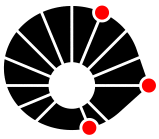
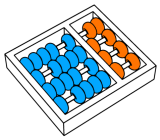




Reconhecimento de Quedas a Partir de Sinais de Giroscópio e Acelerômetro Utilizando Aprendizado de Máquina

F. T. Semissatto *H. Pedrini*

Relatório Técnico - IC-PFG-22-18
Projeto Final de Graduação
2022 - Julho

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Reconhecimento de Quedas a Partir de Sinais de Giroscópio e Acelerômetro Utilizando Aprendizado de Máquina

Felipe Teixeira Semissatto* Hélio Pedrini*

Resumo

Este trabalho tem como proposta principal a criação de um aplicativo iOS com o objetivo de reconhecer quedas físicas utilizando-se de um telefone celular (*smartphone*), o qual detecta sinais de aceleração e rotação a partir dos sensores de acelerômetro e giroscópio. Inicialmente, o relatório descreve as principais atividades para a criação de um modelo de aprendizado de máquina. Tais etapas envolvem os processos de coleta de dados, organização dos dados coletados e, por fim, o processo de treinamento de modelos de aprendizado de máquina. A partir do modelo treinado, foi possível fazer predições de atividades humanas a partir de dados dos sensores em tempo real. Em seguida, o relatório faz um comparativo entre um modelo utilizando a técnica de Árvore de Decisão com uma variante de rede neural recorrente conhecida como *Long Short-Term Memory* (LSTM). Com isso, o relatório enfatiza que, em casos onde os dados são sequenciais e, dessa forma, os dados possuem uma relação entre si (tanto em termos de ordem quanto de agrupamento), o uso de redes neurais recorrentes possui um desempenho melhor visto que aprende relações existentes entre os dados.

1 Introdução

Com o passar do tempo, nós nos tornamos cada vez mais suscetíveis a acidentes domésticos. Isso porque, ao atingirmos uma certa idade, temos uma piora da qualidade da visão, na força dos músculos e no equilíbrio. Esse conjunto de fatores é responsável por causar traumas nas pessoas idosas, as quais precisam ser levadas imediatamente ao hospital.

Além desse grupo de pessoas, também há as pessoas com condições especiais como, por exemplo, os cadeirantes e as pessoas que utilizam próteses para se movimentarem. Assim como os idosos, pessoas com condições especiais também estão mais suscetíveis a acidentes, visto que as mesmas podem possuir deficiências visuais, motoras, mentais ou auditivas.

Tendo em vista esses conjuntos de pessoas que são mais suscetíveis a acidentes, torna-se necessário um monitoramento contínuo desses indivíduos para evitar traumas e, caso algum imprevisto aconteça, levá-las a um hospital o mais breve possível para iniciar os procedimentos de primeiros socorros. Geralmente, essas pessoas são monitoradas por outra pessoa (como um cuidador, por exemplo), mas a situação agrava-se quando as mesmas estão sozinhas.

*Instituto de Computação, Universidade Estadual de Campinas, 13083-852 Campinas, SP.

Tendo isso em mente, é possível notar que cada vez mais os *smartphones* estão mais populares e, com isso, grande parcela da população possui pelo menos um dispositivo em mãos. Além das atividades essenciais de um celular (como fazer chamadas telefônicas ou acesso rápido à Internet), os *smartphones* geralmente possuem dois sensores poderosos embutidos: o giroscópio e o acelerômetro. De forma resumida, o giroscópio captura movimentos de rotação, enquanto o acelerômetro é capaz de adquirir a magnitude e a direção da aceleração do dispositivo.

Portanto, o problema abordado neste projeto final de graduação é monitorar e reconhecer eventuais problemas físicos que possam acontecer com essas pessoas a fim de evitar possíveis traumas. Sendo assim, o principal objetivo deste projeto é capturar os sinais provindos de um *smartphone* e ser capaz de reconhecer eventuais acidentes físicos humanos.

Para isso, o projeto incluiu um processo com várias fases, os quais envolveram a coleta de dados por meio de sinais de rotação e aceleração, o tratamento desses dados sequenciais e a criação de modelos de aprendizado de máquina utilizando um mesmo conjunto de dados.

Além disso, o projeto aborda duas técnicas disponíveis na literatura baseando-se em diferentes descritores e classificadores presentes em diversas referências bibliográficas. A primeira abordagem se refere a um modelo utilizando o algoritmo conhecido como Árvore de Decisão. Essa abordagem é mais simplista e demonstrou-se menos eficiente, pois, apesar do fácil entendimento e implementação, o algoritmo possui uma certa deficiência quando se lida com dados mais complexos (bem como dados sequenciais). A segunda abordagem se refere a uma rede neural recorrente, o qual tem uma eficácia melhor com dados sequenciais. Isso porque uma rede neural recorrente interpreta cada elemento em uma sequência considerando os elementos que já foram vistos.

Para este projeto, foi utilizada a rede neural recorrente do tipo *Long Short-Term Memory* (LSTM), o qual é uma unidade de recorrência melhorada com memória interna que, além de processar o item atual e os itens anteriores, também processa o que está na memória. As descrições e os resultados entre as duas abordagens serão explicitadas na seções seguintes.

2 Conceitos

Esta seção tem como objetivo descrever os principais conceitos envolvidos neste projeto. Dessa forma, será demonstrada a parte teórica que fundamenta as decisões práticas tomadas com o intuito de elucidar o que é visto na literatura.

Inicialmente, é descrito o tipo de conjunto de dados a ser coletado pelos sensores de acelerômetro e giroscópio. Em seguida, serão explicados os principais conceitos que envolvem aprendizado de máquina e as principais fases para criar um modelo. Por fim, os dois últimos tópicos desta seção apresentam primeiramente o algoritmo de aprendizado de máquina supervisionado e, em seguida, o conceito de rede neural recorrente e uma de suas variantes.

2.1 Conjunto de Dados

O conjunto de dados coletado pelos sensores de acelerômetro e giroscópio é primordial neste projeto, visto que o mesmo esteve presente em todas as etapas, isto é, desde o treinamento

do modelo até a predição final a partir de dados em tempo real. Dessa forma, os dados foram utilizados e nortearam o desenvolvimento da aplicação.

Em poucas palavras, os sensores de acelerômetro e giroscópio fornecem dados sobre o movimento de um dispositivo no mundo físico. O primeiro sensor mede a variação da velocidade do dispositivo ao longo dos eixos, enquanto o giroscópio mede a taxa no qual o dispositivo gira em torno de um eixo espacial.

Para este projeto, os dados de entrada (ou *features*) são referentes aos dados de aceleração, gravidade e rotação do corpo nos eixos X, Y e Z. Uma *feature* é uma porção de dados que é considerada interessante para o modelo de aprendizado de máquina. Portanto, nove dados de entrada no total definem nossos dados em um instante de segundo:

- Rotação no eixo X;
- Rotação no eixo Y;
- Rotação no eixo Z;
- Gravidade no eixo X;
- Gravidade no eixo Y;
- Gravidade no eixo Z;
- Aceleração no eixo X;
- Aceleração no eixo Y;
- Aceleração no eixo Z.

Para o reconhecimento de atividades humanas, devemos tratar os dados de forma sequencial, isto é, é necessário ter conhecimento de dados de instantes anteriores para ajudar a interpretar dados que ainda estão por vir. Na Seção 3.1, forneceremos mais detalhes referentes ao conjunto de dados em termos de como eles foram coletados, a quantidade de medições e as condições estabelecidas para fazer uma única medição.

2.2 Aprendizado de Máquina

Para facilitar o entendimento do assunto, esta seção apresenta uma visão geral sobre aprendizado de máquina e os passos para criar um modelo de aprendizado de máquina.

