Universidade do Minho
Escola de Engenharia

Mestrado em Engenharia Informática

Deep Learning aplicado à previsão da idade do cérebro

Aprendizagem Profunda

Grupo 7

Ano Letivo de 2021/2022

PG47164, Eduardo Coelho
PG47578, Pedro Veloso
PG47089, Carlos Preto
PG47677, Simão Monteiro

Gualtar, Maio de 2022

Conteúdo

1 Introdução **2**

2 *Data Understanding and Analysis* **2**

3 Primeira Abordagem - CNN (2 dimensões) **2**

4 Segunda Abordagem - CNN (2 dimensões) seguida de MLP **3**

5 Terceira abordagem - CNN (1 dimensão) seguida de MLP **4**

6 Controlo do Overfitting **5**
- 6.1 Divisão dos dados . . . 5
- 6.2 Batch Normalization . . . 6
- 6.3 Regularização (Ridge) . . . 6
- 6.4 Inicialização dos pesos . . . 6
- 6.5 Regularização com Dropout . . . 7
- 6.6 Restrições dos Pesos . . . 7
- 6.7 Redução da complexidade da rede . . . 7

7 Mais otimizações **7**
- 7.1 Batch-size . . . 7
- 7.2 Learning-rate . . . 7
- 7.3 LeakyRELU . . . 7
- 7.4 Rede final . . . 8

8 Casos mais difíceis de prever **9**

9 Conexões cerebrais mais relevantes para a previsão da idade do cérebro **10**
- 9.1 Predict com cada conexão cerebral . . . 10
- 9.2 Shap . . . 11

10 Conclusão **11**

1 Introdução

No contexto da disciplina de Aprendizagem Profunda, foi proposto o desenvolvimento e otimização de um modelo de aprendizagem profunda capaz de prever a idade do cérebro a partir de características de conectividade estrutural.

Para tal, foram disponibilizados ficheiros de treino e de teste com formatos diferentes. Os ficheiros com extensão **.mat** possuem as matrizes de conectividade estrutural de cada indivíduo, enquanto que os ficheiros com extensão **.csv** possuem a informação acerca dos indivíduos do caso de estudo. O objetivo final será prever, através da matriz de conectividade estrutural e da informação de cada indivíduo, a sua respetiva idade.

Ao longo do relatório serão abordadas todas as etapas e decisões associadas ao desenvolvimento do modelo de aprendizagem profunda, apresentando também os resultados obtidos e a respetiva análise crítica.

2 *Data Understanding and Analysis*

Antes de partir para o desenvolvimento do modelo, foi necessário analisar e interpretar os dados do problema. Existem dois tipos diferentes de dados, ficheiros **.mat** que contêm as matrizes de conectividade resultantes de ressonâncias magnéticas, e os ficheiros com extensão **.csv** possuem 4 atributos distintos, sendo estes:

- **id** - Identificação do Sujeito
- **age** - Idade do Sujeito
- **sex** - Sexo do Sujeito ($0 \rightarrow$ Masculino $\mid 1 \rightarrow$ Feminino)
- **education** - Nível de Escolaridade do Sujeito

Ao analisar cada um destes atributos, percebe-se que não há a presença de nenhum *outlier*.

O principal problema e desafio com os dados fornecidos é que temos 2 tipos distintos de dados, não sendo possível simplesmente juntar os dois e treinar um modelo com base nesse conjunto de dados resultante.

3 Primeira Abordagem - CNN (2 dimensões)

Inicialmente, de modo a ter um modelo e resultado de referência, decidímos utilizar apenas as matrizes de conectividade e a *feature* da idade (que é o objetivo do problema), descartando os outros dados. Deste modo, utilizando uma rede neuronal convolucional baseada na arquitetura **LeNet-5** [1], podemos obter um resultado para utilizar como referência.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 88, 88, 64)        640
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 44, 44, 64)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 42, 42, 64)        36928
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 21, 21, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 19, 19, 64)        36928
_________________________________________________________________
flatten (Flatten)            (None, 23104)             0
_________________________________________________________________
dense (Dense)                (None, 128)               2957440
_________________________________________________________________
dense_1 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 65
=================================================================
Total params: 3,040,257
Trainable params: 3,040,257
Non-trainable params: 0
_________________________________________________________________
```

Figura 3.1: CNN utilizada

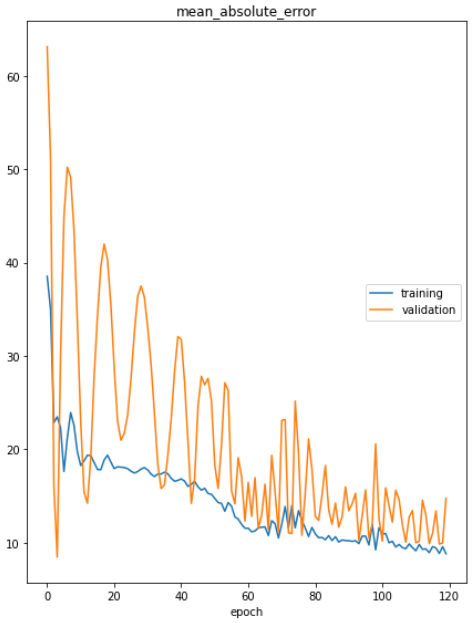

Figura 3.2: Resultados

80% dos dados foram utilizados para treino e os restantes 20% para validação. O modelo foi compilado com o otimizador *Adam* (learning rate = 0.001) e erro absoluto médio como função de *loss*. Foram realizadas 120 epochs com batch size igual a 64. Foi utilizado o *Plot Losses callback* para gerar um gráfico que demonstre a evolução do treino do modelo. Os resultados obtidos foram os seguintes:

```
mean_absolute_error
training           (min:    8.815, max:   38.527, cur:    8.815)
validation         (min:    8.464, max:   63.114, cur:   14.743)
```

A submissão no kaggle deu 7.11111 de public score.

4 Segunda Abordagem - CNN (2 dimensões) seguida de MLP

Numa segunda abordagem, o principal objetivo foi utilizar todos os dados existentes de modo a obter melhores resultados. Assim, continuamos a aplicar uma rede neuronal convolucional às matrizes de conectividade. De seguida, fornecendo este resultado, juntamente com os restantes dados do ficheiro *.csv*, nomeadamente das *features sex* e *education*, a uma *multi perceptron neural network*, obtemos um segundo modelo que é capaz de prever a idade do cérebro mais eficazmente.

Model: "model_1"

Layer (type)	Output Shape	Param #	Connected to
input_8 (InputLayer)	[(None, 90, 90, 1)]	0	
conv2d_10 (Conv2D)	(None, 88, 88, 256)	2560	input_8[0][0]
activation_9 (Activation)	(None, 88, 88, 256)	0	conv2d_10[0][0]
max_pooling2d_9 (MaxPooling2D)	(None, 44, 44, 256)	0	activation_9[0][0]
conv2d_11 (Conv2D)	(None, 42, 42, 256)	590080	max_pooling2d_9[0][0]
activation_10 (Activation)	(None, 42, 42, 256)	0	conv2d_11[0][0]
max_pooling2d_10 (MaxPooling2D)	(None, 21, 21, 256)	0	activation_10[0][0]
conv2d_12 (Conv2D)	(None, 19, 19, 256)	590080	max_pooling2d_10[0][0]
activation_11 (Activation)	(None, 19, 19, 256)	0	conv2d_12[0][0]
max_pooling2d_11 (MaxPooling2D)	(None, 9, 9, 256)	0	activation_11[0][0]
input_9 (InputLayer)	[(None, 2)]	0	
flatten_7 (Flatten)	(None, 20736)	0	max_pooling2d_11[0][0]
flatten_6 (Flatten)	(None, 2)	0	input_9[0][0]
concatenate_2 (Concatenate)	(None, 20738)	0	flatten_7[0][0] flatten_6[0][0]
dense_8 (Dense)	(None, 256)	5309184	concatenate_2[0][0]
dense_9 (Dense)	(None, 256)	65792	dense_8[0][0]
dense_10 (Dense)	(None, 256)	65792	dense_9[0][0]
dense_11 (Dense)	(None, 1)	257	dense_10[0][0]

Total params: 6,623,745
Trainable params: 6,623,745
Non-trainable params: 0

Figura 4.1: Rede neuronal utilizada nesta abordagem

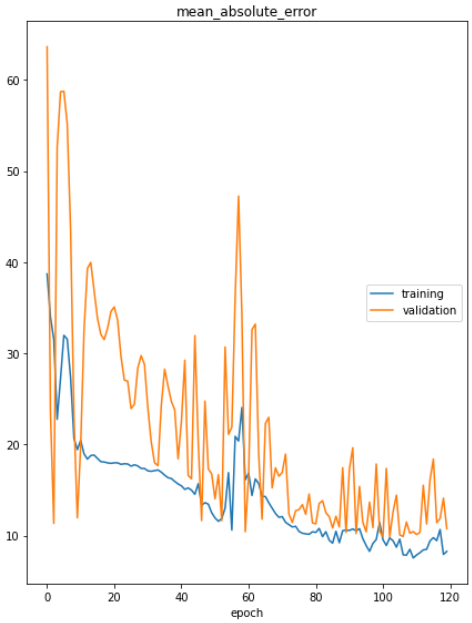

Figura 4.2: Resultados

Novamente, 80% dos dados foram utilizados para treino e os restantes 20% para validação. O modelo foi compilado com o otimizador *Adam* (learning rate = 0.001) e erro absoluto médio como função de *loss*. Foram realizadas 120 epochs com batch size igual a 64. Os resultados obtidos foram os seguintes:

```
mean_absolute_error
training           (min:    7.549, max:   38.700, cur:    8.269)
validation         (min:    9.831, max:   63.621, cur:   10.756)
```

Podemos verificar que o erro absoluto médio baixou significativamente para os casos de validação. A mesma tendência se verificou na submissão no kaggle, que deu 6.43326 de public score.

5 Terceira abordagem - CNN (1 dimensão) seguida de MLP

Após alguma reflexão, apercebemo-nos que interpretar estas matrizes de conectividade como imagens e utilizar camadas convolucionais de duas dimensões não faz muito sentido, uma vez que, as imagens geradas na visualização destas matrizes são apenas as suas projeções. Assim, nesta terceira abordagem, decidimos passar a utilizar camadas convolucionais de uma só dimensão.

