



UNIVERSIDADE DO MINHO

LABORATÓRIO DE ENGENHARIA INFORMÁTICA

Relatório Final

Previsão de efeitos secundários em fármacos usando deep learning

MESTRADO DE ENGENHARIA INFORMÁTICA

Projeto 75

Ângela Barros - PG38407

Manuel Monteiro - PG37158

Braga, Portugal

25 de Junho de 2019

Resumo

A previsão de efeitos secundários e a sua consequente prevenção é algo extremamente importante no mundo farmacêutico. Neste documento irá ser abordado o tema de efeitos secundários de compostos químicos nos medicamentos. Para tal, irão ser abordados conceitos importantes no âmbito da Inteligência Artificial, como *Machine Learning* e *Deep Learning*. São também abordados conceitos relativos às Redes Neurais Convulocionais, Featurization, Embeddings, Multitask Classifier.

Uma das possíveis soluções é a utilização de *deep learning* de forma a prever efeitos secundários. Devido à natureza do projeto, para o desenvolvimento foi utilizado o Deepchem que irá ser apresentado nos capítulos seguintes. O *Deepchem* disponibilizou vários modelos aplicados a vários datasets e, neste trabalho irá se abordar alguns dos modelos que fazem parte dos tutoriais do *Deepchem*. Também se implementou modelos de raiz mas os resultados não se relevaram melhores do que os já registados. Os melhores resultados foram alcançados através de *hyperparameters optimization* dos modelos baseados em soluções já anteriormente desenvolvidos e otimizados. Após o desenvolvimento do projeto, chegou-se à conclusão que os datasets encontravam-se *imbalanced* algo que deveríamos ter resolvido na nossa análise exploratória inicial.



Conteúdo

1	Introdução	4
1.1	Contextualização	4
1.2	Motivação e Objetivos	4
1.3	Estruturação do Documento	5
2	Enquadramento Teórico	6
2.1	Redes Neurais Artificiais	6
2.1.1	Inspiração Biológica	7
2.1.2	Tipos de Redes Neurais	7
2.1.3	Algoritmo de <i>Backpropagation</i>	10
2.1.4	Funções de Ativação	11
2.2	Outros conceitos	14
2.3	Deepchem	15
2.3.1	Featurizations	15
3	Datasets	17
3.1	<i>Sider</i>	17
3.2	<i>Offsides</i>	18
3.2.1	<i>Offsides</i> Original	18
3.2.2	<i>Offsides</i> Após Tratamento	20
3.3	<i>Sider + Offsides</i>	21
4	Tratamento Offsides	22
5	Código	24
5.1	Geração de Embeddings	24
5.2	Graph Convolutacional	24
5.3	<i>Hyperparameters</i>	25
6	Resultados	26
6.1	Métricas utilizadas	26
6.2	Resultados obtidos	26
6.2.1	Sider	26
6.2.2	Offsides	28



6.2.3	Sider+Offsides	29
7	Conclusão	31
8	Trabalho Futuro	31



Lista de Figuras

1	<i>Representação de uma rede Neuronal</i>	6
2	<i>Representação de um Neurónio Biológico</i>	7
3	<i>Representação de um Perceptron Simples</i>	8
4	<i>Rede Neuronal FeedForward MultiCamada</i>	8
5	<i>Exemplo de uma Rede Neuronal Convolutacional</i>	9
6	<i>Representação de uma Rede Neuronal Recorrente</i>	10
7	<i>Função de Ativação linear e sua Derivada</i>	11
8	<i>Função de Ativação Sigmoides</i>	12
9	<i>Função de Ativação Tangente Hiperbólica</i>	13
10	<i>Função de Ativação ReLU</i>	13
11	<i>Função de Ativação Softmax</i>	14
12	<i>Workflow do Mol2Vec</i>	15
13	<i>Colunas do dataset</i>	17
14	<i>Primeiros 5 registos do dataset</i>	17
15	<i>Contagem dos efeitos secundários presentes no dataset</i>	18
16	<i>Colunas do Offsides</i>	18
17	<i>Primeiras 10 linhas do dataset</i>	19
18	<i>Describe do dataset</i>	19
19	<i>Colunas do Offsides após tratamento</i>	20
20	<i>Soma dos efeitos secundários presentes no dataset</i>	20
21	<i>Workflow da API</i>	22
22	<i>Workflow da API</i>	23
23	<i>Código de geração de embeddings</i>	24
24	<i>Código de geração da rede neuronal convolutacional</i>	25
25	<i>Optimização de hyperparameters</i>	25
26	<i>Mol2Vec ROC-AUC</i>	27
27	<i>ROC-AUC do Graph Convolution</i>	27
28	<i>Resultados utilizando o Mol2Vec no Offsides</i>	28
29	<i>Resultados utilizando o Modelo Graph Convolution no Offsides</i>	29
30	<i>Resultados utilizando Mol2Vec no Sider+Offsides</i>	29
31	<i>Resultados utilizando o Graph Convolution no Sider+Offsides</i>	30



1 Introdução

Neste capítulo é realizada uma pequena contextualização 1.1 do problema, bem como a definição dos objetivos e a motivação intrínseca à realização deste trabalho 1.2, sendo que por último é descrito a organização do documento bem como os tópicos abordados em cada capítulo 1.3.

1.1 Contextualização

Hoje em dia o tema de *deep learning* tem estado bastante em voga e tem sido uma aposta em várias indústrias de forma a manterem-se relevantes. A indústria farmacêutica tem se revelado como uma das principais em termos de utilização de Inteligência Artificial para o desenvolvimento de novos medicamentos. O objetivo, tal como a maioria das empresas, é o aumento dos lucros e, em particular no caso das farmacêuticas, a prevenção da possível perda humana e de reputação devido a medicamentos defeituosos ou que tenham reações adversas indesejadas que ponham em risco a vida dos pacientes. Como tal, a utilização de *deep learning* é uma das possíveis soluções para este problema, ou seja, permite a criação de medicamentos cuja probabilidade de efeitos secundários indesejados seja significativamente menor.

Porém, existe uma grande correlação entre Inteligência Artificial, *Machine Learning* e *Deep Learning* e pretende-se nesta secção esclarecer melhor esses tópicos.

- **Inteligência Artificial** - é um sistema que pode perceber o ambiente que o rodeia, tomar decisões, adaptar-se a esse mesmo ambiente.
- ***Machine Learning*** - algoritmos cujo o seu principal objetivo é aprender, e cuja a performance aumenta com o passar do tempo, com a introdução de novos dados.
- ***Deep Learning*** - é um subconjunto do *machine learning*, onde redes neuronais, com múltiplas camadas aprendem, a partir de uma grande quantidade de dados.

1.2 Motivação e Objetivos

Com a realização deste projeto, a maior motivação é tentar perceber, e também desenvolver, soluções de *deep learning* que nos permitam resolver o nosso problema inicial de previsão de efeitos secundários de fármacos. O principal objetivo com este projeto é o desenvolvimento de medicamentos mais seguros, ou seja, poder garantir uma menor probabilidade de efeitos secundários por parte de medicamentos novos resultando numa maior otimização em termos de custos. O principal objetivo de uma empresa, seja ela na área farmacêutica ou não, é gerar lucros e, muito dinheiro é gasto em ensaios clínicos para garantir que os novos produtos (neste caso, medicamentos) não irão ter efeitos indesejados por parte do público-alvo. Portanto, se surgir algum problema nessa fase de testes já será numa fase bastante avançada, ou seja, fazendo com que já tenha sido gasto bastantes recursos no desenvolvimento do dito produto. Ou seja, uma boa aposta por parte das empresas farmacêuticas seria evitar o prolongamento de um processo de desenvolvimento de um novo medicamento que à partida fosse considerado muito provável de ter efeitos secundários indesejados, pois este processo é bastante dispendioso, e para isso a utilização de *deep learning* é uma das possíveis soluções para prever se um produto será viável ou não, resultando numa maior poupança para a empresa. Outro aspecto também bastante relevante é o facto de estes produtos serem críticos, ou seja, estes medicamentos têm como principal objetivo serem utilizados por seres humanos o que acarreta sempre algum



risco (efeitos secundários adversos que podem resultar em lesões graves, inclusive morte dos sujeitos). Numa empresa farmacêutica cujo principal objetivo é gerar lucros, a ocorrência de falhas/retiradas de mercado de um produto tem um grave impacto na percepção do público resultando numa possível perda de confiança e consequente prejuízo. Portanto, com o desenvolvimento deste projeto o objetivo principal seria garantir que os medicamentos seriam notoriamente mais seguros, ou seja, com menos probabilidade de terem efeitos secundários indesejados, garantindo assim um melhor desenvolvimento de novos produtos para uma farmacêutica.

Para isso foi necessário aliar a esta motivação, alguns objetivos, entre eles uma aprendizagem de conceitos sobre *deep learning* e *Graph Convolutions*. Ou seja, o objetivo é explorar diferentes formas de *representation learning* para os compostos, usando o dataset *SIDER*, testar diferentes modelos de *deep learning* para prever efeitos secundários (treinados com o *SIDER*) e comparar resultados. Posteriormente, repetir todo o processo aplicado no *SIDER* no dataset do *OFFSIDES*.

1.3 Estruturação do Documento

Relativamente à estrutura deste documento, no capítulo 1, é realizada uma pequena contextualização onde são demonstradas as diferenças entre alguns termos importantes, como inteligência artificial, *machine learning* e *deep learning*.

É feito também uma pequena contextualização no que respeita ao *deep learning*, bem como a apresentação da motivação e dos objetivos da realização deste projeto. Por último, é abordada a estrutura do mesmo documento.