Podemos dizer que o aprendizado de máquina é o campo de estudo da computação que permite computadores a habilidade de aprender algo sem programação explícita. Em outras palavras, em vez de ter um programador que implementa regras do tipo *se-então-senão*, podemos deixar o computador aprender as regras para resolver esses tipos de problemas a partir de exemplos. Desse modo, aprendizado de máquina é ideal para casos em que não é possível escrever os passos exatos para um programa reconhecer determinada regra. Sendo assim, podemos utilizar algoritmos de aprendizagem que podem reconhecer automaticamente regras que são necessárias para resolver um certo problema.

O aprendizado de máquina tem como um dos conceitos centrais a definição de modelo. O modelo é um algoritmo que foi "ensinado" por um computador para executar uma certa tarefa. Para isso, é necessário um conjunto de dados para treinar e executar o algoritmo. Por isso, podemos dizer que um modelo depende do algoritmo de aprendizagem escolhido

e o conjunto de dados utilizado. Com o modelo treinado, é possível então fazer inferências, ou seja, fazer predições a partir de novos dados ainda não vistos pelo modelo.

Neste projeto, foi adotado o aprendizado de máquina supervisionado, o qual consiste em fornecer exemplos (dados) ao algoritmo com rótulos (ou *labels*) do que eles representam. Além disso, a técnica utilizada no supervisionamento foi a de classificação, pois dividiremos as predições de movimento em categorias (classes) que o modelo será capaz de reconhecer.

No tocante aos passos para criar de fato um modelo de aprendizado de máquina, o projeto seguiu as seguintes etapas (em ordem):

1. Coleção de dados dos sensores de um dispositivo *smartphone*;
2. Adequação e preparação dos dados para criar conjuntos de dados (tais como conjuntos de treinamento, validação e teste);
3. Treinamento de modelos de aprendizado de máquina e criação de uma rede neural que reconhece atividades humanas;
4. Através de um aplicativo iOS, realizar predições de atividades humanas a partir de dados em tempo real.

Referente ao passo 3 listado anteriormente, as duas subseções a seguir explicarão em detalhes as duas abordagens utilizadas neste projeto. A primeira retrata o algoritmo de treinamento de Árvore de Decisão e a segunda explica a rede neural recorrente LSTM.

2.3 Árvores de Decisão

Árvores de decisão são algoritmos de aprendizado de máquina muito versáteis, os quais podem executar tarefas do tipo de classificação e regressão. Apesar de ter uma certa simplicidade e de ser um dos algoritmos mais intuitivos de se implementar, esses métodos demonstram ser poderosos e capazes de lidar com conjunto de dados complexos.

Uma árvore de decisão é um preditor, ou seja, a partir de um número qualquer de entradas, podemos ter como saída uma predição de um rótulo. Sua relação é mostrada na equação a seguir:

$$h : X \to Y \tag{1}$$

em que X representa o conjunto de entradas e Y representa a predição. Como forma de exemplificação, o conjunto de entradas no nosso projeto são as *features* descritas na Seção 2.1, sendo elas os dados de aceleração, gravidade e rotação dos sensores. Além disso, a saída de predição corresponde ao reconhecimento da atividade humana.

Estruturalmente, uma árvore de decisão contém as predições em suas folhas. Assim, a predição ocorre percorrendo a árvore da sua raiz até uma determinada folha. Para determinar qual caminho tomar, nós que estão situados entre a raiz e uma folha decidem o trajeto a ser seguido. Geralmente, a regra da decisão é definida em uma das características de X ou em um conjunto pré-definido de regras de divisão. A Figura 1 ilustra uma árvore de decisão.

Podemos entender que o algoritmo árvore de decisão gera um modelo que segue a regra básica do tipo *se-então-senão*, visto que a mesma toma decisões binariamente de acordo com os dados de entrada. Portanto, o algoritmo foi escolhido neste projeto por ser tratar de um classificador simples de entender e interpretar.

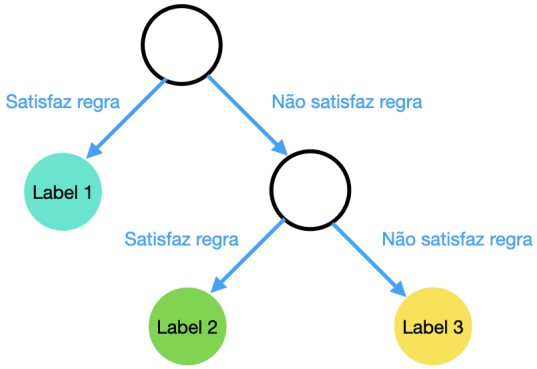


Figura 1: Representação de uma árvore binária. Figura adaptada de [5].

2.4 Rede Neural Recorrente

Uma rede neural recorrente, diferentemente de redes neurais convolucionais, é designada a reconhecer relações temporais. Assim, como uma sequência implica a passagem do tempo, temos que esse tipo de rede neural reconhece relações entre os itens em uma sequência. Para isso, a rede neural analisa as sequências item por item e produz uma saída para cada item baseando-se no item atual e na saída produzida pelo item anterior.

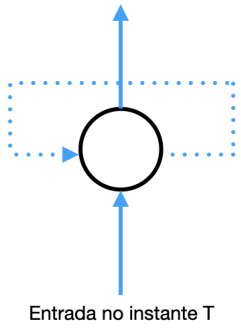


Figura 2: Diagrama de uma rede neural recorrente. Figura adaptada de [6].

Na Figura 2, o círculo representa uma única camada de rede neural recorrente, onde a camada possui qualquer número de nós. Como podemos observar, a entrada de elementos de uma sequência é referenciada pelo seu instante. A camada, então, processa o elemento a todo instante **T** verificando-se a entrada e a própria saída da camada de entradas anteriores em um instante **T-1**. Assim, o laço ilustrado na Figura 2 que retroalimenta a camada é conhecida como conexão recorrente.

Uma rede de múltiplas camadas podem, de forma sequencial, receber elementos de camadas anteriores como entrada, como pode ser visto na Figura 3.

Assim, cada camada recebe o próximo elemento ao longo de uma sequência de entrada e,

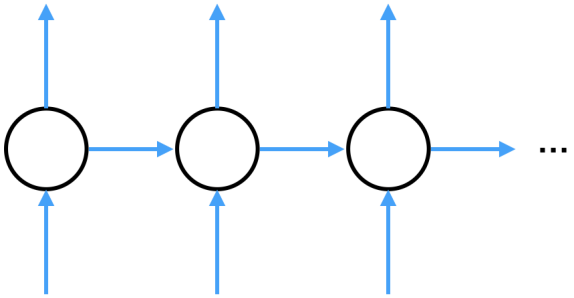


Figura 3: Uma rede neural com várias camadas de rede neural recorrente. Figura adaptada de [7].

ao mesmo tempo, recebe a saída da camada anterior à ela. Desse modo, podemos visualizar como as camadas processam uma sequência, um elemento por vez.

Apesar da rede neural recorrente possuir a vantagem de aprender relações com o item anterior, ela possui uma grande limitação: ela aprende relação apenas com o item imediatamente anterior a ele. Para casos em que há relações entre os dados à longa distância (como é o exemplo deste projeto), a rede neural recorrente não é adequada para lidar com o relacionamento entre os dados à longa distância.

Para contornar essa restrição, foi então utilizada uma rede neural recorrente otimizada conhecida como *Long Short-Term Memory* (LSTM). Podemos dizer que ela é uma variante aprimorada da rede neural recorrente, pois a mesma possui uma memória interna. Desse modo, ela é capaz de lidar com relacionamentos separados por distâncias maiores na sequência, ou seja, em vez de processar apenas sua entrada e saída anterior, ela também considera o conteúdo dentro de sua memória interna. Para isso, além de produzir uma saída usual, ela também atualiza o valor de dentro da memória. Uma ilustração de uma LSTM pode ser vista na Figura 4.

