Instituto Tecnológico de Aeronáutica - ITA
CT-213 - Inteligência Artificial aplicada à Robótica Móvel
Aluno: Ulisses Lopes da Silva

Relatório do Laboratório 7 - *Imitation Learning* com Keras

---

1 Breve Explicação em Alto Nível da Implementação

O laboratório 7 consistiu na implementação de uma rede neural (RN) utilizando o framework da biblioteca *Keras*, presente no conjunto de bibliotecas do *Tensorflow*, com o objetivo de realizar a técnica de *imitation learning*, a partir de um conjunto reduzido de amostras representativas do comportamento de caminhada de um robô humanoide. O arquivo com os dados consistia num conjunto posições de várias juntas do robô durante alguns intervalos de tempo. O modelo de RN segue uma arquitetura sequencial composta por três camadas conectadas. A entrada da rede é unidimensional, e consiste no instante de tempo do movimento do robô. A saída, por sua vez, é multidimensional, e é composta pela posição de cada junta do robô no instante de tempo.

A primeira camada oculta contém 75 neurônios e utiliza uma função de ativação linear seguida de uma ativação *LeakyReLU* com coeficiente $\alpha = 0{,}01$, visando preservar gradientes não nulos mesmo para valores negativos. A segunda camada oculta apresenta 50 neurônios com a mesma combinação de ativações e parâmetro. A camada de saída possui 20 neurônios, esta com ativação linear, de modo a gerar uma saída contínua compatível com o comportamento esperado do agente. AS camadas não utilizam regularização, e foi estabelecido um parâmetro $\lambda_{L2} = 0.00$ (sem penalização) para compor os argumentos, seguindo o exemplo da implementação-teste, presente no arquivo. O otimizador escolhido foi o *Adam*, com função de perda baseada no erro quadrático médio (*Mean Squared Error*), apropriada para tarefas de regressão.

O treinamento foi conduzido em 30 mil épocas, utilizando um *batch size* igual ao número total de amostras disponíveis (40 amostras). Após o treinamento, a rede foi usada para prever valores em um intervalo contínuo, permitindo a visualização da política aprendida pelo agente imitador, e os gráficos de comparação foram gerados porsteriormente, com base em cinco juntas. O treinamento, em média, durou cerca de 20 minutos, com as configurações da máquina conforme seguem abaixo. Cabe ressaltar que, para o treinamento, utilizou-se a GPU nativa da máquina, bem como o argumento `"verbose=0"`dentro de `model.fit()`, a fim de não printar as épocas e obter um resultado mais rápido.

Processador	AMD Ryzen 7 5800H with Radeon Graphics 3.20 GHz
RAM instalada	16,0 GB (utilizável: 15,4 GB)
Tipo de sistema	Sistema operacional de 64 bits, processador baseado em x64
Placa de vídeo	NVIDIA GeForce RTX 3060 Laptop GPU

Tabela 1: Especificações do dispositivo

2 Figuras Comprovando Funcionamento do Código

2.1 Função de Classificação *sum_gt_zeros*

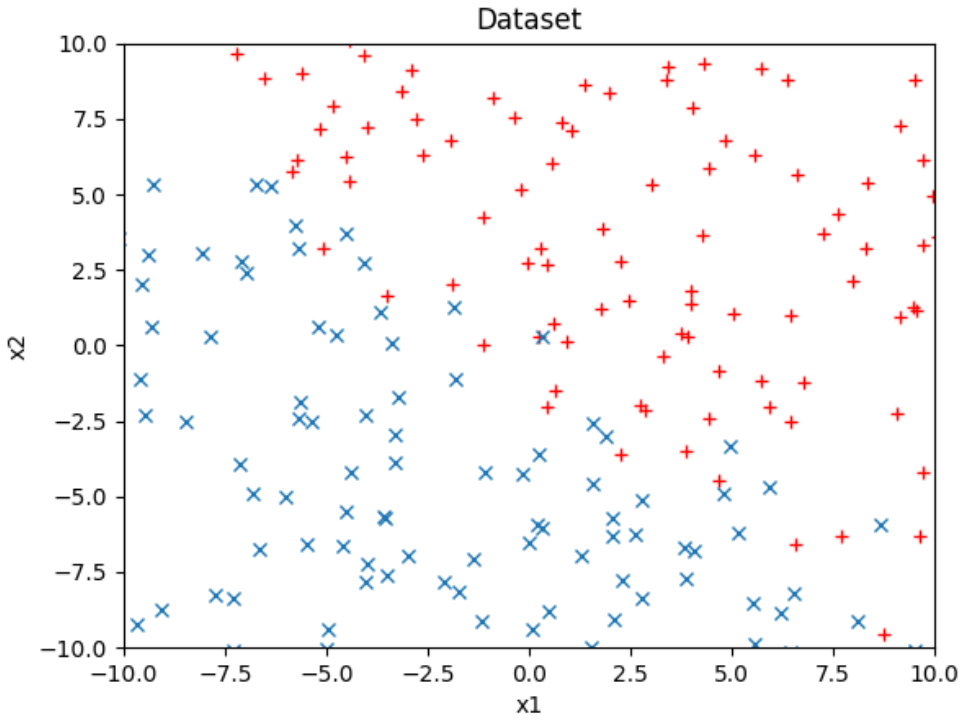

Fig. 1: *Dataset* do conjunto de dados para o treinamento da rede na Função de Classificação *sum_gt_zeros* sem regularização

