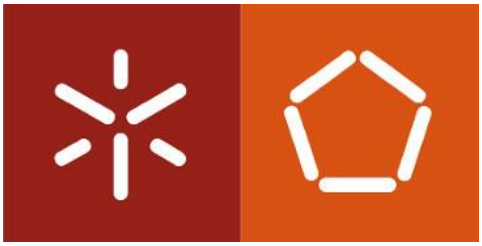
UNIVERSIDADE DO MINHO



Brain Age Prediction Challenge using diffusion MRI structural connectivity features

Mestrado Integrado em Engenharia Biomédica

Imagiologia
(2º Semestre/ Ano Letivo 2021/2022)

Docente: Víctor Alves

Trabalho realizado por:
A88362 Lara Vaz
A88360 Mariana Carvalho
A88397 Tiago Novais

Braga
6 de junho de 2022

Índice

1. Metodologia

O objetivo deste trabalho é desenvolver e otimizar um modelo de aprendizagem profunda capaz de prever a idade do cérebro a partir de características de conetividade estrutural. De modo a atingir este objetivo, diferentes modelos foram desenvolvidos, destacando-se os seguintes:

1. Modelo com CNN 2D, cujo input das *layers* convolucionais era o tensor contendo as 112 matrizes 2D e o input do MLP era a concatenação desse tensor com o tensor que continha as colunas do *sex* e da *education*;
2. Modelo com MLP 2D, cujo input era um tensor que apresentava as 112 matrizes 2D concatenadas com as colunas do *sex* e da *education*, para cada um dos pacientes;
3. Modelo com CNN 1D, cujo input das *layers* convolucionais era o tensor contendo os 112 vetores 1D[1] e o input do MLP era a concatenação desse tensor com o tensor que continha as colunas do *sex* e da *education*;
4. Modelo com MLP 1D, cujo input era um tensor que apresentava os 112 vetores 1D concatenados com as colunas do *sex* e da *education*, para cada um dos pacientes;
5. Modelo com *FullyConnectedNet* do *Monai*, cujo input era um tensor que apresentava os 112 vetores 1D concatenados com as colunas do *sex* e da *education*, para cada um dos pacientes;
6. Modelo com CNN 1D, cujo input era o tensor contendo os 112 vetores 1D;
7. Modelo com MLP 1D, cujo input era um tensor que apresentava os 112 vetores 1D;
8. Modelo com *FullyConnectedNet* do *Monai*, cujo input era um tensor que apresentava os 112 vetores 1D.

Entre todos esses modelos, os modelos 1 e 2 foram descartados numa fase inicial, pois, após terem sido efetuadas várias tentativas, os resultados não eram satisfatórios, pois os modelos apresentavam uma incapacidade em aprender o dataset, provavelmente devido ao facto das matrizes serem muito esparsas.

Assim sendo, o grupo focou-se bastante nos modelos 3 a 5. Contudo, durante o desenvolvimento destes, verificou-se os modelos estavam sempre a prever valores perto da média das idades, devido à inexistência de relação entre os atributos e o *label*. De forma a confirmar se o problema seria mesmo esse, o grupo desenvolveu os modelos 6 a 8, semelhantes aos modelos 3 a 5, mas apenas com os

[1] A obtenção destes vetores 1D será explicada na secção 2.1.

vetores 1D como input, o que eliminaria qualquer problema associado à concatenação destes dados com os dos outros atributos. Com os modelos 6 a 8 obtiveram-se resultados muito melhores, no sentido em que os modelos estavam a generalizar melhor em vez de tender para a média das idades, apresentando também valores de *loss* muito inferiores. Assim sendo, de facto estava comprovada a existência de um problema associado à troca de posições. Para identificar a origem deste problema foram efetuados uma série de *prints* aos vetores e matrizes, aos tensores, *dataloaders* e ao input das redes e concluiu-se que o problema ocorria devido à junção dos diferentes tensores no *TensorDataset*, em que para cada um dos 112 pacientes se juntavam dados do respetivo tensor com os dados da matriz, do tensor com os dados do *sex* e *education* e do tensor com o *label* da idade. Depois, ao fazer o *shuffle* dos dados no *dataloader*, os 3 tensores apareciam trocados para os vários pacientes. De forma a solucionar este problema, procedeu-se à junção dos vetores 1D com as restantes *features sex* e *education* num único *dataframe*. A partir deste, os dados de input foram convertidos num único tensor, evitando possíveis trocas que pudessem ocorrer.

