INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Deteção e identificação de doenças em plantas utilizando Deep Learning

DANIEL CARLOS PIRES GARCIA COSTA BENTO
Outubro de 2019

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

PlantAI

Deteção e identificação de doenças em plantas utilizando Deep Learning

Daniel Carlos Bento

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Engenharia de Aplicações

Orientador: Maria de Fátima Rodrigues

Coorientador: Carlos Gomes Ferreira

Porto, outubro 2019

“The computer is incredibly fast, accurate, and stupid. Man is incredibly slow, inaccurate, and brilliant. The marriage of the two is a force beyond calculation.”

Leo Cherne

Resumo

A automação de trabalho é hoje em dia, uma prática cada vez mais recorrente em diversas áreas, pois permite diminuir a necessidade de mão de obra e muitas vezes melhorar os erros inerentes ao fator humano.

A deteção de doenças em plantas hoje em dia ocorre pela via tradicional, onde é necessário o trabalho de um técnico especializado, que inspeciona e emite um parecer sobre a possível patologia. Este processo leva a uma serie de problemas.

Nesta dissertação, será apresentada uma solução capaz de auxiliar o trabalho de um profissional, utilizando técnicas de *Deep Learning*, nomeadamente recorrendo a *Convolutional Neural Networks* para auxiliar na detenção de patologias, de forma a providenciar um diagnostico atempado, sugerindo também tratamentos caso a patologia os necessite.

Palavras-chave: Automação, *Deep Learning*, *Convolutional Neural Networks*, Doenças em Plantas

Abstract

Work automation is nowadays an increasingly recurring practice in many areas, as it reduces the need for manpower and often improves the inherent errors of the human factor.

The detection of diseases in plants today occurs in the traditional way, where it is necessary for the work of a specialized technician, who inspects and gives an opinion on the possible pathology. This process leads to several problems.

In this dissertation, a solution will be presented that can assist the work of a professional, using Deep Learning techniques, namely using Convolutional Neural Networks to assist in the detection of pathologies, to provide a timely diagnosis, also suggesting treatments if the pathology causes need it.

Keywords: Deep Learning, Convolutional Neural Networks, Automation, Plant disease

Agradecimentos

O meu primeiro agradecimento vai para a todos aqueles que contribuíram direta e indiretamente para a conclusão da minha formação, para os meus professores, para os meus colegas de turma e para os meus colegas de grupo, que são todos aqueles que me ajudaram a ultrapassar as dificuldades e obstáculos, contribuindo para a minha evolução e aprendizagem, que levaram ao sucesso nesta etapa.

Do meu colega Ricardo Rocha não me posso esquecer. Fez comigo vários trabalhos do mestrado, sempre com empenho, dedicação e prontidão para ajudar a resolver problemas. A ele um enorme obrigado, e votos de muito sucesso pessoais e profissionais.

A minha orientadora Maria de Fátima Rodrigues e ao professor Carlos Ferreira, que me orientaram neste enorme trabalho, pelas críticas, opiniões e sugestões, que permitiram o crescimento e desenvolvimento desta dissertação.

Aos meus pais e à minha irmã, pelo apoio perante os desafios, pela alegria e distração nos momentos menos bons e sobretudo, pela compreensão dos momentos ausência e preocupação durante estes últimos anos.

Por fim, um agradecimento especial à minha namorada, pela confiança e pelo valor atribuído ao meu trabalho, pela motivação e alento em momentos de dúvida. Pela amizade, carinho, solidariedade e compreensão.

Índice

Lista de Figuras

Lista de Tabelas

Acrónimos e Símbolos

Lista de Acrónimos

CNN	*Convolutional Neural Network*
RNN	Recurrent Neural Network
AI	Artificial Inteligence
DL	*Deep Learning*
ML	Machine Learning
NCD	*New Concept Development Model*
FEI	*Front End of Inovation*
CRISP-DM	*Cross Industry Standard Process for Data Mining*
API	*Application Programming Interface*
FLOPS	*Floating Point Operation per Second*
NEA	Nível Economico de Ataque
SAVE	*Society of American Value Engineers*

Lista de Símbolos

μ	Taxa de acerto
τ	Tempo

1 Introdução

1.1 Contexto

O desenvolvimento económico de um país, está em parte relacionado com o sucesso da atividade agrícola e da sua produtividade, aliás a sobrevivência humana está diretamente relacionada com a dependência da agricultura. Os agricultores cultivam várias culturas, com base na fertilidade do solo e também dos recursos disponíveis, no entanto as alterações climáticas como as variações de chuva, temperaturas e outros fatores, potenciam infeções nos solos ou plantas.

Doença é o mal funcionamento de células e tecidos do hospedeiro que resulta da sua contínua irritação por um agente patogénico ou fator ambiental e que conduz ao desenvolvimento de sintomas. Doença é uma condição envolvendo mudanças anormais na forma, fisiologia, integridade ou comportamento da planta. Tais mudanças podem resultar em dano parcial ou morte da planta ou de suas partes [1].

As doenças que aparecem frequentemente nas plantas e culturas agrícolas constituem desta forma uma das principais preocupações para quem gosta de ter as plantas bonitas e saudáveis. Cumulativamente, também os técnicos e produtores que têm no sector agrícola a sua principal atividade, preocupam-se cada vez mais em proporcionar as melhores condições edáficas e climáticas para o desenvolvimento das suas culturas, o que é muitas vezes comprometido pelo aparecimento de problemas fitossanitários.

Embora a melhor alternativa nestes casos seja priorizar os métodos preventivos para evitar o desenvolvimento de problemas fitossanitários, muitas vezes é completamente impossível evitar que sejam aplicadas medidas curativas de combate a estas enfermidades.

Atualmente, verifica-se uma maturação e desenvolvimento crescente da tecnologia agrícola em geral, com o aparecimento a um ritmo alucinante de novas aplicações e softwares para facilitar o dia-a-dia do produtor agrícola.

Uma das metodologias mais recentes e que se encontra em claro crescimento, é a metodologia *Machine Learning*, que surge ligado ao conceito de Inteligência Artificial. As grandes vantagens das aplicações que utilizam este tipo de metodologia é identificar de forma automática padrões de comportamento de certos fatores agronómicos, mas também, e não menos importante, auxiliar na previsão de comportamentos agronómicos futuros. *Machine Learning* é uma área de Inteligência Artificial relacionada com o estudo de uma máquina usando de algoritmos, que melhoram automaticamente através da experiência. Na prática isso envolve a criação de software que otimiza um critério de desempenho através de análise de dados [2].

1.2 Problema

Um dos problemas existentes na área de produção no sector agrícola está relacionado com o aparecimento de doenças, que nas plantações provocam um impacto direto nos rendimentos de produção, como tal, este problema gera uma grande preocupação tanto nos técnicos como nos próprios produtores agrícolas. Para identificar as doenças nas plantas é necessário o trabalho de um técnico especializado, que inspeciona e emite um parecer sobre a possível patologia. Esta abordagem traz diversos problemas, entre eles:

- A ausência de disponibilidade do técnico;
- A ambiguidade da classificação que poderá ocorrer caso haja uma variação de algum fator, como por exemplo avaliação do técnico a avaliar, pois a avaliação humana depende da interpretação de quem avalia. Avaliadores diferentes nas mesmas condições, permitem avaliações diferentes.
- O conhecimento do técnico sendo volátil e que se poderá perder caso haja uma situação de reforma; a formação de novos especialistas poderá ser um processo demorado.

De modo a trabalhar o problema, esta dissertação terá como base a utilização do conjunto de dados PlantVillage, uma vez que não existem muitas fontes de informação para trabalhar problemas de doenças em plantas e também atendendo ao tempo disponível, seria inviável elaborar todo o processo de recolha e catalogação de dados.

1.3 Objetivos

Pretende-se elaborar um estudo da viabilidade do reconhecimento de doenças em plantas utilizando técnicas de *Deep Learning* de reconhecimento de imagem. Como resultado desse estudo, pretende-se criar uma solução, denominada de PlantAI, que irá disponibilizar duas aplicações, sendo que uma delas uma aplicação móvel e outra um portal.

A aplicação móvel será desenvolvida com o objetivo de disponibilizar as funcionalidades de reconhecimento de doenças em plantas e obtenção de tratamentos, independentemente da localização do utilizador.

O portal, além dos objetivos anteriormente enunciados, irá possuir um conjunto de funcionalidades de manutenção e gestão do sistema.

1.4 Análise de Valor

A implementação deste protótipo, permitirá estudar a viabilidade de soluções mais completas para auxílio do negócio agrícola. Neste sentido, caso a eficiência consiga ir de encontro aos objetivos do projeto, então passa a fazer sentido investir na criação de uma aplicação desta natureza para um contexto real, onde será um bom auxílio dos agricultores e profissionais agrícolas, reduzindo a dependência humana, aumentando desse modo a autonomia dos utilizadores do sistema.

Alem dos benefícios elencados, anteriormente, esta solução poderá permitir uma redução de custos pois deixa de ser necessário pagar a deslocação e trabalho de um técnico, sendo que se for utilizado de forma mais preventiva, poderá antecipar a identificação da doença e deste modo melhorar a eficiência dos tratamentos nas plantas.

1.5 Contribuições e resultados esperados

As contribuições e resultados esperados pela realização desta tese são a criação de uma solução que permita demonstrar a possibilidade de reconhecer doenças em plantas recorrendo a mecanismos de Inteligência Artificial; criação de uma aplicação móvel capaz de demonstrar o mecanismo criado de um modo prático para poder ser utilizado no terreno; criação de uma alternativa à aplicação, em forma de portal na internet, com funções auxiliares para gestão da base de dados.

1.6 Abordagem Preconizada

A abordagem a adotar inclui o desenvolvimento de modelos de deteção de doenças através da análise de fotografias de plantas usando técnicas de *Deep Learning*. Os modelos terão como base fotografias de plantas com doenças e sem doenças. Para assegurar a qualidade do desenvolvimento dos modelos de deteção de doenças será utilizada a metodologia padrão CRISP-DM [3], que é um padrão de referência que define as fases a seguir, as tarefas a serem executadas e os resultados esperados pela realização de cada tarefa. Este padrão envolve os seguintes processos:

A recolha de dados é primeiro processo sendo determinante em todo o processo, pois a quantidade e qualidade de dados consegue ter impacto em todas as fases posteriores, podendo em caso de insucesso nesta etapa levar ao insucesso também de todo o projeto.

Os dados recolhidos no processo anterior, normalmente são imagens ou conjuntos de imagens, estas não se encontram preparadas para entrar no processo de treino. Para isso, existe este processo de pré-processamento, que pode englobar várias alterações e filtros das imagens, de forma a transformar imagens diferentes em imagens com as mesmas dimensões, alterações de tonalidades e conversões de formatos, nomeadamente do formato de imagem para uma matriz de pixels.

Na fase de treino, é importante testar várias arquiteturas da rede neuronal bem como o melhor conjunto de parâmetros de forma a obter modelos com alta precisão.

Depois da fase de treino estar concluída, é necessário avaliar o modelo gerado, e para isso é utilizado o mecanismo de teste do modelo, que é composto por imagens que não entraram na fase de treino. Caso o modelo não cumpra com os objetivos, poderá ser necessário usar novos algoritmos, ou ajustar parâmetros ou camadas, e voltar novamente ao processo de treino, até que o modelo atinja um desempenho satisfatório.

Depois da fase de avaliação estar concluída, significa que o modelo já tem a precisão desejada, sendo necessário colocar o modelo em modo de produção, de modo a começar a efetuar previsões em contexto real.

1.7 Estrutura do Documento

O primeiro capítulo deste documento apresenta uma introdução e interpretação do problema, apresentando o contexto envolvido nesta tese, o problema com o qual se irá trabalhar e os objetivos para o projeto. Ainda neste capítulo se fez uma breve apresentação da análise de valor do projeto, e foram abordados os objetivos pretendidos com esta tese.

O segundo capítulo é destinado ao contexto, onde irá ser apresentado alguns tópicos relativamente à área de negócio, bem como à análise de valor da solução num contexto de área de negócio.

O terceiro capítulo, corresponde ao capítulo do estado da arte associado à área de Inteligência Artificial. Neste capítulo é importante referir todo o estudo elaborado sobre a área que irá incidir a solução PlantAI, deste modo irá abordar temas e conceitos sobre *machine learning*, redes neuronais, tecnologias envolventes no meio das redes neuronais e ainda demonstrará algumas soluções direta ou indiretamente relacionadas com a solução pretendida.

O quarto capítulo, diz respeito ao design e arquitetura da solução, demonstrando as propostas de arquitetura e alternativas à proposta elencada.

O quinto capítulo, corresponde ao capítulo de desenvolvimento da solução, onde é apresentado todos pontos componentes desenvolvidos no decorrer desta dissertação.

O sexto capítulo, capítulo apresenta toda a matéria relacionada com as experiências e demonstração dos resultados, nomeadamente os mecanismos e metodologias de avaliação da solução, onde são definidos os critérios de aceitação da solução proposta; as metodologias das práticas adotadas para a elaboração do trabalho; os resultados de todas as experiências desenvolvidas para o desenvolvimento do modelo de classificação nesta dissertação.

O último capítulo desta dissertação, corresponde ao capítulo das conclusões da dissertação.

2 Contexto

Este capítulo irá abordar diversos temas relacionados com o contexto da área de negócio, a análise de valor, onde consta a proposta de valor, o modelo *Canvas* e a avaliação da solução ajustada ao modelo *New Concept Development*.

2.1 Área de Negócio

Nos dias de hoje, um dos principais desafios dos produtores agrícolas em todo o mundo prende-se com o facto de ser cada vez mais difícil efetuar uma gestão sustentável das suas culturas.

De facto, as plantas são frequentemente atacadas por pragas e doenças, que interferem negativamente no rendimento final das culturas e acarretam custos elevados para os agricultores nomeadamente custos relacionados com tratamentos fitossanitários e/ou ações de consultadoria feitas por técnicos especializados na tentativa de identificação do foco do problema [4]–[6].

2.1.1 Tomada de decisão: como fazer?

Qualquer técnico ou consultor agrícola no processo de tomada de decisão no que diz respeito a um determinado tratamento fitossanitário, deve cumprir uma série de passos sequenciais para determinar como e quando agir de forma a minimizar danos numa determinada cultura agrícola [4]–[6].

2.1.2 Tratamentos preventivos ou tratamentos curativos: qual a melhor opção?

Antecipar momentos críticos de ataque através de tratamentos preventivos deve ser imperativo para conseguir antecipar momentos críticos de ataque para uma agricultura mais sustentável. Com o principal objetivo de evitar custos desnecessários associados à aplicação de

produtos fitofarmacêuticos de forma desadequada, os tratamentos preventivos permitem fundamentalmente minimizar custos futuros nomeadamente: custos associados aos produtos fitofarmacêuticos, custos associados ao combustível gasto com maquinaria agrícola, custos associados às horas de trabalho da mão-de-obra, entre outros.

Por outro lado, os tratamentos preventivos permitem que se consiga manter os níveis de toxicidade abaixo dos limites admissíveis e aconselhados, o que é sempre o mais aconselhado. Por outro lado, a decisão de aplicação de um determinado tratamento fitossanitário deve ser feita de forma pensada e antecipada (i.e., preventivamente antes de aparecer o foco do problema e não corretivamente, quando o foco do problema fitossanitário já se manifestou).

No entanto, é importante ressalvar que embora seja sempre sugerido fazer tratamentos preventivos e não curativos, aplicar um determinado produto demasiado cedo pode não ter os efeitos que realmente se deseja, e por essa razão, é importante abordar o conceito do Nível Económico de Ataque (NEA).

O NEA corresponde à intensidade de ataque do inimigo da cultura a que devem ser aplicar medidas limitativas, de forma a impedir que a cultura sofra prejuízos superiores ao custo das medidas a adotar. A estes custos, devem ser também incluídos os efeitos indesejáveis que podem ocorrer do uso inapropriado deste tipo de produtos no sector agrícola, e por essa razão, devem ser alvo da máxima atenção por parte do aplicador.

Em suma, é essencial acompanhar frequentemente a suscetibilidade da cultura agrícola a determinadas pragas e doença e após isso, decidir qual o melhor momento para aplicar um determinado tratamento fitossanitário [4]–[6].

2.1.3 Escolha do momento ideal para efetuar o tratamento

É importante estar consciente qual o momento ideal para efetuar o tratamento fitossanitário nas suas culturas agrícolas que estão afetadas, e isso inclui saber qual o melhor dia e hora para tratar as suas culturas, idealmente de forma preventiva.

As condições ótimas para aplicar os tratamentos fitossanitários nem sempre são possíveis de serem garantidas, no entanto, tal é possível de ser minimamente contornado com a consulta da meteorologia para os próximos dias de forma atempada. Fatores climáticos tais como temperatura, humidade, chuva e velocidade do vento são dos fatores mais relevantes que se deve ter em consideração no momento em que pretende tratar determinado problema fitossanitário[4]–[6].

2.1.4 Eficácia do tratamento: como acompanhar?

É importante saber determinar quais os momentos em que uma cultura está mais suscetível a uma determinada doença, e que está pode durar vários dias, dependendo como é óbvio, das condições meteorológicas a que pode estar sujeito.

Assim, fazer um acompanhamento da persistência biológica de um determinado tratamento é fundamental para decidir se deve ou não reaplicar um determinado produto e para que, caso afirmativo, a aplicação seja feita de forma consciente.

Repetir um tratamento implica saber em primeiro lugar se o NEA se mantém igual. Depois é conveniente saber quais as condições metrológicas se mantêm nos próximos dias para decidir o que se deve fazer, e caso estas duas premissas anteriores se verifiquem é fundamental verificar se o produto anteriormente aplicado ainda mantém a eficácia para poder ser novamente aplicado.

Para tal, é fundamental que o seja efetuado um acompanhamento regular do intervalo d segurança dos produtos aplicados e/ou a aplicar, para evitar se iniciar a colheita sem a certeza completa que os níveis de toxicidade são inferiores aos limites recomendados [4]–[6].

2.1.5 Doenças

O propósito desta tese é responder à questão sobre se é possível efetuar uma classificação de doenças de uma planta, com determinado grau de confiança. Desse modo, não é considerado relevante para esta tese fazer um estudo sobre quais as doenças mais significativas a serem abordadas, pois para se efetuar um estudo destes é necessário obter dados, sendo que o grau de confiança cresce com a proporção de dados incluídos para o processo de treino da rede neuronal.

Esta área de doenças em plantas, não é uma área que dispõe de uma grande quantidade de dados, comparativamente com outras áreas estudadas para efeitos de criação de processos automáticos. Os dados utilizados nesta tese são dados disponibilizados por terceiros, sendo que não é o propósito da tese, nem tão pouco seria viável do ponto de vista de tempo estar a recolher milhares de imagens e dados de determinada doença, para depois se analisar esses dados. Deste modo, o capítulo das doenças passa a ser menos relevante, face aos resultados que será possível aferir em função da quantidade de dados que será possível analisar.

2.2 Análise de Valor

Em 1947, Lawrence Miles criou um conceito básico sobre a análise de valor, onde defendia que o valor é o preço mais baixo que se pode pagar pelo fornecimento de uma função ou um serviço confiável e que representa o rácio entre a função e o seu custo. Este conceito, ficou marcado como a base da análise de valor. A *Society of American Value Engineers*, conhecida como SAVE, descreve também a análise de valor como uma abordagem sistemática e estruturada para o melhoramento de projetos, produtos, processos e serviços, onde permite ajudar a obter um equilíbrio entre diversos parâmetros: função; desempenho; qualidade; segurança e Custos. Este equilíbrio resulta na proposta de valor máximo para um projeto [7].

Deste modo, a SAVE, sugere que a definição da proposta de valor pode estar relacionada com uma fórmula matemática:

$$Value = \frac{Worth}{Cost} = \frac{Function}{Cost} \quad (1)$$

Em que a função representa a funcionalidade/utilidade do produto ou serviço pretendido e o custo associado para o seu desenvolvimento e que vai de encontro ao princípio mais básico do consumo, que é o balanceamento entre a tentativa de obter o melhor produto/serviço pelo menor preço possível.

2.2.1 Proposta de Valor

A proposta de valor é um aspeto abordado no modelo de *canvas*, em que Alexander Osterwalder indica que é um processo de análise focado no cliente e nos seus requisitos. No modelo de *canvas*, a proposta de valor pretende responder à necessidade de perceber porque é que um cliente precisa de um serviço ou produto; de que forma é que o cliente tem interesse e olha para determinado produto como um valor extra e também perceber o que é que pode representar como desvantagens [8].

O desenvolvimento de uma ferramenta capaz de classificar doenças em plantas recorrendo a um sistema digital, que não necessite da presença de um técnico ou especialista poderia ser uma solução diferente no mercado, pois seria possível reduzir a dependência humana, pelo menos numa fase inicial de diagnostico. Ao mesmo tempo, aumentaria a autonomia dos profissionais do sector, reduzindo necessidades como disponibilidade de terceiros e isto significa que uma vez que não seria necessário a presença de um especialista, poderiam os profissionais, fazer uma avaliação mais regular das culturas de forma a criar um movimento preventivo mais eficiente.

Além do diagnostico, o sistema deverá providenciar sugestões de tratamentos a aplicar em caso de necessidade para tratamento de doenças.

Além dos benefícios referidos anteriormente, esta aplicação por exemplo associada a utilização de *drones* ou outros robôs, poderá criar uma proposta de valor ainda mais atrativa, de modo a que a criação deste protótipo com os objetivos propostos, permite a criação de novas ideias que podem acrescentar mais soluções ao mercado.

2.2.2 Valor e Valor Percebido

O valor representa o valor monetário, material ou avaliado de um determinado produto ou serviço, que poderá ser diferente do valor atribuído pelo cliente (valor percebido).

O valor percebido, representa o valor que o cliente esta disposto a ceder para beneficiar de um produto ou serviço. É importante perceber que o mercado exige que o valor que uma empresa atribui a um produto ou serviço, tem que ir ao encontro do valor que o cliente está disposto a pagar, porque quando um cliente não consegue olhar para os benefícios como algo que vai proporcionar algo de positivo ou algo diferenciador, caso contrário, perde o interesse e consequentemente o valor.

Tabela 1 - Tabela de Custos e Benefício do produto

	PRODUTO
BENEFICIOS	Qualidade
	Desempenho
	Facilidade de uso
	Redução de dependência
	Redução de tempo na identificação de doenças
	Utilidade
	Sugestões de tratamento
SACRIFÍCIOS	Custo de desenvolvimento

2.2.3 Modelos de análise de valor de negócio

A análise de rede de valor é uma metodologia de modelação de negócio, que visualiza as atividade e relações de uma perspetiva dinâmica, que inclui várias abordagens de analises executivas e outras ferramentas de modelação: processos, sociais e dinâmicas.

Verna Allee, defende que cada vez mais, o conhecimento e outros ativos intangíveis, como a competência humana, a capacidade de formar novas relações fortes, a capacidade de colaboração mútua benéfica, podem ser os alicerces do sucesso[9].

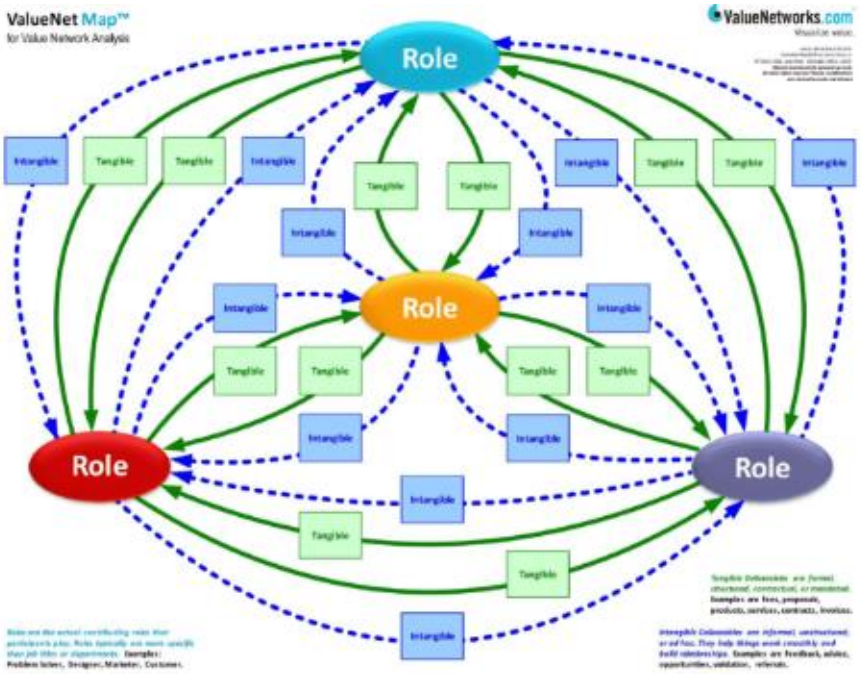


Figura 1 - Rede de valor de Verna Allee

A cadeia de valor de Michael Porter é um modelo que ajuda a analisar atividades especificas através das quais as empresas criam valor e vantagem competitiva, ou seja, é um conjunto de atividades que uma organização realiza para criar valor para os seus clientes. A maneira como as atividades dessa cadeia são realizadas determina os custos e afeta os lucros [10].

Figura 2 - Cadeia de valor de Michael Porter

De acordo com os dois modelos estudados, o modelo de Verna Allee parece o que melhor se enquadra com este projeto, pois parcerias e relações entre profissionais agrícolas, marcas de produtos e outros intervenientes seriam a base do sucesso num negócio ligado a este projeto, pois seria através de uma teia de fatores influenciadores que seria desenvolvido e potenciado o sistema.

2.2.4 New Concept Development Model (NCD)

O modelo NCD fornece uma linguagem comum e visão holística do *front-end*. O modelo divide o *front-end* em três áreas distintas. O primeiro é o motor, ou centro do modelo, responsável pela visão, estratégia e cultura que impulsiona a FEI. A segunda parte interna define os cinco elementos de atividade do *front-end*. O terceiro elemento consiste nos fatores ambientais externos [11].

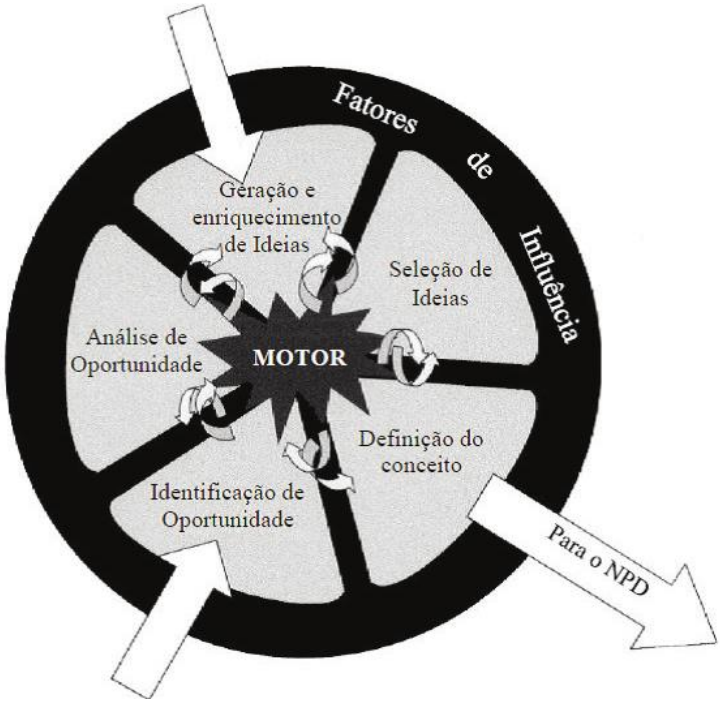


Figura 3 - New Concept Development Model

2.2.4.1 Identificação da oportunidade

O processo de análise de oportunidades inicia pela análise de mercado, para identificação das necessidades do mercado, de forma a explorar as oportunidades de novas ou diferentes ferramentas capazes de acrescentar valor a um mercado.

O sector agrícola, comparativamente com outros sectores, é um sector que ainda se encontra em desenvolvimento informático, ou seja, é um mercado que ainda trabalha muito com a ausência de informática, no entanto, as novas gerações deverão assumir um papel mais relevante nesta matéria. Assim sendo, as aplicações informáticas para o sector agrícola serão um mercado em expansão nos próximos anos. A identificação de doenças em plantas também utiliza o método tradicional como é descrito neste documento.

2.2.4.2 análise de oportunidade

A análise de oportunidade é um processo que visa o estudo das oportunidades de negócio, de forma a perceber, de que forma é que o mercado poderá entender e aproveitar a solução, para que seja rentável a conceção da solução.

A utilização de *Deep Learning*, poderá acrescentar não só um desenvolvimento informático, como também poderá automatizar e agilizar o processo de identificação de doenças, e isso poderá criar impacto económico nas organizações, reduzindo a dependência de terceiros, mas mantendo a autonomia.

2.2.4.3 Geração e enriquecimento de ideias

O fator de geração e enriquecimento de ideias, está relacionado a criação de novas ideias ligadas a ideia original, criando desse modo complementos ou alterações a ideia do projeto de forma a que possa servir melhor o cliente.

Para esta fase, foi discutido o projeto com um profissional do sector, que sugeriu que além de poder obter a classificação da doença de determinada planta, ainda possa obter uma sugestão de tratamento associada a doença. Outra ideia, seria criar também uma linha de apoio consultivo para poder dar suporte aos profissionais do sector para garantir melhor confiança aos clientes.

2.2.4.4 Seleção de ideias

Após a geração de ideias, é necessário definir e priorizar as ideias, para poder transformar uma ideia, no desenvolvimento de uma solução, tendo já em consideração que a escolha das opções deve respeitar uma relação entre necessidade, benefício e custo.

Deste modo, para a solução, ficou definido que seria uma mais valia para uma primeira fase, poder obter uma sugestão de tratamento para as doenças após a classificação da mesma.

2.2.4.5 Definição do conceito

O modelo de NDC, termina com a definição do conceito e isto significa que deve haver várias avaliações, entre elas, das necessidades e benefícios dos clientes, dos requisitos de investimento bem como a avaliação da concorrência existente para o segmento de mercado, as tecnologias e mecanismos necessários e por fim, qual o risco inerente ao projeto. Para este

projeto, esta definição está definida no modelo de *Canvas*, onde apresenta uma avaliação de ambas as vertentes da definição do conceito.

2.2.5 Modelo *Canvas*

A análise de valor de um determinado produto permite identificar o motivo pelo qual o mercado deve aceitar o produto ao qual estamos a analisar em detrimento de um possível concorrente. Nesse sentido, o modelo de *canvas* é um ótimo auxiliar para demonstrar uma proposta de valor, descrevendo o plano estratégico de uma forma rápida, ágil e elucidativa.