No capítulo 2 são apresentados alguns conceitos teóricos que são importantes reter para a realização do trabalho, desde o que são redes neuronais, aos diferentes tipos de arquiteturas. Além disso são abordados os conceitos de *Graph Convolutions*, *Featurization*, *Multitask Classifier*, *Mol2Vec* de forma a estabelecer um conhecimento geral sobre os principais tópicos do projeto de forma a poder melhor compreendê-lo.



2 Enquadramento Teórico

Neste capítulo vão ser abordados alguns tópicos, mais teóricos, que são importantes referir no desenvolvimento deste trabalho. Como é pretendido desenvolver redes neuronais que consigam prever efeitos secundários de compostos químicos utilizados em fármacos, foi então necessário realizar uma pesquisa sobre essa mesma temática, bem como outros tópicos.

Na secção 2.1 são abordados diferentes tópicos, desde uma definição teórica de rede neuronal artificial, à sua inspiração biológica, passando pelos diferentes tipos de redes existentes. São também abordados nesta secção os diferentes paradigmas de aprendizagem, o algoritmo de *backpropagation* e as funções de ativação mais utilizadas.

Já na secção 2.2, como para a realização deste projeto será discutida a temática de *Multitask Classifier* e *Mol2Vec*.

Por último, na secção 2.3 aborda-se um pouco o *Deepchem* que serviu como base para o desenvolvimento do projeto.

2.1 Redes Neuronais Artificiais

Uma rede neuronal é uma estrutura conexionista, onde o processamento se encontra distribuído por um grande conjunto de pequenas unidades densamente interligadas [1]. Tendo como inspiração os sistemas biológicos estas unidades processadoras são usualmente chamadas de neurónios.

Atualmente a variedade de redes neuronais artificiais a serem utilizadas é enorme, com a modelização de cada neurónio a variar desde a simples soma pesada das entradas a conjuntos complexos de equações diferenciais; a informação a circular na rede apenas num sentido ou em vários; os neurónios sendo atualizados simultaneamente ou com intervalos temporais entre eles, etc, resultando daqui um número quase ilimitado de combinações possíveis [1].

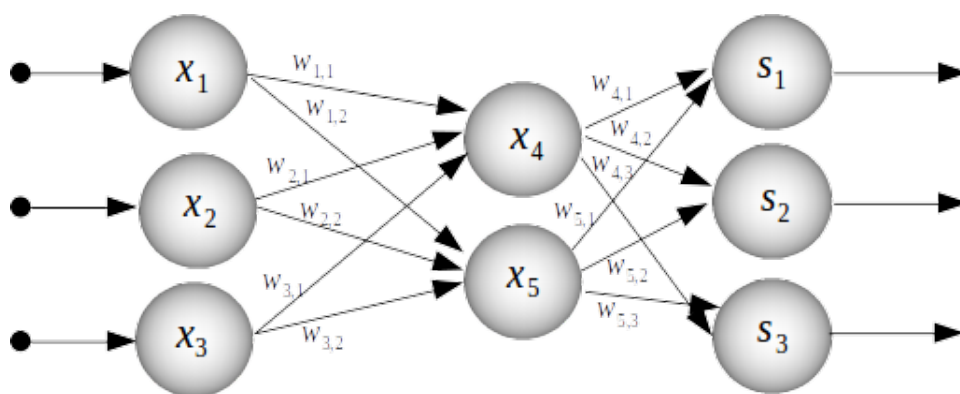


Figura 1: Representação de uma rede Neuronal

Na figura 1 podemos ver a representação de uma rede neuronal *single layer*.

Esta rede consiste numa camada de neurónios de entrada (onde a informação entra), uma camada de neurónios de saída (de onde o resultado pode ser retirado) e um número das chamadas camadas ocultas entre elas [1].



2.1.1 Inspiração Biológica

Um neurónio biológico, apresentado na figura abaixo, consiste numa única célula que tem a capacidade de realizar uma forma simples de processamento. Cada neurónio é estimulado por uma ou mais ligações vindas de outros neurónios às quais são denominadas de sinapses, dependendo o sinal produzido tanto da força das ligações como da sua natureza (inibitória, excitatória, etc). O sinal é propagado ao longo do axónio indo, por sua vez, estimular outros neurónio [2]. O funcionamento dos neurónios artificiais apresentado na figura 2, baseia-se, na maioria dos casos no modelo simplificado dos neurónios biológicos.

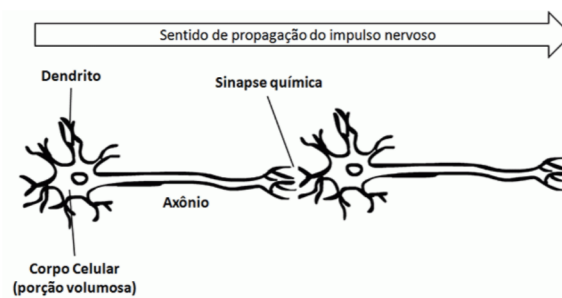


Figura 2: *Representação de um Neurónio Biológico*

O neurónio natural é uma célula que responde a sinais electro-químicos e o seu papel no sistema nervoso central humano é o de conduzir impulsos, estímulos elétricos em paralelo de modo a produzir determinadas reações [2].

O neurónio é composto por um núcleo, por um corpo celular, por muitos dendrites e por axónios [2]. As informações são transmitidas de um neurónio para outro através de reações químicas chamadas sinapses. Os axónios são os responsáveis por transmitir os estímulos a outros neurónios e os dendrites são as entidades que recebem os sinais de outros neurónios.

O desejo dos investigadores sempre foi de desenvolver um sistema com inteligência artificial que pudesse beneficiar do progresso constante da neuro-biologia, processos esses que nos permitem compreender processos complexos do cérebro, além de criar modelos da sua principal unidade que o constitui, o neurónio. As redes neuronais artificiais são a abordagem mais recente e inovadora a esse problema.

2.1.2 Tipos de Redes Neuronais

No âmbito das redes neuronais artificiais existem vários tipos de arquiteturas de rede, como por exemplo:

- Redes *Perceptron*
- Redes *Perceptron* Multi-Camada
- Redes Convolucionais
- Redes Recorrentes

Rede *Perceptron*



A rede *perceptron* simples é uma rede que é constituída por várias unidades interligadas e por uma única camada. O treino de uma *perceptron* consiste simplesmente em adaptar os pesos das suas unidades até que a rede tenha um comportamento que achemos o desejado [3]. Esta operação é realizada de uma forma supervisionada.

A figura 3 representa de forma fiel um *Perceptron*:

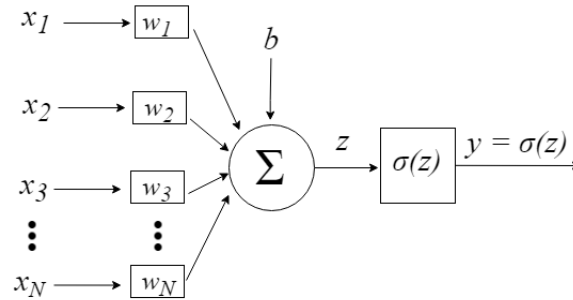


Figura 3: Representação de um *Perceptron* Simples

Os sinais da entrada no neurónio são o vetor $x = [x_1, x_2, x_3, \dots, x_N]$, podendo corresponder aos pixels de uma imagem. Ao chegarem ao neurónio, são multiplicados pelos respetivos pesos sinápticos, que são os elementos do vetor $w = [w_1, w_2, w_3, \dots, w_N]$, gerando o valor z , que é normalmente chamado de potencial de ativação. O termo b dá um grau de liberdade a mais, correspondendo tipicamente ao “*bias*” (viés) [3].

Por fim, o valor z passa então por uma função matemática de ativação, responsável por limitar tal valor a um certo intervalo, produzindo o valor final de saída do neurónio. As funções de ativação mais utilizadas são a sigmoide, tangente hiperbólica, *softmax* e ReLU (*Rectified Linear Unit*).

Rede *Perceptron* Multicamada

As redes *Perceptron* Multi-camada (*MultiLayer Perceptron*) são atualmente conhecidas por redes *feedforward*, e isso acontece porque cada neurónio dessas redes só pode estar conectado a unidades da camada seguinte e nunca da camada anterior. Isto faz com que o fluxo de processamento desde a entrada até à saída seja unidirecional, o que a diferencia de outras redes com realimentação (redes *feedback*) como é o caso das redes de *Hopfield* [4].

Este tipo de redes permitem a criação de múltiplas camadas escondidas o que permite a resolução de problemas cuja separação entre classes não seja linear.

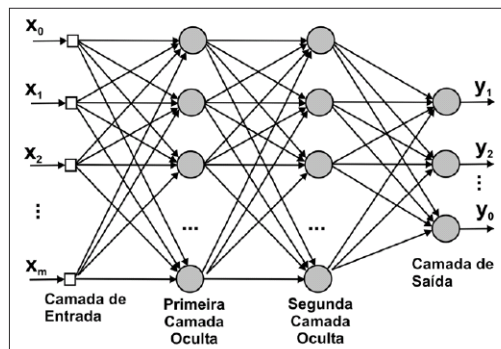


Figura 4: Rede Neuronal *FeedForward* MultiCamada



Neste tipo de redes neuronais, tal como verificado na figura 4, o sinal de entrada propaga-se para a frente, através da rede, sem a existência de ciclos [5]. A existência de nodos intermédios e a não linearidade e o alto grau de conectividade tornam esta arquitetura bastante poderosa como máquina de aprendizagem.

Redes Convolucionais

A Rede Neuronal Convolutiva foi introduzida por *Le Cun et al.* [6]. Este tipo de rede é especialmente desenvolvida e aperfeiçoada para reconhecimento visual, porque a extração de características é feita pela própria rede, que é treinada com esta.