Tendo esse problema sido resolvido, o grupo voltou a focar-se nos modelos 3 a 5, por conter também os atributos do sexo e da educação, que seria de esperar que fossem importantes para o treino do modelo. Após várias tentativas de melhoramento de cada um destes, com diversas alterações da rede e dos hiperparâmetros, verificou-se que o melhor modelo, com maior generalização e menor *loss* era o modelo 5. Contudo, o modelo 5 apresentava claramente um pior desempenho quando comparado com o modelo 8, semelhante ao modelo 5, mas que apenas recebia como input os vetores 1D. Por este motivo, o modelo utilizado para as submissões foi o modelo 8. Apesar disso, será também abordado todo o pré-processamento efetuado ao modelo 5, por ser mais completo, e revelar mais tentativas efetuadas pelo grupo para alcançar melhores resultados.

2. Exploração do dataset

2.1. Explicação detalhada

O dataset fornecido consiste em dados obtidos a partir de 140 pacientes, sendo que 112 estão separados para treino e 28 para teste. Para os pacientes de treino é fornecida a sua idade (*age)*, já para os de teste não, pois o objetivo é prevê-la. Para cada paciente é fornecido também o seu nível de educação (*education*), que varia de 0 a 20 anos e o seu sexo (*sex:* 1 – feminino ou 0 - masculino). Além destes dados, cada paciente tem a si associado uma matriz de conetividade cerebral, obtida a partir da Ressonância Magnética de difusão. As matrizes têm dimensão 90x90, contendo um total de

8100 valores, e permitem averiguar as conetividades existentes entre 90 regiões do cérebro. Os valores desta matriz variam entre 0 e 1. Deste modo, para cada par de regiões cerebrais, a matriz indica qual a probabilidade de conexão entre essas 2 regiões. É também importante realçar que as matrizes são simétricas, pois como mostram a conetividade entre cada par das 90 regiões cerebrais, então, a conetividade entre as regiões x e y é igual à conetividade entre as regiões y e x.

2.2. Pré-processamento

Como referido anteriormente, as matrizes de conetividade cerebral eram muito esparsas e simétricas, e por isso, foi primeiramente efetuado um tratamento de forma a obter apenas os valores da matriz correspondentes à matriz triangular superior exceto a diagonal, eliminando a redundância dos valores. Excluiu-se também a diagonal, porque obviamente a conexão entre uma região com ela própria é sempre 1. Após ter sido efetuado este processo, dos 8100 valores inicialmente existentes em cada matriz, obtiveram-se apenas 4005 (90*90/2 – 90/2). Posteriormente, adicionou-se a um novo *dataframe* as colunas que não verificavam a seguinte condição: ter sempre o valor 0 para todos os pacientes. Este processo foi efetuado quer para os dados de treino quer para os dados de teste. O resultado foram 2 novos ficheiros CSV com a seguinte estrutura:

- "train_dataset.csv": contém 112 linhas (pacientes) e 1206 colunas (*features)*, uma com o id do paciente, o *label* da idade, o respetivo *sex* e *education* e 1202 colunas vindas da matriz e que não verificavam a condição acima mencionada;
- "test_dataset.csv": contém 28 linhas e 1205 colunas, uma com o id do paciente, o respetivo *sex* e *education* e 1202 colunas vindas da matriz e que não verificavam a condição acima mencionada.

Nestes 2 datasets, para os valores de conetividade dos vetores também foi efetuada uma normalização por coluna. Isto porque, apesar dos valores dos vetores 1D já estarem entre 0 e 1, verificou-se que existiam conexões com valores muito baixos relativamente às outras, o que faz com que, ao modelar, esses valores comparados com os mais altos tendessem para 0. Deste modo, para esses valores mais baixos não haveria discriminação entre os sujeitos. Relativamente ao atributo do id, este foi obviamente desconsiderado por não apresentar nenhuma relação nem com os dados de input nem com o *label.* Todo esse tratamento acima referido foi o efetuado para o modelo 8.
Para o modelo 5, além deste pré-processamento, foram ainda executados outros tratamentos para os diferentes atributos dos datasets "train_dataset.csv" e "test_dataset.csv". Para o atributo *education,*

foi efetuada uma normalização dos seus valores entre 0 e 1. Para o atributo *sex,* foi efetuado *One Hot Encoding*, uma vez que este se trata de uma variável categórica e neste caso binária. Assim sendo, a partir da coluna *sex*¸ foram obtidas 2 novas colunas, uma para cada um dos valores disponíveis (sexo 1 ou 0). No fim, foi efetuado o *drop* da coluna de *sex* original.