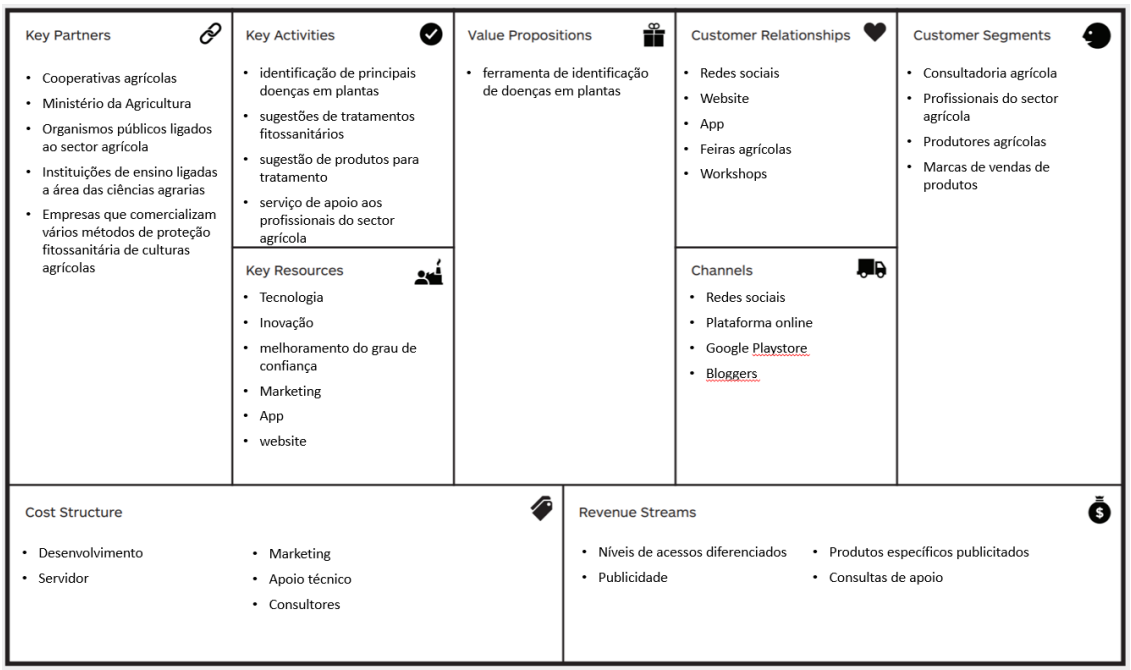


Figura 4 - Modelo de *Canvas*

O modelo de *canvas*, tem nove seções, sendo elas compostas por parcerias chave; atividades chave; proposta de valor; relacionamento com o cliente; segmentos de cliente; Recursos chave; canais; estruturas de custo e por últimas fontes de receita.

Começando pelo primeiro item, os parceiros chave são compostos por entidades que podem contribuir para o sucesso do negócio. Deste modo, as cooperativas agrícolas são grandes grupos, que podem potenciar o negócio. O ministério da agricultura e outros organismos públicos, também podem ser um parceiro importante, pois podem criar uma grande influência junto do mercado deste projeto. Para se atingir um mercado a longo prazo, também foi considerado os institutos de ensino, que podem entrar no sistema como modo de formação de futuros utilizadores e colaboradores do sistema. As empresas que comercializam produtos e métodos

de proteção fitossanitária de culturas agrícolas, pois também têm a capacidade de influenciar massas.

O segundo item diz respeito às atividades chave, estas são os aspetos essenciais, que irão ser oferecidos ao mercado, com o objetivo de criar valor para obter rendimentos. Neste sentido, o principal foco do sistema passa pela identificação de doenças em plantas, no entanto, a aplicação também deverá providenciar sugestões de tratamentos fitossanitários, bem como sugerir produtos para esses tratamentos. Como último aspeto, é considerado o serviço de apoio a profissionais, criando um modo de apoio a quem trabalha diariamente com doenças de plantas.

O terceiro corresponde à proposta de valor, ou seja, o motivo pela qual os clientes devem escolher o nosso produto. Neste item, a proposta apresentada é a criação de uma ferramenta que vise a identificação de doenças em plantas, bem como a possibilidade de obter mais informações relevantes como tratamentos e produtos a aplicar nos tratamentos. Outro aspeto importante é possibilitar o acompanhamento a profissionais. Para finalizar este item é importante referir que este sector não é um sector com muita oferta em material informático, neste sentido uma ferramenta inovadora e interativa poderia ser um aspeto muito relevante no mercado, sendo que uma versão de aplicação móvel que funcione de forma *offline* poderia reduzir a dependência da internet, que em determinados clientes poderá ser um aspeto relevante.

No modelo de *canvas*, o quarto item é a relação com os clientes, onde se pretende descrever o tipo de relação para cada segmento de cliente e nesse sentido, num modelo de negócio, para potenciar a relação com o cliente final as ferramentas digitais que iriam ser utilizadas seriam as redes sociais, um *website* ou alguma plataforma online bem como a aplicação desenvolvida. Como métodos não digitais, a promoção através de feiras agrícolas e *workshops* seriam excelentes métodos para consolidar uma relação de confiança com o mercado.

A secção de segmentos de clientes, serve para descrever os diferentes segmentos que o sistema poderá influenciar, tais como consultadorias agrícolas, bem como os profissionais do sector e produtores agrícolas. As marcas de vendas também podem ser um segmento a ser trabalhado, já que um sistema deste tipo potencia também a publicidade de produtos.

O item seguinte são os recursos chave. Estes são as principais ações que devemos fazer para o modelo funcionar. Este sistema, necessita sempre de estar envolvido com tecnologia, ser inovador e ser um sistema que possa melhorar na medida em que o número de utilizadores aumente. Outro aspeto importante a referir é o website e a aplicação, bem como o *marketing* envolvido.

Outro item importante são os canais de distribuição, que são os mecanismos que fazem chegar o produto ao cliente onde se consideram as redes sociais, *bloggers*, a própria *Google Playstore* e ainda a plataforma.

O item seguinte, é a estrutura de custos, que é uma secção que serve para descrever os principais custos envolvidos no negócio. Neste item é importante destacar o custo de desenvolvimento do sistema, os custos inerentes a manutenção do servidor. O custo de *marketing*, bem como de apoio técnico e ainda uma linha de consultores agrícolas.

Por fim, a secção de fontes de receitas que como o nome indica, serve para representar a proveniência das receitas que o negócio pode obter. Para isso, a publicidade poderá ser uma receita relevante, pois as sugestões de tratamentos requerem muitas vezes produtos, que por sua vez vão gerar rendimentos a terceiros. A ideia seria explorar esses mesmos rendimentos. Outro mecanismo de receitas seria através da criação de diferentes níveis de acesso com mais funcionalidades mediante subscrições bem como consultas de apoio específicas.

3 Estado da arte

Este capítulo refere-se ao estado da arte, sendo este o suporte para todo o trabalho desenvolvido.

Neste capítulo será apresentado todo o estudo efetuado, iniciando pela noção de *machine learning*, onde aborda três diferentes ramos da *machine learning*, sendo eles aprendizagem supervisionada, aprendizagem não supervisionada e aprendizagem por reforço.

Ainda será exposta uma secção dedicada as redes neuronais, desde a história das redes neuronais e será ainda abordado o funcionamento das mesmas. Sendo este trabalho ligado as CNN, será apresentado uma secção dedicada a este tipo de redes, abordando diferentes arquiteturas.

Ainda referente a este capítulo será demonstrado uma secção dedicada as tecnologias e bibliotecas utilizadas em problemas relacionados com redes neuronais.

O capítulo termina com a apresentação de soluções existentes e relacionadas com os objetivos desta dissertação.

3.1 *Machine Learning*

Machine Learning é um conceito relacionado com aprendizagem automática e nesse sentido existem três ramos que serão apresentados nesta secção: aprendizagem supervisionada, não supervisionada e por reforço.

3.1.1 Aprendizagem supervisionada

No ramo da aprendizagem supervisionada, os dados de entrada estão previamente classificados e são usados para a fase do treino da rede neuronal, recorrendo a um modelo, que procura

reduzir o erro, corrigindo-se automaticamente, alterando os pesos na rede. Neste sistema, existe duas abordagens de aplicação diferentes, numa dividem-se os dados em dados de treino e dados de teste, ou noutra em que se dividem em dados de treino, validação e teste. Esta última abordagem é mais eficiente, pois permite criar uma isolação dos dados de teste de todo o modelo de aprendizagem, uma vez que os dados de validação são usados para ajustar os pesos da rede ficando os dados testes apenas para validar o modelo obtido [12]–[14].

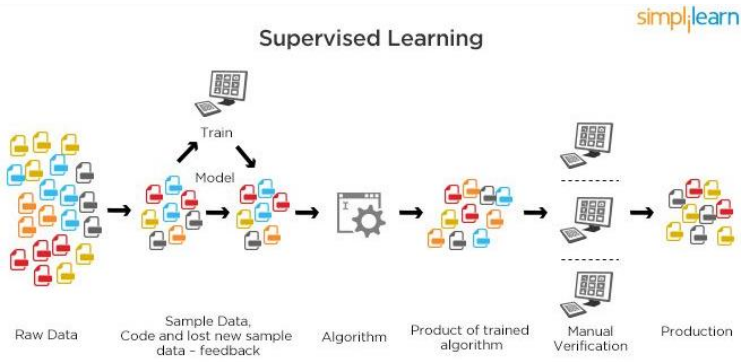


Figura 5 - Aprendizagem não supervisionada

Outro processo dentro da aprendizagem supervisionada, visa o tratamento de problemas de regressão, que estão relacionados com a previsão de um valor contínuo. Os problemas de regressão estão relacionados com funções que visam a previsão de valores exatos, por exemplo, quando existe uma análise de mercado, em que se pretende prever qual o valor de determinada casa. Os problemas de regressão são mais complexos do que os de classificação e daí precisarem de muitos mais dados [12]–[14].

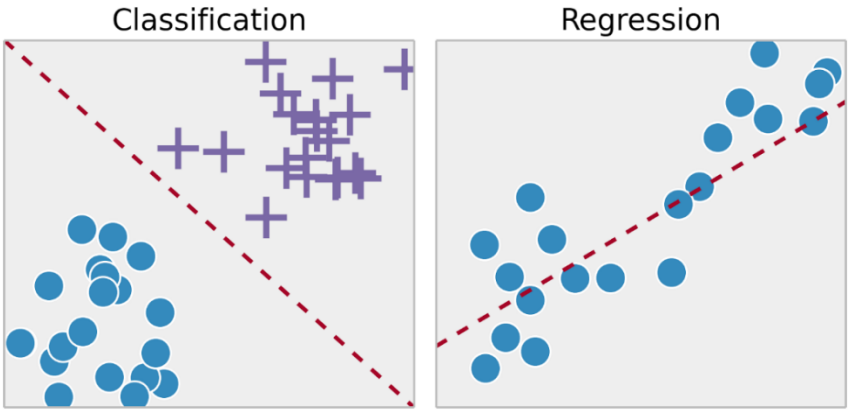


Figura 6 – Diferença entre problemas entre classificação e Regressão

3.1.2 Aprendizagem não supervisionada

Na aprendizagem não supervisionada, os dados de entrada não estão classificados, ou seja, não se conhece a classe a que pertencem. Neste cenário os algoritmos têm de ser capazes de identificar autonomamente as classes nos dados. Este tipo de aprendizagem é similar ao modo de funcionamento do cérebro humano [12]–[14].

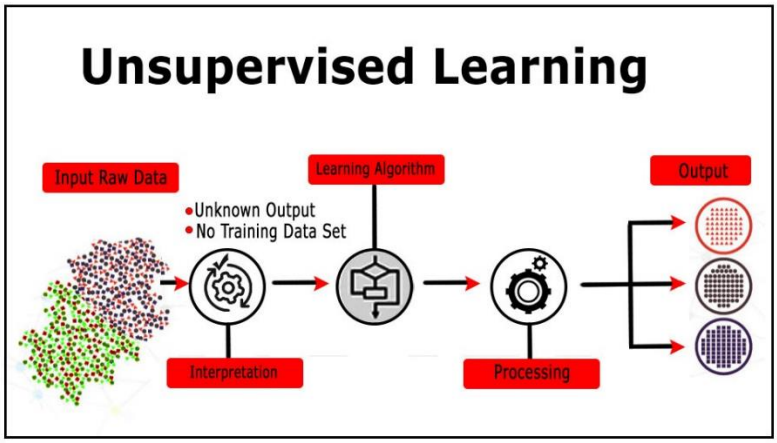


Figura 7 - Aprendizagem não supervisionada[12]

3.1.3 Aprendizagem por reforço

Na aprendizagem por reforço, os dados de entrada são fornecidos com o intuito de obter uma resposta do sistema, avaliar e decidir qual o passo seguinte. Aqui é utilizado o princípio de recompensas e penalizações. Assemelha-se a uma metodologia educacional de uso comum no nosso ensino. São usados, por exemplo na robótica para permitir o controlo dos robôs [12]–[14].

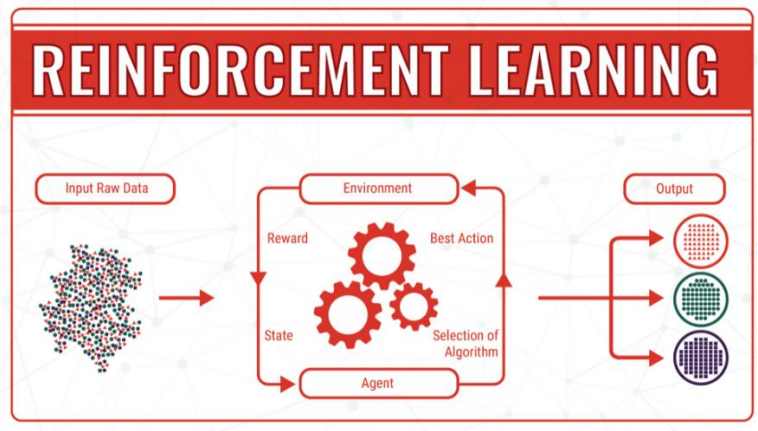


Figura 8 - Aprendizagem por reforço[12]

3.2 Redes Neuronais

Esta secção irá apresentar o estado da arte sobre redes neuronais, iniciando pela apresentação simplificada da história das redes neuronais bem como o modo de funcionamento das mesmas.

3.2.1 História das redes neuronais

O conceito de rede neuronal, já existe há bastante tempo. Em 1943, foi publicado o artigo de um neurobiologista e um matemático [15], que deu início à área da Neuro Computação. Este artigo mostrava que todos os tipos de redes neuronais poderiam computar qualquer função aritmética ou lógica.

Este artigo conseguiu ter impacto, de forma a potenciar mais investigação na área. Norbert Wiener e Von Neumann, também escreveram sobre o tema [16][17] em que sugeria que seria interessante elaborar estudos de computação baseados no funcionamento do cérebro humano [18].

Em 1949 existiu um grande acontecimento na história da Inteligência Artificial, nomeadamente na implementação de redes neuronais artificiais. Neste ano, Donald Hebb, lançou um livro[19], onde implementava a primeira regra de aprendizagem, conhecida como Hebbian Rule, onde demonstrou como a aprendizagem das redes neuronais artificiais é conseguida pelo uso de pesos de entrada dos neurónios, propondo uma teoria para explicar a aprendizagem em neurónio biológico baseada no esforço das ligações sinápticas entre todos os neurónios estimulados. Desse modo, quando dois neurónios, ligados entre si, o primeiro persiste repetidamente em ativar o segundo, então o valor sináptico das suas ligações deve aumentar [18], [20].

Mais tarde em 1958 também existiu um grande acontecimento na área das redes neuronais. Neste ano vários contribuidores, destacando Frank Rosenblatt (fundador) e Charles Wightman, desenvolveram um novo modelo, intitulado de Perceptron[21], que era uma máquina capaz de ser treinada e com capacidade de aprender, classificar e determinar padrões. Este modelo Perceptron, na sua forma mais simples é constituído por uma rede de uma única camada que para um determinado padrão de entrada consegue ajustar os pesos, de forma a produzir uma resposta correta [18], [20].

Pouco depois de Rosenblatt ter desenvolvido o modelo do Perceptron, Bernard Widrow e Marcian Hoff desenvolveram um tipo diferente de elemento de processamento de rede neuronal, chamado ADALINE (ADAptive LInear NEuron), que foi equipado com uma nova regra de aprendizagem. O ADALINE é baseado no modelo neuronal de McCulloch e Pitts, onde ambos utilizam uma única camada (*single-layer*), no entanto enquanto o Perceptron utiliza os rótulos (*labels*) para aprender coeficientes do modelo, o ADALINE utiliza valores de previsão contínua para aprender os coeficientes do modelo, efetivando deste modo uma evolução ao Perceptron, porque conseguia indicar quanto errado ou certo os modelos poderiam estar [18], [20], [22].

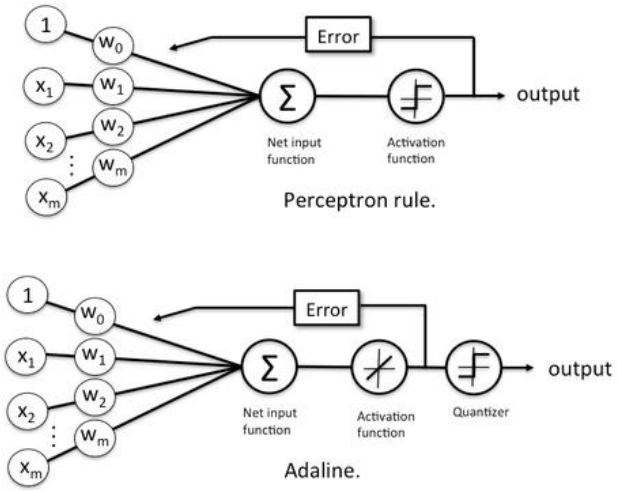


Figura 9 - A evolução Perceptron – Adaline

Mais tarde, em 1969, Minsky e Papet reportaram alguns problemas no modelo do Perceptron, onde defenderam que este modelo apenas resolvia problemas linearmente separáveis, problemas em que a solução pudesse ser obtida dividindo-se o espaço de entrada em duas regiões através de uma reta. Este modelo, não conseguia detetar paridade e simetrias, que são problemas não linearmente separáveis [18], [20].

Esta afirmação poderá ter estado na origem de uma quebra no entusiasmo e estudos produzidos, já que na década de 70, não foram registados grandes acontecimentos de maior relevância [23].

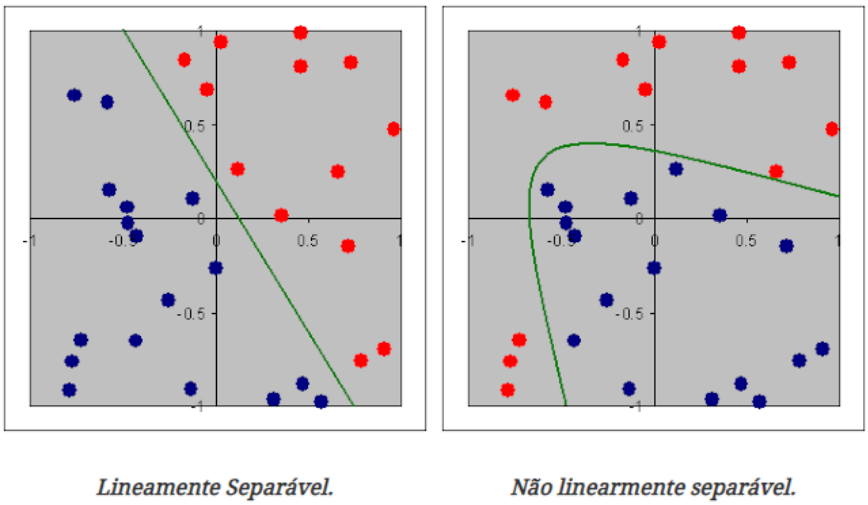


Figura 10 - Grafico representativo de problemas linearmente separáveis e não separáveis

Nos anos 80, este tema voltou a ganhar interesse, quando vários investigadores decidiram investir conhecimento e desenvolver várias propostas, a fim de explorar o desenvolvimento da área da neuro-computação e aplicações de redes neuronais. Em 1983, um físico de renome mundial chamado John Hopfield, escreveu dois artigos sobre redes neuronais e ainda promoveu várias palestras mundiais, criando um novo movimento e interesse na área das redes neuronais [18], [20].

Em 1986, Rumelhart e McClelland, estiveram envolvidos na publicação de livros de PDP (*Parallel Distributed Processing*). Em 1987 foi realizada a primeira conferencia internacional IEEE, sobre redes neuronais, e como consequência, foi formada uma Sociedade Internacional de Redes Neuronais (INNS), que fundou a revista Neural Networks, seguida pela *Neural Computation* e pelo *IEEE Transactions on Neural Networks* [18], [20].

3.2.2 Funcionamento das redes neuronais

Como as redes neuronais devem simular o funcionamento do cérebro humano, devem ter determinadas características, nomeadamente tal como o cérebro têm vários neurónios, as redes neuronais, devem dispor de várias unidades de processamento, ou seja, os neurónios de uma rede neuronal. Outra característica importante é existirem pesos, que são coeficientes associados às ligações entre as unidades de processamento, sendo que o fator de aprendizagem está ligado ao ajuste destes mesmos pesos. O processamento é distribuído e pode ser paralelizado. As redes neuronais são inspiradas no cérebro, mas não são exatamente iguais, segundo a literatura, o cérebro humano consegue processar de forma distribuída, não linearmente e em paralelo, enquanto que nas redes neuronais, o processamento é central, de forma linear e sequencial. O cérebro é quem controla toda a informação, e isso acontece utilizando os neurónios que se ativam através de estímulos, tal como nas redes neuronais, em que através dos inputs, onde para cada input, é atribuído um coeficiente, gerando informação através da soma dos coeficientes. Por fim, há uma função de ativação associada ao neurónio, que utiliza os dados obtidos na função da soma dos coeficientes [24], [25].

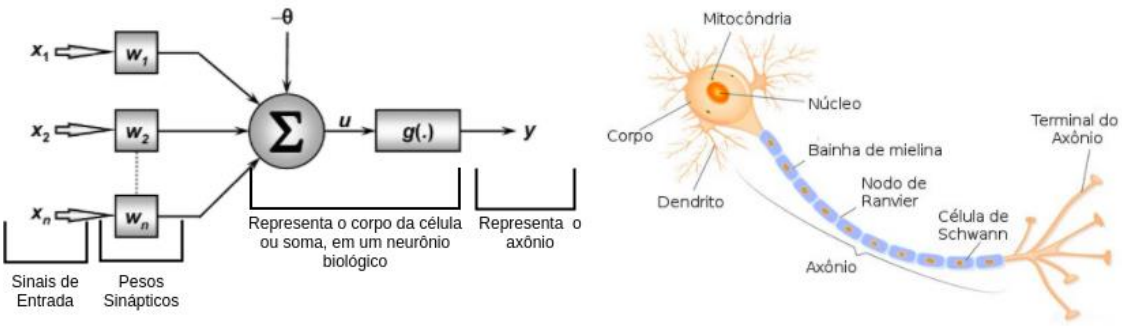


Figura 11 - Redes Neuronais

Dentro do domínio das redes neuronais também é importante referir que existem diversas variantes, sendo as principais CNN (*Convolutional Neural Networks*) e RNN (*Recurrent Neural Networks*).

As RNN são muito utilizadas em processamento de linguagem natural, pois utilizam os inputs anteriores nos cálculos, ou seja, a rede é utilizada para produzir previsões do caractere seguinte, tendo em conta o caractere anterior previsto. Um exemplo muito utilizado para descrever o funcionamento das RNN é quando um manuscrito está a ser analisado, ao analisar uma palavra, a determinada altura é possível prever a próxima letra, pois tem em consideração os elementos anteriores processados, reduzindo as possíveis hipóteses [24], [25].

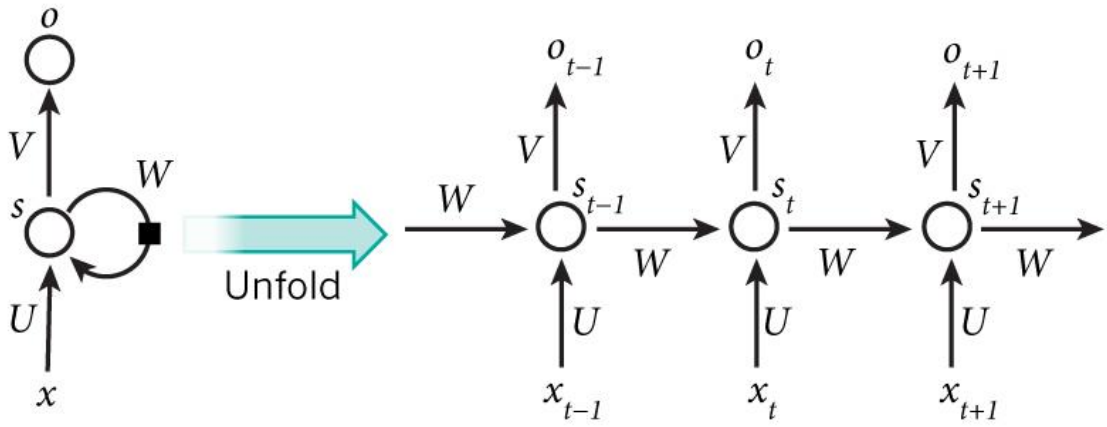


Figura 12 - *Recurrent Neural Network*[26]

As CNN são usadas principalmente em reconhecimento de imagens, tema desta dissertação, pelo que serão em seguida apresentadas.

3.3 CNN

As CNN são bastante utilizadas no campo da computação visual que utiliza *hidden layers*, tipicamente *convolutional layers, pooling layers, fully connected layers* e *normalization layers*. Isto significa que em vez de usar as funções de ativação normais, são utilizadas as funções de convolução e de *pool* como funções de ativação. A convolução opera em dois sinais(1D) ou duas imagens(2D), onde recebe um sinal de entrada e aplica um filtro com o objetivo de produzir o resultado entre a multiplicação dos dois sinais [27].

Pooling é um processo que tem como objetivo reduzir a amostra de uma matriz (representação de uma imagem), reduzindo a sua dimensão permitindo criar sobreposições sobre os recursos contidos nas sub-regiões categorizadas [24], [25].

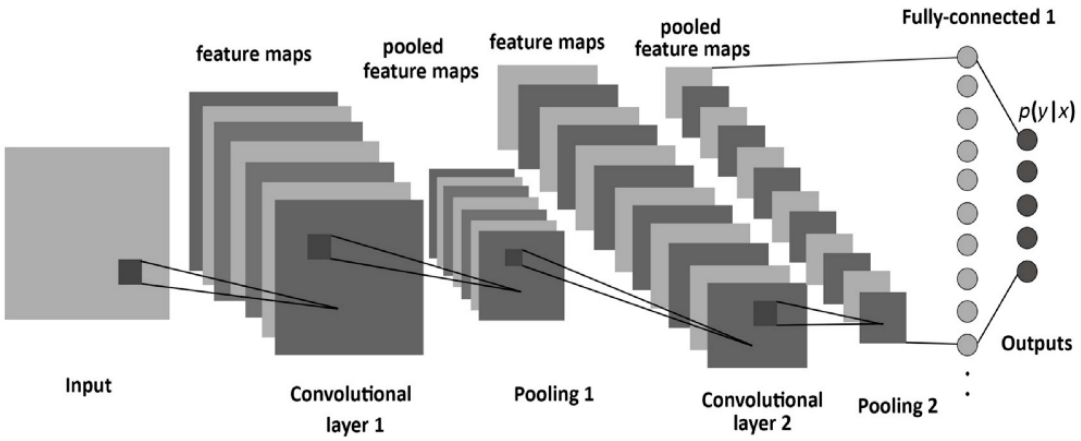


Figura 13 - *Convolutional Neural Networks*[28]

3.3.1 Conceitos chave

Quando se opera com redes neuronais e neste caso as CNN, é importante conhecer alguns termos e conceitos, muito utilizados na área.

3.3.1.1 Overfitting

O *overfitting* é um problema que ocorre nas redes neuronais quando a rede memoriza demasiado um conjunto de dados e não é capaz de generalizar para novos dados, fazendo com que a rede se torne inútil para situações reais já que passa a apenas conseguir reconhecer dados utilizados no treino, mas deixa de ser eficiente no reconhecimento de outros dados, não utilizados no processo de treino.

3.3.1.2 Dropout

O *Dropout* é uma técnica muito importante porque elimina ou congela níveis da rede, em algumas iterações do treino do modelo, para evitar o *overfitting* da rede.

3.3.1.3 Max Pooling

Max pooling é um algoritmo que serve para fazer o *downsapling*, que significa a redução do tamanho do vetor. Neste algoritmo, o vetor é analisado, e processado em vetores de 2x2, onde cria um vetor final com a união do valor mais alto de cada sub-vector de 2x2.

3.3.2 Arquiteturas de Redes Neuronais Convulsionais (CNN)

As CNN são redes neuronais com múltiplas camadas, desenhadas principalmente para reconhecimento e caracterização visual de padrões utilizando por exemplo pixéis das imagens com algum pré-processamento de imagem. Deste modo esta tese utilizará CNN, no entanto, as CNN dispõem de várias arquiteturas possíveis de adotar, destacando LeNet-5, AlexNet, ZFNet, GoogleNet/Inception, VGGNet, ResNet e MobileNet.

Antes de abordar as diferentes arquiteturas, é relevante começar por falar sobre a ImageNet. A ImageNet é uma base de dados com mais de 15 milhões de registos trabalhados para serem utilizados no treino de redes neuronais, contando com várias categorias (*labels*) e cada categoria contém também várias imagens.

Com a ImageNet, em 2010 foi criada também uma nova ferramenta, chamada de ImageNet Large Scale Visual Recognition Challnge (ILSVRC). Esta ferramenta serviu para melhorar o estado da arte de deteções de objetos e classificação de imagens em larga escala [29].

Desta maneira foi possível verificar o estado de desenvolvimento das diferentes arquiteturas.

3.3.2.1 LeNet 5

A LeNet 5 apareceu no primeiro ILSVRC (ILSVRC 2010), desenvolvida por Yann LeCun[30]. Esta arquitetura utilizava inputs de imagem de 32x32 pixéis, que eram colocados num primeiro *layer* de convolução (C1), onde era dividido em várias subimagens criando um *layer* com essas imagens, chamado *pooling* (S2). A terceira etapa do processo era criar uma nova sequencia de convoluções, seguida de nova camada de *pooling*. Por fim existiam redes de pontos ligados entre si, e um deles era o output *layer*. Esta rede neuronal foi usada para reconhecimento de códigos postais, nos postos de distribuição de correios [31][32].

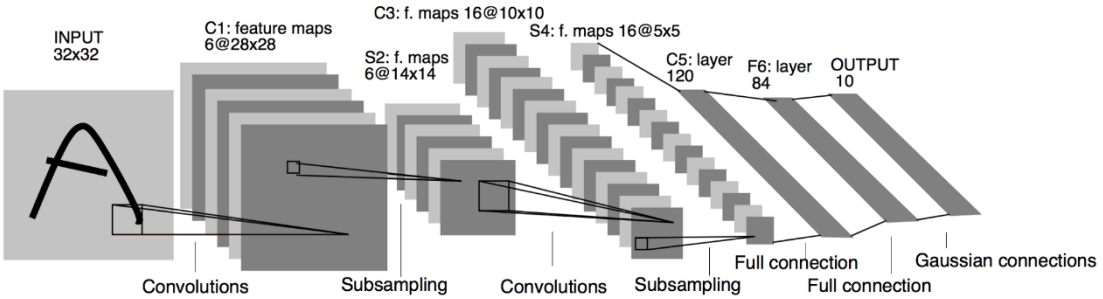


Figura 14 - Arquitetura LeNet5

3.3.2.2 AlexNet

Em 2012, ocorreu o primeiro avanço oficial das arquiteturas CNN. Neste ano, o vencedor da competição (ILSVRC) foi uma arquitetura chamada AlexNet, desenvolvida na universidade de Toronto por Alex Krizhavsky e o professor Jeffry Hinton[33].

Nesta arquitetura, foram introduzidas novas variáveis, *ReLU (Rectified Linear Unit)* e um *Dropout* de 0.5, para combater o *overfitting*. Hoje em dia AlexNet ainda é uma das redes neuronais com mais precisão devido à sua simples estrutura e baixa profundidade [32].

