UNIVERSIDADE FEDERAL FLUMINENSE

RAFAEL RÊGO DRUMOND

PEEK: CLASSIFICAÇÃO DE MOVIMENTO UTILIZANDO DADOS ESPARSOS DE MEMBROS SUPERIORES EM REDES COM LSTM

NITERÓI

2017

UNIVERSIDADE FEDERAL FLUMINENSE

RAFAEL RÊGO DRUMOND

PEEK: CLASSIFICAÇÃO DE MOVIMENTO UTILIZANDO DADOS ESPARSOS DE MEMBROS SUPERIORES EM REDES COM LSTM

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Computação Visual

Orientador:
Prof. Dr. ESTEBAN WALTER GONZALEZ CLUA

Co-orientadora:
Profa. Dra. CRISTINA NADER VASCONCELOS

NITERÓI

2017

Ficha Catalográfica elaborada pela Biblioteca da Escola de Engenharia e Instituto de Computação da UFF

D795 Drumond, Rafael Rêgo
PEEK: classificação de movimento utilizando dados esparsos de membros superiores em redes com LSTM / Rafael Rêgo Drumond. – Niterói, RJ : [s.n.], 2017.
55 f.

Dissertação (Mestrado em Computação) - Universidade Federal Fluminense, 2017.
Orientadores: Esteban Walter Gonzalez Clua, Cristina Náder Vasconcelos.

1. Jogo em computador. 2. Aprendizado de máquina. 3. Rede neural. I. Título.

CDD 006.6

RAFAEL REGO DRUMOND

PEEK: CLASSIFICAÇÃO DE MOVIMENTO UTILIZANDO DADOS ESPARSOS DE MEMBROS SUPERIORES EM REDES COM LSTM

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Computação Visual

Aprovada em 14 de março de 2017.

BANCA EXAMINADORA

Prof. Esteban Walter Gonzalez Clua - Orientador, UFF

Profa. Cristina Nader Vasconcelos - Co-orientadora, UFF

Profa. Aline Marins Paes Carvalho, UFF

Prof. Luis Martí Orosa, UFF

Profa. Soraia Raupp Musse, PUCRS

Niterói

2017

A todos aqueles expulsos de casa por terem nascido diferentes.

Agradecimentos

Em primeiro lugar a Deus, por toda iluminação que tem me dado durante minha vida.

Aos meus pais Alberto e Celi, por me apoiarem nos meus estudos e por (tentarem) aguentar a saudade.

Aos meus irmãos Lucas e Kirsten, pelo carinho, amizade e por sempre me ajudarem e me darem apoio quando precisei.

Aos meus "irmãos de mentira": Enio, Bernardo e Gentil, por serem irmãos, amigos, por me escutarem e sempre prontos para me darem forças.

Para minhas "irmãs de mentira": Carol e Mika. Por estarem sempre me ouvindo reclamar da vida, rindo disso, e fazendo comida comigo ao mesmo tempo.

À minha família, com quem cresci e me ensinou valores em que acredito até hoje. Em especial para minha madrinha Taty, e meu primo João Miguel. Pelo amor incondicional que recebi desde sempre.

Ao Flávio, pela compreensão, carinho e por sempre me ajudar a encontrar paz nos momentos mais difíceis.

Ao meu orientador da graduação, Carlos Salles e a todos os membros do antigo LAWS, pelo apoio dado para que eu chegasse até aqui.

Aos meus orientadores do mestrado Esteban e Cristina. Por terem doado tempo precioso para me auxiliar quando mais precisei.

Aos membros do Medialab, pela ajuda e por serem voluntários nesse trabalho.

Aos amigos do grupo Proudly Nerd. Por sempre me ajudarem a entender que nunca devemos ter vergonha de quem somos.

Ao meus amigos dos "Technonerds", André, Lucas, e Rodrigo, pela amizade mais duradoura e por nunca perderem o contato.

Aos meus senseis Omar, Eraldo, Márcio, André, Pedro, Kataoka e Tsustumi, e aos meus colegas do Kendo do mundo inteiro, pelo espírito de família, por me ajudarem a

aprender o que é coragem e pelas lições que levarei para a vida inteira.

Para Jason Cho. Um grande amigo distante com quem sempre troco ideias sobre carreiras e futuro.

Aos amigos Bernardo, Luma, Jobson, Lima, Medeiros, Júlio, Taís, Carol, Mika, Thalles, Palma, Thadeu, Pedro, Heitor, Levi, Pablo, Alysson, Andressa, Glauco, Sato, Molinaro, Bianca, Tomo, Victor, Fernanda, Thiago, Rico, Henrique, André, Enio, Ricardo e Bruno. Por serem minha família longe de casa.

Ao Breno, Tati, Bruno, Crix, Felipe. Por serem bons quase-vizinhos, e boas companhias.

Aos meus colegas de apartamento, pela boa convivência e amizade.

A todos os meus outros amigos (que são muitos para listar aqui), e que sempre me deram apoio (e outros que me deram carona).

A todos aqueles que contribuíram para este trabalho.

Resumo

Jogos digitais e outras aplicações interativas têm permitido que usuários interajam com o jogo em diferentes formas. Atualmente jogos e aplicações estão usando muitas interfaces intuitivas diferentes, como telas sensíveis ao toque, controles com acelerômetros e com o movimento natural do corpo do usuário. Vários pesquisadores desenvolveram maneiras de processar o movimento humano, classificando a ação ou reconstruindo poses corporais com base em acelerômetros, giroscópios, sensores de calor e câmeras. Este trabalho apresenta uma nova estratégia para a classificação de movimentos de todo o corpo usando Redes Neurais Recorrentes de Memória de Curto Prazo (LSTM), usando apenas dados esparsos de dois sensores de Unidades de Medição Inerciais (IMU). Este modelo, *Peek*, demonstra eficiência obtendo uma precisão global de acertos de 96 % durante os testes.

Palavras-chave: Classificador de Movimento, Sensor IMU, Aprendizado Profundo, Redes Neurais Recorrentes, Dados Esparsos, LSTM, Aprendizado de Máquina.

Abstract

Digital Games and other interactive applications have been allowing users to interact with games in many levels. Nowadays games and applications are using many different intuitive interfaces, such as touch screens, motion controllers and body movements from the user. Several researchers have developed ways to process human motion by either classifying the action or reconstructing body poses based on accelerometers, gyroscopes, heat sensors and cameras. This work presents a novel strategy for whole-body motion classification using Long Short-Term Memory Recurrent Neural Networks (LSTM), by only using sparse data from two separated Inertial Measurements Units (IMU) sensors. Our model, *Peek*, shows efficiency by achieving an overall accuracy of 96% during tests.

Keywords: Motion Classifier, IMU Device, Deep Learning, Recurrent Neural Networks, Sparse Data, LSTM, Machine Learning

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

CNTK	:	Cognitive Toolkit;
GB	:	Gigabyte;
GPU	:	Graphical Processing Unit;
HDM	:	Head Mounted Display;
IMU	:	Inertial Measurement Unit;
LSTM	:	Long Short Term Memory;
MW	:	Motion Window;
RNN	:	Rede Neural Recorrente (Recurrent Neural Network);

Sumário

Capítulo 1

INTRODUÇÃO

Os avanços e a popularização das tecnologias de realidade virtual dá origem a diversas possibilidades de gêneros de jogos eletrônicos. Dentre estes avanços podemos destacar a evolução da pervasividade [3] em jogos onde a sensação de imersão e o sentimento de presença são importantes, exigindo que o jogador interaja em diferentes formas como por exemplo, utilizando gestos e movimentos do corpo [4, 5, 6]. Experiências proporcionadas por sistemas como por exemplo, os HDMs (*head mounted display*, dispositivos para serem vestidos na cabeça de um usuário, composto por um *display* óptico em frente de cada olho) vêm recebendo grande destaque das indústrias de jogos e entretenimento. Contudo a maioria das aplicações ainda utiliza de controles tradicionais como, *joysticks*, teclados e outros periféricos, causando uma quebra na sensação de imersão que tais aplicações podem oferecer.

Algumas interfaces como os controles com sensores de movimento Playstation Move ou controles por movimento da HTC ainda não conseguem oferecer sensações reais indicando o quanto movimentos e gestos são importantes para manter as sensações de imersão e presença [7, 4]. Outro ponto relevante é o fato de que muitos sistemas de classificação de movimentos do corpo, exigem diversos acessórios o que torna a interface financeiramente custosa para o usuário, tal como apresentado no trabalho de Shiratori [8], que necessita que o usuário acople diversas câmeras pelo corpo, ou como o apresentado por Yang [9] que exigem que cinco acelerômetros sejam acoplados no corpo do usuário para que este consiga classificar ações humanas.

Dado a crescente necessidade em utilizar movimentos naturais do usuário para interagir em jogos pervasivos [3] (jogos onde o ambiente do mundo real faz parte do jogo) e da necessidade de diversos acessórios para que se possa inferir sobre o corpo do usuário nota-se a falta de um sistema que tente contornar ambos os problemas.

Neste contexto, foi proposta uma topologia de rede neural chamado *Peek*. *Peek* é capaz de realizar a classificação de movimentos humanos baseado em redes neurais recorrentes com LSTM [10], para classificar diferentes tipos de movimentos contínuos capturados por apenas dois sensores IMU (*Inertial Measurement Unit*). Tais sensores são capazes de medir a inclinação, rotação e velocidade angular dos braços. Dispositivos IMU são compostos por um acelerômetro e um giroscópio (podendo ainda incluir um magnetômetro no caso de dispositivos mais sensíveis).

Sensores IMU podem ser encontrados em dispositivos populares como celulares, pulseiras (como o dispositivo Myo [11]), *Smart Watches* e outros acessórios. Tais acessórios além de permitir a captura de movimento, são intuitivamente anexados ao corpo do usuário, o que auxilia em uma experiência mais natural [12, 13, 14, 15].

A principal contribuição deste trabalho é uma topologia de rede capaz de classificar o movimento baseando-se em apenas dados referentes aos braços exigindo apenas o uso de dois sensores IMU (enquanto outros sistemas exigem muitos dispositivos, ou processamento). Uma segunda contribuição é a criação de uma base de dados composta por sequências de movimentos criada inicialmente para validar a rede *Peek*, disponibilizada publicamente para outros fins.

1.1 Objetivos

O objetivo deste trabalho é desenvolver um sistema de classificação de movimentos robusto, baseado em dados esparsos de movimento dos braços do usuário quanto ao tipo de ação corporal sendo executado. Assim, pretende-se que o *Peek*:

- Seja capaz de analisar vários instantes de tempo de uma sequência de movimento e classificá-los como um tipo de ação ou de atividade corporal.
- Utilizar poucos dados corporais do usuário (como dados dos braços do usuário).
- Requerer que os instantes de tempo devem conter apenas informações que podem ser extraídas de qualquer sensor IMU.
- Seja capaz de trabalhar com diferentes tipos de ações, dependendo do seu escopo de uso. Para isso basta apenas uma base dados que especifique o tipo de movimento necessário.

1.2 Aplicabilidade

Peek pode ter diversos usos. Nesta seção iremos descrever dois possíveis cenários de aplicação o qual esta rede pode ser utilizada.

O primeiro deles se trata de utilizá-lo em um jogo que exija que o jogador se movimente com o corpo em um ambiente amplo. O desenvolvedor poderá incluir na sua aplicação uma forma de analisar, através de sensores nos braços do jogador, qual o atual estado do jogador (se ele está parado, andando, agachando, pulando, entre outros). Isso permitiria uma interface corporal com o jogo, sem a necessidade de utilizar acessórios em demasia.

O segundo cenário está relacionado em um jogo que analisa o comportamento corporal de um usuário ao longo do dia ou de um período de tempo. Enquanto o usuário procede com as atividades diárias, dois sensores IMU registram os estado dos seus braços em um celular com geo-localização através conexão sem fio com os aparelhos. Ao fim do dia ou do período os dados capturados seriam enviados para um servidor que, com uma rede *Peek* treinada, poderia exibir o trajeto feito pelo usuário ao longo do registro em um mapa e qual o estado corporal que este se encontrava no momento do percurso.

1.3 Metodologia

Para viabilizar este trabalho, diversos passos foram seguidos:

Inicialmente, foi feita uma revisão bibliográfica sobre dados esparsos de movimento, captura de movimento, e sistemas de classificação. Trabalhos sobre classificação de movimento foram revisados em seguida para analisar as técnicas mais recorrentes no estado da arte. Em seguida foi feita uma análise sobre trabalhos de classificação de sequência direcionando a pesquisa para a área de aprendizado profundo, redes neurais recorrentes e redes de convolução [10].

Foram coletados dados de movimento de voluntários em uma base de dados para que alguns modelos de classificação fossem testados. Inicialmente foram selecionados modelos de classificação com redes de convolução para serem testados com a base.