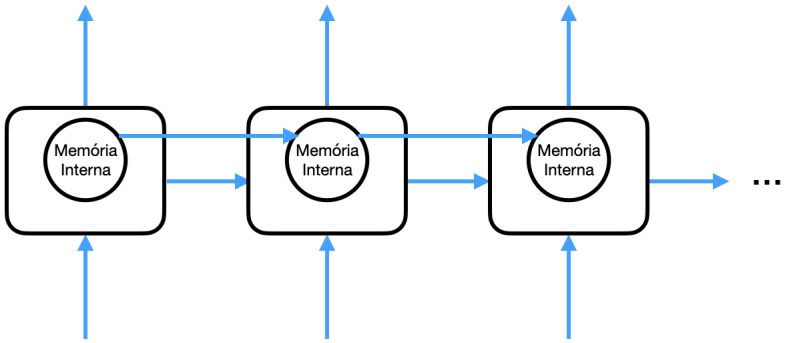


Figura 4: Uma rede neural com várias camadas LSTM. Figura adaptada de [4].

Portanto, redes neurais do tipo LSTM possuem uma eficácia muito melhor em relação

a uma rede neural recorrente simples, além de serem treinadas de forma mais eficaz. Por isso, a LSTM foi escolhida como uma alternativa de solução para o problema abordado neste projeto, visto que o mesmo possui essa característica intrínseca de possuir dados correlacionados distantes entre si.

3 Métodos

Estas seção descreve o desenvolvimento do projeto como um todo. A primeira parte se refere a uma aplicação iOS que tem como principal objetivo coletar dados de forma supervisionada, ou seja, através da aplicação será possível armazenar valores do acelerômetro e do giroscópio de uma determinada atividade humana. Em seguida, com os dados coletados, é feito um tratamento desses valores para auxiliar a criação dos modelos de aprendizado de máquina. Por fim, será demonstrada a aplicação iOS que, de fato, utiliza os modelos para prever atividades humanas.

3.1 Coleção de Dados

Antes de criar o conjunto de dados, foi necessário pré-estabelecer os movimentos de atividades humanas que o modelo irá prever e identificar. Para este projeto, a atividade humana que é reconhecida são quedas físicas de uma pessoa. Como há vários modos de uma pessoa entrar em queda, este projeto selecionou quatro tipos de quedas abruptas separadamente (Figura 5), sendo elas:

1. queda para frente;
2. queda para trás;
3. queda na lateral esquerda;
4. queda na lateral direita.

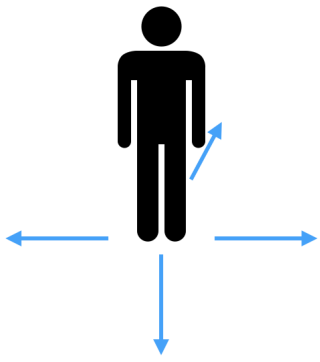


Figura 5: Quatro tipos de queda.

Além dos tipos de quedas, foi definido que nos quatro movimentos temos três estados em ordem cronológica que compõem uma queda: estar em pé, queda abrupta e estar deitado.

Sendo assim, a pessoa deve ficar parada por alguns instantes em pé, realizar, de fato, o movimento da queda abrupta e, em seguida, ficar deitada e imóvel. A Figura 6 ilustra os três estados.

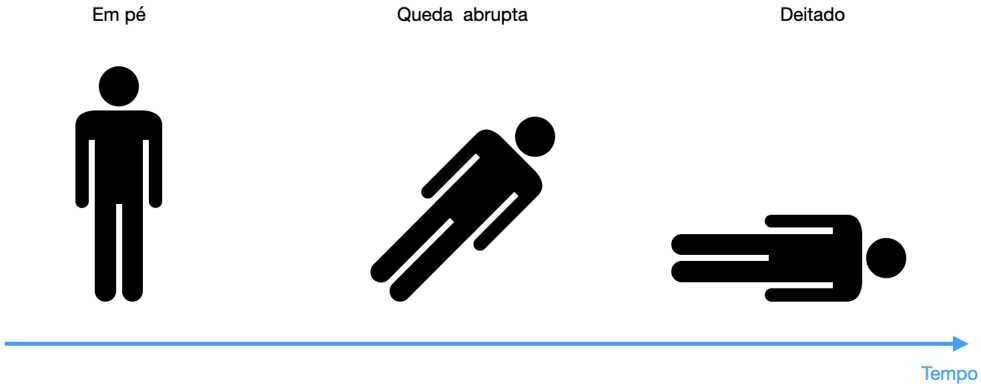


Figura 6: Sequência cronológica de uma queda e seus três estados possíveis (em pé, queda e deitado).

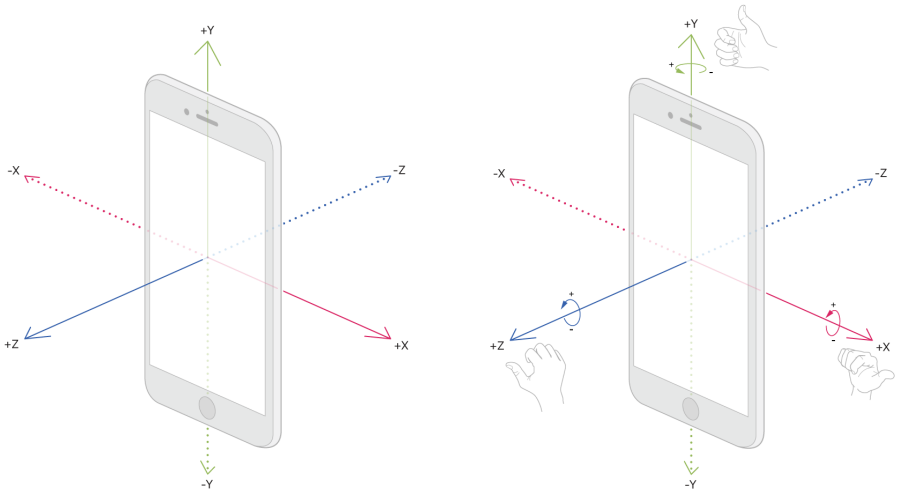
Definido o conjunto de atividades humanas a serem reconhecidas pelo modelo, a próxima seção explica em detalhes como os dados coletados foram armazenados utilizando um *smartphone*.

3.1.1 Gravando Dados

Antes de explicar como os dados foram coletados, é necessário explicar mais detalhes sobre os valores que serão armazenados pelos sensores. Os valores reportados pelo acelerômetro são medidos em incrementos da aceleração gravitacional, com o valor 1.0 representando uma aceleração de 9,8 metros por segundo em uma certa direção. Sendo assim, os valores de aceleração podem ser positivos ou negativos dependendo da direção da aceleração. Da mesma maneira, os valores reportados pelo giroscópio são medidos em radianos por segundo, os quais também podem ser positivos ou negativos dependendo da direção da rotação. A Figura 7 ilustra os eixos X, Y e Z que definem os sensores de acelerômetro e giroscópio.

Dadas as devidas explicações sobre os sensores e seus valores, para obter o conjunto de dados para treinar nosso modelo de aprendizado de máquina, foi necessário desenvolver uma aplicação iOS que armazenasse os valores do acelerômetro e giroscópio. Basicamente, o principal objetivo da aplicação é capturar os valores dos sensores e, ao final de uma gravação, gerar um arquivo de extensão `csv` armazenado localmente no celular.

