de classificar imagens de forma eficiente em termos de tempo e desempenho. Utiliza convoluções 1 X 1 para reduzir a dimensionalidade e o número de parâmetros, bem como *Global Average Pooling* para processar os mapas de características. Além disso, utiliza convoluções paralelas (módulos *Inception*) que permitem à rede capturar características em múltiplas escalas de forma eficaz.

3.1.4 Dino-vit

Desenvolvido pela Meta AI, o DINO-ViT [8] é um modelo supervisionado que utiliza a arquitetura ViT como *backbone*. Recorre a duas redes distintas, *student* e *teacher*, onde a

primeira tenta prever o output da segunda, e a segunda é atualizada via *self-distillation* a partir da *student*. Embora as arquiteturas sejam idênticas, os *inputs* variam através de diferentes transformações. A sua principal vantagem é que por ser um modelo auto-supervisionado este modelo tem uma autocorreção em tempo real, tendendo a ter uma performance superior.

3.1.5 VGG11

Composta por 11 camadas com parâmetros treináveis, a arquitetura VGG-11 [9] foi proposta pelo *Visual Geometry Group* da Universidade de Oxford como uma CNN capaz de aprender características complexas através da utilização de blocos de camadas convolucionais com filtros de pequena dimensão (3x3). Após cada camada convolucional é aplicada a função de ativação ReLU, e, ao longo da arquitetura, são utilizadas camadas de *max-pooling* (2x2) para a redução da dimensionalidade espacial. A rede termina com uma camada *softmax* para a classificação final. A sua principal vantagem é que este tende a ter bons resultados em *datasets* menores, contudo é um modelo pesado e com muitos parâmetros.

3.2 Tecnologias e Ferramentas Utilizadas

3.2.1 Linguagem de programação:

Python é uma linguagem de programação de alto nível criada em 1989 que até hoje é amplamente utilizada em diversas áreas relacionadas com a computação.

Foi utilizada neste projeto a linguagem Python, por se tratar de uma linguagem consolidada na área de *Deep Learning*, e pelo vasto leque de bibliotecas e *frameworks* que permitem integração com esta linguagem, destacando-se o PyTorch, a principal ferramenta para a criação deste projeto.

3.2.2 Bibliotecas/Frameworks:

3.2.2.1 PyTorch*:*

"O PyTorch é uma *framework* de aprendizagem profunda de código aberto, criada para ser flexível e modular para investigação, oferecendo ao mesmo tempo a estabilidade e o suporte necessários para a implementação em produção." [10]

Na necessidade da escolha de uma *framework* para a nossa solução, existem duas soluções possíveis no mercado e a questão seria, qual das duas seria melhor para o que se pretende? De acordo com a pesquisa "The State of Machine Learning Frameworks in 2019" PyTorch tem vindo a ser mais popular no mercado pela sua maior facilidade de aprendizagem, simplicidade e customização de modelos, tudo integrado com Python. [11]

3.2.2.2 Sklearn:

Sklearn é uma biblioteca de *machine learning* que disponibiliza um conjunto abrangente de algoritmos e métricas de avaliação. Foi utilizado pois havia necessidade de dividir a *data* entre

o conjunto de teste e treino como também calcular as métricas para cada arquitetura, para observar os seus desempenhos.

3.2.2.3 Matplotlib:

Matplotlib é uma biblioteca de visualização em Python. O módulo pyplot permite criar gráficos de forma simples, incluindo gráficos de linhas, barras, dispersão e matrizes. Foi utilizado para criar uma visualização da matriz de confusão de cada arquitetura.

3.2.2.4 Transformers – Hugging face

A biblioteca Transformers fornece modelos pré-treinados para processamento de linguagem natural, visão computacional e modelos multimodais. Esta biblioteca foi utilizada para obter o modelo pré treinado Dino ViT.

3.2.2.5 Pytorch Image Models (timm) – Hugging face

A biblioteca timm é dedicada a modelos de visão computacional, disponibilizando diversos modelos de redes neuronais convolucionais (CNN) que não se encontram incluídos no PyTorch. Esta biblioteca foi utilizada para a obtenção e análise de modelos pré-treinados.

3.3 Métricas de avaliação

Para avaliar o desempenho de qualquer modelo de *machine learning*, é essencial recorrer a métricas que permitam quantificar a sua capacidade de generalização e a precisão das suas previsões. No contexto deste Trabalho Final de Curso, foram selecionadas quatro métricas amplamente utilizadas na literatura científica: *accuracy*, *precision*, *recall*, *F1-score* e matriz de confusão. Estas métricas possibilitam uma análise abrangente da qualidade do modelo, permitindo identificar tanto o número de classificações corretas como a sua consistência em diferentes classes de dados.

Temos que:

- TP - Verdadeiros positivos. Dados que foram classificados corretamente com a sua *label*.
- TN - Verdadeiros Negativos. Dados que pertencem a outra *label* e que o modelo classificou corretamente como não pertencente à *label* em estudo.
- FP - Falsos positivos. Dados de uma outra *label*, mas que o modelo classificou sendo da *label* em estudo.
- FN - Falsos negativos. Dados que são da *label* em estudo, mas que o modelo classificou incorretamente como pertencendo a outra *label*.

3.3.1 Accuracy

A *accuracy*, representada na Equação 1, é uma métrica que calcula a margem de acerto do modelo, tanto positivas quanto negativas, em relação ao total de previsões realizadas. Trata-se de uma métrica global que indica o grau de correspondência entre classificações produzidas pelo modelo e as *labels* reais dos dados.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Equação 1: Accuracy

3.3.2 Precision

A *precision*, representada na Equação 2, tem como principal objetivo calcular a percentagem de valores que foram classificados corretamente comparado a todos os valores que foram classificados com a *label* em questão (tanto de forma certa ou errada).

$$Precision = \frac{TP}{TP + FP}$$

Equação 2: Precision

3.3.3 Recall

A *recall*, representada na Equação 3, nos indica a percentagem de previsões corretas sobre a totalidade de valores reais da *label*.

$$Recall = \frac{TP}{TP + FN}$$

Equação 3: Recall

3.3.4 F1-score

A *F1-score*, tal como mostra a Equação 4, representa a média harmónica entre a *precision* e a *recall*, proporcionando uma métrica equilibrada quando é necessário ponderar o impacto de falsos positivos e falsos negativos. É particularmente útil em conjuntos de dados não balanceados, onde a *accuracy* isoladamente pode dar interpretação incorreta dos resultados.

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

Equação 4: F1-score

3.3.5 Matriz de confusão

A matriz de confusão, tal como representada na Figura 1, é uma tabela que avalia o desempenho de um modelo de classificação através da comparação entre os resultados previstos pelo modelo com os valores reais dos dados.

		Predito
Real	**Positivo**	**Negativo**
Positivo	TP	FN
Negativo	FP	TN

Figura 1: Matriz de Confusão

4 Estado de Arte

Neste capítulo apresentam-se e analisam-se as principais soluções existentes relacionadas com o problema em estudo, evidenciando-se as suas limitações e contributos. Adicionalmente, procede-se à contextualização da solução proposta neste TFC, destacando os aspetos que a diferenciam das abordagens já existentes.

4.1 Estado de Arte – Processamento de imagem

O desenvolvimento de arquiteturas de processamento de imagens tem registado um crescimento significativo nos últimos anos com a popularização da Inteligência artificial, motivando a criação de diferentes modelos que utilizam de diferentes técnicas na tentativa de alcançar melhores resultados.

Uma investigação sobre os diferentes modelos permitiu identificar vários estudos sobre o tópico, nomeadamente o estudo "Benchmark Analysis of Representative Deep Neural Network Architectures" [12], a qual se propões o teste de diferentes modelos em diferentes máquinas de forma a determinar a performance de cada modelo sem o fator de capacidade de máquina a influenciar os resultados.

Para este estado de arte exploraremos os três melhores modelos de acordo com a melhor *accuracy* de forma a comparar e identificar os elementos diferenciadores na qual se fazem destacar o modelo escolhido na resolução deste trabalho.