Os primeiros resultados exibiram uma precisão de classificação abaixo de 50%. Uma nova base de dados de movimento com mais amostras e mais informação foi criada com mais cuidado para tentar melhorar o resultado. A nova base causou uma melhoria nos resultados da precisão de acertos das técnicas usadas. Com o objetivo de tentar melhorar

ainda mais o método, buscaram-se trabalhos que utilizavam redes neurais recorrentes e *Long-Short-Term-Memory* (RNN-LSTM) para classificar sequências.

Novos experimentos com modelos utilizando RNN-LSTM foram realizados. Com os resultados dos primeiros experimentos já sendo melhores que os anteriores, foi investido tempo em experimentos com RNN-LSTM realizando melhorias na topologia da rede quanto na formatação da última base de dados gravada.

Experimentos com sequências de diferentes tamanhos de entrada da rede foram realizados para testar o comportamento da rede e encontrar o melhor formato da entrada. Experimentos foram feitos para validar a rede e testar suas capacidades. Os experimentos foram comparados a técnicas utilizadas anteriormente.

1.4 Organização do trabalho

Este trabalho está organizado da seguinte forma:

Capítulo 2 apresenta a fundamentação teórica que sustenta este trabalho. Serão explicados conceitos de Redes Neurais Recorrentes, LSTMs e Sensores IMU.

Capítulo 3 faz um apanhado de trabalhos similares ou com objetivos parecidos com este. Nele são descritos trabalhos relacionados a dados esparsos de movimento, classificação de sequências com redes neurais recorrentes e classificação de movimento.

Capítulo 4 descreve a estrutura de dados utilizada (Motion Window) assim como a topologia e configuração da rede neural recorrente com LSTM, *Peek*.

Capítulo 5 apresenta a base de dados construída para a experimentação da rede, explicando sua estrutura, características e motivação.

Capítulo 6 descreve os experimentos realizados e seus resultados, análises e comparações.

Capítulo 7 conclui o trabalho resumindo o que foi feito, as contribuições conquistadas, e possíveis trabalhos e aplicações futuras.

Capítulo 2

REFERENCIAL TEÓRICO

Este capítulo irá detalhar alguns conceitos importantes sobre redes neurais recorrentes e captura de dados por sensores de movimento. A Seção 2.1 irá descrever os conceitos principais sobre Redes Neurais Recorrentes. Em seguida a seção 2.2 irá brevemente descrever o conceito e propriedades de redes e células de *Long Short Term Memory* (LSTM). A seguinte Seção 2.3 irá descrever sucintamente o plano de fundo de captura de dados de movimento e irá falar brevemente sobre sensores IMU (*Inertial Measurement Unit*).

2.1 Redes Neurais Recorrentes

Redes Neurais Recorrentes (ou RNNs) [10, 16, 17] constituem um grupo de redes neurais especializadas em processar dados sequenciais. RNNs também são capazes de lidar com sequências de tamanhos variados. Tais redes são ideais para atividades como reconhecimento de escrita, fala, entre outros problemas sequenciais (problemas que envolvam sequências de amostras). RNNs podem ser capazes de traduzir uma sequência em uma nova (tradução de texto, interpretação de vídeo), classificar uma sequência (classificar uma cena de um filme de acordo com o gênero), ou prever amostras futuras de uma sequência (completar um texto automaticamente).

A Figura 2.1 mostra uma rede sofrendo o processo de "Unrolling" ou "Unfolding", em outras palavras, um nó RNN sendo replicado no tempo de uma sequência completa. Ou seja, o lado esquerdo da figura representa uma visão geral da rede demonstrando uma repetição da mesma tarefa para cada elemento de uma sequência n mesmo nó. A parte direita da Figura 2.1 mostra uma visão mais específica do desdobramento de um nó no tempo para para operar sobre algum tipo de sequência. Em um instante t temos x_t representando uma amostra da sequência nesse mesmo tempo, o_t representa a saída

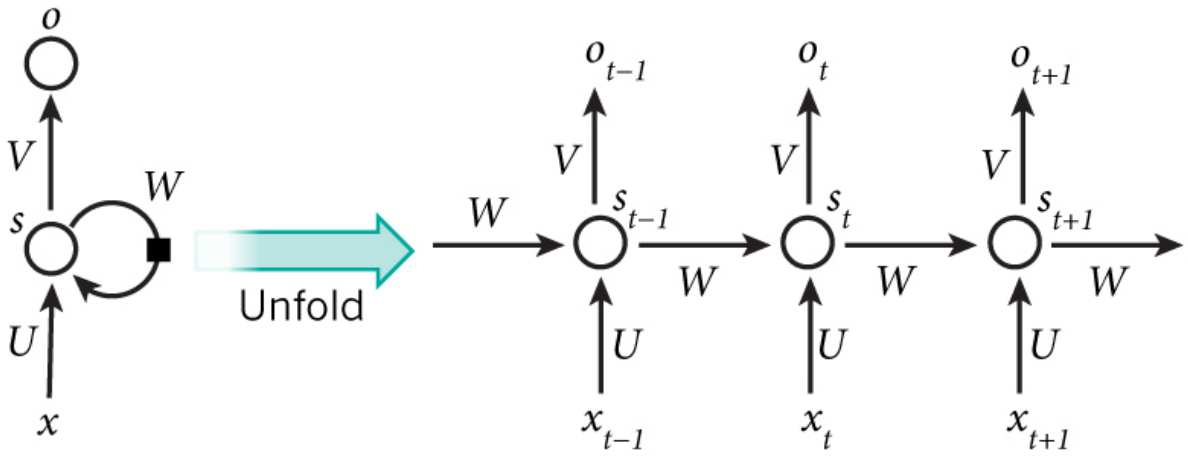


Figura 2.1: Representação de um nó de uma Rede Neural Recorrente sendo "desenrolado" (unfolding) para uma sequência completa. Ou seja o mesmo nó se replica várias vezes no tempo. x_t, representa a amostra da sequência de entrada da rede no tempo t, o_t representa a saída de cada passo no instante t, s_t representa o estado oculto no momento t. U, V, W são parâmetros da rede.

Fonte: LeCun et al.[17]

de um passo de um nó (ou estado) s_t. Os estados s_t representam o estado oculto (ou o resultado do processamento da informação que o nó já analisou) da rede no instante t. Os estados e saídas são calculados como mostram as equações **??**.

$$s_t = f(Ux_t + Ws_{t-1}) \tag{2.1}$$

$$o_t = g(Vs_t) \tag{2.2}$$

Onde f pode ser uma função não linear como por exemplo tahn (tangente hiperbólica). A função g representa uma função que irá gerar um resultado de saída (por exemplo $o_t = softmax(Vs_t)$, uma função que irá ler os valores do estado s_t no tempo t e normalizá-los em um valores cuja soma dos mesmos será igual a 1). Cada estado s_t após ser atualizado é enviado para o passo seguinte $t + 1$.

Em outras palavras a rede computa um novo estado s_t para cada novo instante x_t da sequência x levando em conta o estado anterior s_{t-1} retornando um valor o_t para cada passo t. Todos esses cálculos são regulados pelos mesmos parâmetros (U,V,W) em todos os passos. Tais parâmetros são calculados durante o treinamento da rede para que esta seja capaz de classificar com sucesso o maior número de exemplos da base de testes, utilizando algoritmos de otimização como por exemplo: *Stochatic Gradient Descent* [18].

Existem diversos tipos de RNNs e cada uma delas é direcionada para um tipo diferente de problema, podendo haver mudanças na configuração (arquitetura).

2.2 Rede Neural Recorrente com Long Short Term Memory

Embora eficientes com sequências, RNNs enfrentam uma dificuldade quando se trata de aprender dependências após vários estágios de processamento. Este problema é conhecido como *Exploding Gradient* ou *Vanishing Gradient*[10], ele está relcionado com o fato de que após diversas interações provocando atualizações no estado oculto da RNN, muita informação importante do começo de uma sequência pode se perder no final ou em algum momento em que ela será necessária. Diversas variações das RNNs foram criadas com o objetivo de amenizar este problema, sendo uma delas as células de *Long Short Term Memory* (LSTM) [19, 10, 20]. Tais células possuem "memória", ou seja, conseguem guardar informações por longos períodos.

Uma LSTM RNN possui as mesmas propriedades uma RNN padrão, contudo é capaz de armazenar informações por longos períodos de tempo ao processar uma sequência. Os nós de memória de uma LSTM RNN são chamados de "células". Células possuem propriedades mais complexas que suas equivalentes nas RNNs tradicionais. Estas são capazes de carregar informações até o final do processamento de uma sequência bem como selecionar informações que devem ser "esquecidas" a partir de um certo ponto. As Figuras 2.3 e 2.2 representam o funcionamento de uma célula LSTM.

Para entender melhor como funciona uma rede LSTM iremos analisar a célula detalhada na Figura 2.2. Existem cinco pontos de conexão com o exterior da célula, representados por: $C_{t-1}, h_{t-1}, x_t, C_t$ e h_t. Considere que a célula em questão representa um momento t na sequência a ser processada pela rede. C_t representa o estado da célula no instante t e este estado representa as informações que chegaram até este passo em instantes de tempo passados, ao contrário de apenas a última atualização do estado. Conseguir guardar essa informação é o grande diferencial das Redes LSTMs, pois ela consegue guardar informações relevantes do passado de uma sequência que podem só fazer sentido no futuro da mesma. Essa informação pode ser alterada através das Portas (ou *Gates*) de esquecimento e de aprendizado. A seguir apresenta-se o passo-a-passo de uma Célula de LSTM em cada amostra de sequência processada:

1. A célula recebe a amostra x_t da sequência x no instante t, recebe o valor de saída do passo anterior (h_{t-1}) e o estado de célula C_{t-1}, ambos referentes ao tempo $t-1$.

2. O *gate* de esquecimento f_t analisa h_{t-1} e x_t com uma função Sigmoide σ e decide o quanto de informação de C_{t-1} deve ser "jogado fora" baseado no valor entre 0 e

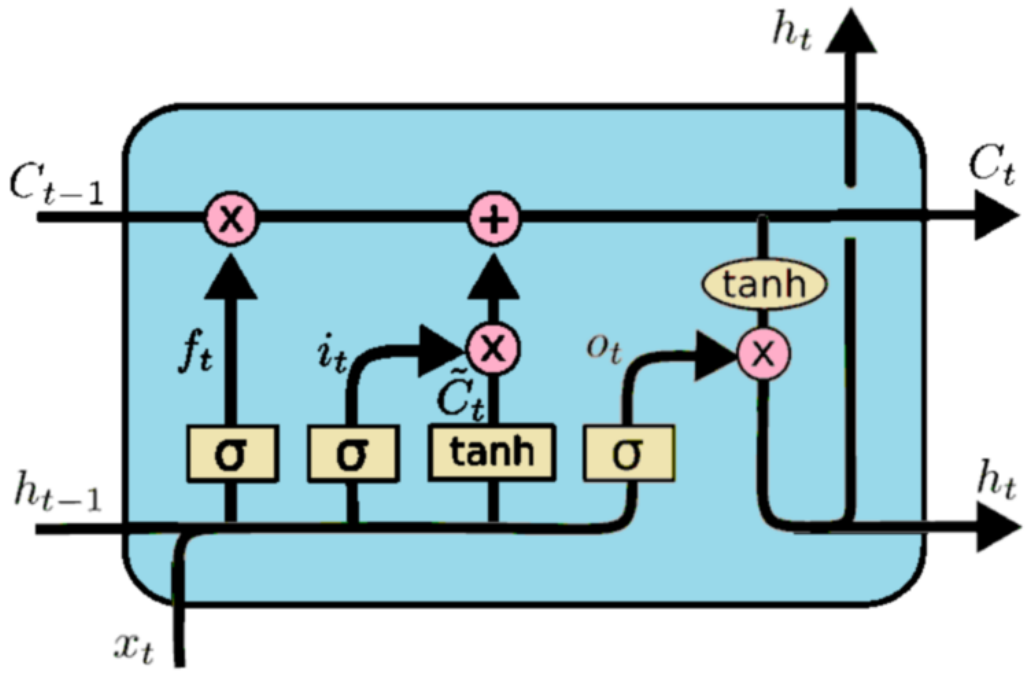


Figura 2.2: Representação de uma Célula LSTM. No instante t, C_t representa o estado de célula, h_t a saída da célula, x_t a amostra da sequência, f_t o *gate* de esquecimento, i_t o *gate* de entrada, o_t o *gate* de saída. Todos esses valores acabam sendo concatenados, multiplicados ou somados, conforme mostra o circuito desta ilustração.
Fontes: Jianshu; Olah [21, 20]

1 de f_t (0 representa esquecimento total e 1 permanência total). f_t é calculado da seguinte forma:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

3. O *gate* de entrada, i_t irá selecionar quais informações do estado de célula serão atualizadas com os valores dos novos candidatos $\widetilde{C}_t$. Cada um destes são calculados da seguinte forma:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\widetilde{C}_t = tahn(W_C \cdot [h_{t-1}, x_t] + b_C)$$

4. O novo estado de memória C_t é atualizado "esquecendo" certas informações do estado de célula anterior C_{t-1} e inserindo os dados dos candidatos $\widetilde{C}_t$ (caso os *gates* indiquem isso):

$$C_t = f_t * C_{t-1} + i_t * \widetilde{C}_t$$

5. Por último, o *gate de saída* é usado para calcular a saída h_t da célula. Esta saída

é baseada em uma filtragem da informação no novo estado de célula C_t e com o estado atual da sequência x_t. A saída é calculada da seguinte forma:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * tahn(C_t)$$

6. A saída h_t e o Estado de Célula C_t serão utilizados na próxima iteração $t + 1$.

Os valores W representam os pesos da rede, que como podemos ver pelas equações, são importantes para cada operação, sendo diretamente responsáveis no aprendizado, esquecimento e na saída de cada estado da célula. Estes pesos serão calculados durante o treinamento da rede para minimizar o erro.