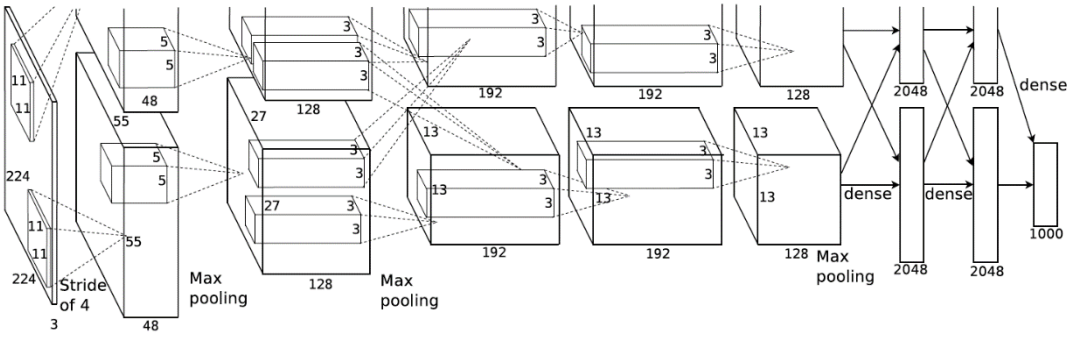


Figura 15 - Arquitetura AlexNet

3.3.2.3 VGGNet

Em 2014, na ImageNet Challenge, apareceu uma nova arquitetura a apresentar bons resultados. Essa arquitetura é VGGNet[34] e foi desenvolvida pela Universidade de Oxford.

Esta CNN destacou-se por ser simples e elegante e conseguir apresentar um resultado de 7.3% de taxa de erro. Existem duas versões, VGG16 e VGG19, sendo que a primeira é composta por 16 *layers*, não contabilizando os *layers* de *pooling*. A VGG19, é composta por 19 *layers*. Para isso foi utilizado um modelo pré-treinado com dois *backends*, Theano e TensorFlow.

Relativamente ao design da arquitetura, esta destaca-se pela profundidade da rede, pois o aumento da profundidade da rede, aconteceu pela adição de mais duas camadas de convolução, adicionando a todas as camadas um filtro de 3x3. O tamanho de imagem padrão neste modelo é de 224x224x3. A imagem passa por uma pilha de *convolutional layers*, filtros entre convulsões, como exemplifica a próxima imagem. Esta CNN apresenta uma particularidade, já que tem aproximadamente 140 milhões de parâmetros, o que pode ser visto como negativo e positivo, podendo ser difícil de configurar, mas é mais personalizável [32] .

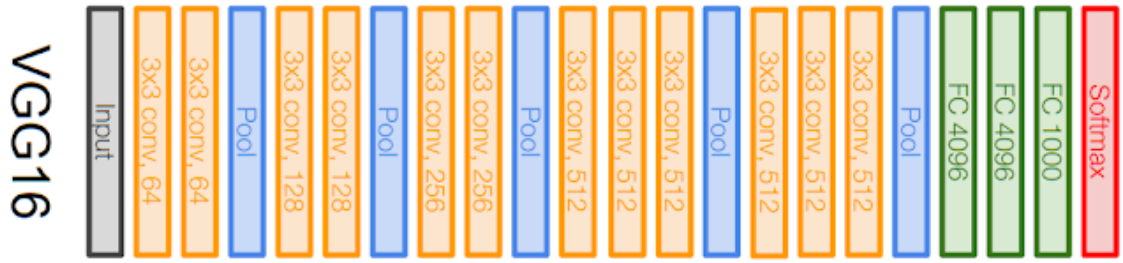


Figura 16 - Arquitetura VGGNet

3.3.2.4 GoogLeNet

Em 2014, também a google lançou a sua própria rede, conhecida como GoogLeNet[27], que relativamente à performance conseguiu ser ligeiramente superior a VGGNet, pois registou 6.7% de taxa de erro, face as 7.3% da VGGNet. O aspeto atrativo desta rede está relacionado com a introdução de um novo conceito, chamado "*inception module*", que reduziu o número de parâmetros para 5 milhões, sensivelmente 12 vezes menos que a AlexNet. Esta rede com 22 *layers* permitiu também um menor consumo de memória e também consumo elétrico [32].

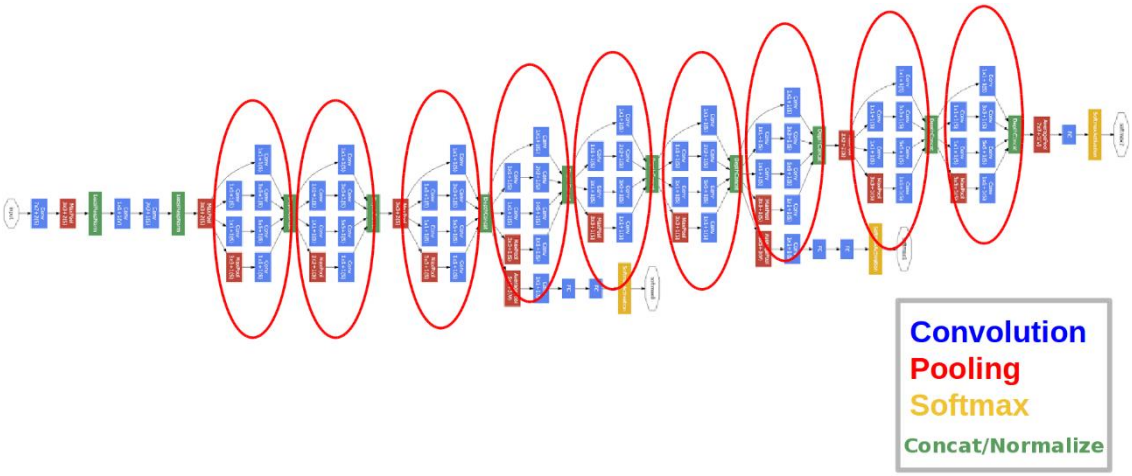


Figura 17 - Arquitetura GoogleNet

3.3.2.5 MobileNet

A MobileNet é uma rede neuronal, desenvolvida pela Google, criada para com uma arquitetura simplificada, que utiliza a profundidade para a criação dos pesos da rede, tornando esta rede mais otimizada para dispositivos onde o processamento não é tão alto, como o caso dos dispositivos moveis. [35]

Segundo a literatura, esta rede neuronal, consegue ter um maior rendimento, devido a relação entre a *accuracy* registada face ao volume da rede neuronal.

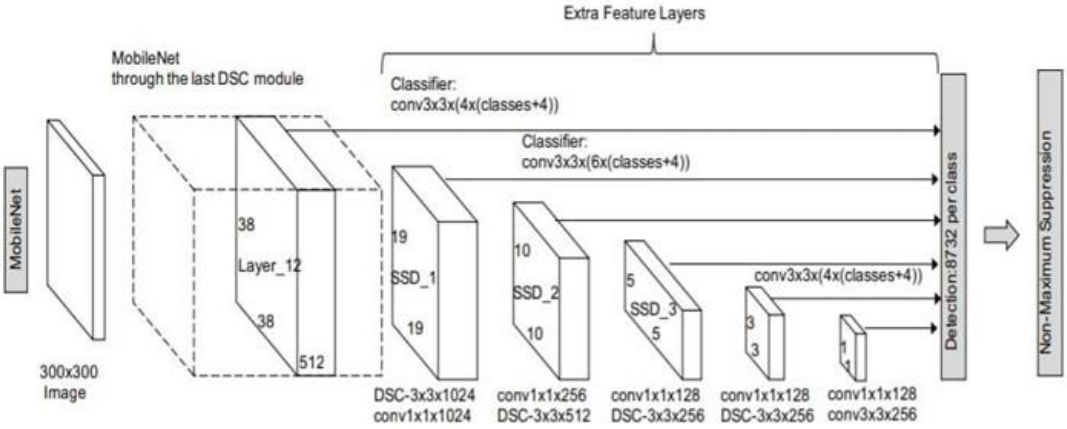


Figura 18 - Arquitectura MobileNet

3.3.3 Estudo comparativo entre arquiteturas

O *transfer learning* é uma metodologia que permite reaproveitar modelos previamente treinados.

A análise elaborada na publicação de análise de arquiteturas *deep learning*[36], pretende demonstrar uma comparação entre vários modelos pré-treinados, avaliando parâmetros como a *accuracy*, complexidade do modelo, utilização da memoria, complexidade computacional e tempo inferido, utilizado como dataset da análise, a Imagenet-1k[37].

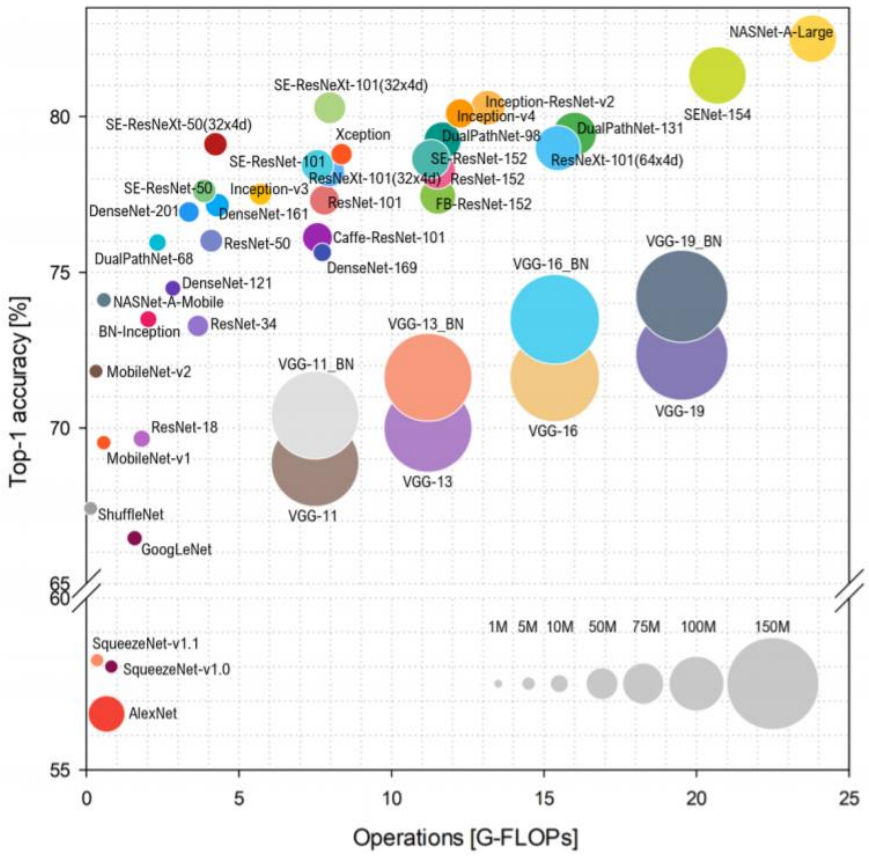


Figura 19 - Análise da complexidade dos modelos

A Figura 19, demonstra o estudo elaborado, onde compara o resultado da *accuracy* no conjunto de dados de validação, em função dos FLOPS (medida da performance do computador) em cada iteração da rede, onde o tamanho de cada bola representa a complexidade do modelo.

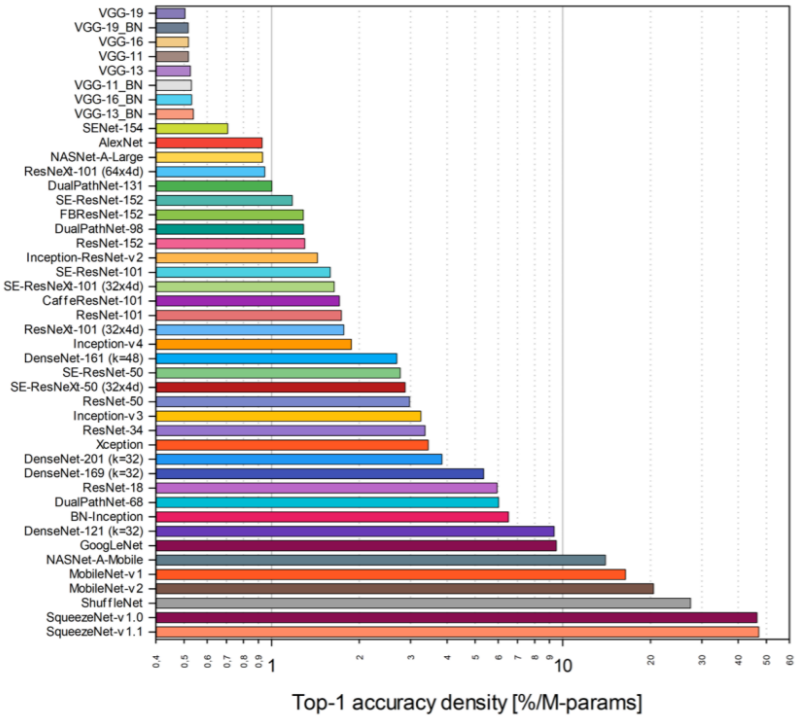


Figura 20 - Análise da eficiência da utilização dos parâmetros

A Figura 20, pretende demonstrar a eficiência com que cada modelo utiliza os parâmetros, que pode ser vista pela divisão entre a *accuracy* obtida e o número de parâmetros.

DNN	1	2	4	8	16	32	64
AlexNet	1.28	0.70	0.48	0.27	0.18	0.14	0.15
BN-Inception	5.79	3.00	1.64	1.10	0.87	0.77	0.71
CaffeResNet-101	8.20	4.82	3.32	2.54	2.27	2.16	2.08
DenseNet-121 (k=32)	8.93	4.41	2.64	1.96	1.64	1.44	1.39
DenseNet-169 (k=32)	13.03	6.72	3.97	2.73	2.14	1.87	1.75
DenseNet-201 (k=32)	17.15	9.25	5.36	3.66	2.84	2.41	2.27
DenseNet-161 (k=48)	15.50	9.10	5.89	4.45	3.66	3.43	3.24
DPN-68	10.68	5.36	3.24	2.47	1.80	1.59	1.52
DPN-98	22.31	13.84	8.97	6.77	5.59	4.96	4.72
DPN-131	29.70	18.29	11.96	9.12	7.57	6.72	6.37
FBResNet-152	14.55	7.79	5.15	4.31	3.96	3.76	3.65
GoogLeNet	4.54	2.44	1.65	1.06	0.86	0.76	0.72
Inception-ResNet-v2	25.94	14.36	8.82	6.43	5.19	4.88	4.59
Inception-v3	10.10	5.70	3.65	2.54	2.05	1.89	1.80
Inception-v4	18.96	10.61	6.53	4.85	4.10	3.77	3.61
MobileNet-v1	2.45	0.89	0.68	0.60	0.55	0.53	0.53
MobileNet-v2	3.34	1.63	0.95	0.78	0.72	0.63	0.61
NASNet-A-Large	32.30	23.00	19.75	18.49	18.11	17.73	17.77
NASNet-A-Mobile	22.36	11.44	5.60	2.81	1.61	1.75	1.51
ResNet-101	8.90	5.16	3.32	2.69	2.42	2.29	2.21
ResNet-152	14.31	7.36	4.68	3.83	3.50	3.30	3.17
ResNet-18	1.79	1.01	0.70	0.56	0.51	0.41	0.38
ResNet-34	3.11	1.80	1.20	0.96	0.82	0.71	0.67
ResNet-50	5.10	2.87	1.99	1.65	1.49	1.37	1.34
ResNeXt-101 (32x4d)	17.05	9.02	6.27	4.62	3.71	3.25	3.11
ResNeXt-101 (64x4d)	21.05	15.54	10.39	7.80	6.39	5.62	5.29
SE-ResNet-101	15.10	9.26	6.17	4.72	4.03	3.62	3.42
SE-ResNet-152	23.43	13.08	8.74	6.55	5.51	5.06	4.85
SE-ResNet-50	8.32	5.16	3.36	2.62	2.22	2.01	2.06
SE-ResNeXt-101 (32x4d)	24.96	13.86	9.16	6.55	5.29	4.53	4.29
SE-ResNeXt-50 (32x4d)	12.06	7.41	5.12	3.64	2.97	3.01	2.56
SENet-154	53.80	30.30	19.32	13.27	10.45	9.41	8.91
ShuffleNet	5.40	2.67	1.37	0.82	0.66	0.59	0.56
SqueezeNet-v1.0	1.53	0.84	0.66	0.59	0.54	0.52	0.53
SqueezeNet-v1.1	1.60	0.77	0.44	0.37	0.32	0.31	0.30
VGG-11	3.57	4.40	2.89	1.56	1.19	1.10	1.13
VGG-11_BN	3.49	4.60	2.99	1.71	1.33	1.24	1.27
VGG-13	3.88	5.03	3.44	2.25	1.83	1.75	1.79
VGG-13_BN	4.40	5.37	3.71	2.42	2.05	1.97	2.00
VGG-16	5.17	5.91	4.01	2.84	2.20	2.12	2.15
VGG-16_BN	5.04	5.95	4.27	3.06	2.45	2.36	2.41
VGG-19	5.50	6.26	4.71	3.29	2.59	2.52	2.50
VGG-19_BN	6.17	6.67	4.86	3.56	2.88	2.74	2.76
Xception	6.44	5.35	4.90	4.47	4.41	4.41	4.36
FPS	>1000	>250	>125	>62.5	>30	>15	>5	<=5
ms	<1	<4	<8	<16	<33	<66	<200	>=200

Figura 21 - Análise da inferência de tempo

A Figura 21, demonstra a inferência de tempo por imagem, utilizando o GPU TITAN Xp, onde é possível aferir o número de imagens processadas por segundo, para os diferentes tamanhos de *batch*.

De acordo com o estudo, relativamente às redes com melhor inferência de treino, são a MobilenetV1, as SequeezeNet e a AlexNet, sendo que em relação a eficiência de parâmetros, das três anteriores, a MobileNet e a SequeezeNet estão no topo do estudo demonstrando-se como duas redes neuronais muito eficientes.

O GitHub[38] é uma plataforma de sistema de controlo de projetos e versões de código desenvolvido para desenvolvedores de software, utilizada mundialmente.

Foi elaborado uma pesquisa na plataforma sobre o número de correspondências para cada uma das três redes neuronais selecionadas: AlexNet, MobileNet e SequeezeNet.

De acordo com os resultados encontrados através das correspondências, a palavra “AlexNet” correspondeu a cerca de 150 mil registos, enquanto que a “MobileNet” correspondeu a mais de 200 mil registos. Por outro lado, a “SequeezeNet” correspondeu a menos de 100 registos.

Sendo esta uma plataforma muito utilizada para desenvolvedores de software, e aliando a informação recolhida no estudo anterior, fica demonstrado um indício forte que a arquitetura da SequeezeNet não é uma arquitetura muito utilizada, enquanto que a "AlexNet e MobileNet oferecem mais garantias quanto à consolidação destas duas arquiteturas para os desenvolvedores de software. Deste modo, quer a MobileNet, quer a AlexNet seriam duas arquiteturas viáveis para serem utilizadas, no entanto, para o trabalho desta dissertação será utilizada a MobileNetV1, que é aquela que oferece mais garantias.

3.4 Tecnologias

Nesta secção apresenta-se as tecnologias envolvidas no domínio das CNN bem como será apresentado um método que auxiliou a seleção da *framework* a utilizar, justificando deste modo a escolha.

3.4.1 Frameworks e bibliotecas

Uma *framework* pode ser vista como um conjunto de ferramentas e bibliotecas, e funciona como uma espécie de esqueleto de uma aplicação, permitindo desenvolvera aplicação através do uso das ferramentas e bibliotecas disponibilizadas na *framework*.

Na área de *Deep Learning*, já existem algumas frameworks, sendo as mais populares: TensorFlow[39]; PyTorch [40]; Caffe [41] e CNTK[42].

Estas três frameworks já dispõem de vários “*model zoos*”, que são coleções de modelos pré-treinados num conjunto de dados [43].

De acordo com a literatura, existem aspetos básicos para a utilização de uma *framework*, nomeadamente a facilidade de instalação, configuração, implementação e usabilidade, no entanto, existe ainda quatro requisitos fundamentais para as ferramentas de *Deep Learning*:

- Possibilidade de trabalhar com tensores, que são objetos matemáticos associados à geometria, e que são compostos por matrizes formadas por funções matemáticas;
- Capacidade para computação gráfica e de otimização
- Ferramentas de autodiferenciação;
- Existência de extensões que permitam trabalhar com BLAS/cuBLAS, que são implementações de sub-rotinas de álgebra linear elementar executadas nos processadores gráficos –GPUs) e cyDNN (bibliotecas que permitem implementação de redes neuronais e que são executadas nos processadores gráficos).

3.4.1.1 Torch/Pytorch

O Torch é uma *framework* computacional, com uma *API* desenvolvida em LUA, que suporta algoritmos de ML. O Pytorch é uma versão do Torch, onde a base é Python, disponibilizada de forma open-source pelo Facebook.

Esta *framework*, segundo a literatura, esta em crescimento desde setembro de 2017, muito devido a ter sido adotado por um portal de cursos de *deep learning* "fast.ai". Desde esse período, o Pytorch começou a ser mais procurado por investigadores de *deep learning*, pois demonstrou que é possível desenvolver de uma forma mais simples arquiteturas complexas [44]–[46].

3.4.1.2 Caffe

Caffe é uma *framework* bastante usada em tratamento de análise à base de imagens, que usou a base de implementação das CNN de Math.Lab, para C e C++. Segundo a literatura Caffe não foi desenhado para trabalhar com aplicações de texto ou som, e utiliza o Python como API.

Atualmente já está disponível uma nova versão, que vem ser a continuação do Caffe, onde também tal como o Pytorch, é utilizada pelo Facebook. Esta nova ferramenta veio melhorar essencialmente em escalabilidade e redução de peso de processamento. É uma ferramenta que oferece uma *API* em Python a utilizar como motor a linguagem C++[45]–[47].

3.4.1.3 Theano

O Theano foi das principais referencias de *frameworks* de *deep learning* que foi escrito em Python e permitia lidar com *arrays* multidimensionais, onde permitia computação gráfica e um bom nível de abstração. No entanto, tinha um grave problema, que era a performance, pois apenas trabalhava como single GPU e modelos grandes, podiam demorar muito tempo a serem compilados. Atualmente o Theano encontra-se em estado desaprovado, depois da equipa que

desenvolvia o sistema ter anunciado que não iriam manter o desenvolvimento da *framework* [20], [45], [46].

3.4.1.4 TensorFlow

TensorFlow é uma *framework open-source*, desenvolvida pela Google Brain Team. É uma *framework* utilizada por algumas empresas, destacando-se Airbnb, Twitter, Snapchat, NVIDIA, Dropbox e a própria Google.

Esta ferramenta é utilizada com propósito de efetuar cálculos numéricos, utilizando gráficos de fluxo de dados (data-flow). Esta ferramenta é o sucessor do anterior sistema da google para treinar redes neuronais, DistBelief. É uma ferramenta que trabalha em larga escala e também em diversos ambientes e também permite a computação em qualquer CPU ou GPU, sendo por isso versátil, já que poderá trabalhar quer num servidor, que num dispositivo móvel.

Segundo a literatura, esta ferramenta apresenta como benefícios a fácil aprendizagem da linguagem; utiliza computação gráfica abstrata e disponibiliza uma ferramenta auxiliar para "*data visualization*". Como principais desvantagens, o Python não é considerado uma linguagem das mais rápidas no mercado, como tal, esta desvantagem também é passada para o TensorFlow; nem tudo no TensorFlow é completamente *open-source* e como ainda é uma ferramenta em fase de crescimento e relativamente recente comparada com outras ferramentas, ainda existe poucos modelos pré-treinados.

Como complemento do TensorFlow, o Keras é uma biblioteca que permite o acesso a uma API intuitiva, baseada no Torch. É uma biblioteca em expansão e muito utilizada com o TensorFlow e Theano [20], [45], [48], [49].

3.4.1.5 Keras

Keras é uma *API* para Redes Neuronais de alto nível, onde é executada usando diversos *backends*, nomeadamente TensorFlow, Theano e CNTK, onde as principais vantagens são a facilidade de utilização, modularidade e extensibilidade, podendo ser executada, quer em CPU, quer em GPU.

É uma *framework* desenvolvida e mantida por Francois Chollet, no entanto, faz parte das equipas do TensorFlow, o que faz com que desse modo, o *backend* preferencial é o TensorFlow.[50]

3.4.1.6 CNTK

CNTK é a *framework Open-source* da Microsoft, que significa Computational Network Toolkit. É uma biblioteca que inclui DNN, RNN e CNN e oferece uma API em Python a funcionar sobre C++ [45], [46].

3.4.1.7 Resumo das ferramentas

A "Tabela 2 - Comparação de Frameworks", apresenta um resumo de varias propriedades das principais *frameworks* para *Deep Learning*[51].

Tabela 2 - Comparação de *Frameworks*

	CAFFE	**CNTK**	**TENSORFLOW/KERAS**	**TORCH/PYTORCH**
Started From	2013	2014	2015	2012/ 218
Main Developers	BVLC	Microsoft	Google	Facebook; Twitter;
Core Languages	C++	C++	C++; Python;	C;Lua;
Suported Languages	C++; Python; MathLab	CLI	C++; Python;	C;Lua; Python;
Parallel Computation	Multi GPU	Multi GPU Multi-node	Multi GPU Multi-node	Multi GPU

3.4.2 Escolha da *framework*

O Analytic Hierarchy Process(AHP) é uma metodologia introduzida por Thomas Saaty em 1980, que serve para auxiliar nas tomadas de decisão. Esta metodologia auxilia a comparação e sintetiza os resultados, facilitando a justificação dos mesmos. Nesse sentido, é oportuno a utilização deste mecanismo para comparar e selecionar a *framework* a ser utilizada nesta tese. Deste modo foram selecionadas quatro das principais *frameworks*, o TensorFlow/Keras, Caffe, Torch e CNTK [52].

Tabela 3 - AHP atribuição de pesos na comparação de *frameworks* AI

	Mercado	***Linguagem***	***Usabilidade***	***Performance***	***Pesos***
Mercado	1,0000	0,6667	0,3333	0,1429	*0.077*
Linguagem	1,5000	1,0000	0,3333	0,1429	*0.094*
Usabilidade	3,0000	3,0000	1,0000	0,5000	*0.262*
Performance	7,0000	7,0000	2,0000	1,0000	*0.566*
Total	*12,5000*	*11,6667*	*3,6667*	*1,7857*	*1*

De acordo com a literatura, foram considerados as seguintes características como pontos mais relevantes na escolha da *framework*: o mercado de utilização da *framework*; a linguagem utilizada na *framework*; a usabilidade associada e a performance.

Os valores atribuídos à linguagem têm em consideração a familiarização do autor com a linguagem associada à *framework*, enquanto que os restantes valores arbitrados, correspondem a um levantamento obtido através da extração de conhecimento de várias referencias [44]–[49], [51], [53].

Os valores atribuídos na tabela significam que das quatro características selecionadas, a que tem maior importância é a performance (0.566), de seguida a usabilidade (0.262), linguagem (0.094) e por fim mercado (0.077).

Tabela 4 - AHP Seleção de *Framework* AI

Framework	*Mercado*	*Linguagem*	*Usabilidade*	*Performance*	*Total*
TensorFlow/Keras	10,0000	7,50	9,0000	8,00	0.272
CNTK	7,0000	8,00	7,0000	9,00	0.265
Torch	8,5000	5,00	6,0000	8,00	0.232
Caffe	6,5000	6,00	6,0000	8,00	0.231
Total	*32,00*	*26,50*	*28,00*	*33,00*	*1*

A tabela a cima, mostra-nos que não há diferenças significativas entre optar por escolher entre a ferramenta da Microsoft ou Google, ou seja, CNTK ou TensorFlow/Keras, pois enquanto a ferramenta da Microsoft obteve um melhor desempenho segundo os *benchmarks* consultados, no entanto a TensorFlow/Keras esta a ser utilizada em várias empresas que servem como fator de influência de mercado. Para esta tese será utilizada a ferramenta TensorFlow/Keras pois esta apresenta o maior score do método AHP, que significa que de acordo com os pesos arbitrados oferece maior grau de confiança para a implementação do sistema de deteção de doenças nas plantas a desenvolver.

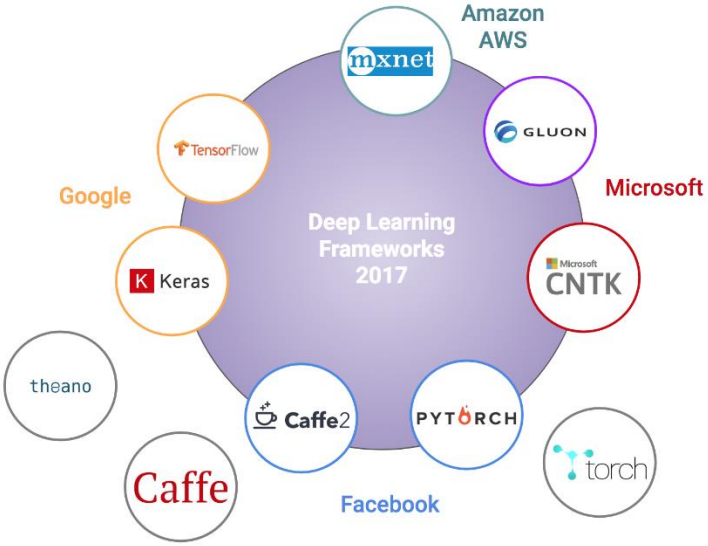


Figura 22 - Frameworks *Deep Learning*

Tal como demonstra o método AHP, a *framework* que melhor que melhor se enquadra e justifica para a realização desta tese é o *TensorFlow*, que será auxiliada também pelo Keras, já que são duas *frameworks* muito compatíveis em que o Keras serve de complemento ao TensorFlow. A utilização desta *framework* do TensorFlow também será uma boa *framework* para ser testada e comparada com os dados recolhidos de um trabalho relacionado [54], desenvolvido em Caffe.

3.5 Soluções existentes

Nesta seção, são apresentadas diversas soluções existentes no mercado, que estejam diretamente ou indiretamente ligados à identificação de doenças em plantas.

3.5.1 Leaf Doctor

Leaf Doctor é uma aplicação[55], que realiza avaliações quantitativas para doenças em folhas de plantas. É uma aplicação onde os utilizadores submetem fotografias de plantas doentes e a aplicação calcula a percentagem de tecido doente. A aplicação utiliza um algoritmo que através do auxílio do utilizador, consegue classificar através das diferenças de cores o cálculo da percentagem da doença. Esta aplicação foi desenvolvida por uma parceria entre a universidade do Hawaii em Manoa, Universidade Cornell e College of Tropical Agriculture and Human Resources (UH-CTAHR).