4.1.1 MobileNet-v2

A arquitetura MobileNet-v2 é uma rede convolucional projetada especificamente para dispositivos móveis, combinando eficiência computacional com desempenho elevado. Trata-se de uma versão aprimorada do MobileNet original, incorporando blocos residuais invertidos (*inverted residuals*), que conectam camadas de diferentes profundidades, permitindo um fluxo de informações mais eficaz ao longo da rede. Além disso, utiliza *bottlenecks* lineares, cuja função é minimizar a perda de informação durante as transformações, garantindo maior precisão e velocidade com baixo custo computacional.

Tal como no modelo original, o MobileNet-v2 emprega convoluções separáveis em profundidade (*depthwise separable convolutions*) para reduzir significativamente o número de parâmetros. O processo envolve duas etapas: a convolução *depthwise*, que aplica um filtro independente por canal de entrada, e a convolução *pointwise* (1×1), responsável pela combinação linear dos canais, construindo novas representações de características.

Estruturalmente, a rede inicia-se com uma camada convolucional inicial, seguida por múltiplos blocos invertidos com *bottlenecks* lineares, culminando em uma camada de global *average pooling* e uma camada totalmente conectada com ativação *softmax* para classificação.

Esta arquitetura destaca-se principalmente pela sua eficiência e desempenho, conseguindo reduzir drasticamente o número de parâmetros e o custo computacional sem comprometer a precisão. No estudo citado anteriormente [12], o modelo apresentou uma

accuracy Top-1 de 71,81%, reforçando seu status como uma das redes mais eficazes para aplicações móveis.

4.1.2 SE-ResNeXt-50

De forma diferente de uma arquitetura ResNet tradicional, a arquitetura SE-ResNeXt-50 baseia-se na variante ResNeXt, incorporando blocos de *Squeeze-and-Excitation* (SE) aos blocos convolucionais, permitindo uma recalibração dinâmica da importância dos canais. O processo de *Squeeze* reduz cada mapa de características a um único valor por canal, gerando um vetor que representa a importância global de cada canal. Em seguida, o processo de *Excitation* utiliza este vetor para gerar pesos de atenção para cada canal, multiplicando-os pelos mapas originais, reforçando os canais mais relevantes e atenuando os menos importantes.

Tal como na arquitetura ResNeXt, o modelo mantém os blocos residuais com *skip connections*, garantindo a propagação eficiente de gradientes em redes profundas. Além disso, utiliza convoluções agrupadas (*grouped convolutions*) com cardinalidade, em que cada bloco divide o mapeamento de características em múltiplos grupos independentes, que são posteriormente combinados, aumentando a expressividade do modelo sem elevar significativamente o número de parâmetros.

A nível estrutural, o modelo inicia-se com uma camada convolucional inicial, seguida por múltiplos blocos SE-ResNeXt, e termina com uma camada de global *average pooling* seguida de uma camada totalmente conectada com função de ativação *softmax*, responsável pela classificação.

Este modelo distingue-se não apenas pela integração das unidades *Squeeze-and-Excitation*, que permitem recalibrar os canais de forma adaptativa, mas também pelo seu elevado desempenho em tarefas de classificação, destacando-se pelo Top-1 *accuracy* superior em comparação a arquiteturas tradicionais de dimensão semelhante, de acordo com o estudo mencionado anteriormente [12].

4.1.3 Nasnet-A-Large

De forma diferente de uma arquitetura CNN tradicional, a arquitetura NASNet-A Large foi concebida automaticamente através do método *Neural Architecture Search* (NAS), um procedimento que visa otimizar e avaliar, de forma sistemática, diferentes combinações de operações convolucionais, com o objetivo de identificar a configuração arquitetural mais eficiente para uma determinada tarefa. A variante *Large* corresponde à versão de maior dimensão e desempenho dentro da família NASNet.

Ao invés de um processo manual de construção camada a camada, o algoritmo de pesquisa identifica uma estrutura base composta por células convolucionais otimizadas, que são posteriormente repetidas ao longo da rede. Nesta arquitetura destacam-se dois tipos principais de células: a *Normal Cell*, responsável por preservar a dimensão espacial das representações intermédias, e a *Reduction Cell*, que reduz a resolução espacial enquanto aumenta o número de filtros, permitindo a extração de características progressivamente mais complexas.

Cada célula recebe como entrada os outputs das duas células anteriores através de ligações residuais (*skip connections*). No interior das células são combinadas diferentes operações, tais como convoluções separáveis 3×3 e 5×5, *average pooling*, *max pooling* e ligações identidade. A arquitetura recorre ainda a convoluções separáveis em profundidade (*depthwise separable convolutions*), nas quais a convolução tradicional é decomposta em duas etapas: uma convolução *depthwise*, que aplica um filtro independente por canal de entrada, e uma convolução *pointwise* (1×1), responsável pela combinação linear dos canais. Esta abordagem permite reduzir significativamente o número de parâmetros e o custo computacional, mantendo simultaneamente um elevado desempenho.

A nível estrutural, o modelo inicia-se com uma camada convolucional inicial, seguida por múltiplas repetições de células normais e de redução. Na fase final, aplica-se uma camada de global *average pooling*, seguida de uma camada totalmente ligada com função de ativação *softmax*, responsável pela classificação.

Este modelo destaca-se não apenas pelo facto de a sua arquitetura ter sido concebida autonomamente através de um processo de *Neural Architecture Search*, ao contrário das arquiteturas tradicionalmente projetadas manualmente, mas também por ter apresentado os valores mais elevados de *accuracy* Top-1 e Top-5 no estudo anteriormente referido [12].

4.2 Estado de Arte – Aplicações móveis

O desenvolvimento de aplicações móveis dedicadas ao ensino de anatomia em Medicina Veterinária tem registado um crescimento significativo nos últimos anos. As soluções atualmente disponíveis no mercado apresentam abordagens diversas, que vão desde aplicações baseadas em reconhecimento de imagem através de métodos de *deep learning*, até plataformas mais tradicionais que funcionam como glossários interativos, recorrendo a fotografias reais e descrições anatómicas detalhadas para fins educativos.

A análise de mercado realizada permitiu identificar várias destas soluções e comparar as suas funcionalidades, metodologias e objetivos pedagógicos. Esta revisão teve como propósito compreender o panorama atual das tecnologias aplicadas ao ensino da anatomia animal e, paralelamente, destacar os elementos diferenciadores que o presente trabalho pretende introduzir.

4.2.1 IMAIOS vet-Anatomy

A IMAIOS vet-Anatomy [13], representada na Figura 2, é uma aplicação para dispositivos móveis de Android e iOS, focada na aprendizagem da anatomia animal, focada não só na anatomia óssea equina, mas como também de outros animais, contendo também anatomia muscular destes. A sua funcionalidade de se comportar semelhante a um atlas portátil de fácil acesso se equivalem a ideia principal de que a *feature* I-Equus-Raio-x propõe, no entanto, a IMAIOS vet-Anatomy não tem a componente interativa de reconhecimento de foto na qual este projeto tem como *core* na sua funcionalidade.

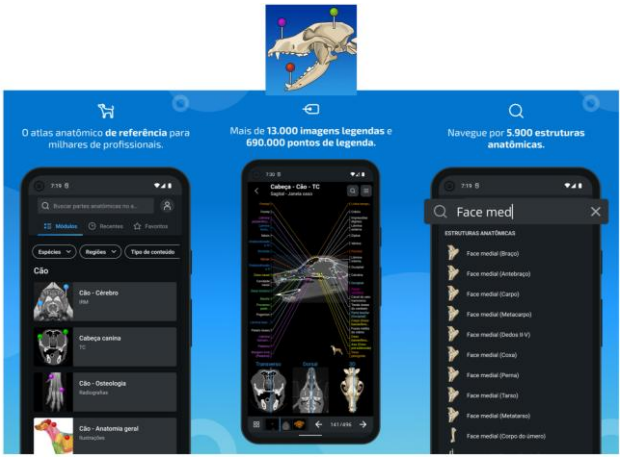

Figura 2: IMAIOS vet-Anatomy

4.2.2 Horse Scanner

A Horse Scanner [14], representada na Figura 3, é uma aplicação para dispositivos móveis de Android e iOS, que permite identificar a raça de um cavalo através do reconhecimento de uma fotografia do mesmo. A aplicação não contém nenhuma *feature* de glossário para anatomia equina, não sendo focada na aprendizagem anatómica, mas por utilizar método de processamento de imagem e reconhecimento da mesma. Assim, esta solução é relevante para o trabalho a desenvolver.