Nestas equações encontramos também o valor de *biases* b cuja utilidade está ligada em possibilitar a movimentação das nossas funções (*gates) para a esquerda ou para a direita.* Tais valores também são calculados durante o treinamento.

Através dos passos descritos para cada célula, as redes LSTM conseguem processar sequências de tamanhos com as quais as RNNs comuns tem dificuldades em lidar. É importante lembrar que existem diversos tipos de variações e configurações destas redes, para diversos tipos de situações. Um exemplo é a célula apresentada por Gers e Schmidhuber [22], onde todos os gates levam C_{t-1} em consideração além de h_{t-1} e x_{t-1}.

Este trabalho possui uma abordagem para lidar com sequências de estados (*frames*) de movimento através do tempo e classificar a sequência quanto a ação gerada. Um passo t da célula LSTM irá receber o t-ésimo estado da sequência de movimento e a saída será uma classificação da ação representada pela sequência tal como mostra a Figura 2.4.

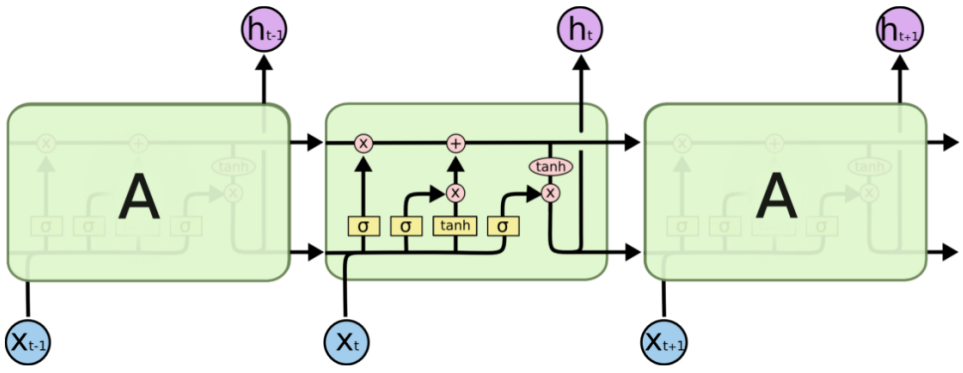


Figura 2.3: Representação de uma célula da Rede Neural Recorrente com LSTM sendo replicada no tempo

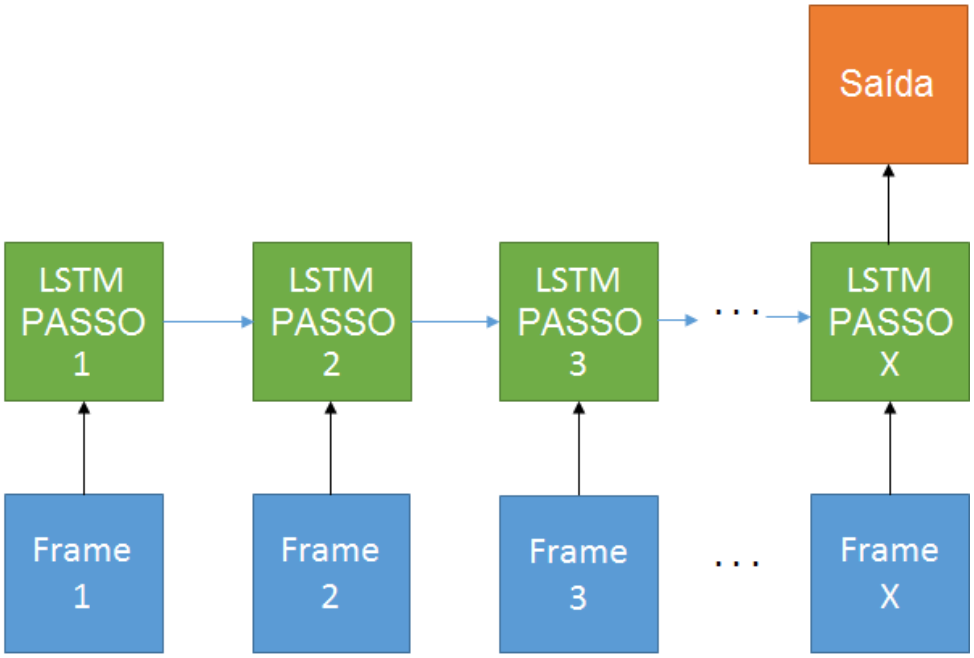


Figura 2.4: Representação de uma Célula LSTM da rede *Peek*onde cada amostra x_t de uma sequência é representado por um $Frame_t$. Cada passo da célula recebe este frame e no final um único valor de classificação será computado.

2.3 Sensores IMU

Em algumas aplicações é necessário saber a orientação ou movimentação da parte de alguma estrutura (de um ser vivo ou objeto), como por exemplo, em um jogo de *video-game* controlado pela mão do usuário. Com o objetivo de monitorar tais informações diversos dispositivos foram criados. Dentre estes, o mais utilizado são os *Inertial Measurement Units*, ou IMU [23]. Sensores IMU são uma composição de dois sensores: Acelerômetros, Giroscópios. IMUs podem ainda incluir um terceiro aparelho: Um magnetômetro. Cada uma dessas peças possui funções e propriedades específicas:

- Acelerômetro: Como o nome diz, é um dispositivo capaz de medir aceleração, tanto estática (como aceleração da gravidade) quanto dinâmica (movimento e vibração).
- Giroscópio: Sensor capaz de medir a inclinação e orientação. O giroscópio, em geral, utiliza a gravidade para medir sua orientação no espaço.
- Magnetômetro: Capaz de medir campos magnéticos, o magnetômetro consegue funcionar como uma bússola e medir a direção do dispositivo em relação ao norte-terrestre.

Tais acessórios podem ter seus dados combinados com técnicas de *Sensor Fusion* para reduzir problemas como ruído sensorial e melhorar a precisão do sensor como feito no trabalho de Zul Azfar [23].

Um IMU é capaz de retornar vetores que podem indicar indicando inclinação, angulação, orientação e velocidade de movimento. Tais medidas podem ser acessadas em até uma certa frequência (que depende da qualidade do acessório), permitindo anotar os estados do sensor em diferentes instantes de tempo.

No mercado existem diversos sensores IMU disponíveis, desde sensores brutos até aparelhos com tais sensores embarcados. Podemos listar chips IMU para dispositivos Arduinos [24] ou chips e sensores IMU da XSENS [25].

Além destes sensores, existem diversos dispositivos com tais sensores embarcados. Como os controles Wii-Mote Plus e Joy-Con da Nintendo [26], diversos smartphones como iPhone 5s da Apple ou a linha Nexus da Google [27]. Destes aparelhos existem ainda os que ganham destaque por serem anexados ao corpo como acessórios (*wearables*), entre eles smartwatches como os da Samsung [28] e também o dispositivo Myo [11] (Figura 2.5). Wearables podem ser incorporados em diversos tipos de aplicação, inclusive para a reconstrução de posturas e mapeamento de movimentos dos membros do corpo humano [29, 30].

Figura 2.5: Bracelete Myo[11]

Capítulo 3

TRABALHOS RELACIONADOS

O presente trabalho, *Peek*, é um sistema capaz de classificar ações humanas baseadas em sequências de dados esparsos de sensores IMU acoplados em indivíduos. Porém, considerando que o corpo humano possui diversas partes que se mexem em conjunto para realizar ações e movimento, é questionável se podemos realizar inferências sobre o corpo inteiro tendo somente informações esparsas de alguns membros do mesmo. J. Tautges[31] mostra que sim, sendo capaz de reconstruir a pose de um corpo inteiro com apenas dados esparsos de 4 sensores IMU amarrados em pernas e braços do usuário com a técnica de *Online Lazy Neighborhood Graph*. Sendo um resultado importante para provar que não é necessário ter todas as informações do corpo para fazer inferências sobre ele, assim como o presente trabalho, *Peek*, que busca classificar ações com dados esparsos dos braços de um indivíduo.

O trabalho de J. Tautges [31] (que é uma continuação do trabalho de B. Krüger[32]) consiste em criar um modelo de reconstrução de animação que utiliza um banco de dados com várias sequências de animações em conjunto com uma pessoa utilizando quatro acelerômetros (um em cada pulso e em cada tornozelo). O usuário executa movimentos e de acordo com os 4 instantes de movimentos anteriores, uma nova pose é calculada baseada no banco de dados e no movimento realizado. O trabalho utiliza primeiramente um servidor para efetuar os cálculos mais rapidamente. Os estados de cada acelerômetro são guardados em janelas fixas de tempo. O algoritmo guarda os últimos quatro estados antes de comparar com o novo. A comparação com o banco é feita usando uma *KD-tree*[33] e um algoritmo chamado de *Lazy Neighborhood Graph* [34, 31]. Em seguida a pose é calculada baseada nas poses do banco de dados mais próxima a esta. A base utilizada continha dados de motion-capture de corpo inteiro. O trabalho mostrou resultados acima do estado da arte, mas alerta que uma base de dados com muitos tipos de movimentos pode causar

dificuldades no reconhecimento. O erro geral apresentado pela técnica é aproximado a 2cm.

Figura 3.1: Trabalho de J. Tautges [31]

Além do trabalho de J. Tautges [31], devemos analisar outros trabalhos para validar o presente trabalho e situá-lo melhor. Esta seção irá contextualizar *Peek* frente a outros trabalhos encontrados na literatura. Inicialmente,a Seção 3.1 apresentará trabalhos que demonstrem como redes recorrentes de LSTM podem ser utilizadas para classificar dados sequenciais com eficiência. Em seguida, a Seção 3.2, irá mostrar trabalhos mais próximos da aplicação deste trabalho mostrando outras formas de classificação de movimento em diferentes escopos. A seção 3.3 apresenta uma sucinta comparação dos trabalhos citados com o modelo proposto, *Peek*.

3.1 Classificação de Sequências com Redes Neurais Recorrentes (RNN) e Long Short-Term Memory (LSTM)

Redes neurais e modelos estatísticos são amplamente utilizados para resolver diversos tipos de problemas, especialmente aqueles que requerem que uma máquina aprenda e generalize padrões de uma certa atividade ou dados [35]. Diversos tipos de redes tem sido criados para resolver problemas de variados tipos.

As redes neurais recorrentes (RNN) [10] são capazes de lidar com dados sequenciais como textos, animação, vídeo, entre outros. Diversos trabalhos utilizam as redes recorrentes utilizando um tipo de nó chamado LSTM (*Long Short-Term Memory*) [19]. Uma rede LSTM é capaz de lidar com informação temporal acumulada, sendo assim, uma ferramenta útil para redes recorrentes.

Esta seção irá brevemente listar trabalhos que exemplifiquem a utilização de Redes Recorrentes e LSTM.

H. Sak [36] mostra como uma RNN LSTM pode ser utilizada para modelar um sistema de reconhecimento de fala eficaz. O trabalho consiste em um algoritmo distribuído em

diversas máquinas para processar um RNN LSTM de duas camadas. Cada camada de LSTM é seguida de uma camada de projeção de recorrência linear. A rede utiliza amostras de 25ms como entrada representada por valores (*features*) 40-dimensional e foi treinada com uma base de dados de 1900 horas de fala previamente transcritas. Os resultados conseguiram passar das marcas dos sistemas de reconhecimento de fala da época.

Venugopalan [37] mostra uma rede de aprendizado profundo composta por uma rede de convolução combinada a uma rede recorrente de LSTMs. O Objetivo deste trabalho é ser capaz de analisar um vídeo e retornar uma frase em linguagem natural que descreva o que se passa na cena. A rede é alimentada pelos quadros de um vídeo. Cada quadro passa por camadas de convolução separadas para extrair um vetor de tamanho fixo de *features* que irão servir de entrada para uma rede neural recorrente de LSTMs. A saída desta última rede consiste em um vetor de palavras em inglês que formam uma frase que tentam expressar o que ocorre no vídeo. Os autores utilizaram a "Microsoft Research Video Description Corpus"[38]. Uma base que contém 1970 trechos de vídeos (de 10 a 25 segundos de duração) do site YouTube.com, onde cada trecho possui várias descrições geradas por indivíduos humanos. A técnica se mostrou superior aos melhores resultados encontrados na época.

