DM

Interface Gráfica para Desenvolvimento de Redes Neuronais Convolucionais Probabilísticas

DISSERTAÇÃO DE MESTRADO

Aníbal João Lopes Chaves
MESTRADO EM ENGENHARIA INFORMÁTICA

UNIVERSIDADE da MADEIRA
A Nossa Universidade
www.uma.pt

julho | 2022

Interface Gráfica para Desenvolvimento de Redes Neuronais Convolucionais Probabilísticas

DISSERTAÇÃO DE MESTRADO

Aníbal João Lopes Chaves
MESTRADO EM ENGENHARIA INFORMÁTICA

ORIENTAÇÃO
Fernando Manuel Rosmaninho Morgado Ferrão Dias

COORIENTAÇÃO
Fábio Rúben Silva Mendonça

Interface Gráfica para desenvolvimento de Redes Neuronais Convolucionais Probabilísticas

Aníbal João Lopes Chaves

Supervisor:
Prof. Dr. Fernando Manuel Rosmaninho Morgado Ferrão Dias

Co-Supervisor:
Prof. Dr. Fábio Rúben Silva Mendonça

Mestrado em Engenharia Informática

Funchal – Portugal
Julho 2022

Abstract

Through the development of artificial intelligence, some capabilities of human beings have been replicated to computers. Among the models developed, Convolutional Neural Networks stand out considerably because they make it possible for systems to have inherent capabilities of humans, such as pattern recognition in images and signals. However, conventional systems are based on deterministic models, which are unable to express the epistemic uncertainty of their predictions. The alternative consists in the use of probabilistic models although these are considerably more difficult to develop. In order to address the problems related to the development of probabilistic networks and the choice of the network architecture, in this dissertationthe development of an application is proposed , which allows the user to choose the desired architecture and obtain the model already trained for the given data. This application named "Graphical Interface for Probabilistic Neural Networks" gives the user the possibility to use the most common Convolutional Neural Networks for different data sets, being the networks adapted to the developed probabilistic model. Contrary to existing models for generic use, which are deterministic and already pre-trained on databases to be used in transfer learning, the approach followed in this work creates the network layer by layer, with training performed on the provided data, originating a specific model for the data in question.

Keywords: Artificial Intelligence; Graphical Interface; Probabilistic Convolutional Neural Network; Probabilistic Model and Deterministic Model

Resumo

Através do desenvolvimento da inteligência artificial, algumas capacidades dos seres humanos têm sido replicadas para os computadores. Entre os modelos desenvolvidos, destacam-se consideravelmente as Redes Neuronais Convolucionais, pois tornam possível aos sistemas terem capacidades inerentes dos humanos tais como o reconhecimento de padrões em imagens e sinais. Todavia, os sistemas convencionais são baseados em modelos determinísticos, sendo estes incapazes de expressar a incerteza epistêmica das previsões. A alternativa consiste no uso de modelos probabilísticos embora estes sejam consideravelmente mais difíceis de desenvolver. Para abordar os problemas relativos ao desenvolvimento de redes probabilísticas e à escolha da arquitetura da rede, nesta dissertação é proposto o desenvolvimento de uma aplicação que permite ao utilizador escolher a arquitetura desejada e obter o modelo já treinado para os dados fornecidos. Esta aplicação "Interface Gráfica para Redes Neuronais Probabilísticos" dá ao utilizador a possibilidade de poder utilizar as Redes Neuronais Convolucionais mais comuns para diferentes conjuntos de dados, sendo as redes adaptadas ao modelo probabilístico desenvolvido. Contrariamente aos modelos existentes para uso genérico que são determinísticos e já pré-treinados em conjuntos de dados para depois serem usados em $transfer\ learning$, a abordagem seguida neste trabalho cria a rede camada a camada, sendo efetuado o treino nos dados fornecidos, originando um modelo específico para os dados em questão.

Palavras-chave: Inteligência Artificial; Interface Gráfica; Rede Neuronal Convolucional Probabilística; Modelo Probabilístico e Modelo Determinístico

Agradecimentos

Gostaria de agradecer ao professor Morgado Dias por todas as orientações dadas, a sua experiência e os seus conhecimentos revelaram-se fundamentais para o desenvolvimento da dissertação.

Ao professor Fábio Mendonça pela sua disponibilidade. Forneceu inúmeras soluções tornando desta forma exequível todo o trabalho desenvolvido.

Um agradecimento aos meus colegas de curso e amigos: André Gonçalves e Diogo Freitas, trabalhamos juntos em diversas situações e trocamos conhecimentos por inúmeras ocasiões.

Por fim, um agradecimento especial à minha família por toda a ajuda dada.

Conteúdo

Lista de Figuras

Lista de Tabelas

Acrónimos

Acc	Exatidão.
API	Application Programming Interface.
AQ	Atributo de Qualidade.
AU	Avaliação da Usabilidade.
AUC	Área Sob a Curva.
CNN	Convolutional Neural Network.
CPU	Central Processing Unit.
CUDA	Compute Unified Device Architecture.
cuDNN	NVIDIA CUDA Deep Neural Network.
DCU	Diagrama de Casos de Utilização.
DER	Diagrama Entidade Relação.
DFD	Diagrama de Fluxo de Dados.
DL	Deep Learning.
ELBO	Evidence Lower Bound.
ER	Engenharia de Requisitos.
FA	Função de Ativação.
FN	Falso Negativo.
FP	Falso Positivo.
GPS	Global Positioning System.
GPU	Graphics Processing Unit.
IA	Inteligência Artificial.
IG	Interface Gráfica.
IV	Inferência Variacional.
KLD	Kullback-Leibler-Distribuition.

LR	Levantamento de Requisitos.
MD	Modelo Determinístico.
ML	Machine Learning.
MN	Mapa de Navegação.
MP	Modelo Probabilístico.
MSE	Erro Quadrático Médio.
NPV	Valor Preditivo Negativo.
PPV	Valor Preditivo Positivo.
QAW	Quality Attribute Workshop.
RAM	Random Access Memory.
ReLU	Rectified Linear Unit.
RF	Requisito Funcional.
RMSProp	Root Mean Square Prop.
RN	Rede Neuronal.
RNA	Rede Neuronal Artificial.
RNF	Requisito Não Funcional.
Sen	Sensibilidade.
Spe	Especificidade.
TF	TensorFlow.
TU	Teste de usabilidade.
V-RAM	Video Random Access Memory.
VN	Verdadeiro Negativo.
VP	Verdadeiro Positivo.

Capítulo 1

Introdução

Na introdução apresentam-se os objetivos, a motivação e as questões de investigação. Descrevem-se os conceitos associados aos Modelos Probabilísticos (MPs) e Modelos Determinísticos (MDs), expoem-se as principais diferenças entre os mesmos e apresenta-se um exemplo de aplicabilidade. Por fim é apresentada a estrutura da dissertação.

1.1 Motivação

Os seres humanos conseguem identificar padrões e objetos, podendo facilmente reconhecer um relógio, uma cadeira, um cão ou um gato. No entanto, para um modelo computacional tal é um desafio considerável.

Através da Inteligência Artificial (IA) [1] é possível aos sistemas computacionais efetuarem o reconhecimento de padrões. Sendo necessário desenvolver um modelo capaz de aprender e, através deste modelo, conseguir que o método de análise do sistema computacional se aproxime da forma como o cérebro humano realiza a análise, permitindo assim que os computadores aprendam a reconhecer padrões.

Machine Learning (ML) [2], define-se como um tipo de IA pois torna possível às aplicações de *software* possuírem uma precisão na previsão de resultados elevada, mesmo não tendo sido devidamente programadas para tal, o ML, traduz-se na execução de algoritmos, criam-se assim de forma automática modelos capazes de dispor de conhecimento tendo como base um conjunto de dados. Devem-se treinar sistemas computacionais, concedendo-lhes um conjunto de dados adequado, para que desta forma o algoritmo "aprenda", ou seja, ajustando-se iterativamente. Após efetuar-se o treino, o modelo criado consegue efetuar previsões com um elevado valor de assertividade.

Na figura 1.1 apresenta-se o resumo histórico das Redes Neuronais Artificiais (RNAs), como podemos verificar as primeiras RNAs remontam à década de quarenta. O poder computacional durante muito anos foi uma limitação, mas com o surgimento de novas bibliotecas aliados aos avanços de poder computacional esta

limitação foi sendo ultrapassada, permitindo desta forma nestes últimos anos uma evolução significativa na área do Deep Learning (DL).

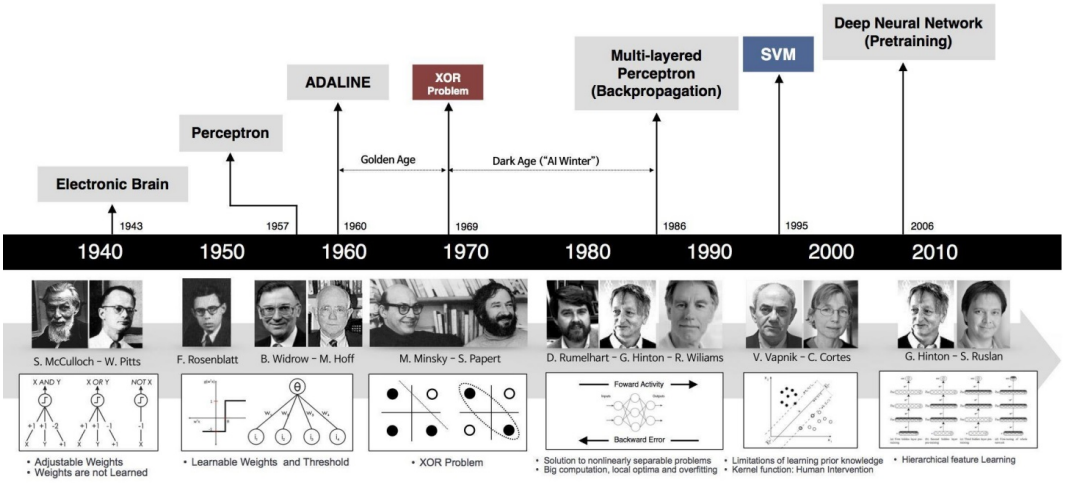


Figura 1.1: Resumo histórico das RNAs [3].

Apesar do número significativo de algoritmos desenvolvidos para DL, as RNAs têm constantemente demonstrado serem as mais adaptáveis, especialmente com o surgimento do DL [4]. Teoricamente, pode-se utilizar as Convolutional Neural Networks (CNNs) (baseadas em camadas *dense*) para a deteção de padrões complexos como os presentes em imagens, mas, na prática, poderá ser impossível, pois o número de cálculos necessários é extremamente elevado. Tal ocorre, pois, numa RNA convencional, cada pixel está ligado a cada neurónio. Neste caso, a carga computacional adicionada torna a rede inviável para análise das imagens [5].

As CNNs tornaram o reconhecimento de padrões em imagem exequível. Ao se considerar uma imagem, a proximidade dos pixeis tem uma forte correlação e as CNNs usam especificamente essa relação. Pixeis próximos têm uma probabilidade mais elevada de estarem relacionados do que pixeis mais afastados. A convolução resolve o problema da necessidade do elevado número de conexões que ocorre nas RNAs convencionais, sendo possível o processamento de imagens por meio da filtragem das conexões por proximidade. Numa determinada camada, ao invés de ligar cada entrada a cada neurónio, as CNNs restringem as conexões intencionalmente para que qualquer neurónio aceite as entradas apenas a partir de uma pequena parte da camada anterior (como, por exemplo, a partir de 5×5 pixeis ou 3x3 pixeis).

A evolução das Graphics Processing Unit (GPU) e a disponibilização de bibliotecas, tais como o TensorFlow (TF) [6] e Keras [7], facilitaram substancialmente o desenvolvimento de CNNs. Atualmente os requisitos ao nível de *hardware* e *software* estão ao alcance de computadores convencionais.

Apesar da elevada precisão que os modelos baseados em DL permitem alcançar, o problema associado ao nível de confiança que o utilizador tem na previsão da rede

ainda se mantém. Tal pode ser resolvido através do uso de MPs pois permitem expressar a incerteza epistémica nas previsões geradas pelos modelos. Estes modelos são computacionalmente exigentes e só recentemente se tornaram viáveis com o desenvolvimento da Inferência Variacional (IV) [8]. O desenvolvimento de CNNs probabilísticas está mais simples, através do surgimento dos modelos baseados nos estimadores como o Flipout e a sua inclusão no TF probability [9].

1.2 Modelos Probabilísticos e Determinísticos

Os MDs com arquiteturas convencionais estão disponíveis com otimização realizada em conjuntos de dados de grande dimensão, permitindo usar técnicas de *transfer learning* para obter elevada precisão. Todavia, os MPs ainda não possuem esta opção de *transfer learning* pois tipicamente as distribuições que estes modelos aprendem estão fortemente associadas aos dados usados para o treino dos modelos. Desta forma, torna-se necessário realizar o treino por completo dos modelos nos conjuntos de dados. Todavia, os MPs têm a grande vantagem de para além da previsão, conseguirem também dar resposta à incerteza epistémica das previsões, indicando o grau de fiabilidade.

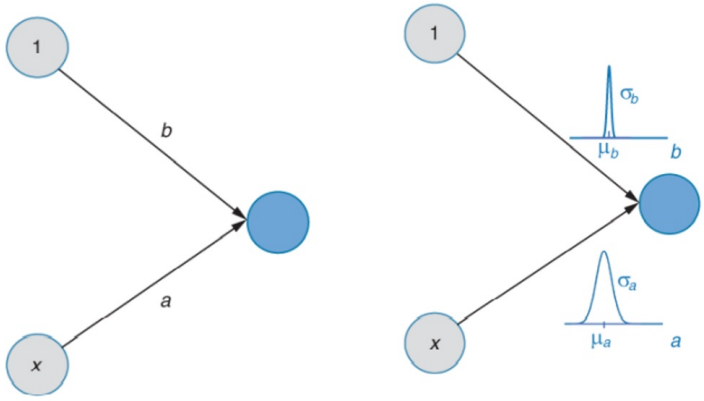


Figura 1.2: MD / MP [9].

No MD representado à esquerda da figura 1.2, a e b são parâmetros treinados pela rede durante a fase de treino, na fase de teste estes parâmetros passam a ser constantes. No MP representado à direita da figura 1.2, os parâmetros são definidos por uma distribuição probabilística, neste caso uma gaussiana, sendo os parâmetros da gaussiana (média e desvio de padrão) ajustados durante a fase de treino. Particularmente, durante esta fase é realizada uma amostragem a cada distribuição para definir o valor dos parâmetros, consequentemente, cada vez que se realizar previsões serão sempre produzidos resultados ligeiramente diferentes. A variância dos

resultados produzidos dirá qual o grau da incerteza epistémica, algo que só os MPs conseguem indicar, apresenta-se na figura 1.3 exemplos de duas classes apresentadas a um modelo, sendo uma conhecida por este (presente no treino do modelo) e a outra desconhecida (não estava presente no treino do modelo).

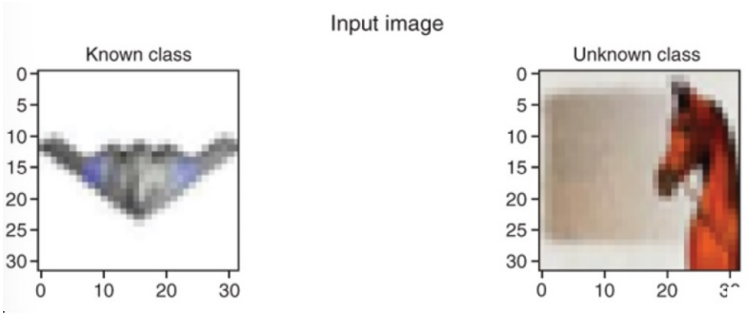


Figura 1.3: Dois exemplos de classes [9].

A diferença entre os resultados produzidos por um MD e um MP é enfatizada pelo seguinte cenário. Considere-se uma RNA determinística, treinada para distinguir entre nove classes. A esta apresenta-se um exemplo de uma classe conhecida e um exemplo de uma classe desconhecida, representadas na imagem da figura 1.3. Os resultados do MD são apresentados na figura 1.4. Na figura observa-se que a resposta do MP é quase 100% na classe correta quando é apresentada uma imagem de uma classe conhecida, representada no lado esquerdo da figura 1.4 porém, quando é apresentada uma imagem de uma classe que o modelo não conhece, ainda assim é apresentado uma resposta que é substancial em duas classes, representado pelo lado direito da figura 1.4.

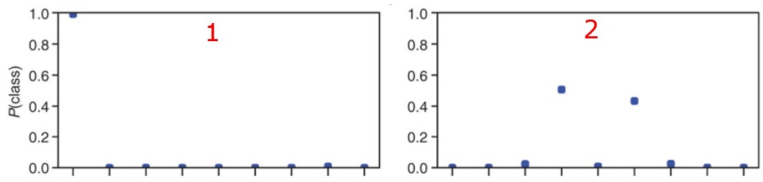


Figura 1.4: Resultado do MD [9].

Considere-se agora o mesmo cenário, mas para uma RNA probabilística, treinada para distinguir entre nove classes, sendo apresentados os resultados na figura 1.5. Analisando a figura, observa-se que o MP apresenta elevada dispersão nos resultados quando efetua análise de um exemplo que não faz parte de nenhuma das nove classes conhecidas, sugerindo que o utilizador não deve confiar nas previsões realizadas.

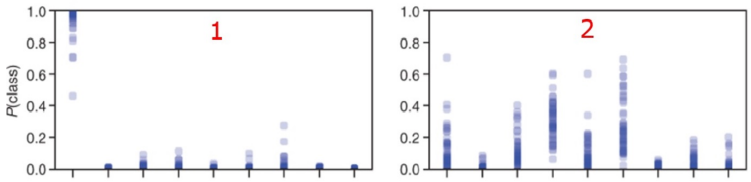


Figura 1.5: Resultado do MP [9].

1.3 Objetivos

Um dos maiores entraves ao uso dos modelos de DL por utilizadores sem experiência é a complexidade exigida do ponto de vista da programação. Tendo em conta o grande interesse que o DL baseado em CNNs probabilísticas tem tido recentemente, especialmente por parte dos programadores interessados em realizar aplicações para o reconhecimento de padrões, torna-se necessário obter uma alternativa simples e de fácil utilização que permita criar os modelos.

O objetivo principal desta dissertação foi o desenvolvimento de uma Interface Gráfica (IG) que permite a qualquer utilizador sem conhecimentos prévios na área da DL o desenvolvimento de MPs baseados em CNNs. Desta forma, duas abordagens foram desenvolvidas; uma contendo modelos com arquiteturas padrão na área do DL e outra onde o utilizador poderá criar um modelo camada a camada com a arquitetura desejada, ou seja, o utilizador tem a possibilidade de selecionar a quantidade de camadas que deseja, criando assim o seu próprio MP treinado nos dados fornecidos pelo utilizador.

A IG apresenta múltiplas opções que guiam o utilizador no processo de seleção da arquitetura da RNA a ser desenvolvida, o treino da mesma e a obtenção do modelo treinado de forma simples e intuitiva. O utilizador deverá conseguir realizar o treino dos modelos com um pequeno número de escolhas, sendo apenas necessário carregar os conjuntos de dados, selecionar o modelo pretendido e iniciar o treino do mesmo. A IG permite que o desenvolvimento de uma CNN probabilística seja feita em três passos: inserção dos parâmetros desejados para o treino; escolha da arquitetura; visualização dos resultados.

Todo o programa foi desenvolvido em Python 3, incluindo a IG e o desenvolvimento dos modelos, está disponível em fonte aberta e acessível no repositório público: (https://github.com/achaves37/Interface-Grafica-para-desenvolvimento-de-Redes-Neuronais-Convolucionais-Probabiliticas/). São disponibilizadas duas versões, uma denominada por "TesteInterface" e outra denominada por "Interface", nesta última o utilizador terá a necessidade de carregar os seus conjuntos de dados, na versão denominada por "TesteInterface" já estão incluídos alguns conjuntos de dados para teste. Todo o código está devidamente documentado e organizado para possibilitar acrescentar/editar algum modelo de uma forma simples e direta.

Os utilizadores devem estar familiarizados com a consola do Python [10]. A interação entre o utilizador e o programa é feita através do Tkinter [11] e do IPython. Os MPs também foram desenvolvidos em Python 3, usando as bibliotecas do TF. Todavia, o desempenho computacional não é afetado pela escolha desta linguagem, pois o TF converte internamente os modelos descritos para Compute Unified Device Architecture (CUDA) [12], C++ e está otimizado para a utilização em GPU.

1.4 Questões de investigação

Tendo em conta os objetivos para o trabalho realizado foram propostas duas questões de investigação:

— É possível desenvolver modelos de DL probabilísticos sem ser necessário ter um conhecimento específico e aprofundado no domínio do DL com MPs?

— É possível implementar uma IG que vá ao encontro das necessidades dos utilizadores que pretendem desenvolver CNNs probabilísticas de forma simples e direta.

1.5 Solução apresentada

A IG para desenvolvimento de CNNs probabilísticas foi proposta neste trabalho para dar resposta às questões de investigação.

A IG desenvolvida permite ao utilizador criar e treinar CNNs probabilísticas de forma simples sem requerer programação adicional por parte do mesmo. Para utilizar a IG é necessário instalar as bibliotecas utilizadas pelo programa desenvolvido, garantindo a compatibilidade entre as mesmas. A instalação das versões de CUDA e NVIDIA CUDA Deep Neural Network (cuDNN) [13] compatíveis são necessárias caso o utilizador pretenda que as RNAs sejam treinadas na GPU.

A IG também permite testar múltiplas arquiteturas e posteriormente comparar os resultados obtidos pelas mesmas. Particularmente, a IG possui várias opções de visualização, possibilitando a representação dos dados que o utilizador pretende examinar.

1.6 Aplicabilidade dos Modelos Probabilísticos

Para se compreender os benefícios e aplicabilidade dos MPs relativamente aos MDs é fornecido o seguinte exemplo prático, ao utilizar-se o sistema de Global Positioning System (GPS) dos automóveis, este indica como é possível deslocar-se do ponto A, localização atual, para o ponto B, o destino.

Para além do percurso sugerido, o GPS também realiza uma previsão do tempo necessário para se realizar o percurso. Esta estimativa é compreendida como a me-

lhor previsão; todavia, sabe-se à partida que em ambiente real existirá incerteza em relação ao tempo necessário. Para um GPS que use um MD, a sua previsão será apenas um único valor para o tempo de viagem, não indicando assim uma gama de valores possíveis, o que ocorreria se fosse usado um MP.

Apresenta-se um exemplo na figura 1.6, onde, à esquerda da figura é possível observar-se o resultado do GPS determinístico, indicando os dois percursos e o tempo do percurso. Considera-se agora, em alternativa, a utilização de um GPS com um MP e apresenta-se o resultado do mesmo no lado direito da figura 1.6, onde para além do tempo médio de viagens, o modelo também apresenta a distribuição dos tempos de viagens, ou seja, o condutor terá acesso a mais informação, podendo desta forma escolher o melhor percurso (por exemplo, aquele que apresente menor desvio padrão) [9]. De igual forma, os MPs trazem inúmeras vantagens em múltiplos ramos como a biomedicina, a visão por computador, a deteção automática de intrusão, entre outros. A apresentação da incerteza na análise permite ao utilizador do modelo saber o grau de confiança que pode ter na previsão e ajustar a ação a ser realizada.

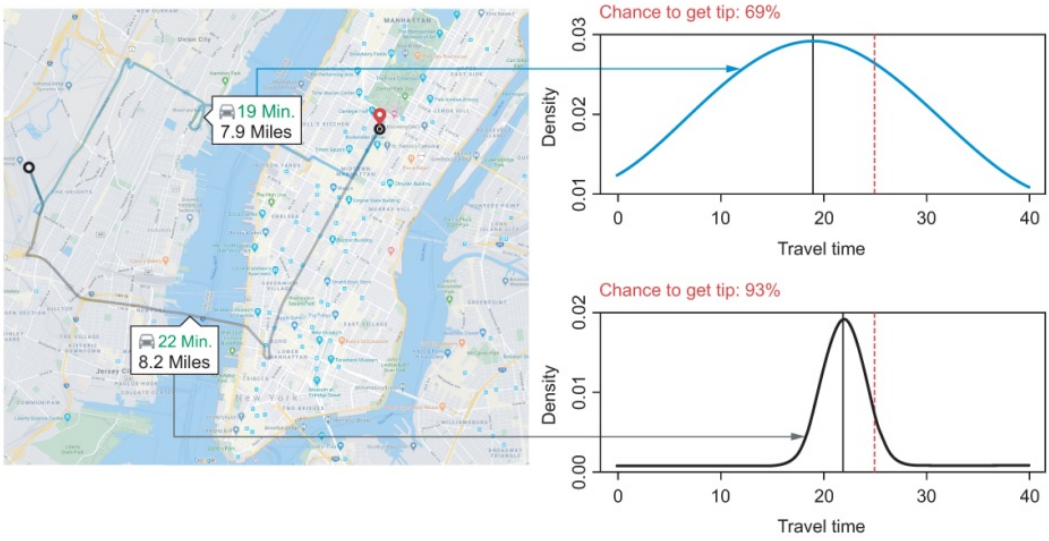


Figura 1.6: MD / MP [9].

1.7 Estrutura do trabalho

O trabalho está dividido em seis capítulos:

Capítulo um: Introdução: apresenta os objetivos, a motivação para o trabalho, as questões de investigação e a solução implementada.

Capítulo dois: Estado de arte: analisa os trabalhos já realizados na área do DL com uso de IG para desenvolvimento de modelos.

Capítulo três: Métodos: aborda as Redes Neuronais (RNs), RNAs e as CNNs; apresenta-se o princípio da IV e da sua utilização em DL. É apresentado o procedimento seguido para o desenvolvimento da IG e as principais bibliotecas utilizadas (Tkinter, TF e Keras).

Capítulo quatro: Interface Gráfica: apresenta o funcionamento da IG e a abordagem seguida para a implementação dos MPs da mesma.

Capítulo cinco: Resultados: apresenta-se os conjuntos de dados utilizados para os testes; analisa-se os resultados obtidos para a validação dos MPs da IG. Também são apresentados os resultados dos testes de usabilidade da mesma.

Capítulo 6: Conclusão: apresenta a conclusão de todo o trabalho realizado, sendo apresentadas as limitações para o mesmo e sugestões para trabalhos futuros.

Capítulo 2

Estado de Arte

Este capítulo contém uma análise do trabalho já realizado na área de estudo, apresentam-se assim aplicações semelhantes à desenvolvida. É desenvolvido um modelo através da aplicação Orange Data Mining o qual se apresenta na parte final do capítulo.

2.1 Trabalhos anteriores

Analisando o estado-da-arte das IGs para desenvolvimento de modelos de DL verificou-se uma lacuna relativa aos MPs, especialmente com CNNs probabilísticas. Esta lacuna é de particular relevância, pois estes modelos requerem elevado conhecimento nas áreas da estatística e do DL, existindo ainda pouca documentação sobre os procedimentos necessários para o desenvolvimento dos modelos. Tal restringe a utilização em massa destes modelos, estando apenas acessível a profissionais com conhecimento na área. Uma abordagem inovadora foi empregue pelo pacote Matlab [14] (aplicação Classification lerner) [15], cuja IG é apresentada na Figura 2.1. Esta utiliza uma interface simples baseada em botões com pequenas mensagens para guiar o utilizador, permitindo que qualquer utilizador possa desenvolver modelos de DL convencionais (incluindo RNAs com alimentação para a frente) sem ter que dominar todos os conceitos subjacentes aos modelos, tendo apenas de saber como usar o modelo desenvolvido. Tal permite acelerar o uso destes modelos nas diversas áreas do saber.

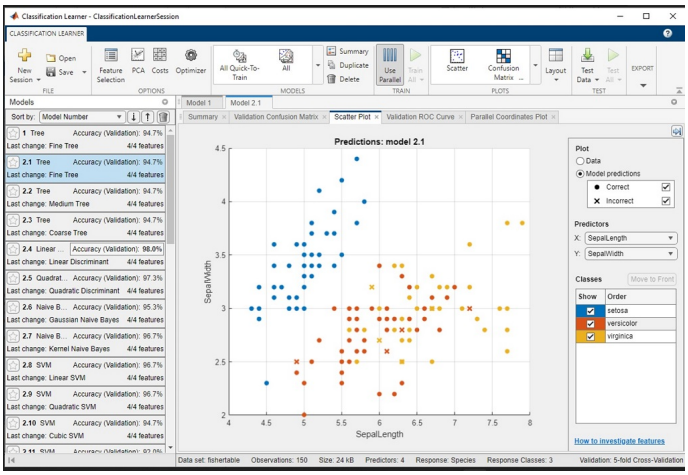


Figura 2.1: Aplicação Classification Lerner [15].

Outra abordagem inovadora foi empregue através do desenvolvimento da aplicação Orange Data Mining [16], disponível em: (https://orangedatamining.com), é gratuita, baseada numa interface simples baseada em botões com pequenas mensagens para guiar o utilizador, desenvolvida em Phyton, mas que apenas permite utilizar MDs simples para DL, a figura 2.2 apresenta a IG da mesma.

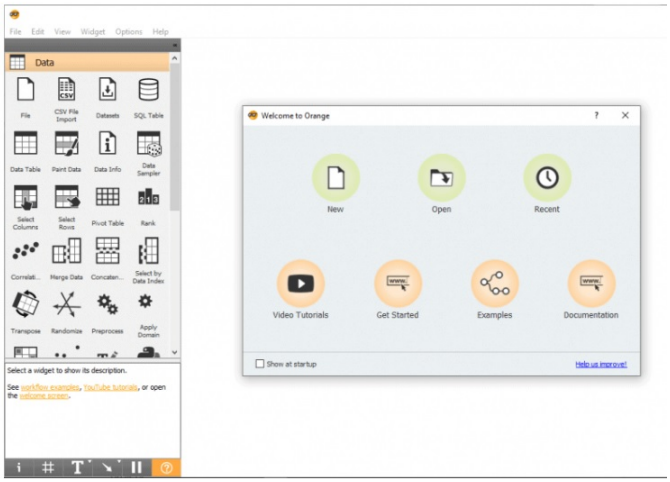


Figura 2.2: Aplicação Orange Data Mining [17].

A aplicação Rapid Miner Studio [18], disponível em: (https://rapidminer.com/platform/). Permite a criação de MDs simples para DL. A aplicação é apresentada na figura 2.3.

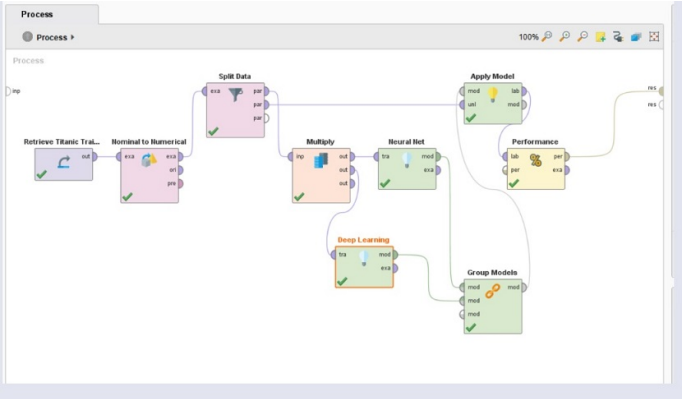


Figura 2.3: Aplicação Rapid Miner [19].

A aplicação Dataiku [20] disponível em: (https://www.dataiku.com/). Tal como a aplicação Rapid Miner permite a criação de MDs, embora tenha mais opções para a criação de modelos, tendo como público alvo empresas e investigadores. Apresenta-se a aplicação na figura 2.4.

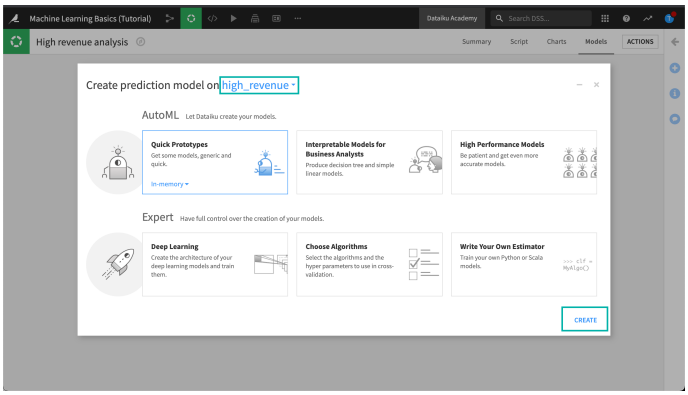


Figura 2.4: Aplicação Dataiku [21].

Outras aplicações foram tidas em consideração tais como: MLjar disponível em: (https://mljar.com/) e knime [22], disponível em: (https://www.knime.com/knime-analytics-platform), estas aplicações tais como as anteriores permitem que qualquer utilizador possa desenvolver modelos de DL convencionais. Após explorar-se todas as aplicações anteriormente mencionadas, verificou-se que em nenhuma são desenvolvidas CNNs probabilísticas. Contudo, a análise efetuada a todas as aplicações anteriormente mencionadas revelou-se essencial para o desenvolvimento da IG desenvolvida, todas as aplicações são intuitivas, existindo diversas semelhanças entre as mesmas.

2.2 Criação de um modelo através da aplicação Orange Data Mining

Escolheu-se a aplicação Orange Data Mining para a criação do modelo, pois após um estudo às diversas aplicações é a que mais se aproxima da IG desenvolvida. Do estudo realizado às diversas aplicações esta releva-se a mais intuitiva e simples. Utilizou-se a aplicação Orange Data Mining para a criação do modelo, apresenta-se a criação do modelo através da figura 2.5. Tal como é possível verificar-se importou-se o conjunto de dados da flor Iris [23], disponível em: (https://archive.ics.uci.edu/ml/datasets/iris). A aplicação é intuitiva, e a edição do modelo é simples.