```
Model: "model"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_2 (InputLayer)            [(None, 90, 90)]     0
__________________________________________________________________________________________________
conv1d_1 (Conv1D)               (None, 88, 256)      69376       input_2[0][0]
__________________________________________________________________________________________________
activation_1 (Activation)       (None, 88, 256)      0           conv1d_1[0][0]
__________________________________________________________________________________________________
max_pooling1d_1 (MaxPooling1D)  (None, 44, 256)      0           activation_1[0][0]
__________________________________________________________________________________________________
conv1d_2 (Conv1D)               (None, 42, 256)      196864      max_pooling1d_1[0][0]
__________________________________________________________________________________________________
activation_2 (Activation)       (None, 42, 256)      0           conv1d_2[0][0]
__________________________________________________________________________________________________
max_pooling1d_2 (MaxPooling1D)  (None, 21, 256)      0           activation_2[0][0]
__________________________________________________________________________________________________
conv1d_3 (Conv1D)               (None, 19, 256)      196864      max_pooling1d_2[0][0]
__________________________________________________________________________________________________
activation_3 (Activation)       (None, 19, 256)      0           conv1d_3[0][0]
__________________________________________________________________________________________________
max_pooling1d_3 (MaxPooling1D)  (None, 9, 256)       0           activation_3[0][0]
__________________________________________________________________________________________________
input_3 (InputLayer)            [(None, 2)]          0
__________________________________________________________________________________________________
flatten_1 (Flatten)             (None, 2304)         0           max_pooling1d_3[0][0]
__________________________________________________________________________________________________
flatten (Flatten)               (None, 2)            0           input_3[0][0]
__________________________________________________________________________________________________
concatenate (Concatenate)       (None, 2306)         0           flatten_1[0][0]
                                                                 flatten[0][0]
__________________________________________________________________________________________________
dense (Dense)                   (None, 256)          590592      concatenate[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 256)          65792       dense[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 256)          65792       dense_1[0][0]
__________________________________________________________________________________________________
dense_3 (Dense)                 (None, 1)            257         dense_2[0][0]
==================================================================================================
Total params: 1,185,537
Trainable params: 1,185,537
Non-trainable params: 0
__________________________________________________________________________________________________
```

Figura 5.1: Rede neuronal utilizada nesta abordagem

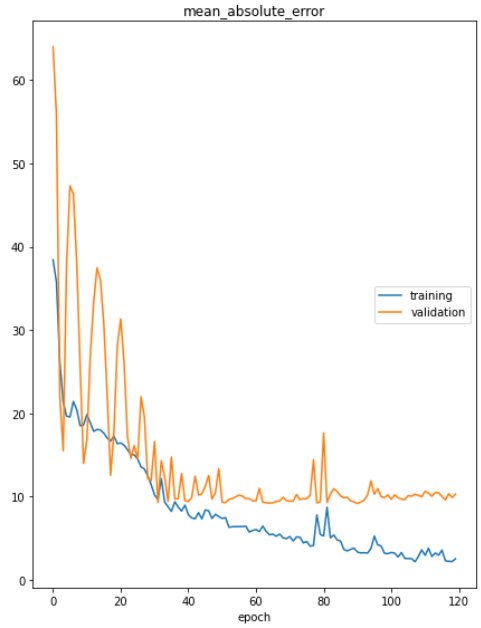

Figura 5.2: Resultados

Mais uma vez, 80% dos dados foram utilizados para treino e os restantes 20% para validação. O modelo foi compilado com o otimizador *Adam* (learning rate = 0.001) e erro absoluto médio como função de *loss*. Foram realizadas 120 epochs com batch size igual a 64. Os resultados obtidos foram os seguintes:

```
    mean_absolute_error
training          (min:    2.185, max:   38.430, cur:    2.541)
validation        (min:    9.180, max:   63.993, cur:   10.303)
```

Podemos verificar que não houve uma melhoria significativa do erro absoluto médio. Aliás, a maior diferença entre esta abordagem e a abordagem anterior é que o modelo está a dar *overfitting* extremo, como é visível pelo facto da curva dos dados de validação estar constantemente acima da curva dos dados de treino, significando que o erro absoluto médio é superior em dados que não foram utilizados para treinar a rede. Após submissão das previsões desta rede no kaggle, obteu-se 6.91853 de public score, sendo este resultado ligeiramente pior que o anterior.

Nas seguintes secções iremos explicar, em detalhe, todas as técnicas e decisões que tomámos para controlar o *overfitting*, e iremos demonstrar o resultado final com mais algumas otimizações resultantes de vários testes.

6 Controlo do Overfitting

Modelos com *overfitting* dificilmente serão úteis com dados reais, uma vez que não conseguirão fazer previsões precisas e acertadas para dados com os quais não treinou. Assim, é extremamente importante afinar a rede de modo a obter um modelo com um equilíbrio entre o menor erro absoluto médio possível e que não esteja a dar *overfitting* nem *underfitting*.

6.1 Divisão dos dados

Quando um modelo está a dar *overfitting*, então é porque se está a adaptar demasiado aos casos de treino, ajustando os seus pesos para minimizar o erro nesses casos. Ora, o dataset é extremamente reduzido, com apenas 112 entradas. Até agora, foram utilizados 20% destes casos para validação, o que corresponde a, aproximadamente, 22 casos de validação. Em teoria, aumentar os casos de treino dificultará o ajuste do modelo para os mesmos, podendo ajudar a reduzir o overfitting e melhorar os resultados obtidos. Deste

modo, o grupo passou a utilizar cerca de 10% do dataset para validação (11 casos). Por último, o grupo teve o cuidado de garantir que estes casos de validação cobriam todos os intervalos de idades, de modo a assegurar uma adaptação uniforme da rede a todas as idades possíveis.

6.2 Batch Normalization

Antes de explicar o que a *batch normalization* faz é necessário entender o objetivo da normalização dos dados. Normalização é uma técnica de pré-processamento dos dados para estandardizar os dados, ou seja, colocar todos os dados dentro do mesmo intervalo de valores. Não normalizar os dados antes do treino da rede pode causar vários problemas tais como dificultar a aprendizagem e piorar a velocidade de treino. Neste caso, os dados utilizados já estavam previamente normalizados.

Batch normalization é uma técnica de normalização feita entre as camadas de uma rede neuronal, em vez dos dados de *input*. É aplicada ao longo dos *mini-batches* em vez do dataset inteiro. Esta técnica procura acelerar o treino e permitir utilizar *learning rates* mais altas.

A fórmula da normalização realizada pela camada de *Batch normalization* pode ser definida como:

$$z^N = (\frac{z - m_z}{s_z}) \tag{1}$$

Onde z^N é o *output* da camada de *Batch Normalization*, m_z a média dos *outputs* dos neurónios e s_z o desvio padrão do *output* dos neurónios.

No artigo original de Sergey et al. [2], foi verificado que *Batch normalization* reduz o deslocamento covariável interno da rede, ou seja, reduz mudanças na distribuição de dados no *input* das camadas internas de uma rede neuronal, ao garantir que a média e desvio padrão dos *inputs* se mantém constante, o que facilita o processo de treino.

Por último, *Batch normalization* também tem um efeito de regularização. Uma vez que este efeito de normalização é aplicado a *batches* e não ao dataset inteiro, os dados que o modelo vê ao longo do treino vão ter algum ruído, causando o tal efeito de regularização, que pode ajudar a ultrapassar o overfitting e obter melhores resultados. No entanto, como iremos ver posteriormente, o ruído adicionado por esta técnica é bastante reduzido, pelo que é comum combinar *Batch normalization* com outras técnicas para obter uma melhor regularização.

Assim, o grupo adicionou camadas de *Batch normalization* entre as várias camadas convolucionais, com um *momentum* de 0.95.

6.3 Regularização (Ridge)

Regularização refere-se a técnicas que são utilizadas para calibrar modelos de *machine learning* de modo a minimizar a função de *loss* e evitar *overfitting* e *underfitting*. Estas técnicas penalizam modelos mais complexos.

Existem dois tipos principais de técnicas de regularização: Regularização *Ridge* e regularização *Lasso*. O nosso grupo decidiu utilizar a regularização *Ridge* uma vez que é mais comum e estável. Esta técnica penaliza modelos complexos ao reduzir a magnitude dos coeficientes do modelo.

O grupo decidiu adicionar regularização *Ridge* (L2) às camadas convolucionais de uma dimensão com *learning rate* de 0.001.