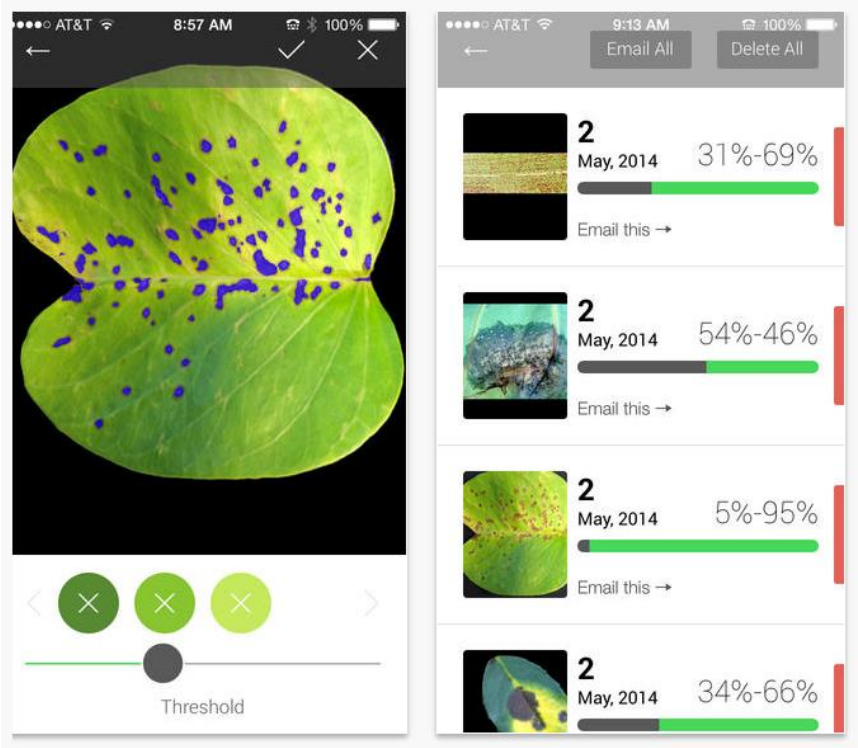


Figura 23 - Aplicação Leaf Doctor

3.5.2 Pl@ntNet

Plant-net é um projeto[56] que já resulta em duas plataformas, uma mobile, outra web, que coleciona imagens, onde as utiliza para efetuar um reconhecimento de plantas utilizando a comparação com imagens catalogadas de uma base de dados botânica. Para cada planta poderá ter algumas informações relativamente à planta, como por exemplo o nome botânico.

É um projeto que envolve um consórcio de cientistas do CIRAD, INRA, INRIA, IRD e da rede Tela Botânica, num projeto financiado por *Agropolis Fondation*.

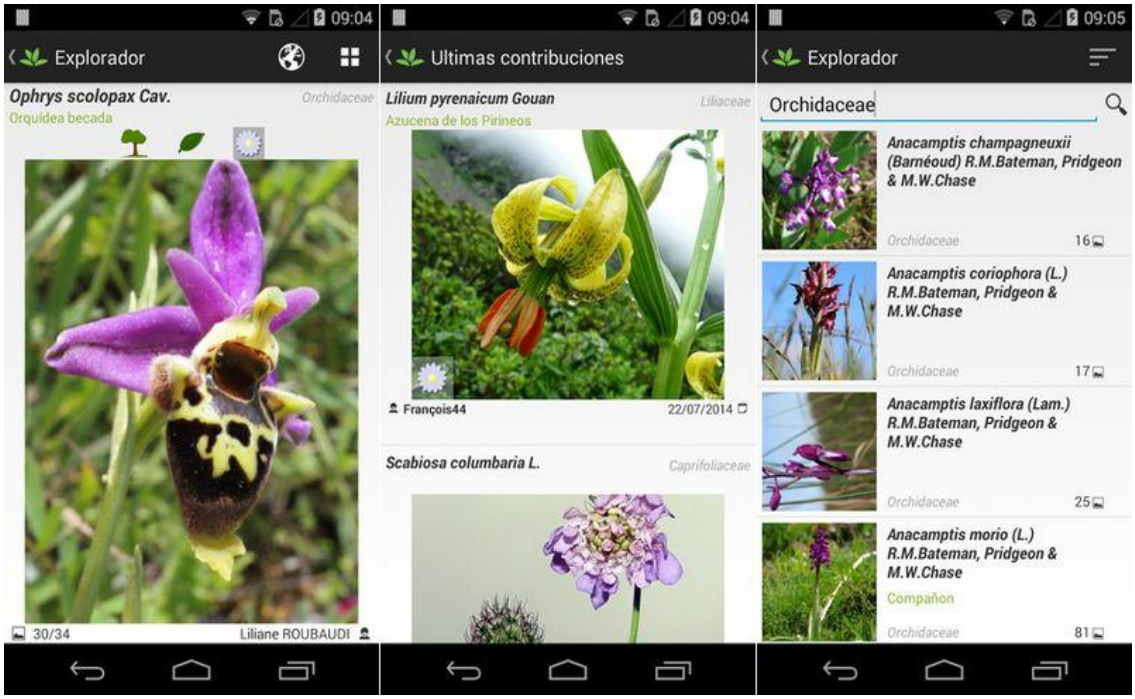


Figura 24 - Aplicação Pl@ntNet

3.5.3 Using Deep *Learning* for Image-Based Plant Disease Detection

Este artigo [54], utiliza o mesmo conjunto de dados selecionado para a exploração nesta tese e apresenta uma *accuracy* de 99.35%. No entanto, ao contrário da solução que irá ser implementada, a solução apresentada neste artigo usa uma versão do Caffe.

Este artigo [48] será muito importante para a comparação de processos e resultados. A Figura 25, apresenta os resultados e os diferentes mecanismos das experiências, que contou com um conjunto de alternativas:

- Arquiteturas:
  - AlexNet;
  - GoogLeNet;
- Mecanismo de treino:
  - *Transfer Learning*;
  - *Training from scratch*;
- Tipo de conjunto de dados:
  - Cores;
  - Gray Scale;
  - Leaf Segmented;
- Diferentes distribuições de treino e teste

		AlexNet		GoogLeNet	
		Transfer learning	Training from scratch	Transfer learning	Training from scratch
Train: 20%, Test: 80%	Color	0.9736 (0.9742, 0.9737, 0.9738)	0.9118 (0.9137, 0.9132, 0.9130)	**0.9820** (0.9824, 0.9821, 0.9821)	0.9430 (0.9440, 0.9431, 0.9429)
	Grayscale	0.9361 (0.9368, 0.9369, 0.9371)	0.8524 (0.8539, 0.8555, 0.8553)	**0.9563** (0.9570, 0.9564, 0.9564)	0.8828 (0.8842, 0.8835, 0.8841)
	Segmented	0.9724 (0.9727, 0.9727, 0.9726)	0.8945 (0.8956, 0.8963, 0.8969)	**0.9808** (0.9810, 0.9808, 0.9808)	0.9377 (0.9388, 0.9380, 0.9380)
Train: 40%, Test: 60%	Color	0.9860 (0.9861, 0.9861, 0.9860)	0.9555 (0.9557, 0.9558, 0.9558)	**0.9914** (0.9914, 0.9914, 0.9914)	0.9729 (0.9731, 0.9729, 0.9729)
	Grayscale	0.9584 (0.9588, 0.9589, 0.9588)	0.9088 (0.9090, 0.9101, 0.9100)	**0.9714** (0.9717, 0.9716, 0.9716)	0.9361 (0.9364, 0.9363, 0.9364)
	Segmented	0.9812 (0.9814, 0.9813, 0.9813)	0.9404 (0.9409, 0.9408, 0.9408)	**0.9896** (0.9896, 0.9896, 0.9898)	0.9643 (0.9647, 0.9642, 0.9642)
Train: 50%, Test: 50%	Color	0.9896 (0.9897, 0.9896, 0.9897)	0.9644 (0.9647, 0.9647, 0.9647)	**0.9916** (0.9916, 0.9916, 0.9916)	0.9772 (0.9774, 0.9773, 0.9773)
	Grayscale	0.9661 (0.9663, 0.9663, 0.9663)	0.9312 (0.9315, 0.9318, 0.9319)	**0.9788** (0.9789, 0.9788, 0.9788)	0.9507 (0.9510, 0.9507, 0.9509)
	Segmented	0.9867 (0.9868, 0.9868, 0.9869)	0.9551 (0.9552, 0.9555, 0.9556)	**0.9909** (0.9910, 0.9910, 0.9910)	0.9720 (0.9721, 0.9721, 0.9722)
Train: 60%, Test: 40%	Color	0.9907 (0.9908, 0.9908, 0.9907)	0.9724 (0.9725, 0.9725, 0.9725)	**0.9924** (0.9924, 0.9924, 0.9924)	0.9824 (0.9825, 0.9824, 0.9824)
	Grayscale	0.9686 (0.9689, 0.9688, 0.9688)	0.9388 (0.9396, 0.9395, 0.9391)	**0.9785** (0.9789, 0.9786, 0.9787)	0.9547 (0.9554, 0.9548, 0.9551)
	Segmented	0.9855 (0.9856, 0.9856, 0.9856)	0.9595 (0.9597, 0.9597, 0.9596)	**0.9905** (0.9906, 0.9906, 0.9906)	0.9740 (0.9743, 0.9740, 0.9745)
Train: 80%, Test: 20%	Color	**0.9927** (0.9928, 0.9927, 0.9928)	**0.9782** (0.9786, 0.9782, 0.9782)	**0.9934** (0.9935, 0.9935, 0.9935)	**0.9836** (0.9839, 0.9837, 0.9837)
	Grayscale	**0.9726** (0.9728, 0.9727, 0.9725)	**0.9449** (0.9451, 0.9454, 0.9452)	**0.9800** (0.9804, 0.9801, 0.9798)	**0.9621** (0.9624, 0.9621, 0.9621)
	Segmented	**0.9891** (0.9893, 0.9891, 0.9892)	**0.9722** (0.9725, 0.9724, 0.9723)	**0.9925** (0.9925, 0.9925, 0.9924)	**0.9824** (0.9827, 0.9824, 0.9822)

Figura 25 - Resultados apresentados no artigo de deteção de doenças em plantas

Dos resultados disponibilizados (Figura 25), foi possível apurar que para cada conjunto de treino e teste, a opção com melhor *accuracy* foi em todos os casos a utilização de imagens com cores e que os resultados obtidos com *transfer learning* obtiveram também uma *accuracy* superior aos obtidos com o treino completo.

De modo geral, ficou também registado que a GoogleNet, obteve valores de *accuracy* superiores aos registados na rede AlexNet.

3.5.4 Leafweb

Leafweb[57] é um projeto desenvolvido com o auxílio da *framework* Pytorch, que disponibiliza protótipo[58] em formato de um *website* onde é possível submeter uma imagem e obter uma classificação para essa imagem.

Nos detalhes deste projeto é possível verificar que utilizam o conjunto de dados disponibilizados pela PlantVillage, onde também aplicam a técnica de *Transfer Learning*, utilizando a rede ResNet101, no entanto não foi possível apurar qual o valor da *accuracy* do projeto.

De acordo com a análise elaborada, este protótipo apenas disponibiliza previsões para classes contidas no conjunto de dados PlantVillage, assim sendo, ao classificar uma imagem externa ao

sistema como por exemplo um animal, o resultado obtido irá ser também uma das categorias existentes no conjunto de dados, como por exemplo, uma doença.

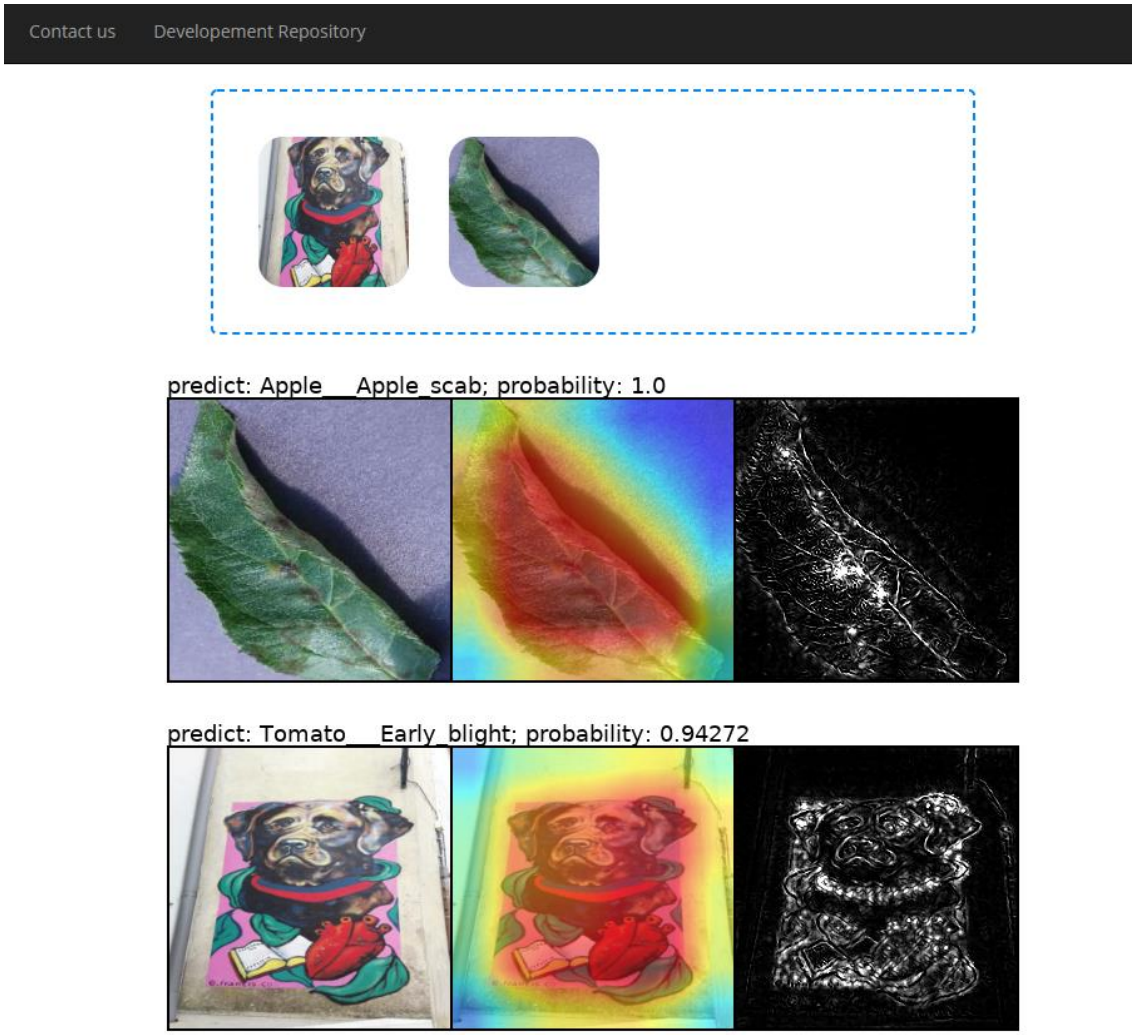


Figura 26 - Classificação de imagens utilizando Leafweb

4 Análise e Design

Este capítulo contém informação sobre a visão do sistema, do ponto de vista estrutural, analisando as funcionalidades pretendidas, a arquitetura e alternativas à arquitetura do sistema.

4.1 Funcionalidades

No sistema foram identificados três papéis distintos, o gestor de conteúdos do sistema, o utilizador comum e o utilizador registado.

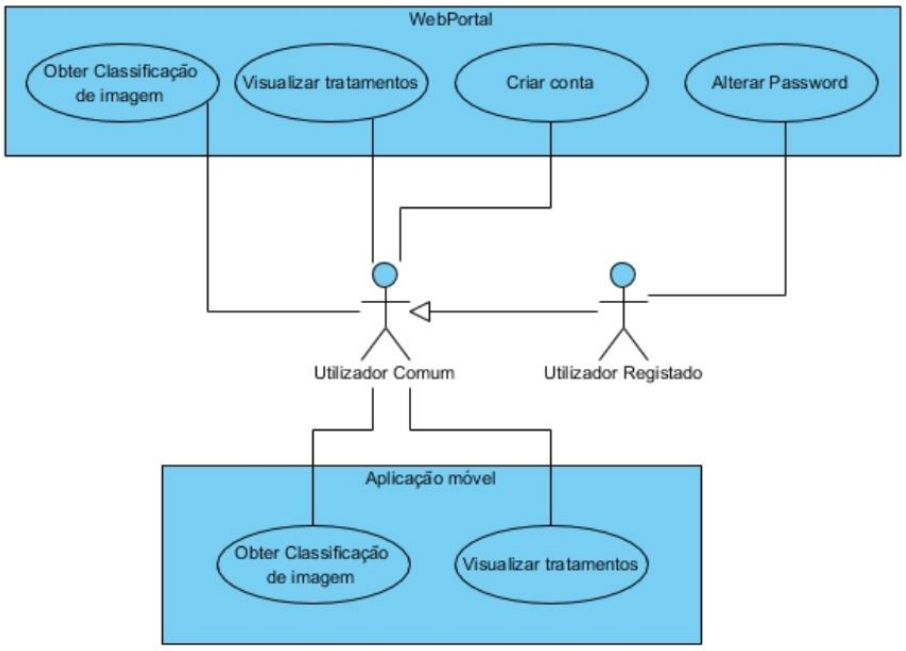


Figura 27 - Funcionalidades de utilizador comum e utilizador registado

O utilizador comum deverá ter acesso as funcionalidades de criação de conta no PlantAiPortal, obtenção de classificação de uma imagem, mais concretamente da doença associada à imagem e a visualização de tratamentos associados as doenças presentes neste sistema. Estas duas funcionalidades coexistiram quer na aplicação móvel, quer no PlantAiPortal. O Utilizador registado é uma evolução do utilizador comum, pois passa a pertencer ao próprio sistema PlantAI, no entanto, a única funcionalidade adicional que terá no sistema é a possibilidade de alteração de password. Este papel poderia ser suprimido face à escassa falta de funcionalidades, no entanto, fica desde já contemplado pois numa eventual evolução do sistema poderá ser introduzido novas funcionalidade uteis nomeadamente de contribuição para o sistema e visualização de histórico.

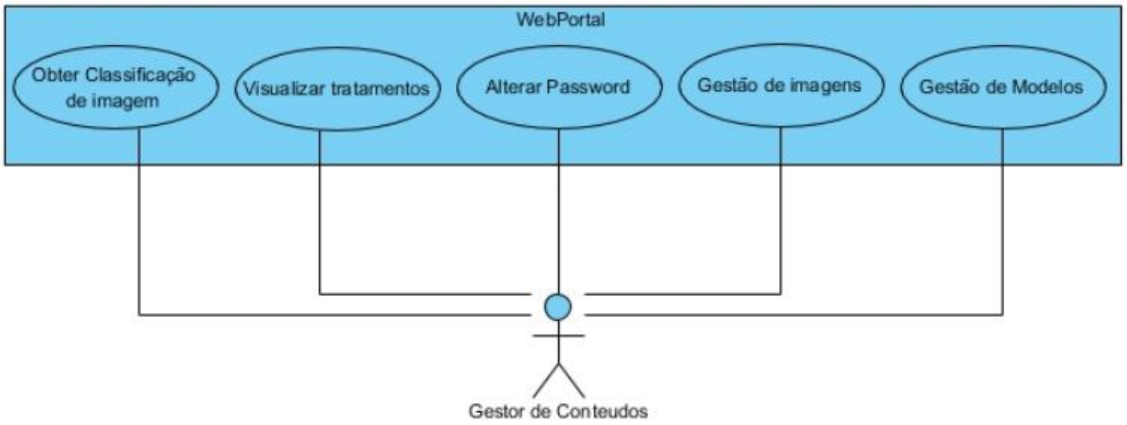


Figura 28 - Funcionalidade do gestor de conteúdos

O gestor de conteúdos, é um papel que apenas está presente no *WebPortal* PlantAI. Este utilizador é necessário para gerir o sistema. Além das funcionalidades de alterar password, obter classificação e visualizar tratamentos, as duas distintas funcionalidades são a gestão de imagens e modelos. A primeira está relacionada com a caracterização das imagens, podendo manipular a classificação de uma imagem bem como excluir essa imagem do conjunto de imagens envolvido nos processos de treino de cada modelo.

A gestão de modelos, é uma funcionalidade que permitirá ao gestor, designar qual o modelo a ser utilizado, utilizando as informações recolhidas no processo de treino do mesmo.

4.2 Arquitetura do sistema

Para o desenho da solução foram consideradas várias possibilidades, em que todas as alternativas foram desenhadas em função de três componentes: um componente de reconhecimento, uma aplicação *mobile*, para demonstrar os resultados conseguidos com a criação do componente de reconhecimento, e um portal para gerir o desenvolvimento do sistema.

4.2.1 Análise de possíveis abordagens

Nesta secção, pretende-se fazer uma avaliação das abordagens possíveis para corresponder aos requisitos projetados para o sistema PlantAI.

4.2.1.1 Solução "A"

A proposta "A" de arquitetura para o sistema consiste em criar um sistema dividido por três áreas distintas, sendo uma delas dedicado a parte AI do sistema, outra dedicado a servir toda a área .NET e por último a área de cliente.

A área de AI, deverá ser um ambiente criado de forma a otimizar o desenvolvimento de Python. Deste modo, foi desenhado uma *API Flask*, que permite servir de ponto de comunicação do sistema AI.

Nesta arquitetura, a API deverá ter como funcionalidades a classificação de imagem e a alteração do modelo de classificação.

Ambas as funcionalidades requerem a utilização de modelos de classificação de imagem previamente treinados. A funcionalidade de classificação de imagem, necessita ainda de um componente dedicado ao pré-processamento de imagem.

O último componente da área de *AI* é o processo de treino. Sendo o processo de treino um componente, esta arquitetura permite que a interação com este componente ocorra de duas formas distintas, a primeira utilizando a *API* como iniciador do processo e a segunda utilizando o agendamento de tarefas. A primeira forma permite um maior poder de controlo já que o início de treino poderá ser a pedido, no entanto, para se treinar um modelo é necessário um grande poder de processamento, o que num sistema a pedido poderia colocar em causa o a disponibilidade do ambiente. A segunda forma poderá garantir um melhor desempenho já que poderá ser efetuada uma orquestração de tarefas para períodos com menor utilização, reduzindo o problema previamente mencionado.

Na solução "A", a base de dados situar-se-á no sector .NET, e estará ligada a um componente chamado PlantAiCore. Este componente servirá para reutilizar toda a logica comum entre a API (.NET) e o Portal.

Ainda neste sector irá existir uma *API* em .NET, que poderá permitir a conexão de diversos sistemas externos, e o Portal PlantAI com as funcionalidades descritas na seção 4.1.

Um aspeto essencial neste sistema é a comunicação da *API* .NET a Flask *API*, bem como do processo de treino a *API* .NET. Esta comunicação, permite que possa ser adicionada uma camada de segurança capaz de isolar o sistema AI do exterior.

Por último, a aplicação android, que fica abstraída de qualquer tipo de processamento ficando apenas dependente da internet para ligar a *API* .NET, sendo que todas as funcionalidades propagam dessa mesma ligação.

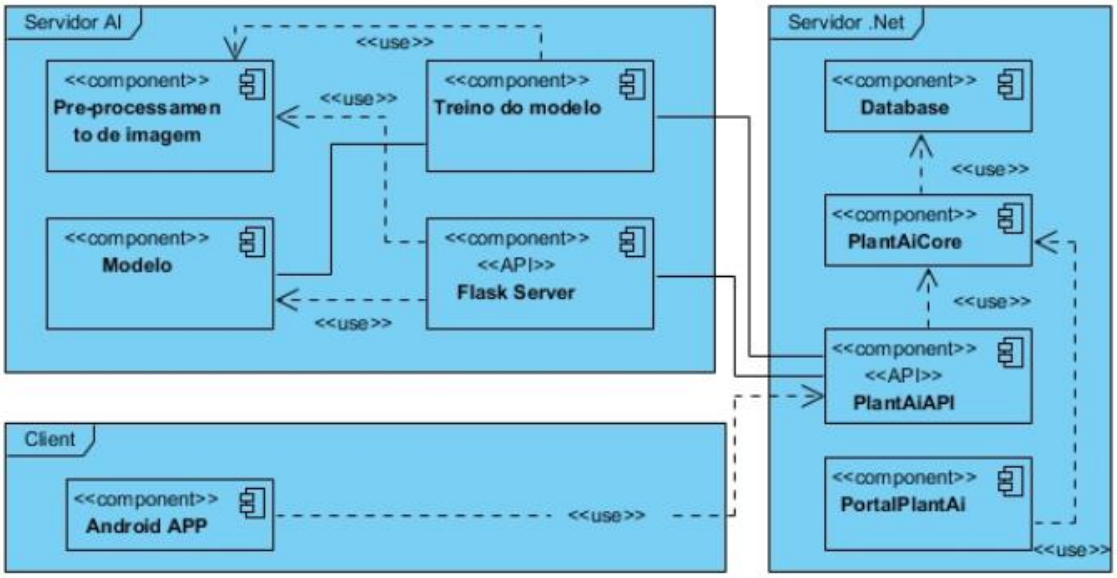


Figura 29 - Arquitetura solução "A"

4.2.1.2 Solução “B”

A solução “B” é em parte muito semelhante à Solução “A”, existindo igualmente três áreas distintas: servidor AI, servidor .NET e lado de Cliente, sendo que a área de servidor AI e servidor .NET são completamente idênticos face à solução anterior.

A diferença entre estas duas soluções está na inclusão de dois componentes na parte de cliente: um modelo e um componente de pré-processamento.

A inclusão destes dois componentes permite que parte das funcionalidades passem a estar associados ao cliente, aliviando o processamento de classificação de imagem do servidor de *AI*, dado que o dispositivo móvel também poderá efetuar esse mesmo processamento.

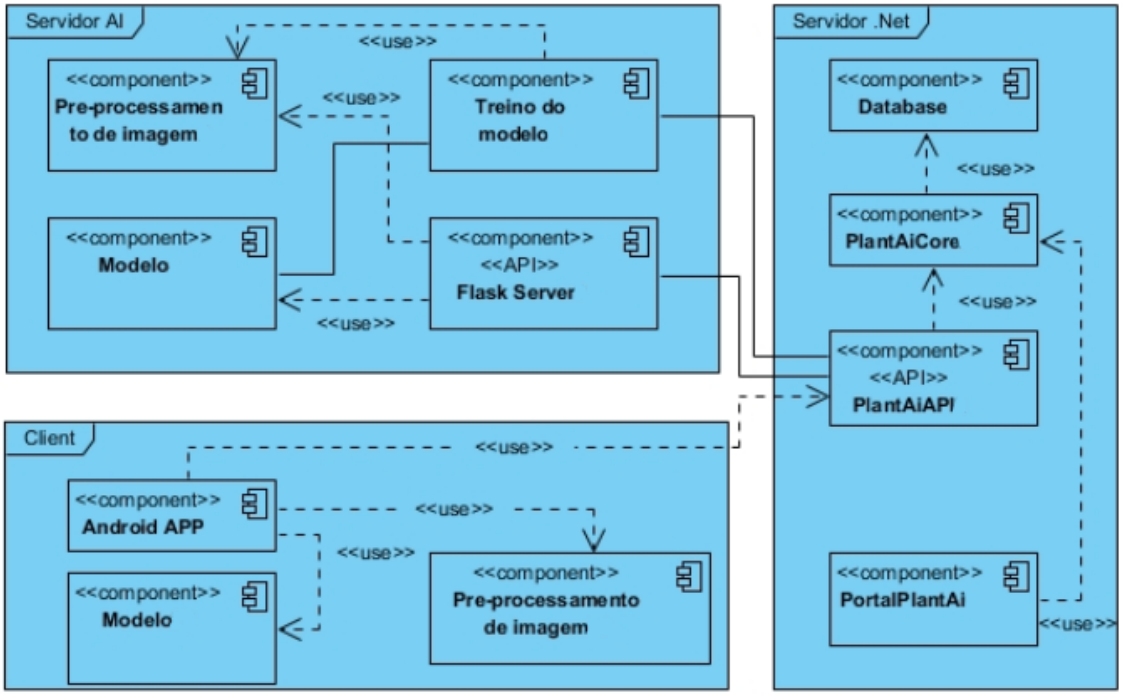


Figura 30 - Arquitetura solução "B"

4.2.2 Análise das soluções

As duas soluções estudadas permitem a resposta a todas as funcionalidades descritas na secção 4.1, sendo que a solução “A” tem como principal desvantagem o requisito de internet para realizar a funcionalidade mais básica do sistema PlantAI que é detetar uma doença numa planta.

A desvantagem da solução “B” é o incremento de complexidade no sistema, já que o sistema necessitará de dois componentes de pré-processamento, e dois componentes de classificação de uma imagem. Outra desvantagem possível nesta abordagem é a dependência de modelos, que são treinados noutro ambiente, sendo que para atualizar o modelo implica atualizar o sistema.

Atendendo a que nos dias de hoje, o acesso à internet é bastante abrangente, e face à desvantagem do aumento de atualizações à aplicação, utilizando a abordagem da solução “B”, a solução A será a abordagem utilizada para o desenvolvimento o protótipo desta dissertação.

5 Desenvolvimento da solução - PlantAI

No quinto capítulo desta dissertação de mestrado, pretende-se apresentar de forma sucinta todas as etapas que compuseram o desenvolvimento do PlantAI, solução desenvolvida nesse âmbito.

Nesta secção estão comtempladas as várias tarefas realizadas para o desenvolvimento desta solução, nomeadamente: as metodologias de desenvolvimento, onde demonstra a utilização de controlo de versões, o *hardware* disponível para a execução de experiências e as bibliotecas utilizadas.

Ainda nesta secção irá contemplar informações relativas à base de dados, o desenvolvimento das duas *API* necessárias na arquitetura do sistema, as duas aplicações desenvolvidas para interagir com o sistema, um em formato de aplicação Android e outra uma aplicação *web* denominado de PlantAiPortal.

Esta secção irá ainda conter o principal elemento do trabalho realizado no decorrer desta dissertação, a elaboração de um modelo de classificação de imagem, que se desdobra em várias matérias, a recolha e pré-processamento de dados, o treino do modelo e a utilização de um modelo previamente treinado para classificar imagens.

Para finalizar a secção de desenvolvimento, ira ser demonstrado a proposta desenvolvida para manter um modelo de crescimento automático.

5.1 Controlo de versões

O controlo de versões é um requisito importante na gestão de um projeto. Independentemente da constituição de uma equipa de trabalho, as boas práticas sugerem a utilização de repositórios de código, que permitem acima de tudo um controlo que garante que a integridade

de todos os ficheiros e ainda o seu controlo de versões, registando as alterações efetuadas ao projeto, potenciando ações importantes como reversões. [59]

Nesse sentido, uma das etapas iniciais foi a criação de um repositório, neste caso recorrendo ao GitHub, que permite gratuitamente a criação de repositórios.

O repositório está configurado de forma privada, dado que o mesmo se destina ao propósito de auxiliar o trabalho desenvolvido no âmbito desta dissertação, mas que futuramente poderá ser disponibilizado através do endereço https://github.com/1101516DanielBento/PlantAI.

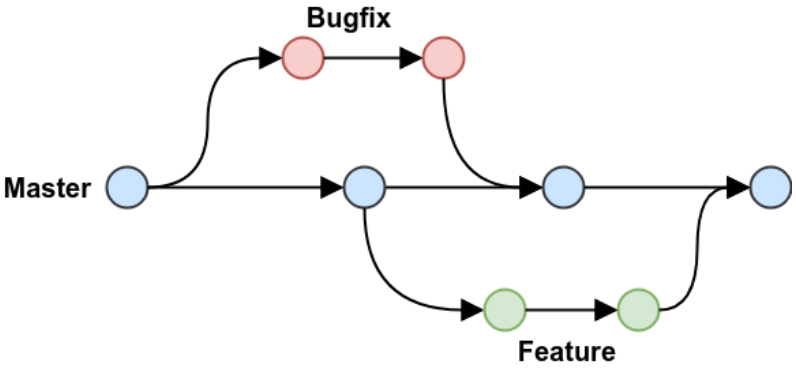


Figura 31 - Controlo de versões[60]

5.2 Hardware

Um dos requisitos principais que permitem construir este modelo, recorrendo a técnicas *Deep Learning*, prende-se com o facto de que é necessário dispor de grande poder de processamento. Um dos motivos principais que justificam tal afirmação é que os algoritmos de técnicas relacionadas com *Machine Learning*, necessitam de grandes quantidades de dados, que estão diretamente ligados ao custo operacional associado para trabalhar com essas grandes quantidades de informação.

Para o processamento deste tipo de tarefas, é possível atualmente utilizar dois tipos de unidades de processamento: as *Graphics Process Units* (GPU) e as *Central Processing Units*(CPU). A principal vantagem das *Graphics Process Units* comparativamente com as *Central Processing Units* centra-se no seu desempenho em desenvolver processamento em paralelo, tornando o mesmo mais célere.