2.3. Preparação dos *dataloaders* e do dicionário

Após se ter concluído todo o pré-processamento, procedeu-se à construção de uma lista com 112 dicionários, cuja chave é o nome da *feature* a que corresponde (1:3, 88:89, …). Cada dicionário apresenta um total de 1202 chaves, pois esse é o número de atributos. É de realçar que as *features* do tipo x_y representam a conetividade cerebral entre a região x e a região y. Esta lista de dicionários foi muito importante, para que no final fosse possível proceder a um estudo de *feature importance.*

Posteriormente, os dados de treino e de teste foram guardados em 2 *TensorDataset* diferentes, o *train* que contém o tensor com os dados de input de treino e o tensor com o *label* para os 112 pacientes e o *test* que contém um único tensor com os dados de input de teste para os 28 pacientes. De seguida, o *train* foi dividido entre os dados que seriam utilizados para treino e os que seriam utilizados para validação. A divisão foi feita utilizando a função *random_split* com uma *seed* fixa e igual a 42. De entre os dados dos 112 pacientes existentes no *train*, considerou-se que 10%, ou seja, os dados de 12 pacientes, seriam utilizados para validação e os restantes para o processo de treino.

Por fim, foram criados 6 *dataloaders*, 2 para o treino, 2 para a validação e 2 para o teste. Para cada par de *dataloaders* foi criado um com o *batch size* igual a 1, tendo sido esses os *dataloaders* utilizados para treino (*train_dl*), validação (*val_dl*) e teste (*test_dl*). O outro *dataloader* do par tem um *batch size* igual ao len do respetivo *TensorDataset*, de forma a conter todos os dados. Assim, o *train_dl_all* apresenta 100 linhas, o *val_dl_all* 12 linhas e o *test_dl_all* 28 linhas.

2.4. Visualização dos dados

Na Figura 1 estão representadas diversas informações relativas aos *dataloaders train_dl_all, val_dl_all* e *test_dl_all,* como por exemplo o número de casos de treino, o valor máximo existente quer nos dados de input, quer no *label* e o *shape* dos tensores que constituem os *dataloaders*. É de realçar que esta visualização de informação foi efetuada para estes *dataloaders* e não para os com *batch size* igual a 1, uma vez que nestes não seria possível visualizar todos os dados presentes no *dataset*, apenas 1 de cada vez.

```
Quantidade de casos de Treino:100
Shape tensor batch casos treino, input: torch.Size([100, 1202]), output: torch.Size([100])
Valor máximo no treino :1.0 Valor mínimo:0.0
Valor máximo na idade :77.0 Valor mínimo:13.0
Quantidade de casos de Validação:12
Shape tensor batch casos validação, input: torch.Size([12, 1202]), output: torch.Size([12])
Valor máximo na validação :1.0 Valor mínimo:0.0
Valor máximo na idade :79.0 Valor mínimo:16.0
Quantidade de casos de Teste:28
Shape tensor batch casos teste: torch.Size([28, 1202])
Valor máximo no teste :1.0 Valor mínimo:0.0
```

Figura 1. Sumário das principais informações relativas aos dados presentes nos *dataloaders train_dl_all, val_dl_all* e *test_dl_all.*

2.5. Balanceamento dos dados

De forma a avaliar a forma como as idades se encontram distribuídas no treino e na validação, foram construídos 2 gráficos, representados nas Figuras 2 e 3. Através da sua observação, é possível verificar que no treino existem algumas idades predominantes, nomeadamente os 15, 16 e 69 anos. Contudo, esta situação não é grave, pois a média dos valores tende para aproximadamente a média das idades (44.03 anos). Relativamente à validação, como seria de esperar por existirem apenas 12 pacientes, existe apenas 1 valor para 12 idades diferentes e por isso, obviamente, os dados estão balanceados. Apesar das distribuições de treino e de validação não serem semelhantes, os seus valores médios são muito próximos (44.03 anos para o treino e 46.67 anos para a validação), pelo que o balanceamento dos dados não deverá influenciar negativamente o processo de treino.