Figura 3: Horse Scanner

4.2.3 Bone Identifier – Osteo+

O Bone Identifier – Osteo+ [15], representado na Figura 4, é uma aplicação para dispositivos móveis iOS, que promete a identificação de ossos de vários animais através da leitura de imagem utilizando *features* de *Artificial Intelligence* (AI) para tal. Ao contrário da nossa proposta, esta aplicação utiliza de fotos de ossos reais para dar ao utilizador uma descrição do mesmo.

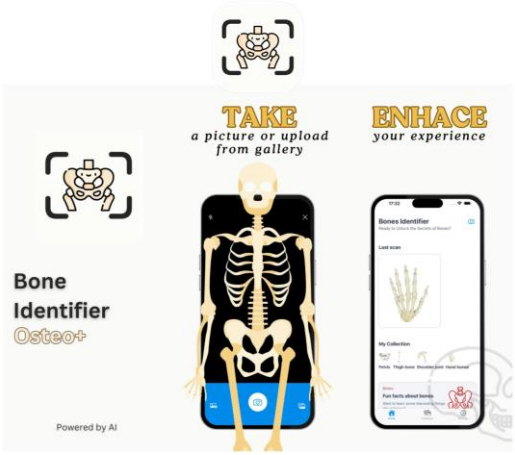

Figura 4: Bone Identifier - Osteo+

Como podemos ver na Tabela 1, cada abordagem distingue-se de maneira diferente com limitações diferentes. O IMAIOS vet-Anatomy por exemplo, embora contenha imagens de raio-X e o nome das estruturas ósseas representadas nestas, esta aplicação não contém o processamento de imagem, a *feature* de maior interesse para este trabalho. Por outro lado, a Horse Scanner contém a *feature* de identificação de imagem, no entanto esta aplicação limita-se ao reconhecimento da raça do cavalo. Por fim, o Bone Identifier – Osteo+ contém tanto a *feature* de reconhecimento de imagem quanto o de glossário, mas baseia-se em imagens de ossos, e sem a parte de identificação de raio-x a explicar a totalidade da estrutura em que o osso está inserido, tal como não parece ter grande popularidade, tendo uma avaliação baixa e muitas reclamações sobre a funcionalidade da mesma.

Solução:	Android	iOS	Raio-X	Processamento Imagem	Glossário	Equinos
IMAIOS vet-Anatomy	X	X	X		X	X
Horse Scanner	X	X		X		X
Bone Identifier – Osteo+		X		X	X	X
IEquus – Raio-X	X	X	X	X	X	X

Tabela 1: Comparação entre soluções

4.3 Proposta de inovação e mais-valias

Como pudemos ver anteriormente a nossa solução distingue-se das outras sendo a única que utiliza de modelos de visualização e processamento de imagem para o reconhecimento de imagens de membros equinos à tempo real, disponibilizando também um glossário sobre a estrutura óssea de equinos na mesma app, o que torna uma solução inovadora pois combina a

praticidade de tirar uma foto com a procura de informação específica, tornando a obtenção das respostas mais práticas e rápidas, automatizando a aprendizagem dos alunos.

5 Solução Proposta

Neste capítulo descreve-se de forma detalhada a abordagem desenvolvida para resolver o problema identificado. Apresenta-se a metodologia adotada, bem como a fundamentação das escolhas relativas às arquiteturas utilizadas. Por fim procede-se à comparação entre os modelos de *deep learning* considerados, evidenciando os critérios que orientaram a seleção da solução final.

5.1 Introdução

Numa primeira fase, a solução desenvolvida baseou-se no treino de diferentes arquiteturas de *deep learning* aplicadas à visão computacional, recorrendo a um conjunto de dados de imagens distinto do *dataset* final.

Nesta segunda fase, já com acesso ao *dataset* definitivo, deu-se continuidade ao estudo de diversos modelos de redes neuronais convolucionais (CNN), bem como à análise do seu desempenho sob diferentes técnicas de *data augmentation*. O objetivo consistiu na identificação do modelo mais adequado, a integrar posteriormente como uma *black box* na aplicação da IEquus.

Para aceder à solução desenvolvida, deverá dirigir-se ao repositório GIT https://github.com/DEISI-ULHT-TFC-2025-26/TFC-DEISI2166-I-Equus-Raio-x e ao vídeo do Youtube https://www.youtube.com/watch?v=Aga81RoAcM8.

5.2 Metodologia

A solução apresentada na Figura 5 consiste no desenvolvimento de uma *black box* que integra uma arquitetura de *deep learning* para visão computacional, implementada em PyTorch, destinada a posterior integração na aplicação móvel IEquus.

O modelo treinado recebe como entrada uma imagem de um membro de um cavalo e produz como saída uma *label* correspondente ao membro identificado e ao tipo de vista presente na fotografia. Com base nessa classificação, a aplicação consulta uma base de dados interna e seleciona a imagem radiográfica associada à *label* obtida, apresentando-a posteriormente ao utilizador.

Assim, o processo inicia-se com a submissão de uma imagem através da aplicação móvel, a qual é encaminhada para o modelo de *deep learning*. O algoritmo procede ao processamento e à categorização da imagem, gerando uma *label* que permite à aplicação recuperar, da base de dados de imagens de raio-X, a radiografia correspondente ao membro e à perspetiva identificados, devolvendo-a ao utilizador.

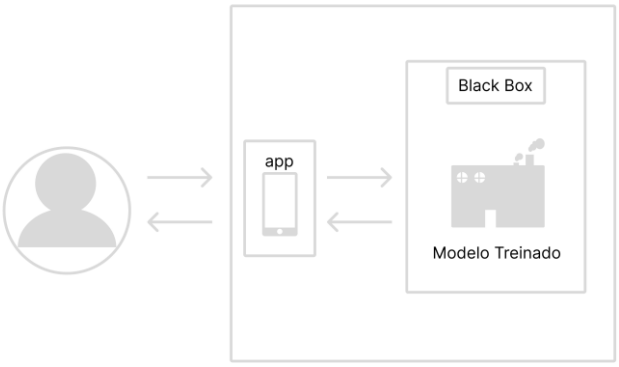

Figura 5: Solução Encontrada

Para o efeito, foi conduzida uma série de testes aos vários modelos apresentados no Capítulo 3.1, explorando diferentes configurações de parâmetros com o objetivo de identificar o melhor desempenho para cada arquitetura.

Foram testados diferentes métodos de processamento de imagens, utilizando *data augmentation*, tal como serão explorados diversos parâmetros, sendo estes *learning rate*, diferentes *optmizers* e *loss functions,* numa terceira fase.

Com o estudo dos diferentes parâmetros haverá a comparação de resultados que irá levar à escolha do modelo na qual será implementado na *blackbox* anteriormente mencionada.

5.3 Recolha de Dados

Numa primeira fase deste TFC o *dataset* com as imagens não foi disponibilizada logo foi utilizado outro *dataset* para o estudo dos modelos, contudo nesta segunda parte já foi possível a disponibilização e utilização dos dados para treino e teste.

Para este TFC as imagens foram recolhidas no Hospital Veterinário de Equinos de Santo Estêvão, utilizando dos equinos do estabelecimento como modelos para criar 8 pastas das diferentes classes com uma média de 21 imagens por pastas, que posteriormente viriam a se juntar para criar o *dataset* com 168 imagens.

5.4 Descrição dos Dados e Análise Exploratória dos Dados

Para esta base de dados, todos os dados recolhidos se resumem a imagens de formato jpg ou png com canais de cor sRGB e dimensão 2296X4080 com resolução de 72ppp, retiradas a partir de uma câmera do modelo Galaxy A54 5G.