6.4 Inicialização dos pesos

A inicialização dos pesos define os pesos para todos os neurónios da rede pela primeira vez, antes do processo do treino. Escolher os pesos corretos é crucial porque queremos obter o valor mais próximo possível do mínimo da função de custo no menor intervalo de tempo possível. Uma vez que estamos a utilizar a função de ativação *Relu*, decidimos adicionar a heurística **He-et-al Initialization** à primeira camada da rede.

6.5 Regularização com Dropout

Dropout é outro tipo de regularização que ignora um subconjunto aleatório de unidades numa camada, ao definir os seus pesos para 0 durante o processo de treino. Decidímos adicionar camadas de dropout entre as várias camadas intermédias da rede com uma probabilidade de 0.2 em cada uma.

6.6 Restrições dos Pesos

O objetivo das restrições dos pesos é verificar o tamanho dos pesos da rede durante o processo de treino e reajustá-los, caso o seu tamanho exceda um limite pré-definido. Decidímos utilizar a restrição *unit_norm* que força os pesos a ter uma magnitude de 1.0.

6.7 Redução da complexidade da rede

Apesar da aplicação de todas estas técnicas ter ajudado a reduzir o *overfitting*, foi necessário reduzir a complexidade da rede. Assim, ao diminuir a profundidade da rede (remover camadas) e, em compensação, aumentar o número de unidades em cada camada (aumentar a largura da rede), foi possível ultrapassar por completo o *overfitting*.

7 Mais otimizações

7.1 Batch-size

Até agora, os treinos foram realizados com um *batch* size relativamente grande (64). Tendo em conta que o tamanho do *dataset* é 112, isto resultaria em 2 *batches* por *epoch*. Ora, a desvantagem de um baixo número de *batches* é que a rede vai reajustar os seus parâmetros menos vezes, o que pode piorar os resultados. No entanto, utilizar *batches* maiores pode acelerar o treino. Assim, após vários testes, o grupo encontrou o ponto de equilíbro para este hyper-parâmetro, sendo ele *batch size* igual a 4, resultando em 28 *batches* por *epoch*. Uma otimização que poderia ser feita aos *batches*, seria garantir que os casos que fazem parte de cada *batch* são parecidos, isto é, as idades desses casos são semelhantes.

7.2 Learning-rate

Ao analisar os gráficos de treino, o grupo apercebeu-se que poderia beneficiar de otimizar o hyper-parâmetro *learning rate*. Até agora, foi utilizado o otimizador *Adam* com *learning rate* de 0.001. Um *learning rate* demasiado alto pode resultar num modelo que converge muito rapidamente para uma solução não ótima (por exemplo, um mínimo local), enquanto que um *learning rate* demasiado pequeno pode causar o modelo ficar preso numa solução também não ótima, para além de poder desacelerar o processo de treino. Após vários testes, o grupo chegou ao valor de *learning rate* igual a 0.00005.

7.3 LeakyRELU

Por último, o grupo decidiu mudar as funções de ativação de *relu* para *LeakyRELU*. Esta função de ativação evita o problema de *dying ReLU* [3], que acontece quando a função de ativação *ReLU* recebe sempre valores negativos, transformando-os sempre em 0.

7.4 Rede final

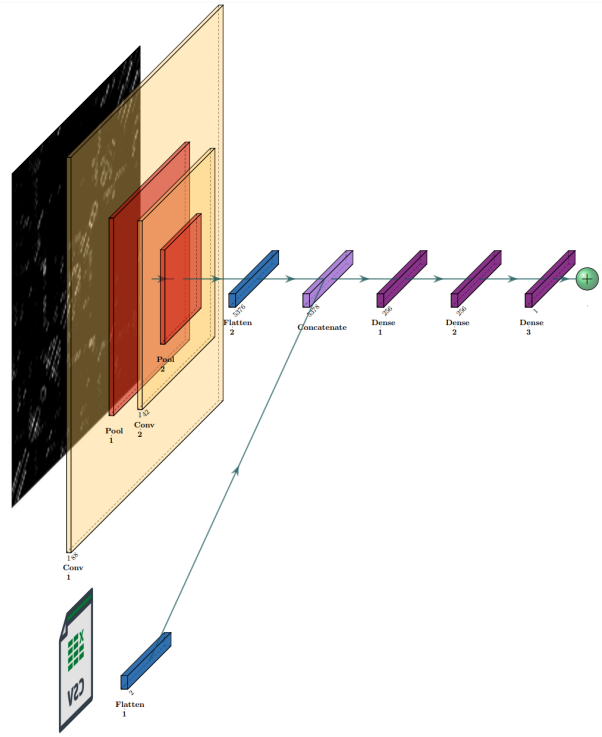

Figura 7.1: Diagrama da rede final

Model: "model"

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 90, 90)]	0	
conv1d (Conv1D)	(None, 88, 256)	69376	input_1[0][0]
activation (Activation)	(None, 88, 256)	0	conv1d[0][0]
batch_normalization (BatchNorma	(None, 88, 256)	1024	activation[0][0]
max_pooling1d (MaxPooling1D)	(None, 44, 256)	0	batch_normalization[0][0]
dropout (Dropout)	(None, 44, 256)	0	max_pooling1d[0][0]
conv1d_1 (Conv1D)	(None, 42, 256)	196864	dropout[0][0]
activation_1 (Activation)	(None, 42, 256)	0	conv1d_1[0][0]
max_pooling1d_1 (MaxPooling1D)	(None, 21, 256)	0	activation_1[0][0]
dropout_1 (Dropout)	(None, 21, 256)	0	max_pooling1d_1[0][0]
batch_normalization_1 (BatchNor	(None, 21, 256)	1024	dropout_1[0][0]
input_2 (InputLayer)	[(None, 2)]	0	
flatten_1 (Flatten)	(None, 5376)	0	batch_normalization_1[0][0]
flatten (Flatten)	(None, 2)	0	input_2[0][0]
concatenate (Concatenate)	(None, 5378)	0	flatten_1[0][0] flatten[0][0]
dense (Dense)	(None, 256)	1377024	concatenate[0][0]
dropout_2 (Dropout)	(None, 256)	0	dense[0][0]
dense_1 (Dense)	(None, 256)	65792	dropout_2[0][0]
dropout_3 (Dropout)	(None, 256)	0	dense_1[0][0]
dense_2 (Dense)	(None, 256)	65792	dropout_3[0][0]
dropout_4 (Dropout)	(None, 256)	0	dense_2[0][0]
dense_3 (Dense)	(None, 1)	257	dropout_4[0][0]

Total params: 1,777,153
Trainable params: 1,776,129
Non-trainable params: 1,024

Figura 7.2: Modelo final

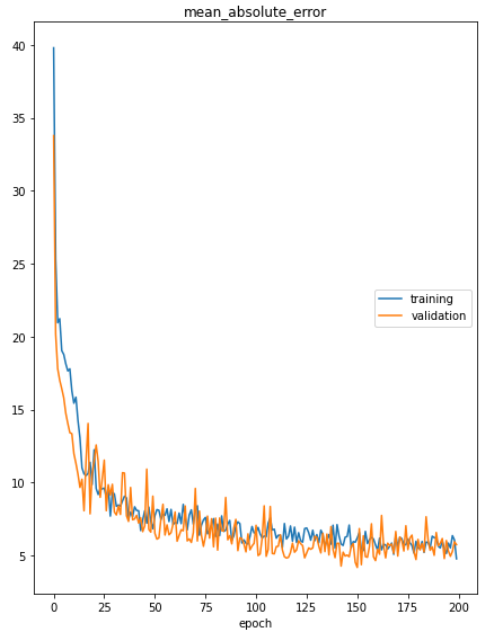

Figura 7.3: Treino do modelo final

Ao contrário das abordagens anteriores, desta vez utilizámos 200 epochs. Como podemos observar, após todas estas otimizações, as curvas de validação e treino estão praticamente sobrepostas, o que demonstra que o modelo não está a dar *overfitting*, nem *underfitting*