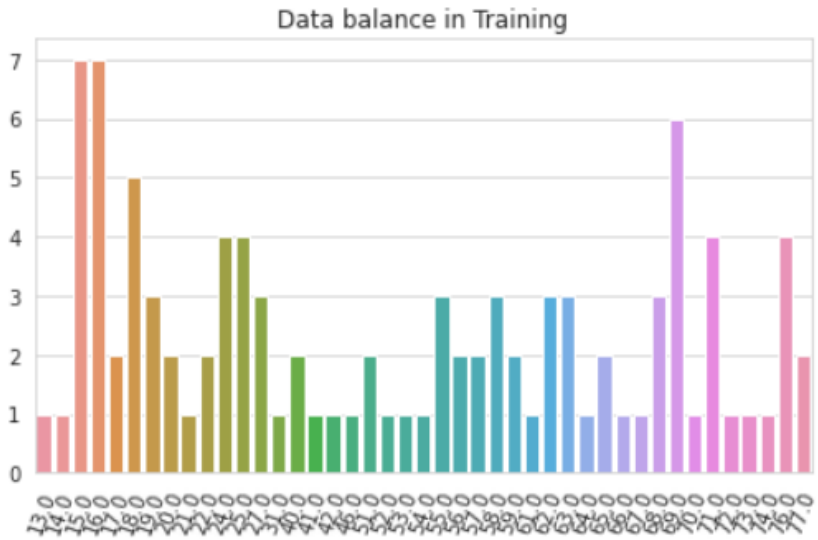


Figura 2. Balanceamento das idades nos dados de treino.

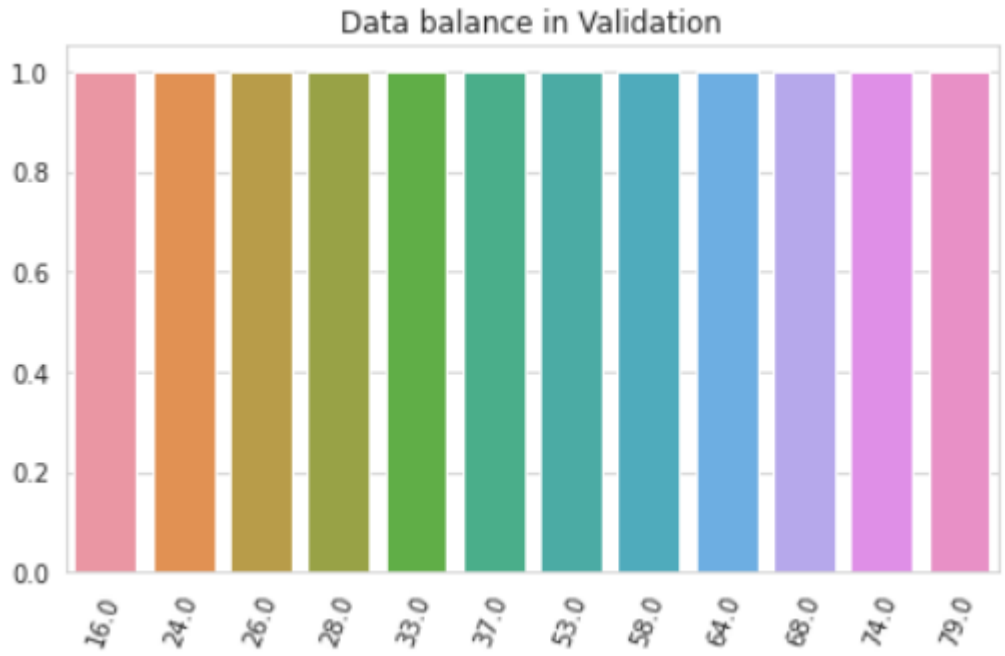


Figura 3. Balanceamento das idades nos dados de validação.

3. Descrição do modelo desenvolvido

Na Figura 4 está representada a arquitetura do modelo 8 desenvolvido, que é uma *Fully Connected Net*, uma rede neuronal pré-definida no Monai [1]. Esta consiste em *layers* totalmente conectadas compostas por uma sequência de camadas lineares com ativação PReLU e *dropout*. A rede aceita como input n canais *in_channels,* tem um output com n canais *out_channels* e pode conter n canais de output de camada oculta dados em *hidden_channels*.