Para a realização do projeto, foi utilizado o modo CPU (limitação de recursos), cujo *hardware* é um Intel Core I7-4700MQ.

5.3 Bibliotecas utilizadas

As bibliotecas utilizadas no sistema PlantAI é um componente essencial para a demonstração e compreensão do trabalho elaborado, sendo que as bibliotecas utilizadas no ambiente Python podem ser consultadas no anexo III.

Para a elaboração deste trabalho foi necessário utilizar o Keras[50] e TensorFlow[61], que já foram explicados nas secções 3.4.1.5 e 3.4.1.4.

A biblioteca NumPy [62] é uma biblioteca de Python que é utilizada para elaborar cálculos em *arrays* multidimensionais, fornecendo um conjunto de funções e operações para auxiliar em cálculos.

Scikit-Learn[63][64] é uma biblioteca de *machine learning* que permite desenvolver treinos, validações de forma simples e com grande abstração

A biblioteca H5py[65] permite o armazenamento de imagens e dados de forma binaria, facilitando o arquivamento. A utilização desta biblioteca permitiu exportar os modelos de classificação para utilizações posteriores.

MatPlotLib[66] é uma biblioteca importante para esta dissertação, devido ao facto de ser uma biblioteca que consegue gerar gráficos utilizando *arrays*. Esta biblioteca permite a geração gráfica de matrizes de confusão facilitando a análise das matrizes obtidas na validação de modelos.

O Pillow[67] é uma biblioteca que deriva da biblioteca PIL que é uma biblioteca de imagem em Python. Esta biblioteca permite a utilização de um conjunto de funções relacionadas com processamento de imagem.

A biblioteca Pandas[68] é uma biblioteca muito utilizada na análise de dados, que auxilia as tarefas de *machine learning* em situações de manipulação, leitura e visualização de dados.

5.4 Base de dados

A base de dados, também é um aspecto importante a referir nesta dissertação, que terá como papel o armazenamento de informação, relativamente a doenças, tratamentos bem como irá conter o repositório de imagens, que servirá para possibilitar o desenvolvimento do treino do modelo, já que ao classificar uma imagem, é possível definir que a nova imagem irá ser incluída para ser disponibilizada no próximo treino do modelo.

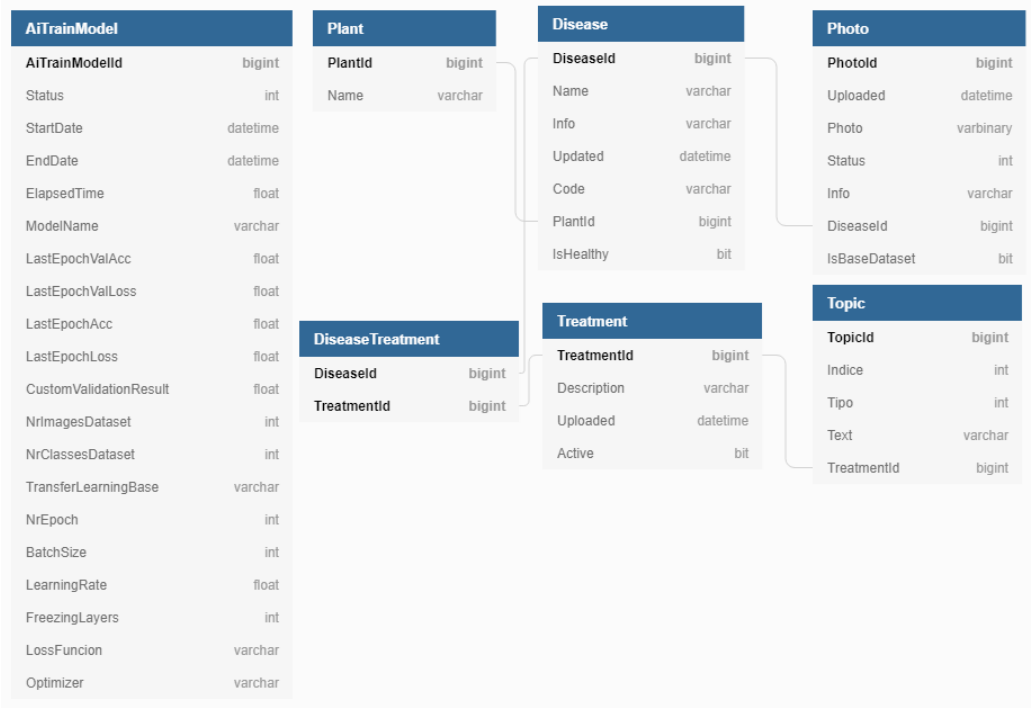


Figura 32 - Modelo de dados

O modelo de dados desenvolvido é simples uma vez que cumpre unicamente com os requisitos propostos nesta dissertação, sendo composto por sete tabelas: AiTrainModel; Plant; Disease; Photo; Treatment; Topic; DiseaseTreatment.

Além das tabelas apresentadas, ainda existem as tabelas necessárias para autenticação no PlantAiPortal desenvolvido, utilizando ASP.NET Identity.

5.5 APIs desenvolvidas

Uma *API* é um conjunto de funções e procedimentos que permite que aplicações externas consigam transferir dados, sendo que o mais comum nos dias de hoje é a utilização do formato JSON e o protocolo utilizado é o HTTP, que é o protocolo padrão e universal para comunicar em aplicações de diversos sistemas distintos, como Java, .Net, Python, entre outros...

O desenvolvimento de uma API foi um dos pontos essenciais de toda a arquitetura do sistema desenvolvido, que conta com duas *API* distintas, uma para interagir com toda parte de AI desenvolvida em Python e outra destinada a servir os dados para serviços externos.

A primeira API foi desenvolvida em Python e utiliza a biblioteca FLASK e tem como principal responsabilidade servir para interação com toda a *AI* do sistema.

Para isso foram expostos dois métodos: alteração do modelo de treino em execução e classificação de uma imagem, em que cada um executa funções diferentes, de forma a obter a funcionalidade pretendida.

A outra API foi desenvolvida em .NetCore 2.1 e tem como principal função servir qualquer tipo de aplicação externa.

Esta aplicação também contém uma ligação com a *API* de *AI*, onde interage com o modelo para a função que retorna a classificação de uma imagem.

5.5.1 .Net Core API

A *API* desenvolvida em .Net Core tem como principal função fazer a ligação entre os dados, aplicação *web*, aplicação mobile e ainda serve como intermediário de toda a parte *AI* do sistema.

Nesse sentido, a *API* está distribuída em 5 controladores, cada um com uma única responsabilidade associada, sendo eles (anexo I):

- AiCore - responsável por devolver a classificação de uma imagem
- Disease – responsável pela gestão de dados relativamente a doenças de plantas;
- Keras – responsável pelas funções associadas ao registo e levantamento de informações dos modelos desenvolvidos em Keras.
- Plant - responsável pela gestão de dados relativamente a plantas;
- Treatment - responsável pela gestão dos tratamentos relativamente a plantas.

5.5.2 Flask API

A *API* desenvolvida em Python, tem como principal funcionalidade servir com as funcionalidades desenvolvidas em Python, como por exemplo, o treino do modelo; alteração do modelo em vigor; classificação de imagem (anexo II).

O desenvolvimento das funcionalidades distribuído sob forma de *API* permite uma vez mais, efetuar a ligação entre o ambiente (.Net e Python) utilizando o protocolo HTTP.

5.6 Xamarin App

A criação de uma aplicação compatível com Android também era um dos requisitos importantes, pois é uma das primeiras camadas de interação entre o modelo AI com o utilizador.

A aplicação Xamarin, permite a criação de uma camada de core, que será utilizada por todos os sistemas desenvolvidos no projeto. Para esta dissertação, foi criada apenas uma versão em

Android, no entanto devido à utilização do Xamarin, poderá no futuro ser desenvolvida uma nova aplicação, para IOS.

Figura 33 - Inicio da aplicação

Relativamente às funcionalidades implementadas na aplicação móvel, é possível obter informação de tratamentos e classificar imagens.

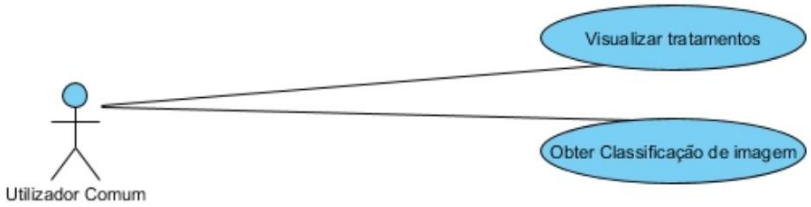


Figura 34 - diagrama de casos de uso para a aplicação móvel

Todos estes processos, requerem internet, visto que cada funcionalidade necessita de obter dados do servidor, sendo que para isso comunica com a API.

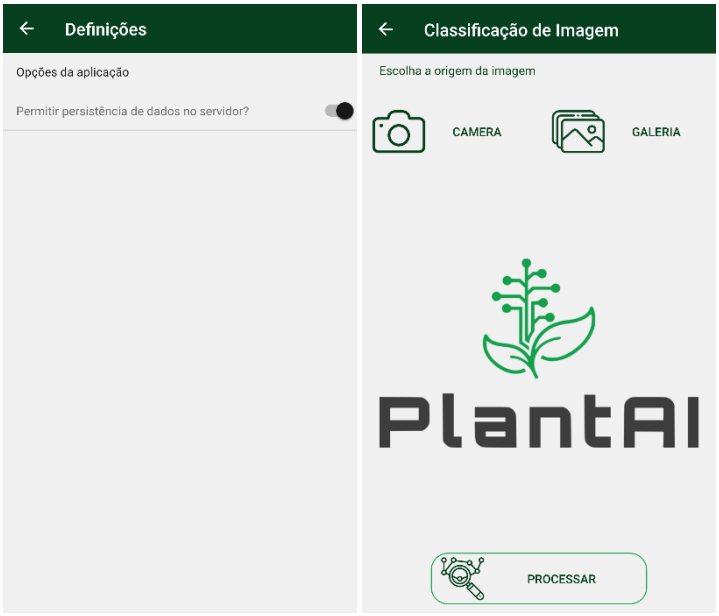


Figura 35 - *Layout* de definições e opção de classificar imagem

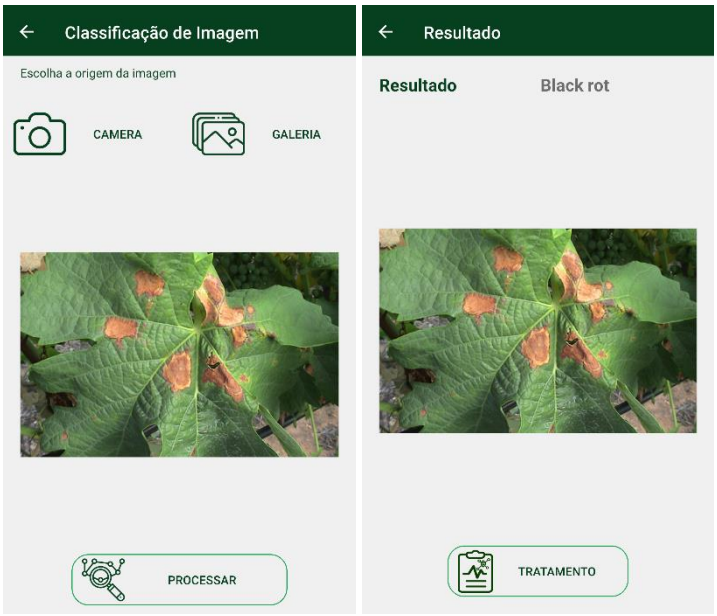


Figura 36 - *Layout* de seleção de imagem e resultado da classificação

Figura 37 - *Layout* de opção de tratamento

5.7 Aplicação Web .Net Core - PlantAiPortal

A aplicação móvel foi projetada para ser destinada aos utilizadores do sistema, de forma a cumprir com os casos de uso pretendidos.

Adicionalmente foi desenvolvido uma Aplicação *Web*, denominada de PlantAiPortal, para responder às necessidades existentes na aplicação móvel e de manutenção do sistema, isto significa que a Aplicação *Web* tem como funcionalidades:

- Gestão de imagens a utilizar no treino;
- Gestão de modelos *AI*;
- Visualização de tratamentos;
- Classificação de imagens;
- Criação de conta;
- Alteração da password de conta.

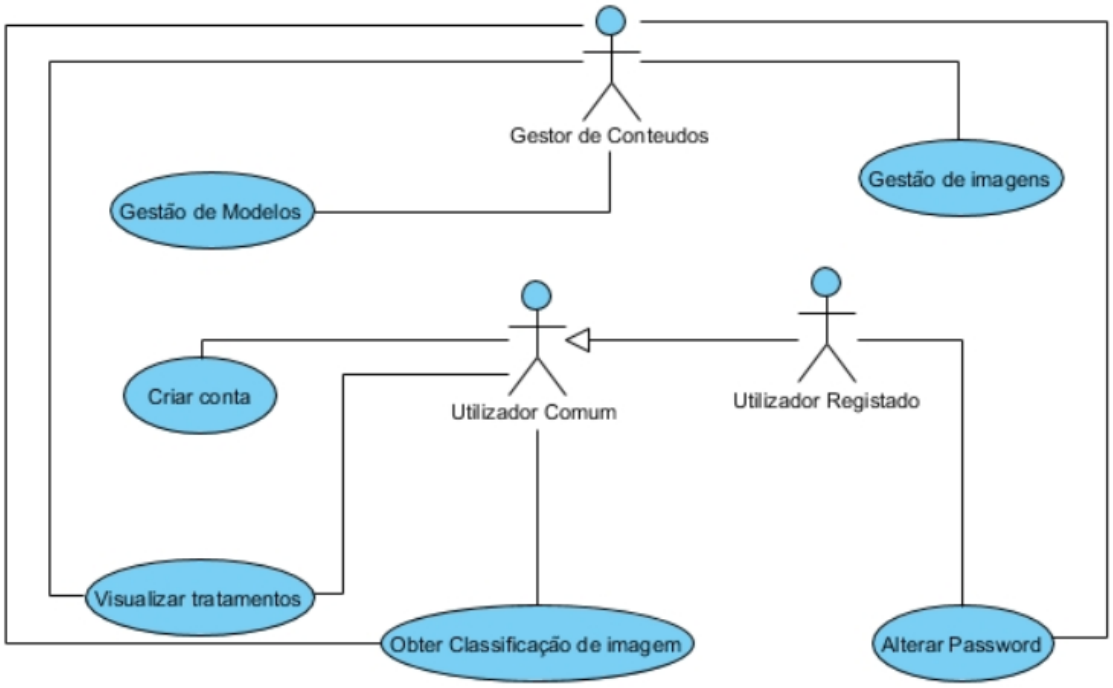


Figura 38 - diagrama de casos de uso para a aplicação web PlantAiPortal

Como mecanismo de autenticação e autorização, é utilizado ASP.NET Core Identity.

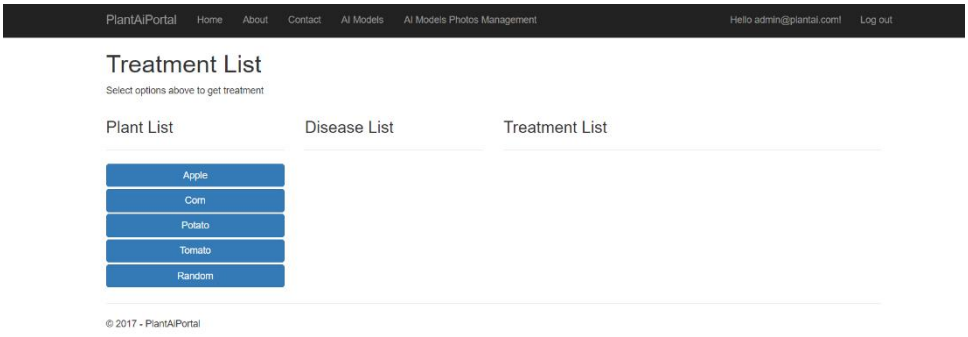


Figura 39 – PlantAiPortal - Homepage

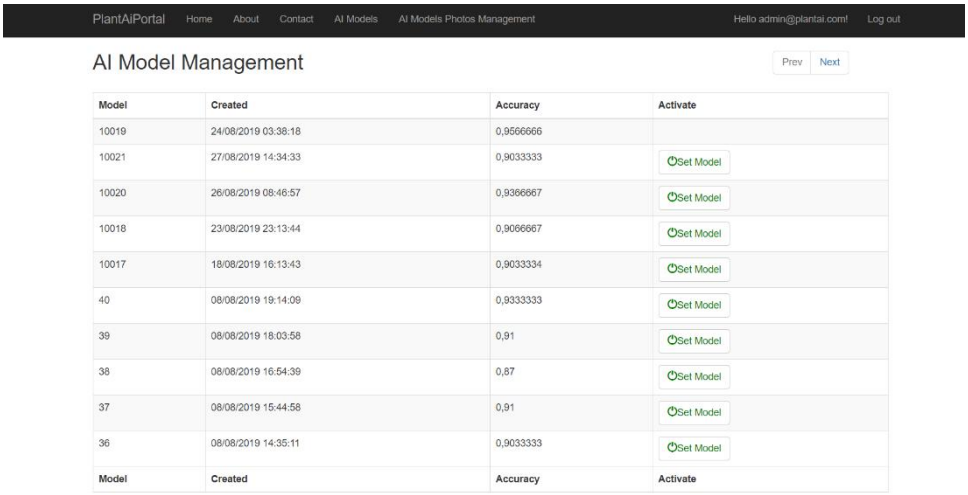


Figura 40 - PlantAiPortal - Gestão de modelos

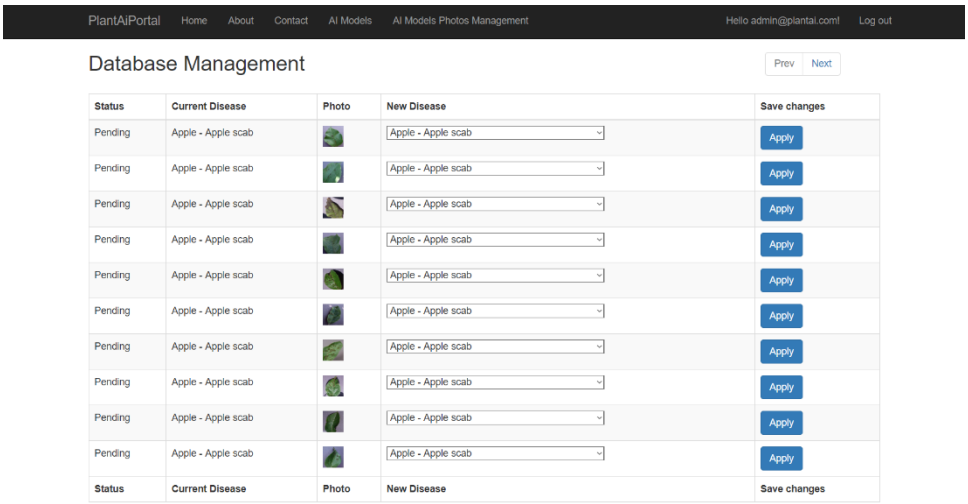


Figura 41 - PlantAiPortal - Gestão de Imagens

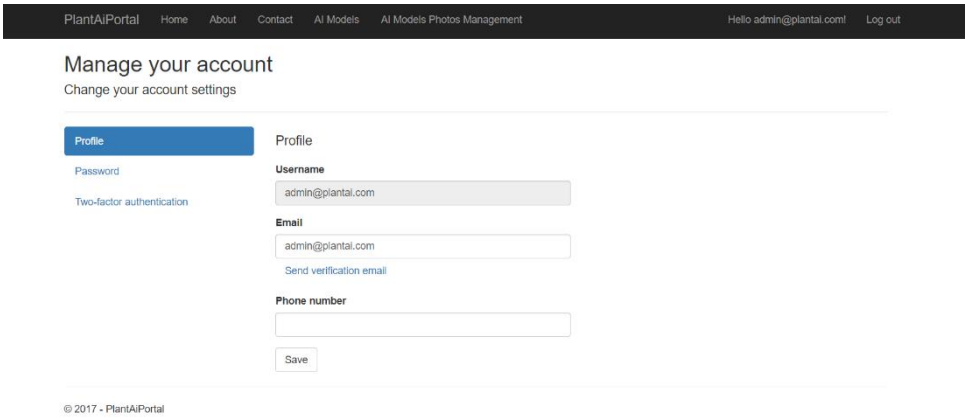


Figura 42 - PlantAiPortal - Gestão de conta

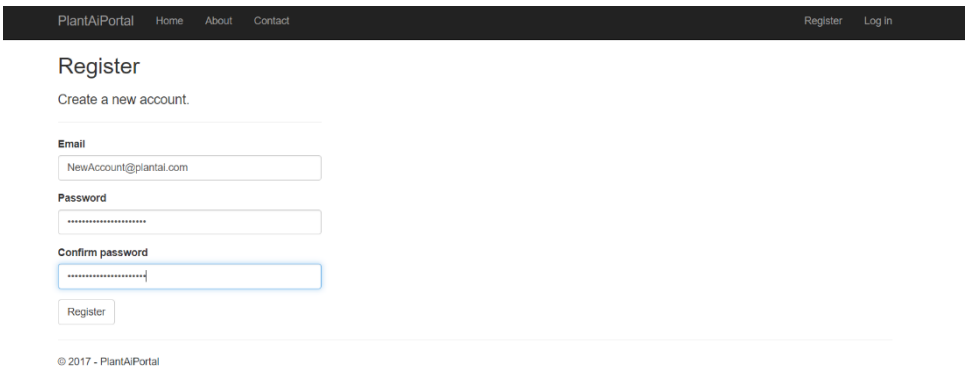


Figura 43 - PlantAiPortal - Criação de conta

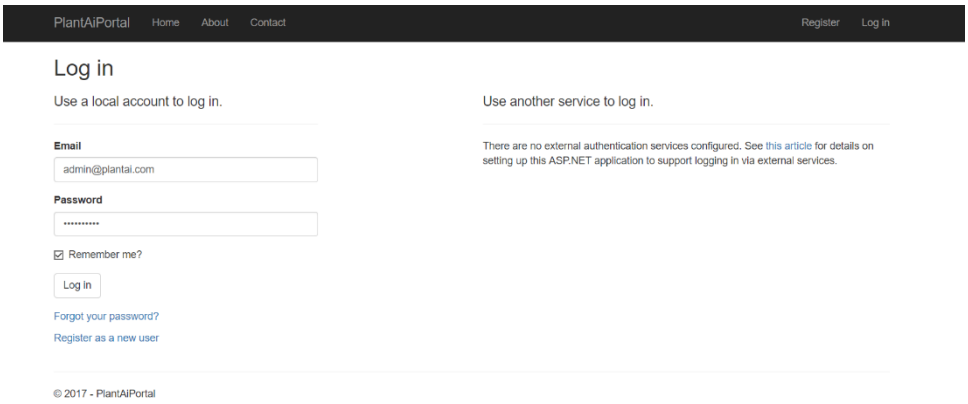


Figura 44 - PlantAiPortal - *Login*

5.8 Modelo de classificação

O desenvolvimento de um modelo de classificação constituiu uma das etapas fundamentais do trabalho desenvolvido, dado que este está diretamente ligado a uma das questões fulcrais desta dissertação: "É possível criar uma aplicação, capaz de classificar uma doença presente numa planta, com um grau de confiança, capaz de ser auxiliar viável para um profissional do sector?".

5.8.1 Dados

Para a conceção de modelos de *machine learning* e *deep learning*, um dos aspetos essenciais que deve ser garantido é a qualidade dos dados que serão utilizados na criação do modelo.

Efetivamente, a construção do modelo utiliza diversos processos que operam sobre informação de dados, sendo que a insuficiência de dados poderá inviabilizar todo o processo.

A fonte de dados que serviu de suporte a esta foi obtida através de uma organização, chamada PlantVillage[69], uma organização que recolheu, agrupou e disponibilizou numa plataforma [70] esses dados para a consulta e utilização que os utilizadores achem conveniente.

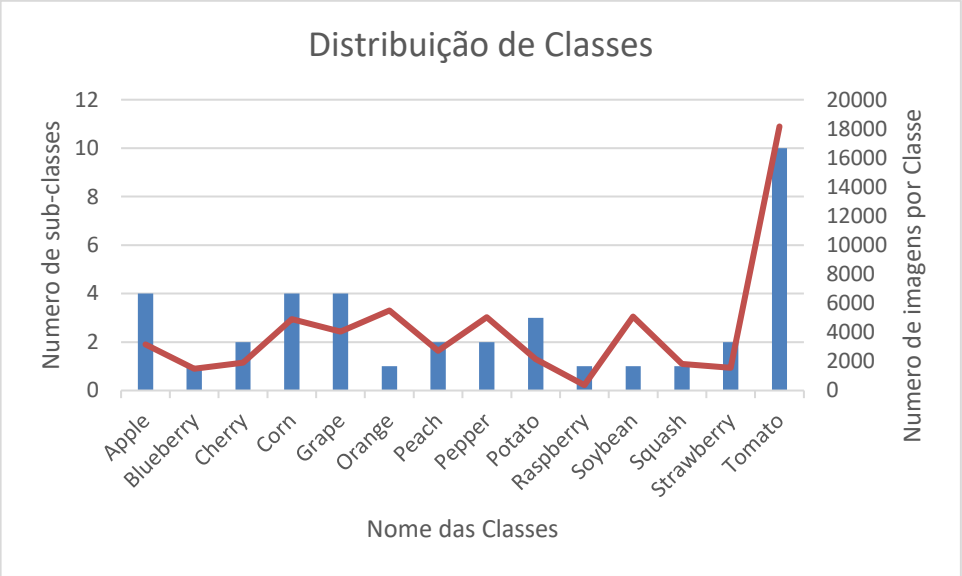


Figura 45 - Distribuição de classes no conjunto de dados PlantVillage

O conjunto de dados contém cerca de cinquenta e quatro mil imagens, distribuídas por trinta e oito classes. Cada classe representa um problema fitossanitário (doença) distinto, pertencentes a catorze espécies de plantas diferentes.

Figura 46 - Sample de imagens do conjunto de dados "PlantVillage"

De forma a agilizar o processo, das catorze espécies, foram selecionadas as quatro espécies com uma base de dados mais substancial, constituída por um total de vinte e uma classes diferentes e representando cerca de vinte e sete mil imagens, (correspondente a 50% da informação do *dataset*).

Adicionalmente, foi também recolhida uma fração de imagens aleatórias utilizando o *dataset* da *Imagenet* [37]. Estas imagens, não apresentam qualquer ligação a doenças nem a plantas e serão utilizadas para a criação de uma nova classe adicional no treino do modelo. Esta operação permitirá obter outro tipo de resultados que não somente as doenças de plantas, dado que poderão ser utilizadas imagens externas ao ambiente de plantas. É importante salientar também que estas não devem ser classificadas como uma doença de uma cultura agrícola.

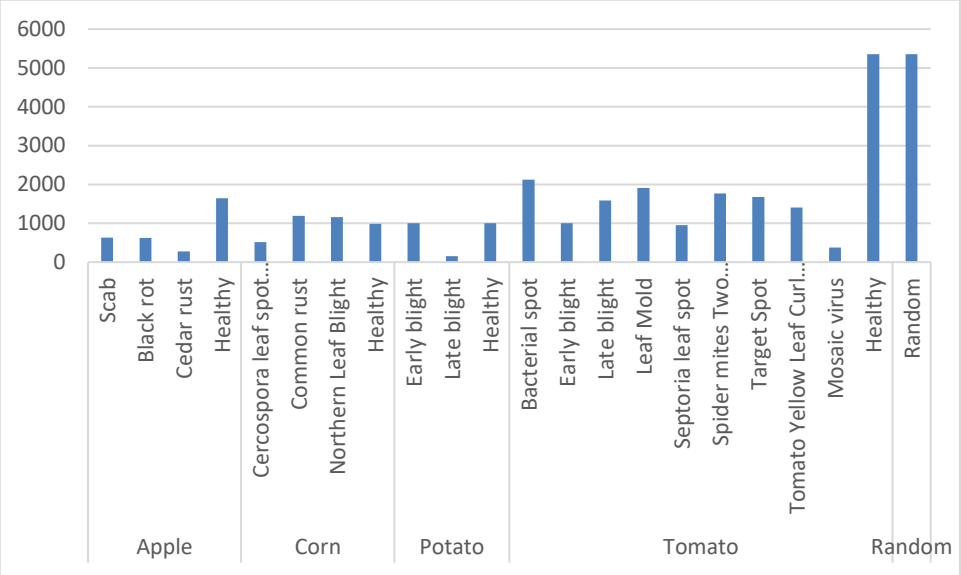


Figura 47 - Conjunto de dados selecionado para o estudo

5.8.2 Pré processamento de dados

No seguimento do capítulo anterior, dentro da gestão de dados para serem utilizados no desenvolvimento de um modelo de *Deep Learning*, outra etapa muito importante é o pré processamento de dados, uma vez que influencia diretamente o resultado final do treino de um modelo.

Um dos principais problemas, encontrado na fonte de dados utilizada, foram os desequilíbrios entre classes, o que significa que existia uma diferença, que neste caso era significativa, entre as várias classes, nomeadamente, a classe com menos dados continha 152 imagens diferentes, enquanto que a maior continha 5357 imagens.

Isto significa que no momento de treino do modelo, haveria um treino muito mais significativo na classe com maior número de imagens, face a com menos, pois existia muita mais informação.

Para combater este problema, e dado que o desequilíbrio era evidente, existem duas técnicas recomendadas, o ajuste dos pesos da rede, o que implica cálculos complexos que anulem os desequilíbrios, ou a utilização da técnica de criação de imagens a partir das existentes, criando pequenas alterações de forma a adicionar mais informação às classes minoritárias. Para a execução desta tarefa foi utilizado ImageDataGenerator pertencente à biblioteca do Keras. Este método, permitiu criar várias imagens, onde aplicava operações como rotações, alterações de dimensões do conteúdo da imagem, *zoom*, inversões, pequenas distorções e ruido.

Outra etapa do pré-processamento é o redimensionamento das imagens, uma vez que os treinos dos modelos, necessitam de imagens sem demasiada complexidade, uma vez que iria aumentar significativamente o tempo de treino. Deste modo, as imagens utilizadas no processo de treino foram redimensionadas para um tamanho de 256x256 pixéis.

Além da geração e do redimensionamento de imagens, outra das etapas do pré-processamento é a divisão de cada classe em diferentes proporções, para obtenção de dados de treino, teste e validação, pois esta divisão permite treinar e avaliar o modelo convenientemente.

Ainda sobre o pré-processamento, é imperativo que todas as imagens de um conjunto de dados ou imagens utilizadas no mecanismo de previsão, sejam uniformes para o sistema, isto significa que para o modelo, todas as imagens deveram sofrer um pré-processamento similar, com filtros, transformações e outras operações, que permitam equivaler as imagens. Para isso foi desenvolvido um método que é utilizado quer no treino, quer ao classificar uma imagem.

```
def preprocess_image(image, target_size):
    image = image.resize(target_size)
    image = img_to_array(image)
    image = xception.preprocess_input(image)
    image = array_to_img(image)
    image = np.expand_dims(image, axis=0)

    return image
```

Este algoritmo utiliza como input uma imagem e uma dimensão, aplicando as operações a essa imagem. Caso as imagens do treino e classificação não sofram o mesmo processamento, pode originar a resultados baixos em *accuracy*.