Este tipo de rede neuronal profunda é usualmente dividida em duas partes, o extrator de características, que pode ser composto por camadas de convolução e de redução, e o classificador, composto por camadas totalmente ligadas, como numa Rede Neuronal Artificial.

As redes neuronais convolucionais surgiram devido à motivação biológica presente no trabalho de *Hubel e Wiesel* [7] que, ao efetuarem testes com o Córtex Visual de gatos, descobriram que tanto as células simples, como as complexas, são sensíveis a certos padrões e orientações.

O interesse comercial nestas redes teve um impulso muito forte em 2012, *xKrizhevsky et al.* [8] ganharam o concurso de reconhecimento de objetos promovido pela *ImageNet*.

Estas redes consistem num conjunto de camadas que extraem características das imagens de entrada através de sucessivas convoluções e redimensionamentos. No final desse processo pretende-se ficar apenas com a marca da classe a que a imagem de entrada pertence. Este tipo de redes são muito fáceis de treinar e com menos parâmetros do que outras redes totalmente ligadas.

Na figura 5 podemos ver uma imagem que representa a topologia da Rede Neural Convolutiva.

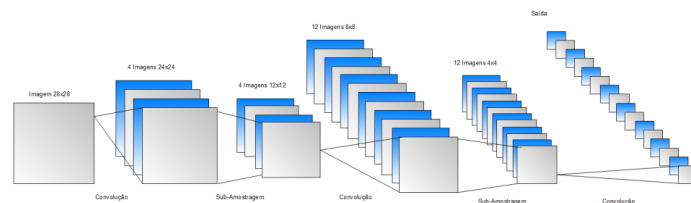


Figura 5: Exemplo de uma Rede Neural Convolutiva

Temos aqui representado um exemplo de uma rede Convolutiva que recebe uma imagem 28x28 que é alvo de uma convolução com um filtro de 5x5 que dá origem ao primeiro plano da primeira camada sendo os restantes três planos resultado da convolução com outros filtros de modo a extrair da mesma imagem diferentes características.

Depois de se realizar a convolução temos a fase de sub-amostragem que tem como objetivo diminuir a quantidade de informação e fazer uma agregação das características obtidas na convolução. Esse processo repete-se se ter uma camada final com o tamanho desejado, normalmente 1x1.

Redes Recorrentes

As Redes Neuronais Recorrentes (*Recurrent Neural Networks*) [9] constituem uma classe de redes onde a evolução do estado depende da entrada corrente e do estado atual. Essa propriedade possibilita a realização de um processamento dependente do contexto permitindo aprender as dependências de longo prazo: Sinais que sejam fornecidos a uma rede recorrente num instante de tempo t podem alterar o comportamentos dessa rede em momentos posteriores $t+k$, $k > 0$.



Uma rede recorrente pode ter conexões que voltem dos nós de saída aos nós de entrada, ou até mesmo conexões arbitrárias entre quaisquer nós [10].

A figura 6 representa uma rede neuronal recorrente.

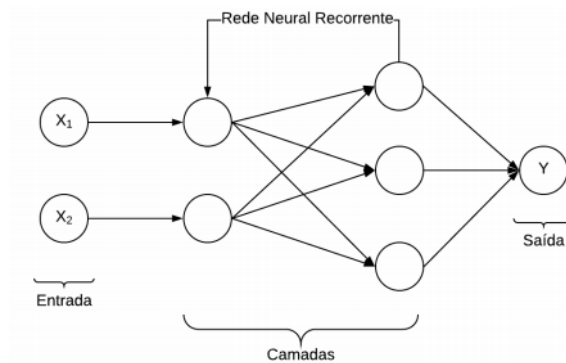


Figura 6: Representação de uma Rede Neuronal Recorrente

Existem variados tipos de arquiteturas de redes recorrentes, entre as mais conhecidas estão as seguintes:

- Redes *Hopfield*
- Redes Hierárquicas
- Redes Recursivas Bi-Direccionais
- Redes *Holman*
- Redes *Long Short Term Memory*

2.1.3 Algoritmo de *Backpropagation*

As redes neurais com uma única camada tem um interesse muito pequeno para a resolução de problemas complexos, estas podem ser empregues somente em problemas que não exijam mais do que um hiperplano a separar o espaço de classificação. Foi necessário, portanto investigar técnicas que permitissem a aprendizagem a redes compostas por mais do que uma camada.

Finalmente os investigadores *Le Cun, Parker, Rumelhart Hinton Williams* [11] apresentaram um processo que permite ajustar os pesos de redes *feedforward* com mais do que uma camada. O método proposto é conhecido por *backpropagation*, ou por regra delta generalizada que significa a propagação do erro no sentido da saída da rede para a sua entrada [12].

O treino de uma rede usando o algoritmo *backpropagation* divide-se em três etapas:

- A primeira etapa conhecida geralmente por *feedforward*, consiste na computação das saídas da rede para um vetor de treino de entrada.
- Na segunda etapa é realizada a comparação do vetor de saída resultante da computação anterior com o vetor pretendido e retro-propaga o erro associado para as camadas anteriores; isto é, no sentido das unidades de entrada.



- Na última etapa os pesos são ajustados de acordo com o erro propagado na etapa anterior. Esta sequência de passos terá de ser repetida durante vários ciclos de conjuntos de padrões de entrada e saída até que o erro entre os padrões de saída e os pretendidos permita que os resultados desejados sejam minimamente aceitáveis.

2.1.4 Funções de Ativação

As funções de ativação não são mais que um componente matemático introduzido nos neurónios artificiais que permitem que pequenas mudanças nos pesos e no *bias* causem alteração no *output*, sendo esta funcionalidade uma das capacidades cruciais para que uma rede neuronal artificial aprenda. Baseado no *input* que recebe, estas funções vão decidir se um determinado neurónio deve ser ativado ou se se deve manter inativo. Uma rede neuronal sem funções de ativação passa a ser apenas um modelo de regressão linear.

Em seguida são apresentadas as funções de ativação mais utilizadas no desenvolvimento de redes neuronais.

Função Linear

É considerada a função de ativação mais básica dentro do leque de funções de ativação porque não altera a saída de um neurónico. Ela é normalmente utilizada nas camadas de saída em redes neuronais de regressão [13].

A expressão para esta função de ativação é:

$$f(x) = p.x \quad (1)$$

Sendo que a sua derivada é apresentada da seguinte forma:

$$f'(x) = p \quad (2)$$

Na figura 7, são apresentadas as saídas lineares e suas respectivas derivadas.

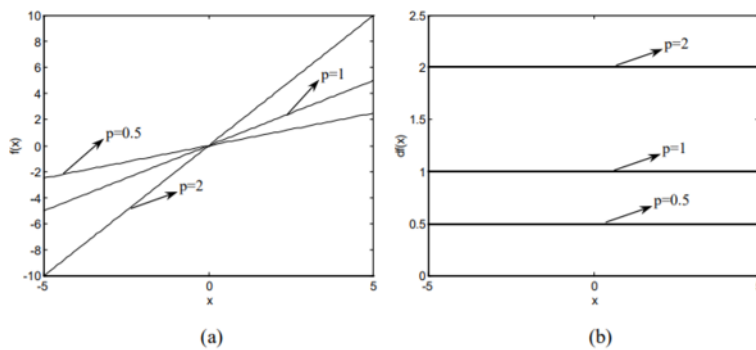


Figura 7: Função de Ativação linear e sua Derivada

Como se pode verificar, por exemplo, a saída linear quando $p = 1$ simplesmente o sinal que entra no neurónio, na sua saída.

Sigmóide



Este é um tipo de função de ativação que é não linear sendo normalmente utilizado em redes neuronais com propagação positiva (*feedforward*). Esses tipos de redes necessitam de ter a sua saída no intervalo $]0, 1[$ [14]. É uma função de ativação muito utilizada, por exemplo, na realização de treinos, porém esta função de ativação poderá ser substituída por funções que utilizam a tangente hiperbólica ou a ReLU (Unidade Linear Retificada).

A função sigmóide, numericamente falando, é a seguinte:

$$\sigma(x) = \frac{1}{1 + e^x} \quad (3)$$

Sendo que a sua respetiva derivada:

$$\sigma'(x) = \sigma(x)(1 - \sigma(x)) \quad (4)$$

Olhando para o gráfico ilustrado na figura 8, podemos verificar que a função vai converter os valores do conjunto de menos infinito até mais infinito para um conjunto compreendido entre os valor 0 e 1.

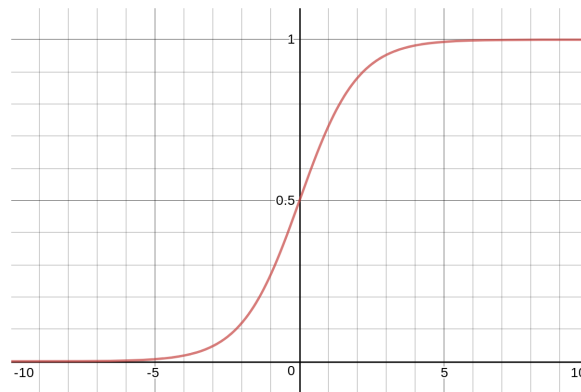


Figura 8: *Função de Ativação Sigmoidal*

Tangente Hiperbólica

A Função Tangente Hiperbólica (\tanh) é muito similar à função sigmoide que descrevemos acima, e tal com a essa também tem um formato de 'S', contudo varia de -1 a 1, ao contrário de 0 a 1 como na sigmoide. A função de ativação \tanh aproxima-se mais da identidade o que a torna uma alternativa mais atraente do que a função sigmoide para servir de ativação às camadas ocultas das Redes Neuronais Artificiais [15].