As principais variáveis nos nossos dados são tanto o membro do equino a ser fotografado quanto o ângulo de visão da imagem em relação ao membro em questão, na qual até agora temos os membros anteriores direito (Ant_D) e esquerdo (Ant_E) de diferentes equinos, como também 4 tipos de vistas para cada. Temos a Vista Dorsomedial-palmarolateral

obliqua (D45°M-PaLO), Vista Dorsolateral-palmaromedial obliqua (D45°L-PaMO), Vista Dorsoproximal-palmarodistal obliqua (DP10°-PaDO), Vista Lateromedial (LM).

Com estas duas variáveis principais foram criadas 8 classes, para o membro direito foram chamadas de Ant_D_D45L_PaMO, Ant_D_D45M_PaLO, Ant_D_DP10_PaDO, ANT_D_LM, e para o membro esquerdo houve uma nomenclatura parecida, mas com substituição do D por E após o Ant.

5.5 Pré-processamento dos dados

A limpeza dos dados foi realizada através da comparação das imagens do *dataset* com imagens oficiais de um documento que apresentava cada imagem e a respetiva classe. Este procedimento permitiu identificar imagens mal rotuladas ou em falta, sendo realizada uma verificação imagem a imagem para todo o *dataset*. Durante este processo, surgiu uma imagem mal classificada que não estava presente no documento oficial, na qual foi retirada da base de dados.

Além disso, ao visualizar as imagens do *dataset*, constatou-se que estas estavam horizontalmente distribuídas. Assim, foi necessário proceder à sua rotação para que ficassem orientadas corretamente, melhorando a compreensão visual.

Para a divisão do *dataset* em treino e teste, optou-se por uma proporção de 80/20 respetivamente. Considerou-se a hipótese de criar um subconjunto de validação, mas, devido ao tamanho reduzido do *dataset*, decidiu-se focar apenas no treino e teste.

Para garantir uma análise consistente, aplicou-se normalização às imagens de treino e teste. Cada pixel foi normalizado nos canais RGB com médias de 0.485, 0.456 e 0.406 e desvios-padrão de 0.229, 0.224 e 0.225. Esta normalização permite que o algoritmo reconheça diferentes *features* de forma equilibrada, prevenindo que variações de luminosidade ou intensidade de cor introduzam viés na classificação.

As dimensões das imagens foram também uniformizadas através de *resize*, uma vez que apresentavam variação de tamanho, e os algoritmos requerem imagens de 224×224 pixels para treino.

Foram aplicadas várias técnicas de *data augmentation* nos dados de treino, com o objetivo de aumentar a diversidade das imagens apresentadas aos modelos, reduzir *overfitting* e melhorar o reconhecimento de *features*.

O *data augmentation* foi implementado utilizando o modelo do PyTorch, permitindo que o número de imagens armazenadas na memória não aumente. Em cada *epoch*, é aplicada uma transformação diferente a cada imagem, simulando a apresentação de novas imagens ao modelo sem aumentar efetivamente o tamanho do *dataset*.

	Real	*Augmentation*
Treino	130	130*n_epochs
Teste	38	38

Tabela 2: Nº de imagens normal X *Augmentation*

5.6 Modelos e Algoritmos Escolhidos

Para o treino dos modelos, foi necessário definir previamente um conjunto de parâmetros comuns. Foram selecionados 15 modelos pré-treinados, recorrendo a técnicas de *fine-tuning*, utilizando os pesos definidos por defeito, ou seja, aqueles que apresentaram melhor desempenho para cada arquitetura na versão atual do PyTorch (2.8.0+cpu).

Relativamente aos principais parâmetros, foi utilizado, em todos os modelos, um *learning rate* de $1e^{-3}$, o otimizador AdamW e a função de perda *Cross-Entropy*, uma vez que, de acordo com as práticas comuns na área, estas configurações são amplamente utilizadas. Foi ainda definida uma *batch size* de 16, tendo em conta a reduzida dimensão do conjunto de dados.

Numa terceira fase deste trabalho, serão exploradas diferentes configurações com o objetivo de otimizar o desempenho dos cinco melhores modelos.

Tal como foi realizado anteriormente no Subcapítulo 3.1, neste capítulo serão descritos apenas os cinco melhores modelos, de um total de 15.

5.6.1 ResNet

Sendo um modelo amplamente consolidado na literatura, a arquitetura ResNet [5] foi selecionada devido ao seu elevado desempenho, decorrente da utilização de *skip connection*, que permite mitigar o problema do desaparecimento do gradiente.

Optou-se pela variante ResNet-18, tendo em consideração a sua menor complexidade e número reduzido de parâmetros, o que a torna particularmente adequada para cenários com *datasets* de dimensão limitada, como é o caso.

5.6.2 ViT

A arquitetura ViT [6] foi selecionada pela sua capacidade de modelar dependências globais através de mecanismos de atenção, superando limitações inerentes às arquiteturas convolucionais tradicionais neste domínio. Esta abordagem revelou-se particularmente adequada para o tipo de dados utilizados.

Foi utilizado o modelo B32 pois da sua família este é o menor deles, o que torna adequado para o cenário de um *dataset* pequeno como o disponível.

5.6.3 GoogLeNet

A arquitetura GoogLeNet [7] foi escolhida devido à eficiência computacional alcançada pelo seu módulo *Inception*, que permite a extração de características em múltiplas escalas.

Adicionalmente, este modelo tem demonstrado um desempenho consistente mesmo em cenários com conjuntos de dados de dimensão reduzida, justificando a sua inclusão neste estudo.

5.6.4 Dino-ViT

O modelo DINO-ViT [8] foi considerado por incorporar uma abordagem de aprendizagem auto-supervisionada sobre a arquitetura ViT, previamente validada neste contexto.

Esta estratégia permite a extração de representações mais robustas sem dependência direta de grandes volumes de dados.

5.6.5 VGG

O modelo VGG [9] foi implementada devido à sua capacidade de extração de características, assente numa arquitetura simples e homogénea baseada em convoluções de pequena dimensão. Apesar da sua exigência computacional, este modelo continua a ser uma referência relevante como *baseline* em tarefas de visão computacional.

Foi escolhida a versão mais pequena disponível, a VGG11, uma vez que, apesar de ser a mais reduzida, já contém milhares de *features*, o que é suficiente para a dimensão do *dataset*.

5.7 Abrangência

Embora, numa fase inicial, a unidade curricular de Introdução à Inteligência Artificial tenha sido considerada relevante para a seleção do estudante responsável por este trabalho, verificou-se, ao longo do desenvolvimento do projeto, que a sua resolução estava mais diretamente relacionada com os conteúdos abordados na unidade curricular de Aprendizagem Automatizada II.

Na unidade curricular de Aprendizagem Automatizada II foram explorados de forma aprofundada conceitos de *deep learning* e várias arquiteturas para processamento de imagens, conhecimentos esses que se revelaram fundamentais e foram posteriormente aplicados na construção da solução apresentada.

6 Método e Planeamento

6.1 Planeamento inicial

Numa fase inicial do Trabalho Final de Curso, foi elaborado um gráfico de Gantt que representava o planeamento temporal das atividades. Contudo, verificaram-se algumas alterações a esse planeamento, conforme ilustrado na Figura 6. Ainda assim, este instrumento revelou-se particularmente útil, permitindo cumprir as principais etapas do projeto de forma organizada e sem atrasos significativos.

A fase inicial foi dedicada ao estudo das diversas arquiteturas disponibilizadas pela biblioteca PyTorch, com o objetivo de estabelecer uma base sólida para a implementação progressiva de diferentes modelos ao longo das semanas seguintes. Posteriormente, procedeu-se à seleção de um conjunto de dados adequado para o treino dessas arquiteturas, bem como à implementação e experimentação de novos modelos.

Numa segunda fase, foram disponibilizados dados reais, tendo-se procedido ao seu tratamento, de modo a possibilitar a sua utilização nos diferentes modelos, os quais foram testados recorrendo a várias métricas de avaliação.

