Universidade da Beira Interior
Departamento de Informática

Nº 1 - 2023: *[TP2] Reconhecimento de Atividade Humana*

Elaborado por:

Simão Andrade a46852

Orientador:

Professor Doutor Hugo Pedro Proença

9 de janeiro de 2023

Agradecimentos

Gostaria de começar agradecendo ao professor Doutor Hugo Proença pelo acompanhamento e orientação nas aulas práticas de aprendizado de máquina. A sua experiência e conhecimento foram fundamentais para compreender os conceitos e técnicas de rede neural e aplicá-los eficientemente.

Também gostaria de agradecer aos meus colegas, João Paixão e Bernardo Claro pela ajuda na testagem dos hiperparâmetros e quantidade de neurónios. A sua colaboração foi fundamental para encontrar os melhores valores e otimizar o desempenho da rede neural.

Conteúdo

Lista de Figuras

Acrónimos

IA Inteligência Artificial

UBI Universidade da Beira Interior

UC Unidade Curricular

Capítulo

1

Introdução

1.1 Enquadramento

Este relatório foi feito no contexto da unidade curricular de Inteligência Artificial (IA) da Universidade da Beira Interior (UBI).

1.2 Objetivos

Neste trabalho prático tem como a criação de uma rede neural para reconhecer atividades praticadas por seres humanos através de um conjunto de dados adquirido do WISDM Lab seria o de treinar a rede de modo que ela consiga realizar o reconhecimento das atividades de maneira precisa e eficiente.

Isso pode ser feito utilizando técnicas de aprendizagem não supervisionado, em que são fornecidas as entradas (dados do conjunto), porém não são fornecidas as saídas desejadas (rótulos das atividades) para a rede durante o processo de treino, pois a mesma vai se basear meramente nas suas amostras.

Outro objetivo importante pode ser o de obter curvas ROC (Receiver Operating Characteristic), AUCs (Area Under the curva), matriz confusão e outras estatísticas relevantes para avaliar o desempenho da rede e o que deve ser melhorado.

1.3 Organização do Documento

De modo a refletir o trabalho que foi feito, este documento encontra-se estruturado da seguinte forma:

1. O primeiro capítulo – **Introdução** – apresenta o projeto, a motivação para a sua escolha, o enquadramento para o mesmo, os seus objetivos e a respetiva organização do documento.

2. O segundo capítulo – **1.ª Passo** – descreve a implementação do algoritmo encarregue de gerar as instâncias e a corresponde organização das mesmas.

3. O terceiro capítulo – **2.ª Passo** – descreve a implementação do algoritmo encarregue de gerar os folds, como eles estão organizados e a sua respetiva normalização.

4. O quarto capítulo – **3.ª Passo** – descreve a elaboração da criação da rede neuronal e o processo de *'encoding'*

5. O quinto capítulo – **4.ª Passo** – descreve e ilustra todas as estatísticas obtidas na própria.

6. O sexto capítulo – **Conclusão** – descrevem-se os objetivos conseguidos e não conseguidos, reunidos em tópicos autoexplicativos, e o trabalho futuro relacionado com o projeto.

Capítulo

2

1ª Passo

2.1 Introdução

Neste capítulo, tratamos da criação de instâncias para treino de uma rede neural de aprendizagem de máquina. Uma rede neural é um modelo de aprendizagem de máquina (machine learning) que consegue aprender padrões e fazer previsões com base em dados de entrada. Para treinar uma rede neural, é preciso fornecer-lhe um conjunto de instâncias de treino.

2.2 Definição de instância

Instância é um termo usado em aprendizagem de máquina (machine learning) para se referir a um exemplo ou caso de treino. Em outras palavras, uma instância é um conjunto de dados usado para treinar um modelo de aprendizagem de máquina. Por exemplo, se estivermos a treinar um modelo de classificação de imagens, cada imagem seria uma instância de treino.

2.3 Aprendizagem

Aprendizagem por instância é um método de aprendizagem onde o modelo é treinado com base num conjunto de instâncias individuais.

Em vez de tentar generalizar a partir de uma abundância de dados, o modelo é treinado para fazer previsões precisas para cada instância individualmente. Isso pode ser útil em casos onde o conjunto de dados é muito pequeno ou quando cada instância é muito importante e precisa ser tratada de maneira muito específica.