Como pode ser observado na Figura 8, a interface do aplicativo possui alguns campos de entradas que permitem a gravação automática dos dados de forma supervisionada. O primeiro campo requer a entrada de um identificador numérico do usuário, o qual permite identificar os arquivos gravados pelo mesmo. Em seguida, o usuário escolhe o tipo de queda que irá realizar e o número de vezes que deseja repetir o movimento em um intervalo de 1



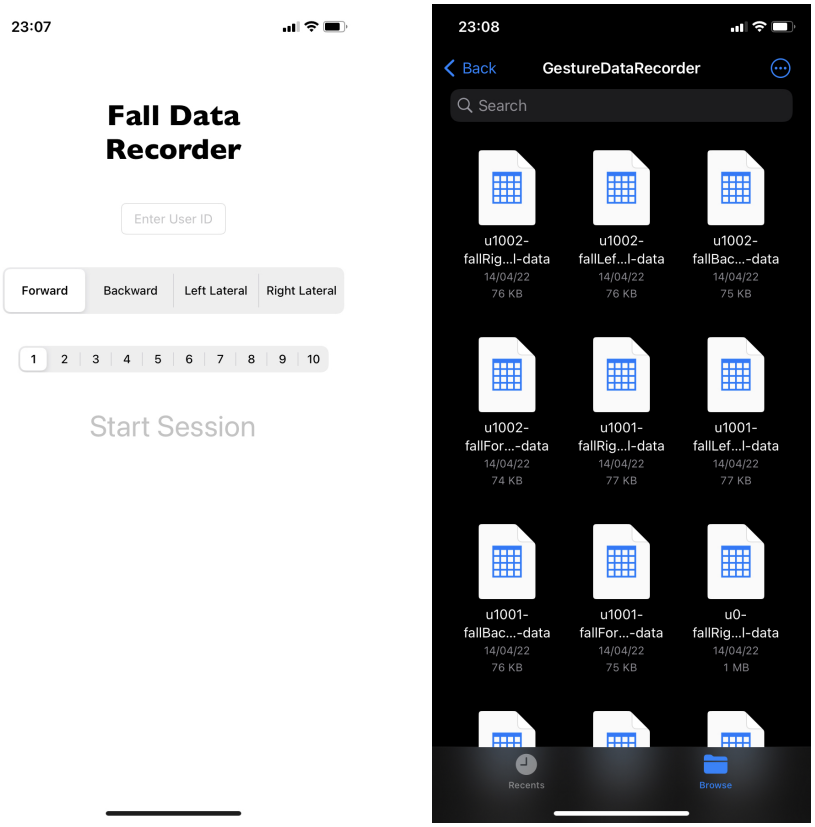
(a) Eixos X, Y e Z do acelerômetro. (b) Eixos X, Y e Z do giroscópio.

Figura 7: Eixos de um *smarthphone*. Figuras extraídas de [3, 1, 2].

a 10 vezes. Por fim, temos um botão (*'Start Session'*) que ativa os sensores e, então, grava os dados dos sensores da queda realizada por um determinado intervalo de tempo.

Dessa forma, foi possível exportar os arquivos com extensão `csv` para visualizar e manipular esses dados para a criação dos modelos. Entretanto, antes de entendermos mais detalhadamente como foi o tratamento dos dados coletados, é necessário salientar algumas tomadas de decisões e informações que são importantes para a criação do modelo de aprendizado de máquina. São elas:

- Em todas as sessões para coletar os movimentos de queda, foi pré estabelecido que o dispositivo deveria obrigatoriamente estar posicionado na altura da cintura do lado direito. Além disso, o celular também deveria estar com a tela orientada ao corpo da pessoa com a câmera posicionada para baixo. Essa restrição foi determinada em virtude de diminuir o número total de medições, já que, se o celular estivesse posicionado arbitrariamente, o modelo teria dificuldade de identificar os movimentos;
- O intervalo de tempo para capturar as quedas foi de 18 segundos;
- Ao coletar os movimentos, o dispositivo registrava os valores dos sensores 10 vezes por segundo. Desse modo, uma única sessão continha 180 linhas de dados (intervalo de tempo em segundos multiplicado pela quantidade de medições por segundo);
- Além dos quatro movimentos estabelecidos, o aplicativo também registrava um "movimento" arbitrário dos sensores entre uma gravação e outra. Esses registros foram essenciais para que o modelo identificasse quando o usuário não realizava uma queda de fato. Sendo assim, podemos chamar essa atividade humana como "Não queda";
- Para realizar a gravação dos dados, dez pessoas se voluntariaram para fazer as medições seguindo protocolos de segurança que garantissem a saúde de todos os indivíduos. Tal grupo de voluntários está dividido entre homens e mulheres de 19 a



(a) Interface do aplicativo iOS auxiliar.

(b) Arquivos do tipo CSV gravados localmente no celular.

Figura 8: Aplicativo iOS auxiliar para colecionar medições dos sensores.

29 anos. Ao todo, foram coletadas 84 medições balanceadas entre os quatro tipos de quedas pré-definidos. A razão de abranger o número de fontes na coleta de dados é uma tentativa de evitar qualquer enviesamento no modelo.

3.2 Tratamento do Conjunto de Dados Coletados

Após a coleta de dados, os arquivos de extensão `csv` gravados localmente foram exportados para um computador. Através de um ambiente virtual e utilizando a linguagem *Python* com versão 3.8, foi possível visualizar e manipular esses dados. A Figura 9 exemplifica como os dados crus estão em um primeiro momento sem qualquer tipo de tratamento.

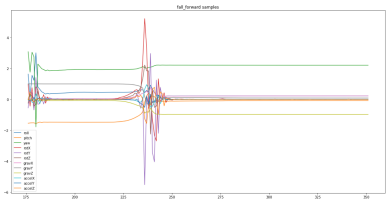
Como podemos ver, a primeira coluna se refere ao tipo de queda. Seus valores variam de 0 a 4, em que cada número representa um tipo de queda. Em seguida, as restantes colunas são os dados do acelerômetro e do giroscópio, já mencionados anteriormente.

A partir dos dados, foi possível visualizar os valores de cada queda separadamente. Com isso, utilizando *scripts* em linguagem Python, gerou-se gráficos dos quatro tipos de quedas,

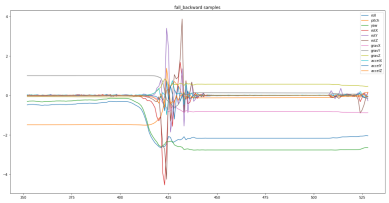
activity	rotX	rotY	rotZ	gravX	gravY	gravZ	accelX	accelY	accelZ
0	0.276131	2.082880	0.058157	-0.454387	0.889336	-0.051121	-1.135290	1.104000	2.338330
0	0.029048	0.197242	0.131645	-0.455392	0.889886	-0.026852	0.297739	-0.097704	0.048106
0	-0.015064	0.785494	-0.564496	-0.457122	0.886218	-0.075214	-0.469389	-0.028587	0.616908
0	0.069010	0.184039	0.564379	-0.470473	0.881210	-0.046088	0.799795	-0.473838	-0.027005

Figura 9: Dados crus de um movimento.

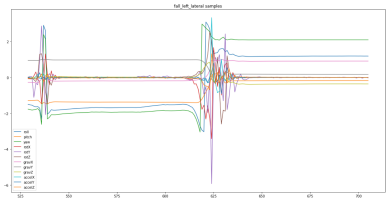
como observado na Figura 10.



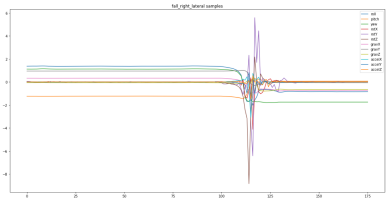
(a) Queda para frente.



(b) Queda para trás.