Lipton [39] apresenta um modelo de rede com LSTM direcionado para diagnósticos médicos. O modelo proposto consiste em coletar dados de pacientes onde cada amostra possui 13 medidas frequentes porém coletadas irregularmente. A rede proposta também é capaz de dar até 128 tipos de diagnósticos não exclusivos (um conjunto de amostras pode ser traduzido em mais de um tipo de diagnóstico). A arquitetura propostas consiste em duas camadas de LSTMs com 128 células em cada. A rede foi treinada com uma base de dados compostas por séries temporais clínicas de vários pacientes anônimos do hospital "Children's Hospital LA"[40]. Os experimentos mostraram que o modelo poderia atingir uma precisão geral de cerca de 85% para as classificações.

3.2 Reconhecimento de Movimento baseado em Aprendizado de Máquina

Nesta seção serão apresentados trabalhos relacionados a classificação de movimento utilizando aprendizagem de máquina. Existe uma vasta literatura sobre movimento do corpo-humano com diferentes objetivos, tais como detecção, classificação e visualização. Este trabalho tem como foco a classificação de movimento, sendo assim esta seção irá descrever

apenas trabalhos relacionados diretamente a esse tema.

R. Zhang e C. Li [1] utilizam uma rede neural de convolução para classificar gestos. O escopo do trabalho é focado em classificar gestos finitos extraídos de um sensor de movimento dependendo do formato e direção do mesmo. A rede possui duas camadas de convolução e *max-pooling* intercaladas, seguida de uma camada oculta e uma camada de regressão logística. Como dado de entrada, os autores utilizam uma estrutura apelidada de *Data-Band* (DB). Um DB consiste em uma matriz com 6 linhas e 96 colunas, sendo que as 6 linhas representam a velocidade angular e a angulação do sensor ao longo de 96 instantes de tempo representado pelas colunas. O modelo proposto foi treinado com uma base de dados chamada de 6DMG [41] que contém sequências de 20 tipos de gestos que foram convertidas em DBs, atingindo uma precisão geral de 98%.

O trabalho de A. Yang [9] tem como foco conseguir classificar a atividade de um indivíduo utilizando acelerômetros no corpo. O experimento utilizou até 5 acessórios acoplados nos braços, pernas e cintura. Os dados dos acelerômetros são enviados para um servidor que registra os dados das sequências expostas. Os autores desenvolveram um algoritmo distribuído que funciona em um rede de computadores para classificar movimento chamado de DSC (*distributed sparsity classifier*). Utilizando sequências pré-gravadas como exemplos, o algoritmo tenta buscar qual a mais próxima e qual a mais capaz de rejeitar ações não presentes na base. O trabalho utilizou uma base própria chamada WARD composta por 20 atores e 13 categorias de ações. O trabalho atingiu uma precisão geral de cerca de 90%.

M. Baccouche [42] apresentou um método capaz de classificar atividades humanas em vídeos utilizando uma combinação de redes neurais de convolução e redes neurais recorrentes. Para conseguir fazer isso, os autores estenderam redes de convolução 2D para 3D. Uma rede com camadas de convolução seguida de uma camada oculta LSTM foi construída para isso. Os autores usaram um conjunto de dados chamado KTH Dataset [43] para testá-lo. Este conjunto de dados contém cerca de 599 vídeos (de 8 até 59 segundos cada) de pessoas executando sequências de seis tipos de ações (caminhadas, corrida, corrida, boxe, acenando a mão, cobrindo as mãos). Neste trabalho cada entrada da rede é composta por quadros do vídeo como matrizes 2D, esses quadros são agrupados para formar a entrada tridimensional para as convoluções 3D. Os autores obtiveram uma taxa de 94,39%, sendo a mais alta no momento da publicação.

Outros trabalhos de detecção de movimento também lidam com problemas similares, como os trabalhos de reconhecimento de atividade. Tais trabalhos não estão ligados só em

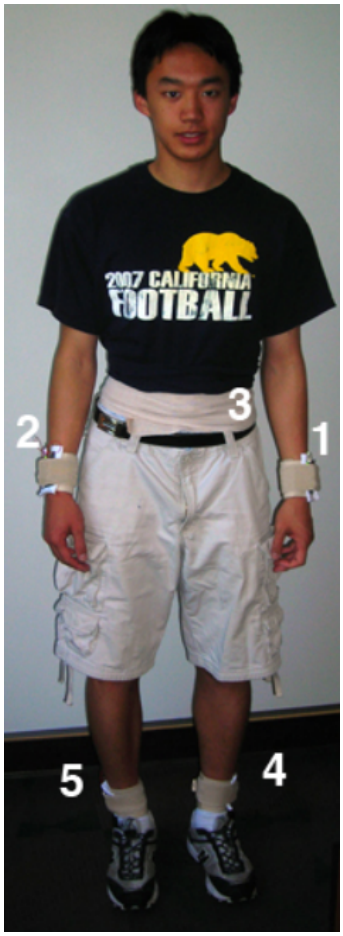


Figura 3.2: Usuário do trabalho de A. Yang equipado com 5 acelerômetros [9]

monitorar o estado do corpo do usuário, mas sim em detectar como este está interagindo, ou que tipo de atividades ele está executando. O trabalho de JY Yang [44] consiste em detectar a atividade realizada por um indivíduo observando sua mão dominante, tais como escovar os dentes, varrer, entre outros. O trabalho de FJ Ordonéz [45] utiliza sensores binários e um métodos baseado em camadas ocultas de Markov para monitorar a atividade de idosos.

3.3 Discussão

O trabalho de Tautges [31] mostra que é possível realizar inferências sobre o corpo com apenas dados esparsos do mesmo coletados com acelerômetros e giroscópios.

A Seção 3.1 discute como diversos tipos de classificação de sequências podem ser modelados através de Redes Neurais Recorrentes de LSTM. O fato de que movimentos corporais são constituídos de sequências de estados do corpo leva a crer que é possível lidar com inferências sobre o mesmo utilizando tais ferramentas.

A Seção 3.2 apresenta três trabalhos que sumarizam diversos trabalhos relacionados com classificação de movimento, tendo em vista que suas limitações podem ser melhoradas através do presente trabalho. O trabalho de Zhang [1] discute uma forma de classificar gestos com base em dados de movimento, porém o modelo proposto se foca em gestos finitos relacionados apenas com o movimento do próprio sensor além de ser direcionado para gestos finitos, enquanto o presente trabalho busca analisar informações de dois sensores para inferir sobre a atividade contínua do corpo inteiro, inviabilizando o uso da base para este fim. O trabalho de Yang [9] requer uma rede distribuída e cinco sensores de movimento, enquanto o presente trabalho exige apenas uma rede neural e dois sensores. Por fim o trabalho descrito por Baccouche [42], mostra uma abordagem próxima, porém é destinado para reconhecer atividades em vídeo, enquanto o trabalho que propomos tem apenas informações esparsas de sensores de movimento.

Trabalhos com reconhecimento de atividade demonstram objetivos similares a este trabalho, porém focam no que o indivíduo está fazendo em termos de atividade e não necessariamente de movimento corporal. Entretanto o fato de utilizarem poucos acelerômetros reforça a ideia de como poucos acelerômetros podem auxiliar em observações do corpo.

Assim sendo, pode-se concluir que a topologia “Peek” apresentado neste trabalho possui contribuições inéditas, tendo como destaque o fato de ser um modelo robusto que utiliza apenas dois sensores de movimento, e ser capaz de lidar com movimentos relacionados ao corpo inteiro.

Capítulo 4

PEEK: REDE NEURAL RECORRENTE LSTM PARA CLASSIFICAÇÃO DE MOVIMENTO CORPORAL COM DADOS ESPARSOS

Peek é uma topologia de Rede Neural Recorrente LSTM capaz de classificar movimentos analisando apenas os últimos x estados de dois sensores IMU. Com o crescimento de jogos de realidade virtual e pervasiva, há uma necessidade crescente de utilização de movimentos naturais para interagir com o ambiente para produzir uma sensação de presença no ambiente digital [7, 4]. Métodos que se limitam a reconstruir a pose do usuário não são suficientes no contextos onde o jogo precisa identificar o que o jogador está fazendo. Isso é uma tarefa difícil pois, em alguns casos, o jogo só possui dados esparsos, referentes a poucas partes do corpo do jogador e considerando que utilizar equipamentos sofisticados de captura de movimentos não é uma opção barata para a indústria de jogos. *Peek* busca contornar este problema sendo capaz de classificar uma sequência de amostras de movimento com um rótulo de um tipo de ação corporal.

Este capítulo irá descrever em detalhes o modelo *Peek*. A seção 4.1 irá listar alguns termos que usaremos para facilitar a leitura deste capítulo. Em seguida a Seção 4.2 irá descrever a formatação do dado com o qual a rede neural trabalhará. Por fim a Seção 4.3 irá descrever a topologia da rede.

4.1 Vocabulário Específico

Alguns termos utilizados neste trabalho podem ser encontrados na literatura com outros significados possíveis, outros são termos utilizados neste trabalho apenas para facilitar a leitura ou são pouco conhecidos. Iremos listar aqui os termos mais importantes.

Frame ou Amostra: Consiste em um instante de tempo e os dados coletados neste momento. Um frame neste trabalho irá representar o conjunto de dados capturados por sensores IMU em um determinado instante de tempo.

Sequência de Movimento: Uma sequência é um conjunto de frames ordenados pelo momento em que foram capturados. Uma sequência pode ter tamanho variado e cada sequência está ligada a uma ação corporal.

Ação corporal: Uma ação executada pelo corpo de um individuo. Não é simplesmente um gesto, mas algo mais geral como, por exemplo, andar e correr.

Motion Window ou Janela de Movimento (MW): Uma MW representa uma parte de uma Sequência de Movimento.

Dicionário: Neste trabalho, um dicionário representa os tipos de classes que possuem uma classificação mapeada. Neste caso, quais e quantos tipos de ações a rede é capaz de classificar.

4.2 Dados de Entrada e Saída da Rede

Antes de apresentar como a rede foi construída e como esta funciona primeiro deve-se especificar o tipo de dado em que iremos trabalhar. *Peek* irá ser receber e processar informações temporais de sensores IMU nos braços de um usuário representado por *Frames* agrupados em uma *Motion Windows*. Cada MW é dividida em *Frames* para representar o tempo, e cada *Frame* possui 20 valores diferentes. Os dados de cada frame são obtidos ao processar o estado dos acelerômetros em cada instante de tempo (Figura 4.1).

Uma MW de tamanho T representa os valores atuais do usuário e os $T - 1$ valores anteriores a este como mostrado na Figura 4.2. Os valores de um frame representam os seguintes dados coletados de sensores:

- Quatérnions representando a orientação de cada braço do usuário no instante de tempo do frame (4 variáveis para cada braço) [46].

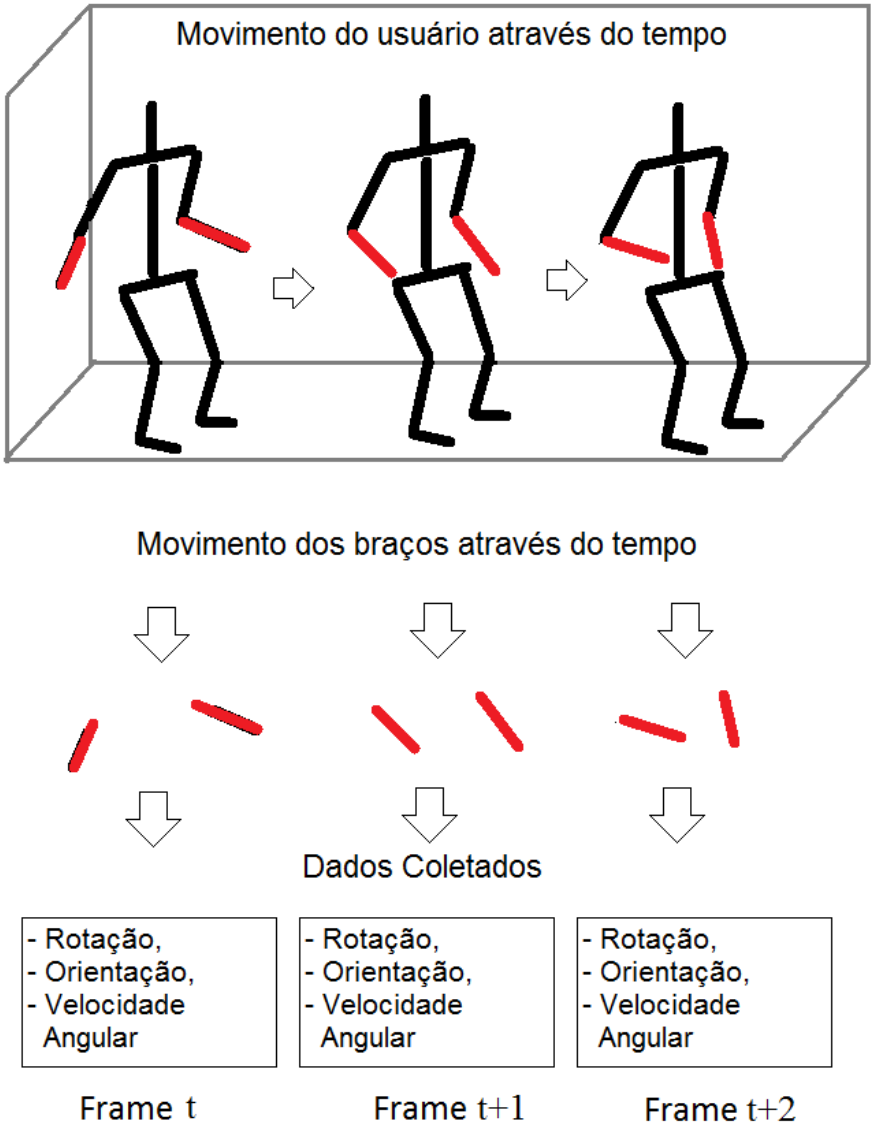


Figura 4.1: Processo de coleta de dados esparsos corporais dos braços. Membros vermelhos representam os braços sendo monitorados por sensores IMU.