No entanto, a aprendizagem por instância também tem algumas desvantagens. Como o modelo é treinado com base em cada instância individualmente, ele pode não ser tão bom em generalizar para novos conjuntos de dados. Além disso, o treino pode ser mais demorado, pois, o modelo precisa processar cada instância individualmente.

2.4 Formato das instâncias

As instâncias de treino são armazenadas num arquivo de texto com o formato CSV (comma-separated values). Cada linha do arquivo representa uma instância e cada coluna representa um atributo.

Os primeiros 60 atributos representam os valores adquiridos pelo celerímetro (x, y, z) numa taxa de 20Hz (20 amostras por segundo). Os dois últimos atributos representam o ID da atividade ('0' para 'downstairs, '1' para 'jogging', '2' para 'sitting', '3' para 'standing', '4' para 'upstairs', '5' para 'walking') e o ID do utilizador (1-36), respetivamente.

2.5 Explicação da função 'create_instances'

A função *'create_instances'* recebe como argumento os dados originais. Ela percorre todas as linhas do ficheiro '.CSV' caso existam mais 20 linhas, e as mesmas pertençam ao mesmo utilizador e à mesma atividade, cria uma instância com os valores dos 60 atributos e os dois últimos atributos (ID da atividade e ID do utilizador). Caso contrário, passa para a próxima linha.

Este processo é repetido até que não existam mais linhas no ficheiro '.CSV', e depois é guardado o resultado num ficheiro '.CSV' com o nome 'instances.csv'.

2.6 Conclusões

Nesta fase do trabalho foi possível perceber como os dados vão ser organizados e melhorar os conhecimentos da linguagem *Python*.

Capítulo

3

2ª Passo

3.1 Introdução

A validação do modelo é uma etapa crucial na criação de um modelo de aprendizagem de máquina. O objetivo é avaliar como o modelo se comporta em dados que ele ainda não viu, para garantir que ele possa generalizar bem para novos dados. Uma das técnicas mais comuns para validar um modelo é dividir os dados em dois conjuntos: um conjunto de treino e um conjunto de teste. O modelo é treinado com o conjunto de treino e, em seguida, é avaliado com o conjunto de teste.

No entanto, às vezes é útil dividir o conjunto de treino em múltiplas "dobras" ou 'folds', para ter uma ideia mais precisa do desempenho do modelo. Isso é conhecido como validação cruzada.

Neste capítulo, discutiremos como criar 'folds' de treino e teste usando o algoritmo *K-fold validation*. Explicaremos como dividir os dados em 'folds' de tamanho igual e como treinar e avaliar o modelo em cada uma dessas 'folds'.

Também discutiremos como foi feita a normalização dos folds resultantes do algoritmo K-fold validation e como realizar essa normalização de maneira eficaz.

3.2 Explicação da função 'create_k_folds'

A função *'create_k_folds'* recebe como argumento o número de 'folds' que se planeia criar.

Ela divide o conjunto de instâncias em 'k' partes iguais (onde em 'folds' diferentes não existem instâncias com o mesmo ID de utilizador), e para cada uma dessas partes, cria um conjunto de treino, um conjunto de teste e um conjunto de validação. O conjunto de treino é composto por todas as instâncias, exceto as do fold atual, e o conjunto de teste é composto pelas instâncias do 'folds' atual. Este processo é repetido até que não existam mais 'folds'.

3.3 Separação do conjunto de treino, teste e validação

A separação dos 'folds' em conjuntos de treino, teste e validação é uma técnica comum em aprendizagem de máquina para avaliar o desempenho de um modelo e evitar o overfitting (quando o modelo é ótimo em prever os dados de treino, mas tem um desempenho fraco em dados novos).

O processo de separação dos 'folds' em conjuntos de treino, teste e validação é feito da seguinte forma:

- Usar um dos k 'folds' como conjunto de validação, os dois seguintes como conjunto de teste e os outros k-3 'folds' como conjunto de treino.
- Repetir o processo k vezes, cada vez usando um 'folds' diferente como conjunto de validação.