5.8.3 Treino do Modelo

A fase de treino e conceção do modelo inicia essencialmente por dois processos: a construção de uma arquitetura que permita criar modelos; injeção de dados na arquitetura. Com estes dois processos, é possível iniciar o treino, onde o algoritmo começa a processar os dados e a ajustar

os pesos, criando deste modo uma aprendizagem, para que no final do processo, seja gerado um artefacto, capaz de ser utilizado para prever resultados, neste caso, classificação de imagens.

O algoritmo utilizado nesta dissertação, utiliza *Transfer Learning*, que é uma metodologia que permite reaproveitar modelos. Neste caso, foram estudados diversos modelos na secção 3.3.3 da presente dissertação, e foi utilizado o MobileNet, que é uma rede neuronal com uma boa relação em função da *accuracy*, complexidade do modelo, utilização da memoria, complexidade computacional e tempo inferido no treino do modelo.

Nesta dissertação foram utilizadas duas formas distintas de operar, numa primeira mais simples, foi utilizado o script disponibilizado na documentação do TensorFlow, onde era possível definir um conjunto de parâmetros, onde o script iniciava o processo de treino e produzia um modelo final.

Do ponto de vista de implementação, esta abordagem não demonstrava com evidência o conhecimento em redes neuronais, já que utilizava um script muito complexo, sendo que a alteração do mesmo, seria uma tarefa também muito complexa.

Deste modo, em vez da utilização do TensorFlow da forma mais simples, foi criado um script que tinha como foco a criação de todo um processo de conceção de um modelo recorrendo à biblioteca Keras e com o TensorFlow a operar como *backend*.

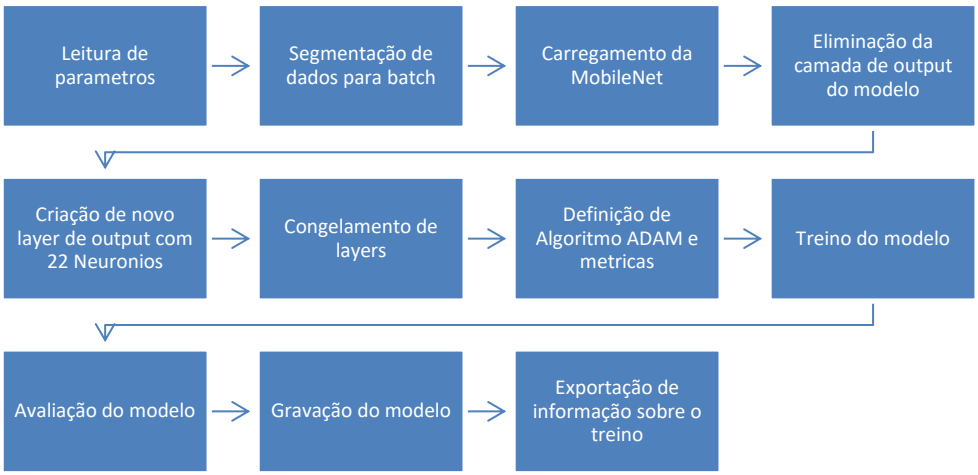


Figura 48 - Processo de treino

Tal como demonstra a Figura 48, a primeira etapa do processo de treino corresponde à leitura dos parâmetros que irão ser manipulados, concretamente a localização do conjunto de dados a ser utilizado no treino; o numero de *epochs*; o *batch size*; numero de camadas de output a serem eliminadas; o parâmetro de *learning rate*; numero de camadas congeladas.

Após a leitura dos parâmetros, é feita uma segmentação dos dados por 3 *batch* diferentes: treino, teste e validação, com respetivamente 80%, 10%, 10% das imagens do conjunto de dados.

Depois da segmentação dos dados, é carregado a rede MobileNet, ao qual é retirada a camada de output, sendo posteriormente adicionada uma nova camada de output, com 22 neurónios. Este processo é muito importante, porque se pretende remover a camada de output existente na rede previamente treinada, ou seja, é removida a camada onde contem todas as opções que a rede neuronal vai prever sendo depois adicionado uma nova camada com o numero de neurónios igual ao numero de opções que a nova rede neuronal deve devolver. No caso do modelo criado para a PlantAi, é necessário devolver 22 opções diferentes, conforme a secção 5.8.1, sendo depois disso definidas as camadas congeladas no modelo.

A definição do algoritmo e métricas de avaliação é também uma etapa muito importante. As métricas utilizadas estão descritas na secção 6.3.1.1 e o algoritmo utilizado é o ADAM[71], onde o único parâmetro manipulado nas experiências foi o *learning rate*, todos os restantes mantiveram-se os valores padrão:

```
keras.optimizers.Adam(learning_rate=_LR, beta_1=0.9, beta_2=0.999,
amsgrad=False)
```

Depois das etapas previamente descritas estarem concluídas, o treino é iniciado, começando a iterar a nova rede com o conjunto de dados de treino, sendo no final de cada epoch confrontado com os dados de validação. Neste passo é obtido o resultado das métricas definidas por cada epoch, de forma a verificar a evolução do treino do modelo, no decorrer das *epochs* definidas.

```
Epoch 1/5
370/370 [==============================] - 159s 430ms/step - loss: 0.8085 - acc: 0.8701 - val_loss: 0.6399 - val_acc: 0.8752
Epoch 2/5
370/370 [==============================] - 114s 308ms/step - loss: 0.1584 - acc: 0.9539 - val_loss: 0.3503 - val_acc: 0.9218
Epoch 3/5
370/370 [==============================] - 117s 315ms/step - loss: 0.1265 - acc: 0.9627 - val_loss: 0.2787 - val_acc: 0.9253
Epoch 4/5
370/370 [==============================] - 115s 311ms/step - loss: 0.0878 - acc: 0.9743 - val_loss: 0.2587 - val_acc: 0.9359
Epoch 5/5
370/370 [==============================] - 116s 312ms/step - loss: 0.0967 - acc: 0.9724 - val_loss: 0.1407 - val_acc: 0.9598
```

Figura 49 – Evolução do treino do modelo

Concluído o processo de criação de um modelo, é elaborada uma avaliação ao modelo previamente criado, utilizando o conjunto de teste, que é um conjunto que não teve nenhum impacto nos processos anteriores, comparando cada imagem com o resultado obtido de cada previsão utilizando o modelo criado na etapa anterior, de forma a obter a *accuracy* do modelo.

A última etapa é a exportação do modelo e dos dados colecionados em todo o processo de treino, de forma a registar as experiências efetuadas.

5.8.4 Processo de classificação de imagem

Um dos pré-requisitos no processo de classificação de uma imagem é um modelo previamente treinado e uma imagem para ser classificada.

No processo elaborado para a classificação de uma imagem, o primeiro interveniente no processo é o utilizador, que submete uma imagem, quer via aplicação móvel, quer via PlantAiPortal. Essa imagem submetida é submetida para a *API* .NET, que a comunica com o método de previsão exposto na Flask *API*.

Nesta fase, a imagem é carregada, bem como também é carregado o modelo classificador.

Após o carregamento destes dois componentes, a imagem é submetida para uma camada de pré-processamento, que aplica diversas transformações e filtragens, as mesmas que ocorreram para cada imagem durante o processo de treino do modelo, de forma a equivaler as operações entre as imagens que foram treinadas e as imagens se deseja prever.

Após estes processos estarem concluídos é invocada a função *predict* do Keras[72]. Esta função devolve um valor de confiança, associado a cada camadas de output definidas no treino no modelo, sendo que o valor mais elevado corresponde a opção com maior confiança, e o menor valor, a de menor confiança.

O resultado desta classificação é devolvido para a *API* .NET, que continua o processo, onde analisa qual a maior taxa de confiança obtida do resultado de classificação. Ao valor de maior confiança, adiciona nesta fase, algumas informações relativamente à classe juntamente com a base de dados.

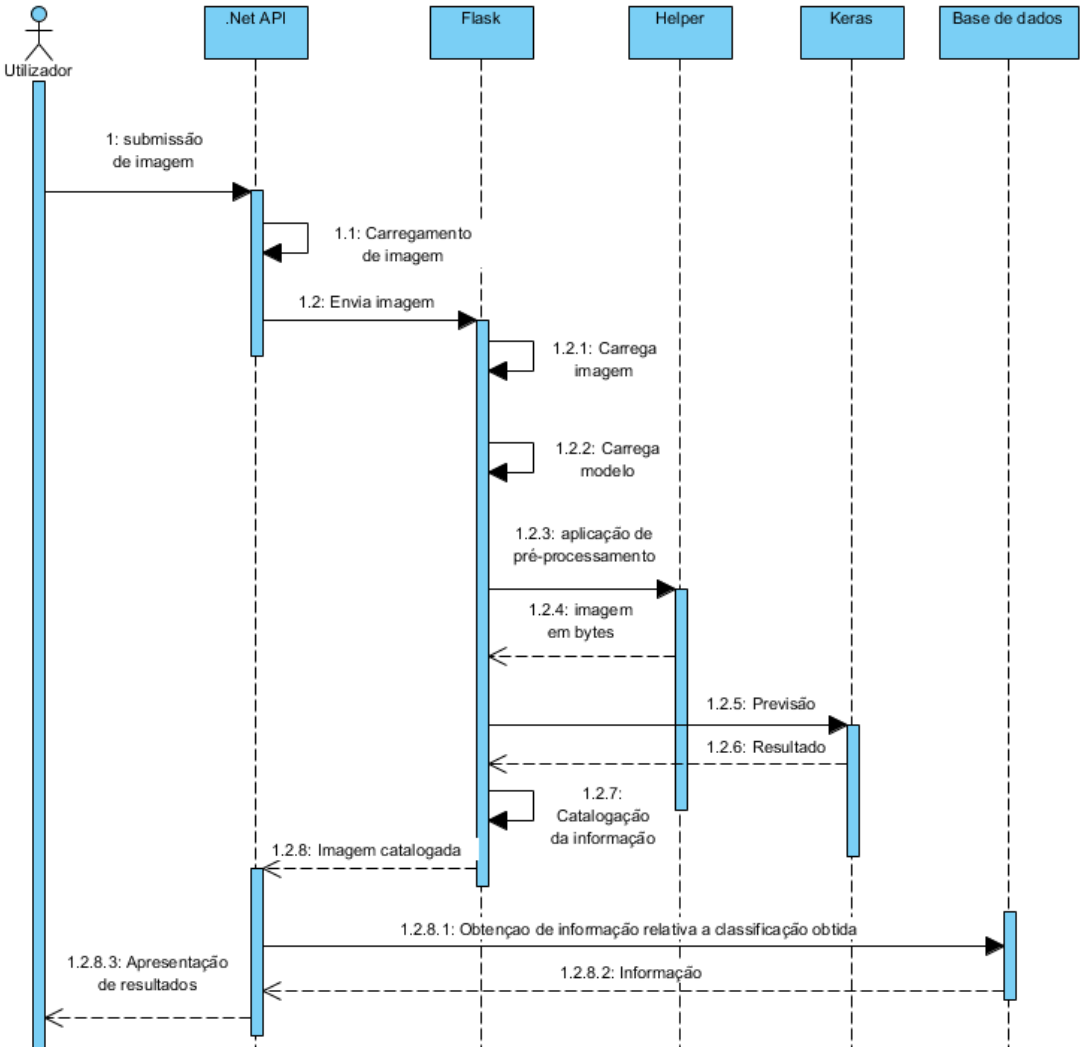


Figura 50 - Processo de classificação de imagem

5.8.5 Evolução automática do modelo

Uma das principais características desta dissertação está relacionada com o desenvolvimento de um modelo, capaz de classificar doenças em plantas, com um determinado grau de confiança.

Além deste ponto, essencial, também era pretendido criar um mecanismo que fosse capaz de evoluir automaticamente. Nesse sentido, foram estudadas algumas abordagens para acrescentar esta característica ao sistema.

A primeira abordagem era desenvolver um método na *API* desenvolvida em Python, que a partir da receção de um pedido HTTP, iniciasse o treino de um novo modelo, em *background*.

Com a arquitetura existente, e dada a limitação de recursos para o desenvolvimento desta dissertação, esta abordagem estava sujeita a riscos, nomeadamente devido à complexidade e carga relativa ao processamento no treino de um modelo, todo o ambiente que estaria

operacional iria ser condicionado pela carga existente no treino, sendo que poderia causar algum tipo de falhas no sistema.

Para dar solução ao pretendido, verificou-se que seria mais eficiente criar um script com todo o processo automatizado para o treino de um modelo e associar esse script a uma tarefa, que poderia ser agendada para ocorrer em determinado período, ficando deste modo ajustável.

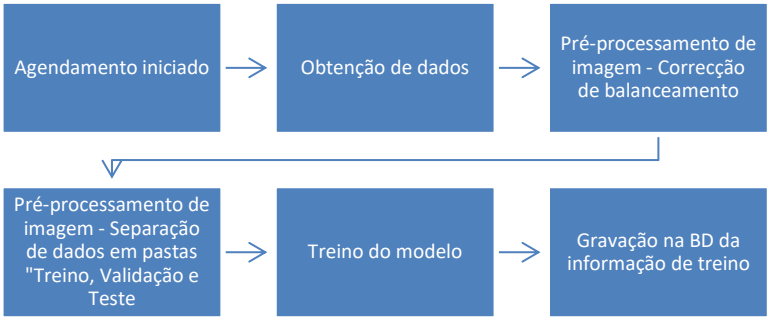


Figura 51 - Processo de automático de evolução do modelo

Quando a tarefa é iniciada, é necessário obter os dados do sistema, desse modo, a primeira etapa do processo de evolução automática é a obtenção dos dados. Esse processo é feito via HTTP, onde é invocado um pedido ao servidor para obter as imagens para o próximo treino.

O serviço desenvolvido carrega da base de dados, todas as imagens que devem fazer parte do novo treino. Após a conclusão do processo de obtenção e categorização de todas imagens da base de dados, é criado um ficheiro em formato ZIP, onde é devolvido para o processo de treino.

Depois de obtido o novo conjunto de dados, é feita uma extração dessa informação para um diretório, onde caso esteja ativo o parâmetro de correção de balanceamento de imagens, será elaborado um processo que para cada classe, analisa o número de imagens que aquela classe necessita ate ficar equivalente com a classe com mais imagens. Após ser obtida essa informação, cada classe processa as suas imagens, utilizando o mecanismo do ImageDataGenerator, até que todas as classes estejam uniformes.

De seguida, é feito uma divisão do novo conjunto de dados em três secções, treino, validação e teste, respetivamente na proporção de 80%, 10% e 10% por cada classe.

O passo posterior é a execução do treino, utilizando o processo definido anteriormente.

6 Avaliação dos resultados

Após a solução ser desenvolvida é necessário verificar se os objetivos traçados no início do projeto foram alcançados com sucesso.

O sucesso do trabalho pode não depender apenas se a solução apresenta o resultado esperado, poderá existir outros fatores igualmente importantes que necessitam de participar nos critérios para a obtenção do sucesso, como o tempo para realizar determinada ação.

6.1 Avaliação do modelo

Um dos principais componentes deste tipo de trabalhos que envolvem reconhecimento de imagem utilizando *Deep Learning* é o componente do modelo de treino.

Para a criação de um modelo de classificação são necessários dados e parâmetros. Os parâmetros permitem ajustar o modelo, provocando variações nos resultados, influenciando quer em tempo de treino, quer no valor da taxa de acerto.

Relativamente aos dados introduzidos no processo de treino, o primeiro passo é recolher dados para serem incluídos no processo. Tal como definido na secção 2.1.5 da presente dissertação, os dados que irão ser utilizados no processo de treino foram obtidos de uma comunidade chamada PlantVillage. No entanto, para efetuar o processo de treino é comum fazer uma divisão dos dados, sendo que existem duas partes, a parte utilizada durante a fase de treino do modelo e a parte utilizada para testar o modelo. Esta última parte é importante, pois é com os dados desta fração que se vai analisar os resultados, devido ao facto de estes dados não constarem no processo de treino, pelo que são entendidos como dados novos no sistema.

De acordo com a literatura existem dois métodos muito utilizados, sendo o método *Hold-out* mais simples, consistindo em dividir 2/3 dos dados para treino e o remanescente para teste.

O segundo método muito utilizado é o *Cross Validation*. Este método consiste em dividir os dados em vários subconjuntos, por exemplo em 10 conjuntos, em que cada conjunto é utilizado para teste uma vez e os restantes para treino. Após a criação de dez modelos um em cada iteração é calculado a taxa de acerto do modelo com o subconjunto não usado na sua criação, e no final a média obtida taxa de acerto nas iterações representa a avaliação final.

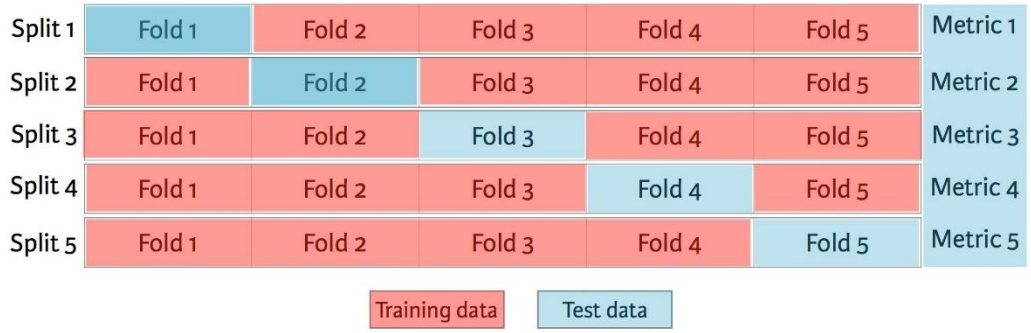


Figura 52 - Funcionamento do método de *Cross Validation*[73]

Segundo a literatura, a grande diferença entre os dois métodos, é que enquanto o método *hold-out*, utiliza 1/3 das imagens do conjunto, o *Cross Validation* utiliza subdivide um conjunto em vários, em que para cada conjunto, existe uma parte de treino e validação, sendo que o valor final é composto pela média para cada iteração do conjunto de dados, deste modo é visto como mais eficiente. [74].

No entanto, o método *Hold-out* é mais simples que o *Cross Validation*, pois a segunda como utiliza múltiplos conjuntos de treino-teste, requer de mais tempo e poder computacional. Desse modo, para o trabalho desta dissertação, será utilizado o mecanismo de *Hold-out*, devido à utilização de menos recursos e simplicidade do mecanismo.

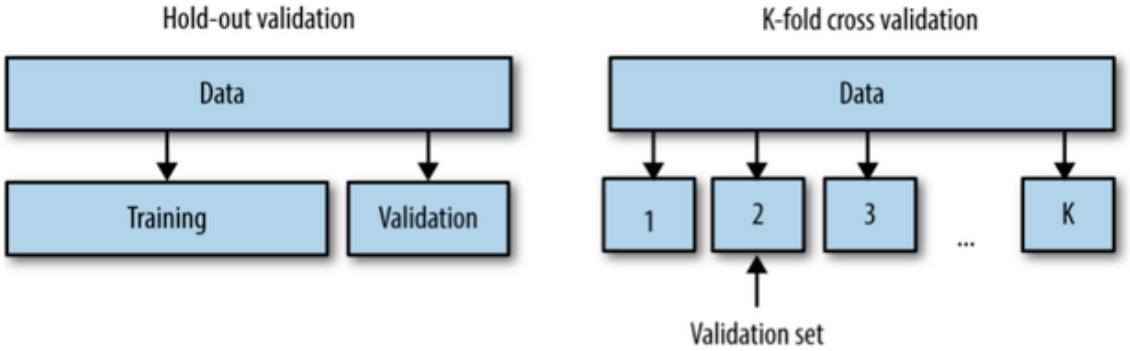


Figura 53 - *Hold-out vs K-Fold Cross Validation*

6.2 Metodologia de avaliação

Esta tese tem como propósito identificar doenças em plantas, no entanto, este propósito já existe no quotidiano, que tradicionalmente ocorre através da análise de especialistas.

Qualquer análise por especialistas está sujeita ao erro humano, deste modo existe uma taxa de acerto na avaliação dos especialistas que pode ser dada por uma fórmula como:

$$accuracy(x) = \frac{Nr\ de\ identificações\ correctas}{Nr\ de\ casos\ analisados} \quad (2)$$

A fim de demonstrar os resultados obtidos, adicionalmente para cada avaliação de parâmetro no decorrer da fase de treino, será apresentado um conjunto de experiências elaboradas, sendo que para cada avaliação irá ser feita em função de uma fórmula f (3), à qual é arbitrada a importância de 30% ao factor tempo de treino do modelo, e de 70% ao factor da *accuracy*, para demonstrar o resultado mais adequado para os parâmetros configurados.

É importante referir que o valor da *accuracy* terá por base uma função criada para a partir de um set de imagens, validar para cada um qual o resultado obtido da previsão e qual o resultado esperado.

$$f = \left(\left(1 - \left(\frac{tempo_{medio}}{MAX_{tempo\ obtido}}\right)\right) * 0.3\right) + (\text{accuracy} * 0.7) \quad (3)$$

6.3 Resultados obtidos

Nesta secção serão apresentados todos os dados obtidos no decorrer do desenvolvimento das várias experiências na elaboração de um modelo para classificação de imagens de doenças de plantas.

6.3.1 Treino do modelo

Em seguida são apresentadas informações relevantes e explicações de todos os elementos envolvidos no treino do modelo.

6.3.1.1 Métricas de avaliação

As métricas de avaliação do treino do modelo utilizadas são a *accuracy* e o tempo de treino do modelo.

A *accuracy* é uma função que permite perceber num determinado número de cenários qual a proporção de resultados corretos obtidos a classificar esses resultados.

Outra das medidas utilizadas para avaliar um modelo é o fator do tempo que um modelo necessita para treinar.

6.3.1.2 Parâmetros a avaliar

Esta secção é importante para aferir os resultados obtidos com a manipulação dos parâmetros, de modo a verificar, dentro de aqueles parâmetros, quais as melhores opções e se existem opções que inviabilizam os objetivos da dissertação.

6.3.1.2.1 *Learning Rate*

O *learning rate* é um parâmetro que permite definir a velocidade a que uma rede altera os seus pesos, sendo que quanto maior o *learning rate* mais variam os pesos da rede neuronal, ou seja é a taxa de aprendizagem da rede, sendo que uma taxa mais elevada pode definir uma aprendizagem mais focada nos últimos padrões identificados, enquanto que taxas mais reduzidas, permitem uma aprendizagem mais contínua.

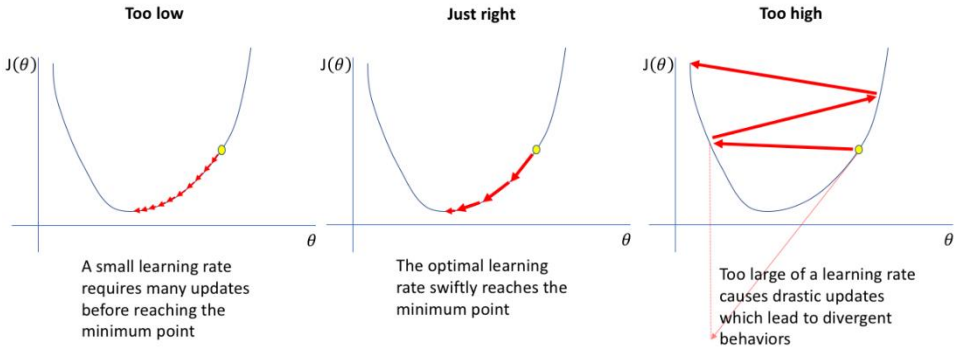


Figura 54 - Diferenças entre *Learning Rate*[75]

6.3.1.2.2 Número de *Epochs*

O número de *epochs* significa o número de vezes que a rede neuronal é iterada com valores, ou seja, é o número de vezes que os dados iteram pela rede neuronal, para provocar o ajuste de valores.

O número de *epoch* faz variar o tempo do treino de um modelo, sendo que quando um sistema estuda um conjunto de dados, de acordo com o grau de aprendizagem (*learning rate*), faz com que o modelo fique mais apurado, sendo que normalmente as variações entre as primeiras *epochs* variam mais que as últimas, precisamente devido ao conhecimento obtido das *epochs* anteriores.

6.3.1.2.3 *Batch Size*

O *batch size* é um parâmetro que está ligado ao número de *epochs*, pois define a quantidade de elementos de um conjunto de dados que são utilizados em cada ciclo de iteração de uma epoch, sendo que uma *epoch* representa uma iteração de dados, composta por vários ciclos, onde cada ciclo tem um tamanho, ao que corresponde um *batch size*.

Na prática significa que no caso de um conjunto de dados com 100 elementos, para completar uma *epoch* é necessário iterar os 100 elementos. Utilizando *batch size*, os elementos são segregados em determinada porção (*batch size*), e para completar uma *epoch*, é necessário criar N ciclos, onde cada ciclo utiliza uma porção.

6.3.1.2.4 Optimizador

O optimizador escolhido nas experiências é o Adam que é um algoritmo de otimização utilizado em *Deep Learning*, que combina várias propriedades de outros algoritmos como AdaGrad e RMSProp, e é um dos algoritmos vulgarmente utilizado em treinos de modelos de classificação, devido à sua eficiência e facilidade de configuração.[76]

6.3.1.2.5 Camadas congeladas

As camadas congeladas correspondem às camadas que são configuradas para não serem treinadas devido a um *overfitting* da rede aos dados.

Uma das principais vantagens da utilização de *Transfer Learning* é a reutilização da definição quer da arquitetura da rede quer dos pesos dessa rede. Normalmente estas redes, sevem de base para o desenvolvimento de outras redes, sendo que por exemplo, a MobileNet é uma rede neuronal desenvolvida para ser eficiente, quer em capacidade quer em dimensão, sendo que é uma rede já bastante maturada ao ponto de ser uma das referências no que toca a redes neuronais.

Para se tirar melhor partido de uma rede neuronal previamente treinada, como no caso da MobileNet, há um determinado número de *layers* que podemos manter os pesos já que efetuam um determinado número de operações, como aplicação de filtros, que podem beneficiar muitas redes neuronais. Por isso, pode ser também importante, manter camadas. A

maneira de manter essas camadas é definir que essas camadas passam a ser congeladas, ou seja, deixam de ser treinadas.

6.3.1.2.6 Avaliação

Depois de treinado um modelo é necessário confrontar os resultados obtidos com um teste através de dados não usados no treino da rede de forma a aferir a veracidade do modelo.

Nesse sentido, a matriz de confusão é uma ferramenta importante para fazer essa aferição. [77]

A matriz de confusão é uma matriz que permite, para cada classe do modelo, visualizar as frequencias de classificação:

- Verdadeiros positivos (VP), que ocorre quando no conjunto de dados, a classe que se está a analisar é a classe obtida do resultado da previsão.
- Falsos positivos (FP), que ocorre quando no conjunto de dados, a classe que se está a analisar não foi a classe obtida do resultado da previsão.
- Falso verdadeiro (FV), que ocorre quando no conjunto de dados, a classe que não se está a analisar foi a classe obtida do resultado da previsão.
- Falso negativo (FN), que ocorre quando no conjunto de dados, a classe que não se está a analisar, não foi a classe obtida do resultado da previsão.

Com estes dados, é possivel obter a partir da matriz de confusão as seguintes métricas:

- Accuracy – define a percentagem de acerto do modelo, já definida na fórmula (1)

$$accuracy = \frac{VP+FV}{VP+FP+FV+FN} = \frac{previsões\ correctas}{total\ de\ previsões} \quad (4)$$

- Recall – define a proporção de positivos que foram identificados correctamente;

$$recall = \frac{VP}{VP+FN} \quad (5)$$

- Precision – define a proporção de identificações positivas que são correctas;

$$precision = \frac{VP}{VP+FP} \quad (6)$$

- F-Score – demonstra o balanco entre a precision e o recall.

$$fscore = 2 * \frac{precision*recall}{precision+recall} \quad (7)$$

Os modelos de classificação binária, utilizam frequentemente as métricas descritas anteriormente. Em modelos de multi-classe verifica-se que a métrica que melhor se adequa é a *accuracy*, que indica a quantidade de vezes que o modelo acertou determinada previsão, que corresponde à diagonal da matriz de confusão.

Tabela 5 – Exemplo Matriz de Confusão com resultados obtido da secção 6.4 – P1

Categorias	A0	A1	A2	A3	C0	C1	C2	C3	P0	P1	P2	R	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	
A0	58	0	1	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	A0
A1	1	60	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	A1
A2	0	1	24	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	A2
A3	2	0	0	146	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	A3
C0	0	0	0	0	42	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	C0
C1	0	0	0	0	0	118	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	C1
C2	0	0	0	0	0	0	115	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	C2
C3	0	0	0	0	13	0	1	83	0	0	0	0	0	0	0	0	0	0	0	0	0	0	C3
P0	0	0	0	0	0	0	0	0	98	0	0	0	0	0	0	0	0	1	0	0	0	0	P0
P1	0	0	0	0	0	0	0	0	0	12	0	0	0	0	1	0	0	0	1	0	1	0	P1
P2	0	0	0	0	0	0	0	0	0	3	89	0	0	1	0	5	0	0	1	0	0	0	P2
R	0	0	0	0	1	0	0	0	0	1	0	147	0	0	0	0	0	0	0	0	0	1	R
T0	0	0	1	0	0	0	0	0	0	0	0	0	141	2	0	0	0	1	0	4	0	1	T0
T1	0	0	0	0	0	0	0	1	0	0	0	0	3	87	0	3	0	3	0	2	0	0	T1
T2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	150	0	0	0	0	0	0	0	T2
T3	0	0	1	0	0	0	0	0	0	0	2	1	0	1	0	139	2	1	2	0	0	1	T3
T4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	89	1	0	1	1	0	T4
T5	0	0	0	0	0	0	0	0	2	0	0	0	5	2	0	2	3	136	0	0	0	0	T5
T6	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	2	135	7	1	2	T6
T7	0	0	0	0	0	0	0	0	1	0	0	0	1	2	4	1	0	1	6	123	0	0	T7
T8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	33	0	T8
T9	0	0	0	0	0	0	0	0	0	0	0	0	2	1	0	0	0	0	1	0	0	146	T9
	A0	A1	A2	A3	C0	C1	C2	C3	P0	P1	P2	R	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	

Resultados obtidos

6.3.2 Experiências

Nesta secção, irão ser apresentados os vários desenvolvimentos e comparativos entre as várias experiências realizadas a fim de demonstrar a influência do ajuste de parametrizações na elaboração de modelos utilizando a API Keras.

É importante referir que ambas as experiências serão compostas por 5 avaliações para cada parametrização distinta, onde serão utilizadas as medidas obtidas dos parâmetros de avaliação de forma a verificar a consistência dos dados das experiências.

Todas as experiências terão como base a CNN MobileNet, e irão ser processadas 23147 imagens com classes não balanceadas, dividas em treino, teste e validação, na proporção de 80%, 10% e 10% respetivamente, sendo que os dois primeiros serão utilizados para processos relacionados com o treino e verificação das evoluções do treino, e um último que será utilizado para testar o modelo com dados que não entraram no processo anterior.

A justificação para as 5 avaliações deve-se ao facto de para cada treino, criação de sequências para serem utilizadas no processo de treino, há um parâmetro ativo no conjunto de treino e validação, que ordena aleatoriamente as imagens. Esse processo, tem impacto na forma de ajuste de valores na rede neuronal, deste modo, para cada experiencia e para cada avaliação individual do uso de determinado parâmetro, no final das 5 avaliações, é definida uma media para a avaliação correspondente e com os resultados dessa media é aplicado as métricas definidas na secção 6.1.

6.3.2.1 CPU VS GPU

Tal como referido no capítulo 4.1.1, um dos grandes requisitos quando se trabalha com redes neuronais e *Deep Learning*, é dispor de poder de processamento.