```
mean_absolute_error
```

```
training            (min:    4.784, max:   39.790, cur:    4.784)
validation          (min:    4.189, max:   33.773, cur:    5.754)
```

Também podemos observar a melhoria que houve com as otimizações referidas acima, obtendo-se valores de erro absoluto médio muito menores aos valores obtidos anteriormente.

Kaggle 2.35776

8 Casos mais difíceis de prever

De modo a verificar quais os casos em que o modelo tem maior dificuldade de prever, aplicámos o modelo aos casos de treino e calculámos o erro absoluto entre a idade prevista e a idade real do cérebro. De seguida, gerámos o seguinte gráfico que representa os 10 casos com maior erro absoluto:

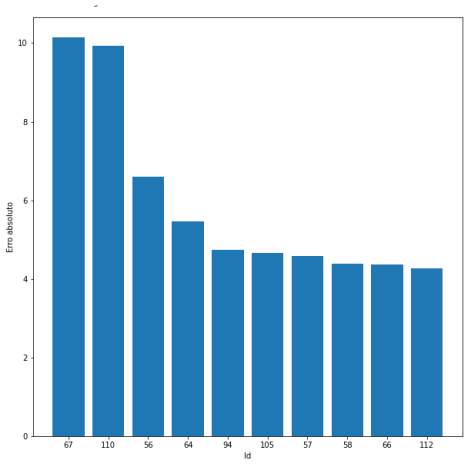

Figura 8.1: Casos com maior erro absoluto

Como podemos observar, os casos em que o modelo tem maior dificuldade para prever a idade (maior erro absoluto), são os casos com os Ids 67, 110, 56 e 64. Organizando a informação de outra forma:

Id	Idade Real	Idade Prevista	Diferença
67	57	67	+10
110	55	65	+10
56	61	67	+6
64	62	68	+6

A questão a colocar aqui é a seguinte: como interpretar estes casos? Se o modelo, ao analisar a matriz de conectividade deste cérebros, infere uma idade bastante superior à idade da pessoa, então pode significar que a mesma sofre de doenças neurodegenerativas tais como: Alzheimer, Parkinson, Esclerose múltipla, entre outras... Estas doenças causam um processo irreversível que resulta em degeneração e/ou morte progressiva de células nervosas, pelo que seria recomendado proceder a um diagnóstico dos cérebros destas pessoas, de modo a identificar e tratar estas possíveis doenças o mais brevemente possível.

Outra observação que se pode fazer destes resultados é que, as pessoas dos casos em que o modelo não foi capaz de prever a idade com grande precisão, têm idades semelhantes e, segundo Yujun et al. [4], suscetíveis ao aparecimento destas mesmas doenças.

No entanto, é necessário notar que, como os dados de treino foram bastante limitados, estes podem ser simplesmente casos muito diferentes da maior parte dos casos conhecidos pelo modelo, não tendo conhecimento suficiente para efetuar uma previsão da idade.

9 Conexões cerebrais mais relevantes para a previsão da idade do cérebro

9.1 Predict com cada conexão cerebral

O primeiro método que utilizámos para avaliar quais as conexões cerebrais mais importantes para a previsão da idade do cérebro foi fornecer à rede neuronal convolucional *input* com todas as variáveis igual a 0 menos a variável correspondente à conexão cerebral que queremos avaliar. Comparando o erro relativo do resultado entre as várias conexões, é possível compreender quais as conexões que são mais úteis para esta previsão (Quanto menor for o erro relativo, mais útil essa conexão é para a previsão da idade).

De seguida, estão presentes, em cada gráfico, as 10 conexões cerebrais mais úteis para 4 idades diferentes: 13, 31, 53 e 71, respetivamente.

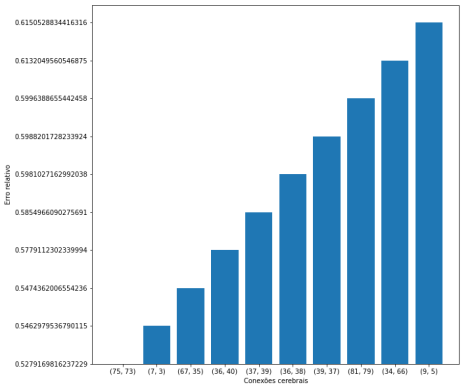

Figura 9.1: Erro relativo das conexões cerebrais para cérebro com 13 anos de idade