Por fim, numa terceira fase, será realizada a avaliação dos modelos, incluindo a eventual realização de ajustamentos com vista à melhoria do seu desempenho. Na etapa final, o modelo com melhor desempenho será integrado na aplicação móvel da IEquus

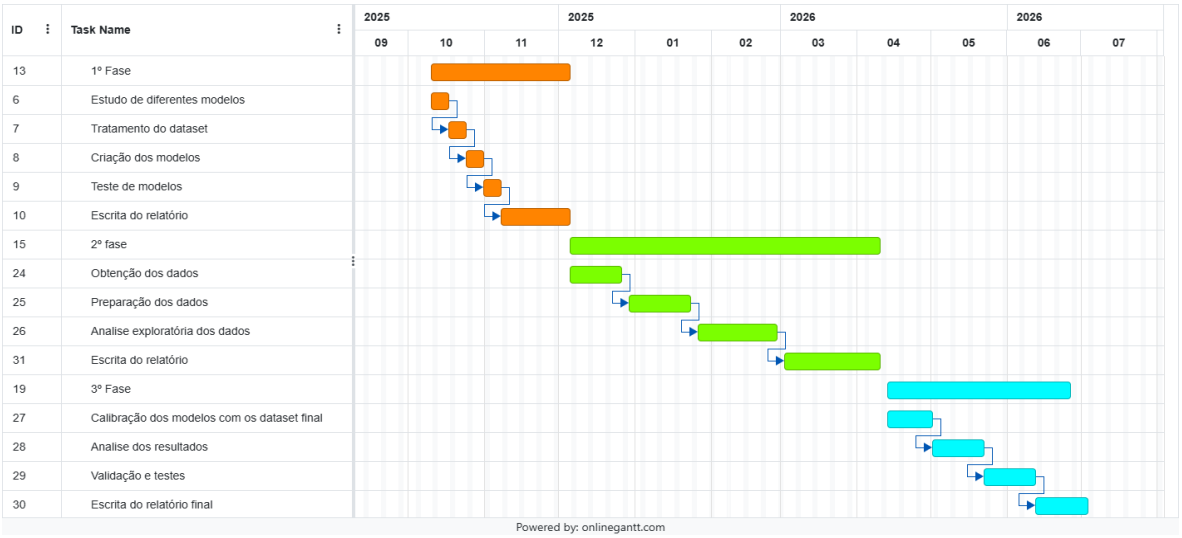

Figura 6: Gantt Chart

6.2 Análise Crítica ao Planeamento

Analisando de uma forma critica, este planeamento permitiu um fluxo de trabalho eficiente, a proposta semanal de novos desafios possíveis de cumprir no período de tempo dado, mais o crescente nível de complexidade construída gradualmente com base nos desafios feitos anteriormente, levou assim a um ritmo fluido da realização do trabalho e a possibilidade de o seguir como planeado.

7 Resultados e Discussão

7.1 Resultados das Análises

Nesta segunda fase os resultados foram obtidos através de uma análise exaustiva de 15 algoritmos de visão computacional com objetivo de encontrar a melhor combinação de métodos que maximizasse o *f1-score*, contudo, uma vez que muitos dos modelos estudados não obtiveram um bom desempenho, nesta secção apenas serão analisados os 5 melhores modelos.

No início desta análise foi criado um grupo de controlo com os dados de treino dos 15 algoritmos usando as imagens sem filtros, cada modelo treinado duas vezes em *seeds* diferentes (42 e 43) para saber os valores de f1-score na qual cada modelo tende a convergir na sua melhor época tal como mostrado na Tabela 3.

Modelos	**1º f1_macro**	**2º f1_macro**
ResNet18	0.404	0.402
ViT	0.309	0.370
GoogLeNet	0.409	0.390
Dino-ViT	0.430	0.442
VGG11	0.387	0.412

Tabela 3: Tabela de Controle

Após a obtenção dos valores de controlo, procedeu-se ao treino dos mesmos modelos, utilizando as mesmas *seeds*, mas aplicando seis técnicas distintas de data *augmentation*, com o objetivo de avaliar quais os métodos que maximizam o desempenho de cada modelo, visando posteriormente a criação de combinações de técnicas.

As seis técnicas consideradas foram:

- **Random Rotation**: transformação que aplica rotações aleatórias às imagens, dentro de um intervalo entre 0 e 10 graus.

- **ColorJitter**: transformação que introduz variações aleatórias no brilho (0.2), contraste (0.2), saturação (0.2) e matiz da imagem (0.05).

- **Random Resized Crop**: transformação que realiza cortes aleatórios na imagem, podendo também redimensionar a região selecionada (tamanho = 224, scale = (0.8, 1.0)).

- **Random Horizontal Flip**: transformação que aplica, de forma aleatória, uma inversão horizontal da imagem, com probabilidade de 50%.

- **Random Affine**: transformação que aplica, de forma aleatória, uma combinação de operações geométricas, incluindo rotação (10), translação (0.1, 0.1), escala (0.9, 1.1) e cisalhamento (5).

- **AutoAugment**: técnica de *data augmentation* que aplica uma política de transformações previamente otimizada através de aprendizagem por reforço num conjunto de dados de referência (ImageNet), consistindo numa combinação fixa de operações e respetivos parâmetros, utilizada para melhorar o desempenho do modelo.

Modelo	**Random Rotation**	**ColorJitter**	**Random Resize Crop**	**Random Horizontal Flip**	**Random Affine**	**Auto Augmentation**
ResNet18	0.268 0.284	0.405 0.397	0.386 0.383	0.384 0.373	0.324 0.334	0.450 0.406
ViT	0.438 0.417	0.356 0.318	0.323 0.282	0.326 0.348	0.405 0.395	0.404 0.396
GoogLe Net	0.278 0.276	0.417 0.395	0.269 0.286	0.354 0.408	0.259 0.256	0.387 0.340
Dino-ViT	0.469 0.460	0.443 0.431	0.486 0.457	0.335 0.341	0.478 0.497	0.460 0.479
VGG11	0.377 0.382	0.412 0.379	0.376 0.388	0.356 0.369	0.349 0.422	0.434 0.413

Tabela 4: F1 Score dos Modelos CNN sob Diferentes Transformações de Dados

De forma geral, os resultados indicam que a maioria dos modelos apresenta melhor desempenho quando sujeita a transformações geométricas, tais como *AutoAugmentation*, *Random Affine*, *Random Resized Crop* e *Random Rotation*. Com base nesta observação, foram definidos quatro novos métodos de *data augmentation*, resultantes da combinação das transformações previamente testadas, com o objetivo de maximizar o desempenho dos modelos.

O primeiro método desenvolvido integrou todas as transformações geométricas que demonstraram impacto positivo. No entanto, considerando que o *AutoAugmentation* já incorpora múltiplas estratégias de aumento de dados, a sua combinação com outros métodos poderia introduzir distorções excessivas nas imagens, comprometendo o processo de aprendizagem. Assim, foi concebido um segundo método semelhante ao primeiro, mas excluindo o *AutoAugmentation*.

Os restantes dois métodos foram definidos com base nos resultados obtidos anteriormente. Conforme evidenciado na Tabela 5, verificou-se que a combinação excessiva de

transformações não produzia melhorias adicionais. Consequentemente, optou-se por estratégias mais simples, com menor número de transformações.

O método 3 consistiu na combinação de *RandomResizedCrop*(*scale*=(0.9,1.0)) com *AutoAugmentation*, enquanto o método 4 combinou *RandomResizedCrop*(*scal*e=(0.9,1.0)) com *RandomAffine*. A escolha do *RandomResizedCrop* deve-se ao facto de constituir uma alternativa ao *resize* convencional, previamente utilizado, enquanto *AutoAugmentation* e *RandomAffine* foram selecionados por apresentarem, respetivamente, o melhor e o segundo melhor desempenho médio entre os modelos analisados.