Cada conjunto um dos conjuntos tem a seguinte função:

- **Conjunto de treino**: usado para treinar o modelo. Ele é usado para aprender padrões nos dados e ajustar os parâmetros do modelo.
- **Conjunto de teste**: usado para avaliar o modelo. Ele é usado para avaliar o desempenho do modelo em dados que ele ainda não viu.
- **Conjunto de validação**: usado para ajustar os parâmetros do modelo. Ele é usado para ajustar os parâmetros do modelo, como o número de épocas de treino, a taxa de aprendizagem, etc.

3.4 Explicação da função 'normalizeData'

A função *'normalizeData'* recebe como argumento o conjunto de instâncias (treino e teste) que se planeia normalizar e o valor mínimo e máximo do conjunto de treino (obtido através da função *getMinMax*).

A função percorre todas as instâncias do conjunto de treino e altera o valor de cada atributo (x, y, z) para o valor normalizado, calculado da seguinte forma:

- VA: Valor do atributo
- VMin: Valor mínimo do conjunto de treino
- VMax: Valor máximo do conjunto de treino
- VA = (VA - VMin) / (VMax - VMin)

Caso o valor máximo do conjunto de treino seja igual ao valor mínimo do conjunto de treino, o valor do atributo é eliminado.

3.5 Conclusões

Através deste capítulo, conseguimos perceber o funcionamento e a motivação da criação dos conjuntos de treino, teste e validação através dos folds. E com isto já estaremos prontos para desenvolver a rede neuronal.

Capítulo

4

3ª Passo

4.1 Introdução

As redes neurais são um tipo poderoso de modelo de aprendizagem de máquina, capaz de realizar tarefas complexas de processamento de dados, como classificação, regressão e deteção de padrões. Elas são inspiradas na estrutura do cérebro humano, compostas por camadas de neurónios intersectados que trabalham juntos para realizar tarefas específicas.

Neste capítulo, apresentarei os principais conceitos envolvidos na criação de uma rede neuronal (em *Python*) e discutiremos como implementar uma rede neural num trabalho prático.

4.2 Processo de criação da rede neuronal

Para esta tarefa, foi utilizada a biblioteca *scikit-learn* (sklearn), que contém um método chamado ***MLPClassifier***, nela é possível especificar os hiperparâmetros da rede neuronal, como o número de camadas ocultas e o número de neurónios em cada camada.

Em cada iteração será extraída a última coluna do conjunto de treino e teste (representando a atividade do utilizador) e será removida do conjunto de treino e teste, com a segunda coluna (representando o ID do utilizador), para não ser utilizada na criação da rede neuronal.

Essa coluna extraída originará uma lista com os valores da atividade do utilizador, que será utilizada para treinar a rede neuronal (rótulos). A lista será convertida para uma versão codificada, através do método ***OneHotEncoder***, que transformará os valores numéricos em valores binários, para a rede neuronal conseguir interpretar os dados.

Por fim, a rede neuronal é treinada com o conjunto de treino. Através do método *'fit'* que recebe como argumentos o próprio conjunto e a lista das atividades codificadas, e o conjunto de teste é utilizado para avaliar o modelo através do método *'predict'*, que recebe como argumento o conjunto de teste e retorna uma lista com as previsões do modelo.

Nota: Por motivos de depuração, a rede será guardada num ficheiro através da biblioteca *joblib*.

4.3 Conclusões

Neste capitulo conseguimos entender todo o processo de criação de uma rede neuronal com ajuda das bibliotecas do *Python*.

Capítulo

5

Estatísticas

5.1 Introdução

O tópico de estatísticas em redes neurais é uma área importante de pesquisa e aplicação na área de aprendizagem de máquina.

Ao treinar e avaliar um modelo de rede neural, é importante medir o desempenho do modelo para determinar se ele consegue realizar a tarefa de maneira eficiente. Para isso, é necessário coletar e analisar dados estatísticos sobre o desempenho do modelo.

Existem várias métricas de desempenho muito utilizadas em redes neurais, como precisão do modelo, curva ROC (Receiver Operating Characteristic), AUC (Area Under the Curve), matriz confusão e a pontuação final da rede e o correspondente desvio padrão. Cada uma dessas métricas fornece uma visão diferente sobre o desempenho do modelo e é importante entender como elas se relacionam para ter uma visão completa do desempenho do modelo.