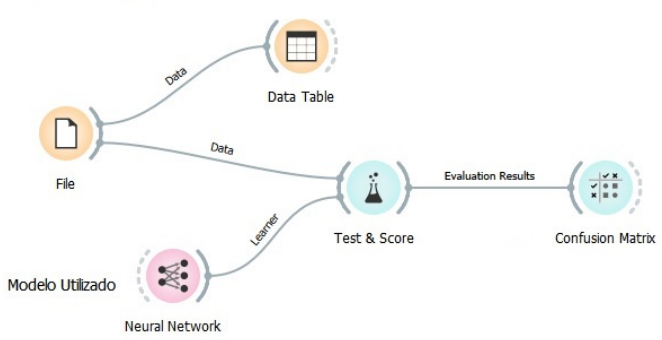


Figura 2.5: Criação do modelo através da aplicação Orange.

Tal como se observa na figura 2.5 criaram-se cinco *Widgets*, denominados por "File", "Neural Network", "DATA Table", "Test & score" e "Confusion Matrix". Através do *Widget* denominado por "File" é possível importar o conjunto de dados Iris. Através do *Widget* denominado por "DATA Table" é possível visualizar-se o conjunto de dados, este *Widget* é apresentado pela figura 2.6.

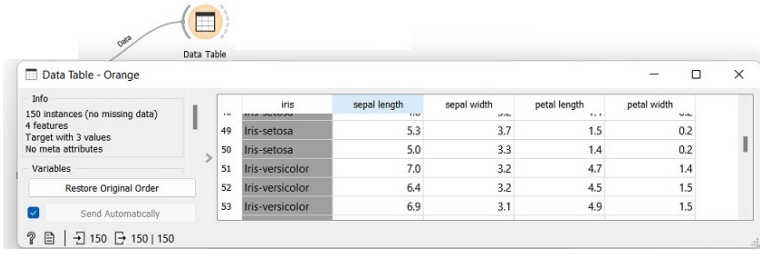


Figura 2.6: Visualização do conjunto de dados Iris.

Através do *Widget* denominado por "Neural Network" é possível treinar o modelo, sendo possível alterar as Função de Ativação (FA), o otimizador, a quantidade de neurónios e a quantidade de iterações, este *Widget*, é apresentado pela figura 2.7.

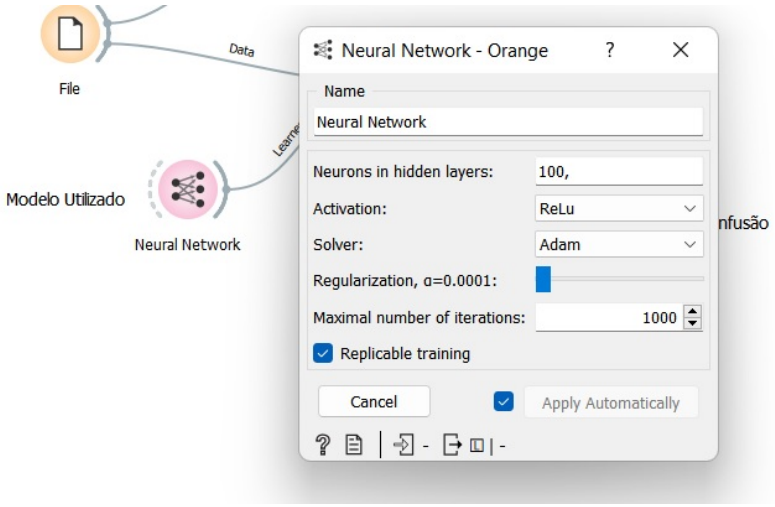


Figura 2.7: *Widget* denominado por "Neural Network".

Através do *Widget* denominado por "Test & Score" é possível treinar o modelo e visualizar-se os resultados, também é possível alterar a quantidade de dados para teste e para treino, o *Widget* é apresentado pela figura 2.8.

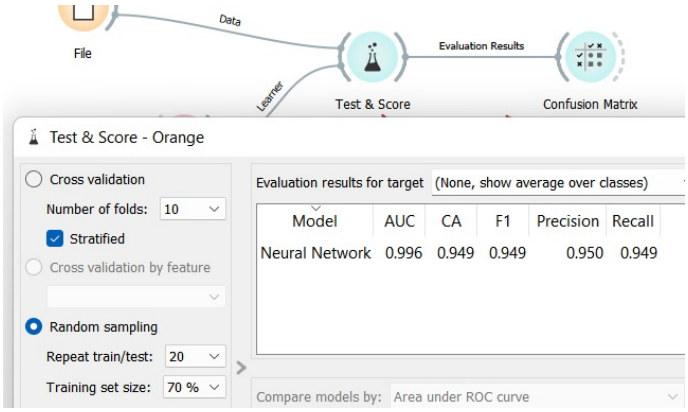


Figura 2.8: *Widget* denominado por "Test & score".

Através do *Widget* denominado por "Confusion Matrix" é possível visualizar-se a matriz de confusão, o *Widget* é apresentado pela figura 2.9.

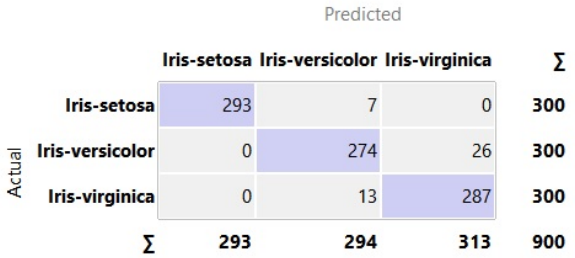


Figura 2.9: *Widget* denominado por "Confusion Matrix".

2.3 Pontos-chave

Para se realizar a IG consideraram-se os exemplos, anteriormente apresentados. Estes exemplos permitem a qualquer utilizador desenvolver modelos de DL simples para MDs, desta forma as aplicações em estudo permitem ao utilizador selecionar um conjunto de parâmetros para o desenvolvimento dos modelos. Após o desenvolvimento estar realizado é permitido ao utilizador ter acesso a um conjunto de resultados. No caso do desenvolvimento da IG para MPs, a IG desenvolvida apresenta opções de seleção dos modelos, treino e obtenção dos modelos de treino de forma simples e intuitiva, de salientar que a IG permite de igual forma a escolha de parâmetros tal como as aplicações tidas em consideração. Alguns dos resultados obtidos após o desenvolvimento dos modelos são equivalentes aos das aplicações tidas em consideração.

A abordagem seguida neste trabalho baseia-se na ideia apresentada pelas diversas aplicações mencionadas durante este capítulo, mas aplicado de forma inovadora à DL com CNNs probabilísticas que possuem enorme aplicabilidade em múltiplos ramos.

Capítulo 3

Métodos

Neste capítulo abordam-se as RNAs e as CNNs probabilísticas.
Para implementar as CNNs probabilísticas serão descritos conceitos essenciais tais como: FA: Rectified Linear Unit (ReLU) e Softmax, Otimizador Adam, IV e o TF.

Para o desenvolvimento da IG, abordam-se todos os métodos necessários tais como: Engenharia de Requisitos (ER), *brainstorming*, Levantamento de Requisitoss (LRs), Atributos de Qualidade (AQs), Quality Attribute Workshop (QAW), Diagrama de Fluxo de Dados (DFD), Diagrama Entidade Relação (DER), Mapa de Navegação (MN), prototipagem, Avaliação da Usabilidade (AU) e Testes de Usabilidade (TUs).

3.1 Redes Neuronais Artificiais

No desenvolvimento das CNNs, umas das primeiras questões prende-se com a arquitetura a ser utilizada. Esta define os tipos e sequência das camadas, os parâmetros, número de entrada, número de saídas e as FAs.

Através do conhecimento das capacidades do cérebro humano e tendo em consideração que existem tarefas que não conseguem facilmente ser desempenhadas por computadores, surgiram diversas tentativas de replicar o funcionamento do cérebro. O objetivo é imitar o seu funcionamento em tarefas-chave como a análise de imagens ou a identificação de estados de espírito, que revelam-se complexas para os computadores.

As RNs [24] são definidas como um conjunto de neurónios ligados entre si de forma semelhante ao cérebro, composto por neurónios com a estrutura apresentada na figura 3.1. O cérebro humano contém cerca de 100 biliões de neurónios, densamente interligados, por cerca de 100 triliões de sinapses [25]. Os neurónios, em termos funcionais, encontram-se agrupados em RNs que frequentemente partilham neurónios com outras redes, estes neurónios comunicam através de disparos em resposta a determinados acontecimentos.

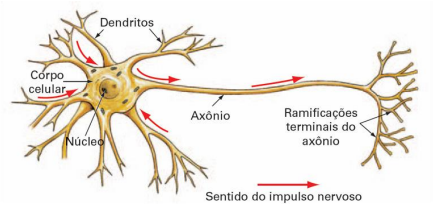


Figura 3.1: Constituição biológica do neurónio [26].

As dendrites formam redes ramificadas de fibras nervosas com capacidade de transportar sinais elétricos até ao corpo do neurónio. Por sua vez, o corpo do neurónio desempenha a função de soma das entradas afetadas pelos respetivos pesos da aplicação de uma função não linear sobre os sinais que recebe; o axónio é uma fibra nervosa única que transporta o sinal de saída do neurónio até outros neurónios; o ponto de contacto entre o axónio de um neurónio e a dendrite de outro neurónio é designado por sinapse e a membrana neuronal delimita o corpo celular separando o meio intracelular e extracelular.

Existe uma enorme diversidade de RNAs com diferentes características. Todas possuem inspiração biológica seguindo a transcrição do neurónio biológico para o neurónico artificial conforme apresentado na figura 3.2.

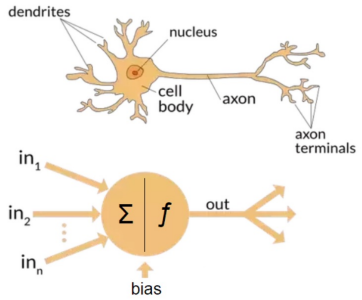


Figura 3.2: Diferença entre um neurónio neuronal e artificial [27].

.

Uma das RNAs mais utilizadas é a *Feedforward Neural Network* [28], onde os neurónios desta implementam uma função definida pela equação 3.1.

$$y = F(\sum_{i=1}^{n} Ii.wi) \tag{3.1}$$

A equação 3.1 apresenta o resultado da saída de um neurónio artificial, sendo w os pesos, i as entradas, n a última entrada e F a FA [29]. Esta função permite escolher diferentes FAs de forma a moldar a resposta, sendo tipicamente usadas funções não lineares. Caso existam múltiplas camadas então é criada uma função composta. Por

exemplo, para o caso com duas camadas, o resultado é dado pela equação 3.2 sendo representado um exemplo deste cenário na figura 3.3.

$$y = F_1(\sum_{j=1}^{nh} w'_{j1}(\sum_{l=1}^{nl} w_{lj} I_l)) \tag{3.2}$$

Embora seja possível utilizar RNAs com um número elevado de camadas, provou-se que as redes com uma camada escondida são aproximadores universais, podendo-se imitar o comportamento de qualquer função [30].

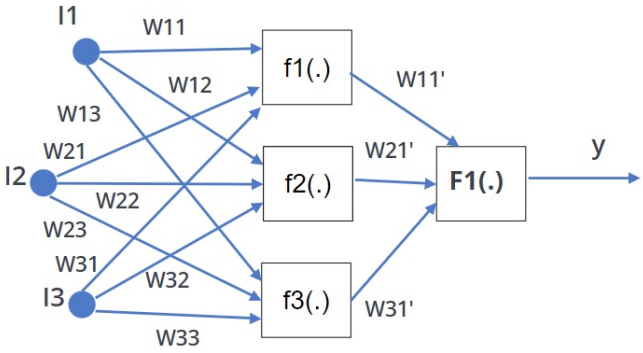


Figura 3.3: RNA com várias FAs.

As RNAs aprendem através da apresentação de exemplos que levam à modificação dos pesos da rede, existindo múltiplos algoritmos de treino. O treino das RNAs é usualmente realizado pelo algoritmo *backpropagation* cujo princípio base é: a partir do erro de saída, fazer um ajuste dos pesos ao longo da rede para esta responder corretamente. Os pesos da RNA vão sendo atualizados de forma iterativa, partindo de uma estimativa inicial, atualizam-se os pesos, a partir da equação 3.3.

$$W_{k+1} = W_k + \alpha K.\rho K \tag{3.3}$$

Onde ρk representa a direção de procura e αk representa a taxa de aprendizagem. Os diversos algoritmos têm formas diferentes de escolher αk e ρk.

As funções de minimização são quadráticas, sendo mais frequente utilizar-se um processo de derivação para procurar um mínimo, dado pela equação 3.4, onde G é o gradiente da função a minimizar na iteração k.

$$W = W_k - \alpha G(x, K) \tag{3.4}$$

3.1.1 Algoritmo Backpropagation

Os cálculos feitos por cada neurónio onde ϕ é a FA, w_i é o peso para a entrada i, b é o $bias$, x_i é a entrada i e n é o número de entradas, o cálculo de neurónios é dado pela equação 3.5.

$$y = \phi(b + \sum_{i=1}^{n} w_i x_i) \tag{3.5}$$

Os pesos e $bias$ necessários para as somas ponderadas são alterados durante o processo de treino, de modo a obter as previsões mais precisas sobre a camada de saída. No entanto, as RNAs podem ser treinadas utilizando vários algoritmos. O algoritmo de aprendizagem *backpropagation*, que na realidade é o Steepest Descent, uma vez que todos os algoritmos de treino baseados em derivadas são do tipo backpropagation, utiliza uma função de custo também conhecida como perda ou desempenho, e alterará os pesos e os $bias$ de modo que a função seja minimizada. O valor da função de custo representa o erro exibido pelo modelo. Uma função de custo normalmente utilizada é o Erro Quadrático Médio (MSE), o seu custo é dado pela equação 3.6, onde t_i é o valor alvo (etiqueta) para a amostra i, p_i é a previsão (saída) para a amostra i e n é o número total de amostras [31].

$$E = \frac{\sum_{i=1}^{n} (t_i - p_i)^2}{n} \tag{3.6}$$

Em cada época os pesos e $bias$ são atualizados tendo em conta os valores de gradiente. Portanto, se o valor da produção o_i dependerá diretamente do valor do peso w_{ij}, o valor do gradiente é dado pela equação 3.7.

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial o_i} \frac{\partial o_i}{\partial w_{ij}} \tag{3.7}$$

A atualização dos pesos é dada pela equação 3.8, onde $\frac{\partial E}{\partial w_{ij}}$ é o valor do gradiente, α é a taxa de aprendizagem e k representa a camada [32].

$$\Delta w_{ij}^{k} = -\alpha \frac{\partial E}{\partial w_{ij}} \tag{3.8}$$

3.1.2 Funções de Ativação ReLU e Softmax

As FAs determinam a saída de cada neurónio de uma RNA. Em particular, numa RNA, a FA é responsável por transformar a entrada ponderada somada. Desta forma, a FA permite que uma RNA possa efetuar tanto mapeamentos lineares como não lineares das entradas x às saídas y.

Nas RNAs as FAs mais comuns são a sigmoid e a tangente hiperbólica [33]. Todavia, estas funções não são tipicamente usadas em DL, pois provocam o problema de

vanishing gradient [34] sendo substituídas pela ativação linear retificada (ReLU) [35] pois são mais eficientes. Esta FA e a sua derivada são dadas, respetivamente, pelas equações 3.9 e 3.10.

$$\mathrm{ReLU}(x) = \max\{0, x\} \tag{3.9}$$

$$\mathrm{ReLU}'(x) = \begin{cases} 1, & x >= 0 \\ 0, & c.c. \end{cases} \tag{3.10}$$

A função ReLU devolve 0 para todos os valores negativos, e o próprio valor para valores positivos. Como o seu resultado é zero para valores negativos, ela tende a "apagar" neurónios durante um passo *forward*, aumentando assim a velocidade do treino. Teoricamente, a derivada não está definida em 0, mas pode-se implementá-la como sendo 0 ou 1. As derivadas são estáveis, sendo 1, quando a entrada x é positiva e zero quando for negativa. Desta forma, a segunda derivada é zero em todo o domínio. Uma desvantagem desta FA é que os neurónios tendem a 'morrer' durante o treino, ou seja, o neurónio passa a produzir apenas o valor zero à saída. Tal acontece quando a soma ponderada antes da aplicação da ReLU se torna negativa, fazendo com que a unidade produza zero. Nesta região, a derivada também é zero, fazendo com que os pesos associados ao neurónio deixem de ser atualizados com o algoritmo de treino.

A FA linear é a FA mais básica, pois não altera a saída de um neurónio, geralmente é utiliza nas camadas de saída em RNA de regressão [36].

A função Softmax [37] é uma generalização da FA sigmoide para problemas multi-classe (mais de duas classes). A FA softmax transforma as saídas de cada classe para valores entre 0 e 1, logo, fornece a probabilidade de a entrada estar numa determinada classe. Desta forma, é utilizada na camada de saída, ou seja, na última camada para realizar a classificação. A FA Softmax é dado pela equação 3.11.

$$\mathrm{Softmax}(x^y) = \frac{e^{x^y}}{\sum_{w=1}^{\Omega} e^{x^y}} \tag{3.11}$$

Onde x representa a distribuição, y a entrada, e Ω representa todas as saídas possíveis.

A função de custo mais comum para a classificação é a *cross-entropy*, dada pela equação 3.12, onde t_c é o valor alvo para a classe c, p_c é a previsão para a classe amostrada c e n é o número total de classes [31].

$$E = -\sum_{c=1}^{n} t_c.\log(p_c) \tag{3.12}$$

3.1.3 Otimizador Adam

O algoritmo de otimização Adam [38] é o mais usado em DL, baseado no conceito da análise da descida de gradiente. Em particular, este método combina os algoritmos AdaGrad e RMSProp.

AdaGrad [39], este algoritmo é usado para acelerar o algoritmo de descida do gradiente, considerando as médias exponencialmente ponderadas dos gradientes. O uso de médias faz com que o algoritmo convirja para os mínimos num ritmo mais rápido, o seu cálculo é dado pelas equações 3.13 e 3.14.

$$W_{t+1} = W_t - \alpha m_t \tag{3.13}$$

$$m_t = \beta m_{t-1} + (1 - \beta) \left[\frac{\delta L}{\delta w_t} \right] \tag{3.14}$$

Onde, w são os pesos, m os gradientes, β é o parâmetro da média móvel α é a taxa de aprendizagem, L a função de perda e t o tempo.

Root Mean Square Prop (RMSProp) [40] é um algoritmo de aprendizagem adaptativo. Em vez de obter a soma cumulativa do quadrado do gradiente, utiliza a média móvel exponencial, o seu cálculo é obtido pelas equações 3.15 e 3.16.

$$W_{t+1} = W_t - \frac{\alpha_t}{(v_t + \xi)^1/2} \left[\frac{\delta L}{\delta w_t} \right] \tag{3.15}$$

$$v_t = \beta v_{t-1} + (1 - \beta) \left[\frac{\delta L}{\delta w_t} \right]^2 \tag{3.16}$$

Onde v_t representa a soma dos quadrados de gradientes passados, w são os pesos; m os gradientes, β é o parâmetro da média móvel e α é a taxa de aprendizagem.
O otimizador Adam herda os atributos positivos dos dois métodos e constrói uma descida do gradiente mais otimizada, dado pela equação 3.17.

$$m_t = \beta m_{t-1} + (1 - \beta_1) \left[\frac{\delta L}{\delta w_t} \right] v_t = \beta_2 m_{t-1} + (1 - \beta_2) \left[\frac{\delta L}{\delta w_t} \right]^2 \tag{3.17}$$

3.2 Redes Neuronais Convolucionais

As CNNs normalmente são desenvolvidas para serem determinísticas e estão associadas ao reconhecimento de padrões em imagens ou sinais, apresentando uma elevada eficácia na classificação. O processo de classificação dos dados pelas redes é realizado por meio das várias camadas que as compõem [41].

Estas camadas são tipicamente de três tipos: *convolution*; *pooling*; *dense*. A alteração de dados por parte das RNAs acontece em todas estas camadas, tendo normalmente componentes não lineares para aumentar a capacidade de aprendizagem. Um exemplo de uma CNN [42] é apresentado na Figura 3.4.

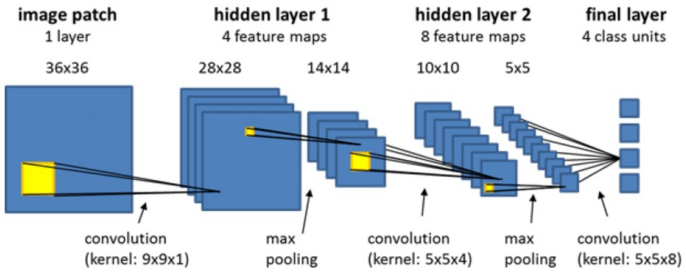


Figura 3.4: Exemplo de uma CNN [43].

As camadas de entrada e saída são as que interagem com o exterior (não são ocultas). O número de parâmetros de entrada depende diretamente do número de dimensões. Para uma RNA de uma dimensão (unidimensional) tem-se apenas um parâmetro de entrada, então a camada de entrada possui apenas uma unidade.

As convoluções funcionam como filtros, utilizando-os para a seleção de pequenos conjuntos de píxeis. A seleção é feita por toda a imagem, escolhendo desta forma as partes com informação mais relevante. Por exemplo, uma imagem de 32×32 pixeis com um filtro, selecionando uma área de 5×5 da imagem e com um movimento de 2 saltos (*stride*), levará o filtro a percorrer toda a imagem, criando um mapa de características (*feature map*) de 28×28 pixeis, conforme apresentado na figura 3.5.

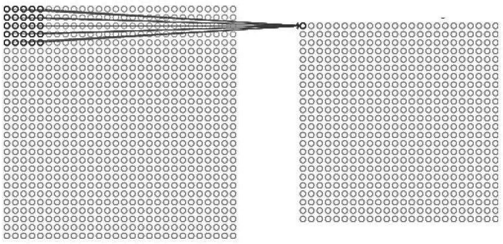


Figura 3.5: Entrada 32×32 com um filtro de 5×5 [44].

O filtro é constituído por pesos inicializados aleatoriamente, sendo posteriormente atualizados a cada nova entrada, através do algoritmo *backpropagation*. A região da entrada onde o filtro é aplicado é chamada *receptive field*.

As FAs são usadas para permitir à rede a identificação das não linearidades nos dados sendo a FA ReLU a mais usada. *Pooling* é utilizada para reduzir a informação da camada anterior. Assim como na convolução, é escolhida uma unidade de área, por exemplo, 2×2, para transitar por toda a saída da camada anterior. A unidade é responsável por resumir a informação daquela área num único valor. Para o exemplo mencionado, se a saída da camada anterior for 24×24, a saída da *pooling* será 12×12; além disso, é necessário escolher como será feita a filtragem. O método mais utilizado é a escolha do valor máximo (*maxpooling*), onde apenas o maior número da unidade é passado para a saída, conforme apresentado na figura 3.6. Essa filtragem de dados serve para diminuir a quantidade de pesos a serem aprendidos e também para evitar *overfitting* [45].

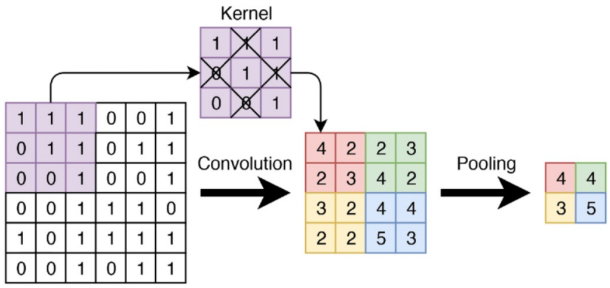


Figura 3.6: Utilização do *maxpooling*, selecionando o maior valor [46].

3.3 Redes Neuronais Convolucionais probabilísticas implementadas

Duas arquiteturas padrão (LeNet-5 e AlexNet) foram escolhidas para já estarem implementadas na IG, de forma a permitir ao utilizador desenvolver rapidamente um modelo para os dados fornecidos. Todavia, caso o utilizador pretenda também pode criar uma arquitetura camada a camada.

LeNet-5

LeNet-5 (1998) é uma das arquiteturas mais simples. Após a entrada o modelo tem sete camadas no total, sendo duas camadas *convolution*, duas camadas de *pooling* e três camadas *dense*. Esta arquitetura tem cerca de 60.000 parâmetros e está apresentada na Figura 3.7.

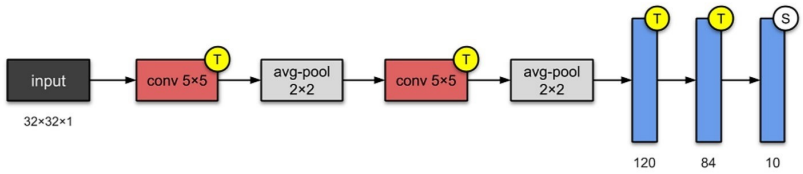


Figura 3.7: Arquitetura LeNet-5 [47].

AlexNet

AlexNet (2002), é constituída por cinco camadas *convolution*, três camadas *dense* e duas camadas de *pooling*, totalizando onze camadas após a entrada. Esta está apresentada na Figura 3.8.

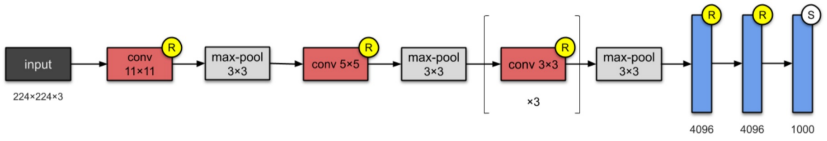


Figura 3.8: Arquitetura AlexNet [47].

3.4 Modelos Probabilísticos

Uma das principais vantagens dos MPs face aos MDs é a capacidade de apresentar a incerteza epistémica das previsões. Todavia, o desenvolvimento destes MP é tipicamente mais complexo. Um dos métodos utilizados nos MPs é o método de aproximação da IV, sendo este utilizado na implementação empregue pela IG desenvolvida.

A IV [48] permite aproximar a inferência probabilística como um problema de otimização, procurando uma distribuição à posteriori alternativa que minimize a divergência de Kullback-Leibler-Distribuition (KLD) [49] com a distribuição à posteriori verdadeira. Desta forma, todos os pesos da rede são substituídos por uma distribuição probabilística e a distribuição é otimizada durante o treino das redes. Neste caso, a distribuição Gaussiana [50] é frequentemente usada para representar estas distribuições. Para diminuir o tempo necessário para treinar as redes, às camadas onde a operação de convolução é realizada é aplicado o estimador de gradiente Flipout [51] para minimizar a divergência de KLD até ao Limite Inferior de Evidência Negativo. Este é composto por dois termos, o (*log-likelihood*) (logaritmo da verosimilhança) negativo esperado que é estimado por amostragem de Monte Carlo [52], e a KLD, atuando como um termo regularizador. Se for considerada uma imagem como entrada do modelo, a fórmula para cálculo da probabilidade da classe de saída tendo em conta a entrada e as características do modelo é dado pela equação 3.18.

$$\mathrm{P}_{OUT}(i,j) = \sum_{r=0}^{n-1}\sum_{c=0}^{m-1} \mathrm{P}_{IN}(i.S+r, j.S+c)\mathrm{PESOS}_{Filtro}(r,c) \tag{3.18}$$

Onde P_{OUT} (i, j) corresponde ao valor do píxel da imagem de saída na i-enésima linha e j-enésima coluna. P_{IN} indica os píxeis da imagem de entrada, S é o *stride*, r n e m definem o tamanho do kernel.

Resumidamente a IV aproxima diretamente a distribuição posterior com uma distribuição mais simples, evitando o cálculo da probabilidade de ocorrência (*likelihood*) [53], tal como veremos mais adiante.

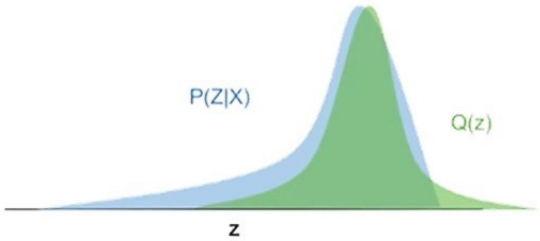


Figura 3.9: Distribuição verdadeira e distribuição gaussiana [9].

Na figura 3.9 a parte representada a azul da imagem diz respeito à distribuição verdadeira do modelo e a parte representada a verde é uma distribuição Gaussiana padrão, o MP tenta fazer o (*fitting*) da Gaussiana à distribuição real, ajustando a largura (sigma) e onde se localiza (média). Todavia, a IV permite adaptar uma distribuição que já é conhecida (a Gaussiana) a uma distribuição desconhecida. No caso das RNAs profundas, é inviável descobrir as caraterísticas da sua distribuição ao certo, desta forma, a IV viabilizou a utilização destes modelos, pois não requer que esta distribuição seja determinada de forma exata (faz uma aproximação). Neste contexto, o objetivo da IV é realizar a melhor aproximação possível, ou seja, descobrir quando o erro das previsões do modelo é mais baixo, sabendo que tal ocorre quanto mais baixa for a KLD. O desenvolvimento do TF *probability* simplificou a implementação dos MPs, pois já possui funções para realizar os cálculos da IV, na figura 3.10, apresenta-se uma RNA probabilística.

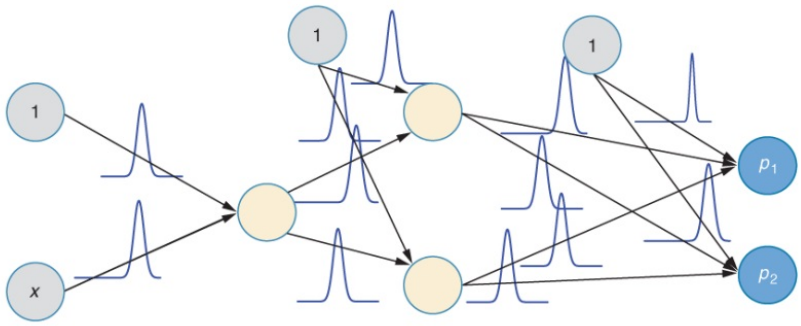


Figura 3.10: RNA probabilística [9].

3.4.1 Inferência Variacional

Cálculo da Inferência Variacional:

A distribuição posterior dos pesos fornecidos pelos dados de treino, é dada por: $\mathrm{P}(W|D)$, onde D são os dados de treino, W os pesos fornecidos e P e E_p distribuições posteriores, o cálculo de uma previsão $\hat{y}$ para os dados $\hat{x}$, é dado da equação 3.19.

$$\mathrm{P}(\hat{y}|\hat{x}) = \mathrm{E}_{p(W|D)}[\mathrm{P}(\hat{y}|\hat{x}, W)] \tag{3.19}$$

Portanto, todos os valores possíveis para os pesos ponderados pela distribuição posterior farão uma previsão para $\hat{y}$ dado $\hat{x}$. Porém, este resultado é equivalente à utilização de um conjunto de um número infinito de RNAs, tornando esta abordagem inviável para um classificador com um número de parâmetros que podem ser utilizados para uma aplicação prática. Foram propostos inúmeros algoritmos com

metodologias alternativas para calcular a distribuição posterior. Os mais frequentemente utilizados são o Markov Chain [54] e a IV. Os algoritmos Markov Chain calculam a distribuição posterior enquanto a IV se aproxima da distribuição posterior [55].

Uma forma de implementar a IV é através do algoritmo *backprogation* [56], conforme indicado por Blundell et al. [57]. De forma a interpretar o princípio da IV, considere-se a figura 3.11 onde θ representa os pesos da variante determinística da RNA, D os dados de treino e λ os parâmetros da distribuição; porém, o parâmetro θ na rede probabilística não é fixo, pois é representada por uma distribuição.
Na imagem à esquerda da figura 3.11 são representadas as distribuições possíveis, o ponto na imagem representa a distribuição posterior P($\theta|D$). Ao invés de se calcular a distribuição posterior diretamente, efetua-se uma aproximação através da IV usando como referência $q\lambda(\theta)$ que neste caso é uma gaussiana (representada pela imagem à direita da figura 3.11). A IV altera λ para que $q\lambda(\theta 1)$ se aproxime o máximo possível da distribuição posterior verdadeira P($\theta 1$|D). A imagem do lado direito da figura 3.11 apresenta o resultado do ajuste realizado pela IV.

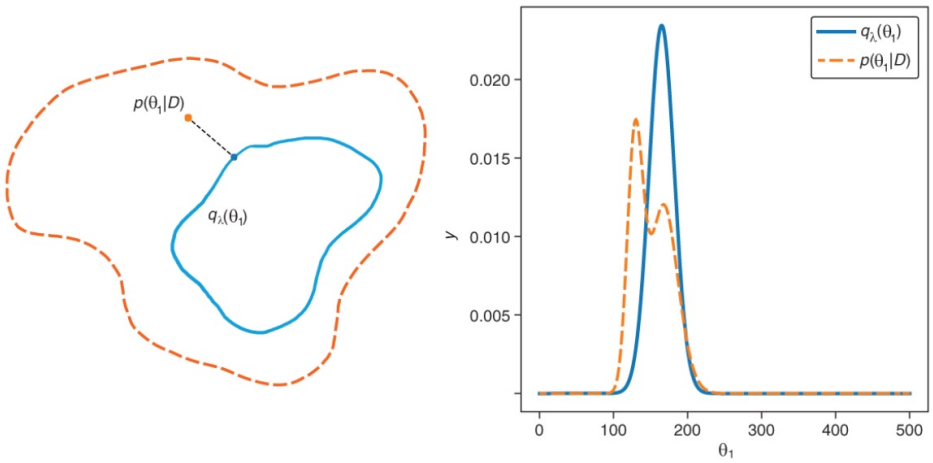


Figura 3.11: Princípio da Inferência Variacional [9].