A expressão numérica desta função de ativação é a seguinte:

$$\tanh(x) = 2\sigma(2x) - 1 \quad (5)$$

Sendo a sua derivada é expressa da seguinte forma:

$$\tanh'(x) = 1 - \tanh^2(x) \quad (6)$$

O gráfico da figura 9, ilustra a representação gráfica da tangente hiperbólica, bem como da sua derivada. Podemos verificar então um formato de 'S' desta função, tal como supracitado.

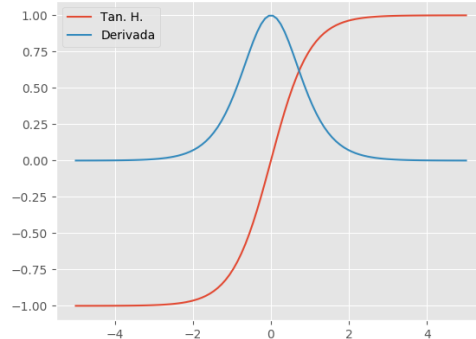


Figura 9: *Função de Ativação Tangente Hiperbólica*

Unidade Linear Retificada (ReLU)

A função de ativação Unidade Linear Retificada (ReLU) foi introduzida em 2000, e é neste momento uma das mais populares e usadas em processos de *deep learning*.

A ReLU é recomendada por ser linear e não saturante, é implementada com funções menos complexas que a sigmóide e a tangente hiperbólica, que utilizam exponenciais.

A função retorna 0 se receber qualquer entrada negativa, mas para qualquer valor positivo X retorna esse valor de volta [16].

A fórmula matemática desta função é dada pela expressão:

$$ReLU(x) = \max\{0, x\} \quad (7)$$

Sendo a sua derivada representada por a expressão:

$$ReLU'(x) = 4 \quad (8)$$

Esta função de ativação tem a sua representação gráfica, ilustrada na figura 10.

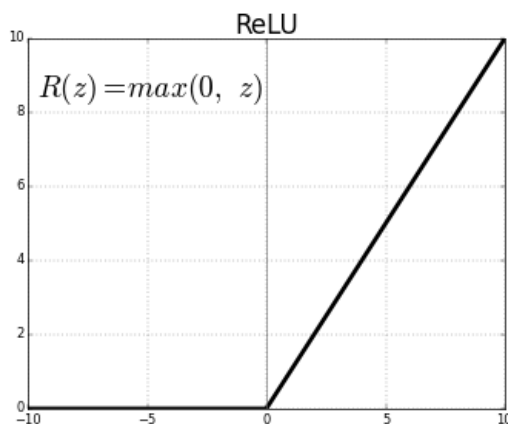


Figura 10: *Função de Ativação ReLU*



Softmax

Por último, a função de ativação *softmax* é utilizada principalmente em redes neuronais de classificação, e é adequada para bases de dados não balanceadas por possuir um neurónio de saída para cada classe definida. A função implica que a saída da rede represente a probabilidade dos dados serem de uma das classes definidas. Sem a utilização deste tipo de função, as saídas dos neurónios são simplesmente valores numéricos em que o maior valor indica a classe ganhadora [17].

A fórmula matemática desta função de ativação é dada pela expressão:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^k e^{z_k}}, j = 1, \dots, k \quad (9)$$

A figura 11 representa esta mesma função.

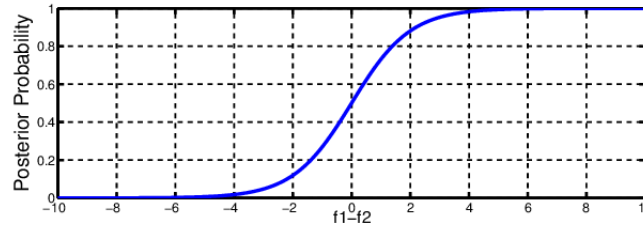


Figura 11: Função de Ativação Softmax

A regressão *Softmax* é uma forma de regressão logística que normaliza um valor de entrada em um vetor de valores que segue uma distribuição de probabilidade cujo total é igual a 1.

Os valores de saída estão entre o intervalo $[0,1]$ que é considerado uma vantagem porque é capaz de evitar a classificação binária e acomodar quantas as classes queiramos no nosso modelo de rede neuronal. Essa é a razão pelas qual a *softmax* é por vezes referido como uma regressão logística multinomial.

2.2 Outros conceitos

De forma a cobrir todos os temas necessários para uma melhor compreensão do projeto em causa, é necessário abordar os temas de *Multitask Classifier* e *Mol2Vec*.

2.2.0.1 Multitask Classifier

"Multitask Learning is an approach to inductive transfer that improves generalization by using the domain information contained in the training signals of related tasks as an inductive bias. It does this by learning tasks in parallel while using a shared representation; what is learned for each task can help other tasks be learned better." [18]

Portanto, se várias previsões fossem necessárias, uma rede *Multitask* seria uma boa escolha. Um exemplo seria uma previsão de um valor real contínuo, a par de uma ou mais previsões categóricas como *outcomes* previstos.



2.2.0.2 Mol2Vec

"Inspired by natural language processing techniques, we here introduce Mol2vec, which is an unsupervised machine learning approach to learn vector representations of molecular substructures. Like the Word2vec models, where vectors of closely related words are in close proximity in the vector space, Mol2vec learns vector representations of molecular substructures that point in similar directions for chemically related substructures. Compounds can finally be encoded as vectors by summing the vectors of the individual substructures and, for instance, be fed into supervised machine learning approaches to predict compound properties." [19]

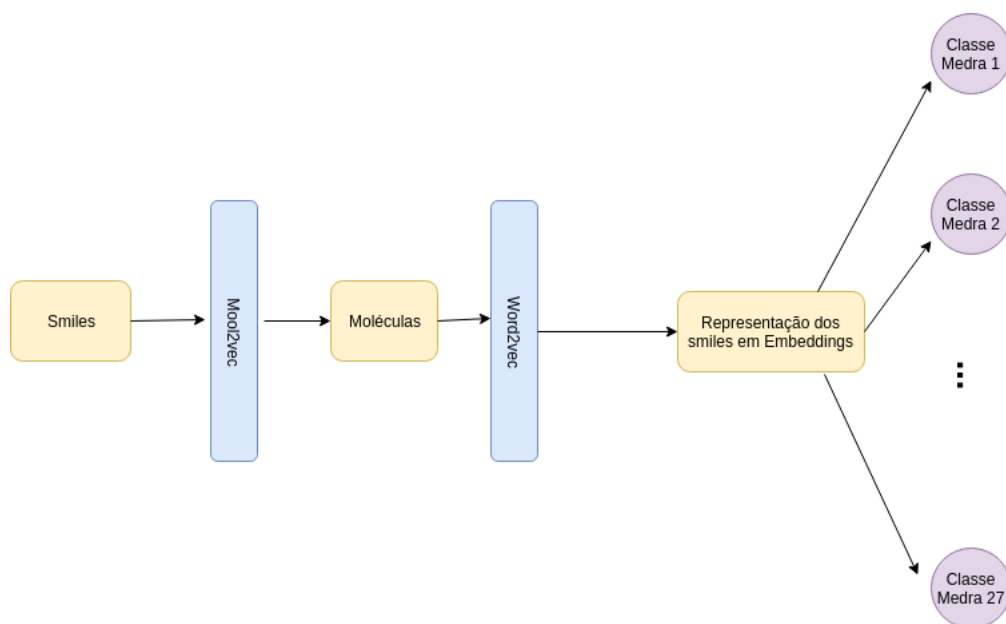


Figura 12: Workflow do Mol2Vec

2.3 Deepchem

O Deepchem é uma framework de Deep Learning utilizada para o descobrimento de compostos químicos. É open-source, pois o seu código é disponibilizado no GitHub. Desenvolvida em Python, o Deepchem oferece um conjunto de funcionalidades rico em atributos para aplicar *deep learning* a problemas na descoberta de drogas/compostos químicos e na quiminformática.

Frameworks antigas de *deep learning*, tal como o *scikit-learn* também foram aplicadas a quiminformática, mas o *Deepchem* foi o primeiro a acelerar a computação com GPUs NVIDIA.

O *Deepchem* utiliza *Tensorflow* a par do *Scikit-learn*, para expressar redes neuronais para *deep learning*.

2.3.1 Featurizations

SMILES são *strings* que representam inequivocamente moléculas e que podem ser usadas como um atributo molecular.



- **Extended-Connectivity Fingerprints (ECFP)** para datasets moleculares que não contêm grandes números de interações *non-bonded*
- **Graph Convolutions** Tal como o ECFP, *Graph Convolutions* produz representações granulares de topologias moleculares. Em vez de aplicar *hash* funções fixas, como no ECFP, o *Graph Convolutions* utiliza um conjunto de parâmetros que podem ser aprendidos através do treino de uma rede neuronal associada com uma estrutura molecular.



3 Datasets

3.1 Sider

O dataset do *Sider* é categorizada como sendo "*Physiology*" e é um problema de classificação de efeitos secundários. O dataset é constituído por 1,427 *compounds*, e este foi obtido do *MoleculeNet.ai*, um *benchmark* especialmente desenhado para testar metodos de *machine learning* com propriedades moleculares.

Devido ao facto de usarmos o dataset disponibilizado pelo *MoleculeNet* este já vem tratado e limpo, ou seja, não foi necessário fazer nenhum tipo de limpeza de dados nem pré-processamento pois os dados já estavam prontos a serem processados.