5.2 Explicação das Métricas

5.2.1 Precisão do Modelo (Model Accuracy)

A precisão do modelo é uma métrica de avaliação que mede a percentagem de previsões corretas do modelo em relação ao total de previsões feitas.

5.2.2 Curva ROC

A curva ROC é uma métrica de avaliação usada para comparar a capacidade de classificação de dois ou mais modelos em problemas de classificação biná-

ria. Ela é *'plotada'* usando o número de verdadeiros positivos e o número de falsos positivos em função da taxa de verdadeiros positivos e da taxa de falsos positivos.

Como podemos ver nas imagens seguintes dos testes realizados, apesar de terem sido obtidos bons valores, temos atividades com uma maior região de valores (onde os modelos variam demasiado nestas atividades) no caso 'Downstairs' (a descer) e o 'Upstairs' (a subir).

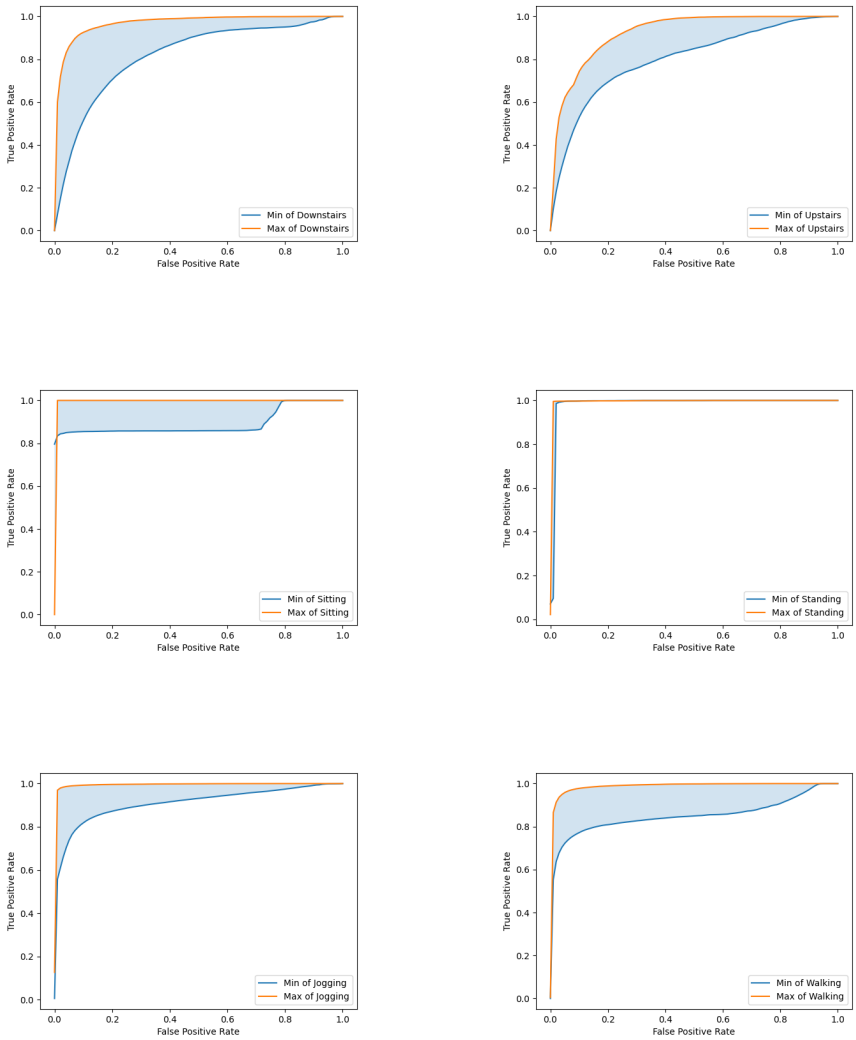

Figura 5.1: Região de valores de cada atividade nas 10 iterações

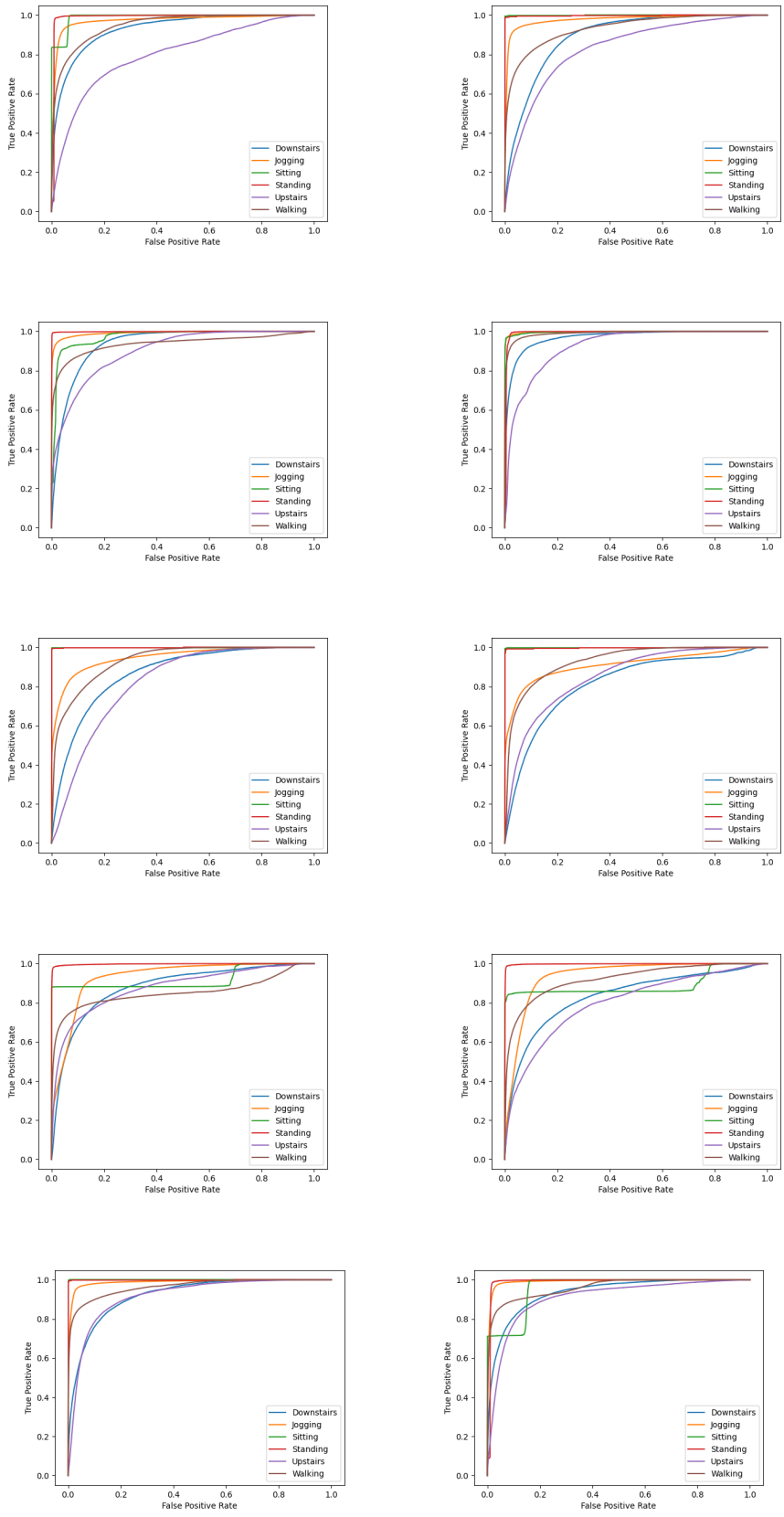

Figura 5.2: Curvas ROC por ordem de iteração (’fold’)

5.2.3 AUC (Area Under the Curve)

A AUC é uma métrica de avaliação que mede a capacidade de classificação de um modelo em problemas de classificação binária. Ela é calculada pela área sob a curva ROC.

Quanto maior a AUC, melhor é o desempenho do modelo (sendo 1.0 o melhor valor possível e 0.5 sendo um desempenho aleatório).

As imagens abaixo representam, em percentagem, as AUC's dos 10 modelos e a média deles em cada atividade. Todos eles usaram duas camadas e 120 neurónios em cada.