A IV baseia-se na seleção de um membro q de uma família de distribuições Q próxima da distribuição alvo, utilizando o KLD para a medição da proximidade. O objetivo de IV é resolver o problema da aproximação de uma densidade condicional de variáveis latentes, parametrizada por parâmetros variacionais livres, dadas as variáveis observadas, utilizando a otimização. Portanto, a otimização encontra um conjunto de parâmetros θ para q que estão mais próximos, no que diz respeito ao KLD, do interesse condicional P($W|D$), onde a melhor solução é dada pela equação 3.20.

$$q^*(W|\theta) =^{arg\,\min}_{q(W|\theta)\sim Q} \mathrm{KLD}[q(W|\theta)||\mathrm{P}(W|D)] \tag{3.20}$$

Como $\mathrm{P}(W|D)$ é desconhecido, é necessário otimizar um objetivo diferente para minimizar o KLD. A abordagem padrão é utilizar a Evidence Lower Bound (ELBO), sendo o seu cálculo dado pela equação 3.21.

$$-\,\mathrm{ELBO} = \mathrm{KLD}[q(W|\theta)||\mathrm{P}(W)] - E_{q(W|\theta)\sim Q}[\log[\mathrm{P}(D|W)]] \tag{3.21}$$

A função de custo poderá ser vista como uma composição entre duas partes. A primeira parte da função é tipicamente vista como o custo de complexidade levando assim a procura de soluções cujas densidades estão próximas da probabilidade a priori. A segunda parte é dependente dos dados; é normalmente denominada *likelihood cost*, pois, descreve a probabilidade dos dados dado o modelo, levando-se a soluções que expliquem melhor os dados observados. Como resultado, esta função de custo interpreta um compromisso entre satisfazer a simplicidade do $\mathrm{P}(W)$ anterior e a complexidade dos dados [57]. Embora esta aproximação possa ser realizada pela IV, utilizando uma distribuição prévia dos pesos do modelo para produzir um efeito de regularização. Todavia, o cálculo exato do ELBO tal como apresentado é computacionalmente proibitivo. Outro aspeto relevante é a abordagem habitual para utilizar uma distribuição gaussiana para os pesos do modelo, com dois parâmetros, a média e o desvio padrão, pois esta facilita a implementação. A maximização do ELBO é equivalente à minimização da KLD. Dado que os problemas de minimização são geralmente mais fáceis de executar, por conseguinte, é frequentemente preferível minimizar o ELBO negativo. No entanto, é necessária uma aproximação ao ELBO para permitir a otimização de RNAs com métodos de descida gradiente baseados nesta função de perda [54]. Esta aproximação é dada pela equação 3.22.

$$\mathrm{ELBO} \approx \sum_{i=1}^{n} \log[q(w^i|\theta] - \log[\mathrm{P}(W^i)] - \log[\mathrm{P}(D|W^i)] \tag{3.22}$$

Uma abordagem habitual para o treino de RNAs com IV é baseada em técnicas de perturbação do peso, as quais estocasticamente amostram os pesos da rede no momento do treino. No entanto, a perturbação de peso tem geralmente uma variância elevada das estimativas de gradiente, visto que todas as amostras de pequenos grupos partilham a mesma perturbação. O Flipout, dado pela equação 3.23 foi introduzido para reduzir este problema, sendo efetuada amostragem de perturbações de peso pseudo-independentes para cada amostra. Uma perturbação de base $\hat{W}$ é partilhada por todas as amostras, cada amostra será perturbada pelo resultado da multiplicação de $\widehat{\Delta W}$ (perturbação estocástica) com uma matriz de sinal [51].

$$\Delta WS = \widehat{\Delta W}.a_s b_s^T \tag{3.23}$$

Onde a e b são vetores aleatórios amostrados uniformemente de -1 a 1. Como resultado, o Flipout pode produzir um estimador imparcial para os *loss gradients*. Ao

utilizar o estimador de Flipout, a saída de cada camada para todos os exemplos no subgrupo(X) é dada por:

$$Y = \phi[X\bar{W} + [(X.B)\widehat{\Delta W}].A \tag{3.24}$$

Onde $\bar{W}$ são os pesos médios. Dado que A e B são amostrados independentemente a partir de $\Delta\hat{W}$ e $\bar{W}$, é possível retro propagar-se com respeito a X, $\Delta\hat{W}$ e $\bar{W}$ [51], otimizando-se assim os pesos da RNA.

3.5 Desenvolvimento de Interfaces Gráficas

3.5.1 Engenharia de Requisitos

Para o desenvolvimento da IG deve-se seguir um conjunto de etapas definidas pela ER [58] e Engenharia de *Software* [57]. Um dos conceitos a ter em consideração na ER é a necessidade de se separar o problema da solução, considerando as etapas de validação, verificação e correção, conforme representado na figura 3.12.

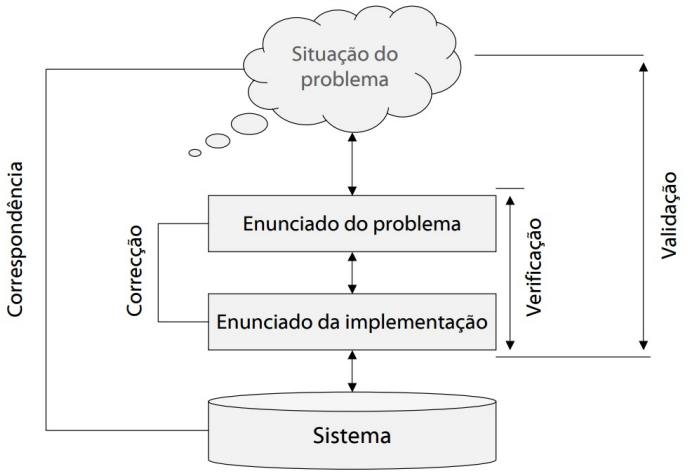


Figura 3.12: ER separa o problema da solução [59].

A necessidade de separar o problema da solução advém de múltiplos fatores como:

- Nem sempre o problema óbvio é o problema que deve ser resolvido primeiro.
- O enunciado do problema deverá ser discutido com os *stakeholders* [60].
- O enunciado deverá ser usado para avaliar as decisões de design e também será uma boa fonte para a realização de testes à IG.
- A solução resolve corretamente o problema do enunciado.

- O enunciado do problema corresponde às necessidades dos *stakeholders*.

A importância da ER é reafirmada pelos argumentos do Chaos Report [61], mencionando que os fracassos nos projetos de *software* se devem a três fatores:

- 31,1% dos projetos são cancelados antes de terminarem.
- 52,7% dos projetos custam mais 189% do que a estimativa inicial.
- Apenas 16% dos projetos são completos a tempo e dentro do orçamento.

Na tabela 3.1 apresentam-se os motivos pelos quais as IGs falham.

Tabela 3.1: Motivos pelos quais as IGs falham.

Percentagens	Motivos
13,1%	Requisitos incompletos
12,4%	Falta de envolvimento com o utilizador
10,6%	Falta de recursos
9,9%	Expetativas irrealistas
9,3%	Falta de suporte executivo
8,1%	Falta de planeamento
7,5%	Deixou de ser necessário

Após verificar-se os motivos pelos quais a ER é necessária e pelos quais as IGs falham, descrevem-se desta forma as principais conclusões:

- A ER é o conjunto de atividades para a identificação do objetivo de uma determinada IG.
- A ER é efetuada através da identificação das necessidades dos *stakeholders*.
- Os requisitos são definidos durante a fase inicial da IG, descreve-se assim como a IG deverá comportar-se, além de propriedades e atributos que a IG deverá apresentar.

3.5.2 Brainstorming

Numa primeira fase é necessário realizar o LRs, uma das técnicas para este propósito é o *brainstorming* [62], técnica de dinâmica de grupo ou até mesmo individual desenvolvida para aumentar a criatividade. Esta poderá ser elaborada de forma rápida em pequenos papeis onde cada participante deverá escrever as suas ideias, podendo esta técnica ser elabora em uma ou mais fases do desenvolvimento da IG, e é geralmente utilizada no LRs de uma IG. Na figura 3.13 é representado um exemplo de um *brainstorming* realizado através de um conjunto de papéis correspondentes às ideias que posteriormente poderão ser organizadas em categorias.

Figura 3.13: Exemplo de um *brainstorming* [63].

3.5.3 Levantamento de Requisitos

Uma das primeiras etapas para o desenvolvimento de uma IG é o LRs [64]. Geralmente tem de se identificar quais os serviços, desempenho solicitado e as restrições que poderão ser aceitáveis. Estes requisitos são recolhidos através de pesquisas, entrevistas, questionários entre outros.

Após concluído o LRs, deve-se realizar a análise de requisitos. Após esta análise é necessário especificar e documentar os requisitos. Nesta fase é possível utilizar diferentes diagramas de forma a descrever os requisitos em vários níveis de detalhe, tais como:

- DFD [65], permite modelar dados processados pela IG, sendo essencialmente uma representação gráfica do "fluxo" de dados através da IG.
- DER [66], permite relacionar as entidades da IG.
- Diagrama de Casos de Utilização (DCU) [67], resume os detalhes dos utilizadores da IG e das suas interações.

Problemas frequentes no levantamento de requisitos:

- Os utilizadores não sabem o que querem.
- Os utilizadores sabem o que querem, mas não conseguem articulá-lo.

- Os utilizadores pensam que sabem o que querem até aos programadores lhes fornecerem o que solicitaram.
- Os analistas acreditam que compreendem os problemas dos utilizadores melhor que os mesmos.

Requisitos de negócio incorporam todas as necessidades, características ou funcionalidades exigidas do negócio, tais como:

- A IG deverá estar disponível em repositório público.
- A IG deverá ser compatível com telefones Android e IPhone.

Requisitos funcionais [68] são todas as necessidades, características ou funcionalidades exigidas num processo que podem ser respondidas pela IG. Poder-se-á dizer que um Requisito Funcional (RF) é o que a IG deve realizar. Um RF expressa uma ação que deve ser realizada pela IG. Exemplos de RF são:

- A IG deve permitir o registo de novos utilizadores, sendo o processo de validação feito através de um endereço eletrónico de confirmação.
- Deve ser possível aos utilizadores solicitar a redefinição da sua senha sempre que seja solicitado; o utilizador deverá ser informado através da sua conta de endereço eletrónico.
- A IG deve disponibilizar janelas apropriadas para o utilizador ler documentos guardados.
- O utilizador pode pesquisar todo ou um subconjunto de dados.

Requisito Não Funcional (RNF) definem restrições da forma como a IG deverá ser implementada, devendo estes requisitos ser mensuráveis como, por exemplo:

- A IG deve ter garantida uma disponibilidade maior que 99.0%.
- A IG funciona em múltiplos sistemas operativos: Windows, Linux e macOS.
- O desenvolvimento da IG é em linguagem Phyton.

3.5.4 Atributos de Qualidade

Um AQ é uma propriedade, mensurável ou verificável de um sistema, utilizada para indicar como o sistema satisfaz as necessidades das suas partes interessadas. Pode-se pensar num atributo de qualidade como a medição da "perfeição" de um produto ao longo de alguma dimensão de interesse para as partes interessada [69].

Disponibilidade: refere-se à capacidade de uma IG mascarar ou reparar falhas, de forma que o período de interrupção de serviço não exceda um valor requerido. Capacidade de uma IG para ser utilizada, principalmente após a ocorrência de falhas. A falha deve ser reconhecida e então depois a IG trata da mesma.

Interoperabilidade: é a capacidade de uma IG interagir ou comunicar de forma transparente (ou o mais próximo disso) com outra IG (semelhante ou não).

Modificabilidade: analisa os custos das mudanças na IG; vários tipos de mudança poderão ser necessários (funcionalidades, mudança de plataforma, aumento de capacidade) em diferentes fases do desenvolvimento da IG.

Desempenho: examina capacidade de a IG cumprir os requisitos de tempo, ou seja, é a capacidade de a IG reagir a uma determinada ação num determinado tempo.

Segurança: é a medida da habilidade de uma IG de proteger dados e resistir a acessos não autorizados enquanto se continua a facultar o acesso a utilizadores autorizados.

Testabilidade: refere-se à facilidade com que uma IG poderá ser testada para se demonstrar os seus erros.

Usabilidade: avalia o quão fácil é para o utilizador realizar tarefas e o tipo de suporte que a IG oferece ao utilizador.

Nem sempre é possível dar a mesma resposta aos AQs. Dever-se-á ter sempre em atenção quais os AQ mais relevantes à IG e desta forma definir a arquitetura de *software* que beneficiará os AQ identificados.

Realização de um QAW [70] — Após ter-se realizado o LRs, dever-se-á realizar o QAW. Desta forma, descobrir-se-á quais os AQs que se destacam como mais relevantes para os clientes. Na figura 3.14 é apresentado um exemplo da realização de um QAW. A lista deste exemplo dá origem ao resultado de votos por AQ apresentado na tabela 3.2. Observa-se que as maiores preocupações dos clientes se focam, principalmente, na questão da usabilidade e disponibilidade do sistema.

Requisito	Votos	Atributo de Qualidade
O sistema deve estar disponível durante a existência de atualizações	5	Disponibilidade
O sistema deve proteger contra log-in inválido	4	Segurança
O sistema deverá aturalizar conforme as cadeiras que está a ter o aluno cada vez que passa de semestre	4	Interoperabilidade
O sistema deverá estar disponível durante o período lectivo	3	Disponibilidade
O sistema deverá fazer um log em caso de falha/crash ou erros	2	Testabilidade
O sistema deverá negar o acesso a pessoas não autorizadas	2	Segurança
O sistema deverá mostrar os grupos disponíveis num máximo de 3 segundos	2	Desempenho
O sistema deverá mostrar todos os grupos aos que o utilizador pertence	2	Usabilidade
O sistema deverá notificar o candidado a um grupo com resposta ao criador	2	Usabilidade
A avaliação dos elementos do grupo deve ser obrigatoriamente feita antes de continuar a usar o sistema.	2	Usabilidade
Um membro pode sair de um grupo voluntáriamente antes do inicio do projeto	2	Usabilidade
Um utilizador pode avaliar numericamente os colegas após o fim do projeto	1	Usabilidade
O sistema deverá funcionar em várias plataformas	0	Modificabilidade
Havendo alunos suficientes para o grupo, este deve ser criado em 5s.	0	Desempenho

Figura 3.14: Exemplo de um QAW [71].

Tabela 3.2: Votos por AQ

Atributos de qualidade	Número de votos
Usabilidade	9
Disponibilidade	8
Segurança	6
Interoperabilidade	4
Testabilidade	2
Desempenho	2
Modificabilidade	0

3.5.5 Diagrama de Fluxo de Dados

O DFD sinaliza o fluxo de informações para qualquer processo e utiliza símbolos pré-definidos, tais como retângulos, círculos, setas, caixas de texto, bases de dados. Este utiliza-se para a análise de uma IG existente ou na modelação de uma nova e poderá visualmente dar informações difíceis de serem transmitidas por palavras. A figura 3.15 apresenta um exemplo de um DFD, o exemplo dado é referente a um processo de pagamento, através do DFD pode visualizar-se todo o fluxo necessário para a realização desse pagamento.

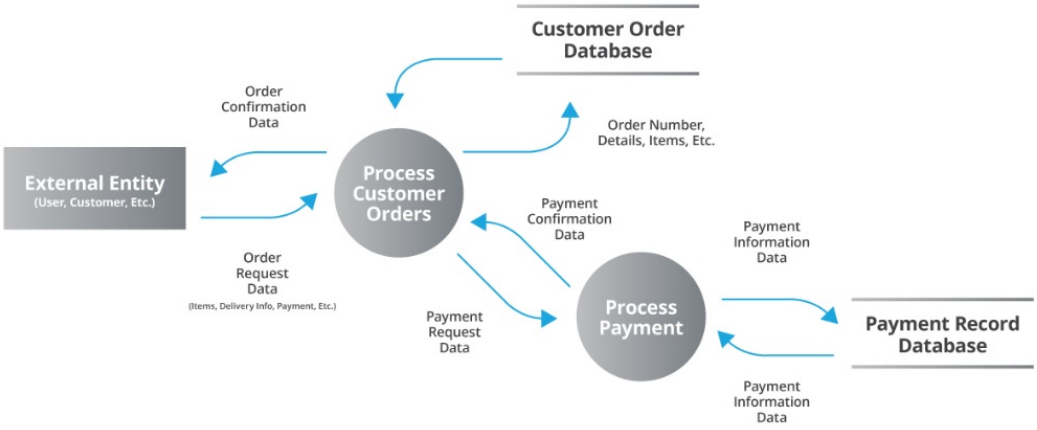


Figura 3.15: Exemplo de um DFD [72].

3.5.6 Diagramas de Entidade Relação

O DER é uma técnica de modelação de dados, constituído por entidades, relacionamentos e atributos. Este é utilizado como um modelo de dados lógico de alto nível, sendo útil no desenvolvimento de um projeto conceptual para bases de dados. O DER é constituído pelos seguintes elementos:

- Uma entidade é um item ou conceito real que existe por conta própria. Entidades são equivalentes a tabelas da base de dados e cada linha da tabela representa uma instância dessa entidade.
- Um atributo de uma entidade é uma propriedade particular que descreve a entidade.
- Um relacionamento é a associação que descreve a interação entre entidades. A cardinalidade, é o número de instâncias de uma entidade que pode ser associada a cada instância de outra entidade. Pode existir relacionamentos de um-para-um, um-para-muitos ou muitos-para-muitos.

Na figura 3.16 apresenta-se um exemplo de um DER, este exemplo refere-se à lista de pedidos, efetuado por clientes, observa-se que cada cliente poderá possuir inúmeros pedidos, mas um pedido apenas pode ser efetuado por um cliente. A entidade cliente tem as seguintes instâncias: CPF, código, endereço e nome e a entidade pedido tem as seguintes instâncias: código, data e valor.

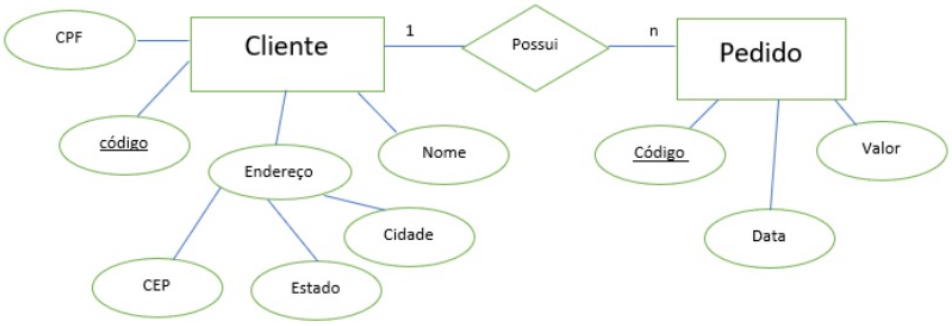


Figura 3.16: Exemplo de um DER [73].

3.5.7 Diagramas de Casos de Utilização

Os DCU resumem os detalhes dos utilizadores da IG e as interações com a IG. Um DCU deverá conter:

- Cenários onde a IG interage com os utilizadores.
- Metas que a IG ajuda os utilizadores a atingir e o funcionamento da IG.
- Definir e organizar RF na IG.
- Especificar o contexto e os requisitos da IG.
- Representar os objetivos das interações entre a IG e os utilizadores.
- Modelar o fluxo básico de eventos no caso de uso.

O DCU apresenta uma visão geral do relacionamento entre casos de utilização, utilizadores e a IG. É recomendado utilizar-se o DCU para complementar-se casos de utilização descritos em texto. Na figura 3.17 apresenta-se um exemplo de um DCU, o exemplo é referente a um restaurante como se pode observar é constituido por dois utilizadores denominados por “Food Critic” e “Chef”. O utilizador denominado por “Food Critic” tem os seguintes casos de utilização: “Eat Food”, “Pay for Food”, “Drink Wine”. O utilizador denominado por “Chef” tem um caso de utilização “Cook Food”, através deste DCU, é possível observar-se o funcionamento do restaurante relativamente ao relacionamento entre os casos de utilização e os utilizadores.

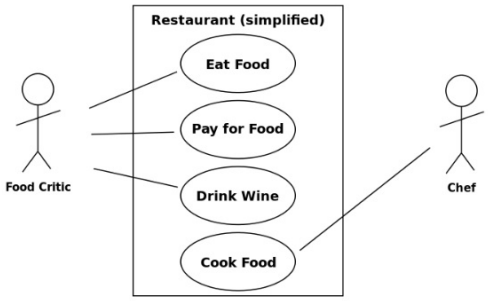


Figura 3.17: Exemplo de um DCU [74].

3.5.8 Mapa de Navegação

Um MN [75] permite realizar-se uma descrição de conteúdos, ou seja, é possível descrever-se de forma simples todo o conteúdo da IG. Após a elaboração do MN, realiza-se a descrição de todos os cenários possíveis. Na figura 3.18 é apresentado um exemplo de um MN, seguido da descrição de cenários possíveis (apenas se descrevem alguns cenários). Este exemplo refere-se às aventuras do Tazz e do Egg [76], sendo a descrição de alguns cenários:

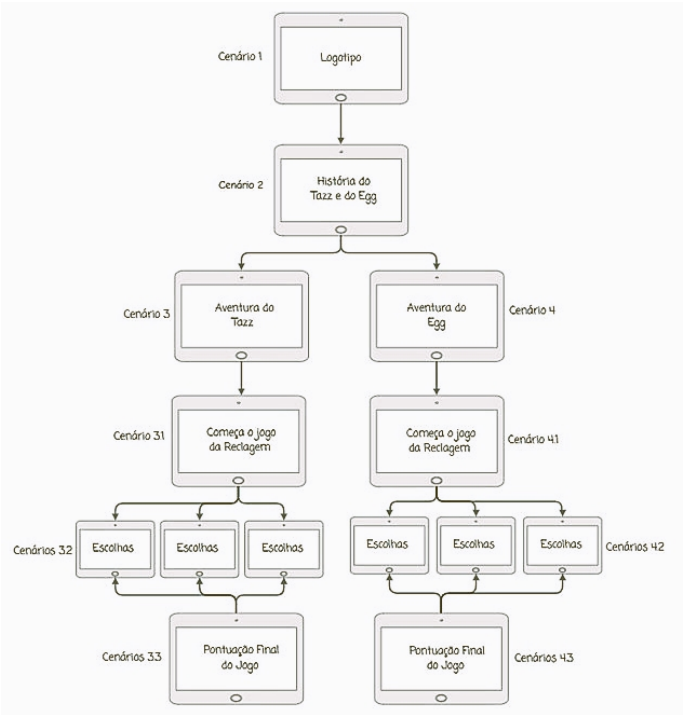


Figura 3.18: Exemplo de MN [76].

Como exemplo apresenta-se o cenário 1. Refere-se ao início da história: neste cenário exibe-se uma imagem de fundo alusiva à história e um botão para avançar na aventura.

3.5.9 Prototipagem

Para o desenvolvimento IG é essencial a efetuar-se prototipagem [77], pois auxilia na descoberta de novas ideias de forma rápida e económica, ajudando a visualizar, avaliar, organizar e a testar a mesma. Na figura 3.19, é apresentado um exemplo de um protótipo de baixa fidelidade.

Como pode observar-se do lado esquerdo da figura 3.19 está realizado um protótipo da informação relativamente à visualização das informações sobre um determinado aluno, designado na figura por "Student Information", do lado direito da figura 3.19 está realizado um protótipo da informação relativamente à adição de um seminário, designado na figura por "Add a Seminar".

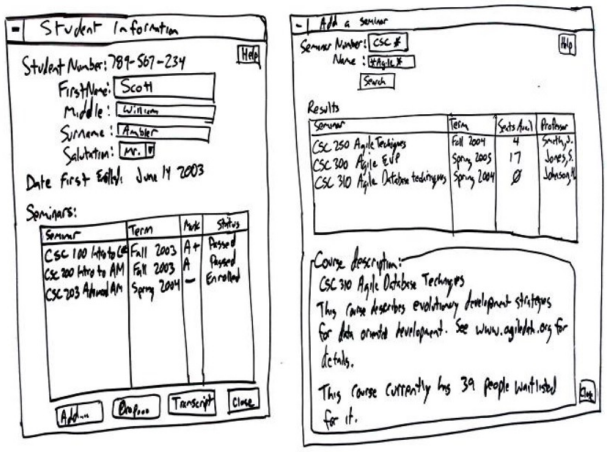


Figura 3.19: Protótipo de baixa fidelidade — "Student Information" e "Add a Seminar" [78].

3.5.10 Avaliação da Usabilidade

Para realizar-se uma AU às IGs são executadas um conjunto de tarefas a um conjunto de utilizadores. Através das Heurísticas de Nielsen é possível identificar um conjunto de falhas de usabilidades.

As dez Heurísticas de Nielsen (Nielsen, 1994) [79] utilizadas para se identificar possíveis falhas de usabilidade são as seguintes:

1. Visibilidade do estado da IG: a IG deve manter sempre os utilizadores informados sobre o que está acontecer, dando $feedback$ apropriado dentro de um intervalo de tempo aceitável.

2. Correspondência entre a IG e o mundo real: a IG deve utilizar a linguagem dos utilizadores.

3. Liberdade de controle: os utilizadores enganam-se frequentemente, logo é necessário fornecer opções de retroceder e repetir.

4. Consistência e padrões: deve-se adotar as convenções de cada IG.

5. Prevenção de erros: situações passíveis de erros devem ser eliminadas ou corretamente identificadas.

6. Reconhecer, em vez de lembrar: não deverá ser necessário memorizar informações ou procedimentos.

7. Flexibilidade e eficiência: ao realizar algumas ações com frequência é importante dar ao utilizador opções para automatizar ou agilizar estas ações.

8. Desenho estético e minimalista: o desenho deve conter apenas informações relevantes.

9. Ajudar o utilizador a reconhecer, diagnosticar e recuperar dos erros: erros devem ser tratados de forma clara e eficiente, as mensagens devem ser diretas e esclarecer o problema além sugerindo uma solução).

10. Suporte e documentação: deve ser presente e de preferência simplificada.

3.5.11 Testes de usabilidade

Deve-se realizar TUs ao longo do processo de desenvolvimento. Na fase inicial devem-se realizar em papel, sendo estes classificados como protótipos de baixa fidelidade. Na fase mais avançada do desenvolvimento deve-se realizar protótipos de alta fidelidade, estando estes mais próximos do produto final, mas ainda em fase de construção. O teste de validação, ou testes finais realizam-se numa fase mais adiantada do desenvolvimento. A avaliação da usabilidade da IG, deve realizar-se por especialistas em áreas como a interação Humano-Computador e multimédia.

Os TUs efetuam-se usualmente por formulários ou por observação realizada de forma presencial ou por videoconferência. Todavia, o mais importante neste tipo de TUs será sempre o *feedback* que se pretende obter [80]. Na figura 3.20 apresenta-se um formulário com dez perguntas, este formulário é um exemplo de uma TUs e designa-se por *system usability scale*.

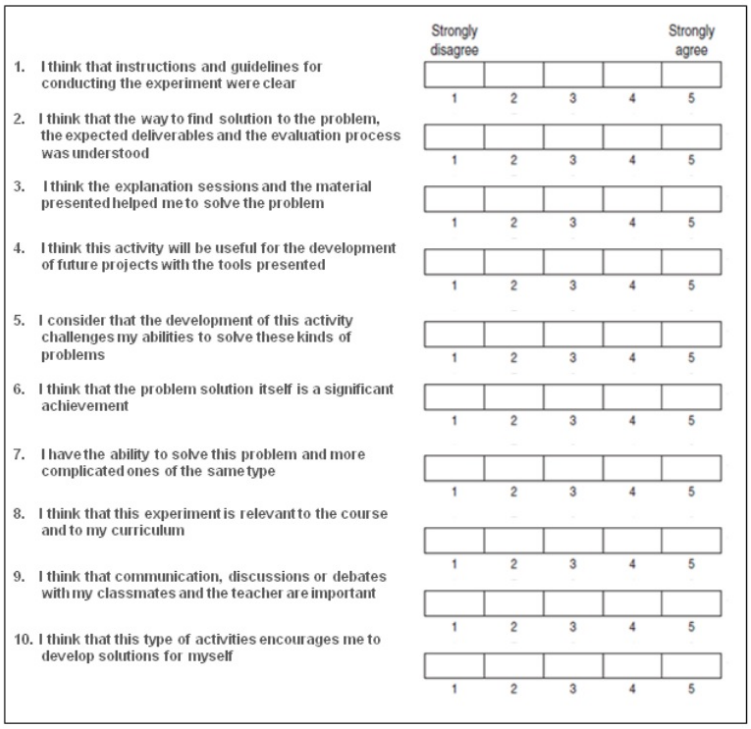


Figura 3.20: Formulário - *system usability scale*. [81]

3.6 Principais bibliotecas utilizadas

Para a criação da IG e implementação das CNN probabilísticas utilizaram-se múltiplas bibliotecas disponíveis em Python, sendo o Tkinter e o TF as mais relevantes.

3.6.1 Tkinter

Tkinter é um dos módulos do Python para o desenvolvimento de IGs. Este possui múltiplos recursos de desenho, permitindo a criação de IGs complexas, estando a biblioteca bem documentada. A utilização desta é relativamente simples e tem a vantagem de fazer parte do pacote básico de Python, pelo que qualquer computador que tenha o interpretador de Phyton instalado poderá executar a IG com o Tkinter. Desta forma, os motivos para a escolha do Tkinter para o desenvolvimento da IG foram:

- Facilidade de utilização.
- Recursos disponíveis.
- Linguagem utilizada (Python) ser a mesma linguagem usada para o desenvolvimento dos MPs.

Na figura 3.21 apresenta-se o código em Phyton para a criação de uma IG utilizando o Tkinter.

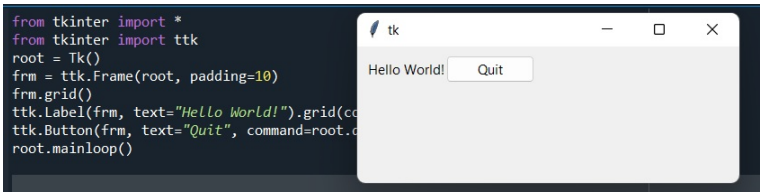


Figura 3.21: Exemplo da utilização do Tkinter [82].

3.6.2 TensorFlow

TF é uma biblioteca Python de código aberto criada para computação numérica com foco em DL. Foi desenvolvida pelo Google em 2015 e rapidamente se tornou uma das principais ferramentas para desenvolvimento de modelos de DL. Os seus calculos são baseados em tensores. Estes são matrizes multidimensionais de tipo uniforme, apoiados pela memória do acelerador do GPU. Particularmente, o TF cria grafos para o cálculo das operações matemáticas, onde as arestas representam o fluxo de dados entre nós (os tensores). Na figura 3.22 apresenta-se um exemplo para o cálculo da equação 3.25 e na figura 3.23 apresenta-se a representação do grafo do TF para o cálculo da equação 3.25.

$$z = (x \times y) - (x + y) \tag{3.25}$$

Relativamente à passagem da equação matemática, dada pela equação 3.25 para código apresentado pela figura 3.22, observa-se que:

Na equação (x+y) é dado pelo código: tf.add(x, y), salvando o resultado em o1.

Na equação (x × y) é dado pelo código: tf.multiply(x, y), salvando o resultado em o2.

Na equação (x × y) - (x+y) é dado pelo código: tf.subtract(o2, o1), salvando o resultado em o3.

```
2 import tensorflow as tf
3
4 # function to be traced
5 @tf.function # tensorflow graph function
6 def myFunction(x, y):
7   o1 = tf.add(x, y)
8   o2 = tf.multiply(x, y)
9   o3 = tf.subtract(o2, o1)
10  return o3
```

Figura 3.22: Cálculo da equação 3.25, utilizado o TF.

Relativamente à tradução do código, apresentado pela figura 4.12, para o grafo apresentado pela figura 3.23 observa-se que:

Na equação o1 é apresentado no grafo por "Add".
Na equação o2 é apresentado no grafo por "Mul".
Na equação o3 é apresentado no grafo por "Sub".

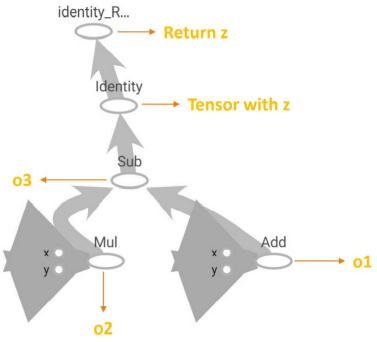


Figura 3.23: Representação do grafo TensorBoard [30].

Módulos do Keras

Keras é uma Application Programming Interface (API) de RNA de alto nível desenvolvida em Python integrada no TF a partir da versão 2.

Outro fator importante são as funções associadas ao treino dos modelos. Nestas destaca-se o Early Stopping que pára o treino se alguma propriedade deixou de melhorar por um número de iterações definidas por Patience, sendo definido um valor, mínimo de melhoria. Relativamente à escolha da penalização dos erros, realiza-se cost sensitive learning que utiliza a class weigths.