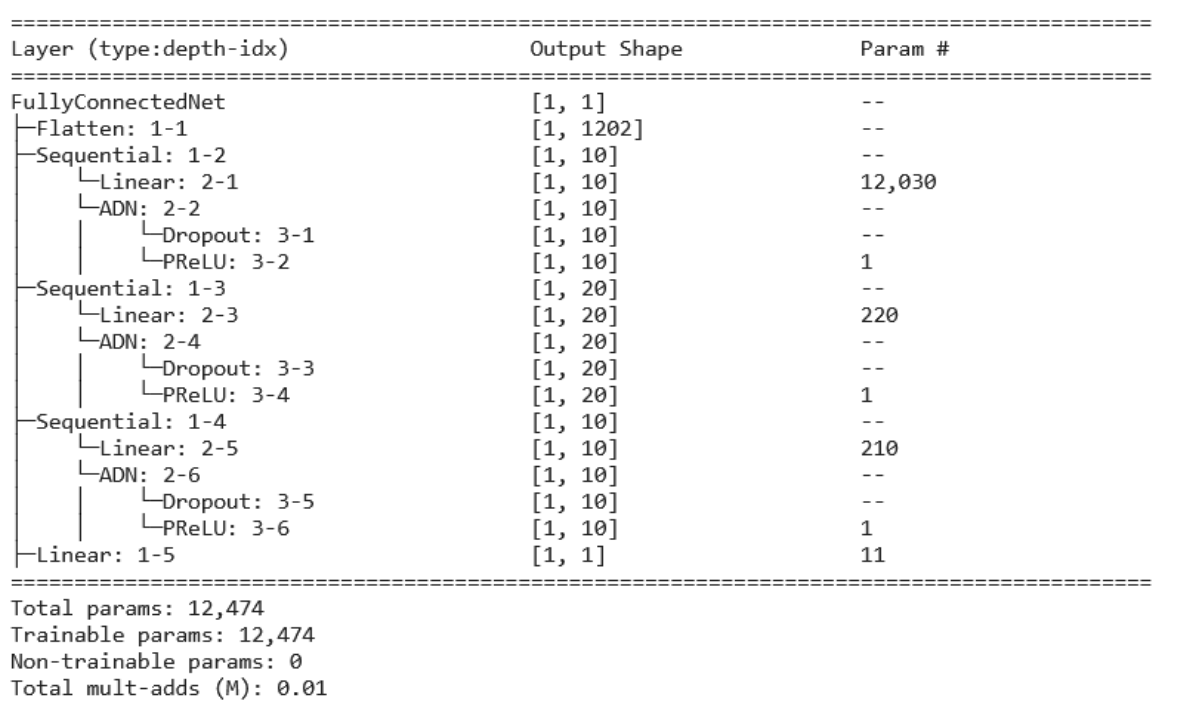
A partir dessa rede já existente, foram-lhe passados os seguintes argumentos:

- ***in_channels:*** 1202, correspondentes às 1202 *features* obtidas a partir das matrizes iniciais;
- ***out_channels:*** 1, pois o modelo deve ser capaz de devolver 1 valor de previsão de idade;
- ***hidden_channels:*** 3 *hidden layers*, com os valores 10, 20 e 10 como output. Estes valores foram escolhidos, após várias tentativas com valores menores e maiores, menos e mais *layers*, mas estes foram os que proporcionaram melhores resultados. Verificou-se que perante o mesmo número de *learning parameters*, a rede aprendia melhor se existissem menos *layers* com valores superiores nos nodos, do que mais *layers* com menores valores nos nodos.
- ***dropout:*** 0.2. Escolheu-se este valor, pois foi o que levou a melhores desempenhos da rede. Com valores superiores, inúmeras ligações da rede eram "cortadas" e a capacidade de

aprendizagem do modelo piorava substancialmente. Contudo, com valores inferiores de *dropout,* o modelo tendia a decorar os valores, levando a *overfiting;*

- ***activation:*** PReLU. Esta foi a função de ativação utilizada e foi também a primeira a ser testada por ser utilizada por *default* na rede. Contudo, outras funções de ativação, como a LeakyReLU, a GeLU e a ReLU também foram testadas, mas todas estas levavam a piores desempenhos do modelo, que se traduziam em valores de *loss* superiores. É também de realçar que a LeakyReLU foi a que apresentou um pior desempenho, provocando aumentos muito acentuados da *loss* do treino.

No final, obteve-se a rede com um total de 12.474 *learning parameters.*



```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
FullyConnectedNet                        [1, 1]                    --
├─Flatten: 1-1                           [1, 1202]                 --
├─Sequential: 1-2                        [1, 10]                   --
│    └─Linear: 2-1                       [1, 10]                   12,030
│    └─ADN: 2-2                          [1, 10]                   --
│    │    └─Dropout: 3-1                 [1, 10]                   --
│    │    └─PReLU: 3-2                   [1, 10]                   1
├─Sequential: 1-3                        [1, 20]                   --
│    └─Linear: 2-3                       [1, 20]                   220
│    └─ADN: 2-4                          [1, 20]                   --
│    │    └─Dropout: 3-3                 [1, 20]                   --
│    │    └─PReLU: 3-4                   [1, 20]                   1
├─Sequential: 1-4                        [1, 10]                   --
│    └─Linear: 2-5                       [1, 10]                   210
│    └─ADN: 2-6                          [1, 10]                   --
│    │    └─Dropout: 3-5                 [1, 10]                   --
│    │    └─PReLU: 3-6                   [1, 10]                   1
├─Linear: 1-5                            [1, 1]                    11
==========================================================================================
Total params: 12,474
Trainable params: 12,474
Non-trainable params: 0
Total mult-adds (M): 0.01
```

Figura 4. Arquitetura do modelo desenvolvido a partir da rede *Fully Connected Net* do Monai.