- Dois conjuntos de Ângulos de Euler representando a rotação de cada braço do usuário no instante de tempo do frame (3 variáveis para cada braço) [46].
- Dois vetores tridimensionais representando a aceleração angular de cada braço no instante de tempo do frame. (3 variáveis para cada braço).

Como dado de saída, *Peek* irá retornar um vetor com probabilidades para cada tipo de ação corporal conhecida pela rede. Este vetor representa qual tipo de ação corporal a sequência da MW mais se aproxima baseado no maior valor. A figura 4.3 representa a estrutura.

Instante:	*Braço Esquerdo*	*Braço Direito*
$frame(0)$	$q_0(v,i,j,k), e_0(\alpha,\beta,\gamma)v_0(ax,ay,az)$	$q_0(v,i,j,k), e_0(\alpha,\beta,\gamma)v_0(ax,ay,az)$
$frame(1)$	$q_1(v,i,j,k), e_1(\alpha,\beta,\gamma)v_1(ax,ay,az)$	$q_1(v,i,j,k), e_1(\alpha,\beta,\gamma)v_1(ax,ay,az)$
.	.	.
.	.	.
.	.	.
.	.	.
.	.	.
.	.	.
.	.	.
$frame(T-1)$	$q_{T-1}(v,i,j,k), e_{T-1}(\alpha,\beta,\gamma)v_{T-1}(ax,ay,az)$	$q_{T-1}(v,i,j,k), e_{T-1}(\alpha,\beta,\gamma)v_{T-1}(ax,ay,az)$

Figura 4.2: Representação de uma Motion Window de tamanho T. Cada linha representa uma amostra (Frame(t)) no instante de tempo t. Para cada braço temos que: q_t representa os quatro valores do quatérnion no instante t, e_t representa os 3 valores dos Ângulos de Euler no instante t e v_t representa os vetores 3D da aceleração angular no instante t.

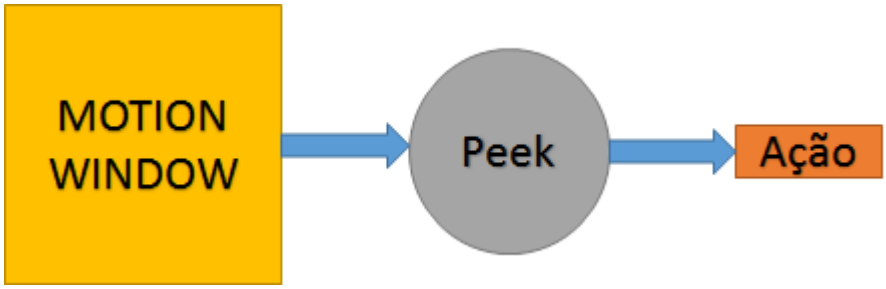


Figura 4.3: Representação em alto nível do funcionamento do *Peek*. Ao receber uma Janela de Movimento, *Peek* deverá classificar o tipo de ação corporal executado naquele período de tempo.

4.3 Topologia e Configuração da Rede

Peek possui uma camada de entrada para sequências de amostras de movimento representadas por 20 valores (*features*) para cada amostra. Em seguida uma camada oculta de 100 neurônios LSTM irá receber as sequências e processar suas amostras ordenadamente. Por fim uma camada de projeção que irá unir os resultados de cada seguida da camada de saída irão retornar um vetor de probabilidades representando cada classe no dicionário (Figura 4.4):

- Camada de Entrada (Input Layer): Esta camada irá receber uma Motion Window e em seguida irá quebrá-la em Frames separados. Cada Frame será enviado para as células LSTM de modo ordenado. Isso irá garantir que cada célula receba um frame para processar o estado de célula de cada instante de tempo.

- Camada Oculta (Hidden Layer): Esta camada possui 100 células LSTMs. Todas as

saídas de cada LSTM serão capturadas pela camada de projeção.

- Camada de projeção: A última camada irá receber todos os valores obtidos das células LSTMs e realizar uma projeção linear para unir os dados em um único vetor.

- Camada de Saída: A camada de saída recebe o vetor da camada de projeção e, através do algoritmo *Softmax*, irá gerar um vetor o qual representa o quanto cada tipo de ação mais se aproxima da sequência (a soma de todos os valores do vetor é igual a 1), ou seja criar um vetor de probabilidades de tamanho igual ao tamanho do dicionário de ações. A camada de saída tem tamanho igual ao tamanho do dicionário de ações.

É importante mencionar que cada célula irá receber os mesmos dados, mas irão possuir pesos e parâmetros diferentes. Em outras palavras, podemos dizer que cada célula irá encontrar valores diferentes para a mesma sequência. Tais valores serão unidos em um vetor só dentro da camada de projeção através de uma projeção linear.

Para que *Peek* possa classificar uma Motion Window de qualquer tamanho x com precisão, a rede precisa conhecer exemplos previamente classificados. Para isso é necessário uma base de dados com Motion Windows previamente rotuladas com uma ação. Tais MWs devem vir de ações gravadas por diversas pessoas para que *Peek* consiga generalizar as ações de diversos tipos de indivíduos. Durante o treinamento, a rede irá comparar os resultados com os valores reais (*ground truth*), e irá tentar ajustar os valores dos parâmetros da rede utilizando o algoritmo de *Stochatsic Gradient Descent* [18] para tentar minimizar o erro de classificação dos elementos presentes no treinamento. Este processo pode ocorrer em vários ciclos (épocas) até que *Peek* possa conseguir generalizar tais etapas.

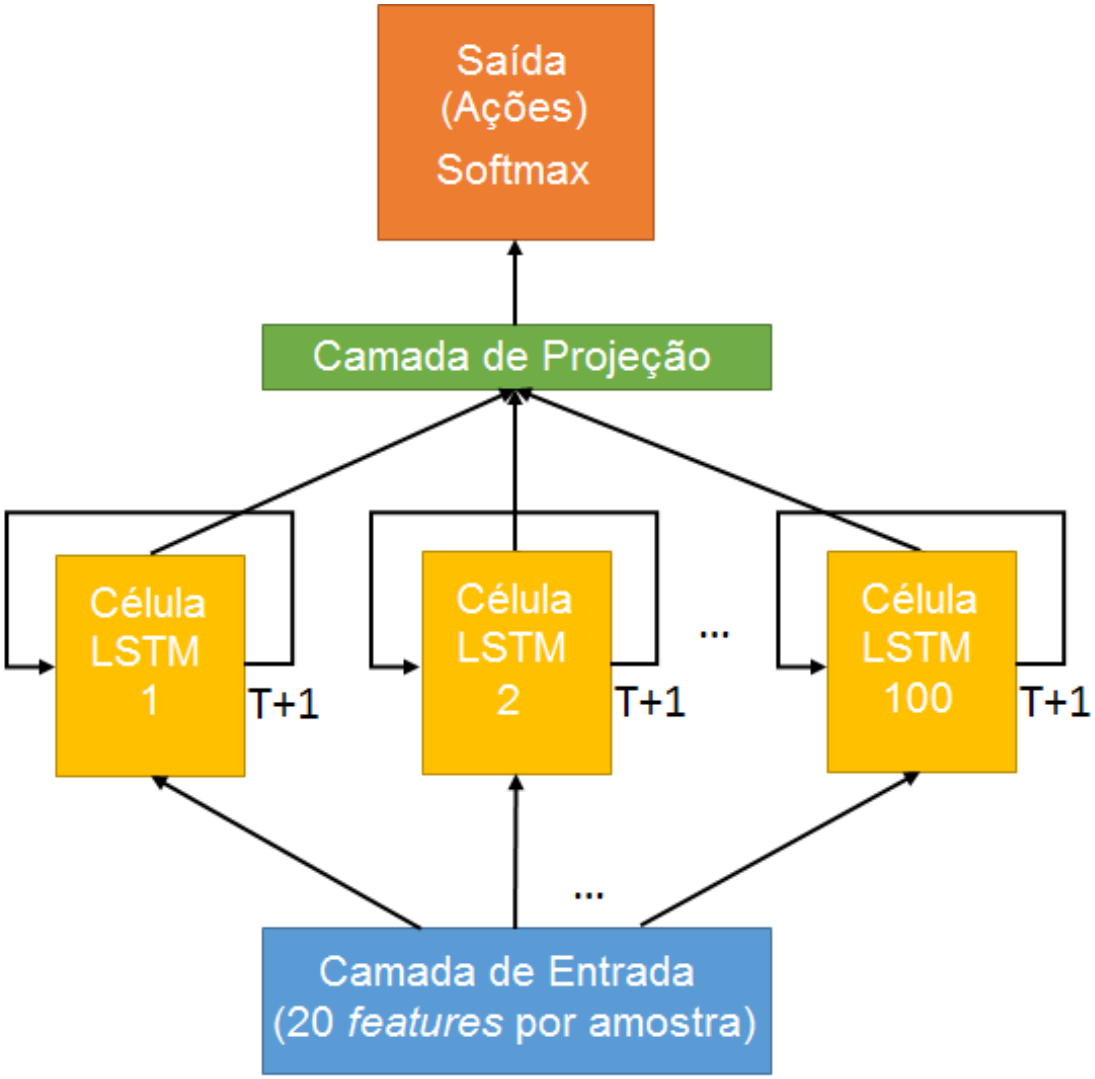


Figura 4.4: Representação da topologia de *Peek*. A camada de entrada irá enviar para todas as células LSTM um frame de maneira ordenada. A camada oculta de células LSTMs irá processar os dados recebidos e enviar dados de saída para a camada de projeção. Esta por sua vez irá unir todas as saídas em uma só através de uma projeção linear e em seguida, na camada de saída, calcular a probabilidade para cada classe utilizando a função Softmax.

Capítulo 5

DESCRIÇÃO DA BASE DE DADOS

Para testar a eficácia da rede proposta no Capítulo 4, foi construída uma base de dados contendo diversas sequências de movimento gravadas por diferentes pessoas. A base possui dados de 11 voluntários de diferentes perfis e estruturas corporais. Cada voluntário (ou ator) realizou 5 tipos de ações diferentes:

- Andar
- Correr
- Esperar (sem se mover, mas não completamente inerte)
- Agachar
- Brandir um objeto/Atacar

A escolha das ações usou como critério, ações que utilizam ou envolvem o corpo inteiro. Foram escolhidos 'Andar' e 'Correr' por serem atividades corporais frequentes e que, apesar de serem feitas com as pernas, podem ter algum reflexo nos braços. Escolheu-se a ação 'Agachar' por ser uma ação de corpo inteiro e presentes em atividades do dia-a-dia. Colocar a ação 'Esperar' no dicionário, permite que *Peek* seja capaz de detectar por si só se o usuário está de fato executando alguma ação. Por fim, decidimos acrescentar uma ação que dependeria mais dos braços ('Brandir/Atacar'), mas que poderia ser feita em movimento. Isso nos permitiria analisar se *Peek* conseguiria classificar estar ação mesmo se deslocando (sem errar classificando-a como 'Andar' ou 'Correr'). Tais ações são comuns também em cenários de jogos eletrônicos. Outras ações também poderiam se encaixar, porém essas não são problemáticas para gravações.

Este capítulo irá descrever o processo de gravação da base na Seção 5.1, e como as Motion Windows (MW) foram criadas para alimentar a rede na seção 5.2.

5.1 Construção da Base

Antes de realizar as gravações foi analisado a disponibilidade dos voluntários e especulada uma frequência de gravação de amostras. Estipulou-se pouco mais de 2 minutos e 20 segundos para cada ator gravar uma ação, pois nesse espaço de tempo era possível que o ator realizasse a mesma ação em diferentes ângulos além de ser possível enxergar várias repetições no movimento dos braços. Foi selecionada uma frequência de 50 amostras por segundo a serem coletadas durante a gravação, já que era a maior frequência que o software usado conseguia melhor manter sem quedas na taxa de gravação além de ser uma frequência acima do mínimo estipulado por alguns autores [47]. O software utilizado foi criado utilizando o motor de jogos, Unity [48], por este já possuir bibliotecas específicas para utilizar o os sensores IMU escolhidos.