Modelo	**Método 1**	**Método 2**	**Método 3**	**Método 4**
ResNet18	0.376 0.287	0.316 0.287	0.455 0.447	0.290 ---
ViT	0.318 0.312	0.335 0.347	0.393 0.360	0.369 0.377
GoogLeNet	0.355 0.320	0.280 0.286	0.285 ---	0.294 ---
Dino-ViT	0.458 0.392	0.472 0.385	0.473 0.501	0.462 0.429
VGG11	0.293 0.313	0.354 0.332	0.369 ---	0.331 ---

Tabela 5: Desempenho dos Modelos CNN com Combinações Personalizadas de Data Augmentation

Vale notar que na Tabela 5 os valores em falta se devem a uma decisão metodológica adotada durante o processo experimental. Modelos que apresentavam desempenho inferior ao controlo não foram novamente treinados nas fases subsequentes, com o objetivo de otimizar o tempo de execução associado ao treino dos métodos 3 e 4.

7.2 Comparação de Modelos e Abordagens

A análise da Tabela 4, em comparação com os dados de controlo da Tabela 3, permite observar que o método *Random Horizontal Flip* apresentou os piores resultados, conduzindo a uma diminuição do score na maioria dos modelos. Este comportamento sugere que, no contexto dos dados utilizados, a orientação das imagens constitui uma característica relevante para a tarefa de classificação. Tal hipótese é sustentada pela inspeção visual das imagens, onde se verifica que classes como Ant_D_D45L_PaMO e Ant_E_D45L_PaMO apresentam elevada semelhança estrutural, distinguindo-se essencialmente pela posição ou direção do membro do equino.

Adicionalmente, constata-se que o processo de *AutoAugmentation* foi o método mais consistente entre os modelos, promovendo melhorias significativas no valor de *f1-score* na generalidade dos casos.

Por outro lado, métodos como *Random Rotation*, *Random Resized Crop* e *Random Affine* evidenciaram melhorias em 2 dos 5 modelos analisados. Destaca-se ainda que a combinação de *Random Affine* com o modelo DINO-ViT produziu o melhor resultado global de *f1-score* na tabela, atingindo um valor de 0.497 numa segunda execução.

O método *ColorJitter* revelou utilidade apenas para um modelo, o GoogLeNet. Ainda assim, este foi o único modelo que beneficiou consistentemente deste método, alcançando um valor de 0.417 no primeiro teste.

Na Tabela 5 a análise dos novos resultados revelou uma diminuição significativa do desempenho global. Este comportamento sugere que, embora os modelos beneficiem de transformações geométricas, a sua sobreposição, particularmente quando combinada com operações de *cropping*, pode conduzir a uma perda excessiva de informação e contexto, prejudicando o processo de aprendizagem.

Contudo, de forma diferente dos demais modelos, o modelo ResNet18 apresentou uma melhoria significativa com a aplicação do método 3, indicando uma maior robustez à perda de informação e contexto em comparação com os restantes modelos.

A Tabela 6 apresenta a comparação de desempenho entre os modelos testados, considerando a métrica *f1-macro* nos dois melhores cenários de cada modelo, e o método de data *augmentation* associado.

Modelos	**1º f1_macro**	**2º f1_macro**	**Método**
Dino-ViT	0.478	0.497	Random Affine
ResNet18	0.455	0.447	Método 3
ViT	0.438	0.417	Random Rotation
VGG11	0.434	0.413	Auto Augmentation
GoogLeNet	0.417	0.395	ColorJitter

Tabela 6: Resultados de F1-score máximo obtidos por cada modelo

Observa-se que o modelo Dino-ViT alcançou o maior valor de f1 (0.497) com o método *Random Affine*, superando o segundo melhor modelo ResNet18 a 0.013 valores no 1º *f1-score* e 0.050 no 2º *f1-score*.

O modelo ViT, *backbone* do modelo Dino-ViT, ficou em terceiro lugar, demonstrando valores de *f1-score* de 0.438 e 0.417, inferiores ao Dino-ViT por 0.04 e 0.08 respetivamente. E o modelo ViT mostrou-se ligeiramente superior ao modelo VGG11 por 0.002 e 0.004 valores no primeiro e segundo *f1-score*.

O modelo GoogLeNet foi aquele com piores resultados, não chegando a meta de 0.4 valores no segundo *f1-score*.

Os restantes modelos não mostrados neste estudo tiveram na maior parte dos testes valores a baixo de 0.4, muitas vezes chegando penas a valores de 0.2, e apresentando valores mais instáveis, com grande disparidade de valores entre o 1º e 2º *f1-score*. Portanto foi escolhido não apresentar estes dados, contudo estes podem ser obtidos no github deste TFC.

7.3 Interpretação dos Resultados

A presença de dois modelos do tipo *Vision Transformer* na tabela sugere que o *dataset* pode possuir um elevado grau de informação contextual. Modelos como o *Vision Transformer* baseiam-se na aprendizagem de dependências globais entre diferentes regiões da imagem, utilizando mecanismos de *self-attention*. Para isso, incorporam informação posicional que permite distinguir a localização relativa de cada *patch*. Assim, o bom desempenho deste tipo de modelos pode indicar que a estrutura espacial e as relações entre diferentes partes da imagem são relevantes para a tarefa, sugerindo que o *dataset* apresenta características fortemente contextualizadas.

Podemos notar também que o modelo Dino-ViT obteve os melhores resultados comparado ao modelo VIT, que ficou em 3º lugar. Este resultado pode ser um forte indício que a natureza de autoaprendizado do Dino-ViT dá uma vantagem ao modelo, comparado aos restantes. Outro indício que a natureza de autoaprendizado do Dino-VIT melhora os resultados é que para todos os testes concretizados de métodos de auto *augmentation*, quase nenhum tinha qualquer impacto no modelo ViT, enquanto o modelo Dino-VIT tenha um aumento na performance em parte dos testes.

Devido aos baixos valores de *F1-score macro*, não é possível determinar com certeza se o *dataset* é intrinsecamente difícil de classificar, nem se o problema se deve à insuficiência de imagens disponíveis. De facto, os modelos utilizados foram previamente treinados com conjuntos de dados de grande dimensão, contendo milhares a milhões de imagens, ao contrário do *dataset* em estudo.

Ainda assim, a obtenção de novas imagens e o consequente treino dos modelos poderão constituir uma forma de esclarecer esta questão.

7.4 Limitações da Análise

A principal limitação que ocorreu neste trabalho foi a pequena quantidade de dados disponibilizados para o treino e teste dos nossos algoritmos, tal como a semelhança entre as imagens fornecidas. Isto é um problema pois uma vez que os modelos têm poucos dados de treino e os dados disponíveis são semelhantes, estes não têm como aprender padrões importantes de cada classe, nem aprendem a fazer generalização para novos dados, levando a *overfitting* em maior parte dos cassos pois estes “decoram” as imagens dadas a treino.

8 Implementação

8.1 Recursos e Infraestrutura

Para o desenvolvimento e execução desta solução, foram utilizados os seguintes recursos de hardware e software:

Hardware: Processador Intel® Core™ i7-1185H de 11.ª geração (2,50 GHz), duas GPUs NVIDIA T1200 Laptop GPU, GPU integrada Intel® UHD Graphics e 32 GB de memória RAM.

Software: O desenvolvimento foi realizado utilizando a IDE Visual Studio Code, com suporte Python 3.13. Foram instaladas bibliotecas específica, indicadas em 3.2.2.

Base de dados: O estudo utilizou o *dataset* indicado em 5.3.

Ambiente de execução: Os experimentos foram conduzidos em ambiente local, garantido compatibilidade com s versões das bibliotecas e sistema operativo Windows 11.

8.2 Processo de Implantação

Para aceder ao trabalho, é necessário dirigir-se ao repositório Git (https://github.com/DEISI-ULHT-TFC-2025-26/TFC-DEISI2166-I-Equus-Raio-x) e descarregar todos os arquivos do projeto.

Uma vez descarregados, antes de correr qualquer código, devem ser seguidos dois procedimentos.