(c) Queda na lateral esquerda.



(d) Queda na lateral direita.

Figura 10: Exemplos dos quatro tipos de quedas que mostram os dados de acelerômetro e de giroscópio em relação ao tempo.

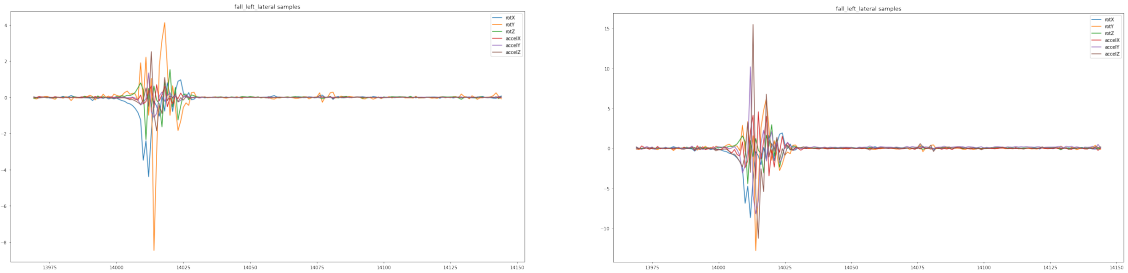
Ao observarmos os gráficos, ficam evidentes os três estados que compõem os quatro tipos de queda. Em um primeiro momento, é possível visualizar quando a pessoa fica por alguns instantes em pé devido aos valores constantes em relação ao tempo. Em seguida, vemos uma forte perturbação nos sinais dos sensores, o qual claramente representa o movimento da queda abrupta. Por fim, assim como o estado estar de pé, o último estado logo após a queda mostra valores constantes em relação ao tempo, porém com valores diferentes ao primeiro estado. O último estado representa quando a pessoa está deitada ao chão.

Com as visualizações, foi possível notar que uma queda, apesar de suas variações, possui um comportamento semelhante. Assim, podemos salientar que uma queda é dada por uma grande variação da aceleração e do giroscópio em relação a um eixo. Esse comportamento padrão no conjunto de dados foi um bom sinalizador que realmente poderíamos, de fato, aplicar alguma solução de aprendizado de máquina para esse problema, visto que o mesmo seria capaz de reconhecer essas similaridades e prever quando um movimento é uma queda

ou não.

Apesar do padrão entre as medições, foi possível notar que algumas medições continham valores atípicos. Esses ruídos podem ser visualizados nas Figuras 10(a), (c) e (d). Como podemos notar, há picos de valores no primeiro e no terceiro estágios desses movimentos. Como ambos os estágios retratam estados de inércia, esses valores devem ser atenuados ou até mesmo removidos porque são incoerentes com o movimento.

Com o objetivo de fornecer um conjunto de dados mais limpo possível para treinar nossos modelos, foi utilizado então um método de pré-processamento de padronização dos dados de forma a atenuar os ruídos. Fazendo o uso da biblioteca *Scikit-Learn*, o método *StandardScaler* permitiu padronizar os dados de cada medição. O método passa por cada valor e subtrai da mesma a média da coluna. Em seguida, divide-se o resultado da subtração pelo desvio padrão.



(a) Dados não padranizados. (b) Dados padronizados.

Figura 11: Atenuando valores atípicos através do método *StandardScaler*.

Analisando a Figura 11, nota-se que alguns valores atípicos (principalmente após a queda) são atenuados quando comparamos um exemplo de uma queda com os dados crus em relação aos dados padronizados pelo método do *Scikit-Learn*. Além disso, a natureza padrão de uma queda não foi alterada, pois ainda é nítido o intervalo do tempo onde ocorreu o movimento.

Após a visualização e o tratamento desses dados, foi possível, então, treinar modelos a partir desse conjunto. A próxima seção explicará detalhadamente como foi o processo de criar modelos de aprendizado de máquina.

3.3 Treinando os Modelos

Uma vez que o conjunto de dados foi construído e tratado, criar os modelos de aprendizado de máquina foi relativamente mais simples se comparado com o processo de coleta dos dados dos movimentos. Além do mais, com o auxílio das bibliotecas do *Scikit-Learn* e do *Keras*, o processo da criação dos modelos ficou menos complexo.

O *Scikit-Learn* foi usado para a criação do modelo utilizando o algoritmo de Árvore de Decisão, enquanto o *Keras* foi utilizado para treinar a rede neural da variante *Long Short-Term Memory* (LSTM). Em ambos os processos, o modelo criado foi convertido para um modelo do tipo *Core ML*. Essa conversão foi necessária com o objetivo de inserir os modelos

nos sistemas operacionais da Apple que, em nosso caso, é um *smartphone*.

3.3.1 Árvore de Decisão

Para o algoritmo de árvore de decisão, separou-se o conjunto em duas partes: conjunto de treinamento e conjunto de teste. O primeiro conjunto recebeu 70% do conjunto total de dados, enquanto o segundo ficou com 30% restante.

No caso desse algoritmo, há dois parâmetros importantes para sua criação: profundidade da árvore (*depth*) e o número de folhas (*leaf*). Para deduzir quais os melhores parâmetros para criar o modelo, foi necessário iterar os números de folhas pela profundidade. Para isso, escolheu-se iterar a profundidade em uma altura de 1 a 40 e com um número de folhas variando entre 1 a 3. Assim, a cada iteração, foi gerada uma predição e sua respectiva precisão. A Figura 12 ilustra a precisão do modelo usando árvore de decisão de acordo com sua profundidade.

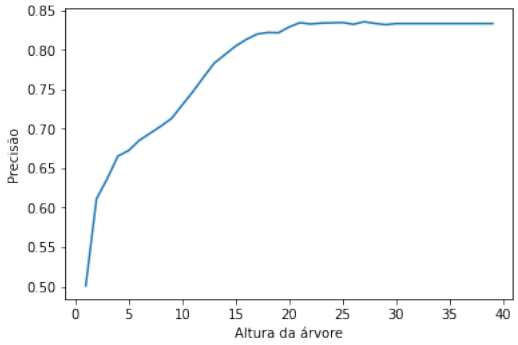


Figura 12: Dados crus de um movimento.

Como podemos perceber, há uma saturação dos dados a partir da altura 20, em que a precisão torna-se aproximadamente 82%. Por isso, foi determinado que o modelo final utilizando o algoritmo de árvore de decisão teria uma profundidade de altura 20 e com uma folha. Na Seção 5.1, outros resultados serão mostrados sobre o modelo gerado.

3.3.2 Long Short-Term Memory

Assim como o treinamento do modelo utilizando árvore de decisão, o conjunto de dados total foi dividido entre um conjunto de treinamento e um conjunto de testes, sendo que o primeiro recebeu agora 80% dos dados e o segundo com os 20% restantes. Entretanto, para criar a rede neural recorrente, foi necessário rearranjar o conjunto de dados.

Como explicitado anteriormente, há três tipos de sinais principais em nossos dados: aceleração total, aceleração do corpo e rotação do corpo. Cada um deles tem três eixos de dados, o que significa que há nove variáveis a cada instante em que os dados são coletados. Dessa forma, em vez de utilizarmos os dados como uma tabela mostrada na Figura 9, os dados foram empilhados com o intuito de construir uma matriz com três dimensões. A

matriz possui o seguinte formato:

$$[valor, time\ steps, features]$$

tal que:

1. a primeira dimensão possui tamanho igual ao número de medições feitas (84);
2. a segunda dimensão possui tamanho 180, o qual representa a quantidade de medições realizadas pelos sensores durante uma única sessão;
3. a terceira dimensão são as nove variáveis referentes aos movimentos (rotação no eixo X, rotação no eixo Y e assim por diante).