Procedimento para desenvolvimento e uso de uma CNN com o Keras:

- Pré-processamento: é necessário realizar-se o pré-processamento dos dados.
- Criação da CNN: é necessário criar as camadas do modelo para depois treinar o mesmo. As camadas são responsáveis pela extração de representações do conjunto de dados apresentados à RNA.
- Treino da CNN: após a criação do modelo e do conjunto de dados ter sido pré-processado, treina-se o modelo e guarda-se o mesmo.
- Utilização do modelo criado para previsão: usa-se o modelo treinado para efetuar previsões para dados que lhe são apresentados.

Outras bibliotecas utilizadas:

- NumPy [83]: permite trabalhar com matrizes multidimensionais e possui ferramentas para computação numérica.
- Matplotlib [84]: permite obter representações gráficas de dados através do Python.
- Scikit-learn [85]: permite executar os algoritmos de DL (supervisionada e não-supervisionada) mais conhecidos.

3.7 Pontos-chave

Este capítulo iniciou-se com uma introdução às RNs e RNAs, explica-se assim a ligação biológica do neurónio e o funcionamento do mesmo, relativamente às RNAs explica-se como é possível imitar-se as RNs.

Seguidamente aborda-se o algoritmo *backpropagation*, através dele é possível efetuar o treino de modelos profundos, pois torna-os computacionalmente exequíveis e apresentam-se as equações responsáveis pelos seus cálculos.

Apresenta-se as FAs: ReLU e Softmax, a ReLU é utilizada no treino nos MPs e a Softmax é utilizada na última camada do treino dos modelos, são assim apresentados as várias equações responsáveis pelos seus cálculos.

Para o otimizador Adam, utilizado para acelerar o algoritmo de descida do gradiente, apresentam-se as equações responsáveis pelos seus cálculos.

Para as CNNs, descreve-se o seu funcionamento e apresentam-se as duas CNNs: LeNet e AlexNet, implementadas na IG.

Apresenta-se o MP, revela-se a complexidade da sua implementação, o princípio da IV e apresentam-se as equações responsáveis pelos seus cálculos.

Para o desenvolvimento da IG, descrevem-se todas as etapas: ER, LRs, AQs, QAW, DFD, DER, DCU, MN, AU e TUs.

Capítulo 4

Interface Gráfica

Neste capítulo apresentam-se todos os métodos desenvolvidos para o desenho e implementação da IG tais como: *brainstorming*, LRs, DER, MN, Prototipagem, DCU, identificação de AQs, QAW, AU e TUs. No final do capítulo apresenta-se a IG.

4.1 Desenvolvimento da Interface Gráfica de Redes Neuronais Convolucionais Probabilísticas

4.1.1 Brainstorming para a Interface Gráfica

Após reunião com especialistas da área do ML elaborou-se o *brainstorming* cujo resultado está presente na figura 4.1. Através de vários sessões de *brainstorming*, foi possível, separar as ideias mais relevantes, ou seja, as ideias mais votadas foram consideradas mais relevantes, algumas das ideias menos votadas foram excluídas. Através desta categorização de ideias, foi possível a realização do LRs.

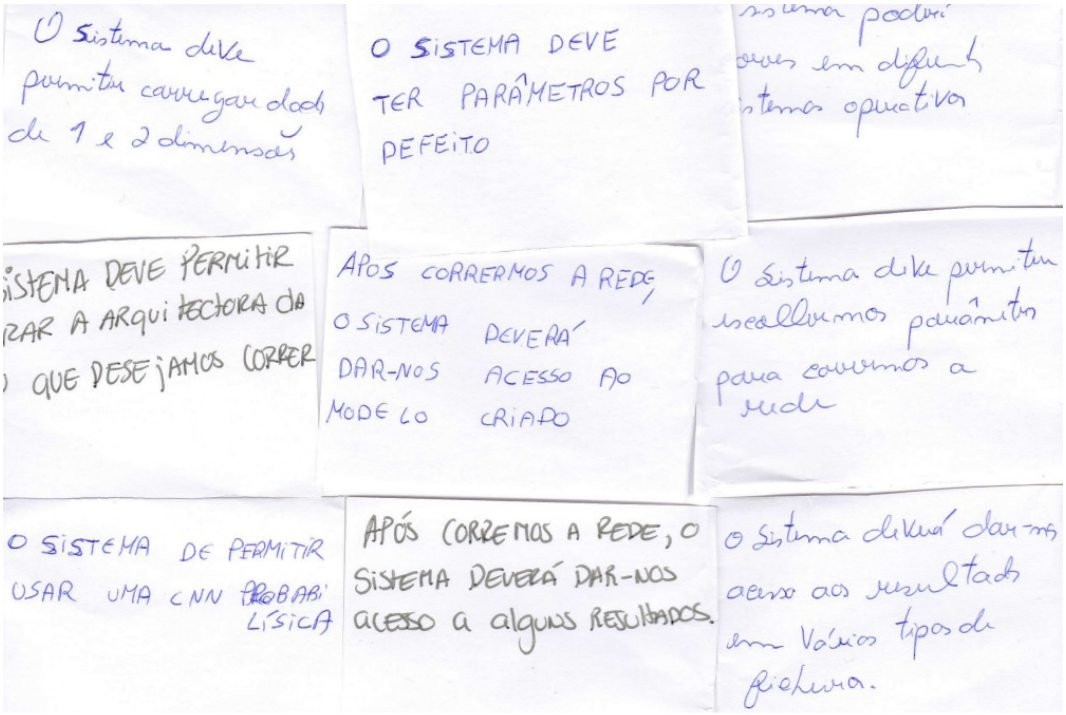


Figura 4.1: *Brainstorming* realizado na primeira fase do desenvolvimento da IG.

4.1.2 Levantamento de Requisitos para a Interface Gráfica

O LRs foi o passo seguinte no desenvolvimento da IG, sendo o resultado em reunião com especialistas da área do DL que serão posteriormente os clientes da mesma.

Os RF identificados:

- A IG permite carregar dados de 1 e 2 dimensões.
- A IG deve ter parâmetros por defeito.
- A IG deve permitir escolher parâmetros para treinar a CNN.
- A IG deve permitir utilizar uma CNN probabilística para problemas de classificação ou regressão.
- A IG permite alterar a arquitetura da CNN que deseja treinar.
- A IG deve estar sempre disponível à exceção de quando treina a rede.
- A IG deverá dar sempre *feedback* em caso de erros/falhas.
- A navegação entre janelas da IG não deverá demorar mais de 5 segundos.

- Após treinar a rede, a IG deverá dar acesso ao modelo criado.
- Após treinar a rede, a IG deverá dar acesso a resultados.

Os RNF identificados:

- A IG poderá ser executada em diferentes sistemas operativos.
- Treinar a CNN poderá ser um processo demoroso, a IG deverá dar ao utilizador *feedback* sobre o progresso neste passo.
- A IG deverá confirmar os parâmetros que inserir, antes de treinar a CNN.
- A IG dá acesso aos resultados guardados em ficheiros.

Tabela 4.1: Resumo do QAW realizado por 5 elementos.

Requisito (Resumo)	Número de votos	Atributo
Permite dados de 1 e 2 dimensões	3	Usabilidade
Ter parâmetros por defeito	3	Usabilidade
Disponível à exceção de treinar a rede	2	Disponibilidade
Feedback em caso de erros/falhas	2	Testabilidade
Navegação não excede 5 segundos	4	Desempenho
Parâmetros para treinar a rede	2	Usabilidade
Permitir utilizar-se uma CNN	5	Usabilidade
Alterar a arquitetura da CNN	2	Usabilidade
Acesso ao modelo criado	2	Disponibilidade
Acesso a resultados	3	Disponibilidade
Diferentes sistemas operativos	2	Modificabilidade
Feedback sobre o progresso	2	Disponibilidade
Confirmar os parâmetros	3	Segurança
Vários tipos ficheiros como resultado	2	Disponibilidade

Tabela 4.2: Votos por QAW para a IG.

Atributos de qualidade |	Número de votos
Usabilidade	15
Disponibilidade	11
Desempenho	4
Segurança	3
Testabilidade	2
Modificabilidade	0

Na tabela 4.1, apresentam-se os RFs mencionados anteriormente e o seu respetivo número de votos, o QAW foi realizado por cinco elementos, ou seja, cada RF poderia obter um máximo cinco votos. Na tabela 4.2, apresenta-se a quantidade de votos que cada AQ obteve, após os resultados, as maiores preocupações focaram-se nas questões da usabilidade da IG, ou seja, o quão fácil é para o utilizador realizar tarefas, desta forma a IG, foi desenvolvida para beneficiar mais o atributo de qualidade usabilidade.

4.1.3 Diagrama Entidade Relação para a Interface Gráfica

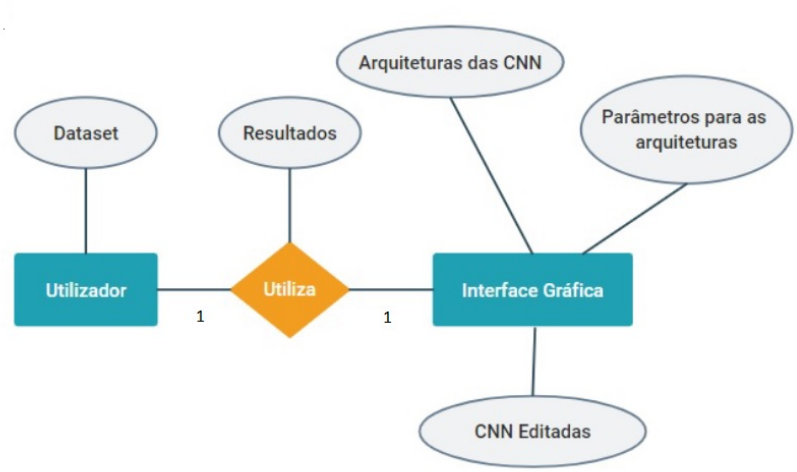


Figura 4.2: DER realizado para a IG.

Na figura 4.2 apresenta-se o DER realizado para a IG, é constituído por duas entidades denominadas por “Utilizador” e “Interface Gráfica” o “Utilizador” tem um atributo denominado por “Dataset” e a “Interface Gráfica” tem três atributos denominados por "CNN Editadas”, “Arquiteturas das CNN” e ”Parâmetros para as arquiteturas”.

4.1.4 Diagrama de Casos de Utilização para a Interface Gráfica

O DCU apresenta uma visão geral do relacionamento entre casos de utilização, utilizadores e a IG, sendo na figura 4.3 apresentado o DCU realizado para a IG.

Através do DCU, pode-se obter quais os casos de utilização da IG, ou seja, no desenvolvimento da IG, estes casos de utilização serão tidos em consideração.

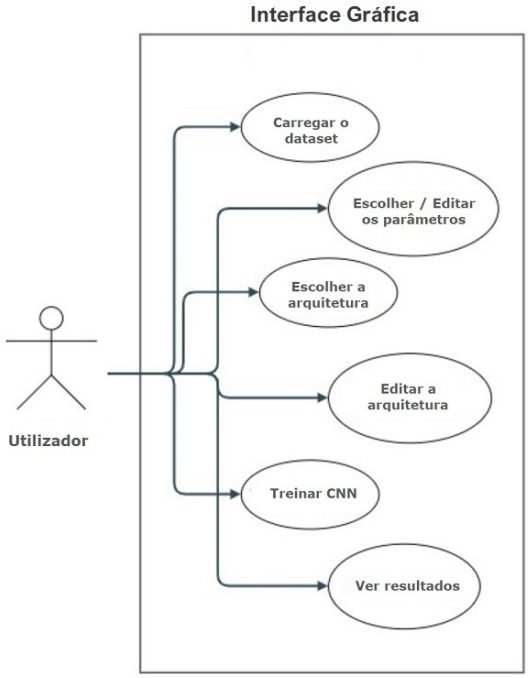


Figura 4.3: DCU desenvolvido para a IG.

4.1.5 Modelo de Navegação para Interface Gráfica

Para visualização dos cenários possíveis foi elaborado o MN, apresentado na figura 4.4. Os cenários criados são:

- Cenário 1: representa o início da IG; são exibidas informações que guiam o utilizador no processo de utilização da IG.
- Cenário 2: apresenta as questões mais frequentes para a utilização da IG.
- Cenário 3: contem informações de desenvolvimento da IG.
- Cenário 4: neste cenário devem ser inseridos os parâmetros para o modelo; os parâmetros diferem para diferentes conjuntos de dados, sendo possível escolher o modelo com arquitetura pré-definida ou criar um modelo com a arquitetura definida camada a camada.
- Cenário 5: após a escolha/edição do modelo é possível treinar o modelo.
- Cenário 6: neste cenário é possível observar os resultados de diferentes formas.

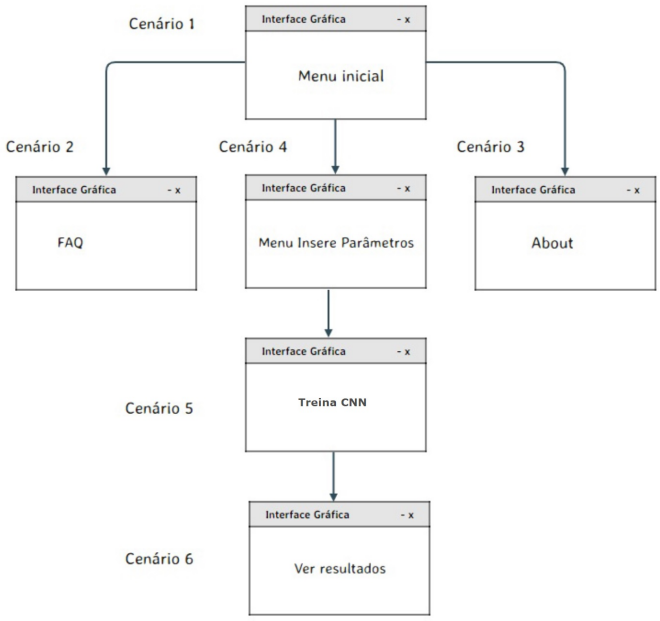


Figura 4.4: MN desenvolvido para a IG.

4.1.6 Prototipagem para a Interface Gráfica

Foram realizados protótipos de baixa e alta-fidelidade [86] durante o desenvolvimento da IG, após a realização dos TUs os protótipos de alta fidelidade sofreram algumas alterações. Na figura 4.5 apresenta-se um protótipo de baixa e alta fidelidade do separador "Start" da IG, referente à informação dos conjuntos de dados de uma dimensão. A IG, é constituída por 5 separadores denominados por: "Start", "Insert|Run", "Results", "FAQ" e "About".

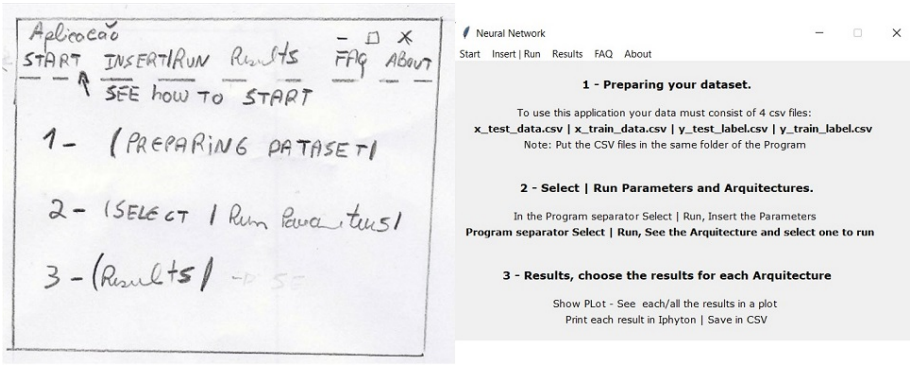


Figura 4.5: Protótipos do separador "Start".

Nas figuras 4.6 e 4.7 apresentam-se, respetivamente, os protótipos de baixa e alta

fidelidade do separador "Insert|Run" da IG, referente ao formulário de inserção de parâmetros para conjuntos de dados com uma e duas dimensões.

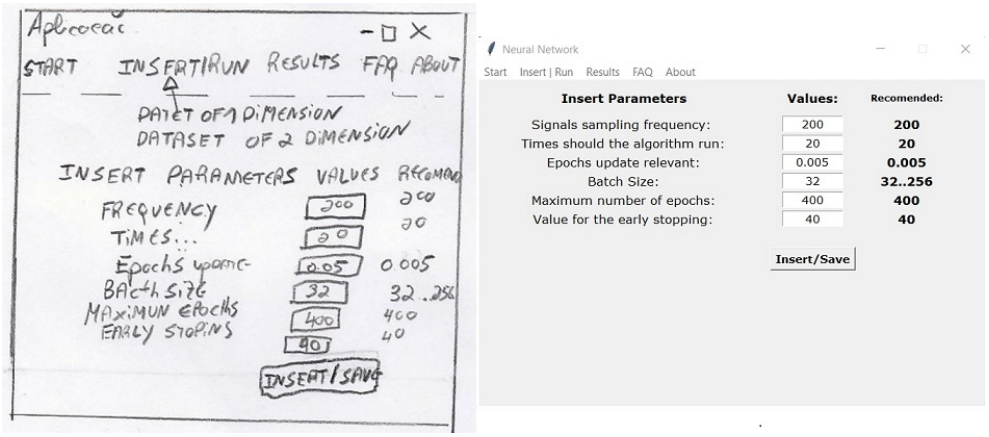


Figura 4.6: Protótipos do separador "Insert|Run" para os conjuntos de dados de uma dimensão.

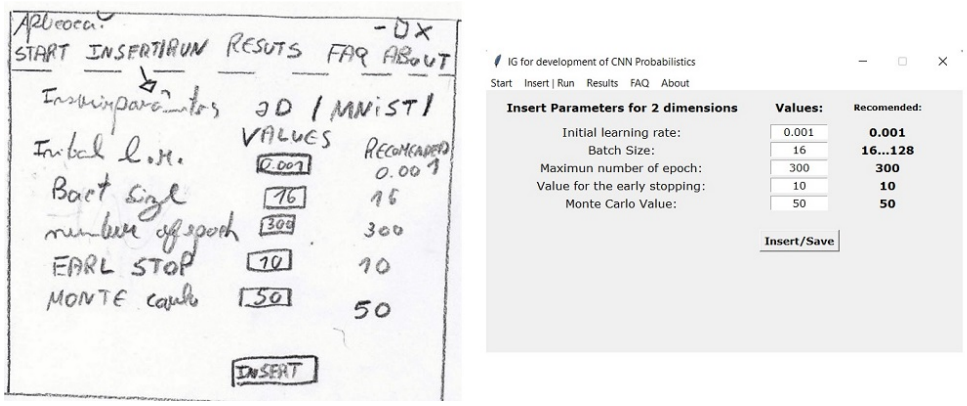


Figura 4.7: Protótipos do separador "Insert|Run", para conjuntos de dados de duas dimensões.

Após inserir os parâmetros o utilizador é levado para a página onde pode especificar a arquitetura desejada para a CNN, sendo possível treinar a mesma. Na figura 4.8 apresenta-se um protótipo de baixa e alta fidelidade do separador "Insert|Run" da IG, da seleção anteriormente mencionada.

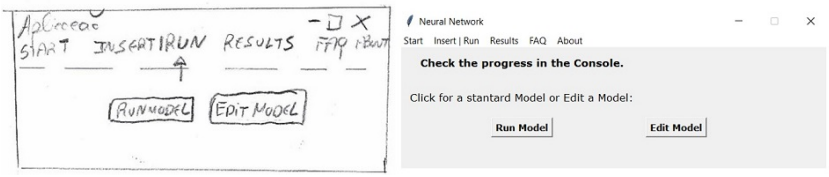


Figura 4.8: Após inserir os parâmetros poderá treinar com uma arquitetura já conhecida ou treinar o modelo.

Ao pressionar o botão "Edit Model" é possível treinar o modelo. Para tal, o utilizador deve acrescentar as camadas que deseja. Nas figuras 4.9, 4.10, 4.11 e 4.12 apresentam-se os protótipos de baixa e alta fidelidade do treino do modelo.

Os parâmetros apresentados na figura 4.9 referem-se ao formulário apresentado ao utilizador quando este inicializa o treino, adicionando uma camada *Convolution*. Para inserir uma camada de *Pooling* deverá inserir os parâmetros apresentados na figura 4.10. Para inserir-se uma camada *dense* deverá inserir os parâmetros apresentados na figura 4.11. Por fim deverá inserir a camada de *output* apresentada na figura 4.12, ao inserir esta última camada o modelo iniciará o seu treino.

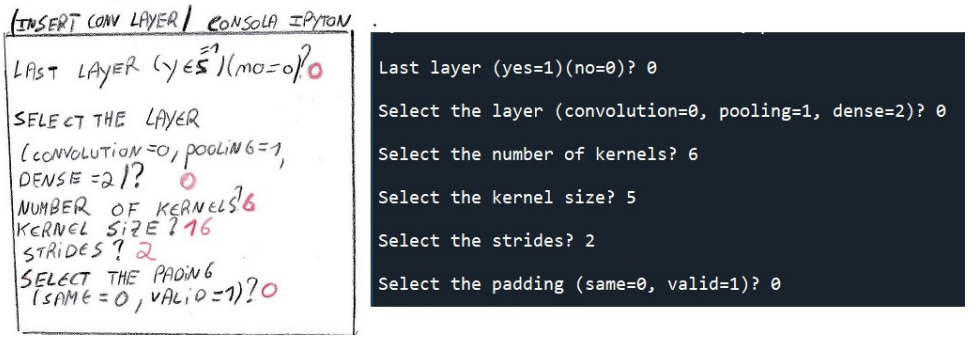


Figura 4.9: Protótipos do separador "Insert|Run, treino do modelo — camada *convolution*.

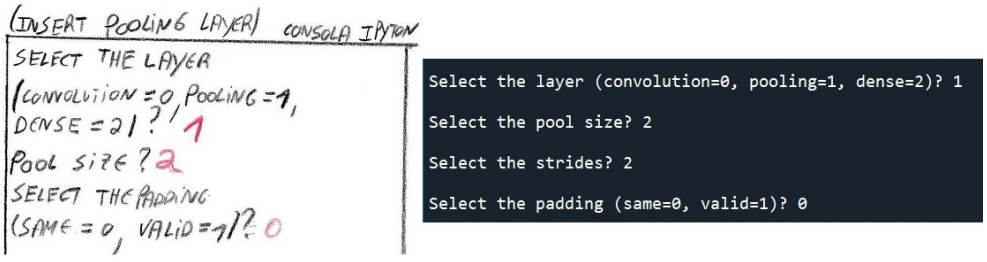


Figura 4.10: Protótipos do separador "Insert|Run, treino do modelo — camada *pooling*.

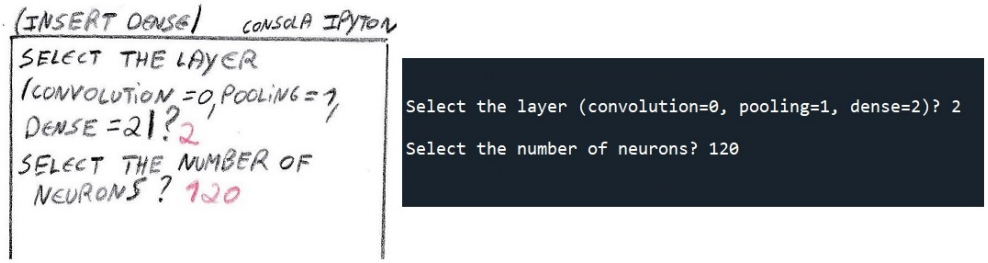


Figura 4.11: Protótipos do separador "Insert|Run, treino do modelo — camada *dense*.

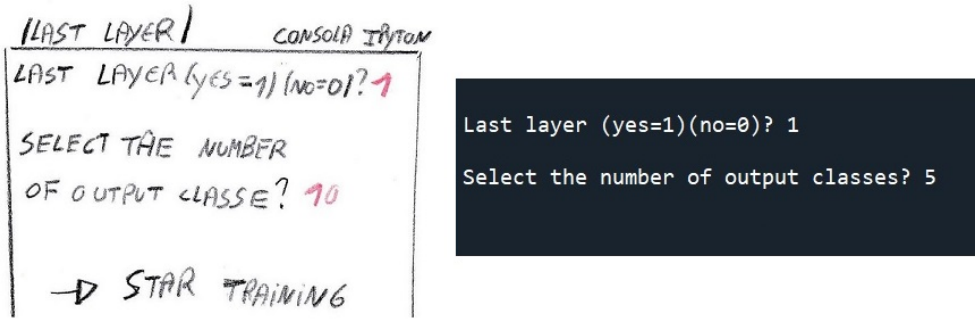


Figura 4.12: Protótipos do separador "Insert|Run, treino do modelo — camada *output*.

No separador "Results" pode visualizar-se os resultados do treino referentes à classificação e regressão dos modelos padrão (Lenet-5 e AlexNet) e do modelo treinado, dos conjuntos de dados de uma e duas dimensões. Na figura 4.13 apresenta-se um protótipo de baixa e alta fidelidade do separador "Results".

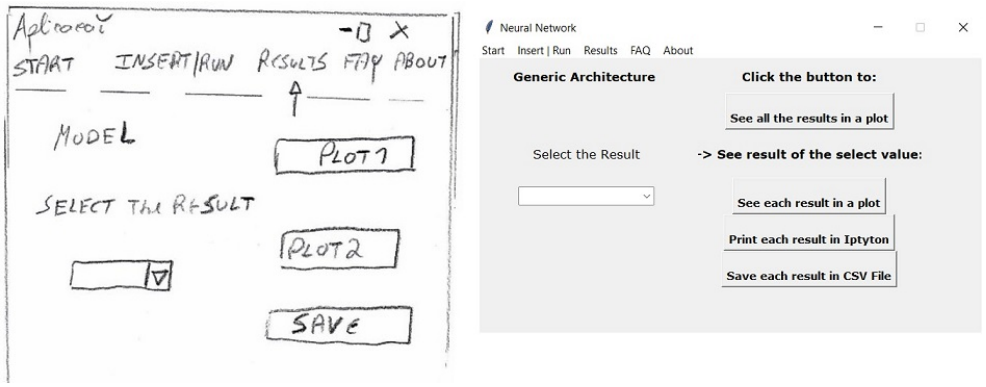


Figura 4.13: Protótipos do separador "Results".

Na figura 4.14 apresenta-se o protótipo de baixa e alta fidelidade do separador "FAQ".

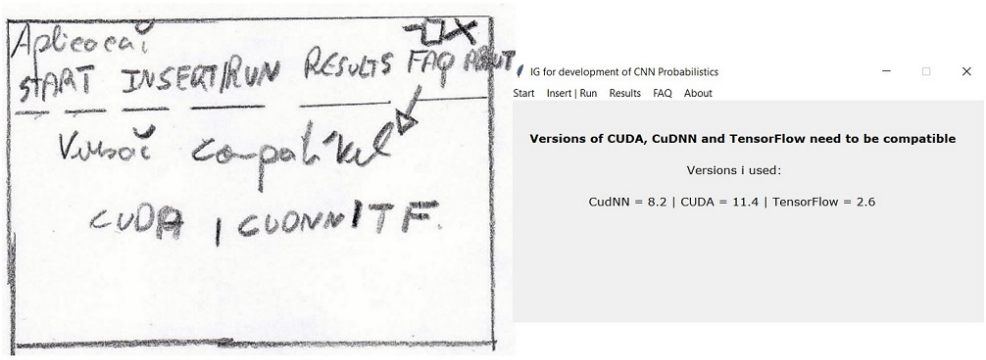


Figura 4.14: Protótipos do separador "FAQ".

Na figura 4.15 apresenta-se o protótipo de baixa e alta fidelidade do separador "About", nesta figura apresenta-se uma mensagem do desenvolvedor e a sua data de desenvolvimento.

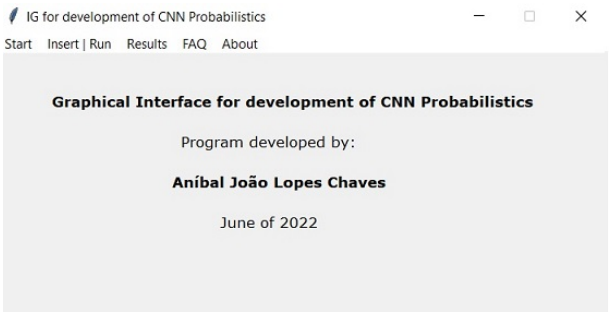


Figura 4.15: Protótipos do separador "About".

4.1.7 Avaliação de Usabilidade

Após o utilizador inserir os parâmetros para o desenvolvimento do modelo não é possível retroceder de forma simples. Esta situação não respeita a heurística número dez, para correção insere-se um botão denominado por "Back", desta forma passa a ser possível ao utilizador retroceder de forma simples. Na figura 4.16 apresenta-se a correção.

Quando o utilizar treina o modelo camada a camada pressiona o botão denominado por "Edit Model". Esta situação não respeita a heurística número dois, visto o termo mais utilizado é treinar, desta forma alterou-se "Edit Model" por "Train Model", pela mesma razão foi alterado o botão denominado por "Run Model" pe-

los botões denominados por: "Train LeNet-5" e "Train AlexNet". na figura 4.16 apresenta-se a correção.

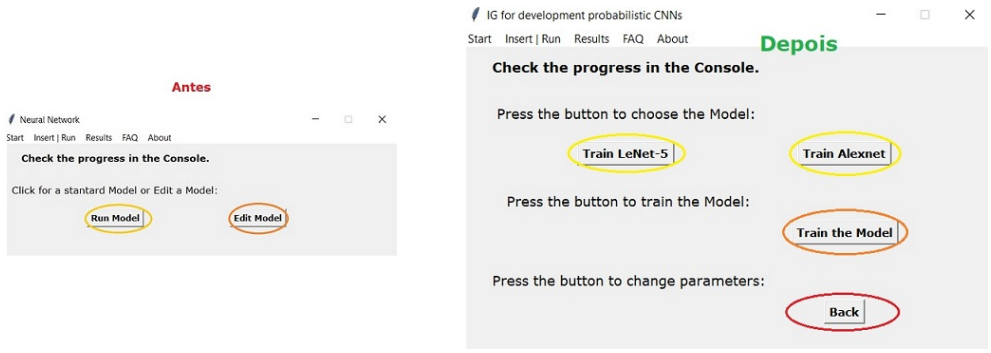


Figura 4.16: Alterações no separador denominado por "Insert|Run".

Deverá ser dado ao utilizador ajuda e documentação, o separador denominado por "FAQ" não respeita a heurística número dez, desta forma adicionou-se um botão denominado por "See Videos", desta forma o utilizador poderá ter a ajuda necessária, esta correção apresenta-se na figura 4.17

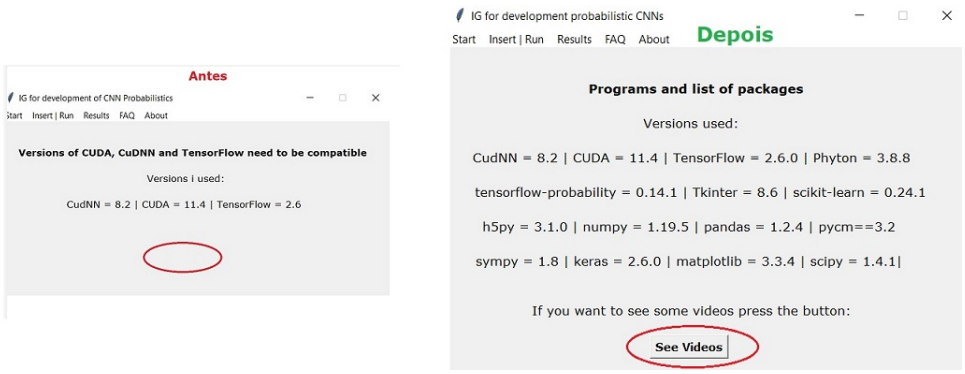


Figura 4.17: Alterações no separador denominado por "FAQ".

4.1.8 Testes de Usabilidade

Os TUs foram realizados a um conjunto de treze utilizadores com experiência em ML, desta forma elaborou-se uma tarefa e pediu-se a este conjunto de utilizadores para a realizar. Após a realização da mesma aplicou-se o formulário designado por *system usability scale* apresentado no capítulo anterior pela figura 3.20. Após o preenchimento dos formulários os resultados são apresentados pelas seguintes figuras: na figura 4.18 apresenta-se a idade dos utilizadores, na figura 4.19 apresenta-se os dados dos utilizadores tais como: género, educação e nível de experiência em ML, na figura 4.20 apresentam-se os resultados relativos às questões número um e dois, na

figura 4.21 apresentam-se os resultados relativos às questões número três e quatro, na figura 4.22 apresentam-se os resultados relativos às questões número cinco e seis, a figura 4.23 apresentam-se os resultados relativos às questões número sete e oito e na figura 4.24 apresentam-se os resultados relativos às questões número nove e dez.

Relativamente aos resultados obteve-se uma média de 4,4 em 5 possíveis, equivalente a 88%, conforme a escala fornecida pelo *system usability scale* os resultados com valor superior a 85,5% são considerados "Excellent", como se obteve 88% o resultado é classificado como "Excellent".

A média de idades dos utilizadores é de 29,4 anos.