Em primeiro lugar, a instalação dos pacotes e dependências. Caso o ambiente de desenvolvimento (IDE) não o faça automaticamente, a instalação deve ser efetuada através do ficheiro requirements.txt, utilizando o seguinte comando no terminal: pip install -r requirements.txt.

Em segundo lugar, a criação de uma conta no website Hugging Face, caso ainda não possua uma, bem como o pedido de acesso ao modelo DINO-ViT da Meta e a geração de um *token* de acesso.

É importante associar o *token* ao ambiente de desenvolvimento, caso ainda não o tenha feito, caso contrário, não será possível aceder ao modelo DINO-ViT.

Uma vez as dependências instaladas, o código já pode ser executado. Existem três documentos na qual pode interagir: analiseexploratoria.ipynb, TFC_codigo.ipynb e melhores_modelos.py.

Para executar o analiseexploratoria.ipynb tenha a certeza de que os dados se encontram em uma pasta denominada “Dados“. Para executar o ficheiro TFC_codigo.ipynb, será necessário correr o ficheiro melhores_modelos.py caso ainda não existam os ficheiros CSV necessários, bem como uma pasta denominada “dados_split”, criada a partir da pasta “Dados” no ficheiro analiseexploratoria.ipynb.

8.3 Testes de Validação e Desafios

Neste Trabalho Final de Curso procuramos principalmente fazer o estudo entre vários modelos de CNN de forma a encontrar um modelo ótimo capaz de identificar tanto as diferenças entre elas quanto os pequenos detalhes de cada classe de forma a classificar corretamente uma nova imagem uma vez o modelo exposto a esta.

De início houve a obtenção dos dados através de diferentes pastas organizadas pelo tipo de classe de imagens dentro destas, e o agrupamento de todas numa só para a criação do *dataset* principal. Para facilitar no momento de classificar cada imagem houve a mudança do nome de cada pasta para o nome das classes as que as imagens de cada pasta representavam.

Com os dados organizados houve a limpeza dos mesmos. Através de um documento oficial que continha as imagens e as classes a que estas pertenciam, houve uma análise pasta por pasta a comparar as fotografias com as imagens do documento oficial. Uma vez encontrado uma imagem mal classificada havia a mudança da imagem para a pasta da classe certa, e uma fez encontrada imagens que não existiam no documento oficial, houve a exclusão das mesmas do *dataset*.

Com os dados limpos foi possível o treino dos modelos. Através de um estudo prévio de modelos disponíveis no mercado, foi escolhido 15 algoritmos a serem treinados e testados com o nosso *database*. Foi escolhido modelos já pré treinados pois o *dataset* atual contem poucas imagens, então resolveu-se utilizar *fine tunning* com a adição de uma *layer* final de treino e classificação para todos os modelos.

Cada modelo foi treinado e testado três vezes, sendo a seleção dos melhores modelos realizada com base na média do *F1-score*. Numa fase inicial, optou-se por treinar as arquiteturas sem recurso a *data augmentation*, aplicando apenas um pré-processamento básico aos dados. Esta abordagem permitiu obter resultados de referência, utilizados como grupo de controlo para comparação com experiências posteriores envolvendo diferentes parametrizações.

Numa segunda fase, foram aplicadas várias técnicas de *data augmentation*, procedendo-se à realização de dois ciclos de treino para cada configuração, com o objetivo de avaliar o impacto de cada técnica no desempenho dos modelos. Com base nos resultados obtidos, foi posteriormente definida uma estratégia de aumento de dados que combina as técnicas que demonstraram melhor desempenho.

Na terceira entrega serão selecionados os cinco melhores modelos com base na média do *F1-score*. Nesta etapa, serão ainda exploradas diferentes *learning rates*, diferentes *optimizers* e *loss functions*, com vista à maximização do desempenho final.

9 Conclusão

O estudo da anatomia equina apresenta uma complexidade significativa, uma vez que a estrutura óssea do cavalo é constituída por um grande número de elementos que, para estudantes do curso de Medicina Veterinária, se revelam difíceis de memorizar. Esta dificuldade origina desafios académicos recorrentes, tornando pertinente o desenvolvimento de soluções que apoiem o processo de aprendizagem. Este trabalho surge precisamente com o propósito de analisar esse problema e propor uma abordagem tecnológica que contribua para a sua mitigação.

A análise inicial da pertinência e viabilidade demonstrou que o projeto é simultaneamente relevante e exequível. O testemunho do Doutor José Prazeres reforçou a necessidade de um recurso educacional adicional que auxilie os estudantes no estudo da anatomia equina. A viabilidade do projeto foi igualmente confirmada pelo facto de este constituir um complemento a um trabalho previamente aprovado, a aplicação móvel IEquus.

A revisão do estado de arte permitiu contextualizar o projeto no panorama atual das aplicações tecnológicas relacionadas com a saúde e ensino veterinário, tal como contextualizar o estado na qual a tecnologia utilizada se encontra atualmente. Esta análise possibilitou identificar soluções existentes, compreender as suas limitações e destacar os aspetos que tornam o presente trabalho particularmente distinto e necessário.

Posteriormente, procedeu-se ao teste e à comparação de diferentes algoritmos de reconhecimento de imagem. Através do teste de vários métodos de transformação de imagem e avaliação dos resultados através de métricas de desempenho comparados a dados de controlo, foi possível determinar que o modelo DINO-ViT apresentou o melhor desempenho entre os 15 modelos testados. No entanto, dado que os resultados obtidos são todos inferiores a 0.5, não é possível concluir se este desempenho se deve à insuficiência de dados ou à dificuldade intrínseca do *dataset* em ser corretamente classificado.

Bibliografia

[1] S. Xiao *et al.*, “Review of applications of deep learning in veterinary diagnostics and animal health,” 2025, *Frontiers Media SA*. doi: 10.3389/fvets.2025.1511522.

[2] P. Arsomngern, N. Numcharoenpinij, J. Piriyataravet, W. Teerapan, W. Hinthong, and P. Phunchongharn, “Computer-Aided Diagnosis for Lung Lesion in Companion Animals from X-ray Images Using Deep Learning Techniques.”

[3] Duarte Chen, “TFC IEquus.” Accessed: Nov. 25, 2025. [Online]. Available: https://teses.deisi.ulusofona.pt/media/teses/main.pdf

[4] PyTorch, “PyTorch.” Accessed: Nov. 25, 2025. [Online]. Available: https://pytorch.org

[5] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” Dec. 2015, [Online]. Available: http://arxiv.org/abs/1512.03385

[6] A. Dosovitskiy *et al.*, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale,” Jun. 2021, [Online]. Available: http://arxiv.org/abs/2010.11929

[7] C. Szegedy *et al.*, “Going Deeper with Convolutions,” Sep. 2014, [Online]. Available: http://arxiv.org/abs/1409.4842

[8] M. Caron *et al.*, “Emerging Properties in Self-Supervised Vision Transformers,” May 2021, [Online]. Available: http://arxiv.org/abs/2104.14294

[9] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” Apr. 2015, [Online]. Available: http://arxiv.org/abs/1409.1556

[10] PyTorch, “PyTorch - PyTorch.” Accessed: Nov. 22, 2025. [Online]. Available: https://pytorch.org/projects/pytorch

[11] Horace He, “The State of Machine Learning Frameworks in 2019.” Accessed: Nov. 22, 2025. [Online]. Available: https://thegradient.pub/state-of-ml-frameworks-2019-pytorch-dominates-research-tensorflow-dominates-industry

[12] S. Bianco, R. Cadene, L. Celona, and P. Napoletano, “Benchmark Analysis of Representative Deep Neural Network Architectures,” Oct. 2018, doi: 10.1109/ACCESS.2018.2877890.

[13] IMAIOS SAS, “IMAIOS vet-Anatomy.” Accessed: Nov. 22, 2025. [Online]. Available: https://play.google.com/store/apps/details?hl=en&id=net.imaios.vetanatomy&utm_source=chatgpt.com

[14] Siwalu Software GmbH, “Horse Scanner.” Accessed: Nov. 22, 2025. [Online]. Available: https://play.google.com/store/apps/details?id=com.siwalusoftware.horsescanner

[15] Osteo+, “Bone Identifier – Osteo+.” Accessed: Nov. 22, 2025. [Online]. Available: https://apps.apple.com/us/app/bone-identifier-osteo/id6738054129

10 Formulário de declaração de uso de ferramentas de Inteligência Artificial

Todos os relatórios deverão incluir anexo com cópia, devidamente preenchida, do formulário abaixo.
Assinalar as opções aplicáveis e completar os campos solicitados.