Com os dados adaptados, criou-se o modelo utilizando o pacote *Keras*. O modelo criado é do tipo sequencial, o qual permite empilhar camadas na rede neural em ordem. Basicamente, a primeira camada se refere à camada LSTM. Em seguida, uma camada do tipo *dropout* foi inserida com o intuito de reduzir o *overfitting*, ou seja, evitar que o modelo comece a se lembrar de exemplos específicos usados no treinamento. Por fim, a penúltima camada é responsável por interpretar os valores da camada LSTM e a última camada é responsável por gerar a predição.

Ao final desse processo, o modelo foi compilado com sucesso. Parâmetros de qualidade da rede neural serão mostrados na Seção 5.2.

3.4 Predições em Tempo Real

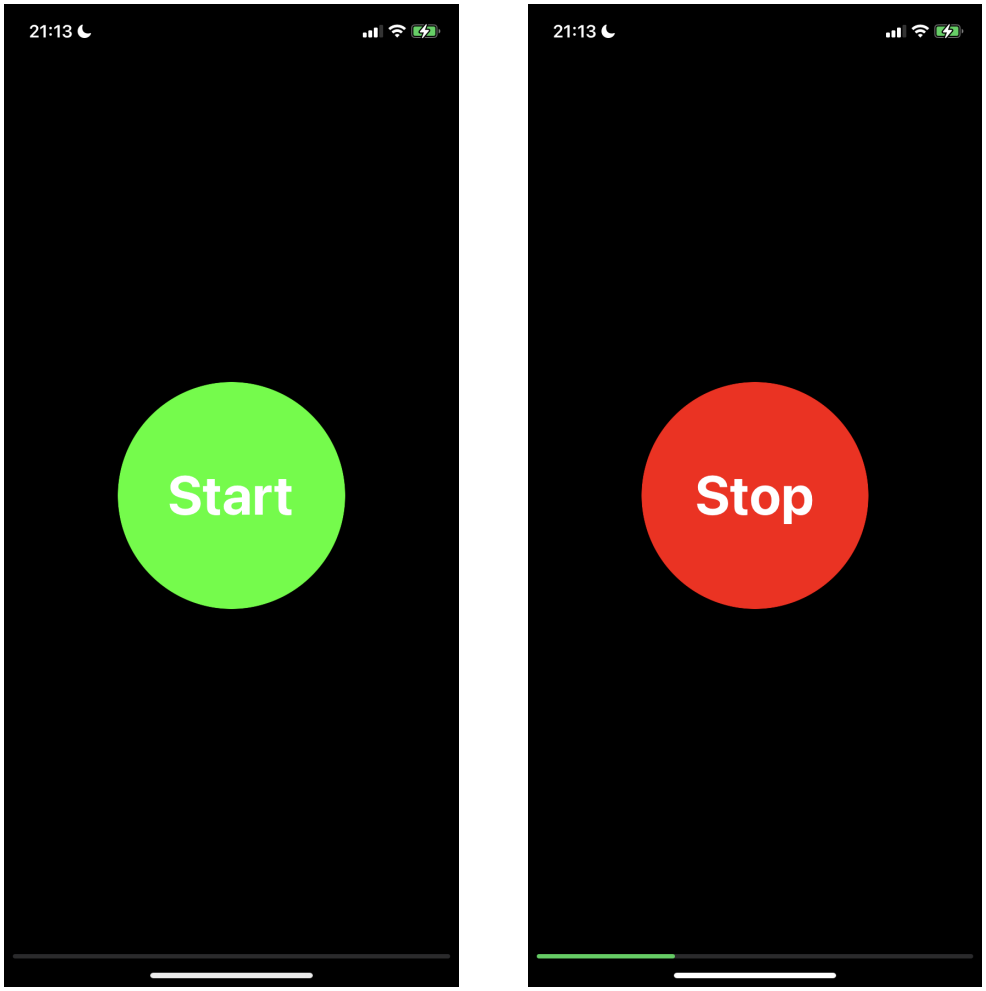
Com o intuito de utilizar os modelos criados na Seção 3.3, foi desenvolvido um aplicativo iOS que é capaz de identificar uma queda e seu tipo.

Assim como o aplicativo de coleta de dados, esta aplicação também ativa os sensores de acelerômetro e de giroscópio e possuem as mesmas configurações, ou seja, os dados são coletados 10 vezes por segundo e o tempo total do movimento deve ser de 18 segundos. Além disso, o aplicativo informa, através de comandos por áudio, qual é a posição correta do celular e quantos segundos foram percorridos até então.

Como podemos ver na tela da Figura 13(a), o projeto possui dois componentes majoritários: um botão e uma barra de progresso. O botão tem como objetivo ativar, de fato, os sensores, enquanto a barra de progresso tem a finalidade de mostrar o tempo que foi percorrido. A Figura 13(b) ilustra o estado da aplicação quando os valores dos sensores estão sendo observados. Nela, podemos ver que o botão inicial é substituído por um botão de parada, cujo objetivo é reiniciar uma sessão. Ao final da sessão, o aplicativo mostra a predição do modelo a partir dos dados visto durante os 18 segundos.

4 Experimentos

Com o aplicativo desenvolvido, foi possível então realizar experimentos com ambas abordagens. Desse modo, quatro tipos de quedas foram realizadas para cada modelo.



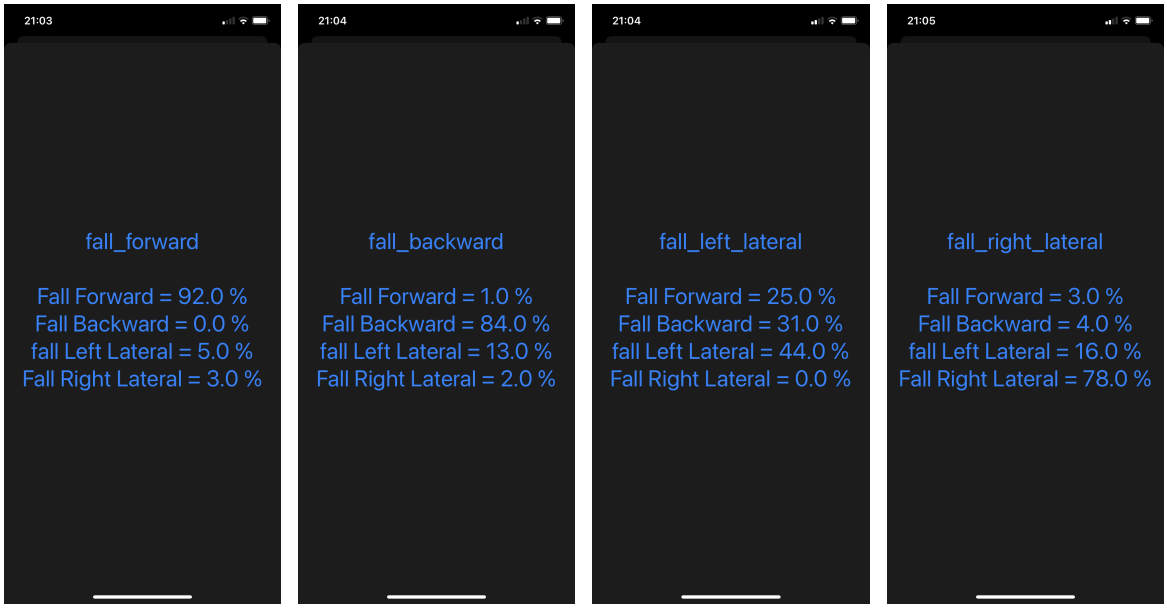
(a) Aplicativo antes de iniciar uma sessão. (b) Aplicativo durante uma sessão

Figura 13: Aplicativo iOS que utiliza os modelos de aprendizado de máquina.