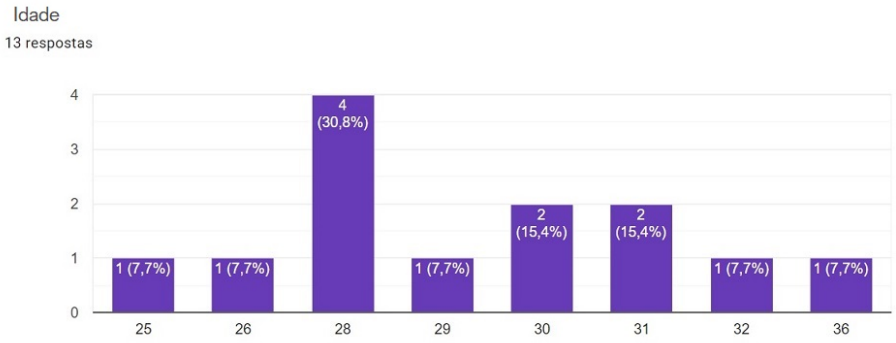


Figura 4.18: Idades dos utilizadores para os TUs.

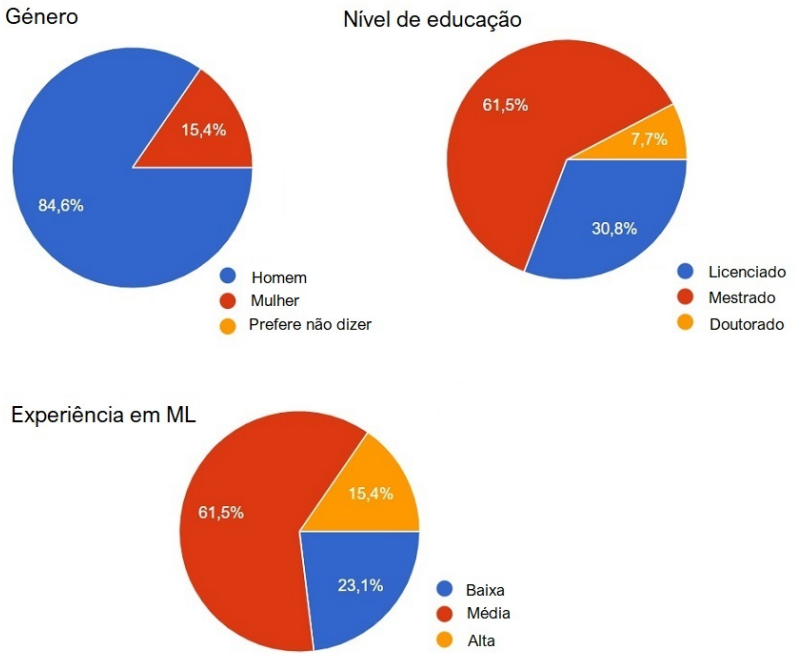


Figura 4.19: Género, nível de educação e experiência em ML dos utilizadores.

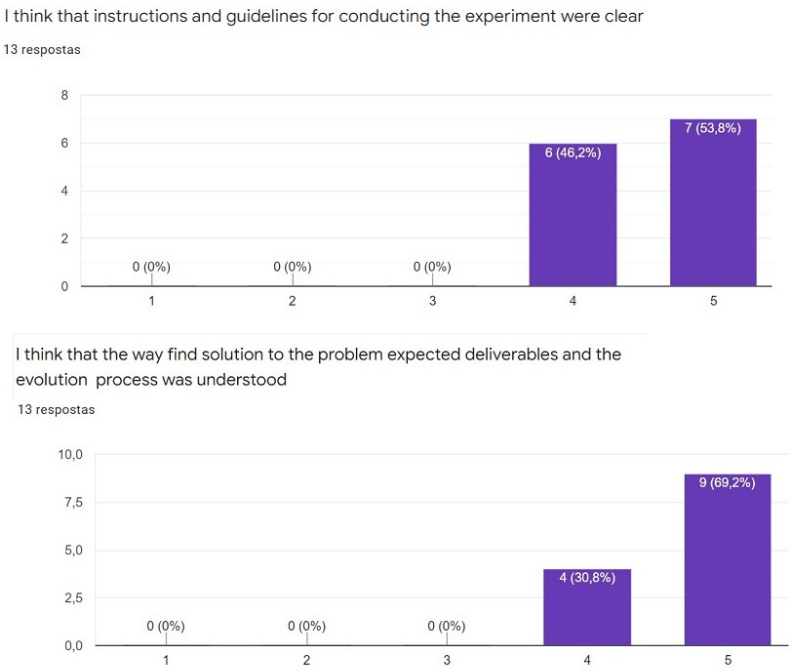


Figura 4.20: Questões número um e dois relativamente aos TUs.

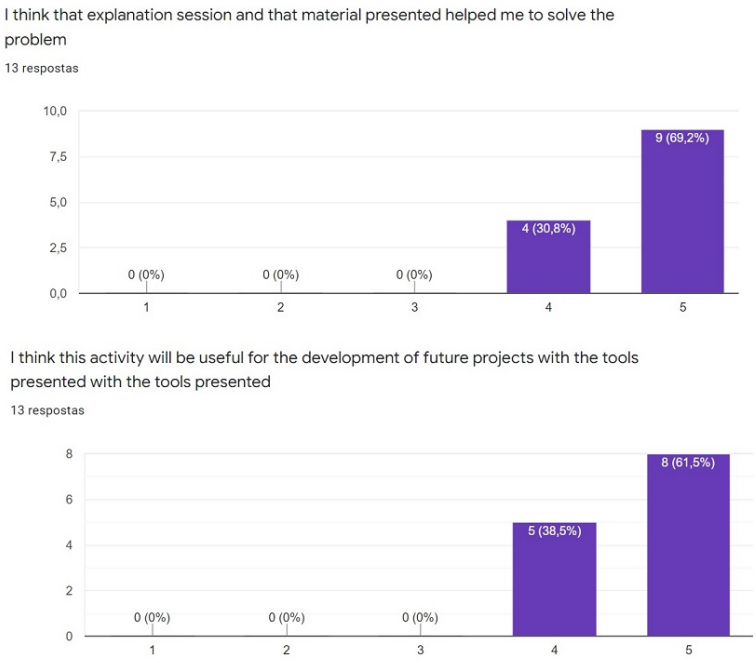


Figura 4.21: Questões número três e quatro relativamente aos TUs.

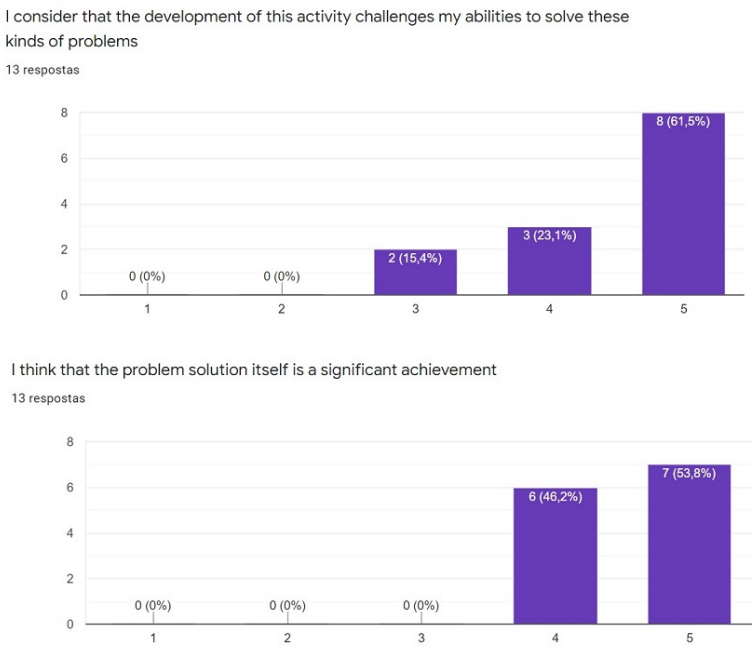


Figura 4.22: Questões número cinco e seis relativamente aos TUs.

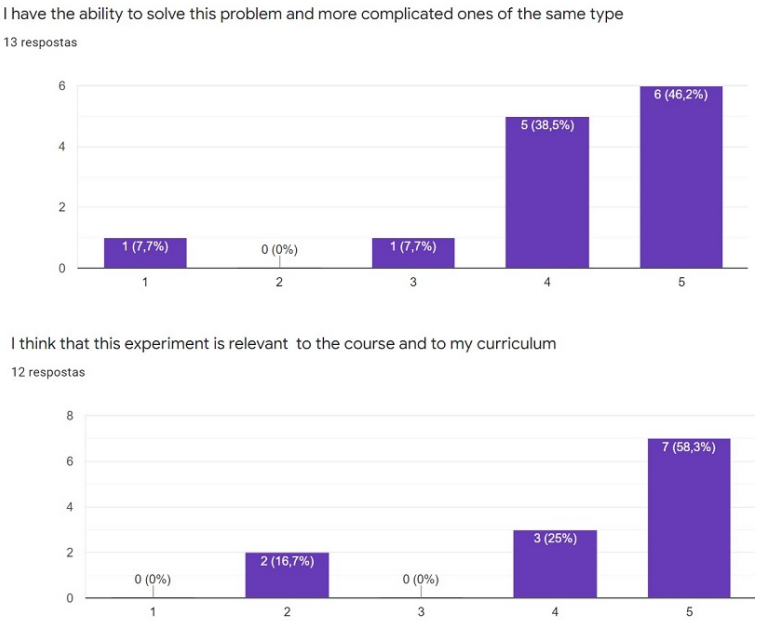


Figura 4.23: Questões número sete e oito relativamente aos TUs.

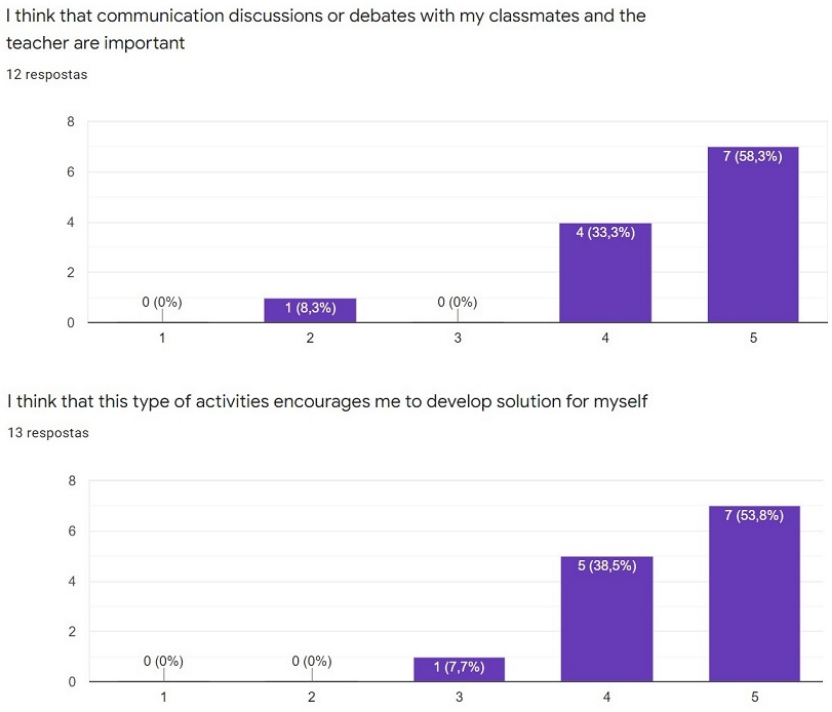


Figura 4.24: Questões número nove e dez relativamente aos TUs.

4.2 Interface Gráfica

A figura 4.25 apresenta o diagrama de utilização da aplicação do ponto de vista do utilizador. O primeiro passo é apresentado no diagrama e denomina-se por "carrega os conjuntos de dados". Caso não sejam carregados os dados o utilizador ficará limitado a algumas funcionalidades da IG. Após carregar os dados, o utilizador deverá escolher o conjunto de dados, adicionar os parâmetros, escolher o modelo/arquitetura e o treino poderá ser iniciado. Por fim, o utilizador poderá visualizar todos os resultados dos modelos/arquiteturas já treinados. São disponibilizadas duas versões uma denominada por "TesteInterface" e outra denominada por "Interface", o diagrama apresentado pela figura 4.25 é referente à versão "Interface", na versão denominada por "TesteInterface" o utilizador não realiza o carregamento dos dados.

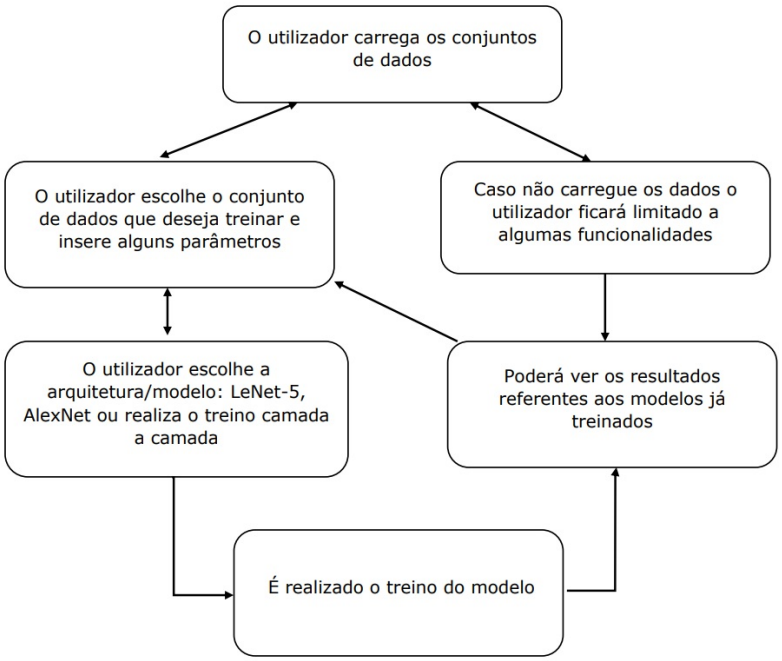


Figura 4.25: Diagrama de utilização da IG.

A aplicação é constituída por um conjunto de ficheiros e pastas, representados pela figura 4.26. Relativamente aos ficheiros:

- Interface.py: contém o código relativamente à parte gráfica da interface.
- RunModel.py: contém o código para a implementação do treino do modelo e os modelos Lenet-5 e Alexnet para todos os conjuntos de dados.
- Results: contém o código para a visualização dos resultados de todos os conjuntos de dados.
- Help.txt: contém informações relativamente ao funcionamento da IG.
- requirements.txt: contém os módulos necessários para a utilização da IG.

Relativamente às pastas:

- Results: Contem um conjunto de pastas: AlexNet1d, ALexNet2d, LeNet1d, LeNet2d, Edit1d, Edit2d, RegressionLenet, RegressionAlexNet e RegressionEdit.
- AlexNet1d: contém os resultados do modelo AlexNet para o conjunto de dados de uma dimensão.

- AlexNet2d: contém os resultados do modelo AlexNet para o conjunto de dado de duas dimensões.
- LeNet1d: contém os resultados do modelo LeNet-5 para o conjunto de dados de uma dimensão.
- LeNet2d: contém os resultados do modelo LeNet-5 para o conjunto de dados de duas dimensões.
- Edit1d: contém os resultados do modelo treinado para o conjunto de dados de uma dimensão.
- Edit2d: contém os resultados do modelo treinado para o conjunto de dados de uma dimensão.
- RegressionLenet: contém os resultados da regressão para o modelo LeNet-5.
- RegressionAlexNet: contém os resultados da regressão para o modelo AlexNet.
- RegressionEdit: contém os resultados da regressão para o modelo treinado.
- DataSet1d: contém o conjunto de dados para uma dimensão.
- DataSet2d: contém o conjunto de dados para duas dimensões.
- DataSetRegression: contém o conjunto de dado para regressão.

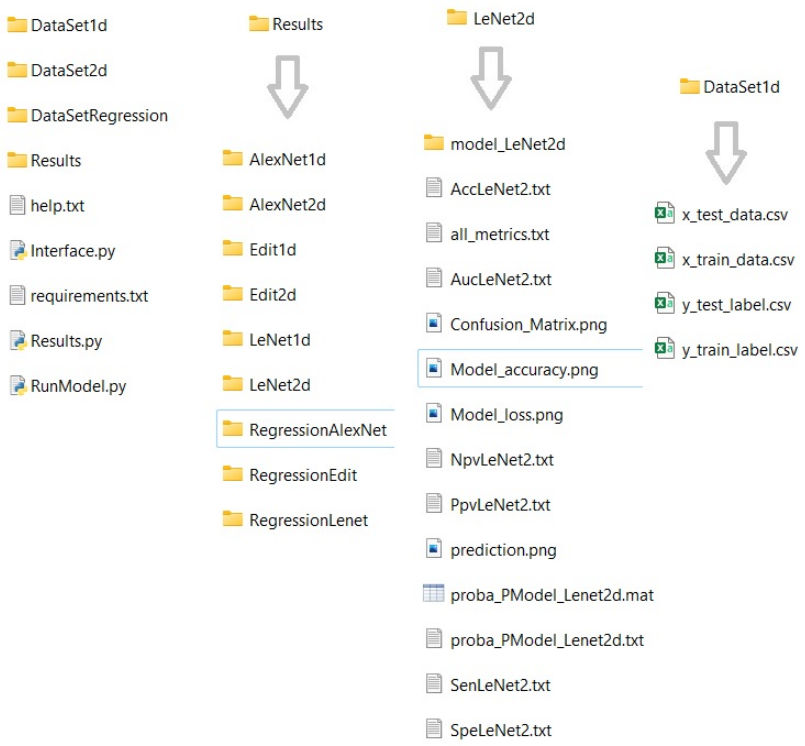


Figura 4.26: Conjunto de ficheiros e pastas da IG.

Ao abrir o ficheiro "Interface.py" é mostrada a IG representada na figura 4.27. Esta é constituída por cinco separadores:

- O primeiro separador denominado "Start" apresenta um breve resumo do funcionamento da aplicação para a utilização dos diferentes conjuntos de dados.
- No segundo separador denominado "Insert|Run", é possível editar os parâmetros para desenvolver o modelo pretendido e posteriormente treinar o modelo.
- O terceiro separador denominado "Results", permite a visualização dos resultados.
- O quarto separador denominado "FAQ", é possível visualizar-se vídeos e verificar-se as versões e bibliotecas instaladas para o funcionamento da IG.
- Por último, o quinto separador denominado "About" exibe uma mensagem referente à autoria do desenvolvimento da IG.

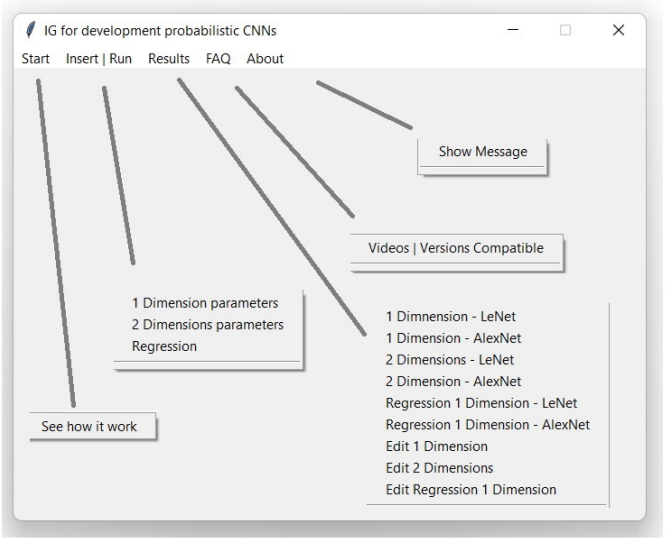


Figura 4.27: Separadores da IG.

Na figura 4.28 apresenta-se o separador denominado "Start", onde a IG fornece informações referentes à sua utilização para todos os conjuntos de dados.

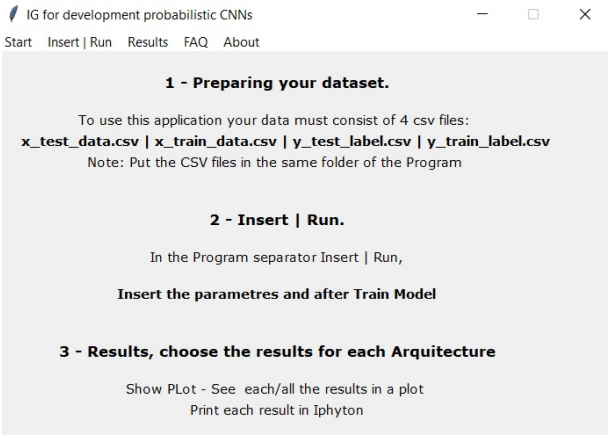


Figura 4.28: Separador "Start". Aqui apresentam-se os passos para a utilização da IG para todos os conjuntos de dados.

Na figura 4.29 apresenta-se o separador seguinte denominado por "Insert|Run". Aqui permite-se ao utilizador a introdução dos parâmetros que irão controlar o treino dos modelos. Caso seja necessário, o utilizador poderá comparar com os valores recomendados. Estes valores já vêm pré-preenchidos e caso o utilizador deseje utilizar esses valores apenas deverá pressionar no botão denominado por "Insert|Save". Caso os deseje alterar, terá de inserir os valores e por fim pressionar no botão. Ao pressionar o botão é apresentada a mensagem de confirmação na consola do IPhyton (para

confirmação visual dos parâmetros inseridos anteriormente), conforme apresentado no exemplo da figura 4.30.

O lado esquerdo da figura 4.30 é referente ao conjunto de dados de uma dimensão e o lado direito para o conjunto de dados para regressão.

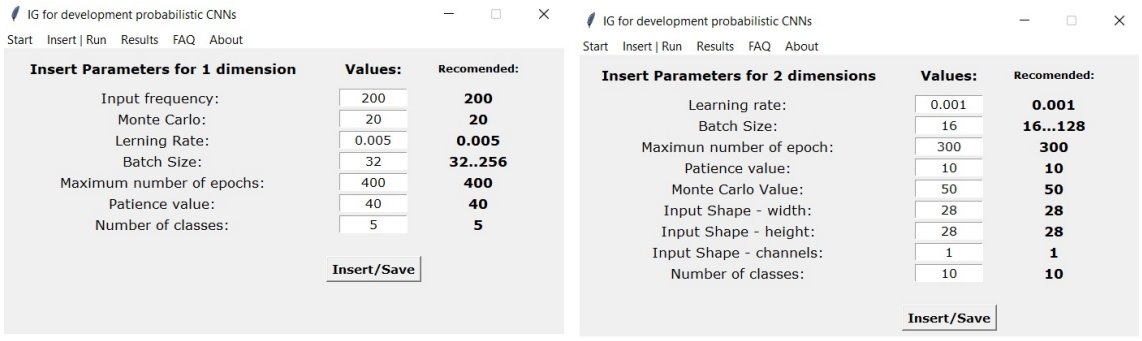


Figura 4.29: Menu para adicionar os parâmetros da IG, separador "Insert|Run".

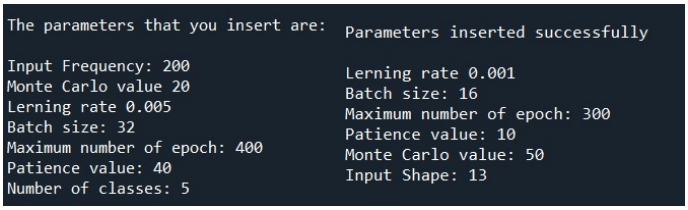


Figura 4.30: Consola IPhyton - Confirmação dos parâmetros inseridos.

Na figura 4.31 apresenta-se o separador denominado por "Insert|Run". Aqui permite-se ao utilizador selecionar que modelo deseja treinar, ou se preferir criar um modelo (criar camada a camada).

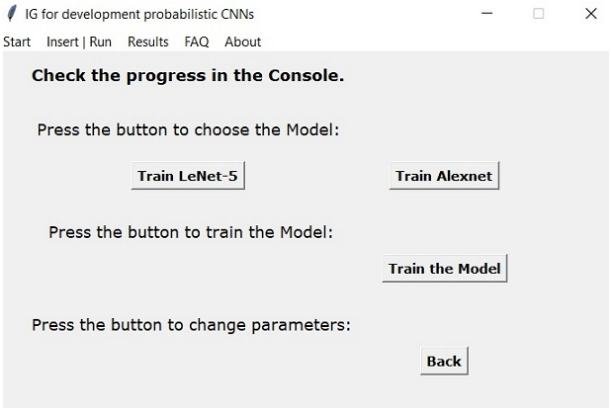


Figura 4.31: Separador "Insert|Run", após a inserção de parâmetros.

Após pressionar o botão "Train Model" o programa inicia o treino do modelo. A aplicação inicia automaticamente o modelo selecionado. Este processo poderá ser demorado. O tempo necessário para realizar o treino dependerá dos parâmetros, arquitetura e do *Hardware* utilizados.

Na figura 4.32 é apresentada a consola com indicação do progresso no treino do modelo.

Para além da janela da IG, também é possível receber *feedback* através da consola do IPython. Após pressionar no botão "Train LeNet-5" ou "Train Alexnet" é exibida a informação apresentada na Figura 4.32.

```
NVIDIA GeForce GTX 1050 Ti, pci bus id: 0000:01:00.0, compute capability: 6.1
2022-01-10 18:12:59.198377: W tensorflow/python/util/util.cc:348] Sets are not currently
considered sequences, but this may change in the future, so consider avoiding using them.
2022-01-10 18:13:12.322629: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:185]
None of the MLIR Optimization Passes are enabled (registered 2)
2022-01-10 18:13:21.449104: I tensorflow/stream_executor/cuda/cuda_dnn.cc:369] Loaded cuDNN
version 8201
98/98 [==============================] - ETA: 0s - loss: 10324.5762 - accuracy: 0.1547
```

Figura 4.32: Consola IPhyton com informação do progresso no treino do modelo.

Caso não deseje utilizar as arquiteturas padrão (Alenet ou Lenet-5), o utilizador poderá criar o modelo inserindo camada a camada, tendo de selecionar o tipo de camada; neste caso: *convolution*, *pooling* ou *dense*. A janela de inserção é apresentada na figura 4.33.

```
Last layer (yes=1)(no=0)? 0
Select the layer (convolution=0, pooling=1, dense=2)? 0
Select the number of kernels? 6
Select the kernel size? 5
Select the strides? 2
Select the padding (same=0, valid=1)? 0
Last layer (yes=1)(no=0)? 0
Select the layer (convolution=0, pooling=1, dense=2)? 1
Select the pool size? 2
Select the strides? 2
Select the padding (same=0, valid=1)? 0
Last layer (yes=1)(no=0)? 0
Select the layer (convolution=0, pooling=1, dense=2)? 2
Select the number of neurons? 84
Last layer (yes=1)(no=0)?
```

Figura 4.33: Treino do modelo, adicionar camadas: *convolution*, *pooling* e *dense*.

No separador denominado por "Results", apresentado na Figura 4.34, apresentam-se os resultados do treino dos modelos. O utilizador deve escolher o modelo para os quais já realizou o treino. Após selecionar a arquitetura podem-se visualizar os resultados.

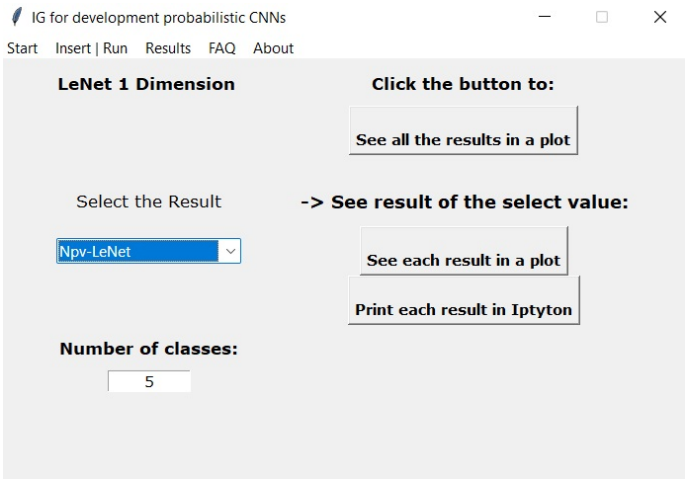


Figura 4.34: Separador "Results".

Ao pressionar "See all the results in a plot", é apresentado um único *plot* com os resultados. Ao pressionar "See each result in a plot", é apresentado um *plot* com o resultado escolhido. Ao pressionar "Print each result in IPhyton" é apresentado o valor escolhido na consola do IPhyton.

O separador denominado por "FAQ", representado na figura 4.35, apresenta informação ao utilizador sobre as bibliotecas que deverá instalar. O separador denominado "About", representado pela figura 4.36, apresenta a informação ao utilizador sobre o desenvolvimento da IG.

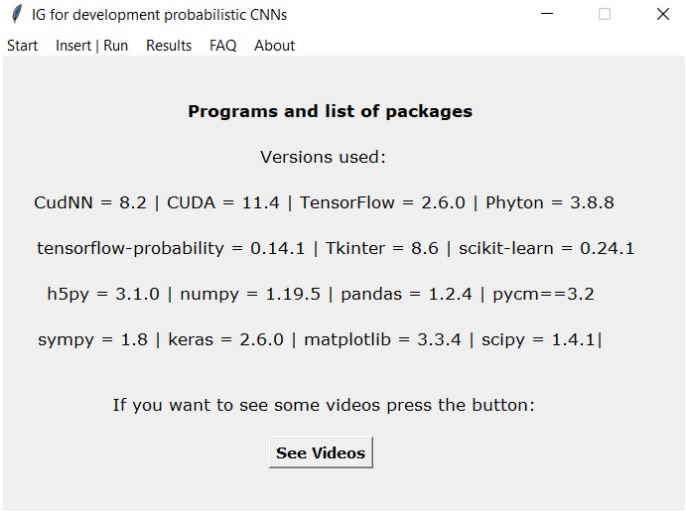


Figura 4.35: Separador FAQ. Apresentação dos módulos/bibliotecas e o botão para visualizar os vídeos.

Figura 4.36: Separador About.

Vídeos relativamente ao funcionamento da IG disponíveis em: (https://www.youtube.com/channel/UC9bZjefkicHKC6VJPGLLZJA).

4.3 Pontos-Chave

Após a realização do *brainstorming* implementou-se o LRs e categorizaram-se os requisitos através dos AQs. Em função destes efetuou-se o QAW.

Realizou-se a modelação dos dados através do DER.

Identificaram-se os casos de utilização através do DCU.

Definiram-se os cenários e a navegação na IG através da elaboração do MN.

Realizaram-se protótipos de baixa e alta fidelidade. Através dos protótipos de baixa, alta fidelidade, AU e TUs efetuaram-se algumas alterações.

No final deste capítulo é apresentada a IG e todo o seu funcionamento.

Capítulo 5

Exemplos de utilização da Interface Gráfica

Neste capítulo apresentam-se os conjuntos de dados utilizados, os recursos de *hardware* para a realização dos testes e os resultados obtidos.

5.1 Recursos de Software e Hardware

A IG foi desenvolvida no ambiente de desenvolvimento integrado Spyder [87], pois permite a utilização de Python num ambiente interativo, facilitando a edição de *scripts*, produção de teste, *debugging* e visualização gráfica.

Relativamente ao *hardware*, realizaram-se testes à IG num computador pessoal (*desktop*) com as seguintes especificações:

- Central Processing Unit (CPU): AMD Rayzen-5 3600, 6 processadores.
- GPU: NVIDIA RTX 3060, 12 GB Video Random Access Memory (V-RAM).
- Random Access Memory (RAM): 32 GB DDR4 2600 HZ.
- SSD 480GB.

Também se realizaram testes num servidor com as seguintes especificações:

- GPU: NVIDIA RTX A6000, 48 GB V-RAM.
- RAM: 64 GB DDR4.

5.2 Conjuntos de dados utilizados

Utilizaram-se um total de quatro conjuntos de dados, classificação para dados de uma e duas dimensões pois são os mais comuns e regressão fez-se a análise de séries temporais e $features$, desta forma e através dos resultados apresentados ao longo do capítulo é possível realizar a validação dos MPs criados através da IG.

Dois conjuntos de dados para regressão, sendo para tal utilizados: The Boston Housing Dataset [88] disponível em: (https://www.cs.toronto.edu/ delve/data/boston/bostonDetail.html) para determinação do preço das casas sendo utilizada regressão tendo em consideração as $features$ e o Wind Speed Prediction Dataset [89] disponível em: (https://www.kaggle.com/datasets/fedesoriano/wind-speed-prediction-dataset) para previsão da velocidade do vento através da temperatura máxima, tendo em consideração as séries temporais. Desta forma é realizada a previsão da velocidade do vento para o dia seguinte, tendo em conta os treze dias anteriores.

Dois conjuntos de dados utilizados para classificação, examinando dados com uma e duas dimensões. Neste caso, pretende-se utilizar dados relativos à determinação dos estados do sono para analisar os dados com uma dimensão, sendo para tal usado o ISRUC-SLEEP Dataset [90], disponível em: (https://sleeptight.isr.uc.pt/). Para a classificação de dados com duas dimensões usaram-se os dados relativos a imagens de números desenhados à mão [91], disponível em:(http://yann.lecun.com/exdb/mnist).

O pré-processamento dos dados deverá ser realizado antes destes serem carregados na IG, partilha-se um exemplo para cada conjunto de dados disponível em: (https://drive.google.com/drive/folders/1tJsDZMmWGPB7i0avxGEu5myptjCTAnN?usp=sharing), desta forma são partilhados os conjuntos de dados para Wind Speed Prediction Dataset; ISRUC-SLEEP Dataset e MNIST Dataset.

Na figura 5.1 apresenta-se o conjunto de dados Wind Speed Prediction Dataset, onde são apresentadas as nove classes.

Na figura 5.2 apresenta-se o conjunto de dados Boston Housing Dataset, onde são apresentadas as catorze classes, as quais serão utilizadas para determinar o preço das casas.

Na figura 5.3 apresenta-se o conjunto de dados ISRUC-SLEEP Dataset, relativos aos sinais dos vários estados do sono, são apresentados os cinco estados de sono (W, N1, N2, N3, R).