4. Resultados obtidos e Discussão

Estando o modelo completamente desenvolvido, procedeu-se ao treino do mesmo durante 180 épocas e com os seguintes parâmetros:

- ***Learning Rate:*** 0.001 com uso do *scheduler ReduceLROnPlateau,* com factor 0.95 e *patient* 70. Escolheu-se este valor de LR, pois foi o que apresentou melhores resultados, após terem sido experimentadas outras alternativas (como 0.0005, 0.0015, 0.002 e 0.003);
- **Função de perda:** L1Loss (), que corresponde à *Mean Absolute Error.* Esta foi a função de perda utilizada, uma vez que no enunciado pedia especificamente para a utilizar;
- ***Batch Size:*** 1. Sempre que se aumentava o *batch size*, mesmo que fosse para 2, a *loss* aumentava consideravelmente, pelo que 1 foi o valor escolhido, proporcionando melhores resultados, ou seja, menores valores de *loss;*
- **Otimizador:** *AdamW,* pois, após terem sido testados todos os optimizadores existentes no PyTorch, este foi sem dúvidas o que levou a melhores desempenhos do modelo.

O resultado do treino está representado na Figura 5, onde é visível uma diminuição da *loss* do treino, assim como a interseção da curva do treino com a de validação. Como as 2 curvas se intersetaram, estando a partir de cerca das 90 épocas sobrepostas, então é possível concluir que o modelo não apresenta *overfitting.* Verifica-se também que o valor mínimo de *loss* foi de 5.895 anos, o que é um valor muito bom para este tipo de aplicações. Relativamente à variação da *learning rate* efetuada pelo *scheduler*¸ verifica-se que este apenas diminuiu 0.00005 após aproximadamente 130 épocas.

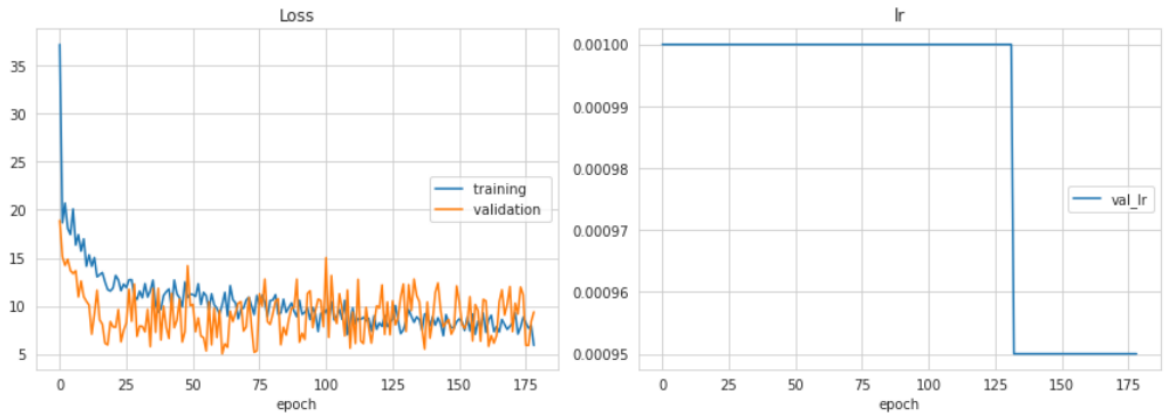


Figura 5. Curvas de *loss* do treino e da validação (à esquerda) e variação da *learning rate* (à direita), obtidas a partir do treino do modelo 8 por 180 épocas.

Estando o modelo treinado, foi possível utilizá-lo para a previsão de idades de 28 pacientes. Os resultados encontram-se no ficheiro "submission.csv", em anexo a este relatório, onde é visível uma boa generalização das idades, que variam de aproximadamente 15.2 a 83.8 anos. É também de realçar que a submissão do ficheiro CSV no *Kaggle* resultou num *score* de 5.58130, sendo este valor ainda inferior ao valor mínimo de *loss* apresentado pela rede. Isto mostra que o modelo está a generalizar bastante bem, não estando *overfited.*

Por último, após ter sido desenvolvido e testado o modelo, procedeu-se a uma análise de *feature importance*, de forma a determinar quais as *features* o modelo considerou mais importantes para fazer a previsão das idades. Para tal, recorreu-se à função *Feature Permutation* existente na biblioteca do Captum. Esta consiste numa abordagem baseada em perturbação para calcular a atribuição, que toma cada *feature* de input, permuta os seus valores dentro de um *batch* e calcula a diferença entre os outputs originais e os misturados para essa determinada *batch*. Essa diferença significa a importância da *feature* para a *feature* permutada [2].