A primeira análise elaborada corresponde exatamente ao comparativo entre treinos, em condições e parametrizações equivalentes, onde a única diferença entre ambas é que uma experiência foi desenvolvida recorrendo à capacidade de processamento do CPU enquanto a outra experiência foi desenvolvida utilizando GPU (características presentes na secção 5.2).

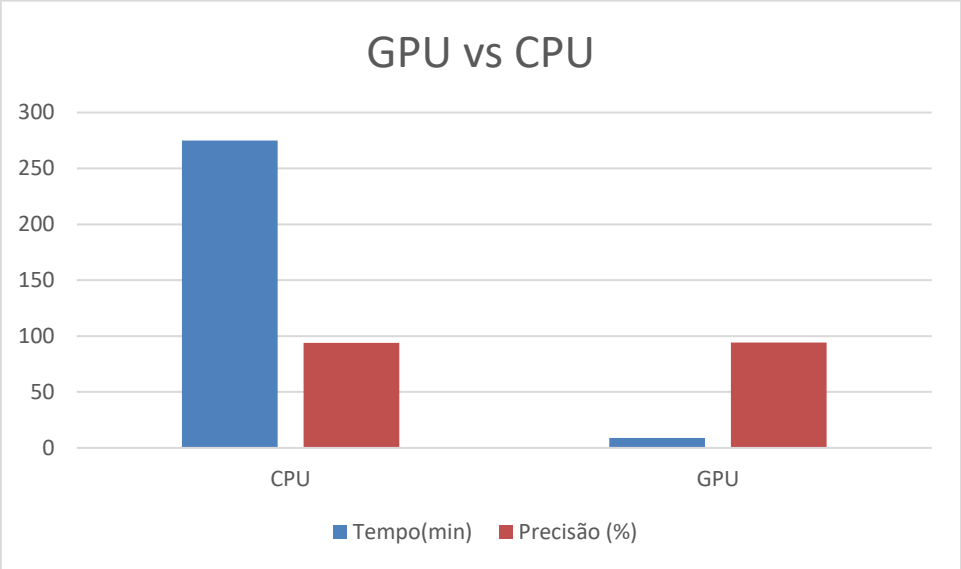


Figura 55 - Treino em CPU vs GPU

Relativamente às parametrizações, ambas as experiências utilizaram o mesmo conjunto de dados, operando 23147 imagens, utilizando 5 *epochs*, em *batches* de 50 imagens, com uma *learning rate* de 0.00001 e utilizando como base de *Transfer Learning* a CNN *Mobilenet*.

Comparando os dados recolhidos, verifica-se que na experiência elaborada, utilizando CPU foram contabilizados 275 minutos, enquanto que utilizando GPU, para a mesma experiência, foram gastos 9 minutos de treino. Relativamente à *accuracy* obtida, é possível verificar que são equivalentes, onde em CPU foi registado 0,9403% de *accuracy*, enquanto que em GPU 0,9412.

Esta experiência demonstra que efetivamente verificou-se uma mais valia na utilização de GPU para o treino de modelos, pois para obter taxas de acerto com resultados equivalentes o GPU registou aproximadamente 3% do tempo gasto em CPU.

6.3.2.2 Balanceamento de dados

Tal como a experiência anterior, é relevante aferir os efeitos provocados pelo não balanceamento dos dados.

Deste modo, foi elaborado 2 experiências com parâmetros exatamente iguais, onde apenas difere o balanceamento dos dados.

Atendendo que é impossível em tempo útil recolher mais dados, e que não foi encontrado mais nenhum conjunto de dados capaz de complementar o existente, recorrendo à *API* Keras, nomeadamente a função *ImageDataGenerator,* é possível executar um algoritmo que consegue gerar imagens utilizando as existentes, aplicando algumas transformações às imagens, como a adição de ruído, distorção, inversão de eixos e alteração da escala da imagem.

A experiência foi elaborada utilizando os seguintes parâmetros:

Tabela 6 - Experiência - Impacto do balanceamento dos dados

	Epochs	*BatchSize*	*LearningRate*	Camadas Congeladas	Imagens
Balanceados	10	100	0.0001	2	116.666
Não Balanceados	10	100	0.0001	2	31.959

Tabela 7 - Demonstração de resultados de dados balanceados vs não balanceados

	minutos	Exp1	total	certos	Exp2	total	certos
Balanceados	100	0,9936	11660	11585	0,9864	440	434

Não Balanceados	33	0,9862	3191	9147	0,9909	440	436

Para analisar esta experiência, foram desenvolvidas duas camadas de validação de resultados, uma (Exp1) que utiliza dados não incluídos no processo de treino, mas com uma proporção de 10%, e outra (exp2) que também utiliza dados não incluídos no processo de treino, mas não relacionada em forma percentual, apenas contém 20 imagens de cada classe treinada.

A razão pela qual foi criada a segunda experiência, foi para analisar de forma independente o impacto provocado pelo não balanceamento de dados, já que o não balanceamento pode ocorrer de forma natural.

Deste modo, arbitrando um conjunto de imagens, neste caso 20 por cada classe fica excluído qualquer impacto causado pela proporção das imagens em teste.

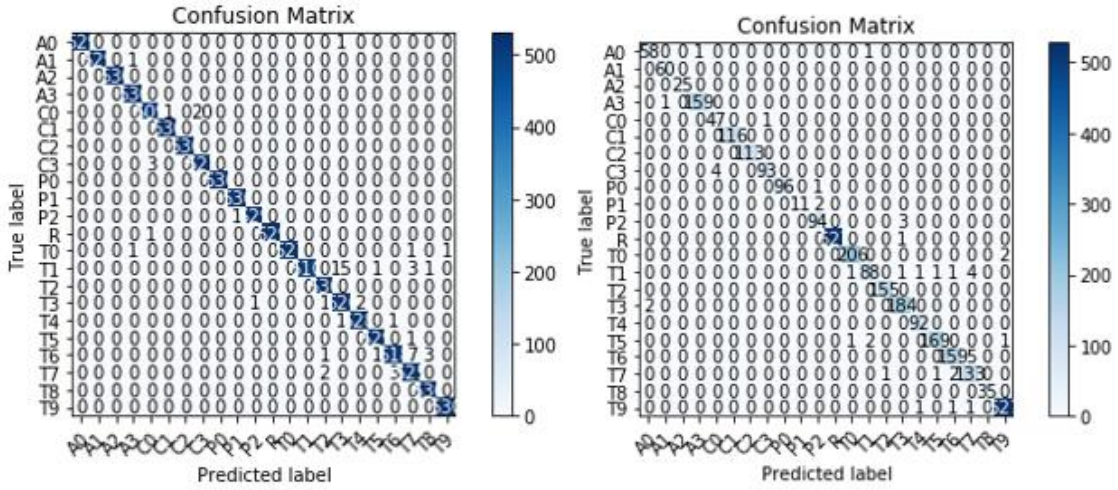


Figura 56 – Matriz de confusão de dados balanceados *vs* não balanceados

Tal como foi possível verificar, quer pela Tabela 6 e pela análise das matrizes de confusão da exp1, ou seja, método proporcional é possível aferir que ao efetuar um balanceamento utilizando imagens geradas a partir das imagens existentes, existiu um acréscimo significativo em relação ao tempo despendido por cada treino, no entanto, esse aumento de imagens, não provocou um melhoramento da taxa de acerto do modelo.

6.3.2.3 Número de *Epochs*

Tal como referido anteriormente, um dos fatores que contribui diretamente para a variação da *accuracy* e do tempo de treino é o numero de *Epochs*, sendo que a configuração das *epochs* pode ser vista como uma espécie de *trade-off*, porque por um lado, o aumento do número de *epochs*, faz aumentar o tempo total do treino do modelo, precisamente porque aumenta o número de iterações da rede neuronal para o ajuste de pesos, por outro lado, uma diminuição

do número de *epochs*, condiciona o ajuste dos pesos, podendo provocar um aumento da *loss function* e diminuição da *accuracy*.

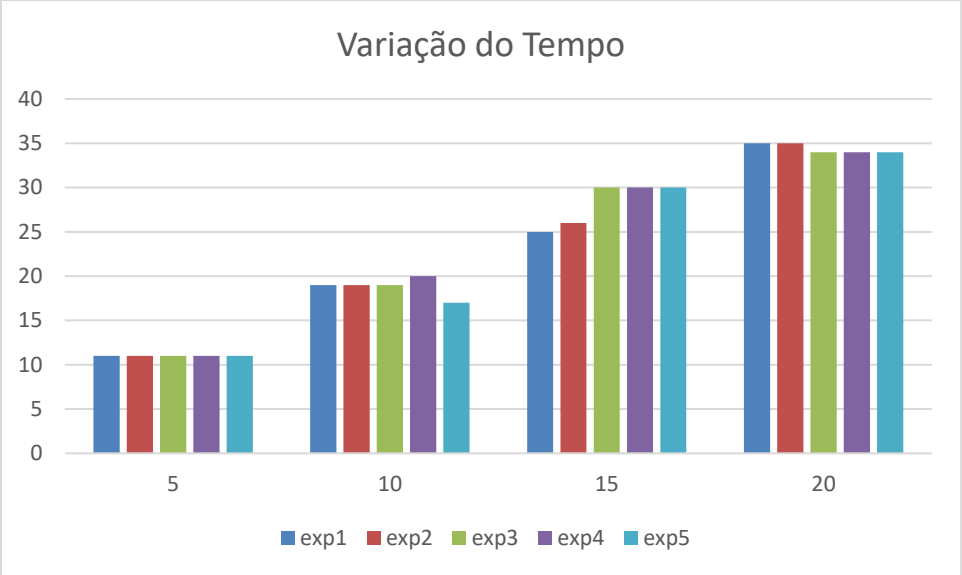


Figura 57 - Experiência - Tempo de treino *vs* número *Epochs*

Os gráficos da Figura 57 e da Figura 58 registam o resultado de 20 experiências, onde as condições fixas são o número de imagens (23147), utilizando como base *Transfer Learning*, a CNN Mobilenet, com um *batch size* de 50 imagens e um *learning rate* de 0.001. A variável da experiência é o número de *epochs*, onde varia para 5, 10, 15 e 20 *epochs*.

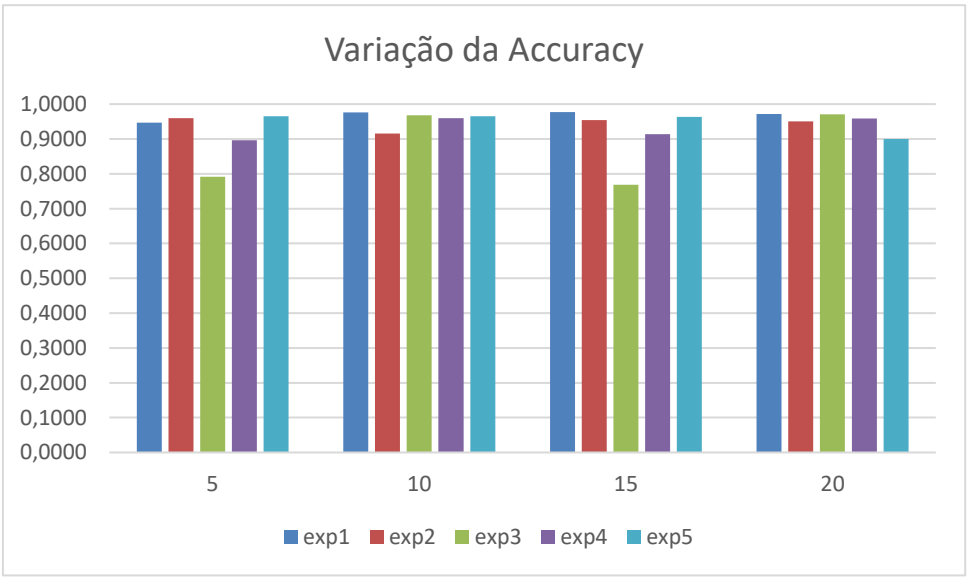
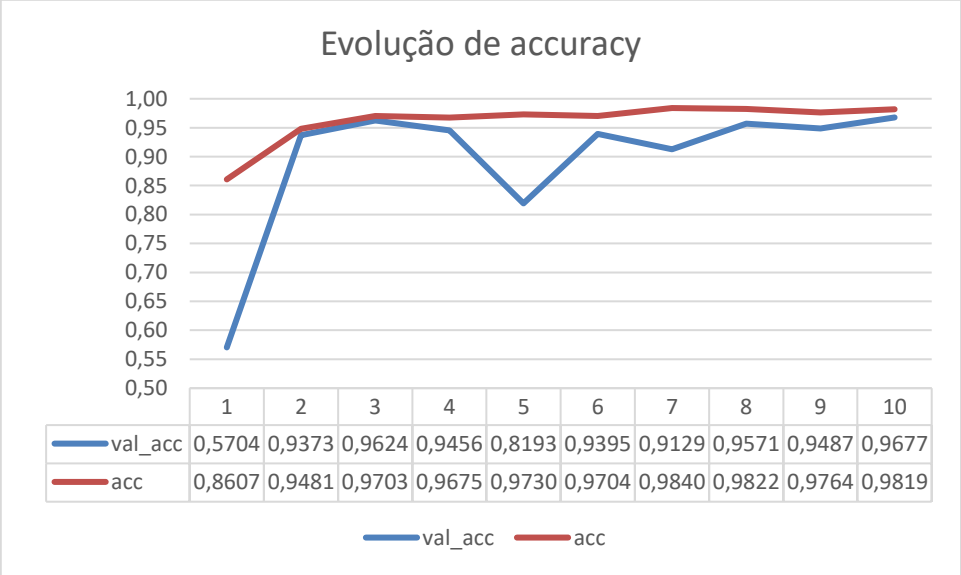


Figura 58 - Experiência - *Accuracy vs* Número *Epochs*

Um aspeto importante a demonstrar quando se definem as *epochs* é verificar a variação dos resultados em função dos ajustes na rede neuronal, sendo que a variação verificada nas primeiras *epochs* é superior à verificada nas últimas, verificando-se inclusive que algumas *epochs* poderão ser desnecessárias face ao aumento de tempo que provocam.



	1	2	3	4	5	6	7	8	9	10
val_acc	0,5704	0,9373	0,9624	0,9456	0,8193	0,9395	0,9129	0,9571	0,9487	0,9677
acc	0,8607	0,9481	0,9703	0,9675	0,9730	0,9704	0,9840	0,9822	0,9764	0,9819

Figura 59 - Evolução da *accuracy* no decorrer de uma experiência de treino

Tabela 8 - Experiência - Número *Epochs* e Variação de *accuracy* e tempo

Número	Tempo(m)	*Accuracy*	Fórmula f	Amplitude
5	11	0,9121	0,842513	17%
10	18,8	0,9567	0,805722	6%
15	28,2	0,9153	0,694753	21%
20	34,4	0,9504	0,665257	7%

Esta tabela indica que, de acordo com a fórmula definida para avaliar as parametrizações do treino dos modelos na secção 6.1, o número de *epochs* recomendado para o sistema, varia entre 5 e 10, no entanto, de acordo com a Figura 59, verificou-se que na avaliação da experiência de 5 *epochs*, existiu uma diminuição do valor da *accuracy* em 17% face ao valor com o melhor registo de *accuracy*, enquanto que a variação entre o melhor e pior registo do teste

de 10 *epochs*, apenas demonstra uma variação de 6 % entre a melhor e pior *accuracy* obtida, gerando mais confiança nos dados resultados recolhidos na experiência com 10 *epochs*.

6.3.2.4 *Batch Size*

Para a experiência do ajuste de parâmetros do batch size, foram analisados os cenários de 25, 50, 75, 100, 125 e 150 para o parâmetro de *Batch* Size.

Relativamente às restantes parametrizações, as condições fixas desta demonstração de resultados são o número de imagens (23147), a base de *Transfer Learning*, Mobilenet, com 10 *epochs* e um *learning* rate de 0.001.

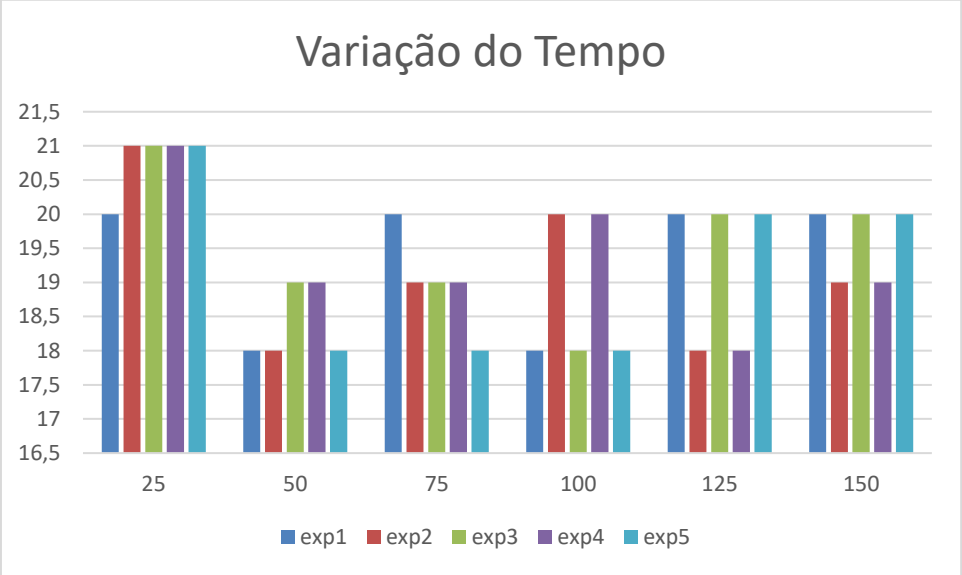


Figura 60 - Experiência - Tempo de treino *vs Batch Size*

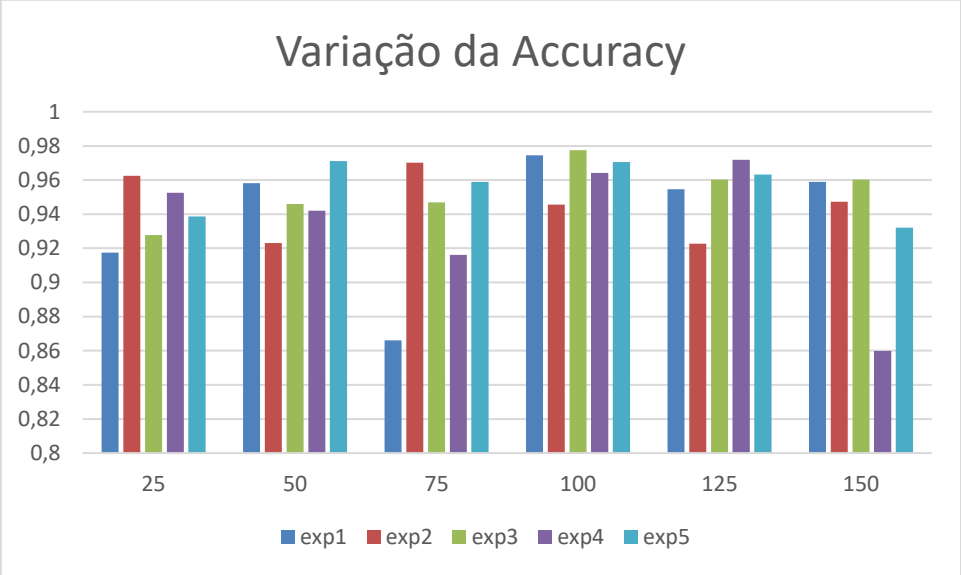


Figura 61 - Experiência - *Accuracy vs Batch Size*

De acordo com os dados recolhidos, não foi possível verificar um grande impacto causado pelo ajuste do valor do batch size, quer a nível de *accuracy*, quer a nível de tempo, sendo que a melhor relação utilizando a métrica de avaliação de parâmetros definida na secção[6.1] são os valores de 50, 100 e 125, para o conjunto de dados selecionado.

Tabela 9 - Experiência - *batch size* e variação de *accuracy* e tempo

Batch Size	Tempo	*Accuracy*	Fórmula f
25	20,8	0,9397	0,657812
50	18,4	0,9480	0,698238
75	19	0,9316	0,678084
100	18,8	0,9665	0,705362
125	19,2	0,9545	0,691239
150	19,6	0,9317	0,669491

6.3.2.5 *Learning Rate*

A verificação e avaliação de resultados para a influência da manipulação deste parâmetro utilizou como condições fixas o número de imagens (23147), a base de *Transfer Learning*, Mobilenet, com 10 *epochs*, um *batch size* de 50 imagens.

Relativamente à avaliação efetuada, foram elaboradas 20 experiências, distribuídas por 4 diferentes parâmetros, a *learning rate* de 0.01, 0.001, 0.0001 e 0.00001.

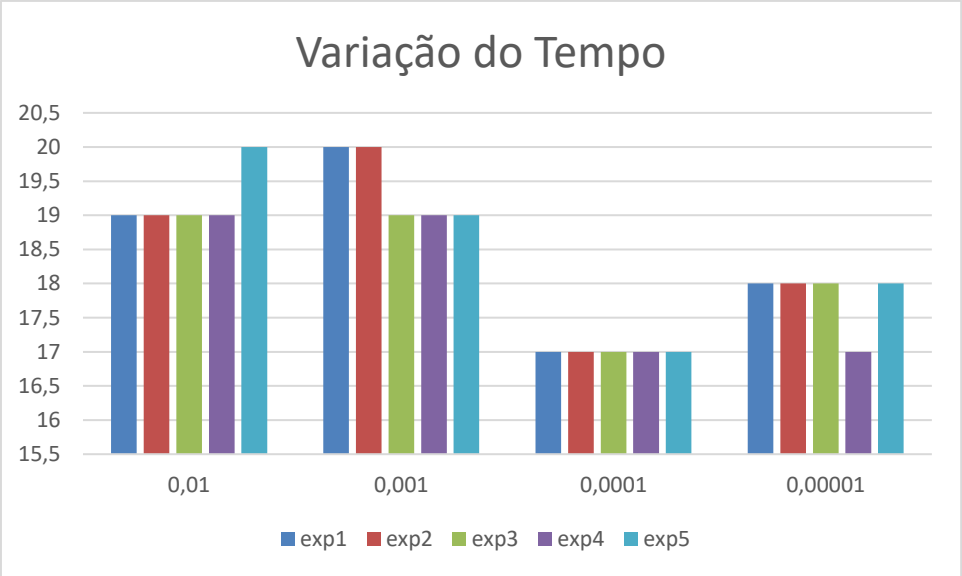


Figura 62 - Experiência - Tempo vs *Learning Rate*

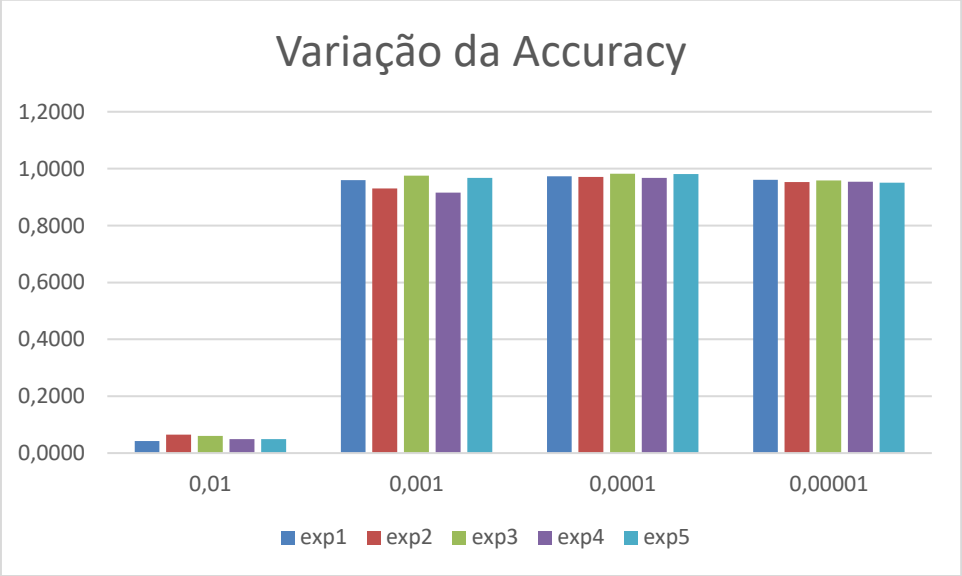


Figura 63 - Experiência - *Accuracy vs Learning Rate*

Como é possível verificar, relativamente ao impacto de tempo, as experiências com *learning rate* mais elevadas, demonstraram que o tempo de treino aumentou.

A nível de *accuracy*, ficou demonstrado que o *learning rate* de 0.01 é demasiado elevado e que provoca resultados muito baixos na *accuracy* do modelo.

Todas as experiências seguintes, demonstraram já bons índices de *accuracy* do algoritmo, sendo qualquer um destes uma opção válida para o conjunto de dados trabalhado.

Tabela 10 - Experiência - *Learning Rate* e variação da *accuracy* e tempo

Learning Rate	Tempo	Accuracy	Fórmula f
0,01	19,2	0,0534	0,040499
0,001	19,4	0,9498	0,664894
0,0001	17	0,9751	0,719681
0,00001	17,8	0,9550	0,693268

Depois de analisados e comparados, a melhor relação entre *accuracy* e tempo de treino foi obtida na experiência com o *learning rate* de 0.0001, onde a média dos intervalos demonstrou que foi possível aliar o tempo de treino inferior à melhor *accuracy*. Assim sendo para os dados utilizados, a experiência 0.0001 deverá ser a melhor opção.

6.3.2.6 Camadas Congeladas

Tal como explicado na secção 6.3.1.2.5, as camadas congeladas são as camadas configuradas para manterem os pesos previamente estabelecidos na CNN de base.

No caso desta experiência, a CNN é a Mobilenet, o *batch size* é de 50 imagens para um conjunto de dados com 23147 imagens, treinado no decorrer de 10 *epochs*.

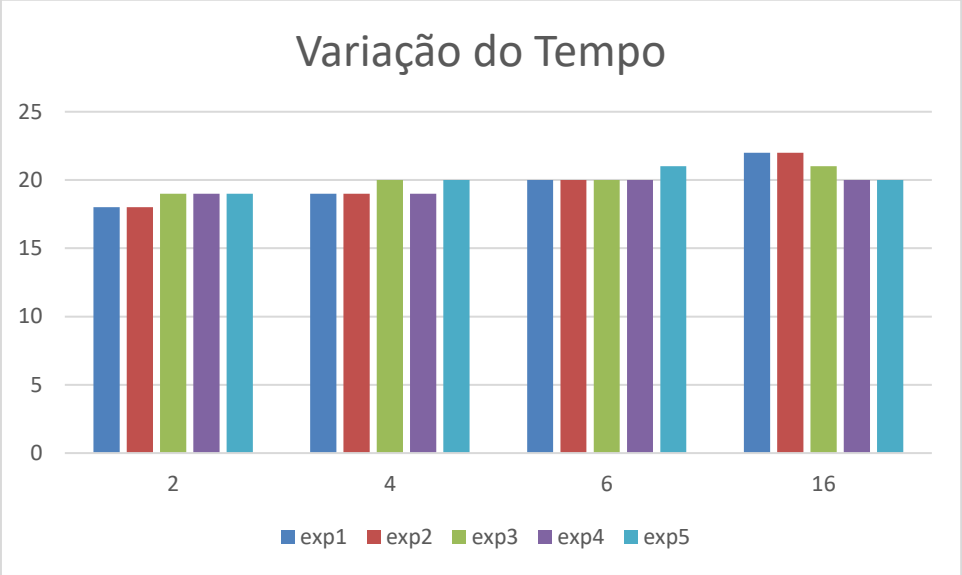


Figura 64 - Experiência - tempo *vs* camadas congeladas

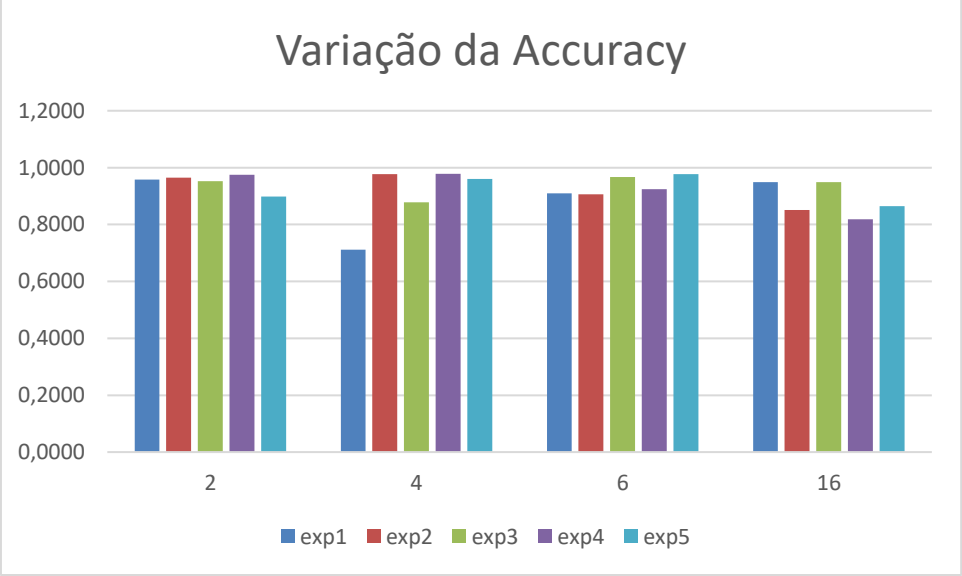


Figura 65 - Experiência - *accuracy vs* camadas congeladas

Analisando os dois gráficos obtidos das experiências realizadas é possível constatar que não existe uma grande diferença nos parâmetros recolhidos, quer em tempo de treino, quer em *accuracy* obtida na avaliação do modelo, no entanto é de salientar que se registou num dos resultados da primeira experiência com 4 *layers* congeladas, a *accuracy* obtida oscilou entre 71 e 97%, criando um diferencial superior ao encontrado nas outras experiências.

Ainda relativamente à análise dos gráficos, é possível verificar que a *accuracy* baixa ligeiramente no cenário das 16 camadas congeladas, sendo que o tempo de treino nesse cenário também aumenta ligeiramente comparando com as anteriores.

Tabela 11 - Experiência - Camadas congeladas e variação de *accuracy* e tempo

Layers	Tempo	*Accuracy*	Fórmula f
2	18,6	0,9498	0,699119
4	19,4	0,9010	0,653553
6	20,2	0,9370	0,667304
16	21	0,8866	0,620649

Analisando a tabela obtida, com as medidas de *accuracy* e tempo, e utilizando a fórmula de avaliação, ficou registado que a melhor opção para o treino com o conjunto de dados escolhido é a experiência de congelamento de 2 *layers*, onde se registou a média de tempo mais baixa e a *accuracy* media mais elevada face as outras experiências.

6.4 Validação de resultados

Depois da análise efetuada na secção de experiências 6.3.2, foi desenvolvida uma nova experiência que consiste na comparação entre o treino de um modelo utilizando as conclusões obtidas na secção de experiências, onde é feita a avaliação do modelo em 5 experiências entre o valor com melhor e segundo melhor resultado verificado nas experiências anteriores.