Na figura 5.4 apresenta-se o conjunto de dados MNIST Dataset para classificação, números desenhados à mão, e apresenta-se um exemplo de cada número possível.

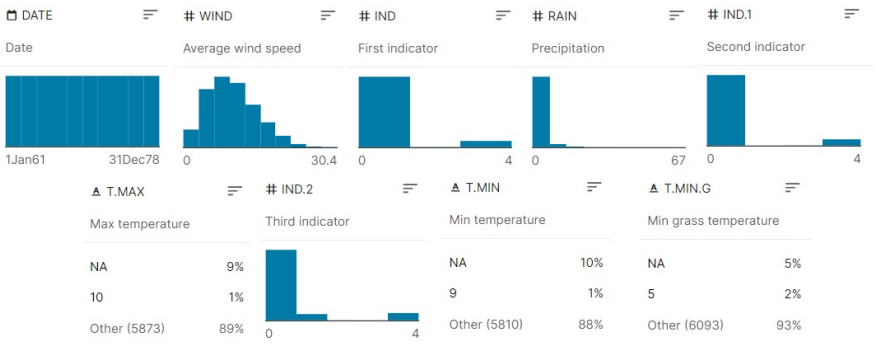


Figura 5.1: Wind Speed Prediction Dataset — nove atributos.

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33

```
:Attribute Information (in order):
    - CRIM     per capita crime rate by town
    - ZN       proportion of residential land zoned for lots over 25,000 sq.ft.
    - INDUS    proportion of non-retail business acres per town
    - CHAS     Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
    - NOX      nitric oxides concentration (parts per 10 million)
    - RM       average number of rooms per dwelling
    - AGE      proportion of owner-occupied units built prior to 1940
    - DIS      weighted distances to five Boston employment centres
    - RAD      index of accessibility to radial highways
    - TAX      full-value property-tax rate per $10,000
    - PTRATIO  pupil-teacher ratio by town
    - B        1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
    - LSTAT    % lower status of the population
    - MEDV     Median value of owner-occupied homes in $1000's
```

Figura 5.2: The Boston Housing — Catorze atributos existentes e o seu significado.

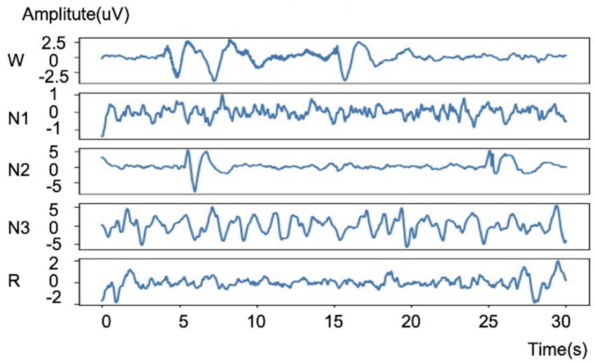


Figura 5.3: ISRUC-SLEEP — cinco estados de sono [92].

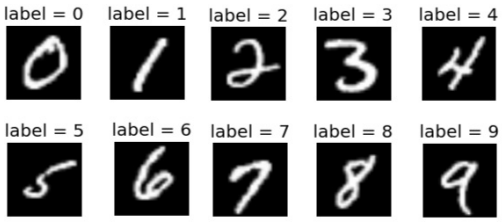


Figura 5.4: MNIST — exemplos dos números.

5.3 Métricas para análise do desempenho

A IG apresenta resultados relativos ao desempenho dos modelos desenvolvidos, sendo as métricas apresentadas definidas pelas equações: 5.1 para a Exatidão (Acc); 5.2 para a Sensibilidade (Sen); 5.3 para a Especificidade (Spe); 5.4 para o Valor Preditivo Negativo (NPV); 5.5 para o Valor Preditivo Positivo (PPV).

$$Acc = \frac{VP + VN}{VP + FN + FP + FN} \tag{5.1}$$

$$Sen = \frac{VP}{VP + FN} \tag{5.2}$$

$$Spe = \frac{VN}{VN + FP} \tag{5.3}$$

$$NPV = \frac{VN}{VN + FN} \tag{5.4}$$

$$PPV = \frac{VP}{VP + FP} \tag{5.5}$$

Os Verdadeiro Positivos (VPs) são amostras positivas que foram corretamente classificadas como positivas, enquanto os Verdadeiro Negativos (VNs) são amostras negativas que foram corretamente classificadas como negativas. Os Falso Positivos (FPs) são amostras negativas que foram incorretamente classificadas como positivas e os Falso Negativos (FNs) são amostras positivas incorretamente classificadas como negativas [93]. Acc é a percentagem de amostras corretamente classificadas (VPs e VNs). Sen, também conhecida como verdadeira taxa negativa, indica a resposta do classificador relativamente aos dados positivos. A Spe mostra como o classificador responde em dados negativos. A Spe também é conhecida como a verdadeira taxa negativa. Idealmente, um bom classificador tem valores altos em Spe, Acc e Sen [94].

Os cálculos relativos ao Erro do Modelo e ao MSE do Modelo são obtidos através das equações apresentadas no capítulo número três, o cálculo do Erro do Modelo é dado pela equação 3.12 e o cálculo do MSE pela equação 3.6. Relativamente aos

cálculos do intervalo de confiança de 95% [95] são dados pelas equações 5.6 para o limite superior e 5.7 para o limite inferior.

$$\text{Limite Superior} = \bar{X} + 1.96 \times \frac{\sigma}{\sqrt{n}} \tag{5.6}$$

$$\text{Limite Inferior} = \bar{X} - 1.96 \times \frac{\sigma}{\sqrt{n}} \tag{5.7}$$

Onde $\bar{X}$ representa a média das amostras, σ o desvio de padrão e n os dados das amostras.

5.4 Resultados

5.4.1 MNIST (números desenhados à mão)

LeNet-5 (classificação):

Conjunto de dados MNIST (números desenhados à mão): os parâmetros utilizados para treinar o modelo LeNet-5 são apresentados na figura 5.5.

Após treinar o modelo LeNet-5 são obtidos à saída do modelo os seguintes resultados: Matriz de Confusão, apresentada na figura 5.7, Erro do Modelo apresentado à direita da figura 5.6, Exatidão do Modelo apresentado à esquerda da figura 5.6, Incerteza Epistémica, apresentada na figura 5.8 e uma janela onde é possível colocar o número de uma amostra, apresentada na figura 5.9.

A função de perda utilizada foi a categorical crossentropy [96].

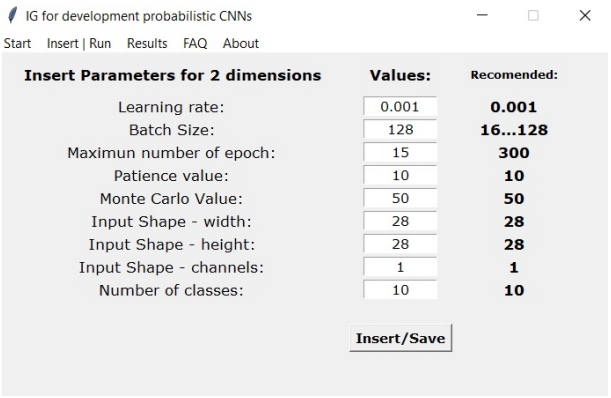


Figura 5.5: Parâmetros utilizados para o modelo LeNet-5 relativamente ao conjunto de dados de duas dimensões.

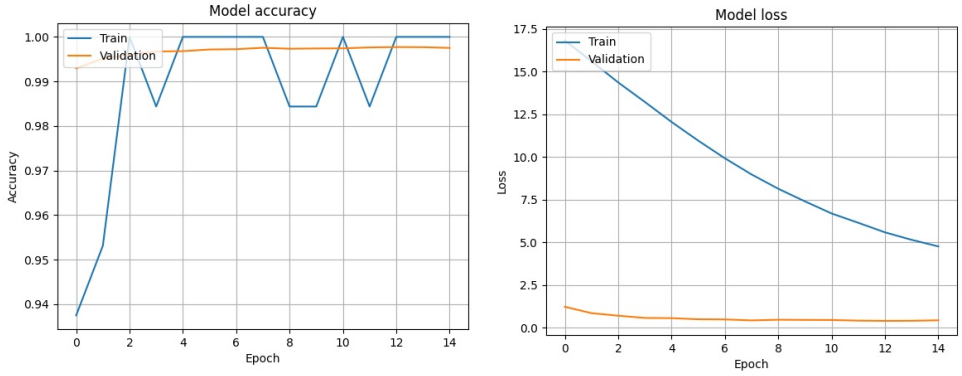


Figura 5.6: Exatidão do Modelo e Erro do Modelo relativos ao modelo LeNet-5 para o conjunto de dados de duas dimensões.

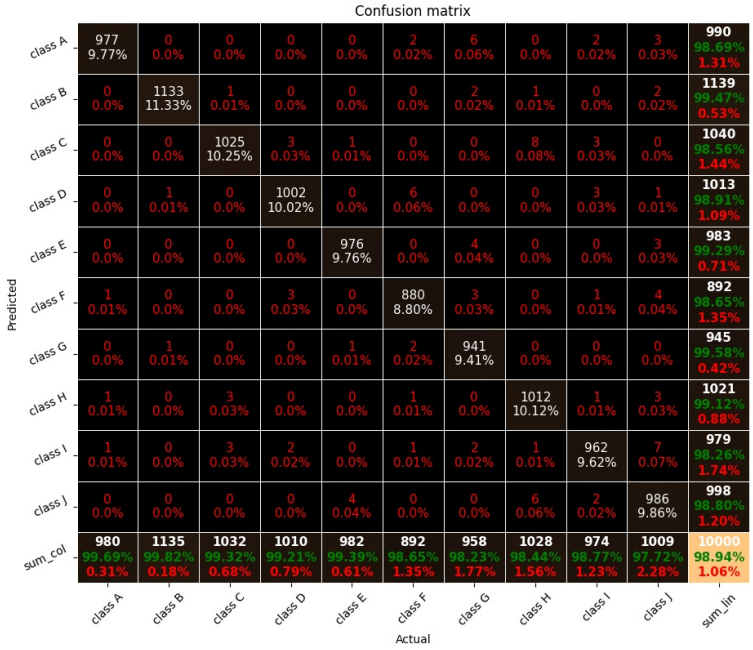


Figura 5.7: Matriz de Confusão relativa ao modelo LeNet-5 para o conjunto de dados de duas dimensões.

À esquerda, na figura 5.8, é apresentada a Incerteza Epistémica, desta forma escolheu-se um dos pontos onde a Incerteza Epistémica é mais elevada e do lado direito é efetuado *zoom* da figura para se identificar o número da amostra.

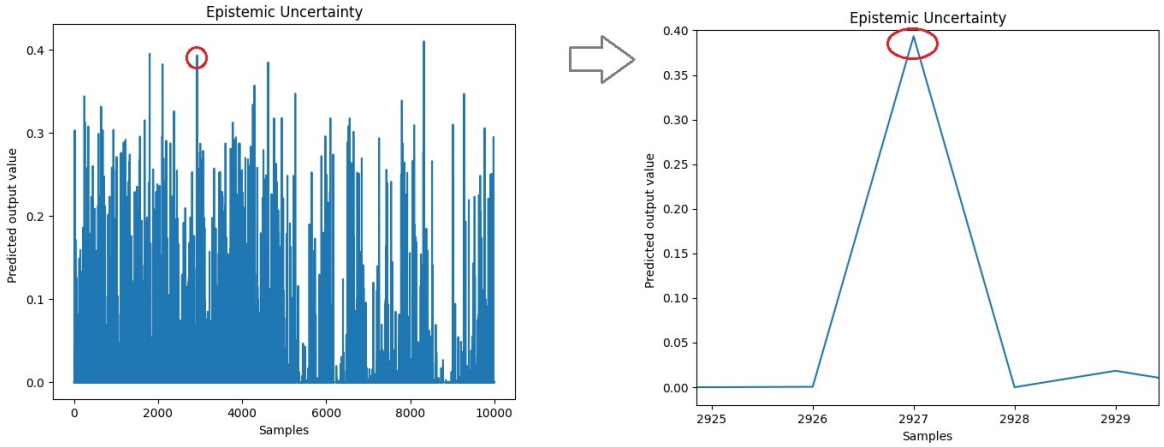


Figura 5.8: Incerteza Epistémica, identificada a amostra número: 2927, relativamente ao modelo LeNet-5 para o conjunto de dados de duas dimensões.

Através da figura 5.8 identifica-se uma amostra com um valor de Incerteza Epistémica elevado, a amostra número: 2927. Analisaremos desta forma a amostra 2927, a sua análise é apresentada pela figura 5.9.

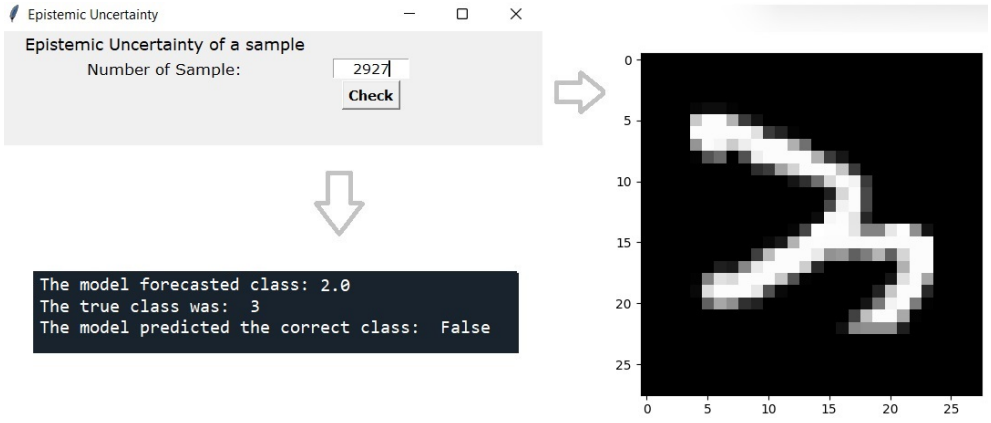


Figura 5.9: Análise da amostra número: 2927, relativamente ao modelo LeNet-5 para o conjunto de dados de duas dimensões.

Na imagem 5.9 ao pressionar o botão denominado por "Check" é apresentada a amostra, neste caso a amostra número: 2927, através do Iphyton é possível verificar-se qual a classe que o modelo classificou a amostra, qual a sua verdadeira classe e o resultado da classificação do modelo. Observa-se que: o modelo classificou a amostra número: 2927 como sendo 2, a sua classe é verdadeira é 3, ou seja, o modelo

não conseguiu classificar a amostra corretamente. Na figura 5.10 é apresentada a Incerteza Epistémica distribuída pelas diversas classes.

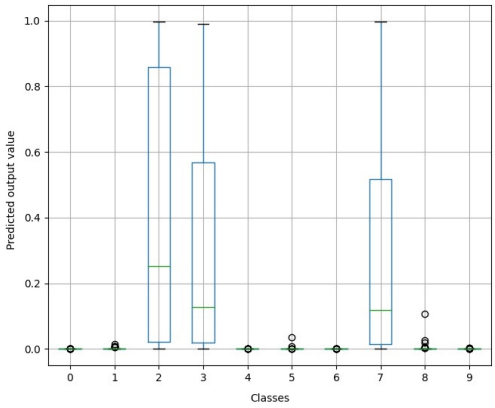


Figura 5.10: Incerteza Epistémica distribuída pelas diversas classes, relativamente ao modelo LeNet-5 para o conjunto de dados de duas dimensões.

À esquerda, na figura 5.11, é apresentada a Incerteza Epistémica, desta forma escolheu-se uma zona de pontos onde a Incerteza Epistémica é mais baixa e do lado direito é efetuado *zoom* da imagem para se identificar o número de uma amostra.

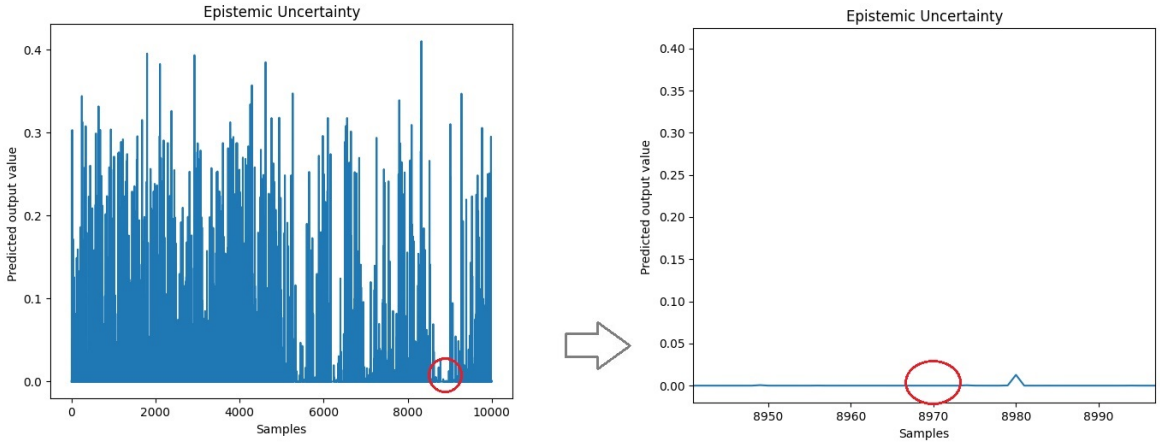


Figura 5.11: Incerteza Epistémica, identificada a amostra número: 8970, relativamente ao modelo LeNet-5 para o conjunto de dados de duas dimensões, procura de valores mais baixos.

Através da figura 5.11 verifica-se que uma das amostras com o valor de Incerteza Epistémica mais baixo é a amostra número: 8970. Analisaremos desta forma a amostra 8970, a sua análise é apresentada pela figura 5.12.

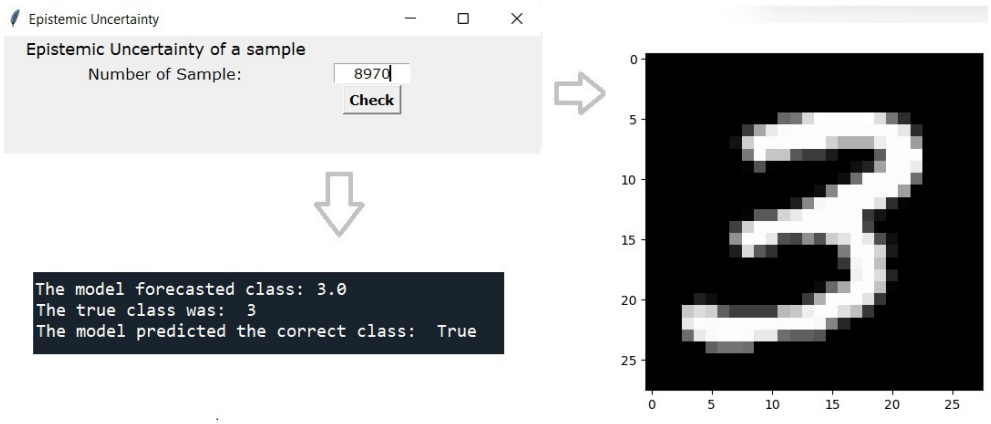


Figura 5.12: Análise da amostra número: 8970, relativamente ao modelo LeNet-5 para o conjunto de dados de duas dimensões.

Na imagem 5.12 ao pressionar o botão denominado por "Check" é apresentada a amostra, neste caso a amostra número: 8970, através do Iphyton é possível verificar qual a classe que o modelo classificou a amostra, qual a sua verdadeira classe e o resultado da classificação do modelo. Observa-se que: o modelo classificou a amostra número: 8970 como sendo 3, a sua classe é verdadeira é 3, ou seja, o modelo conseguiu classificar a amostra corretamente. Na figura 5.13 é apresentada a Incerteza Epistémica distribuída pelas diversas classes.

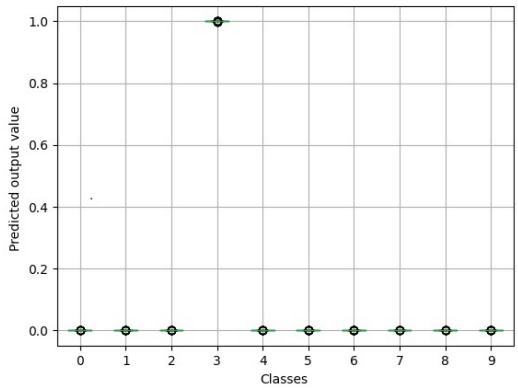


Figura 5.13: Incerteza Epistémica distribuída pelas diversas classes, relativamente ao modelo LeNet-5 para o conjunto de dados de duas dimensões para a amostra número: 8970.

No separador denominado por "Results" ao pressionar "See all the results in a plot", é apresentado um único plot numa nova janela com os resultados: Acc, Área

Sob a Curva (AUC), NPV, PPV, Sen e Spe. O resultado obtido é apresentado à esquerda, na Figura 5.15. Do lado direito apresenta-se o resultado referente ao "See each result in a plot" do Acc.

No mesmo separador ao pressionar "Print each result in IPyton" são impressos na consola do IPyton os valores do resultado escolhido estes valores são apresentados na figura 5.14.

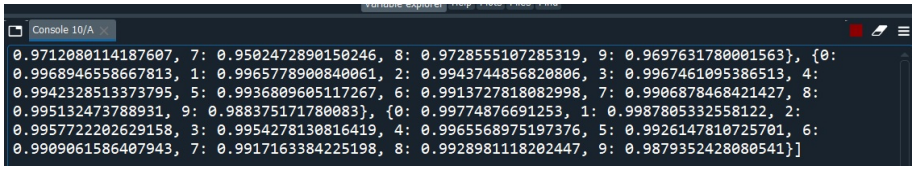


Figura 5.14: Resultado "Print each result in IPyton" relativos ao modelo LeNet-5 para o conjunto de dados de duas dimensões.

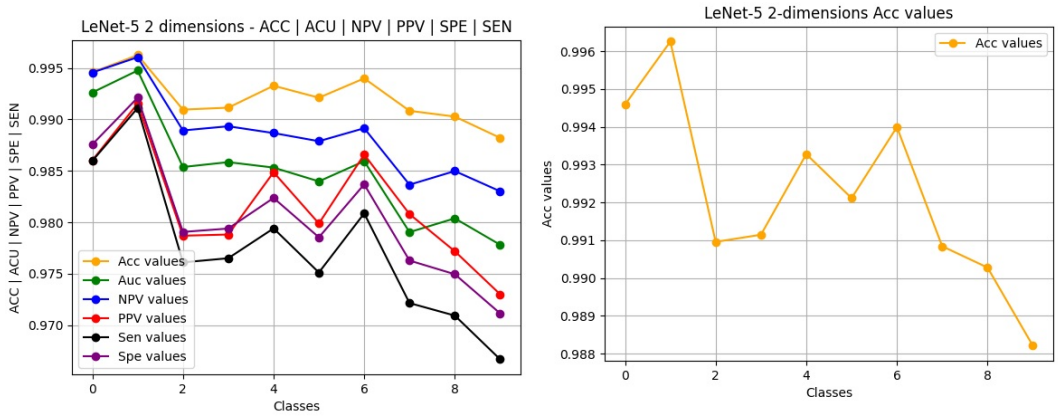
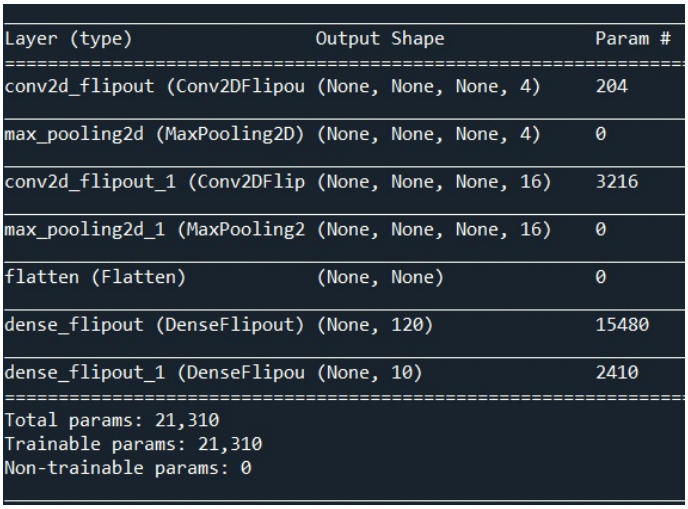


Figura 5.15: Resultado "See all the results in a plot" e "See each result in a plot" relativos ao modelo LeNet-5 para o conjunto de dados de duas dimensões.

LeNet-4 (classificação):

Relativamente ao treino do modelo, foi desenvolvido o modelo com a arquitetura do LeNet-4 [97]. O desenvolvimento do modelo é apresentado na figura 5.17. Este modelo é constituído por duas camadas *convolution*, duas de *pooling* e duas *dense*, uma das camadas *dense* é a camada de *output*. Os parâmetros utilizados para treinar o modelo LeNet-4 são os mesmos utilizados anteriormente, apresentados na figura 5.5.

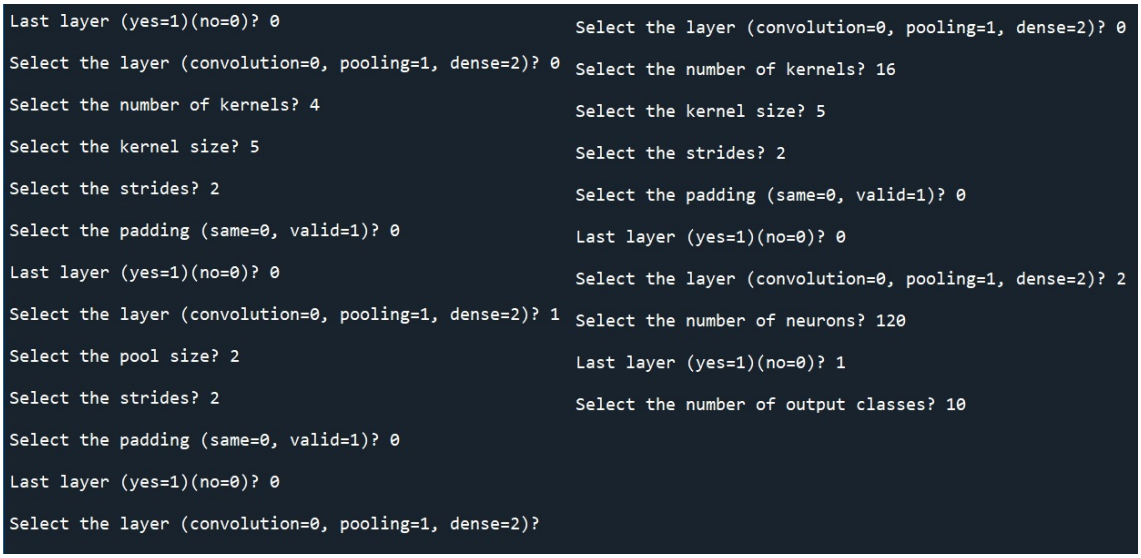
Após treinar o modelo LeNet-4 são obtidos os seguintes resultados: Sumário do Modelo, apresentado na figura 5.16, Matriz de Confusão, apresentada na figura 5.18, Erro do Modelo apresentado à direita na figura 5.19 e a Exatidão do Modelo apresentado à esquerda da figura 5.19.



Layer (type)	Output Shape	Param #
conv2d_flipout (Conv2DFlipou	(None, None, None, 4)	204
max_pooling2d (MaxPooling2D)	(None, None, None, 4)	0
conv2d_flipout_1 (Conv2DFlip	(None, None, None, 16)	3216
max_pooling2d_1 (MaxPooling2	(None, None, None, 16)	0
flatten (Flatten)	(None, None)	0
dense_flipout (DenseFlipout)	(None, 120)	15480
dense_flipout_1 (DenseFlipou	(None, 10)	2410

Total params: 21,310
Trainable params: 21,310
Non-trainable params: 0

Figura 5.16: Sumário do Modelo para o treino do modelo, relativamente ao conjunto de dados de duas dimensões, LeNet-4.



```
Last layer (yes=1)(no=0)? 0
Select the layer (convolution=0, pooling=1, dense=2)? 0
Select the number of kernels? 4
Select the kernel size? 5
Select the strides? 2
Select the padding (same=0, valid=1)? 0
Last layer (yes=1)(no=0)? 0
Select the layer (convolution=0, pooling=1, dense=2)? 1
Select the pool size? 2
Select the strides? 2
Select the padding (same=0, valid=1)? 0
Last layer (yes=1)(no=0)? 0
Select the layer (convolution=0, pooling=1, dense=2)?
Select the layer (convolution=0, pooling=1, dense=2)? 0
Select the number of kernels? 16
Select the kernel size? 5
Select the strides? 2
Select the padding (same=0, valid=1)? 0
Last layer (yes=1)(no=0)? 0
Select the layer (convolution=0, pooling=1, dense=2)? 2
Select the number of neurons? 120
Last layer (yes=1)(no=0)? 1
Select the number of output classes? 10
```

Figura 5.17: Desenvolvimento do modelo LeNet-4 relativamente ao conjunto de dados de duas dimensões.

Figura 5.18: Matriz de Confusão relativa ao modelo LeNet-4 para o conjunto de dados de duas dimensões.

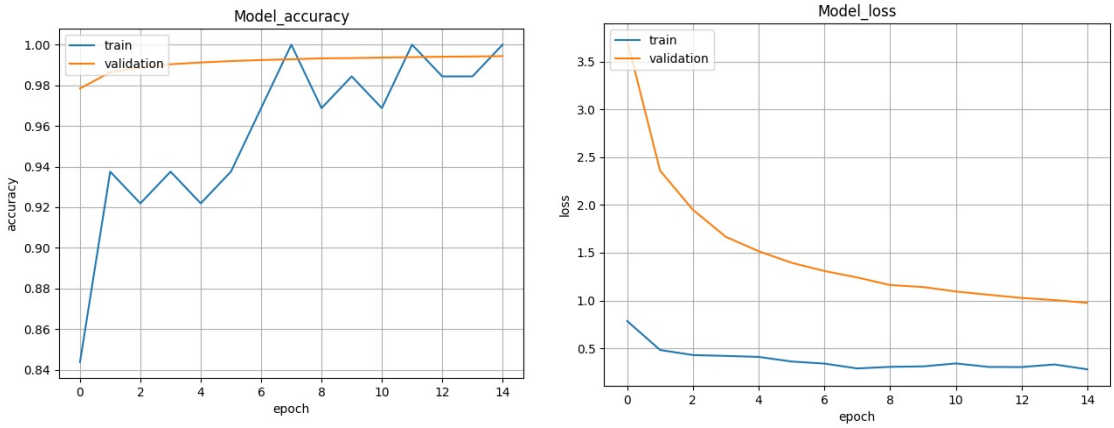


Figura 5.19: Exatidão do Modelo e Erro do Modelo relativos ao modelo LeNet-4 para o conjunto de dados de duas dimensões.

No separador denominado por "Results" ao pressionar "See all the results in a plot", o resultado obtido é apresentado na Figura 5.20. À esquerda apresenta-se o

resultado referente ao "See all the results in a plot" e do lado direito apresenta-se o resultado referente ao "See each result in a plot" do Acc. No mesmo separador ao pressionar "Print each result in IPyton" é impresso na consola do IPyton os valores do resultado escolhido estes valores são apresentados na figura 5.21.

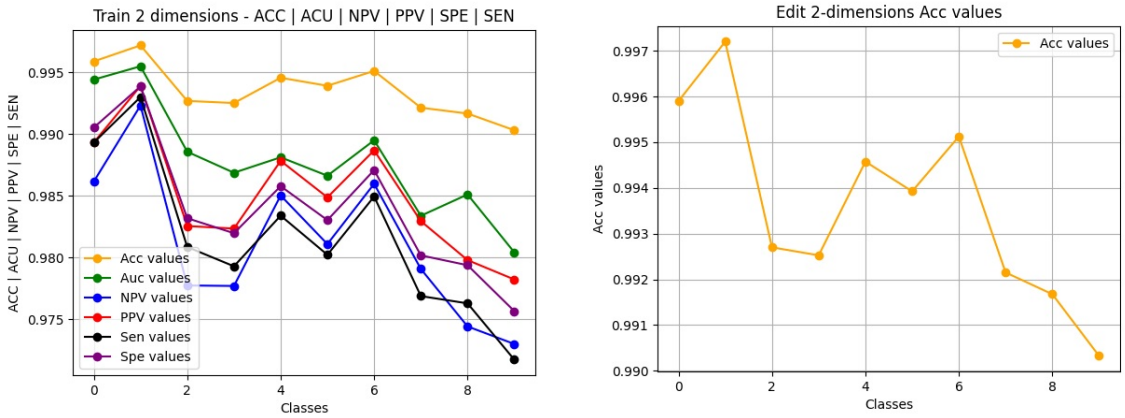


Figura 5.20: Resultado "See all the results in a plot" e "See each result in a plot" relativos ao modelo LeNet-4 para o conjunto de dados de duas dimensões.

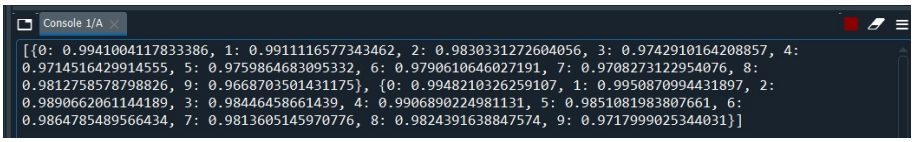


Figura 5.21: Resultado "Print each result in IPyton- LeNet-4, conjunto de dados de duas dimensões.