Para realizar as gravações foram utilizados dois braceletes Myo [11], com uma taxa fixa de 50 amostras por segundo. Cada ator colocou um bracelete em cada antebraço e começou a realizar uma ação de cada vez até ter gravado 7000 amostras (equivalente a 2 minutos e 20 segundos) para cada ação (Figura 5.2). As amostras foram todas rotuladas de acordo com a ação e agrupadas por ator. Todos os atores gravaram aproximadamente o mesmo número de amostras para ter uma base de treinamento balanceada. Cada ator foi instruído para que fizesse as ações em diferentes direções.

Cada amostra possui dados de ambos os braços do ator. Tais dados correspondem aos dados necessários para treinar e usar *Peek* descritos na Seção 4.2.

A base totalizou cerca de 385.000 amostras, sendo 35.000 por ator.

5.2 Geração de Motion Windows

Para treinar e testar a rede, foram geradas diversas Motion Windows (MW). Uma MW de tamanho x consiste em uma sub-sequência contendo x frames adjacentes, em outras palavras um pedaço da sequência completa. Contudo, gerar MWs somente dividindo as sequências gravadas em partes iguais pode gerar uma base de treinamento pequena. Para resolver o problema, as MWs foram geradas com de um janelamento sobre a sequência (Figura 5.3). Ou seja ao invés de somente subdividir em pedaços de tamanho X, o

Figura 5.1: Voluntários executando ações para a base de dados

sequenciamento consiste em gerar uma nova MW para cada frame (instante de tempo) da base inserindo nela o frame da base e seus X-1 frames subsequentes. Essa medida tem como objetivo tentar fazer com que a rede aprenda padrões presentes em diferentes pedaços da sequência, que não seriam vistos próximos caso tivessem sido somente subdivididos.

Utilizando a base gravada, foram criadas MW com 100 frames para as etapas de treinamento e validação da rede. Também foram criadas MW dos mesmos dados com tamanhos diferentes para realização de testes com a rede. Os tamanhos utilizados foram: 100, 60, 50, 30, 25 e 10 frames.

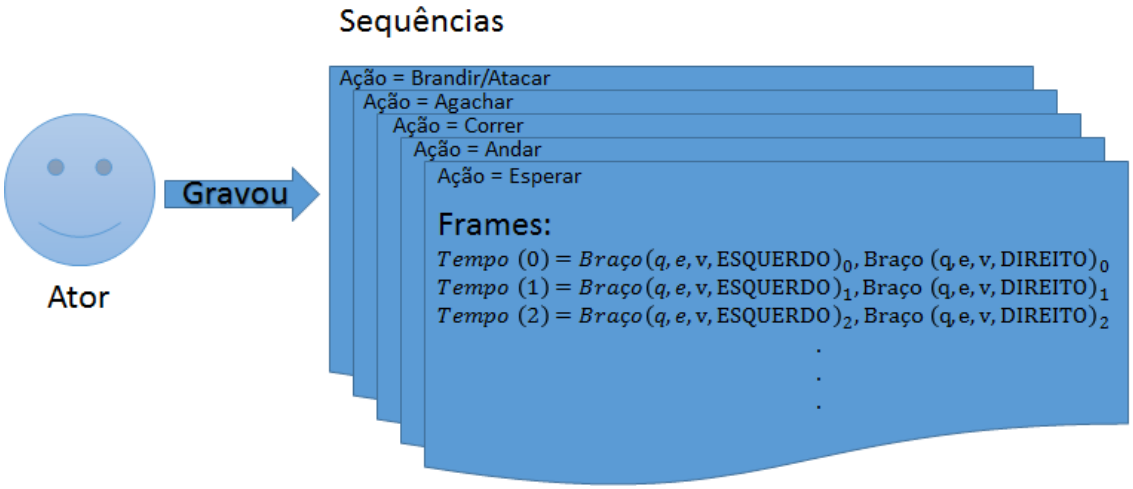


Figura 5.2: Representação de ator para Base. Cada Ator é representado por 5 sequências de movimento contendo por volta de 7000 frames cada. Cada sequência é gravada separadamente. Cada Frame da sequência representa dados do acelerômetro de cada braço em um instante de tempo. Os dados para cada braço são os três ângulos de Euler e quatro coordenadas de um quaternion ambos representando a orientação do braço.

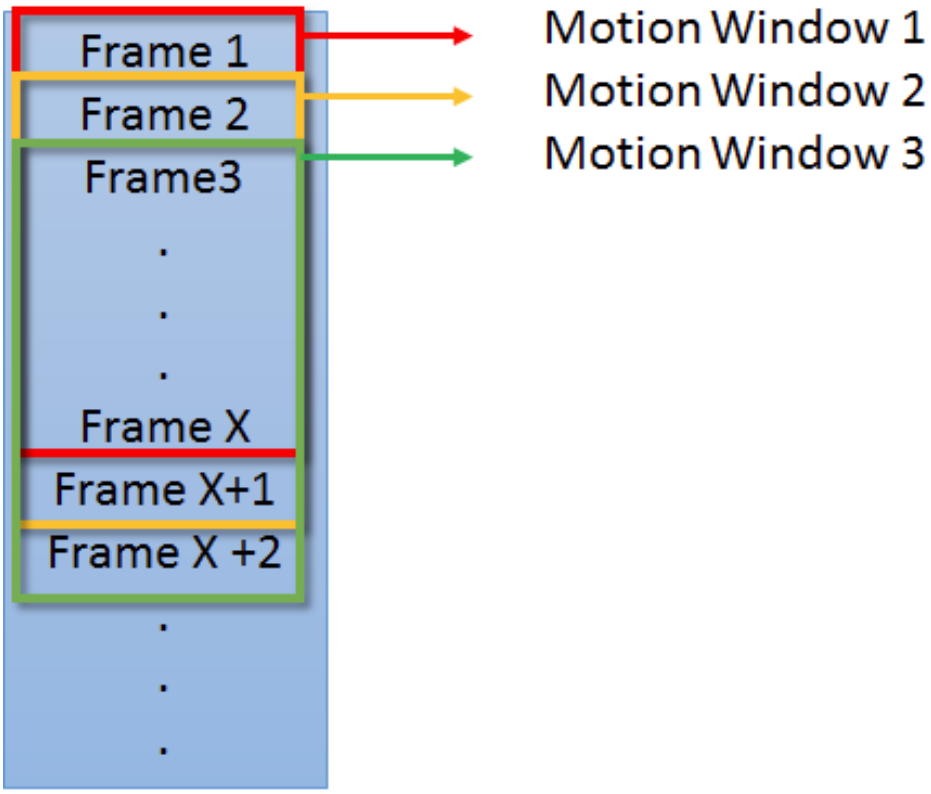


Figura 5.3: Janelamento para geração de Motion Windows. Consideramos que o bloco azul represente a sequência gravada pelos atores. A figura mostra como foi feito o janelamento (janela deslizante) para ser utilizado com o modelo *Peek*: É gerado uma nova MW para cada frame na base. Tal MW de tamanho X irá conter o frame de onde foi gerada mais os X-1 frames seguintes. É importante mencionar que cada MW carrega o rótulo da sequência de onde foi obtida.

Capítulo 6

RESULTADOS

Peek foi instaciado utilizando a biblioteca CNTK [49, 50] (Microsoft Cognitive Toolkit), uma ferramenta capaz de instanciar redes Neurais além de ser capaz de ser integrada em aplicações que utilizem C++, C# ou Python como linguagem de programação, o que permite uma vasta integração com outras aplicações. Utilizando a base de dados, Peek foi treinado utilizando Motion Windows extraídas das sequências dos atores. As sequências dos atores utilizados para o treinamento não foram utilizados na validação ou no teste. Os conjuntos de Treinamento e Validação foram compostos de MWs com 100 frames. Para os conjuntos de testes, foram extraídas Motion Windows de seis tamanhos diferentes (10, 25, 30, 50, 60 e 100 frames) para testar a rede em seis experimentos diferentes (utilizando as mesmas sequências porém com Motion Windows de diferentes tamanhos). É importante lembrar que *Peek* pode processar MW de quaisquer tamanho (embora o tamanho influencie na qualidade da classificação). Os experimentos iniciais consistiram em rotacionar um ator para teste e um para a validação e os restantes para o treinamento. Os experimentos tiveram uma variação baixa entre eles. Destes, foram selecionados os experimentos que tiveram resultados medianos em relação aos outros. Os resultados comparando estes experimentos para cada tamanho testado de Motion Windows se encontram na Tabela 6.1.

Para comparar *Peek* com outras técnicas, nossa base gerada foi aplicada a duas outras abordagens da literatura: O trabalho de X. Zhang [2] (utilizado para classificar texto baseado em sequências de caracteres com redes de convolução) e o trabalho de R. Zhang [1](classificação de gestos utilizando estruturas chamadas de *Data Bands* com redes de convolução). Os resultados comparando tais técnicas com *Peek* estão apresentadas na Tabela 6.2. Ambos os métodos foram adaptados para os dados de *Peek*, ou seja, as estruturas das redes de convolução foram modificadas ligeiramente para comportar os

mesmos dados que *Peek*. Já que o melhor caso visto foi utilizando 60 frames, nossas adaptações utilizaram janelas 64 frames (pois era o tamanho mais próximo de potências de 2 a 60, o que é recomendado em alguns casos de redes de convolução). As configurações finais estão presentes no Apêndice A. As técnicas foram comparadas com o melhor caso de teste dos experimentos listados.

Tabela 6.1: Tabela comparativa dos resultados dos experimentos utilizando Motion Windows de tamanho diferentes

Tamanho da Motion Window	Precisão
100 frames	96.40%
60 frames	**96.63%**
50 frames	96.48%
30 frames	95.09%
25 frames	93.85%
10 frames	81.69%

Tabela 6.2: Resultados comparativos de *Peek* com outras duas técnicas diferentes usando a mesma base de dados

Técnica	Precisão
Peek (60 frames)	**96.63%**
Classificador de Databands com Redes de Convolução [1] (Zhang, R. et al.)	75,11%
Classificador baseado em caracteres usando Redes de Convolução [2](Zhang, X. et al.)	87,5%

É possível notar que os resultados do teste com 60 frames foram os mais altos. Os resultados com 50 frames ficaram um pouco abaixo do alcançado com MWs de 60 frames, ainda conseguindo atingir a marca de 96% de acertos na classificação. Para melhor analisar o resultado obtido, foi construída uma matriz de confusão do melhor resultado (60 frames). É possível ver que todas as classes conseguem atingir uma precisão de acerto de 98%, exceto "Esperar" o qual foi confundido muitas vezes com "Agachar". Acredita-se que tal resultado é devido ao fato de que as sequências gravadas como "Agachar" incluíam alguns poucos momentos em que o usuário ainda estava de pé em posição semelhante a alguns momentos da ação "Esperar". Existe uma grande possibilidade de reduzir este problema de classificação eliminando Motion Windows relacionadas ao início da sequência. Mesmo com este problema, os resultados acima são aceitáveis.

As matrizes de confusão dos outros experimentos estão listadas da Tabela 6.4 até a Tabela 6.8. Para enxergar melhor a variação entre as taxas de acerto por classe (que podem ser vistas na diagonal principal das matrizes de confusão), a Figura 6.1 mostra um box-plot exibindo uma comparação da variação destas taxas entre os experimentos.

Tabela 6.3: Matriz de Confusão para o experimento com Motion Windows de 60 Frames Linhas representam a classificação real, e colunas a classificação do modelo *Peek*. (Resultados dados em porcentagem)

Rótulo/Resultado(%)	**Andar**	**Esperar**	**Correr**	**Brandir/ Atacar**	**Agachar**
Andar	**98.06**	0	0.57	0.21	1.15
Esperar	0.27	**90.08**	0	0	9.65
Correr	0	0	**98.59**	1.37	0.04
Brandir/Atacar	0.86	0.0	0.71	**98.31**	0.11
Agachar	1.69	0.06	0.0	0.14	**98.11**

Tabela 6.4: Matriz de Confusão para o experimento com Motion Windows de 10 Frames. As linhas representam a classificação real e colunas a classificação do modelo *Peek*. (Resultados dados em porcentagem)

Rótulo/Resultado(%)	**Andar**	**Esperar**	**Correr**	**Brandir/ Atacar**	**Agachar**
Andar	**65.53**	14.11	0.1	2.5	17.77
Esperar	0.17	**96.86**	0	0	2.97
Correr	3.24	0.29	**68.16**	24.48	3.83
Brandir/Atacar	3.04	0.65	1.2	**89.8**	5.32
Agachar	2.7	8.97	0.0	0.2	**88.12**

Foram testadas diferentes configurações para a rede. Usando menos células, a precisão da rede caiu, não atingindo a marca de 70% de acerto durante os testes (*under-fitting*). Experimentos com 5, 20, 50 e 100 células mostraram que a camada oculta com 100 unidades conseguia melhores resultados de classificação. Adicionando mais camadas ocultas, a precisão caiu. A rede atingiu apenas a marca de 80% de acertos, quando adicionada uma camada oculta a mais (*over-fitting*).