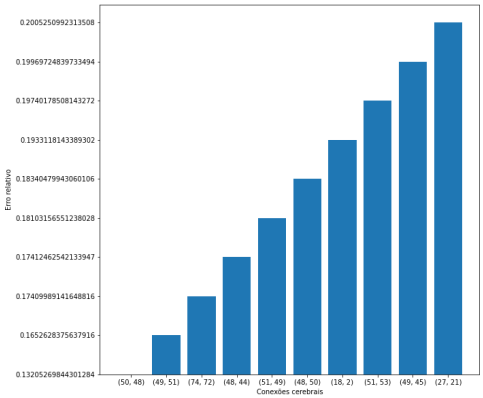

Figura 9.2: Erro relativo das conexões cerebrais para cérebro com 31 anos de idade

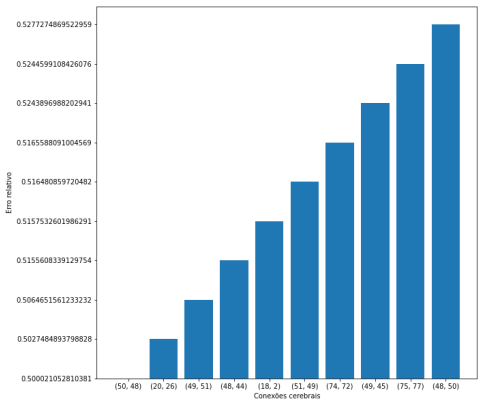

Figura 9.3: Erro relativo das conexões cerebrais para cérebro com 53 anos de idade

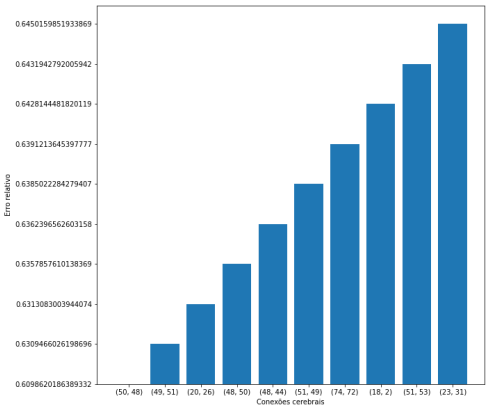

Figura 9.4: Erro relativo das conexões cerebrais para cérebro com 71 anos de idade

Analisando os gráficos, é possível retirar várias conclusões. Primeiramente, é visível que, dependendo da idade do cérebro, as conexões cerebrais mais relevantes para a previsão são diferentes. Por exemplo, a conexão cerebral mais útil para prever a idade igual a 13 anos foi a conexão (75,73), seguida das conexões (7,3) e (67, 35). Ora, estas conexões não estão presentes nos gráficos dos restantes cérebros, o que demonstra que a interpretação de uma matriz de conectividade num cérebro adulto deve ser feita de modo diferente do que de num cérebro jovem.

Para além disso, podemos verificar que, nos cérebros adultos (31, 53 e 71 anos de idade), as conexões mais úteis para a previsão da idade são bastante semelhantes. Por exemplo, nestes 3 cérebros, as conexões (50, 48), (49,51), (48,50), (48,44), (51, 49), (74,72) e (18,2) fazem parte das conexões cerebrais com

menor erro relativo. Isto é deveras interessante pois demonstra que o modo de interpretação de matrizes de conectividade de cérebros adultos não varia muito.

Esta abordagem é, no entanto, ingénua, na medida em que não tem em conta casos em que conjuntos de variáveis, isto é, conjuntos de conexões cerebrais são importantes para a previsão da idade do cérebro.

9.2 Shap

SHAP (SHapley Additive exPlanations) é uma abordagem baseada em teoria de jogos para explicar o output de qualquer modelo de machine learning. Combina alocação de crédito ótima com explicações locais, ao utilizar os clássicos valores **Shapley** da teoria de jogos e as suas extensões (ver os papers [5, 6]).

Assim, ao contrário da abordagem anterior, fomos capazes de avaliar o impacto das diferentes conexões cerebrais tendo em conta possíveis conjuntos de conexões que, separadamente, não têm grande influência na previsão da idade do cérebro. Para além disso, este método foi aplicado ao conjunto de dados de validação, em vez de ser aplicado a um caso de cada vez como na primeira abordagem, permitindo obter melhores resultados.

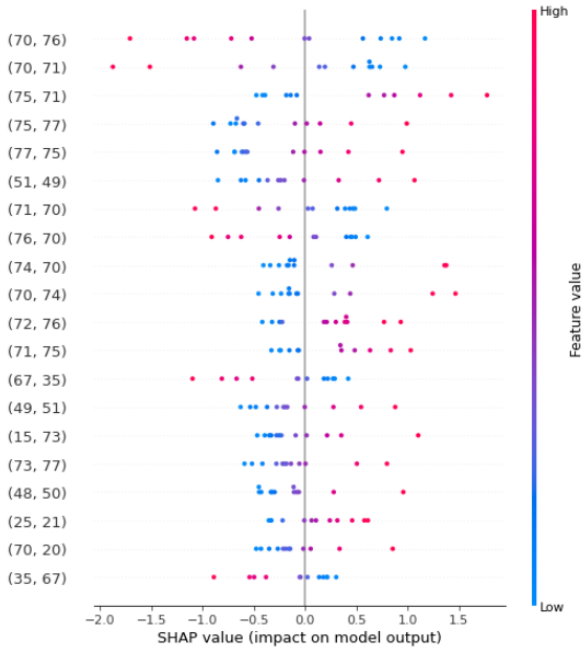

Figura 9.5: Resultado do SHAP

Como podemos observar, para cada conexão cerebral temos um conjunto de pontos distribuídos horizontalmente. Cada ponto corresponde a um caso dos dados de validação utilizados pelo Shap para realizar os cálculos. A cor do ponto representa a utilidade dessa conexão para a previsão da idade do cérebro, sendo a cor azul pior e a cor avermelhada melhor.

Podemos verificar que estão presentes algumas conexões cerebrais que foram consideradas muito úteis na abordagem anterior, tais como as conexões (48,50) e (51,49). No entanto, temos presentes muitas conexões que não apareceram nos resultados da abordagem anterior, que acreditamos se dever ao facto destas conexões (isoladas) não contribuírem muito para a previsão da idade, pelo que a técnica anterior não foi capaz de se aperceber da sua relevância.