5.4.2 ISRUC-SLEEP (estados do sono)

LeNet-5 (classificação):

Conjunto de dados ISRUC-SLEEP (estados do sono): os parâmetros utilizados para treinar o modelo LeNet-5 são apresentados na figura 5.22. Após treinar o modelo LeNet-5 são obtidos os seguintes resultados: Matriz de Confusão, apresentada na figura 5.23, Erro do Modelo, apresentado à direita da figura 5.24, Exatidão do Modelo, apresentado à esquerda da figura 5.24, Incerteza Epistémica, apresentada na figura 5.25 e uma janela onde é possível colocar o número de uma amostra, apresentada na figura 5.26. A função de perda utilizada foi a categorical crossentropy.

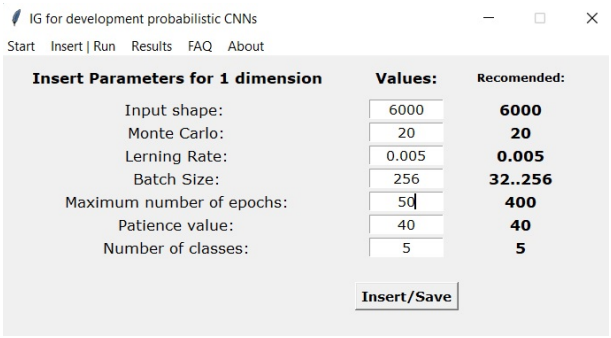


Figura 5.22: Parâmetros utilizados para o modelo LeNet-5 para o conjunto de dados de uma dimensão.

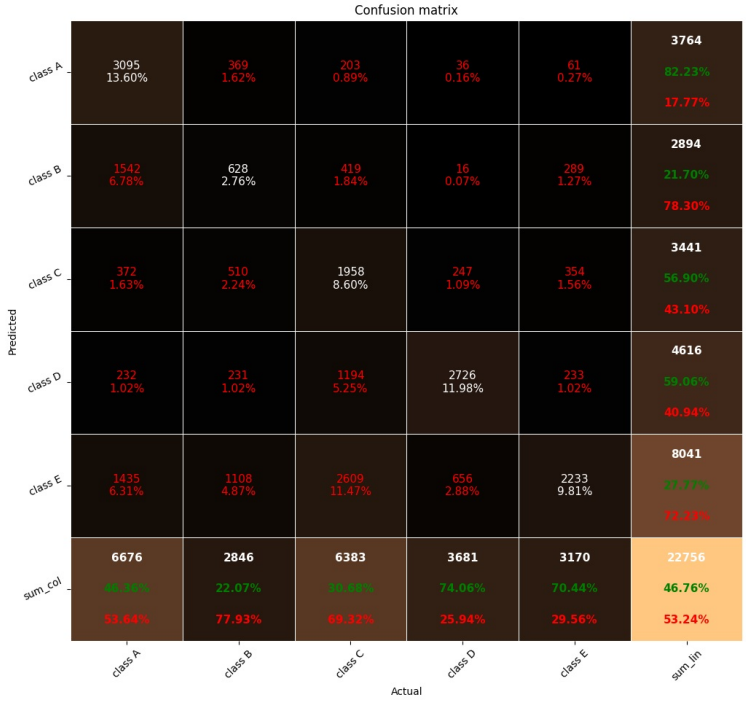


Figura 5.23: Matriz de Confusão relativamente ao modelo LeNet-5 para o conjunto de dados de uma dimensão.

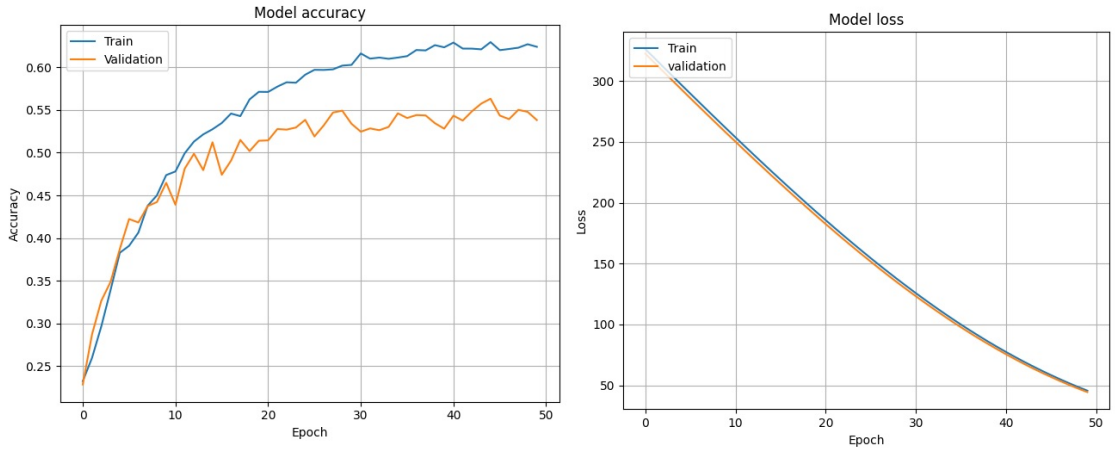


Figura 5.24: Exatidão do Modelo e Erro do Modelo relativos ao modelo Lenet-5 para o conjunto de dados de uma dimensão.

À esquerda da figura 5.25 é apresentada a Incerteza Epistémica, desta forma escolheu-se um dos pontos onde a Incerteza Epistémica é mais elevada e do lado direito é efetuado *zoom* da figura para identificar-se o número da amostra.

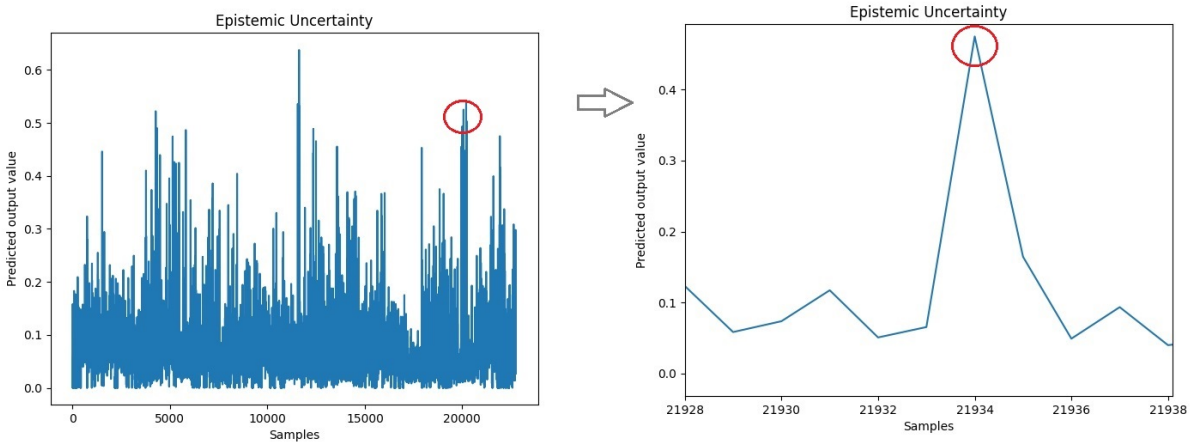


Figura 5.25: Incerteza Epistémica, identificada a amostra número: 21934, relativamente ao modelo LeNet-5 para o conjunto de dados de uma dimensão.

Através da figura 5.25 identifica-se uma amostra com um valor de Incerteza Epistémica elevado, a amostra número: 21934. Analisaremos desta forma a amostra 21934, a sua análise é apresentada pela figura 5.26.

Figura 5.26: Análise da amostra número: 21934, relativamente ao modelo LeNet-5 para o conjunto de dados de uma dimensão.

Na imagem 5.26 ao pressionar o botão denominado por "Check" é apresentada a amostra, neste caso a amostra número: 21934, através do Iphyton é possível verificar-se qual a classe que o modelo classificou a amostra, qual a sua verdadeira classe e o resultado da classificação do modelo. Observa-se que: o modelo classificou a amostra número: 21934 como sendo 1, a sua classe é verdadeira é 0, ou seja, o modelo não conseguiu classificar a amostra corretamente. Na figura 5.27 é apresentada a Incerteza Epistémica distribuída pelas diversas classes.

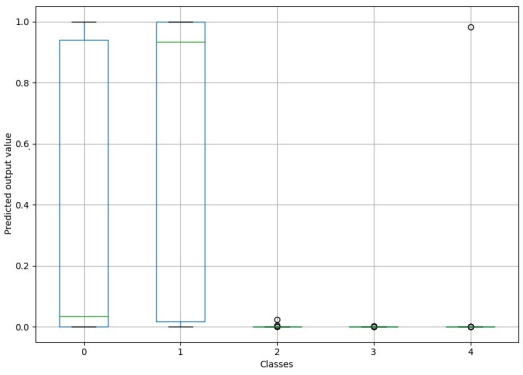


Figura 5.27: Incerteza Epistémica distribuída pelas diversas classes, relativamente ao modelo LeNet-5 para o conjunto de dados de uma dimensão.

À esquerda, na figura 5.28, é apresentada a Incerteza Epistémica, desta forma escolheu-se uma zona de pontos onde a Incerteza Epistémica é mais baixa e do lado direito é efetuado *zoom* da imagem para se identificar o número de uma amostra.

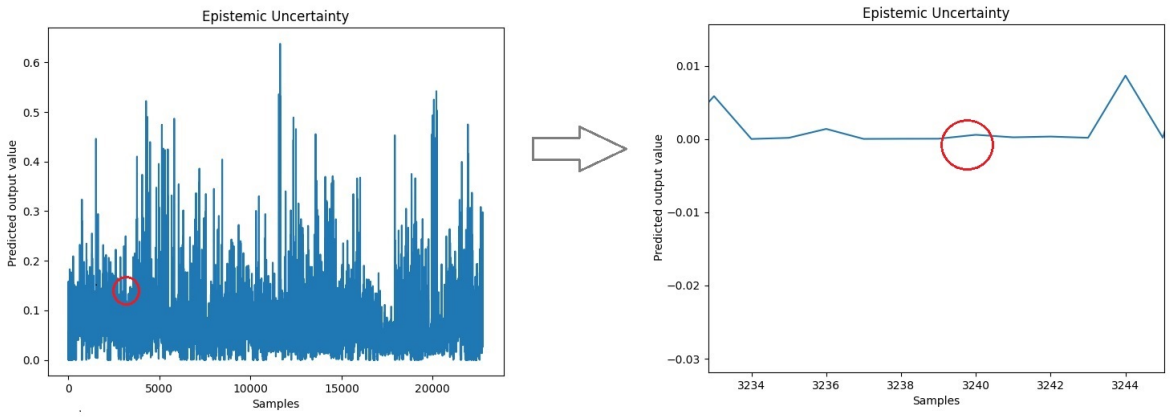


Figura 5.28: Incerteza Epistémica, identificada a amostra número: 3240, relativamente ao modelo LeNet-5 para o conjunto de dados de uma dimensão, procura de valores mais baixos.

Através da figura 5.28 verifica-se que uma das amostras com o valor de Incerteza Epistémica mais baixo é a amostra número: 3240. Analisaremos desta forma a amostra 3240, a sua análise é apresentada pela figura 5.29.

Figura 5.29: Análise da amostra número: 3240, relativamente ao modelo LeNet-5 para o conjunto de dados de uma dimensão.

Na imagem 5.29 ao pressionar o botão denominado por "Check" é apresentada a amostra, neste caso a amostra número: 3240, através do Iphyton é possível verificar-se qual a classe que o modelo classificou a amostra, qual a sua verdadeira classe e o resultado da classificação do modelo. Observa-se que: o modelo classificou a amostra número: 3240 como sendo 3, a sua classe é verdadeira é 3, ou seja, o modelo conseguiu classificar a amostra corretamente. Na figura 5.13 é apresentada a Incerteza Epistémica distribuída pelas diversas classes.

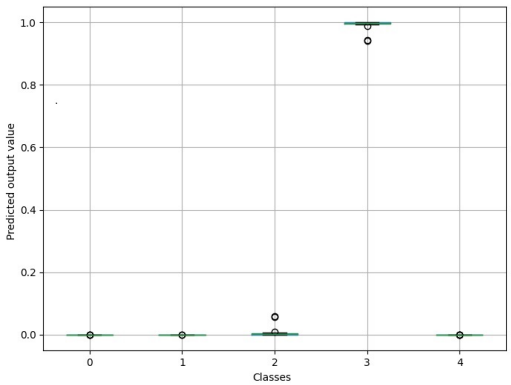


Figura 5.30: Incerteza Epistémica distribuída pelas diversas classes, relativamente ao modelo LeNet-5 para o conjunto de dados de uma dimensão para a amostra número: 3240.

No separador denominado por "Results" ao pressionar "See all the results in a plot" o resultado obtido é apresentado à esquerda da Figura 5.31, do lado direito apresenta-se o resultado referente ao "See each result in a plot" do Acc.

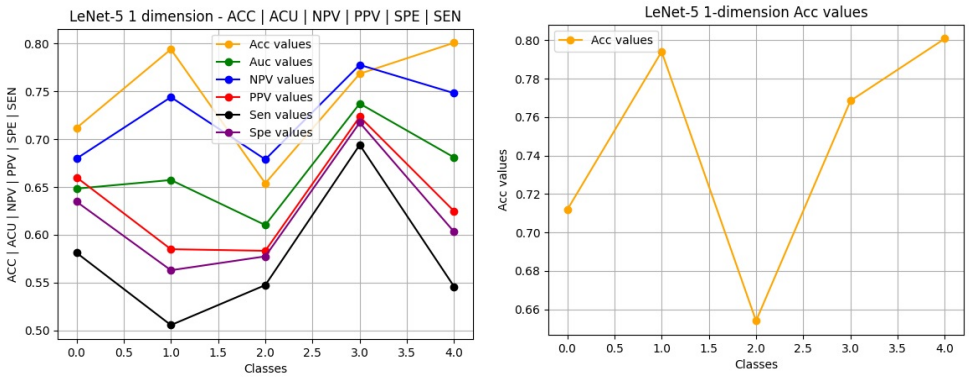


Figura 5.31: Resultado "See all the results in a plot" e "See each result in a plot" relativamente ao modelo LeNet-5 para o conjunto de dados de uma dimensão.

LeNet-4 (classificação):

Relativamente ao treino do modelo, foi desenvolvido o modelo com a arquitetura do LeNet-4, o desenvolvimento do modelo é apresentado na figura 5.17, à exceção do número de classes sendo neste caso igual a cinco, os parâmetros utilizados para treinar o modelo LeNet-4 são os mesmo utilizados anteriormente apresentados na figura 5.22. Após treinar o modelo LeNet-4 são dados à saída do modelo os seguintes resultados: Sumário do Modelo, apresentado pela figura 5.16, Matriz de Confusão,

apresentada na figura 5.32, Erro do Modelo apresentado à direita da figura 5.33 e a Exatidão do Modelo apresentado à esquerda da figura 5.33.

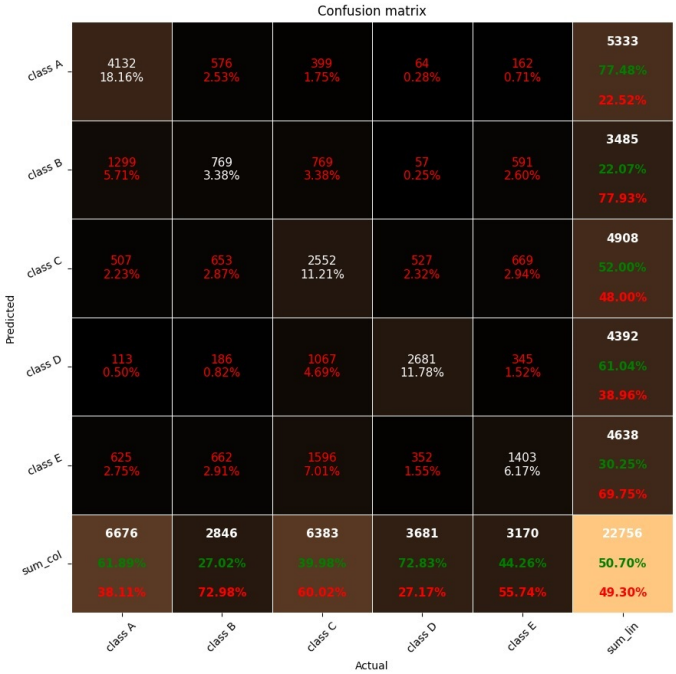


Figura 5.32: Matriz de Confusão relativamente ao modelo Lenet-4 para o conjunto de dados de uma dimensão.

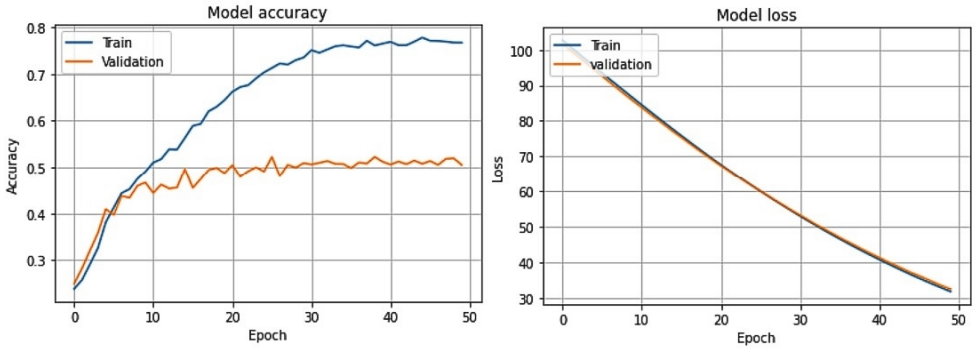


Figura 5.33: Exatidão do Modelo e Erro do Modelo relativos ao modelo Lenet-4 para o conjunto de dados de uma dimensão.

No separador denominado por "Results" ao pressionar "See all the results in a plot" o resultado obtido é apresentado na Figura 5.34. À esquerda apresenta-se o resultado referente ao "See all the results in a plot" e do lado direito o apresenta-se o resultado referente ao "See each result in a plot" do Acc.

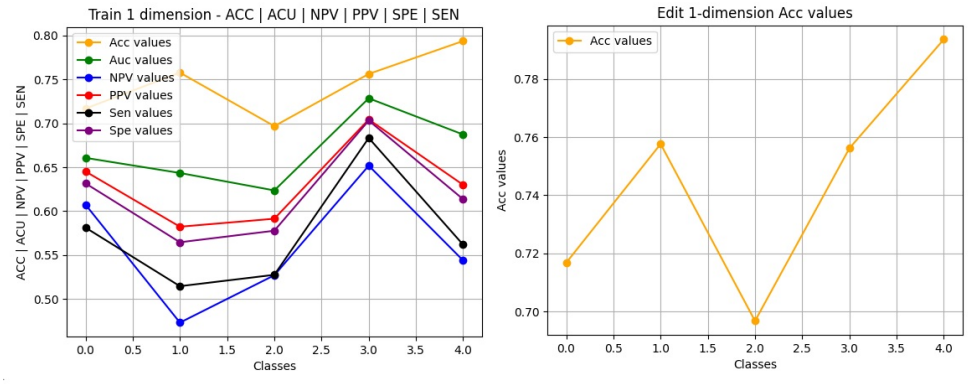


Figura 5.34: Resultado "See all the results in a plot" e "See each result in a plot" relativos ao modelo Lenet-4 para o conjunto de dados de uma dimensão.

5.4.3 The Boston Housing (preço das casas de Boston)

LeNet-5 (preço das casas de Boston):

The Boston Housing Dataset (preço das casas de Boston): os parâmetros utilizados para treinar o modelo LeNet-5 são apresentados na figura 5.35.

Após treinar o modelo Lenet-5 são obtidos os seguintes resultados: Erro do Modelo apresentado à direita na figura 5.36, MSE do Modelo apresentado à esquerda na figura 5.36 e a Previsão do Modelo apresentada pela figura 5.37.

A função de perda utilizada foi a MSE [98].

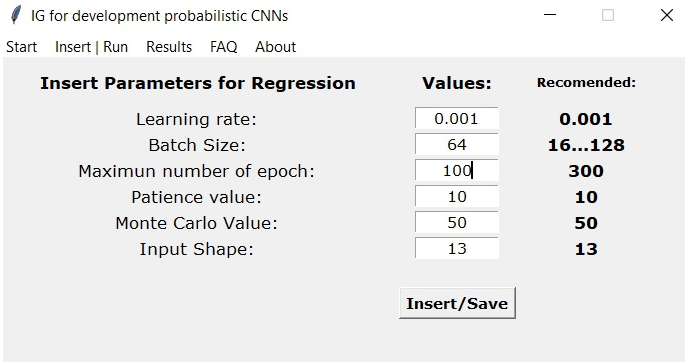


Figura 5.35: Parâmetros utilizados para o modelo LeNet-5 para regressão (Boston).

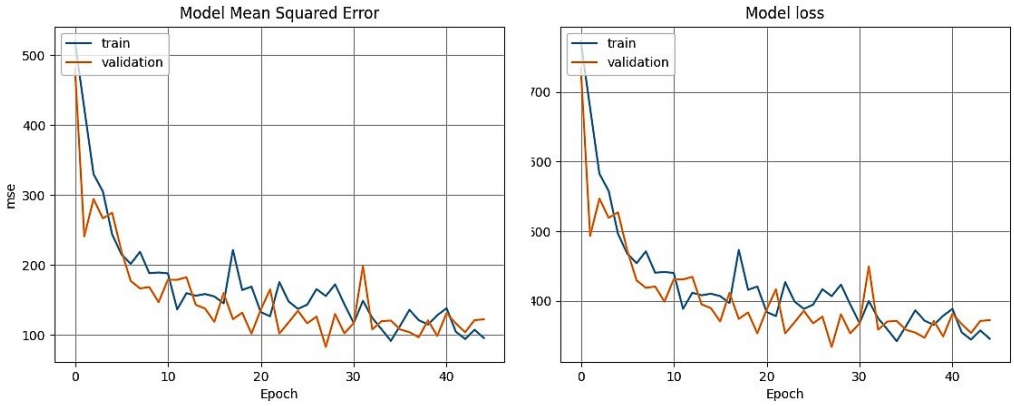


Figura 5.36: MSE do Modelo e Erro do Modelo relativos ao modelo LeNet-5 para regressão (Boston).

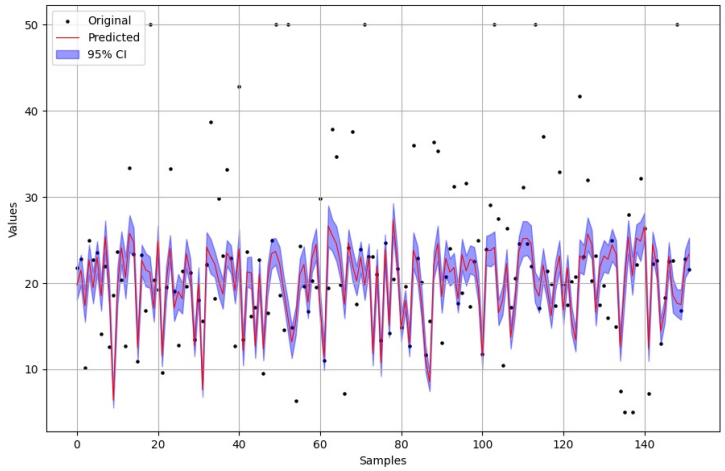


Figura 5.37: Resultado da Previsão do Modelo utilizando LeNet-5 (Boston).

LeNet-4 (regressão):

Relativamente ao treino do modelo, foi desenvolvido o modelo com a arquitetura do LeNet-4. O desenvolvimento do modelo é apresentado na figura 5.17. Os parâmetros utilizados para treinar o modelo LeNet-4 são os mesmo utilizados anteriormente apresentados na figura 5.35. Após treinar o modelo LeNet-4 são obtidos os seguintes resultados: Sumário do Modelo, apresentado pela figura 5.38, Erro do Modelo apresentado à direita da figura 5.39, MSE do Modelo apresentado à esquerda da figura 5.39 e o resultado da Previsão do Modelo apresentado pela figura 5.40.

Layer (type)	Output Shape	Param #
conv2d_flipout (Conv2DFlipou	(None, None, None, 4)	204
max_pooling2d (MaxPooling2D)	(None, None, None, 4)	0
conv2d_flipout_1 (Conv2DFlip	(None, None, None, 16)	3216
max_pooling2d_1 (MaxPooling2	(None, None, None, 16)	0
flatten (Flatten)	(None, None)	0
dense_flipout (DenseFlipout)	(None, 120)	15480
dense_flipout_1 (DenseFlipou	(None, 10)	2410

Total params: 21,310
Trainable params: 21,310
Non-trainable params: 0

Figura 5.38: Sumário do Modelo para o treino do modelo, relativamente ao conjunto de dados para regressão (Boston).

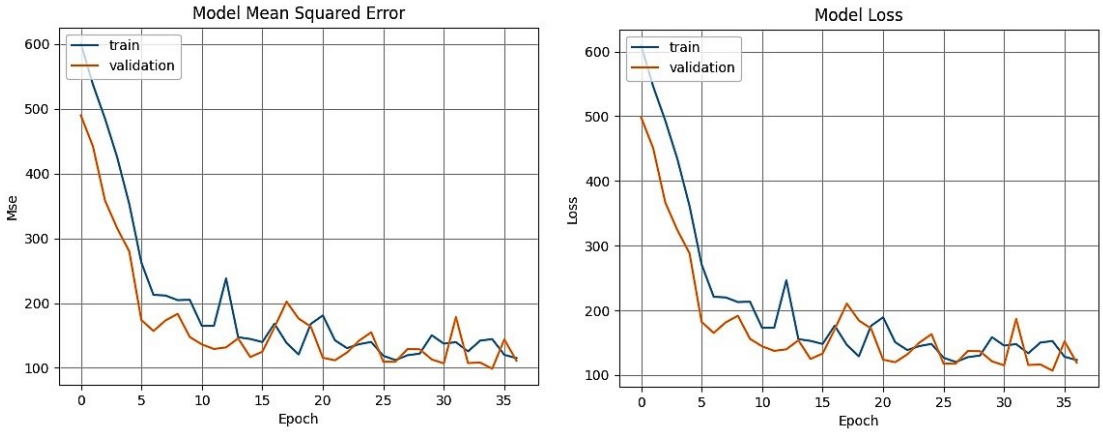


Figura 5.39: MSE do Modelo e Erro do Modelo relativos ao modelo LeNet-4 para regressão (Boston).

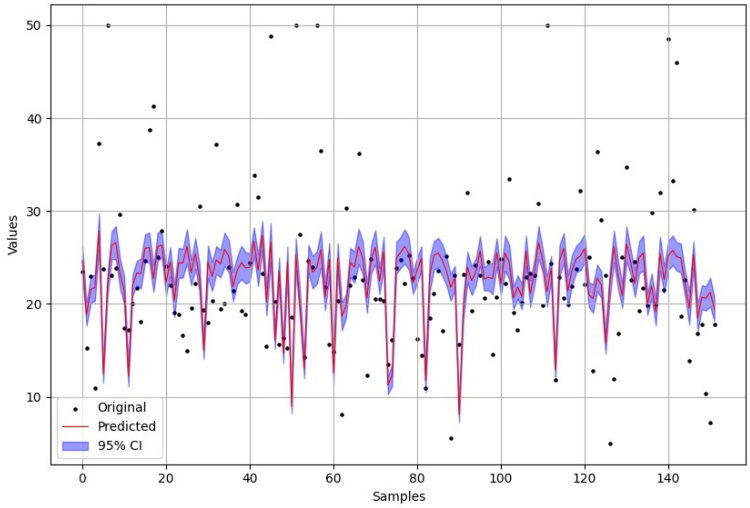


Figura 5.40: Resultado da regressão utilizando o modelo LeNet-4 (Boston).

5.4.4 Wind Speed Prediction (velocidade do vento)

LeNet-5 (regressão):

Wind Speed Prediction Dataset (velocidade do vento): os parâmetros utilizados para treinar o modelo LeNet-5 são apresentados na figura 5.36, à exceção do número máximo de épocas que é igual a dez. Após treinar o modelo Lenet-5 são obtidos os seguintes resultados: Erro do Modelo apresentado à direita da figura 5.41, MSE do Modelo apresentado à esquerda da figura 5.41 e o resultado da Previsão do Modelo apresentada pela figura 5.42. A função de perda utilizada foi a MSE.

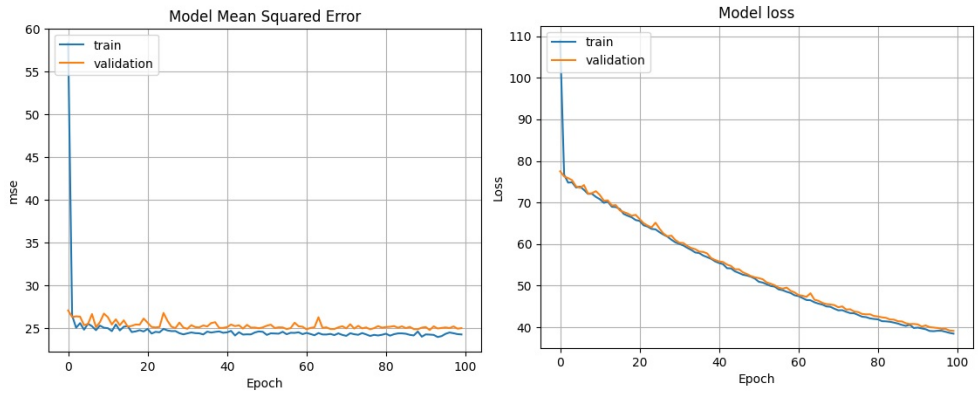


Figura 5.41: MSE do Modelo e Erro do Modelo relativos ao modelo LeNet-5 para regressão (Wind).

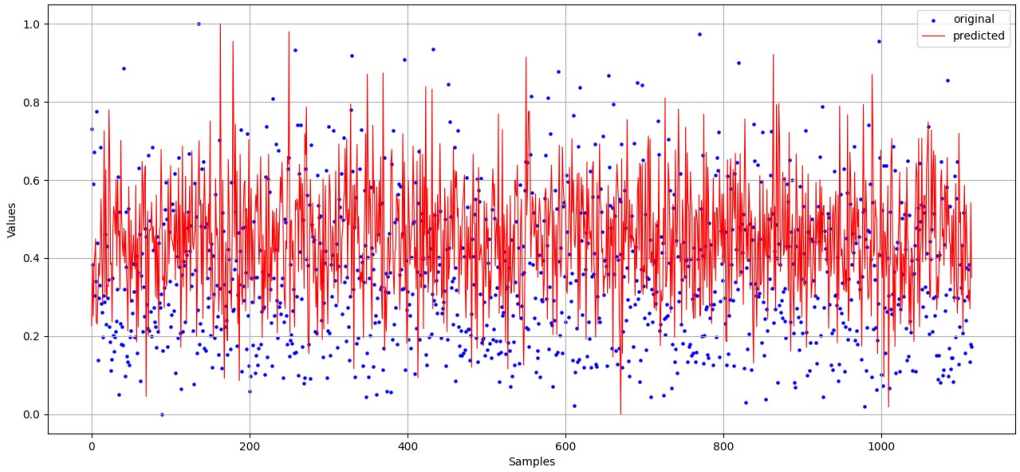


Figura 5.42: Resultado da Previsão do Modelo LeNet-5 (Wind).

5.5 Comparação do Modelo Determinístico e do Modelo Probabilístico

Para comparar os MDs e os MPs treinaram-se modelos com os conjuntos de dados para uma dimensão (ISRU Sleep Dataset) e duas dimensões (MNIST Dataset).

Para o conjunto de dados de uma dimensão, ISRU Sleep Dataset, utilizaram-se os parâmetros apresentados anteriormente pela figura 5.22. Na figura 5.43 é apresentada a Matriz de Confusão para o MD e na figura 5.44 é apresentada a Matriz de Confusão para o MP.

Para o conjunto de dados de duas dimensões (MNIST Dataset), utilizaram-se os parâmetros apresentados anteriormente pela figura 5.5. Na figura 5.45 é apresentada a Matriz de Confusão para o MD e na figura 5.46 é apresentada a Matriz de Confusão para o MP.

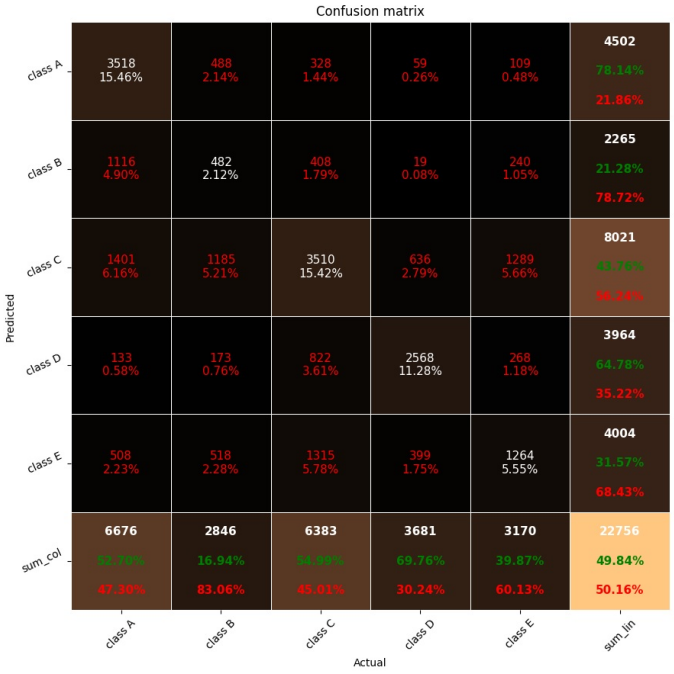


Figura 5.43: MD — Matriz de Confusão (ISRU).