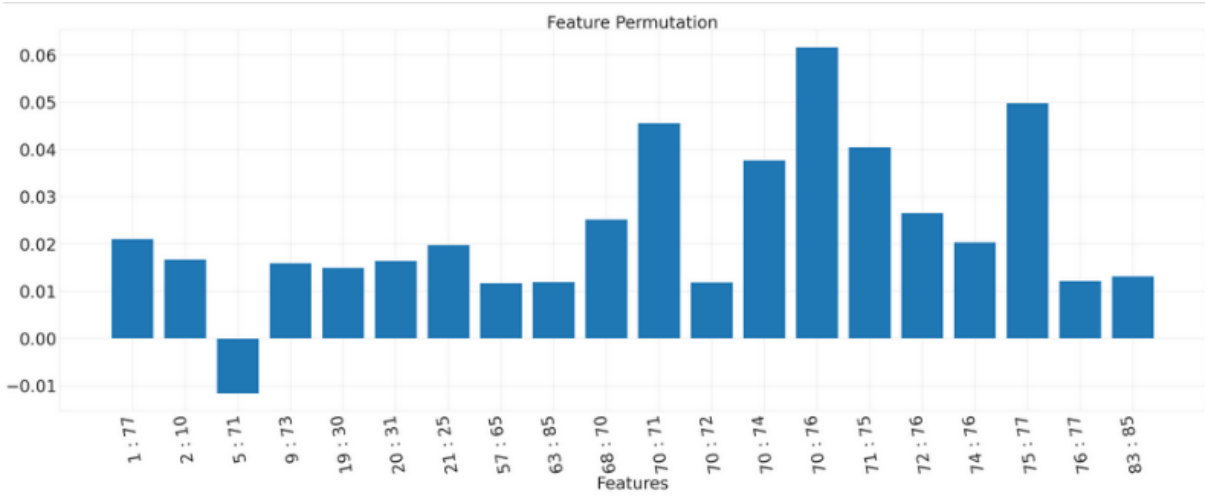
Nas Figuras 6 e 7 estão representadas as 20 *features* mais importantes (negativa ou positivamente), segundo o *Feature Permutation*, com a legenda numérica e com a legenda por extenso, respetivamente.

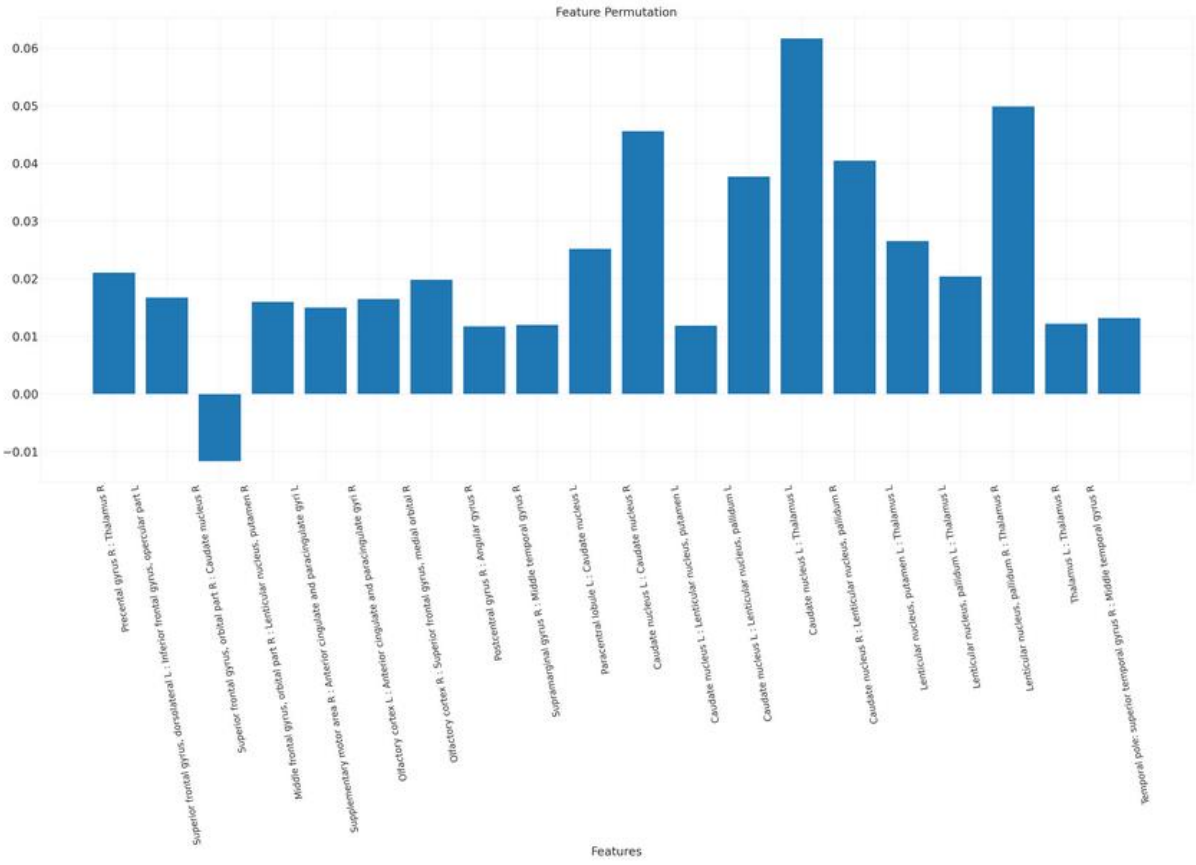
Pela observação das Figuras 6 e 7 verifica-se que as *features* mais importantes positivamente são:

1. 70 e 76: Paracentral lobule R e Lenticular nucleus, pallidum R;
2. 75 e 77: Lenticular nucleus, pallidum L e Thalamus L;
3. 70 e 71: Paracentral lobule R e Caudate nucleus L;
4. 71 e 75: Caudate nucleus L e Lenticular nucleus, pallidum L;
5. 70 e 74: Paracentral lobule R e Lenticular nucleus, putamen R.

Tendo em conta essas relações, verifica-se que essas regiões cerebrais próximas entre si se relacionam todas umas com as outras. Assim, é visível uma forte relação entre as regiões 70 a 77. Relativamente às *features* que mais influenciam negativamente o modelo, entre as 20 apresentadas, verifica-se que a mais influente e a única presente na Figura é a conexão entre as regiões 5 e 71, ou seja, entre Superior frontal gyrus, orbital part L e Caudate nucleus L.

Relativamente às regiões que mais aparecem nas conexões importantes para o modelo, estas correspondem às regiões 70 (Paracentral lobule R) e 71 (Caudate nucleus L), pelo que é possível concluir que estas regiões apresentam conexões com várias regiões, sendo essas conexões importantes para a previsão da idade de um indivíduo.





Figuras 6 e 7. Representações gráficas da *Feature Permutation* obtido para o modelo 8, com as regiões identificadas numericamente e por extenso, respetivamente.

5. Reflexão Crítica e Propostas de melhoria

O objetivo principal deste trabalho foi cumprido, uma vez que se desenvolveu um modelo, capaz de prever, com um erro de 5.895 anos, a idade de um paciente, com base nos seus dados de conetividade cerebral. Contudo, existiram algumas limitações no desenvolvimento do modelo, nomeadamente a dificuldade em diminuir a *loss,* mesmo tendo sido testadas diversas combinações de várias arquiteturas e hiperparâmetros. Isto porque, sempre que nas diferentes tentativas efetuadas a *loss* diminuia, o modelo apresentava sempre *overfitting*, e consequentemente más previsões da idade, pois, como o modelo decorou os dados de treino, não tinha a capacidade de generalizar para casos que não conhecia. Outra dificuldade que ocorreu numa fase inicial do trabalho, foi a troca das posições dos tensores no *dataloader,* que fazia com que as previsões tendessem sempre para a média das idades.

Como propostas de melhoria sugere-se a incorporação de novas arquiteturas e o desenvolvimento de novos modelos. Como o grupo se focou mais no modelo 8, e não tanto nos restantes, poderá ter perdido oportunidade de melhorar esses modelos e obter melhores resultados relativamente aos apresentados. Inclusive, deveriam ter sido mais explorados os modelos que usam todos os *features* e não apenas os de conetividade cerebral, pois seria de esperar que estes melhorassem os resultados obtidos, apesar de na prática, não se ter alcançado essa meta. Por último, apesar do grupo não ter encontrado nenhum modelo já existente e aplicado para o mesmo tipo de dados utilizado neste projeto, poderia ter sido efetuada uma investigação mais profunda nesse sentido.

6. Referências

[1] Monai. https://docs.monai.io/en/stable/networks.html#fullyconnectednet. Acedido a 04 maio 2022.

[2] Captum. https://captum.ai/api/feature_permutation.html. Acedido a 04 maio 2022.