Para medir o quanto a rede é capaz de variar na classificação de casos não vistos, utilizamos uma validação cruzada com o método 5-Fold. O método consistiu em separar aleatoriamente os atores em cinco grupos (quatro grupos com dois atores e um com três). Foram realizados 5 experimentos onde as Motion Windows dos primeiros três grupos (folds) foram usados no processo de treinamento, o quarto no processo de validação e o quinto no de teste. Em seguida foram feitos 4 novos experimentos no mesmo formato, porém rotacionando a posição dos folds no início de cada experimento (o quinto fold se torna o primeiro, o primeiro se torna o segundo, seguindo nesta sequencia). O desvio médio padrão entre os resultados dos testes foi de 2.87 (do valor porcentual da precisão). Isso mostra como *peek* é capaz de generalizar para casos não vistos. A melhor precisão de acerto geral no 5-Fol foi de 89%.

Tabela 6.5: Matriz de Confusão para o experimento com Motion Windows de 25 Frames. As linhas representam a classificação real e colunas a classificação do modelo *Peek*. (Resultados dados em porcentagem)

Rótulo/Resultado(%)	**Andar**	**Esperar**	**Correr**	**Brandir/ Atacar**	**Agachar**
Andar	**88.75**	0.98	0.08	1.13	9.05
Esperar	0.6	**94.23**	0	0	5.72
Correr	0.25	0	**95.88**	3.7	0.17
Brandir/Atacar	2.57	0.08	1.31	**94.16**	1.89
Agachar	2.47	1.33	0.0	0.03	**96.17**

Tabela 6.6: Matriz de Confusão para o experimento com Motion Windows de 30 Frames. As linhas representam a classificação real e colunas a classificação do modelo *Peek*. (Resultados dados em porcentagem)

Rótulo/Resultado(%)	**Andar**	**Esperar**	**Correr**	**Brandir/ Atacar**	**Agachar**
Andar	**91.21**	0.37	0.17	0.88	7.37
Esperar	0.07	**93.33**	0	0	6.6
Correr	0.14	0	**98.38**	1.43	0.04
Brandir/Atacar	2.3	0.0	1.11	**98.31**	0.11
Agachar	2.3	0.65	0.0	0.03	**97.02**

O tempo de treinamento durou cerca de duas horas. Testar cerca de 35.000 Motion Windows diferentes com a rede treinada leva menos de 1,5 segundos nas mesmas configurações. Detalhes da configuração da rede *Peek* e da configuração da máquina utilizada para treinar e testar a rede estão presentes no Apêndice A.

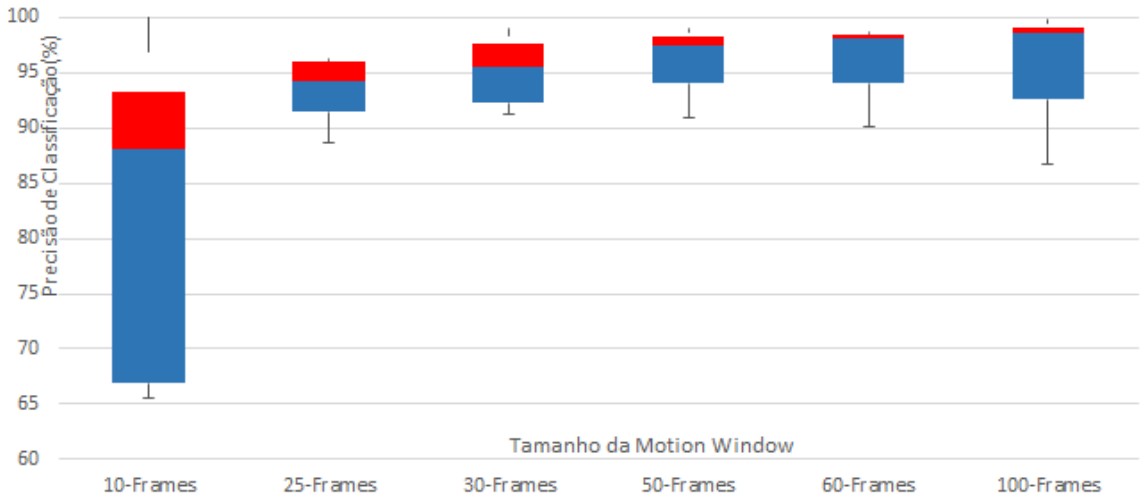
Tabela 6.7: Matriz de Confusão para o experimento com Motion Windows de 50 Frames. As linhas representam a classificação real e colunas a classificação do modelo *Peek*. (Resultados dados em porcentagem)

Rótulo/Resultado(%)	**Andar**	**Esperar**	**Correr**	**Brandir/ Atacar**	**Agachar**
Andar	**97.23**	0	0.37	0.24	2.16
Esperar	0.28	**90.63**	0	0	8.78
Correr	0	0	**98.67**	1.3	0.03
Brandir/Atacar	1.37	0.0	0.68	**97.5**	0.45
Agachar	1.77	0.1	0.0	0.06	**98.07**

Tabela 6.8: Matriz de Confusão para o experimento com Motion Windows de 100 Frames. As linhas representam a classificação real e colunas a classificação do modelo *Peek*. (Resultados dados em porcentagem)

Rótulo/Resultado(%)	**Andar**	**Esperar**	**Correr**	**Brandir/ Atacar**	**Agachar**
Andar	**98.81**	0	0.39	0.24	0.56
Esperar	0.27	**86.73**	0	0	13.0
Correr	0	0	**98.38**	1.6	0.01
Brandir/Atacar	0.09	0	0.38	**99.53**	0
Agachar	1.36	0	0	0.1	**98.54**

Figura 6.1: Box-Plot das precisões por classe para cada experimento



Capítulo 7

CONCLUSÃO

Neste trabalho apresentamos *Peek*, uma topologia rede neural recorrente com uma camada oculta de LSTM. *Peek* é capaz de generalizar a classificação de ação corporal e deve ser treinado utilizando um dicionário de tipos de ações e exemplos para cada um dos movimentos. Os exemplos devem ser organizados em Motion Windows, sequências de estados (frames) de sensores IMU presentes no braço do usuário. Cada frame deve conter dados referente a inclinação e velocidade angular dos braços (Ângulos de euler, quatérnions e vetores de velocidade angular).

O modelo *Peek* foi desenvolvido utilizando a ferramenta CNTK devido ao fato de ser integrável com uma muitas outras linguagens de programação, facilitando sua portabilidade para vários tipos aplicação como jogos. A base utilizada nos testes foi gerada utilizando o motor de jogos Unity, alimentada por dados de voluntários dispostos a gravarem informações para a mesma.

Os testes utilizando a base conseguiram atingir uma precisão de 96% de acerto, sendo que a menor precisão por classe chegou a 90%.

Este trabalho mostrou que é possível classificar ações corporais utilizando somente informações de dois sensores IMU, sem a necessidade de câmeras, pré-processamento pesado, ou quantidade excessiva de acessórios. Dessa forma, utilizar o modelo apresentado, é pouco custoso e não inconveniente em termos de custo financeiro ou de número de acessórios (já que não exige mais de dois acessórios acoplados ao usuário).

Algumas observações podem ser feitas, como a necessidade de uma base de treinamento limpa e variada. “Limpa” não significa que os atores não devam ser espontâneos. Contudo, para evitar problemas como mencionado no Capítulo 6, quando o estado “Esperar” foi confundido com “Agachar”, as gravações devem ser bem definidas evitando que

sejam gravadas ações com um rótulo quando em prática representam uma outra classe de movimento. Dessa forma, uma atenção a base de dados deve ser tomada.

Após treinar a rede com exemplos com Motion Windows de 100 frames foi possível classificar janelas de movimentos a partir de 25 frames com precisão acima de 90%. Levando em conta que o caso de teste utilizou dados gravados a 50 frames por segundo, podemos concluir que 0.5 segundo é tempo suficiente para representar uma atividade, embora um segundo de informações consiga uma precisão maior. Contudo, não podemos afirmar se modificando a frequência de gravação é possível conseguir resultados diferentes. Para isso novos dados devem ser obtidos em frequências diferentes e realizar novos testes. Esse questionamento levantado poderia servir como um novo tópico de pesquisa de maneira a aprimorar o modelo *Peek* para buscar definir o quanto o valor em tempo real e a taxa de amostragem influenciam em uma classificação de sequência.

Dentre aplicações possíveis e treinando com a base adequada, *Peek* pode ser utilizado em diversas situações já mencionadas. Sua principal utilização são em jogos pervasivos, onde as ações do jogador no "mundo real" influenciam seu desempenho. *Peek* pode oferecer a chance para desenvolvedores de interpretar o comportamento corporal do jogador desde que a rede seja treinada com o dicionário correspondente. Entre outros usos, pode-se citar como exemplo, análise de atividade física ao longo do dia, que com o dicionário e base de dados devidamente implementados, podem ser usados para analisar o tempo que o usuário andou, passou usando o computador, entre outros.

Este trabalho possui diversos trabalhos futuros possíveis. Dentre eles podemos listar:

- Encontrar se existem outros dados a serem coletados por sensores IMU capazes de auxiliar na classificação.
- Descobrir outras variações das células ou redes LSTM que poderiam aumentar a eficiência do modelo proposto.
- Encontrar outros dados corporais que possam ser utilizados na classificação desde que o usuário não tenha que utilizar mais acessórios.
- Buscar melhorias e/ou mudanças na topologia para a rede para que seja possível detectar a pose do corpo do usuário ou de parte dele.
- Desenvolver uma rede capaz de ler uma sequência de frames e ao invés de classificar com um rótulo, ser capaz de segmentar a sequência de acordo com a ação em cada segmento.

Este trabalho contribuiu para o estado-da-arte mostrando como RNN com LSTM podem ser usadas para classificar movimento utilizando dados provenientes de sensores IMU acoplados aos braços.

Referências

[1] ZHANG, R.; LI, C. Motion sequence recognition with multi-sensors using deep convolutional neural network. In: *Intelligent Data Analysis and Applications*. [S.l.]: Springer, 2015. p. 13–23.

[2] ZHANG, X.; ZHAO, J.; LECUN, Y. Character-level convolutional networks for text classification. In: *Advances in Neural Information Processing Systems*. [S.l.: s.n.], 2015. p. 649–657.

[3] MONTOLA, M.; STENROS, J.; WAERN, A. *Pervasive games: theory and design*. [S.l.]: Morgan Kaufmann Publishers Inc., 2009.

[4] RAUTARAY, S. S.; AGRAWAL, A. Vision based hand gesture recognition for human computer interaction: a survey. *Artificial Intelligence Review*, Springer, v. 43, n. 1, p. 1–54, 2015.

[5] SILVA, A. R.; VALENTE, L.; CLUA, E.; FEIJÓ, B. An indoor navigation system for live-action virtual reality games. In: IEEE. *Computer Games and Digital Entertainment (SBGames), 2015 14th Brazilian Symposium on*. [S.l.], 2015. p. 1–10.

[6] VALENTE, L.; CLUA, E.; SILVA, A. R.; FEIJÓ, B. Live-action virtual reality games. *arXiv preprint arXiv:1601.01645*, 2016.

[7] WACHS, J. P.; KÖLSCH, M.; STERN, H.; EDAN, Y. Vision-based hand-gesture applications. *Communications of the ACM*, ACM, v. 54, n. 2, p. 60–71, 2011.

[8] SHIRATORI, T.; PARK, H. S.; SIGAL, L.; SHEIKH, Y.; HODGINS, J. K. Motion capture from body-mounted cameras. *ACM Transactions on Graphics (TOG)*, ACM, v. 30, n. 4, p. 31, 2011.

[9] YANG, A. Y.; JAFARI, R.; SASTRY, S. S.; BAJCSY, R. Distributed recognition of human actions using wearable motion sensor networks. *Journal of Ambient Intelligence and Smart Environments*, IOS Press, v. 1, n. 2, p. 103–115, 2009.

[10] GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016. `http://www.deeplearningbook.org`.

[11] MYO Gesture Control Armband - Wearable Technology by Thalmic Labs. 2015. Disponível em: <https://www.myo.com>.

[12] PRATHIVADI, Y.; WU, J.; BENNETT, T. R.; JAFARI, R. Robust activity recognition using wearable imu sensors. In: IEEE. *IEEE SENSORS 2014 Proceedings*. [S.l.], 2014. p. 486–489.

[13] KUNI, R.; PRATHIVADI, Y.; WU, J.; BENNETT, T. R.; JAFARI, R. Exploration of interactions detectable by wearable imu sensors. In: IEEE. *2015 IEEE 12th International Conference on Wearable and Implantable Body Sensor Networks (BSN)*. [S.l.], 2015. p. 1–6.