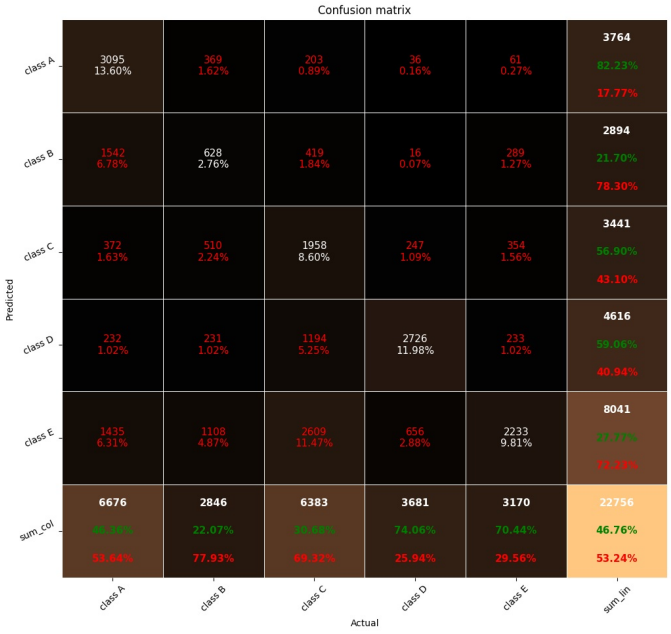


Figura 5.44: MP — Matriz de Confusão (ISRU).

Figura 5.45: MD — Matriz de Confusão (MNIST).

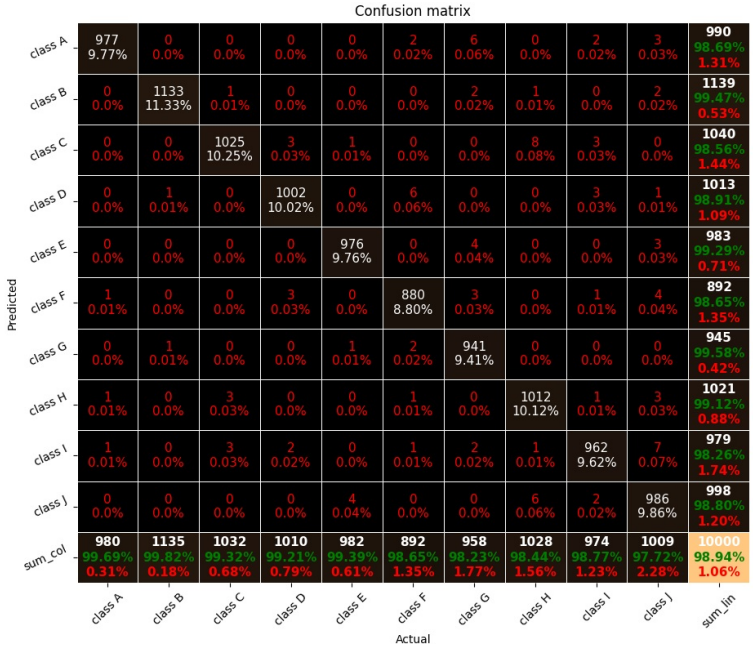


Figura 5.46: MP — Matriz de Confusão (MNIST).

À esquerda da figura 5.47 são apresentados os resultados para o conjunto de dados de uma dimensão (ISRU): Acc, AUC, NPV, PPV, Spe e Sen para o MD e do lado direito da figura 5.47 são apresentados os mesmos resultados relativamente ao MP. À esquerda da figura 5.48 são apresentados os resultados para o conjunto de dados de duas dimensões (MNIST): Acc, AUC, NPV, PPV, Spe e Sen para o MD e do lado direito da figura 5.48 são apresentados os mesmos resultados relativamente ao MP.

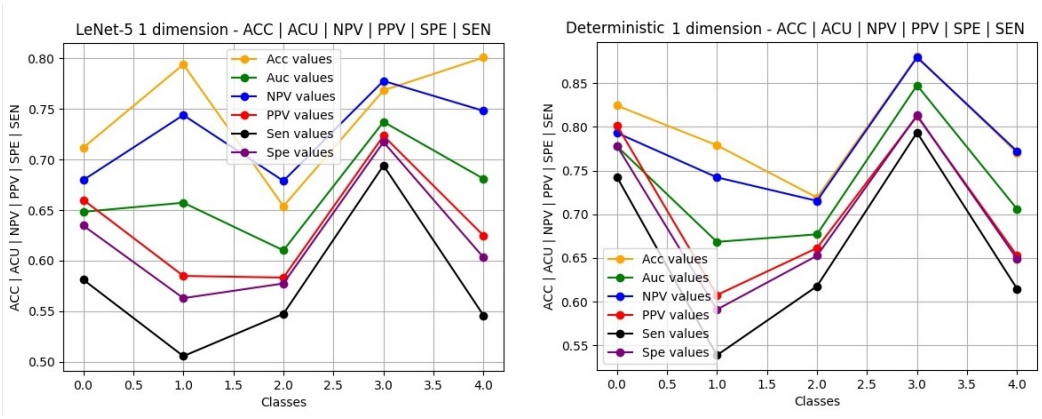


Figura 5.47: ISRU — Resultados do MD | Resultados do MP.

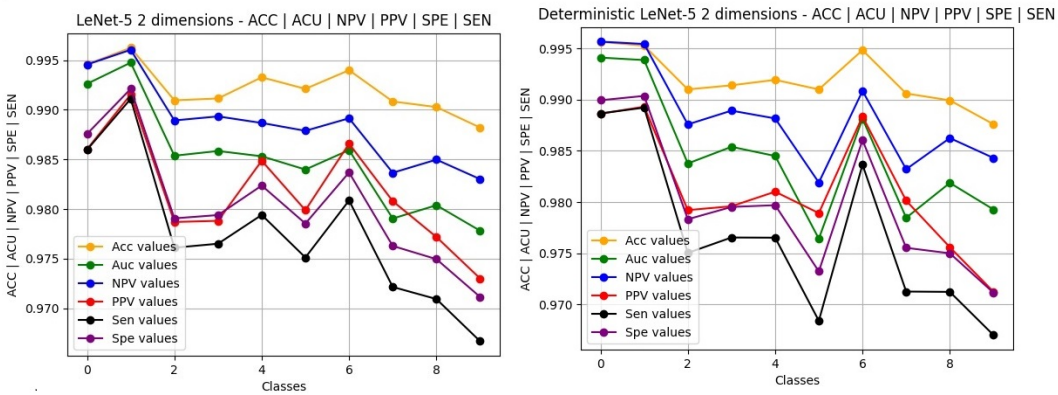


Figura 5.48: MNIST — Resultados do MD | Resultados do MP.

Relativamente aos resultados apresentados para comparação entre os MDs e os MPs, verifica-se que os resultados dos MDs são equivalentes aos dos MPs. Não se esperava necessariamente que os resultados dos MPs fossem melhores que os dos MDs, contudo os MPs dão informação associada à Incerteza Epistémica do modelo e tal como se verifica anteriormente este resultado é uma mais-valia. A comparação entre os MDs e os MPs e os exemplos fornecidos dos MPs têm por objetivo mostrar o que são e como funcionam os MPs.

5.6 Pontos-chave

Neste capítulo apresentam-se os recursos de *hardware* utilizados, sendo neste caso um computador pessoal (*desktop*) e um servidor da Universidade da Madeira e apresentam-se as caraterísticas técnicas dos equipamentos. Posteriormente apresentam-se os quatro conjuntos de dados para a realização de testes, dois para classificação e dois de regressão.

Relativamente aos resultados, para classificação são apresentados os parâmetros utilizados, o modelo utilizado para o treino, a Matriz de Confusão, o Sumário do Modelo, a Exatidão do Modelo, o Erro do Modelo, a Acc, a AUC, o NPV, o PPV, a Spe, a Sen e a Incerteza Epistémica.

Para regressão são apresentados os parâmetros utilizados, o modelo utilizado para o treino, o MSE do Modelo, o Erro do Modelo, Sumário do Modelo e o resultado da Previsão do Modelo.

Através dos resultados da Incerteza Epistémica fornecidos apenas pelos MPs é possível verificar-se que: para resultados elevados da Incerteza Epistémica é possível que o modelo tenha classificado a amostra incorretamente, para resultados baixos da Incerteza Epistémica é provável que a amostra tenha sido classificada corretamente pelo modelo.

Apenas os MPs dão os resultados da Incerteza Epistémica, através MDs não é possível obter-se a Incerteza Epistémica.

Relativamente à comparação dos resultados dos MDs e dos MPs é possível verificar-se através da matriz de confusão que para o conjunto de dados ISRU Sleep Dataset os resultados são um pouco melhores para os MDs, face aos MPs; relativamente ao conjunto de dados MNIST Dataset os resultados são idênticos.

Capítulo 6

Conclusões e Trabalho Futuro

6.1 Conclusão

Através da IG criada, é possível construir modelos probabilísticos. A título de exemplo foram criados dois modelos de classificação e dois de regressão. Relativamente aos modelos de classificação foram treinados um para o conjunto de dados de uma dimensão relativos aos vários estados do sono e o outro para o conjunto de dados de duas dimensões. Relativamente aos modelos de regressão foram treinados para o conjunto de dados relativos aos preços de casas de Boston e para o conjunto de dados relativos à velocidade do vento. Todos os quatro modelos foram treinados com sucesso para a arquitetura LeNet-5, Alexnet e para uma arquitetura criada pelo utilizador, camada a camada e os correspondentes resultados apresentados.

Tendo em conta os objetivos para o trabalho realizado foram propostas duas questões de investigação:

Relativamente à primeira questão: É possível desenvolver modelos de DL probabilística sem ser necessário ter um conhecimento específico e aprofundado no domínio do DL com modelos probabilísticos? Observou-se que sim, é possível, uma vez que para utilizarmos a IG desenvolvida não é necessário ter um conhecimento específico e aprofundado no domínio do DL e através da IG, foi possível desenvolver os modelos de DL probabilística.

Relativamente à segunda questão: É possível implementar uma IG que vá ao encontro das necessidades dos utilizadores que pretendem desenvolver CNNs probabilísticas de forma simples e direta? Observou-se que sim, é possível, uma vez que a IG desenvolvida vai ao encontro das necessidades dos utilizadores que pretendem desenvolver CNNs probabilísticas de forma simples e direta.

Apenas os MPs dão informação associada à Incerteza Epistémica do modelo, embora os resultados dos MPs não sejam necessariamente melhores do que os MDs. A comparação entre: MDs e os MPs e os exemplos fornecidos dos MPs têm por objetivo mostrar o que são e como funcionam os MPs.

6.2 Limitações do Trabalho

A IG está disponível para conjuntos de dados de uma e duas dimensões, não estando assim disponível para conjuntos de dados de três dimensões, quatro dimensões e por aí diante.

A IG apenas considera distribuições gaussianas, não foram consideradas outras distribuições tais como: distribuição de Bernoulli, distribuição binomial entre outras [99].

A criação de modelos com uma arquitetura criada pelo utilizador, camada a camada, poderia ser efetuado de forma mais intuitiva.

6.3 Trabalho Futuro

Os próximos passos para aperfeiçoamento da aplicação desenvolvida requerem a resolução das limitações apresentadas e o estudo da viabilidade do desenvolvimento de uma versão web para a aplicação.

Também seria importante disponibilizar ao utilizador uma maior variedade de arquiteturas já implementadas na IG, tais como:

- VGG-16 [100], criada pelo Visual Geometry Group, contem 13 camadas *convolution* e 3 camadas *dense*. Para aumentar o desempenho, tornou-se a RNA mais profunda através da adição de camadas comparativamente à AlexNet.

- Inception-V1 [101] possui 22 camadas e cinco milhões de parâmetros, é possível processar vários *kernels* de diferentes tamanhos no mesmo *input map*.

- Inception-v3 [102] sucessor do Inception-v1, possui 42 camadas e 24 milhões de parâmetros, foram realizados ajustes no otimizador, adição de normalização às camadas auxiliares, reduziram-se as dimensões de entrada da próxima camada e obtiveram-se cálculos mais eficientes.

- ResNet-50 [103], até ao momento, as novas CNNs apenas aumentavam o número de camadas. Todavia, a Microsoft Research seguiu uma abordagem diferente e criou o ResNet que utiliza conexões de salto (também conhecidas como conexões de atalho).

- Xception [104] é uma adaptação do Inception, os módulos de Inception são substituídos por convoluções separáveis em profundidade. Foram adicionadas conexões residuais, melhorou-se desta forma a velocidade de treino. Utiliza a técnica *Depthwise Separable Convolution*, esta técnica consiste na utilização de diferentes filtros para cada canal da imagem, ou seja, ao invés de utilizar um único filtro de três dimensões, utiliza um filtro para cada canal e em seguida aplica uma convolução padrão com um filtro de tamanho 1×1.

- Inception-v4 [105], sucessor do Inception-v3. A principal diferença é no módulo *Stem*, o módulo *Stem* é referente às operações iniciais anteriores à introdução dos blocos do Inception, no módulo Inception-C, foram adicionadas conexões residuais resultando assim de uma redução do tempo necessário para treinar o modelo.

- Inception-ResNet-V2 [106], utiliza duas técnicas, Inception e Residual Network, é constituída por: três módulos Inception e três módulos residuais, o primeiro módulo residual é repetido 10 vezes, o segundo módulo residual é repetido 20 vezes e o terceiro módulo residual é repetido 10 vezes.

Bibliografia

[1] P. Hamet and J. Tremblay, "Artificial intelligence in medicine," *Metabolism*, vol. 69, pp. S36–S40, 2017.

[2] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborova, "Machine learning and the physical sciences," *Reviews of Modern Physics*, vol. 91, no. 4, p. 045002, 2019.

[3] R. Armand, "Deep learning with data science experience." [Online] = https://medium.com/@armand$_ruiz/deep-learning-with-data-science-experience-8478cc0f81ac$.

[4] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[5] S. Ravindra, "How convolutional neural networks accomplish image recognition," 2017. [Online] = https://medium.com/@savaramravindra4/how-convolutional-neural-networks-accomplish-image-recognition-277033b72436, Acedido: 15 de Abril de 2022.

[6] J. V. R. Falcão, V. Moreira, F. Santos, and C. Ramos, "Redes neurais deep learning com tensorflow," 2019. [Online] = https://revistas.unifenas.br/index.php/RE3C/article/view/232, Acedido: 12 de Maio de 2022.

[7] J. Moolayil, "An introduction to deep learning and keras," in *Learn Keras for Deep Neural Networks*, pp. 1–16, Springer, 2019.

[8] K. Shridhar, F. Laumann, and M. Liwicki, "A comprehensive guide to bayesian convolutional neural network with variational inference," *arXiv preprint arXiv:1901.02731*, 2019.

[9] O. Durr, B. Sick, and E. Murina, *Probabilistic deep learning: With python, keras and tensorflow probability*. Manning Publications, 2020.

[10] E. Bisong, "Python," in *Building Machine Learning and Deep Learning Models on Google Cloud Platform*, pp. 71–89, Springer, 2019.

[11] D. Beniz, A. Espindola, *et al.*, "Using tkinter of python to create graphical user interface (gui) for scripts in lnls," *WEPOPRPO25*, vol. 9, pp. 25–28, 2016.

[12] I. Mocanu *et al.*, "An introduction to cuda programming," *Journal of Information Systems & Operations Management*, vol. 2, no. 2, pp. 495–506, 2008.

[13] M. Jorda, P. Valero-Lara, and A. J. Pena, "Performance evaluation of cudnn convolution algorithms on nvidia volta gpus," *IEEE Access*, vol. 7, pp. 70461–70473, 2019.

[14] D. M. Etter, D. C. Kuncicky, and D. W. Hull, *Introduction to MATLAB*. Prentice Hall Hoboken, NJ, USA, 2002.

[15] A. Pomares, J. L. Martínez, A. Mandow, M. A. Martínez, M. Morán, and J. Morales, "Ground extraction from 3d lidar point clouds with the classification learner app," in *2018 26th Mediterranean Conference on Control and Automation (MED)*, pp. 1–9, IEEE, 2018.

[16] J. Demsar, T. Curk, A. Erjavec, C. Gorup, T. Hocevar, M. Milutinovic, M. Movina, M. Polajnar, M. Toplak, A. Staric, *et al.*, "Orange: data mining toolbox in python," *the Journal of machine Learning research*, vol. 14, no. 1, pp. 2349–2353, 2013.

[17] Avcontentteam, "Building machine learning model." [Online] = https://www.analyticsvidhya.com/blog/2017/09/building-machine-learning-model-fun-using-orange/, Acedido: 20 de Maio de 2022.

[18] M. Mardalius, "Pemanfaatan rapid miner studio 8.2 untuk pengelompokan data penjualan aksesoris menggunakan algoritma k-means," *JURTEKSI (Jurnal Teknologi dan Sistem Informasi)*, vol. 4, no. 2, pp. 123–132, 2018.

[19] R. M. Comunity, "Forum rapid miner." [Online] = https://community.rapidminer.com/, Acedido: 26 de Maio de 2022.

[20] V. L. Uskov, J. P. Bakken, P. Putta, D. Krishnakumar, and K. S. Ganapathi, "Smart education: predictive analytics of student academic performance using machine learning models in weka and dataiku systems," in *Smart Education and e-Learning 2021*, pp. 3–17, Springer, 2021.

[21] D. Knoledge, "Hands-on: Create the model." [Online] = https://knowledge.dataiku.com/latest/courses/machine-learning/create-model/create-model.html, Acedido: 27 de Maio de 2022.

[22] A. Fillbrunn, C. Dietz, J. Pfeuffer, R. Rahn, G. A. Landrum, and M. R. Berthold, "Knime for reproducible cross-domain analysis of life science data," *Journal of biotechnology*, vol. 261, pp. 149–156, 2017.

[23] C. L. Devasena, T. Sumathi, V. Gomathi, and M. Hemalatha, "Effectiveness evaluation of rule based classifiers for the classification of iris data set," *Bonfring International Journal of Man Machine Interface*, vol. 1, no. Special Issue Inaugural Special Issue, pp. 05–09, 2011.

[24] L. Fleck, M. H. F. Tavares, E. Eyng, A. C. Helmann, and M. d. M. Andrade, "Redes neurais artificiais: Princípios básicos," *Revista Electronica Cientifica Inovação e Tecnologia*, vol. 1, no. 13, pp. 47–57, 2016.

[25] J. M. Stibel, *Conectado pelas Ideias: como o cérebro está moldando o futuro da internet*. DVS Editora, 2013.

[26] A. C. Neves, I. González, J. Leander, and R. Karoumi, "A new approach to damage detection in bridges using machine learning," in *International Conference on Experimental Vibration Analysis for Civil Engineering Structures*, pp. 73–84, Springer, 2017.

[27] J. A. Adamu, "Superintelligent digital brains: distinct activation functions implying distinct artificial neurons," in *Emerging Topics in Artificial Intelligence 2020*, vol. 11469, p. 114691L, SPIE, 2020.

[28] S. Razavi and B. A. Tolson, "A new formulation for feedforward neural networks," *IEEE Transactions on neural networks*, vol. 22, no. 10, pp. 1588–1598, 2011.

[29] I. V. Rizzo and R. L. C. Canato, "Inteligência artificial: funções de ativação," *Prospectus (ISSN: 2674-8576)*, vol. 2, no. 2, 2020.

[30] F. Mendonça, F. Morgado-Dias, and F. Diogo, "Formação de inteligência artificial: Uma introdução à aprendizagem automática," 2022.

[31] X. Yu, M. O. Efe, and O. Kaynak, "A general backpropagation algorithm for feedforward neural networks learning," *IEEE transactions on neural networks*, vol. 13, no. 1, pp. 251–254, 2002.

[32] S. I. Gallant and S. I. Gallant, *Neural network learning and expert systems*. MIT press, 1993.

[33] D. H. B. Binoti, M. L. M. da Silva Binoti, and H. G. Leite, "Configurão de redes neurais artificiais para estimação do volume de árvores," *Revista Ciência da Madeira (Brazilian Journal of Wood Science)*, vol. 5, no. 1, pp. 10–12953, 2014.

[34] M. Roodschild, J. Gotay Sardiñas, and A. Will, "A new approach for the vanishing gradient problem on sigmoid activation," *Progress in Artificial Intelligence*, vol. 9, no. 4, pp. 351–360, 2020.

[35] B. Hanin, "Universal function approximation by deep neural nets with bounded width and relu activations," *Mathematics*, vol. 7, no. 10, p. 992, 2019.

[36] P. E. Canhanga, "Modelo matemático para previsãoo de vendas: Regresão linear simples," *Cadernos do IME-Série Matemática*, no. 14, pp. 71–81, 2020.

[37] M. Jiang, Y. Liang, X. Feng, X. Fan, Z. Pei, Y. Xue, and R. Guan, "Text classification based on deep belief network and softmax regression," *Neural Computing and Applications*, vol. 29, no. 1, pp. 61–70, 2018.

[38] Z. Zhang, "Improved adam optimizer for deep neural networks," in *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*, pp. 1–2, IEEE, 2018.

[39] T. Xiang, J. Wang, and X. Liao, "An improved particle swarm optimizer with momentum," in *2007 IEEE Congress on Evolutionary Computation*, pp. 3341–3345, IEEE, 2007.

[40] Y. Yu, L. Zhang, L. Chen, and Z. Qin, "Adversarial samples generation based on rmsprop," in *2021 IEEE 6th International Conference on Signal and Image Processing (ICSIP)*, pp. 1134–1138, IEEE, 2021.

[41] P. Mahajan, P. Abrol, P. K. Lehana, *et al.*, "Scene based classification of aerial images using convolution neural networks," *Journal of Scientific and Industrial Research (JSIR)*, vol. 79, no. 12, pp. 1087–1094, 2020.

[42] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights into imaging*, vol. 9, no. 4, pp. 611–629, 2018.

[43] J. Heyman, "Different pooling layers for CNN." [Online] = https://jacobheyman702.medium.com/different-pooling-layers-for-cnn-4652a5103d62, Acedido: 27 de Abril de 2022.

[44] G. Alves, "Understanding convnets CNN." [Online] = https://medium.com/neuronio/understanding-convnets-cnn-712f2afe4dd3, Acedido: 26 de Abril de 2022.

[45] Y. Liu, J. A. Starzyk, and Z. Zhu, "Optimized approximation algorithm in neural networks without overfitting," *IEEE transactions on neural networks*, vol. 19, no. 6, pp. 983–995, 2008.

[46] K. Lambers *et al.*, "Learning to look at lidar: The use of r-cnn in the automated detection of archaeological objects in lidar data from the netherlands," *Journal of Computer Applications in Archaeology*, vol. 2, no. 1, pp. 31–40, 2019.

[47] R. Karim, "Illustrated: 10 cnn architectures." [Online] url = https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d, Acedido: 10 de Março de 2022.

[48] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *Journal of the American statistical Association*, vol. 112, no. 518, pp. 859–877, 2017.

[49] T. Van Erven and P. Harremos, "Renyi divergence and kullback-leibler divergence," *IEEE Transactions on Information Theory*, vol. 60, no. 7, pp. 3797–3820, 2014.

[50] R. B. Queiroz, A. G. Rodrigues, and A. T. Gomez, "Estudo comparativo entre as técnicas máxima verossimilhança gaussiana e redes neurais na classificação de imagens ir-mss cbers 1," *I WorkComp Sul*, vol. 1, 2004.

[51] Y. Wen, P. Vicol, J. Ba, D. Tran, and R. Grosse, "Flipout: Efficient pseudo-independent weight perturbations on mini-batches," *arXiv preprint arXiv:1803.04386*, 2018.

[52] D. P. Kroese and R. Y. Rubinstein, "Monte carlo methods," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 4, no. 1, pp. 48–58, 2012.

[53] K. Shridhar, F. Laumann, and M. Liwicki, "Uncertainty estimations by softplus normalization in bayesian convolutional neural networks with variational inference," *arXiv preprint arXiv:1806.05978*, 2018.

[54] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing personalized markov chains for next-basket recommendation," in *Proceedings of the 19th international conference on World wide web*, pp. 811–820, 2010.

[55] G. Gunapati, A. Jain, P. Srijith, and S. Desai, "Variational inference as an alternative to mcmc for parameter estimation and model selection," *Publications of the Astronomical Society of Australia*, vol. 39, 2022.

[56] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and variational inference in deep latent gaussian models," in *International conference on machine learning*, vol. 2, p. 2, Citeseer, 2014.

[57] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural network," in *International conference on machine learning*, pp. 1613–1622, PMLR, 2015.

[58] R. S. de Espindola, A. Majdenbaum, and J. L. N. Audy, "Uma análise crítica dos desafios para engenharia de requisitos em manutenção de software.," in *WER*, pp. 226–238, 2004.

[59] P. Campos, "Engenharia de requisitos." [Online] = http://cee.uma.pt/edu/er/slides/ER-Intro.pdf, Acedido: 18 de Maio de 2022.

[60] D. Riehle, "The economic motivation of open source software: Stakeholder perspectives," *Computer*, vol. 40, no. 04, pp. 25–32, 2007.

[61] J. Eveleens and C. Verhoef, "The rise and fall of the chaos report figures," *IEEE software*, vol. 27, no. 1, pp. 30–36, 2009.

[62] H. Al-Samarraie and S. Hurmuzan, "A review of brainstorming techniques in higher education," *Thinking Skills and Creativity*, vol. 27, pp. 78–91, 2018.

[63] M. Loureiro, "Brainstorming na educação." [Online] = https://ensinotec.com/brainstorming-na-educacao/, Acedido: 2 de Maio de 2022.

[64] R. A. R. de Mendonça, "Levantamento de requisitos no desenvolvimento ágil de software," *Semana da Ciência e Tecnologia da PUC Goias*, p. 12, 2014.

[65] V. F. A. Santander, A. A. Vicente, F. G. Koerich, and J. B. de Castro, "Modelagem de requisitos organizacionais, nao - funcionais e funcionais em software legado com enfase na tecnica i*.," in *CIbSE*, pp. 47–60, 2007.

[66] F. P. Rodrigues, "Interface para acessibilidade de alunos cegos na construcaoo de um diagrama de entidade relacionamento (der) em banco de dados: Modelagem conceitual," *Revista Cientifica UMC*, vol. 3, no. 3, 2018.

[67] A. G. Peixoto, "O uso de metodologias ativas como ferramenta de potencializacao da aprendizagem de diagramas de caso de uso," *Outras Palavras*, vol. 12, no. 2, 2016.

[68] E. Figueiredo, "Requisitos funcionais e requisitos não funcionais," *Icex, Dcc/Ufmg*, 2011.

[69] L. Bass, P. Clements, and R. Kazman, *Software architecture in practice*. Addison-Wesley Professional, 2003.

[70] M. R. Barbacci, R. Ellison, A. J. Lattanze, J. A. Stafford, and C. B. Weinstock, "Quality attribute workshops (qaws)," tech. rep., Carnegie-Mello university pittesburgh pa software engineering institute, 2003.

[71] A. Chaves, G. André, P. João, R. Sérgio, and R. Nuno, "Projeto de arquitetura de software – grupo 3 universidade da madeira 2020/2021 group-finder," 2021.

[72] T. S. Sammi LaBello, "Why you need a data flow diagram." [Online] = https://pratum.com/blog/512-why-you-need-a-data-flow-diagram, Acedido: 22 de Maio de 2022.

[73] A. Oliveira, "Criar um modelo entidade-relacionamento." [Online] = https://brainly.com.br/tarefa/49316533, Acedido: 10 de Maio de 2022.

[74] C. E. Vazquez and G. S. Simões, *Engenharia de Requisitos: software orientado ao negócio*. Brasport, 2016.

[75] M. A. Amaral, K. A. Oliveira, and V. F. Bartholo, "Uma experiência para definição de storyboard em metodologia de desenvolvimento colaborativo de objetos de aprendizagem," *Ciências & Cognição*, vol. 15, no. 1, pp. 19–32, 2010.

[76] G. André, C. Aníbal, and A. Pedro, "Projeto da unidade curricular sistemas multimédia – aventuras do tazz e egg desenvolvido em universidade da madeira 2020/2021 group-finder," 2021.

[77] W. R. Bischofberger and G. Pomberger, *Prototyping-oriented software development: Concepts and tools*. Springer Science & Business Media, 2012.

[78] S. W. Ambler, "Tailoring usability into agile software development projects," in *Maturing Usability*, pp. 75–95, Springer, 2008.

[79] A. BARROS, "Avaliação da usabilidade do portal conecta apoiado pelas 10 heurísticas propostas por jakob nielsen," 2017.

[80] J. R. Lewis, "Usability testing," *Handbook of human factors and ergonomics*, vol. 12, p. e30, 2006.

[81] G. Gutierrez-Carreon, T. Daradoumis, and J. Jorba, "Integrating learning services in the cloud: An approach that benefits both systems and learning," *Journal of Educational Technology & Society*, vol. 18, no. 1, pp. 145–157, 2015.

[82] A. D. Moore, *Python GUI Programming with Tkinter: Develop responsive and powerful GUI applications with Tkinter*. Packt Publishing Ltd, 2018.

[83] S. Van Der Walt, S. C. Colbert, and G. Varoquaux, "The numpy array: a structure for efficient numerical computation," *Computing in science & engineering*, vol. 13, no. 2, pp. 22–30, 2011.

[84] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in science & engineering*, vol. 9, no. 03, pp. 90–95, 2007.

[85] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[86] R. J. F. d. A. Pereira, *Processo de desenho de um prototipo de alta fidelidade, em IOS-ROOMS*. PhD thesis, 2014.

[87] A. Kadiyala and A. Kumar, "Applications of python to evaluate environmental data science problems," *Environmental Progress & Sustainable Energy*, vol. 36, no. 6, pp. 1580–1586, 2017.

[88] C. C. Davila, C. F. Reinhart, and J. L. Bemis, "Modeling boston: A workflow for the efficient generation and maintenance of urban building energy models from existing geospatial datasets," *Energy*, vol. 117, pp. 237–250, 2016.

[89] K. Trebing and S. Mehrkanoon, "Wind speed prediction using multidimensional convolutional neural networks," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 713–720, IEEE, 2020.

[90] S. Khalighi, T. Sousa, J. M. Santos, and U. Nunes, "Isruc-sleep: A comprehensive public dataset for sleep researchers," *Computer methods and programs in biomedicine*, vol. 124, pp. 180–192, 2016.

[91] A. Baldominos, Y. Saez, and P. Isasi, "A survey of handwritten character recognition with mnist and emnist," *Applied Sciences*, vol. 9, no. 15, p. 3169, 2019.

[92] K. Kwon, S. Kwon, and W.-H. Yeo, "Automatic and accurate sleep stage classification via a convolutional deep neural network and nanomembrane electrodes," *Biosensors*, vol. 12, no. 3, p. 155, 2022.

[93] A. Baratloo, M. Hosseini, A. Negida, and G. El Ashal, "Part 1: simple definition and calculation of accuracy, sensitivity and specificity," 2015.

[94] A. G. Lalkhen and A. McCluskey, "Clinical tests: sensitivity and specificity," *Continuing education in anaesthesia critical care & pain*, vol. 8, no. 6, pp. 221–223, 2008.

[95] B. Dai, O. Nachum, Y. Chow, L. Li, C. Szepesvári, and D. Schuurmans, "Coindice: Off-policy confidence interval estimation," *Advances in neural information processing systems*, vol. 33, pp. 9398–9411, 2020.

[96] Z. Zhang and M. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels," *Advances in neural information processing systems*, vol. 31, 2018.

[97] Y. LeCun, L. Jackel, L. Bottou, A. Brunot, C. Cortes, J. Denker, H. Drucker, I. Guyon, U. Muller, E. Sackinger, *et al.*, "Comparison of learning algorithms for handwritten digit recognition," in *International conference on artificial neural networks*, vol. 60, pp. 53–60, Perth, Australia, 1995.

[98] O. M. Kvalheim, R. Arneberg, B. Grung, and T. Rajalahti, "Determination of optimum number of components in partial least squares regression from distributions of the root-mean-squared error obtained by monte carlo resampling," *Journal of Chemometrics*, vol. 32, no. 4, p. e2993, 2018.

[99] K. H. Cho, T. Raiko, and A. Ilin, "Gaussian-bernoulli deep boltzmann machine," in *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–7, IEEE, 2013.

[100] S. Tammina, "Transfer learning using vgg-16 with deep convolutional neural network for classifying images," *International Journal of Scientific and Research Publications (IJSRP)*, vol. 9, no. 10, pp. 143–150, 2019.

[101] R. K. Mohapatra, K. Shaswat, and S. Kedia, "Offline handwritten signature verification using cnn inspired by inception v1 architecture," in *2019 Fifth International Conference on Image Information Processing (ICIIP)*, pp. 263–267, IEEE, 2019.

[102] C. Wang, D. Chen, L. Hao, X. Liu, Y. Zeng, J. Chen, and G. Zhang, "Pulmonary image classification based on inception-v3 transfer learning model," *IEEE Access*, vol. 7, pp. 146533–146541, 2019.

[103] L. Wen, X. Li, and L. Gao, "A transfer convolutional neural network for fault diagnosis based on resnet-50," *Neural Computing and Applications*, vol. 32, no. 10, pp. 6111–6124, 2020.

[104] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251–1258, 2017.

[105] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Thirty-first AAAI conference on artificial intelligence*, 2017.

[106] F. Baldassarre, D. G. Morin, and L. Rodess-Guirao, "Deep koalarization: Image colorization using cnns and inception-resnet-v2," *arXiv preprint arXiv:1712.03400*, 2017.