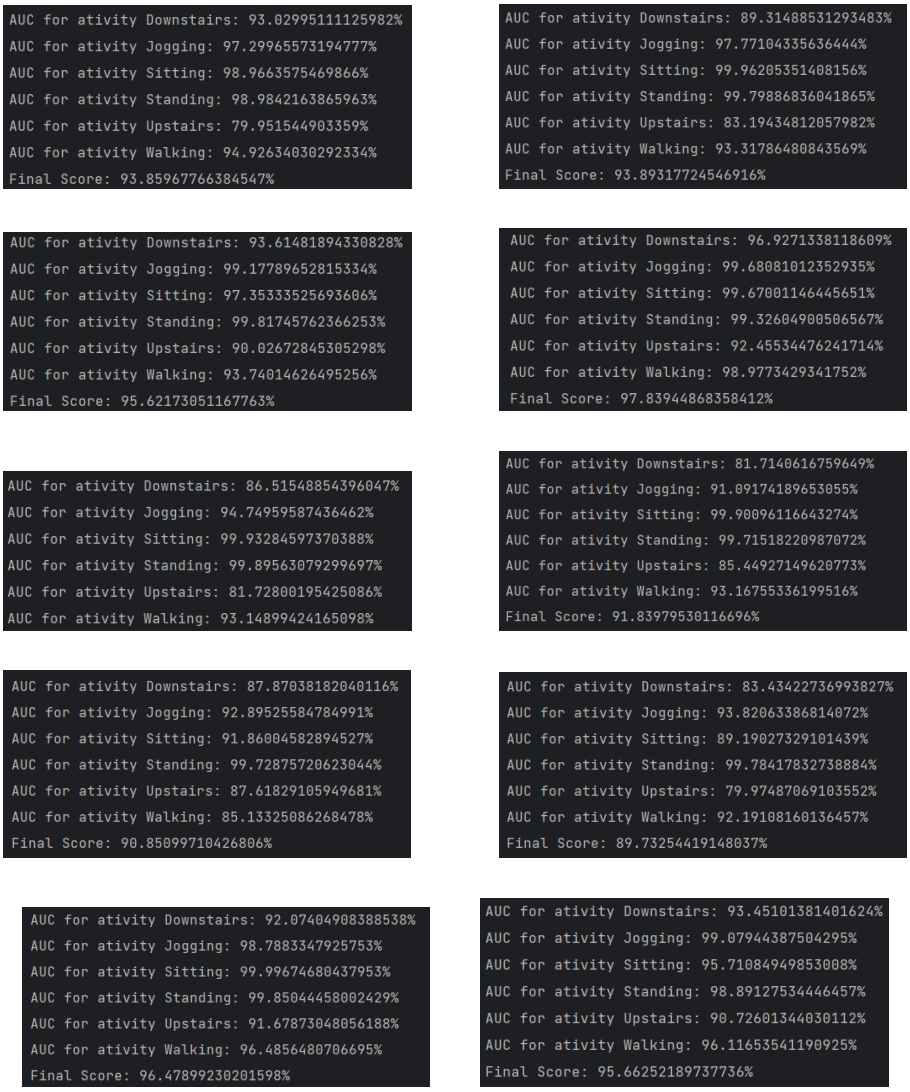

Figura 5.3: Resultados AUC de todas as iterações e atividades.

5.2.4 Matriz Confusão

A matriz de confusão é uma métrica de avaliação em aprendizagem de máquina que mostra como o modelo efetua previsões em relação às classes verdadeiras. Ela é composta por quatro elementos: verdadeiros positivos (TP), falsos positivos (FP), falsos negativos (FN) e verdadeiros negativos (TN).

A matriz de confusão é útil, pois permite avaliar o desempenho do modelo em cada classe individualmente e fornece informações sobre os erros cometidos pelo mesmo.

As imagens abaixo contém a matriz confusão por atividade da iteração número oito, as restantes encontram-se na pasta 'Imagens' no projeto enviado em anexo.