Nas imagens em baixo é possível ver o estado do dataset retirado do *MoleculeNet*.

```
In [16]: siderDataset.columns
Out[16]: Index(['smiles', 'Hepatobiliary disorders',
               'Metabolism and nutrition disorders', 'Product issues', 'Eye disorders',
               'Investigations', 'Musculoskeletal and connective tissue disorders',
               'Gastrointestinal disorders', 'Social circumstances',
               'Immune system disorders', 'Reproductive system and breast disorders',
               'Neoplasms benign, malignant and unspecified (incl cysts and polyps)',
               'General disorders and administration site conditions',
               'Endocrine disorders', 'Surgical and medical procedures',
               'Vascular disorders', 'Blood and lymphatic system disorders',
               'Skin and subcutaneous tissue disorders',
               'Congenital, familial and genetic disorders',
               'Infections and infestations',
               'Respiratory, thoracic and mediastinal disorders',
               'Psychiatric disorders', 'Renal and urinary disorders',
               'Pregnancy, puerperium and perinatal conditions',
               'Ear and labyrinth disorders', 'Cardiac disorders',
               'Nervous system disorders',
               'Injury, poisoning and procedural complications'],
              dtype='object')
```

Figura 13: Colunas do dataset

```
In [15]: siderDataset.head(5)
Out[15]:
```

	smiles	Hepatobiliary disorders	Metabolism and nutrition disorders	Product issues	Eye disorders	Investigations	Musculoskeletal and connective tissue disorders	Gastrointestinal disorders
0	C(CNCCNCCNCCN)N	1	1	0	0	1	1	1
1	CC(C) (C)C1=CC(=C(C=C1NC(=O)C2=CNC3=CC=CC=C3C2=...	0	1	0	0	1	1	1
2	CC[C@]12CC(=C)[C@H]3[C@H]([C@@H]1CC[C@]2(C#C)O...	0	1	0	1	1	0	1
3	CCC12CC(=C)C3C(C1CC[C@]2(C#C)O)CCC4=CC(=O)CCC34	1	1	0	1	1	1	1
4	C1C(C2=CC=CC=C2N(C3=CC=CC=C31)C(=O)N)O	1	1	0	1	1	1	1

5 rows × 28 columns

Figura 14: Primeiros 5 registos do dataset



```
Out[18]: Hepatobiliary disorders          743
Metabolism and nutrition disorders      996
Product issues                          22
Eye disorders                           876
Investigations                          1151
Musculoskeletal and connective tissue disorders 997
Gastrointestinal disorders              1298
Social circumstances                    251
Immune system disorders                 1024
Reproductive system and breast disorders 727
Neoplasms benign, malignant and unspecified (incl cysts and polyps) 376
General disorders and administration site conditions 1292
Endocrine disorders                     323
Surgical and medical procedures         213
Vascular disorders                     1108
Blood and lymphatic system disorders   885
Skin and subcutaneous tissue disorders 1318
Congenital, familial and genetic disorders 253
Infections and infestations             1006
Respiratory, thoracic and mediastinal disorders 1060
Psychiatric disorders                   1016
Renal and urinary disorders             911
Pregnancy, puerperium and perinatal conditions 125
Ear and labyrinth disorders             659
Cardiac disorders                       988
Nervous system disorders                1304
Injury, poisoning and procedural complications 946
dtype: int64
```

Figura 15: Contagem dos efeitos secundários presentes no dataset

3.2 Offsides

O dataset do *Offsides* é um recurso com 438,801 *off-labels*, dos quais esses efeitos não estão listados na lista oficial de drogas do FDA, tem efeitos secundários para 1332 drogas e 10,097 eventos adversos.

Portanto, como é possível deduzir há muitas repetições algo que foi preciso tratar no decorrer do projeto.

O nosso objetivo foi transformar o dataset do *Offsides* e deixá-lo à imagem do dataset do *Sider* de forma a poder aplicar os mesmos modelos e poder comparar os resultados e daí retirar conclusões. Para tal, foi necessário transformar cada linha do dataset de forma a incluir a informação do *SMILE* de cada linha, tal como existe no *Sider*, posteriormente foi necessário fazer um mapeamento das 27 classes do *MedDRA*.

Justificação da transformação: Tal como se verificou no *Sider*, foi originalmente testada com os originais 5868 categorias de efeitos secundários mas devido a uma falta de sinal a ao nível da granularidade foram agrupados os efeitos secundários nas 27 *Organ Classes* de acordo com a classificação do *MedDRA*. [20]

3.2.1 Offsides Original

Nesta sub-seção iremos demonstrar o dataset original do *Offsides*, antes de efetuarmos algum tipo de modificações.

```
In [9]: dataset = pd.read_csv("/Users/angelabarras/Documents/MESTRADO/DATA SCIENCE/LEI/offsides.csv", sep = ';', error_bad_lines=False)

In [11]: dataset.columns

Out[11]: Index(['stitch_id', 'drug', 'umls_id', 'event', 'rr', 'log2rr', 't_statistic',
               'pvalue', 'observed', 'expected', 'bg_correction', 'sider',
               'future_aers', 'medeffect'],
              dtype='object')
```

Figura 16: Colunas do Offsides



```
In [12]: dataset.head(10)
```

```
Out[12]:
```

	stitch_id	drug	umls_id	event	rr	log2rr	t_statistic	pvalue	observed	expected	bg_correction
	CID000000076	dehydroepiandrosterone	C0000737	abdominal pain	2.250000	1.16992500144231	6.537095	6.156712e-07	9	4.000000	0.00284883
	CID000000076	dehydroepiandrosterone	C0001622	hyperadrenalism	11.000000	3.45943162512943	4.782699	1.644408e-03	2	0.181818	4.1156e-01
	CID000000076	dehydroepiandrosterone	C0001623	adrenal insufficiency	2.200000	1.1375035238942	4.315199	9.884952e-03	2	0.909091	0.00249053
	CID000000076	dehydroepiandrosterone	C0002792	anaphylactic reaction	2.588235	1.37196877781128	4.590918	3.355380e-03	4	1.545455	0.00050312
	CID000000076	dehydroepiandrosterone	C0002940	aneurysm	7.333333	2.87446912176333	4.598374	3.399457e-03	2	0.272727	0.00013488
	CID000000076	dehydroepiandrosterone	C0003467	Anxiety	1.360825	0.444481277201072	4.467544	5.116250e-03	12	8.818182	0.00797747
	CID000000076	dehydroepiandrosterone	C0003838	arterial occlusive disease	5.500000	2.45943161983954	8.151512	2.663939e-10	6	1.090909	0.00142051
	CID000000076	dehydroepiandrosterone	C0003862	Aching joints	3.215385	1.68499131907462	12.399402	7.202613e-20	19	5.909091	0.00100272
	CID000000076	dehydroepiandrosterone	C0003873	Arthritis rheumatoid	4.888889	2.28950661751558	6.248953	2.607877e-06	4	0.818182	0.00096930
	CID000000076	dehydroepiandrosterone	C0004093	asthenia	1.958904	0.970046777977424	5.731688	2.279501e-05	13	6.636364	0.000749784999999999

Figura 17: Primeiras 10 linhas do dataset

Neste dataset não foi necessário fazer nenhum tipo de tratamento de valores nulos.

Nas imagens seguintes utilizamos o método describe do Pandas de forma a podermos analisar melhor o dataset em questão.

```
In [17]: dataset.describe()
```

```
Out[17]:
```

	rr	t_statistic	pvalue	observed	expected	sider	future_aers	medeffect
count	438801.000000	438801.000000	4.388010e+05	438801.000000	438801.000000	438801.000000	438801.000000	438801.000000
mean	5.819368	7.839682	3.993400e-03	16.576241	6.519147	0.042940	0.100492	0.035205
std	3.358977	5.957051	9.496948e-03	67.387218	31.994774	0.202722	0.300655	0.184298
min	0.018108	3.144852	3.695847e-189	1.000000	0.090909	0.000000	0.000000	0.000000
25%	2.750000	4.946780	2.158702e-11	3.000000	0.454545	0.000000	0.000000	0.000000
50%	5.133333	6.260961	5.080864e-06	6.000000	1.181818	0.000000	0.000000	0.000000
75%	8.800000	8.792227	1.612345e-03	13.000000	4.000000	0.000000	0.000000	0.000000
max	11.000000	641.934930	4.999926e-02	8433.000000	3994.363636	1.000000	1.000000	1.000000

Figura 18: Describe do dataset



3.2.2 Offsides Após Tratamento

```
In [19]: data_processed = pd.read_csv("/Users/angelabarros/Documents/MESTRADO/DATA SCIENCE/LEI/offsides_ordenado3.csv")

In [20]: data_processed.columns

Out[20]: Index(['smiles', 'Renal and urinary disorders',
               'Respiratory, thoracic and mediastinal disorders',
               'Skin and subcutaneous tissue disorders',
               'Congenital, familial and genetic disorders', 'Hepatobiliary disorders',
               'Infections and infestations', 'Cardiac disorders',
               'Ear and labyrinth disorders', 'Investigations', 'Vascular disorders',
               'Endocrine disorders', 'Immune system disorders', 'Product issues',
               'Blood and lymphatic system disorders',
               'Pregnancy, puerperium and perinatal conditions',
               'Surgical and medical procedures', 'Psychiatric disorders',
               'Reproductive system and breast disorders', 'Social circumstances',
               'Eye disorders', 'Gastrointestinal disorders',
               'General disorders and administration site conditions',
               'Neoplasms benign, malignant and unspecified (incl cysts and polyps)',
               'Injury, poisoning and procedural complications',
               'Musculoskeletal and connective tissue disorders',
               'Metabolism and nutrition disorders', 'Nervous system disorders'],
              dtype='object')
```

Figura 19: *Colunas do Offsides após tratamento*

```
In [39]: new_data

Out[39]: Renal and urinary disorders      100
         Respiratory, thoracic and mediastinal disorders      123
         Skin and subcutaneous tissue disorders      141
         Congenital, familial and genetic disorders      196
         Hepatobiliary disorders      71
         Infections and infestations      308
         Cardiac disorders      88
         Ear and labyrinth disorders      32
         Investigations      460
         Vascular disorders      79
         Endocrine disorders      59
         Immune system disorders      52
         Product issues      31
         Blood and lymphatic system disorders      81
         Pregnancy, puerperium and perinatal conditions      71
         Surgical and medical procedures      259
         Psychiatric disorders      112
         Reproductive system and breast disorders      118
         Social circumstances      64
         Eye disorders      138
         Gastrointestinal disorders      173
         General disorders and administration site conditions      147
         Neoplasms benign, malignant and unspecified (incl cysts and polyps)      268
         Injury, poisoning and procedural complications      200
         Musculoskeletal and connective tissue disorders      114
         Metabolism and nutrition disorders      85
         Nervous system disorders      184
         dtype: int64
```

Figura 20: *Soma dos efeitos secundários presentes no dataset*

Portanto, no decorrer do tratamento do dataset, o *Offsides* sofreu uma redução drástica de tamanho, passando de 438801 linhas para 881. Ou seja, uma redução de cerca de 99,8% em relação ao dataset original.