4.1 Predições com Árvores de Decisão

A Figura 14 mostra os resultados do aplicativo usando o modelo de predição com o algoritmo de árvores de decisão. A partir das imagens, pode-ser perceber que as precisões dos movimentos da queda para frente, queda para trás e queda na lateral direita possuem um valor relativamente satisfatório, pois possuem uma precisão acima dos 78%.

Por outro lado, o movimento de queda na lateral esquerda teve o pior desempenho, com uma precisão de 44%. Não obstante, para o mesmo movimento, o modelo nos indica que o movimento tem 25% de chance de ser queda para frente e 31% de chance de ser queda para trás. Esse fato demonstra que o modelo ficou confuso em diferenciar esse movimento.



(a) Queda para frente com 92% de precisão.

(b) Queda para trás com 84% de precisão.

(c) Queda na lateral esquerda com 44% de precisão.

(d) Queda na lateral direita com 78% de precisão.

Figura 14: Experimentos utilizando o modelo com algoritmo de árvores de decisão.

4.2 Predições com LSTM

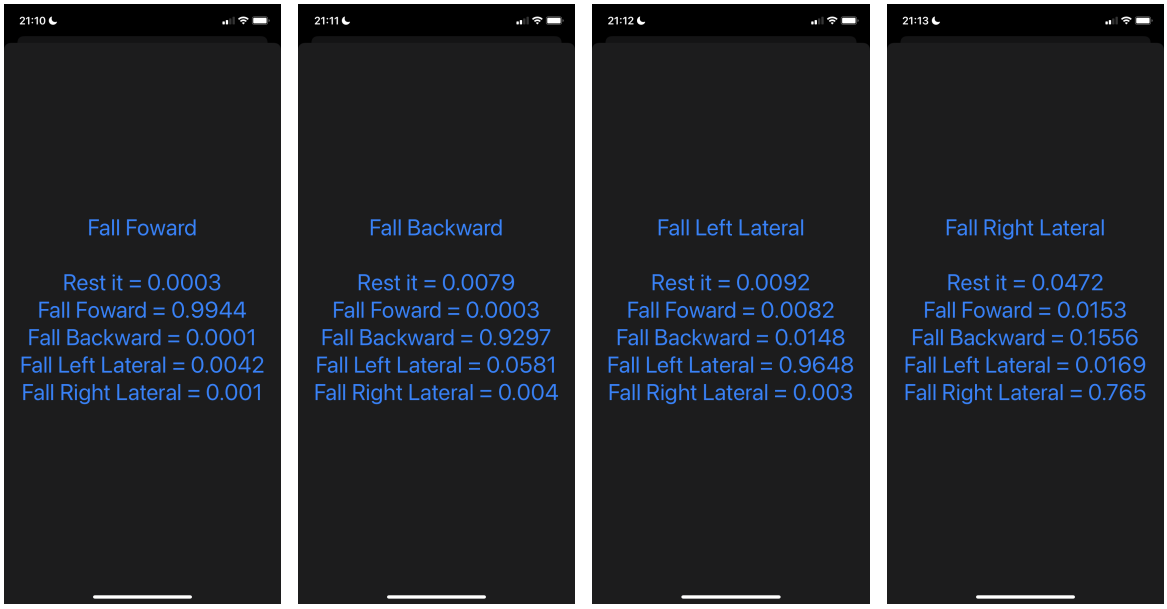
A Figura 15 mostra os resultados obtidos para cada tipo de movimento fazendo-se o uso da rede neural recorrente LSTM. Como podemos perceber a partir deste experimento, todos os movimentos possuem uma precisão relativamente satisfatória, visto que o intervalo desse índice está entre 76% a 99%.

5 Resultados

Esta seção reporta os resultados obtidos a partir de cada modelo com relação aos parâmetros de qualidade. Sendo assim, será possível relacionar o que foi observado nos experimentos com as métricas extraídas dos seus respectivos modelos.

5.1 Métricas do Modelo Utilizando Árvore de Decisão

Um fato importante de salientar na solução utilizando o modelo de árvore de decisão é que as predições realizadas são pontuais. Em outras palavras, a partir dos noves valores provenientes dos sensores de acelerômetro e giroscópio, foi possível fazer uma predição indicando qual o movimento desses dados de entrada. Porém, como enfatizado em seções anteriores, o algoritmo ser capaz de identificar qual o tipo de queda em um instante não



(a) Queda para frente com 99% de precisão.

(b) Queda para trás com 92% de precisão.

(c) Queda na lateral esquerda com 96% de precisão.

(d) Queda na lateral direita com 76% de precisão.

Figura 15: Experimentos utilizando a rede neural LSTM.

é capaz de solucionar o problema, visto que os dados possuem dependência com dados de instantes anteriores.

Diante desse problema, o projeto contornou essa dificuldade definindo que a predição final é a moda das predições, ou seja, a predição é o valor mais frequente das predições totais que o modelo previu. Para ficar mais claro, considere o seguinte exemplo: como sabemos, em uma única janela de predição, há 180 marcações dos dados dos sensores. Desse modo, temos que o modelo com árvore de decisão fez 180 predições. Do conjunto de 180 predições, 100 resultaram que o movimento era queda para frente, 40 indicaram que o movimento não era uma queda, 20 para queda na lateral direita, 20 para queda na lateral esquerda e. em nenhum momento. o modelo previu queda para trás. Portanto, como o modelo previu mais vezes queda para frente (100 vezes), a predição final será considerada queda para frente.

Dito isso, uma das métricas que o modelo de árvore de decisão possui é referente a sua acurácia. A acurácia desse modelo foi de 83%, ou seja, a eficácia geral do modelo revelou que existe uma grande proximidade do resultado previsto com seu valor real. Uma visão geral da acurácia de cada movimento pode ser observada na Figura 16.

Além disso, a matriz de confusão (Figura 17) foi gerada a partir dos dados de saídas do conjunto de testes com as saídas preditas pelo modelo criado. Essa matriz coloca as classes preditas versus os rótulos das classes reais dos movimentos. Ela é muito útil, pois mostra cenários de problemas potenciais para o modelo. Sendo assim, é possível visualizar onde o modelo tende a cometer os erros. No geral, é esperado ver muitos valores altos na diagonal

activity	acertos	total	erros	% acertos
fall_backward	891	1118	227	79.695886
fall_forward	980	1130	150	86.725664
fall_left_lateral	950	1140	190	83.333333
fall_right_lateral	933	1086	153	85.911602
rest_it	3262	3989	727	81.774881

Figura 16: Acurácia de cada movimento.

da matriz, pois essas são as correspondências corretas.