Desse modo, as características dos treinos será:

Tabela 12 Características do treino de validação

Modelo	*Epochs*	*BatchSize*	*LearningRate*	Camadas Congeladas	Modo
P1	10	100	0.0001	2	GPU
P2	5	50	0.00001	6	GPU

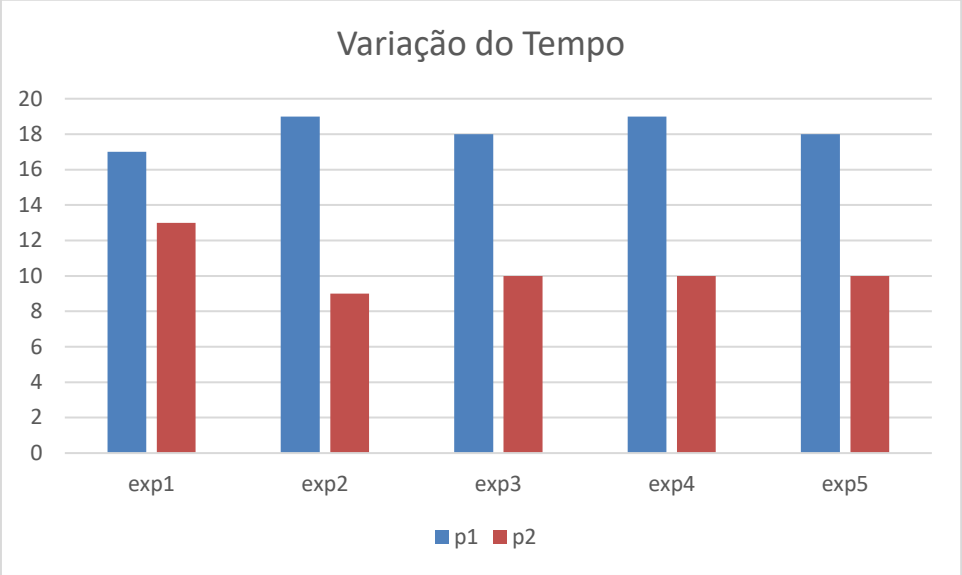


Figura 66 - Variação de tempo do modelo das Experiências P1 e P2

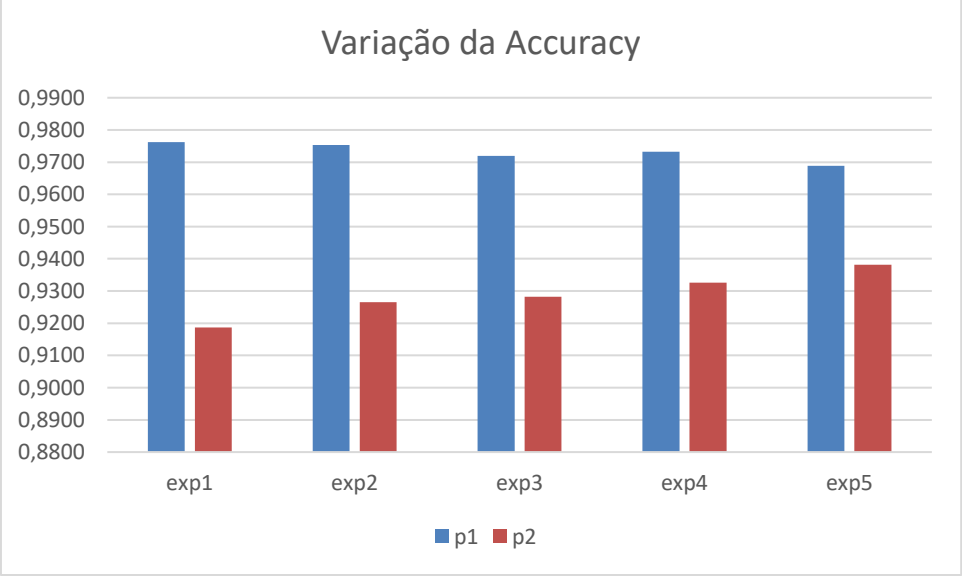


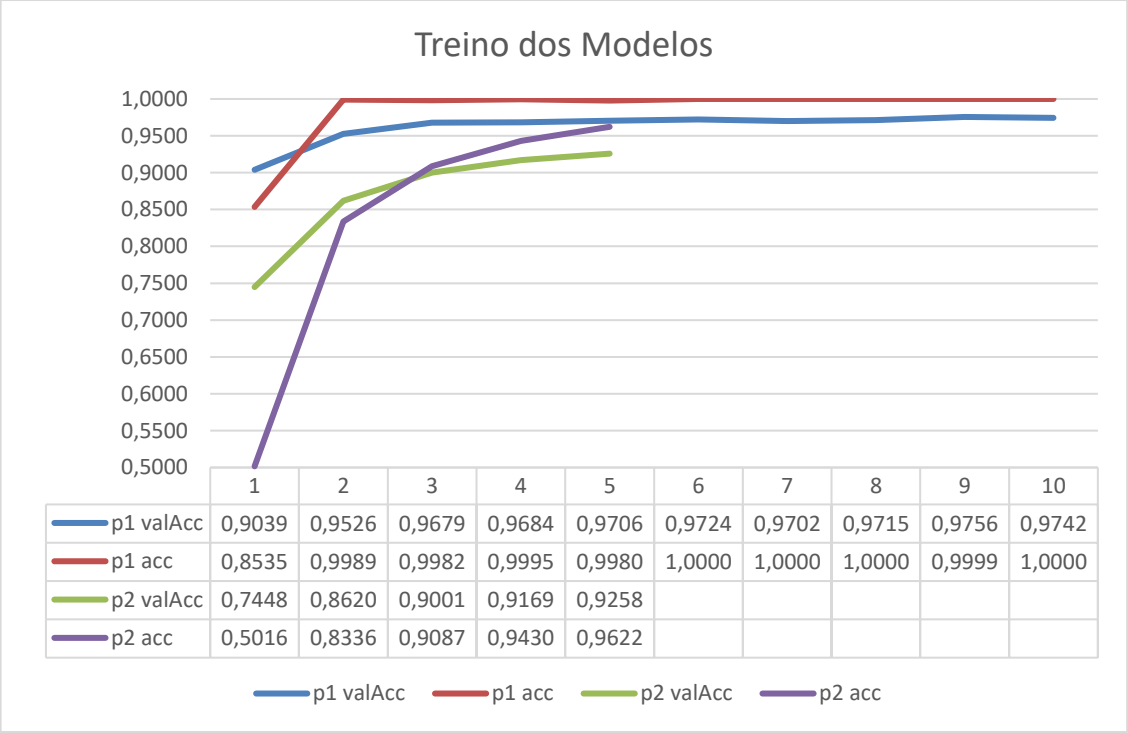
Figura 67 - Variação da *accuracy* do modelo das Experiências P1 e P2

Tabela 13 - Avaliação das experiências P1 e P2

EXP	Tempo	*Accuracy*	Fórmula f
p1	18,2	0,9731	0,681175
p2	10,4	0,9288	0,778757

De acordo com o resultado recolhido a partir de 5 treinos, sobre 2 experiências de parametrizações recolhidas dos dados anteriores, foi possível verificar que efetivamente ficou concluído que os parâmetros utilizados em ambas as definições, são adequados à obtenção de bons resultados, quer em tempo de treino, quer na *accuracy* obtida.

Por um lado, a experiência P2 traz como principal benefício o tempo de treino, já que consegue ser cerca de 42% inferior ao P1, no entanto o P1 apresenta uma accuracy do modelo superior ao P2 em cerca de quase 5%, atingindo uma *accuracy* de 97,31% na média de 5 experiências.



	1	2	3	4	5	6	7	8	9	10
p1 valAcc	0,9039	0,9526	0,9679	0,9684	0,9706	0,9724	0,9702	0,9715	0,9756	0,9742
p1 acc	0,8535	0,9989	0,9982	0,9995	0,9980	1,0000	1,0000	1,0000	0,9999	1,0000
p2 valAcc	0,7448	0,8620	0,9001	0,9169	0,9258					
p2 acc	0,5016	0,8336	0,9087	0,9430	0,9622					

Figura 68 - Evolução do treino das experiências P1 e P2

7 Conclusão

A elaboração deste trabalho permitiu um contributo para a área agrícola, nomeadamente para a deteção de doenças em plantas, via reconhecimento de imagens utilizando técnicas de *Deep Learning*.

Através das experiências realizadas, foi possível responder afirmativamente à questão inicial desta dissertação: "É possível criar uma aplicação, capaz de classificar uma doença presente numa planta, com um grau de confiança, capaz de ser auxiliar viável para um profissional do sector?"

Através dos resultados obtidos, foi possível concluir que utilizando o conjunto de dados disponibilizado pela PlantVillage, conseguiu atingir tempos de treino inferiores a 30 minutos, com taxas de acerto superiores a 90%, sendo possível obter um diagnóstico em menos de 1 minuto.

A aplicação móvel e o PlantAiPortal, ajudam a criar uma aplicabilidade ao trabalho desenvolvido em *Deep Learning*.

No entanto, este trabalho ainda poderia ser alvo de futuros desenvolvimentos, como a inclusão de novas funcionalidades que se demonstrem úteis aos utilizadores, tais como o acesso ao histórico de avaliações, secção de debate e apoio. Seria importante também realizar uma nova experiência de desenvolvimento de trabalho em *Deep Learning*, mas não utilizando o conjunto de dados disponibilizado pelo PlantVillage, mas antes, colecionando e catalogando novos dados para criar uma maior diversidade de doenças em plantas, bem como disponibilizar uma maior opção de escolha para tratamentos das doenças nas plantas do sistema.

Ainda no campo do desenvolvimento do modelo de reconhecimento, poderia ser desenvolvido um novo estudo, baseado em diferentes CNN, como as VGG, Imagenet, entre outros, bem como o desenvolvimento da construção de uma rede neuronal desde o estado zero, ou seja, definindo a constituição dos neurónios, camadas, pesos, a fim de se verificar se existem opções mais eficientes que a utilizada nesta dissertação.

Referências

[1] G. Agrios, *Plant Pathology*, 5th Editio. 2005.

[2] M. Sewell, “Machine Learning,” 2007.

[3] R. Wirth and J. Hipp, “CRISP-DM: Towards a Standard Process Model for Data Mining.”

[4] R. Moreira, “Doenças nas plantas: fique a conhecer as principais.” [Online]. Available: https://acientistaagricola.pt/controlo-de-pragas-e-doencas/. [Accessed: 26-Jan-2019].

[5] A. Brooks and A. Halstead, *Pragas e Doenças das Plantas*. Publicações Europa-America.

[6] M. R. Finckh, A. H. C. van Bruggen, and L. Tamm, *Plant Diseases and Their Management in Organic Agriculture*. .

[7] SAVE, “Society of American Value Engineers - About.” [Online]. Available: https://www.value-eng.org/page/AboutVE. [Accessed: 05-Feb-2019].

[8] P. Mulder, “Value Proposition Canvas by Alexander Osterwalder.” [Online]. Available: https://www.toolshero.com/marketing/value-proposition-canvas/. [Accessed: 05-Feb-2019].

[9] V. Allee, “ValueNet Works Fieldbook,” 2006.

[10] A. Serafim, “O Modelo de Cadeia de Valor de Michael Porter - Portal Gestão,” 2013. [Online]. Available: https://www.portal-gestao.com/artigos/6991-o-modelo-de-cadeia-de-valor-de-michael-porter.html. [Accessed: 19-Feb-2019].

[11] P. Koen, “Front End Innovation - What is the New Concept Development (NCD) model?” [Online]. Available: http://frontendinnovation.com/fei/what-is-the-new-concept-development-ncd-model. [Accessed: 05-Feb-2019].

[12] R. van Loon, “Machine learning explained: Understanding supervised, unsupervised, and reinforcement learning.” [Online]. Available: https://bigdata-madesimple.com/machine-learning-explained-understanding-supervised-unsupervised-and-reinforcement-learning/. [Accessed: 26-Jan-2019].

[13] C. Ledur, “Machine Learning: Introdução aos Métodos de Aprendizagem,” pp. 1–9, 2018.

[14] B. J. Gabriel Cánepa, “What You Need to Know About Machine Learning,” *Harv. Bus. Rev.*, no. April, pp. 121–130, 2000.

[15] W. S. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity,” *Bull. Math. Biophys.*, vol. 5, no. 4, pp. 115–133, 1943.

[16] J.V. Neumann, “The General and Logical Theory of Automata,” *Wiley, New York*, 1951.

[17] J.V. Neumann, “Probabilistic logics and the synthesis of reliable organisms from unreliable components, in Automata Studies,” *Princet. Univ. Press*, pp. 43–98, 1956.

[18] A. Kurenkov, “A ‘Brief’ History of Neural Nets and Deep Learning.” [Online]. Available:

http://www.andreykurenkov.com/writing/ai/a-brief-history-of-neural-nets-and-deep-learning/.

[19] D. O. Hebb, *The Organization of Behavior*. 1946.

[20] Pires João Miguel, “Aprendizagem Profunda: Estudo e Aplicações,” 2017.

[21] F. Rosenblatt, “THE PERCEPTRON: A PROBABILISTIC MODEL FOR INFORMATION STORAGE AND ORGANIZATION IN THE BRAIN 1.”

[22] Sebastian Raschka, “What is the difference between a Perceptron, Adaline, and neural network model?” [Online]. Available: https://sebastianraschka.com/faq/docs/diff-perceptron-adaline-neuralnet.html. [Accessed: 26-Jan-2019].

[23] M. Caraciolo, “Introduzindo Redes Neurais e Perceptron - Artificial Intelligence in Motion,” 2008. [Online]. Available: http://aimotion.blogspot.com/2008/12/artigo-introduzindo-redes-neurais-e_5497.html. [Accessed: 09-Feb-2019].

[24] K. Hager and R. Airola, “Image Classification , Deep Learning and Convolutional Neural Networks,” 2017.

[25] B. S. Phelan and J. Salomon, “Lung Cancer Detection using Deep Learning Lung Cancer Detection using Deep Convolutional Networks Final Year Project Report BSc in Computer Science,” Dublin Institute of Technology, 2018.

[26] D. Britz, “Recurrent Neural Networks Tutorial, Part 1 – Introduction to RNNs – WildML.” [Online]. Available: http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/. [Accessed: 04-Oct-2019].

[27] C. Szegedy *et al.*, “Going Deeper with Convolutions.”

[28] A. YA, “How To Teach A Computer To See With Convolutional Neural Networks.” [Online]. Available: https://towardsdatascience.com/how-to-teach-a-computer-to-see-with-convolutional-neural-networks-96c120827cd1. [Accessed: 04-Oct-2019].

[29] S. Das, “CNN Architectures: LeNet, AlexNet, VGG, GoogLeNet, ResNet and more,” 2017. [Online]. Available: https://medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5. [Accessed: 26-Jan-2019].

[30] Y. LeCun, “LeNet-5, convolutional neural networks.” [Online]. Available: http://yann.lecun.com/exdb/lenet/. [Accessed: 09-Feb-2019].

[31] W. Brand, *Pratical Artificial Intelligence*, vol. 91. 2017.

[32] M. Sewak, R. Karim, and P. Pujari, *Practical Convolutional Neural*. Packt Publishing, 2018.

[33] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” pp. 1–8.

[34] K. Simonyan and A. Zisserman, “Very Deep CNNS for Large-Scale Visual Recognition.” [Online]. Available: http://www.robots.ox.ac.uk/~vgg/research/very_deep/. [Accessed: 09-Feb-2019].

[35] T. Narayan, “Object Detection : A Comparison of performance of Deep learning Models on Edge Using Intel Movidius Neural Compute Stick and Raspberry PI3.” [Online]. Available: https://medium.com/intel-student-ambassadors/object-detection-a-comparison-of-performance-of-deep-learning-models-on-edge-using-intel-f66eb7f45b17. [Accessed: 04-Oct-2019].

[36] S. Bianco, R. Cadene, L. Celona, and P. Napoletano, “Benchmark Analysis of Representative Deep Neural Network Architectures.”

[37] “ImageNet - Image Collection.” [Online]. Available: http://www.image-net.org/download.php. [Accessed: 04-Oct-2019].

[38] “GitHub.” [Online]. Available: https://github.com/. [Accessed: 12-Oct-2019].

[39] “TensorFlow.” [Online]. Available: https://www.tensorflow.org/. [Accessed: 13-Feb-2019].

[40] “PyTorch.” [Online]. Available: https://pytorch.org/. [Accessed: 13-Feb-2019].

[41] Y. Jia *et al.*, “Caffe: Convolutional Architecture for Fast Feature Embedding,” Jun. 2014.

[42] “Microsoft Cognitive Toolkit.” [Online]. Available: https://www.microsoft.com/en-us/cognitive-toolkit/. [Accessed: 13-Feb-2019].

[43] J. Jokela, “Person counter using real-time object detection and a small neural network,” 2018.

[44] C. Vasudevan, *Concepts and programming in pytorch*. 2018.

[45] Skymind, “Comparison of AI Frameworks.” [Online]. Available: https://skymind.ai/wiki/comparison-frameworks-dl4j-tensorflow-pytorch. [Accessed: 26-Jan-2019].

[46] Anton Shaleynikov, “10 Best Frameworks and Libraries for AI - DZone AI,” 2018. [Online]. Available: https://dzone.com/articles/progressive-tools10-best-frameworks-and-libraries. [Accessed: 26-Jan-2019].

[47] V. Kovalev, A. Kalinovsky, and S. Kovalev, “Deep Learning with Theano, Torch, Caffe, TensorFlow, and Deeplearning4J: Which One Is the Best in Speed and Accuracy? Recognition of multi-drug resistant tubercolusis based on CT and X-ray image analysis View project UAV: back to base problem View project,” *XIII Int. Conf. Pattern Recognit. Inf. Process.*, no. October, 2016.

[48] A. Shatnawi, G. Al-Bdour, R. Al-Qurran, and M. Al-Ayyoub, “A comparative study of open source deep learning frameworks,” *2018 9th Int. Conf. Inf. Commun. Syst. ICICS 2018*, vol. 2018-Janua, pp. 72–77, 2018.

[49] Giancarlo Zaccone, *Getting Started With TensorFlow | TensorFlow*. 2017.

[50] G. Tanner, “Introduction to Deep Learning with Keras - Towards Data Science.” [Online]. Available: https://towardsdatascience.com/introduction-to-deep-learning-with-keras-17c09e4f0eb2. [Accessed: 09-Sep-2019].

[51] M. Yagües Gomà, “Image Recognition with Deep Learning Techniques and

TensorFlow," 2016.

[52] T. L. Saaty, "The Analytic Hierarchy Process," *McGraw-Hill, New York*, 1980.

[53] A. Ahmed, "Choosing a Machine Learning Framework in 2018 - Project AGI," 2018. [Online]. Available: https://agi.io/2018/02/09/survey-machine-learning-frameworks/. [Accessed: 26-Jan-2019].

[54] S. P. Mohanty, D. P. Hughes, and M. Salathé, "Using Deep Learning for Image-Based Plant Disease Detection," *Front. Plant Sci.*, vol. 7, 2016.

[55] S. J. Pethybridge, "Leaf Doctor." [Online]. Available: https://play.google.com/store/apps/details?id=com.leafdoctor&hl=en_US. [Accessed: 26-Jan-2019].

[56] Consortium and Plantnet-project, "Pl@ntNet - https://identify.plantnet-project.org." [Online]. Available: https://identify.plantnet-project.org.

[57] C. G. Xu, H and Zhao, "Leafweb Gitlab repository." [Online]. Available: https://gitlab.com/huix/leaf-disease-plant-village. [Accessed: 13-Oct-2019].

[58] C. G. Xu, H and Zhao, "Leaf Diagnosis Demo - Diagnosis plant with AI." [Online]. Available: https://leafweb.herokuapp.com/. [Accessed: 13-Oct-2019].

[59] M. Levit, "Importance of Version Control and Why You Need It - WeAreServian - Medium." [Online]. Available: https://medium.com/weareservian/importance-of-version-control-and-why-you-need-it-aae53dac208a. [Accessed: 28-Sep-2019].

[60] "Git Workflows | Programster's Blog." [Online]. Available: http://blog.programster.org/git-workflows. [Accessed: 28-Sep-2019].

[61] "Keras documentation." [Online]. Available: https://keras.io/. [Accessed: 12-Oct-2019].

[62] NumPy, "NumPy." [Online]. Available: https://numpy.org/. [Accessed: 12-Oct-2019].

[63] T. Mitchell, *Machine Learning*, 1 ed. McGraw-Hill Science/Engineering/Math, 1997.

[64] Scikit-learn, "scikit-learn: machine learning in Python — scikit-learn 0.21.3 documentation." [Online]. Available: https://scikit-learn.org/stable/. [Accessed: 12-Oct-2019].

[65] "HDF5 for Python." [Online]. Available: https://www.h5py.org/. [Accessed: 12-Oct-2019].

[66] "Matplotlib: Python plotting — Matplotlib 3.1.1 documentation." [Online]. Available: https://matplotlib.org/. [Accessed: 12-Oct-2019].

[67] "Pillow — Pillow (PIL Fork) 6.2.0 documentation." [Online]. Available: https://pillow.readthedocs.io/en/stable/. [Accessed: 12-Oct-2019].

[68] "Python Data Analysis Library — pandas: Python Data Analysis Library." [Online]. Available: https://pandas.pydata.org/. [Accessed: 12-Oct-2019].

[69] "PlantVillage." [Online]. Available: https://plantvillage.psu.edu/. [Accessed: 24-Feb-

2019].

[70] PlantVillage, “PlantVillage Disease Classification Challenge.” [Online]. Available: https://www.crowdai.org/challenges/1. [Accessed: 24-Feb-2019].

[71] D. P. Kingma and J. Lei Ba, “ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION.”

[72] Keras Documentation, “Model (functional API) - Keras Documentation.” [Online]. Available: https://keras.io/models/model/. [Accessed: 06-Oct-2019].

[73] E. Allibhai, “Hold-out vs. Cross-validation in Machine Learning - Eijaz Allibhai - Medium.” [Online]. Available: https://medium.com/@eijaz/holdout-vs-cross-validation-in-machine-learning-7637112d3f8f. [Accessed: 12-Oct-2019].

[74] E. Allibhai, “Holdout vs. Cross-validation in Machine Learning – Eijaz Allibhai – Medium.” [Online]. Available: https://medium.com/@eijaz/holdout-vs-cross-validation-in-machine-learning-7637112d3f8f. [Accessed: 17-Feb-2019].

[75] J. Jordan, “Setting the learning rate of your neural network.” [Online]. Available: https://www.jeremyjordan.me/nn-learning-rate/. [Accessed: 22-Sep-2019].

[76] J. Brownlee, “Gentle Introduction to the Adam Optimization Algorithm for Deep Learning.” [Online]. Available: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/. [Accessed: 22-Sep-2019].

[77] “Simple guide to confusion matrix terminology.” [Online]. Available: https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/. [Accessed: 22-Sep-2019].

Anexo I – Definição da .Net API

Tabela 14 - .NetCore API - Plantas

Serviço	Plant
Endpoint	/api/Plant
Verbo	GET
Descrição	Este método devolve uma lista de plantas
Parâmetros	--

Response	[
	{
	"plantId”: int,
	"plantName”: string
	}
	]

Endpoint	/api/Plant/{id}
Verbo	GET
Descrição	Este método devolve uma planta com base no Id
Parâmetros	id - id da planta

Response	{
	"plantId”: int,
	"plantName”: string
	}

Tabela 15- .NetCore API - Doenças

Serviço	Disease
Endpoint	/api/Disease
Verbo	GET
Descrição	Este método devolve uma lista de doenças de plantas
Parâmetros	--

Response	[
	{
	"diseaseId”: int,
	"diseaseName”: string,
	"info”: string,
	"code”: string,
	"plantId”: int,
	"plantName”: string,
	"isHealthy”: bool
	}
	]

Endpoint	/api/Disease/{id}

Verbo	GET
Descrição	Este método devolve uma doença de plantas com base no Id
Parâmetros	id - id da doença

Response	{
	"diseaseId”: int,
	"diseaseName”: string,
	"info”: string,
	"code”: string,
	"plantId”: int,
	"plantName”: string,
	"isHealthy”: bool
	}

Endpoint	/api/Disease/Plant/{id}
Verbo	GET
Descrição	Este método devolve uma lista de doenças de plantas
Parâmetros	id - id da planta

Response	[
	{
	"diseaseId”: int,
	"diseaseName”: string,
	"info”: string,
	"code”: string,
	"plantId”: int,
	"plantName”: string,
	"isHealthy”: bool
	}
	]

Tabela 16 - .NetCore API - Tratamentos

Serviço	Treatment
Endpoint	/api/Treatment
Verbo	GET
Descrição	Este método devolve uma lista de tratamentos.
Parâmetros	--
Response	zipFile
Endpoint	/api/Treatment/{id}

Verbo	GET
Descrição	Este método devolve um de tratamento.
Parâmetros	id - id do tratamento

Response	{
	"title”: string,
	"disease”: string,
	"treatmentTopics": [
	{
	"type”: int,
	"text”: string,
	"index": int
	}
	]
	}

Endpoint	/api/Treatment/{id}/byDiseaseId
Verbo	GET
Descrição	Este método devolve um de tratamento usando o id de uma doença.
Parâmetros	id - id da doença da planta

Response	{
	"title”: string,
	"disease”: string,
	"treatmentTopics": [
	{
	"type”: int,
	"text”: string,
	"index": int
	}
	]
	}

Endpoint	/api/Treatment/{id}/bycode
Verbo	GET
Descrição	Este método devolve um de tratamento usando o código associado a doença.
Parâmetros	code - código da doença da planta

Response	{
	"title”: string,
	"disease”: string,
	"treatmentTopics": [
	{
	"type”: int,
	"text”: string,
	"index": int
	}
	]
	}

Tabela 17 - .NetCore API - Keras

Serviço	Keras
Endpoint	/api/Keras/GetPhotoCollection
Verbo	GET
Descrição	Este método devolve um arquivo ZIP com todas as imagens configuradas para entrar no proximo treino do modelo.
Parâmetros	--
Response	zipFile
Endpoint	/api/Keras
Verbo	GET
Descrição	Este método devolve uma lista de detalhes de treinos de modelos.
Parâmetros	--

Response	[
	{
	"aiModelId”: int,
	"created”: datetime,
	"modelPb”: string,
	"modelTxt”: string,
	"status”: int,
	"startTrain”: datetime,
	"endTrain”: datetime,
	"output”: string,
	"numberOfImages”: int,
	"args”: string,
	"accuracy”: string
	}
	]

Endpoint	/api/Keras/{id}
Verbo	GET
Descrição	Este método devolve o detalhe de um resultado de treino.
Parâmetros	id - id do modelo

Response	{
	"aiModelId”: int,
	"created”: datetime,
	"modelPb”: string,
	"modelTxt”: string,
	"status”: int,
	"startTrain”: datetime,
	"endTrain”: datetime,
	"output”: string,
	"numberOfImages”: int,
	"args”: string,
	"accuracy”: string
	}

Endpoint	/api/Keras
Verbo	POST
Descrição	Este método cria um registo de detalhe de informação sobre um modelo de treino.

Parâmetros	{
	"status”: int,
	"startDate”: string,
	"endDate”: string,
	"modelName”: string,
	"acc”: int,
	"valAcc”: int,
	"loss”: int,
	"valLoss”: int,
	"parameters": {
	"epoch”: int,
	"batchSize”: int,
	"nrClasses”: int,
	"learningRate”: int,
	"popLayers”: int,
	"freezingLayers”: int,
	"baseFolder”: string,
	"countImages”: int,
	"stepsPerEpoch”: int,
	"validationSteps”: int
	}
	}

Response	--

Tabela 18 - .NetCore API - AI Core

Serviço	AiCore
Endpoint	/api/AiCore
Verbo	POST
Descrição	Este método devolve a classificação de uma imagem

Parâmetros	{
	"saveToDatabase”: bool,
	"isMock”: bool,
	"image”: string
	}

Response	{
	"imageProcessed”: bool,
	"imageName”: string,
	"classificationTime”: string,
	"classification": [
	{
	"diseaseId”: int,
	"plantId”: int,
	"plantName”: string,
	"diseaseName”: string,
	"diseaseCode”: string,
	"classification”: int,
	"isHealthy”: bool
	}
	],
	"highClassification": {
	"diseaseId”: int,
	"plantId”: int,
	"plantName”: string,
	"diseaseName”: string,
	"diseaseCode”: string,
	"classification”: int,
	"isHealthy”: bool
	}
	}

Anexo II – Definição da Flask API

Tabela 19 – Flask Python API - Keras

Serviço	Keras
Endpoint	/changeKerasModel
Verbo	POST
Descrição	Este método serve para alterar o modelo de classificação em utilização.

Parâmetros	{
	"modelName": string
	}

Response	["true"]|["false"]
Endpoint	/reloadModel
Verbo	POST
Descrição	Este método serve para efectuar um refresh ao modelo existente, carregando-o novamente.
Parâmetros	--
Response	["true"]
Endpoint	/predict
Verbo	POST
Descrição	Este método devolve um de tratamento usando o id de uma doença.

Parâmetros	{
	"image": string - imagem em base64
	}

Response	{
	"Prediction" : [
	"Key": string,
	"Value": float
	]
	}

Anexo III – Packages e Versões utilizados no ambiente Python

Packages e Versões		
absl-py==0.8.0	jsonschema==3.0.2	pyrsistent==0.15.4
astor==0.8.0	jupyter-client==5.3.3	python-dateutil==2.8.0
attrs==19.1.0	jupyter-core==4.5.0	pytz==2019.2
backcall==0.1.0	Keras==2.3.0	pywin32==223
bleach==3.1.0	Keras-Applications==1.0.8	pywinpty==0.5.5
blis==0.2.4	Keras-Preprocessing==1.1.0	PyYAML==5.1.2
boto==2.49.0	kiwisolver==1.1.0	pyzmq==18.1.0
boto3==1.9.240	lightgbm==2.3.0	requests==2.22.0
botocore==1.12.240	Markdown==3.1.1	s3transfer==0.2.1
certifi==2019.9.11	MarkupSafe==1.1.1	scikit-learn==0.21.3
chardet==3.0.4	matplotlib==3.1.1	scipy==1.3.1
Click==7.0	mistune==0.8.4	Send2Trash==1.5.0
colorama==0.4.1	mkl-fft==1.0.14	six==1.12.0
cycler==0.10.0	mkl-random==1.1.0	sklearn==0.0
cymem==2.0.2	mkl-service==2.3.0	smart-open==1.8.4
Cython==0.29.13	mmdnn==0.2.5	spacy==2.1.8
decorator==4.4.0	murmurhash==1.0.2	srsly==0.1.0
defusedxml==0.6.0	nbconvert==5.6.0	stop-words==2018.7.23
docutils==0.15.2	nbformat==4.4.0	tensorboard==2.0.0
entrypoints==0.3	nltk==3.4.5	tensorflow==2.0.0
Flask==1.1.1	notebook==6.0.1	tensorflow-estimator==2.0.0
gast==0.2.2	numpy==1.16.5	termcolor==1.1.0
gensim==3.8.1	opt-einsum==3.1.0	terminado==0.8.2
google-pasta==0.1.7	pandas==0.25.1	testpath==0.4.2
grpcio==1.24.0	pandocfilters==1.4.2	thinc==7.0.8
h5py==2.10.0	parso==0.5.1	tornado==6.0.3
idna==2.8	pickleshare==0.7.5	tqdm==4.36.1
ipykernel==5.1.2	Pillow==6.2.0	traitlets==4.3.2
ipython==7.8.0	plac==0.9.6	urllib3==1.25.6
ipython-genutils==0.2.0	preshed==2.0.1	wasabi==0.2.2
itsdangerous==1.1.0	prometheus-client==0.7.1	wcwidth==0.1.7
jedi==0.15.1	prompt-toolkit==2.0.9	webencodings==0.5.1
Jinja2==2.10.1	protobuf==3.9.2	Werkzeug==0.16.0
jmespath==0.9.4	Pygments==2.4.2	wincertstore==0.2
joblib==0.14.0	pyparsing==2.4.2	wrapt==1.11.2