Neste formato, cada linha do dataset apresenta o(s) efeito(s) secundário(s) presentes no *MedDRA* e não há repetições em termos de *SMILES*, ou seja, um *SMILE* terá associado a si todos os efeitos secundários *MedDRA* na própria linha fazendo assim com que não haja repetições de *SMILES* ao longo do dataset.



3.3 *Sider + Offsides*

De seguida, juntou-se os datasets tratados do *Sider* e do *Offsides* de forma a se obter um terceiro dataset com a informação proveniente das duas fontes. Apelidou-se esse terceiro dataset de *Sider+Offsides*.

Para a geração do *Sider+Offsides* criou-se um script em Python que permitisse fazer o *append* dos dois datasets-fontes. Contudo, após feita a geração do terceiro dataset, concluiu-se que a geração estava incompleta, ou seja, foi feito o *append* dos datasets para um novo dataset mas não se verificou as linhas duplicadas e, juntando-se os *side effects* provenientes do *Sider* e do *Offsides* no terceiro dataset. Ou seja, fez-se o *append* e o *drop_duplicates* de forma a garantir que não havia *SMILES* repetidos, mas antes de fazer essa eliminação de repetidos não se fez a junção de efeitos secundários para o dataset final. Foi um lapso que o grupo apenas verificou depois de terem sido lembrados pela orientadora e, que devido à altura em que o aviso foi feito não foi possível refazer essa parte do código e consequentemente ter um terceiro dataset *inteiramente* correto. Portanto, o terceiro dataset, o *Sider+Offsides* tem como base dados que não estão 100% corretos, algo que irá influenciar os resultados finais, ou seja, os resultados não poderão ser tidos muito em conta.



4 Tratamento Offsides

Nota: Todo o código desenvolvido pode ser encontrado no repositório de GitHub disponibilizado juntamente com o relatório.

Como foi referido previamente, o *Offsides* não possuía o mesmo formato que o dataset *Sider* possuía e foi necessário fazer uma re-estruturação de forma a poder aplicar os mesmos modelos aplicados ao *SIDER* ao *Offsides* e podermos comparar resultados.

Para tal, foi necessário utilizar duas API's distintas de forma a re-estruturação ser bem sucedida:

- **Data.Bioontology**

O primeiro passo no tratamento dos dados do *Offsides* foi a utilização da API do Data.Bioontology. Após a leitura da documentação fornecida pela API viu-se que era necessário passar uma **API KEY** de forma a se poder fazer pedidos à API. Portanto, ao fazer um pedido no cabeçalho do mesmo tinha que se enviar a **API KEY** juntamente com o SOC id. Foi necessário iterar os resultados até ser assegurado que o resultado a ser retornado era o SOC raiz, ou seja, na classificação dos efeitos secundários há um agrupamento em hierarquia que define todos os efeitos secundários e, o que era mais relevante para o projeto era obter todos os efeitos secundários raiz, aqueles que englobavam todos os outros. Para tal, foi necessário iterar e "escalar" a hierarquia de SOC's de forma a obter o último e guardar esse SOC como o correspondente. Nesta primeira fase foi gerado um ficheiro .csv que continha todos os dados do dataset *Offsides* e mais uma coluna com os valores dos SOC's correspondentes a cada linha do dataset.

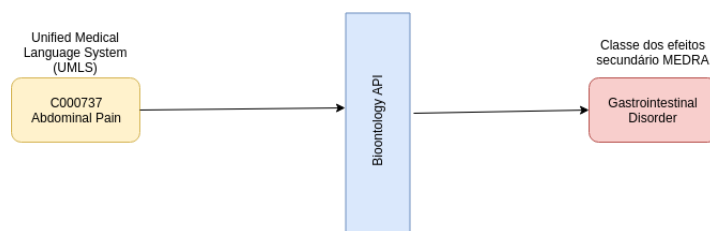


Figura 21: *Workflow da API*

- **PubChemPy**

A segunda fase do tratamento do dataset consistiu em pegar no ficheiro previamente gerado e modificá-lo para deixar no mesmo estado/formato que o *SIDER* continha. Ou seja, ter a coluna dos *SMILES* mais as 27 classes do *MedDRA*.

Para tal, pegou-se na coluna que continha o STICH-ID e chamou-se a API passando o STICH-ID como parâmetro. O retorno dessa chamada é o SMILE correspondente. Ou seja, o SMILE que foi retornado da API foi colocado numa nova coluna chamada smiles. Posteriormente, analisou-se qual o SOC que cada linha tinha e transformou-se as colunas para as 27 classes do MedDRA. Para tal, viu-se qual(ais) SOC(s) cada linha tinha e colocou-se um 1 em cada efeito secundário observado. Os efeitos secundários que não se manifestavam para cada linha, colocou-se um 0. Ou seja, cada linha é formada por vários 1 (efeitos secundários observados) e 0 (efeitos secundários não observados).

O resultado da operação anterior resultou num dataframe com várias linhas com SMILES repetidos, para tal fez-se um `drop_duplicates` de forma a garantir que todas as linhas estão bem preenchidas mas apenas ocorriam uma vez ao longo do dataframe.

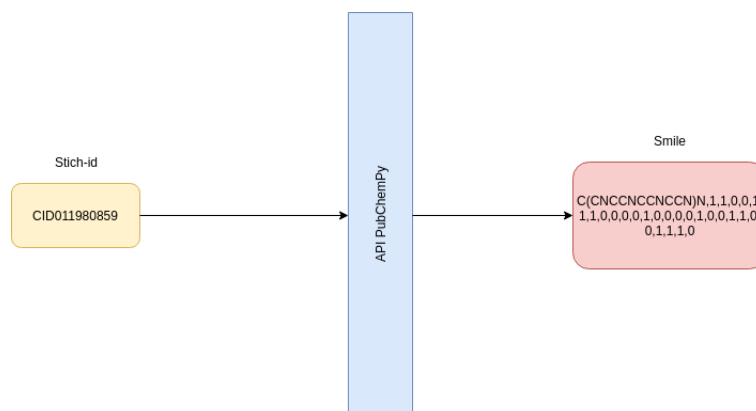


Figura 22: *Workflow da API*



5 Código

Nesta secção ir-se-á abordar o código desenvolvido no presente projeto. Como foi antes referido, desenvolveu-se vários modelos para ser aplicado a cada dataset.

5.1 Geração de Embeddings

Quando são gerados os *embeddings* utiliza-se um modelo Mol2Vec pré-treinado que é não-supervisionado, logo as variáveis de output não são utilizadas nesta fase. A partir dos *SMILES* o Mol2Vec retira as sub-estruturas da molécula, usando o algoritmo de Morgan. Estas sub-estruturas são consideradas "palavras" e a molécula inteira é considerada a "frase". Tendo os compostos nesta representação o Mol2Vec depois utiliza o modelo Word2Vec para gerar os *embeddings*.

```
In [ ]: #Transformar Smiles em Embeddings

train_embeddings_generated_from_mol2vec = generateEmbeddings(train_smiles_from_offsides, '/home/mamonteiro/source-cod
valid_embeddings_generated_from_mol2vec = generateEmbeddings(valid_smiles_from_offsides, '/home/mamonteiro/source-cod
test_embeddings_generated_from_mol2vec = generateEmbeddings(test_smiles_from_offsides, '/home/mamonteiro/source-cod

In [109]: train_embeddings_generated_from_mol2vec.head()
Out[109]:
```

	smiles	mol2vec_1	mol2vec_2	mol2vec_3	mol2vec_4	mol2vec_5	mol2vec_6	mol2vec_7	mol2vec_8
0	CC12CCC3C(C1CCC2=O)CC=C4C3(CCC(C4)O)C	-1.687902	-4.574107	-3.788245	0.820349	-1.165568	-7.593350	-8.685431	3.990403
1	C[N+](C)(C)CC(CC(=O)O)O	2.769547	-3.059868	-1.797269	-0.953987	4.924242	0.454827	-9.468398	1.997672
2	C(CC(=O)O)CN	-0.599405	0.307458	-0.714178	-0.674214	2.840279	-0.157350	-3.893690	-0.316774
3	C1C(N(C2=C(N1)NC(=NC2=O)N)C=O)CNC3=CC=C(C=C3)C...	1.644671	-4.325292	-5.132446	4.536877	2.365319	-1.383428	-16.086567	-2.104912
4	CCCCC(C=C1C(CC(=O)C1CC=CCCC(=O)O)O)O	-0.658838	-3.338789	-3.176840	-0.045156	6.762016	-4.120831	-17.303051	-0.583435

5 rows x 301 columns

Figura 23: Código de geração de embeddings

5.2 Graph Convolucional

Na imagem presente em baixo pode-se ver a construção do modelo *graph* convolucional utilizada para dar resposta ao problema. As *layers* gc1, gc2 gc3 são *GraphConv layers* com 64 nós e com a função de ativação ReLu.