10 Conclusão

Neste projeto, pretendia-se aplicar um modelo de Deep Learning a matrizes de conectividade obtidas a partir de ressonâncias magnéticas de difução, de modo a prever a idade do cérebro. Para além disso, iríamos avaliar e interpretar os resultados obtidos com várias ferramentas e técnicas distintas.

Na primeira fase, acreditámos ter conseguido criar um bom modelo de Deep Learning, não só capaz de prever a idade do cérebro nos casos utilizados para treino, mas também capaz de generalizar, na medida em que é capaz de realizar previsões com precisão com dados reais e desconhecidos ao modelo. Esta afirmação pode ser corroborada pela boa classificação que o grupo obteve na competição na plataforma Kaggle, obtendo o 2º lugar, com *private score* de 4.08960 (valor semelhante ao erro absoluto médio obtido durante o treino da rede).

Na parte de análise dos resultados acreditámos que os casos em que o modelo tem maior erro ao prever a idade do cérebro são possíveis pacientes que sofram de alguma doença neurodegenerativa, o que explica a diferença entre a idade prevista e a idade cronológica. Deste modo, este modelo de Deep Learning poderia ser utilizado para auxiliar na identificação destes casos e, possivelmente, ajudar na prevenção e tratamento antecipado destas doenças.

Para além disso, os diferentes métodos de avaliação da importância de *features*, neste caso das conexões cerebrais, possibilitaram-nos entender quais as partes do cérebro nas quais o modelo se baseia mais para realizar a previsão da idade. Estas informações talvez possam auxiliar os profissionais de saúde a detetar anomalias nestas matrizes de conectividade.

Em suma, acreditamos ter alcançado todos os objetivos pretendidos, uma vez que obtemos um modelo de Deep Learning capaz de realizar previsões da idade cerebral com um erro relativamente baixo e porque conseguimos retirar conclusões coerentes e úteis dos resultados obtidos.

Referências

[1] Richmond Alake (2020), Understanding and Implementing LeNet-5 CNN Architecture (Deep Learning).

[2] Sergey Ioffe and Christian Szegedy (2015), Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift

[3] Lu, Lu & Shin, Yeonjong & Su, Yanhui & Karniadakis, George. (2020). Dying ReLU and Initialization: Theory and Numerical Examples

[4] Hou, Yujun & Dan, Xiuli & Babbar, Mansi & Wei, Yong & Hasselbalch, Steen & Croteau, Deborah & Bohr, Vilhelm. (2019). Ageing as a risk factor for neurodegenerative disease

[5] LIME: Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin. "Why should i trust you?: Explaining the predictions of any classifier."Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2016.

[6] Shapley sampling values: Strumbelj, Erik, and Igor Kononenko. "Explaining prediction models and individual predictions with feature contributions."Knowledge and information systems 41.3 (2014): 647-665.