[14] YUAN, Q.; CHEN, I.-M. Localization and velocity tracking of human via 3 imu sensors. *Sensors and Actuators A: Physical*, Elsevier, v. 212, p. 25–33, 2014.

[15] CHEN, X. Human motion analysis with wearable inertial sensors. 2013.

[16] SCHMIDHUBER, J. Deep learning in neural networks: An overview. *Neural networks*, Elsevier, v. 61, p. 85–117, 2015.

[17] LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *Nature*, Nature Research, v. 521, n. 7553, p. 436–444, 2015.

[18] BOTTOU, L. Large-scale machine learning with stochastic gradient descent. In: *Proceedings of COMPSTAT'2010*. [S.l.]: Springer, 2010. p. 177–186.

[19] GREFF, K.; SRIVASTAVA, R. K.; KOUTNÍK, J.; STEUNEBRINK, B. R.; SCHMIDHUBER, J. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, IEEE, 2016.

[20] OLAH, C. *Understanding LSTM Networks*. 2015. Disponível em: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>.

[21] RNN and LSTM. 2016. Disponível em: <http://www.jianshu.com/p/f3bde26febed>.

[22] GERS, F. A.; SCHRAUDOLPH, N. N.; SCHMIDHUBER, J. Learning precise timing with lstm recurrent networks. *Journal of machine learning research*, v. 3, n. Aug, p. 115–143, 2002.

[23] AZFAR, A. Z.; HAZRY, D. A simple approach on implementing imu sensor fusion in pid controller for stabilizing quadrotor flight control. In: IEEE. *Signal Processing and its Applications (CSPA), 2011 IEEE 7th International Colloquium on*. [S.l.], 2011. p. 28–32.

[24] ARDUINO. 2017. Disponível em: <https://www.arduino.cc/>.

[25] 3D Motion Tracking - XSENS. 2015. Disponível em: <https://www.xsens.com>.

[26] NINTENDO Official Website. 2015. Disponível em: <http://www.nintendo.com/>.

[27] MOURCOU, Q.; FLEURY, A.; FRANCO, C.; KLOPCIC, F.; VUILLERME, N. Performance evaluation of smartphone inertial sensors measurement for range of motion. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 15, n. 9, p. 23168–23187, 2015.

[28] SAMSUNG Official Website. 2017. Disponível em: <http://www.samsung.com>.

[29] SHEN, S. Arm posture tracking using a smartwatch. In: ACM. *Proceedings of on MobiSys 2016 PhD Forum*. [S.l.], 2016. p. 9–10.

[30] ZHU, X.; LI, K. F. Real-time motion capture: An overview. In: IEEE. *Complex, Intelligent, and Software Intensive Systems (CISIS), 2016 10th International Conference on*. [S.l.], 2016. p. 522–525.

[31] TAUTGES, J.; ZINKE, A.; KRÜGER, B.; BAUMANN, J.; WEBER, A.; HELTEN, T.; MÜLLER, M.; SEIDEL, H.-P.; EBERHARDT, B. Motion reconstruction using sparse accelerometer data. *ACM Transactions on Graphics (TOG)*, ACM, v. 30, n. 3, p. 18, 2011.

[32] KRÜGER, B.; TAUTGES, J.; WEBER, A.; ZINKE, A. Fast local and global similarity searches in large motion capture databases. Citeseer.

[33] OOI, B. C.; MCDONELL, K. J.; SACKS-DAVIS, R. Spatial kd-tree: An indexing mechanism for spatial databases. In: *IEEE COMPSAC*. [S.l.: s.n.], 1987. v. 87, p. 85.

[34] KRÜGER, B.; TAUTGES, J.; WEBER, A.; ZINKE, A. Fast local and global similarity searches in large motion capture databases. In: EUROGRAPHICS ASSOCIATION. *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. [S.l.], 2010. p. 1–10.

[35] SARLE, W. S. Neural networks and statistical models. Citeseer, 1994.

[36] SAK, H.; SENIOR, A. W.; BEAUFAYS, F. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In: *Interspeech*. [S.l.: s.n.], 2014. p. 338–342.

[37] VENUGOPALAN, S.; XU, H.; DONAHUE, J.; ROHRBACH, M.; MOONEY, R.; SAENKO, K. Translating videos to natural language using deep recurrent neural networks. *arXiv preprint arXiv:1412.4729*, 2014.

[38] CHEN, D. L.; DOLAN, W. B. Collecting highly parallel data for paraphrase evaluation. In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL-2011)*. Portland, OR: [s.n.], 2011.

[39] LIPTON, Z. C.; KALE, D. C.; ELKAN, C.; WETZELL, R. Learning to diagnose with lstm recurrent neural networks. *arXiv preprint arXiv:1511.03677*, 2015.

[40] MARLIN, B. M.; KALE, D. C.; KHEMANI, R. G.; WETZEL, R. C. Unsupervised pattern discovery in electronic health care data using probabilistic clustering models. In: ACM. *Proceedings of the 2nd ACM SIGHIT International Health Informatics Symposium*. [S.l.], 2012. p. 389–398.

[41] CHEN, M.; ALREGIB, G.; JUANG, B.-H. 6dmg: A new 6d motion gesture database. In: ACM. *Proceedings of the 3rd Multimedia Systems Conference*. [S.l.], 2012. p. 83–88.

[42] BACCOUCHE, M.; MAMALET, F.; WOLF, C.; GARCIA, C.; BASKURT, A. Sequential deep learning for human action recognition. In: SPRINGER. *International Workshop on Human Behavior Understanding*. [S.l.], 2011. p. 29–39.

[43] SCHULDT, C.; LAPTEV, I.; CAPUTO, B. Recognizing human actions: a local svm approach. In: IEEE. *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*. [S.l.], 2004. v. 3, p. 32–36.

[44] YANG, J.-Y.; WANG, J.-S.; CHEN, Y.-P. Using acceleration measurements for activity recognition: An effective learning algorithm for constructing neural classifiers. *Pattern recognition letters*, Elsevier, v. 29, n. 16, p. 2213–2220, 2008.

[45] ORDÓNEZ, F. J.; TOLEDO, P. de; SANCHIS, A. Activity recognition using hybrid generative/discriminative models on home environments using binary sensors. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 13, n. 5, p. 5460–5477, 2013.

[46] DIEBEL, J. Representing attitude: Euler angles, unit quaternions, and rotation vectors. *Matrix*, v. 58, n. 15-16, p. 1–35, 2006.

[47] MAGNENAT-THALMANN, N.; THALMANN, D. *Modelling and Motion Capture Techniques for Virtual Environments: International Workshop, CAPTECH'98, Geneva, Switzerland, November 26-27, 1998, Proceedings*. [S.l.]: Springer, 2003.

[48] UNITY - Game Engine. 2016. Disponível em: <https://unity3d.com/>.

[49] MCCAFFREY, J. Machine learning - exploring the microsoft cntk machine learning tool. In: . MSDN Magazine Blog, 2017. v. 32. Disponível em: <https://msdn.microsoft.com/en-us/magazine/mt791798.aspx>.

[50] GITHUB: Microsoft Cognitive Toolkit (CNTK). 2017. Disponível em: <https://github.com/Microsoft/CNTK/wiki>.

[51] NVIDIA. *NVIDIA DIGITS - Interactive Deep Learning GPU Training System*. 2017. Disponível em: <https://developer.nvidia.com/digits>.

[52] VISION, B. *Caffe: Deep learning framework*. 2017. Disponível em: <http://caffe.berkeleyvision.org/>.

[53] TORCH: A SCIENTIFIC COMPUTING FRAMEWORK FOR LUAJIT. 2017. Disponível em: <http://http://torch.ch/>.

APÊNDICE A – Configuração dos Experimentos

Este apêndice tem como objetivo oferecer as configurações dos experimentos descritos no Capítulo 6. Todos os testes e treinamentos foram realizados com o auxílio de uma máquina equipada com uma placa de vídeo ou GPU (*Graphical Processing Unit*) do tipo *GeForce GTX TITAN X (12gb)*. A Seção A.1 irá descrever detalhes da implementação de *Peek*. Enquanto a Seção A.2 irá descrever detalhes de implementação para cada um dos dois experimentos.

A.1 PEEK

Peek foi instanciado utilizando a biblioteca CNTK [49]. Em uma descrição simples, *Peek* foi instanciado com uma camada de entrada de tamanho 20 de comprimento variável. Seguido de uma camada de Recorrente de LSTM com 100 células seguido de uma cada "Dense Layer" (Camada de Projeção). Os pesos de ambas camadas foram inicializados com uma função gaussiana. Seguidamente instanciamos uma camada de saída com tamanho igual ao número de rótulos utilizando uma função softmax. Outras especificações estão na Tabela A.1. O código para melhor especificação da inicialização está descrito na Figura A.1 .

Tabela A.1: Outras especificações de *Peek*

Outras Especificações	
Tamanho Minibatch	512
Total de épocas	5
Momentum por MiniBatch	0.9
Taxa de aprendizado por MiniBatch	0.1

```
BrainScriptNetworkBuilder=[

    lstmDim = 100
    numLabels = 5
    vocabDim = $vocabDim$

    model = Sequential
    (
        RecurrentLSTMLayer {lstmDim, init="gaussian"} :
        BS.Sequences.Last :
        DenseLayer {numLabels, init="gaussian"}
    )

    t = DynamicAxis{}
    features = Input {$vocabDim$, dynamicAxis=t}
    labels = Input {numLabels}
    z = model (features)
    zp = ReconcileDynamicAxis(z, labels)
    ce = CrossEntropyWithSoftmax (labels, zp)
    err = ClassificationError (labels, zp)


    featureNodes = (features)
    labelNodes = (labels)
    criterionNodes = (ce)
    evaluationNodes = (err)
    outputNodes = (z)
]

SGD = [
    epochSize = 0
    minibatchSize = 512
    maxEpochs = 5
    momentumPerMB = 0.9
    learningRatesPerMB = 0.1
]
```

Figura A.1: Pedaço de código CNTK para definir *Peek*

A.2 Trabalhos Para Comparação

Ambos os experimentos foram realizados com o auxílio do Nvidia Digits [51]. Utilizando os mesmos atores utilizados nos experimentos descritos nas tabelas do Capítulo 6 para validação teste e treinamento, foram gerados imagens de tamanho 64 por 20 (representado os 20 valores de cada frame em 64 instantes de tempo). cada valor foi mapeado em um pixel com um valor normalizado entre 1 e 255. O intervalo de validação usado foi para cada um quarto de época. Foi utilizado uma taxa de aprendizado fixa de 0,0001. Ambas possuem uma camada de transformação para o tamanho do número de classes seguida de uma camada *softmax*.

A.2.1 Data-Band Convolutional Network

Este experimento atingiu convergência com 12 épocas de treinamento. As redes de convolução estão descritas na Tabela A.2. Para instanciar a rede foi utilizado a linguagem Caffé [52]. Tirando o tamanho da entrada, este experimento seguiu a risca as configurações do melhor experimento listado.

Tabela A.2: Camadas do Experimento com redes de Databands[1]

Camada	Tamanho
1 Convolução	Saída: 30 Kernel: Altura 20 x Largura 6 Stride: 1
2 Max-Pooling	Altura 1 x Largura 3
3 Convolução	Saída: 40 Kernel: Altura 1 x Largura 5 Stride: 1
4 Max-Pooling	Altura 1 x Largura 2
5 Produto Interno	Tamanho 500

A.2.2 Character Based Convolutional Network

Este experimento atingiu convergência com 22 épocas de treinamento.As redes de convolução estão descritas na Tabela A.3. ara instanciar a rede foi utilizado a linguagem Lua juntamente a biblioteca Torch [53]. Como o trabalho original utilizava sequencias de caracteres onde cada um era convertido em um *one-hot-vector*, a entrada seguiu um formato parecido, manteve a largura para refernciar o tempo na sequência e as colunas para representar o estado atual no lugar de cada caractere. Isso causou uma mudança drástica no tamanho da entrada o que necessitou reajustar o tamanho das convoluções e *max-poolings*.

Tabela A.3: Camadas do Experimento com redes baseadas em caracteres[2]

Camada	Tamanho
1 Convolução Temporal	Entrada: 20 Saída: 256 Kernel: 3 (tamanho)
2 Max-Pooling	Altura 3 Largura 3
3 Convolução Temporal	Entrada: 256 Saída: 256 Kernel: 1 (tamanho)
4 Max-Pooling	Altura 3 Largura 3
5 Convolução Temporal	Entrada: 256 Saída: 256 Kernel: 1 (tamanho)
6 Convolução Temporal	Entrada: 256 Saída: 256 Kernel: 1 (tamanho)
7 Convolução Temporal	Entrada: 256 Saída: 256,Kernel: 1 (tamanho)
8 Convolução Temporal	Entrada: 256 Saída: 256,Kernel: 1 (tamanho)
9 Max Pooling	Altura 3 Largura 3
10 Ajuste de Tamanho	Tamanho: 512
11 Transformação Linear	512 para 1024
12 Camada de Dropout	0.5
13 Transformação Linear	1024 para 2014
14 Camada de Dropout	0.5