1. Utilização de IA

[ ] Não foram utilizadas ferramentas de IA na realização deste trabalho.
[X] Foram utilizadas ferramentas de IA na realização deste trabalho.

---

2. Ferramentas utilizadas

Assinalar todas as que se aplicam.

Assistência geral à escrita, análise ou ideação

[X] ChatGPT
[X] Microsoft Copilot
[X] Gemini
[ ] Claude
[ ] Perplexity
[ ] Outras. Quais? _____

Assistência à programação / desenvolvimento

[X] GitHub Copilot

[ ] Claude

[ ] OpenAI Codex
[ ] Cursor
[ ] Tabnine
[ ] Amazon CodeWhisperer / Amazon Q
[ ] Outras. Quais? ChatGPT _____

Geração de imagem / design / multimédia

[ ] DALL·E
[ ] Midjourney
[ ] Stable Diffusion
[ ] Canva AI / Magic Design
[ ] Outras. Quais? _____

Outros usos

[ ] Contexto: Ferramentas? _____

---

3. Fases do trabalho em que foi utilizada IA

[X] Planeamento do trabalho
[X] Pesquisa exploratória / levantamento inicial de informação
[X] Documentação técnica
[X] Redação do relatório
[ ] Desenho / modelação / arquitetura
[ ] Design / prototipagem / interface
[X] Geração de código
[X] Revisão / refatoração / debugging de código
[ ] Criação de testes / casos de teste
[X] Análise de resultados
[ ] Preparação de apresentação ou materiais auxiliares
[ ] Outros. Quais? _____

---

4. Tipo de utilização

Descrever sucintamente como a IA foi utilizada.

Exemplos: brainstorming, estruturação de secções, revisão linguística, sugestão de arquitetura, geração de exemplos, explicação de conceitos, geração parcial de código, correção de erros, criação de casos de teste, apoio ao design.

R: Revisão linguística, reformulação de texto, sugestão de conteúdo técnico, sugestão de referência, sugestão de arquitetura, explicação de conceitos, brainstorming, geração parcial de código, correção de erros, otimização de código, revisão de código, analise de resultado.

---

5. Partes do trabalho afetadas

Indicar as secções, componentes, módulos, ficheiros, entregáveis ou atividades que foram influenciados pelo uso de IA.

Em todo o relatório houve utilização de AI, principalmente para correção linguística e reformulação de texto quando era necessário. No código, no arquivo untils.py, imports.py e analiseexploratoria.ipynb.

6. Exemplos de *prompt*

Inserir exemplos de *prompt*, diferenciando por âmbito (enquadrado na questão 2) e fase (enquadrado na questão 4)

ChatGPT

Tipo: Analise de resultado

- "Tendo estes dados, para qual epoch cada modelo tende a convergir? (segue abaixo a tabela com os dados)"

Tipo: Correção de erro

- "Tendo este código (código) que me dá este erro (erro). Onde pode estar o problema?"

Tipo: Revisão linguística, reformulação de texto

- "Ajuda-me a reescrever este texto com uma linguagem mais científica/reescrever sem erros gramaticais, utilizando português de Portugal"

Tipo: Reformulação de texto, sugestão de conteúdo técnico

- "Tenho este texto sobre a arquitetura CNN (nome da arquitetura), mas sinto que não tem detalhes o suficiente sobre a arquitetura, ajuda-me a reformular o texto: (texto escrito sobre a arquitetura)"

Microsoft Copilot

Tipo: Reformulação de texto, sugestão de conteúdo técnico

- "Diga-me se o seguinte texto faz sentido para a estrutura de CNN (nome do modelo): (texto escrito sobre a arquitetura)"

Gemini

Tipo: Brainstorming

- "Sobre treinamento de modelos CNN.

Tenho uma função de treino usando fine-tuning, na qual também calcula o f1-score a cada época. Sendo assim, notei que, para alguns modelos, havia um pico de f1-score num determinado momento, mas depois baixava, apenas para, após 50/60 epochs, surgir um pico ainda maior.

A minha pergunta é: faria sentido eu colocar um early stopping após X epochs, sempre que o modelo calcular um f1-score menor do que o best_f1, ou devo deixar treinar por todas as epochs estimadas?

O número de epochs estimado é uma média de epochs em que cada modelo tende a convergir (dados com treinos anteriores)"

GitHub Copilot (Não houve prompts, apenas utilização do preenchimento automático)

---

7. Validação, revisão e intervenção dos autores

Descrever que verificação, revisão, correção, adaptação ou reescrita foi realizada pelos autores.

Nota: se a IA tiver sido usada em código, testes, scripts, modelos, consultas, configurações ou outros artefactos técnicos, deve ser indicado de que forma os autores validaram o funcionamento e confirmaram a sua compreensão.

Para a elaboração do relatório a verificação e revisão das sugestões de texto geradas pela IA basearam-se no pensamento critico e no contexto do TFC. Sempre que a ferramenta produzia informação discordantes ou fora do âmbito do projeto, o conteúdo era descartado e reescrito manualmente para garantir a fidelidade dos dados reais. No que diz respeito à discrição dos modelos, a validação foi feita através da triangulação de dados: os outputs da IA foram rigorosamente confrontados com pesquisas prévias em fontes científicas.

No desenvolvimento do código e nos momentos de brainstorming, foram seguidas as normas e boas práticas da investigação científica na área de CNNs. A validação do código incluiu o teste de execução, a análise detalhada dos outputs e a comparação do código com exemplos de códigos consolidados em estudos análogos.

Todas as sugestões de referências, modelos ou ferramentas fornecidas pela IA foram utilizadas apenas como ponto de partida. Cada elemento foi alvo de um estudo aprofundado em bases de dados científicas como IEEE Xplore, Elsevier e arXiv, garantindo a precisão técnica e a integridade académica do trabalho.

---

8. Grau de utilização

[ ] Residual
[X] Moderado
[ ] Extensivo

[X] Utilização homogénea
[ ] Grau de uso diferenciado por fase ou componente de trabalho

Descrever sucintamente os diferentes usos.

A utilização da IA foi homogenia em todo o trabalho. Na fase de escrita serviu principalmente para revisão linguística e polimento textual.

Na fase conceptual e teórico foi utilizada numa fase inicial como motor de busca para identificar e sugerir tecnologias.

Já na componente técnica a IA apoiou na geração parcial de código, a otimização do mesmo e a correção de erros, sendo cada sugestão validada e integrada manualmente após testes de funcionamento.

---

9. Trabalhos em parceria

Protecção de dados confidenciais e recursos proprietários de parceiros

[X] O trabalho foi realizado em parceria com entidade externa ao DEISI

No caso da resposta anterior ser verdadeira, responder às seguintes questões:

[ ] O parceiro tem regras para restringir submissão de dados

[X] As submissões validam aplicação de regras de tratamento de dados

[ ] Foram implementados mecanismos para restringir a partilha de recursos proprietários

---

10. Declaração de responsabilidade

Ao assinarem a presente declaração, os autores declaram que:

- a informação acima é verdadeira e reflete o uso efetivo de ferramentas de IA na realização do trabalho;
- compreendem que a IA não substitui autoria nem responsabilidade académica;
- verificaram a validaram e veracidade das referências bibliográficas incluídas no relatório
- assumem integralmente a responsabilidade técnica, científica, ética e académica por todo o conteúdo submetido, incluindo texto, código, modelos, testes, imagens, diagramas e restantes artefactos entregues.

---

11. Identificação dos autores

Nome(s): Matilde Giusti Ferreira Rodrigues

Número(s): 22306236

Data: 11/04/2026

Assinatura(s): Matilde Rodrigues