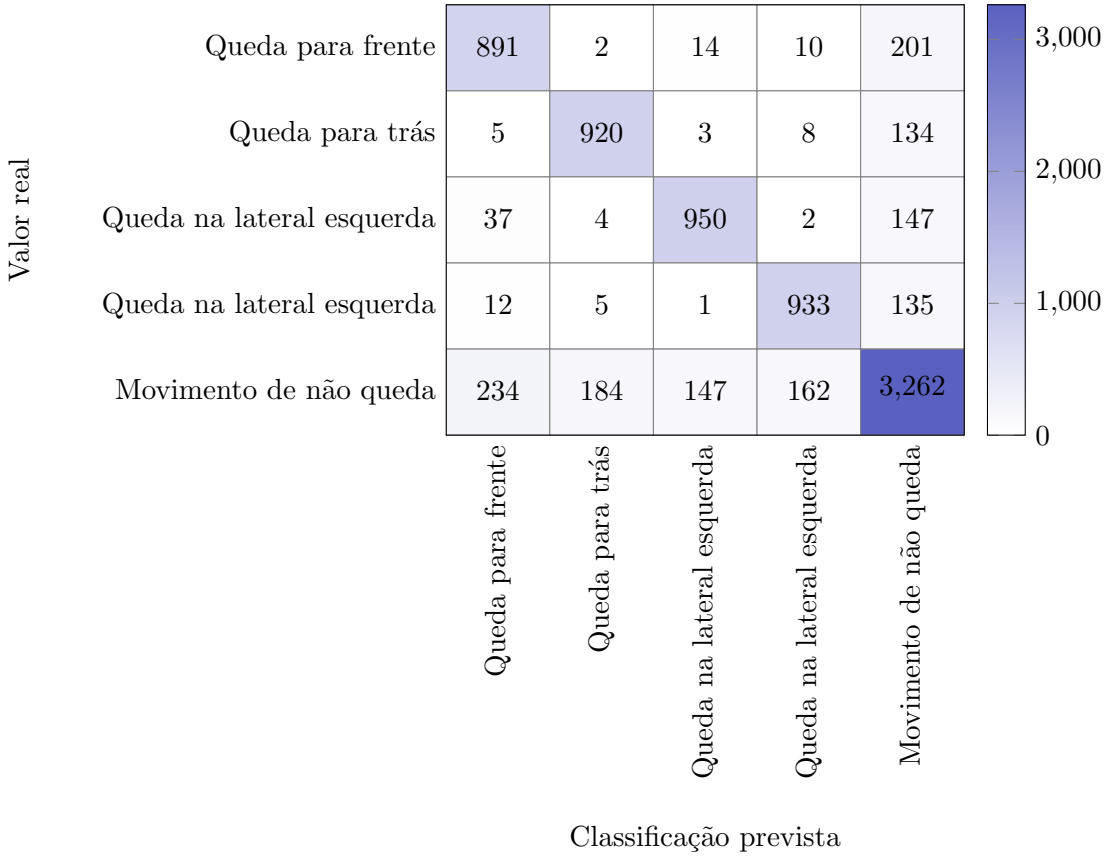


Figura 17: Matriz de confusão para a classificação dos movimentos.

A partir da matriz de confusão desse modelo em particular, fica claro que ele já aprendeu a maioria dos movimentos, já que a diagonal da matriz se destaca, mas ainda está longe de ser perfeita. Entretanto, pode-se notar que em todo os quatro tipos de quedas há uma

quantidade significativa de predições que são preditas erroneamente como um movimento de não queda. Essa confusão entre os movimentos se deu pelo fato de que o movimento de "não queda" ser muito parecido com o primeiro estágio das quedas (estado de ficar em pé). Por isso, e em muitas situações, uma queda foi considerada incorretamente como um movimento qualquer, pois o movimento de "não queda' foi apenas ficar em pé.

5.2 Métricas da Rede Neural Recorrente LSTM

Assim como o modelo de árvore de decisão, também foi calculada a acurácia da rede neural recorrente do tipo LSTM. Para tal, construiu-se várias vezes a rede neural para avaliar a capacidade do modelo. A razão para isso é que as redes neurais são estocásticas, o que significa que um modelo específico diferente resultará ao treinar a mesma configuração de modelo nos mesmos dados.

Diante disso, repetiu-se a avaliação do modelo várias vezes e, em seguida, deu-se uma visão geral do desempenho do modelo em cada uma dessas execuções. Para isso, foi resumido a amostra de pontuações calculando e relatando a média e o desvio padrão do desempenho. A média forneceu a precisão média do modelo no conjunto de dados, enquanto o desvio padrão forneceu o desvio médio da acurácia em relação à média. Portanto, ao iterar e criar 100 vezes a rede neural LSTM utilizando o mesmo conjunto de dados, a acurácia da rede neural LSTM foi:

$$\text{Acurácia} = 69.86\% \pm 7.53.$$

6 Conclusões e Trabalhos Futuros

Neste projeto, podemos constatar a grande importância do conjunto de dados, que serviu para a construção dos modelos. Esta etapa consumiu grande parte do desenvolvimento deste trabalho. É notório enfatizar que trabalhar com dados sequenciais é uma tarefa mais complexa quando comparada com dados pontuais, pois exige que o modelo a ser criado entenda a relação existente entre os dados e não apenas a gerar uma predição com os dados apresentados em um único instante.

Em termos de praticidade, a solução da rede neural recorrente do tipo LSTM foi adequada e apropriada em relação ao modelo usando o algoritmo de árvore de decisão. Apesar do modelo utilizando o algoritmo mais simples possuir uma maior acurácia comparada ao modelo usando a rede neural (83% contra aproximadamente 70%), sua solução não identifica as dependências entre os dados e faz apenas predições pontuais. Por outro lado, o LSTM é capaz de identificar essas relações e realiza uma única predição observando os valores do movimento como um todo.

Em termos de usabilidade, o aplicativo com o modelo de árvore de decisão possui deficiências. Como a solução se baseou na moda das predições, isso acarretou uma incerteza na predição final e resultou em uma experiência de usuário ruim. Um exemplo dessa imperfeição ocorre quando o usuário está fazendo algum movimento e apenas no final da janela de predição realiza a queda. Como a predição final é a moda, mesmo que o usuário realize o movimento de queda, a predição final acabará indicando que o movimento não foi de queda, visto que a maioria das predições indicou o mesmo. Em contrapartida, a solução com a

LSTM não teve o mesmo problema, sendo possível realizar a queda em qualquer instante dentro da janela de predição e obtendo uma melhor experiência.

Em relação à eficácia de cada modelo, a justificativa para que o modelo de árvore de decisão possuir uma maior acurácia se deve ao fato de que, ao treinar o modelo, o algoritmo tratava os dados pontualmente. Enquanto isso, para treinar a rede neural com LSTM, foi necessário comprimir cada sessão em uma única linha de dado. Numericamente, o primeiro modelo possuía aproximadamente 30 mil dados para treinamento, enquanto o segundo comprimiu esse grande número de dados para 84 linhas. Apesar da diferença, ambas soluções tiveram eficácia satisfatória.

Finalmente, este relatório sugere algumas direções para trabalhos futuros. Com o intuito de aumentar o desempenho da rede neural, a principal tarefa a ser realizada é aumentar a base de dados considerando os seguintes fatores: (i) variar mais a disposição do celular em relação a sua orientação e posição no corpo, (ii) variar os tipos de quedas, (iii) variar o movimento de "não queda" para evitar confusão do modelo entre as quedas e (iv) diminuir o tempo da janela de predição. Além disso, como o problema deste projeto foi referente a monitorar pessoas suscetíveis a quedas, um possível trabalho futuro é notificar algum agente (por meio de uma mensagem ou até mesmo uma ligação) para que alguma providência possa ser tomada e prestar socorro ao monitorado.

Referências

[1] Apple. Getting Raw Accelerometer Events, 2022. `https://developer.apple.com/documentation/coremotion/getting_raw_accelerometer_events`.

[2] Apple. Getting Raw Gyroscope Events, 2022. `https://developer.apple.com/documentation/coremotion/getting_raw_gyroscope_events`.

[3] Apple. Gyroscope and Accelerometer, 2022. `https://developer.apple.com/design/human-interface-guidelines/inputs/gyro-and-accelerometer/`.

[4] J. Brownlee. LSTMs for Human Activity Recognition Time Series Classification, 2022. `https://machinelearningmastery.com/how-to-develop-rnn-models-for-human-activity-recognition-time-series-classification/`.

[5] A. Gallagher, M. Hollemans, A. Tam, and C. LaPollo. *Machine Learning by Tutorials: Beginning Machine Learning for Apple and iOS*. Razeware LLC, 2021.

[6] A. Géron. *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow - Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, 2019.

[7] S. Shalev-Shwartz and S. Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.