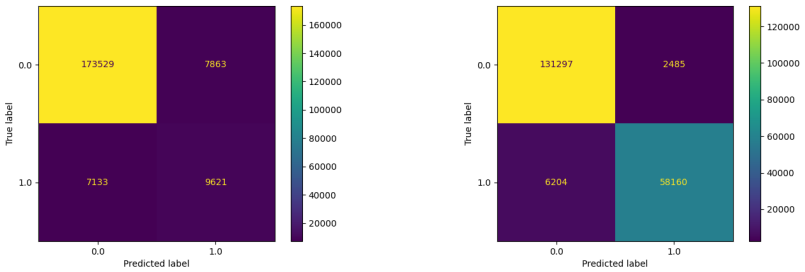

Figura 5.4: Matriz confusão das atividades 'Downstairs e 'Jogging'

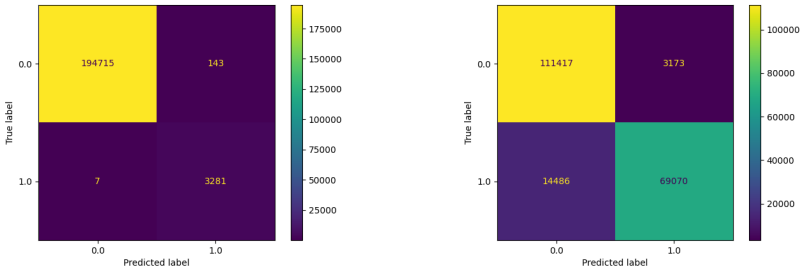

Figura 5.5: Matriz confusão das atividades 'Sitting' e 'Walking'

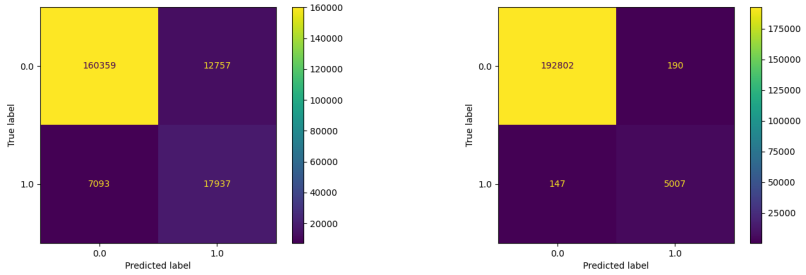

Figura 5.6: Matriz confusão das atividades 'Upstairs' e 'Standing'

5.2.5 Pontuação Final da Rede e Desvio Padrão

O cálculo da pontuação final da rede e do desvio padrão é feito com o propósito de obter uma estimativa mais precisa do desempenho do modelo. Isso é especialmente útil quando o conjunto de dados é dividido em partes (como é o caso) para avaliar o desempenho do modelo e ajustar os hiperparâmetros.

A imagem abaixo representa esse cálculo na nossa rede.

```
Final Answer: 0.94 +/- 0.03
Average AUC for ativity Downstairs: 0.8979460114875302
Average AUC for ativity Jogging: 0.9643544118944989
Average AUC for ativity Sitting: 0.9725434803454667
Average AUC for ativity Standing: 0.995792059836719
Average AUC for ativity Upstairs: 0.8628031453612628
Average AUC for ativity Walking: 0.937204757860761
```

Figura 5.7: Média de AUC's por atividade e *Final Answer*

5.3 Conclusão

Neste capítulo foi exibido os valores obtidos da rede neuronal através dos dados fornecidos e a devida explicação de cada um destes valores.

Capítulo

6

Conclusões e Trabalho Futuro

6.1 Conclusões Principais

No presente trabalho, foi desenvolvida uma rede neural capaz de reconhecer qual atividade um ser humano praticava. Foram coletados dados de celerímetro e giroscópio de dispositivos portáteis e utilizados para treinar e testar o modelo de rede neural.

Em suma, o modelo de rede neural desenvolvido neste trabalho mostrou conseguir reconhecer com boa precisão qual atividade era praticada, no entanto, ainda contém certa dificuldade em distinguir algumas das atividades (nomeadamente as atividades 'Downstairs' e 'Upstairs'). Além disso, não chegou a ser usado o conjunto de validação no modelo e, apesar de não ter sido exibido, foi feita uma comparação com a previsão através do ID do utilizador, porém não teve resultados tão positivos.

6.2 Trabalho Futuro

Num futuro trabalho na temática desta Unidade Curricular (UC) gostaria de desenvolver mais exemplos de aplicações como classificação de imagem ou até mesmo algoritmos de recomendação de anúncios, entre outros.

Bibliografia