A segunda camada da rede neuronal é uma camada de *Dropout* de forma a que a rede não aprenda demasiado, para tal utilizou-se uma *layer Dropout* com uma taxa de 50%. Posteriormente, temos uma *BatchNorm layer* que está ligada à segunda camada (a de Dropout).

A sétima camada trata-se de uma camada *GraphPool* que está ligada à sexta camada que é uma *BatchNorm*. O objetivo desta nova camada é criar uma *pooling* operação sobre um grafo arbitrário. A penúltima camada (dense) trata-se de uma camada densa com 128 nós.



Implementação de uma rede graph convolutional!

```
In [ ]: batch_size = 50

gc1 = GraphConv(64,activation_fn=tf.nn.relu,in_layers=[atom_features, degree_slice, membership] + deg_adjs)
dropout_1 = Dropout(dropout_prob=0.5, in_layers=[gc1])
batch_norm1 = BatchNorm(in_layers=[dropout_1])
gp1 = GraphPool(in_layers=[batch_norm1, degree_slice, membership] + deg_adjs)

gc2 = GraphConv(64,activation_fn=tf.nn.relu,in_layers=[gp1, degree_slice, membership] + deg_adjs)
batch_norm2 = BatchNorm(in_layers=[gc2])
gp2 = GraphPool(in_layers=[batch_norm2, degree_slice, membership] + deg_adjs)

gc3 = GraphConv(64,activation_fn=tf.nn.relu,in_layers=[gp1, degree_slice, membership] + deg_adjs)
batch_norm3 = BatchNorm(in_layers=[gc3])
gp3 = GraphPool(in_layers=[batch_norm3, degree_slice, membership] + deg_adjs)

dense = Dense(out_channels=128, activation_fn=tf.nn.relu, in_layers=[gp3])
batch_norm4 = BatchNorm(in_layers=[dense])

readout = GraphGather(batch_size=batch_size,activation_fn=tf.nn.tanh,in_layers=[batch_norm4, degree_slice, membersh
```

Figura 24: Código de geração da rede neuronal convolucional

5.3 Hyperparameters

Como uma forma de otimizar os resultados obtidos com os modelos, modificou-se vários parâmetros de forma a encontrar o conjunto que melhor resultado originaria. Como é possível ver na imagem, introduziu-se várias funções de ativação, tal como o tamanho do *batch*, a taxa de *dropout*, etc. Para efeitos ilustrativos foi feito um exemplo simples com poucas combinações de forma a podermos ter uma imagem que desse para utilizar para este presente relatório. No desenvolvimento do projeto foram utilizados mais parâmetros resultando num número muito mais elevado de combinações possíveis. Na presente imagem é possível ver que foram geradas 48 combinações, enquanto que no desenvolvimento foram geradas mais de 300,000 combinações.

hyperparameters

```
In [32]: params_dict = {"activation": ["relu", "sigmoid", "tanh"],
                        "optimizer": ['RMSprop', 'adam'],
                        "momentum": [0.9],
                        "dropouts": [0.5, 0.2],
                        "learning_rate": [1e-3],
                        "batch_size": [50],
                        "nb_layers": [64, 128, 256, 512],
                        "penalty": [0.]}

n_features = train_dataset.get_data_shape()[0]

def model_builder(model_params, model_dir):
    model = dc.models.MultitaskClassifier(
        len(sider_tasks), n_features, **model_params)
    return model

In [33]: metric = dc.metrics.Metric(dc.metrics.roc_auc_score, np.mean)
optimizer = dc.hyper.HyperparamOpt(model_builder)
best_dnn, best_hyperparams, all_results = optimizer.hyperparam_search(
    params_dict, train_dataset, valid_dataset, [], metric)

Fitting model 1/48
hyperparameters: {'activation': 'relu', 'optimizer': 'RMSprop', 'momentum': 0.9, 'dropouts': 0.5, 'learning_rate': 0.001, 'batch_size': 50, 'nb_layers': 64, 'penalty': 0.0}
computed metrics: [0.7778864970645794, 0.665813953488372, 0.9366197183098591, 0.6607142857142858, 0.7309734513274336, 0.6645145863223338, 0.859375, 0.6979714500375658, 0.7184418145956608, 0.7198748043818466, 0.6920133907221424, 0.667016806722689, 0.6621294615849969, 0.5626811594202898, 0.7480842911877394, 0.7314401622718054, 0.7601851851851852, 0.5736357659434583, 0.6368932038834951, 0.7431192660550459, 0.7868321640582838, 0.6403940886699506, 0.639763779527559, 0.6775289884046382, 0.7904761904761906, 0.808457711442786, 0.6841224294595887]
Model 1/48, Metric mean-roc_auc_score, Validation set 0: 0.712480
best validation_score so far: 0.712480
Fitting model 2/48
```

Figura 25: Optimização de hyperparameters



6 Resultados

6.1 Métricas utilizadas

A métrica utilizada para avaliar a *performance* dos modelos criados/utilizados durante o decorrer do desenvolvimento do projeto foi o **ROC-AUC**. Inicialmente irá ser feita uma breve explicação do que é o **ROC-AUC** e a sua importância.

A curva AUC é derivada da curva ROC, que significa “*Receiver Operating Characteristic*”. A ROC mostra o quão bom o modelo criado pode distinguir entre duas coisas (já que é utilizado para classificação). Essas duas coisas podem ser 0 ou 1, ou positivo e negativo. Os melhores modelos podem distinguir com precisão entre as duas coisas que está a avaliar.

O ROC possui dois parâmetros:

- Taxa de verdadeiro positivo (True Positive Rate), que é dado por $\text{true positives} / (\text{true positives} + \text{false negatives})$
- Taxa de falso positivo (False Positive Rate), que é dado por $\text{false positives} / (\text{false positives} + \text{true negatives})$

Assim, na tentativa de simplificar a análise da ROC, a AUC (“area under the ROC curve”) nada mais é que uma maneira de resumir a curva ROC num único valor, agregando todos os limiares da ROC, calculando a “área sob a curva”. O valor do AUC varia de 0,0 até 1,0 e o limiar entre a classe é 0,5. Ou seja, acima desse limite, o algoritmo classifica em uma classe e abaixo na outra classe. **Quanto maior o AUC, melhor.**

6.2 Resultados obtidos

Nesta sub-seção irá ser apresentado os resultados obtidos com o *Sider*, *Offsides* e *Sider+Offsides*.

6.2.1 Sider

- **Mol2Vec**

Na imagem apresentada em seguida pode-se ver os resultados das versões desenvolvidas. O melhor ROC-AUC é de 0,639 e corresponde ao **Second Scenario** que é o original do tutorial e feita uma otimização dos *hyperparameters*. Este resultado sofre os efeitos de os datasets estarem *imbalanced* e do pouco poder computacional que foi possível utilizar para a obtenção destes resultados.

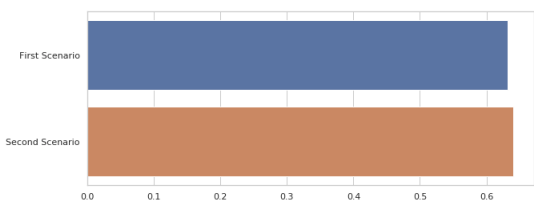
O **First Scenario** é referente ao modelo original do tutorial sem nenhum tipo de alteração feita, o resultado foi um ROC-AUC de 0.632.



```
In [43]: import seaborn as sns

sns.set(rc={'figure.figsize':(10,4)})
sns.set(style="whitegrid")
ax = sns.barplot(x=[test_score['mean-roc_auc_score'],test_scores_v1['mean-roc_auc_score']],
                y=['First Scenario','Second Scenario'])
ax.set(xlim=(0, None))

Out[43]: [(0, 0.6718843518190393)]
```



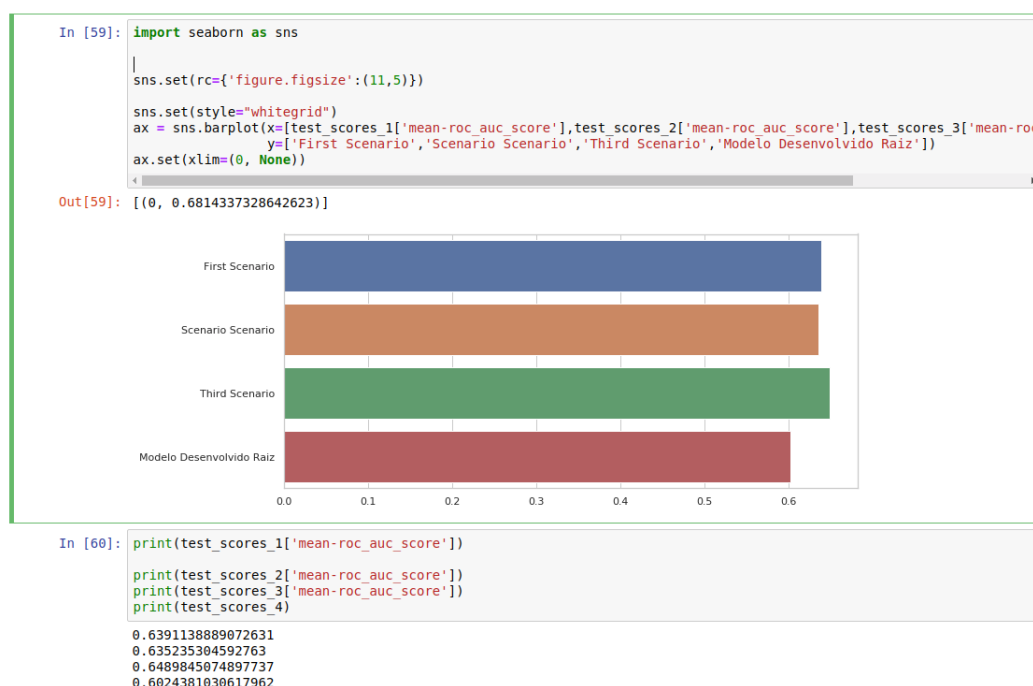
```
In [46]: print(test_score['mean-roc_auc_score'])
          print(test_scores_v1['mean-roc_auc_score'])

0.6321786868544663
0.6398898588752755
```

Figura 26: *Mol2Vec ROC-AUC*

• Graph Convolution

Como é possível observar na imagem em baixo, o modelo desenvolvido de raiz pelo grupo teve um ROC-AUC menor do que todos os outros modelos. O resultado foi um ROC-AUC de 0.6024 o que é bastante mais baixo ao melhor observado (**Third Scenario**) com um ROC-AUC de 0.6489

Figura 27: *ROC-AUC do Graph Convolution*

Portanto, o **First Scenario** trata-se do modelo original que está implementado no *Deepchem* e com esse modelo conseguiu-se atingir um ROC-AUC de 0.6391.

O **Second Scenario** é referente ao modelo original do *Deepchem* com uma ligeira modificação: o número de *epochs* a serem passadas como parâmetro ao modelo. Como é possível observar na imagem em baixo, o ROC-AUC diminui ligeiramente passando de 0.6391 para 0.6352.



O **Third Scenario** é referente ao modelo original do *Deepchem* mas com várias modificações nos parâmetros, ou seja, foi feita *hyperparameter optimization* de forma a tentar obter os melhores parâmetros para o modelo. Graças a esta técnica, conseguiu-se o melhor resultado de ROC-AUC de todas as tentativas com um ROC-AUC de 0.6489.

6.2.2 Offsides

- **Mol2Vec**

Na imagem apresentada em seguida podemos ver os resultados das versões desenvolvidas. O melhor ROC-AUC de 0,560 corresponde ao segundo cenário que foi desenvolvido e que foi obtido utilizando alguns *hyperparameters* no treino do modelo. Este resultado sofre os efeitos de os datasets estarem *imbalanced* e do pouco poder computacional que foi possível utilizar para a obtenção destes resultados.



Figura 28: Resultados utilizando o Mol2Vec no Offsides

- **Graph Convolution**

Na imagem apresentada abaixo apresenta-se os resultados dos ROC-AUC obtidos com os modelos desenvolvidos. O melhor ROC-AUC é de 0,560 e corresponde ao primeiro cenário que foi desenvolvido. Este resultado sofre os efeitos de os datasets estarem *imbalanced*.



Figura 29: Resultados utilizando o Modelo Graph Convolution no Offsides

6.2.3 Sider+Offsides

- Mol2Vec

A aplicação do modelo multitask sobre os embeddings criados permitiu obter um valor para o ROC-AUC de 0,52.



Figura 30: Resultados utilizando Mol2Vec no Sider+Offsides

- Graph Convolution



Fruto da utilização do modelo Graph convolutions sobre o dataset criado a partir do sider e offsides, obteve-se como melhor ROC-AUC o valor de 0,528 e corresponde ao cenário em que foram utilizados hyperparameters para o treino. Novamente, este resultado reflete fortemente os efeitos de ter datasets *imbalanced* e do pouco poder computacional que aplicado na obtenção dos resultados.



Figura 31: Resultados utilizando o Graph Convolution no Sider+Offsides



7 Conclusão

Durante o desenvolvimento do projeto, várias questões e desafios foram levantadas, desde a instalação do deepchem até ao balanceamento dos dados dos datasets utilizados no projeto. Contudo, o maior desafio enfrentado pelo grupo foi a questão do domínio da área, ou seja, o facto de se tratar de um tema com vários termos/terminologias por desconhecidos. Para além de ser necessário aprender técnicas de *deep learning*, foi necessário que ler vários artigos de forma a poder se compreender todo o jargão utilizado neste tipo de projetos. Portanto, houve uma grande curva de aprendizagem no que toca à familiarização do assunto em estudo.

Para além do desafio que o tema nos proporcionava, enfrentou-se vários outros reptos tecnológicos, desde a falta de poder de processamento para ser possível desenvolver o trabalho até a erros devido à inexperiência na área. Apesar de posteriormente ter sido dado acesso ao cluster do Departamento de Informática da Universidade do Minho, houve várias instâncias em que não foi possível submeter *jobs* no cluster, pelo que foi necessário fazer a maior parte do processamento nos computadores pessoais o que por si só atrasou ligeiramente o desenvolvimento e fez com que os resultados obtidos não fossem os desejados.

O projeto levantou problemas interessantes do ponto de vista tecnológico mas que, por serem inovadores e pouco utilizados/investigados pelo mercado obrigou a um acréscimo de tempo na sua investigação como a limpeza e transformação do dataset do Offsides devido à documentação pouco explícita das API's usadas no desenvolvimento deste projeto. Esta fase do projeto deu resultado num grande derrapamento em termos de previsão de execução pois foram necessárias várias iterações/modificações para se poder ter o dataset transformado final totalmente correto.

O desenvolvimento deste projeto permitiu porém executar tarefas diferentes e desafiantes como sejam a transformação dos smiles em embeddings utilizando o Mol2Vec onde foram utilizados modelos pré-construídos e utilização de redes graph convolutions e multitask.

Outro desafio encontrado, numa fase final, foi a questão do não balanceamento dos dados, *imbalanced data*, que foi constatado nos datasets utilizados. Isto deveria ter sido detetado inicialmente na análise exploratória que realizada no arranque do trabalho, contudo nenhuma referência foi feita em nenhum *paper* que sugerisse que este problema estivesse presente. É certo que isto é fruto da inexperiência e revelou-se uma grande oportunidade de aprendizagem para o grupo.

Podemos, em resumo, concluir que tivemos um trabalho desafiante e de grande riqueza tecnológica que nos desafiou desde a fase de pré-processamento passando pela fase de transformação dos dados até à fase de modelação.

8 Trabalho Futuro

Relativamente ao trabalho que ainda se pode desenvolver no âmbito deste tema de trabalho existem alguns aspetos que vale a pena enunciar. Desde logo, quando o pré-processamento dos dados deveria ser mais aprofundadamente estudado de maneira a resolver alguns problemas de imbalancing data que prejudica os resultados que foram apresentados neste nosso relatório. Além disso, na fase de modelação poderá-se estudar outros tipos de algoritmos de deep learning e outros tipos de tipologias. Finalmente, o poder computacional é muito importante na abordagem que se seguiu neste trabalho, logo os resultados finais poderão beneficiar da aplicação de hyperparameters que possam ajudar o sistema a descobrir as melhores configurações que deveriam ser usado no treino de cada um dos modelos utilizados.



Referências

- [1] John A Hertz. *Introduction to the theory of neural computation*. CRC Press, 2018.
- [2] Nancy Forbes. *Imitation of life: how biology is inspiring computing*. Mit Press Cambridge, 2004.
- [3] Jorge S Marques. Reconhecimento de padrões: métodos estatísticos e neuronais. 2005.
- [4] Thomas Walter Rauber. Redes neurais artificiais. *Universidade Federal do Espírito Santo*, 2005.
- [5] Luís Daniel Torres Rebelo et al. Avaliação automática do resultado estético do tratamento conservador do cancro da mama. 2008.
- [6] Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pages 396–404, 1990.
- [7] Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2722–2730, 2015.
- [8] April Yu, Raphael Palefsky-Smith, and Rishi Bedi. Deep reinforcement learning for simulated autonomous vehicle control. *Course Project Reports: Winter*, pages 1–7, 2016.
- [9] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.
- [10] Sara Marina Albino Rijo. Técnicas de deep learning para detecção de eventos em áudio: treino de modelos acústicos a partir de sinais puros. Master’s thesis, Universidade de Évora, 2018.
- [11] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.
- [12] Henry Leung and Simon Haykin. The complex backpropagation algorithm. *IEEE Transactions on signal processing*, 39(9):2101–2104, 1991.
- [13] Leandro Nunes de Castro Silva et al. Análise e síntese de estratégias de aprendizado para redes neurais artificiais. 1998.
- [14] Daniel dos Santos Vieira et al. *Protótipo para extração de metadados de vídeo*. PhD thesis, 2017.
- [15] Daniel Henrique Breda Binoti, Mayra Luiza Marques da Silva Binoti, and Helio Garcia Leite. Configuração de redes neurais artificiais para estimação do volume de árvores. *Revista Ciência da Madeira (Brazilian Journal of Wood Science)*, 5(1):10–12953, 2014.
- [16] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.
- [17] Katarzyna Janocha and Wojciech Marian Czarnecki. On loss functions for deep neural networks in classification. *arXiv preprint arXiv:1702.05659*, 2017.
- [18] R Caruana. Multi-task learning. 1997.
- [19] Turk S Jaeger S, Fulle S. Mol2vec: Unsupervised machine learning approach with chemical intuition. 2018.



- [20] Aneesh S. Pappu Vijay Pande Han Altae-Tran, Bharath Ramsundar. Low data drug discovery with one-shot learning. 2017.