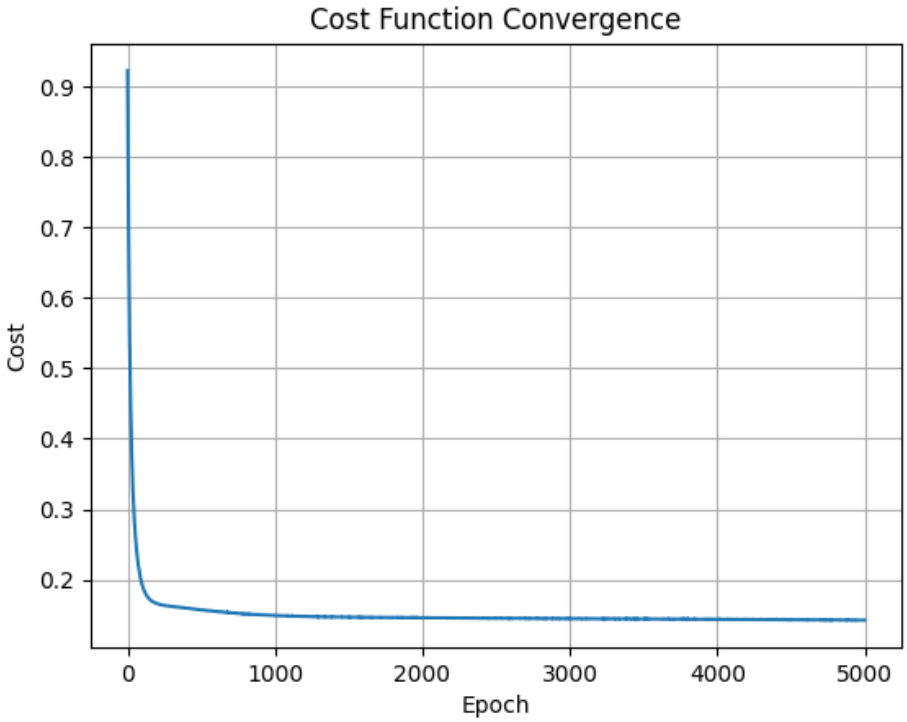

Fig. 2: Convergência da função de custo no treinamento da rede na Função de Classificação *sum_gt_zeros* sem regularização

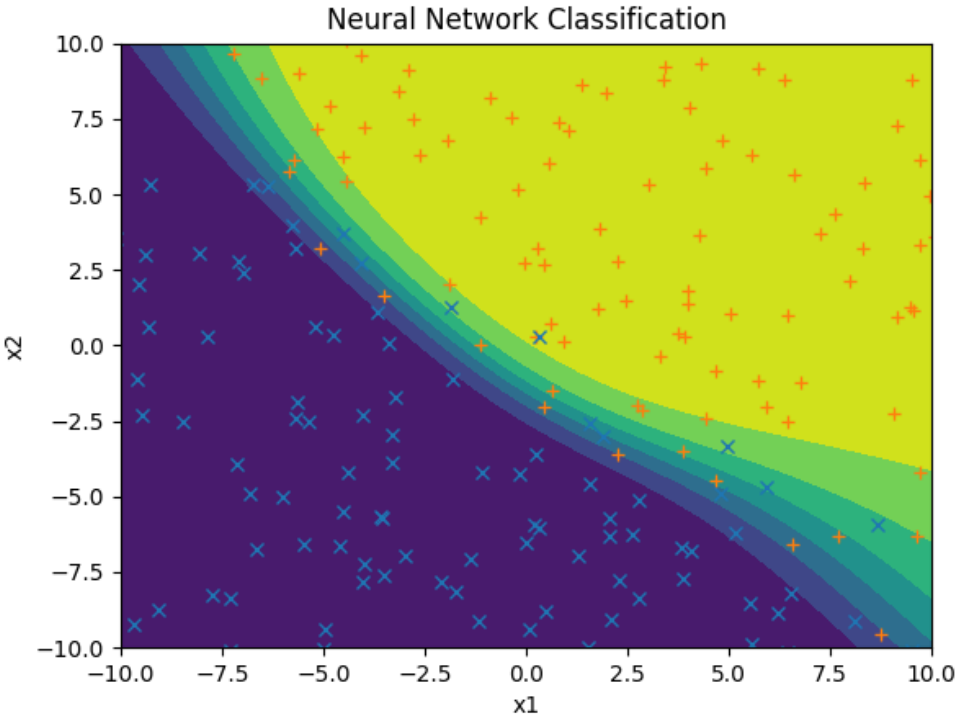

Fig. 3: Resultado do treinamento sem regularização

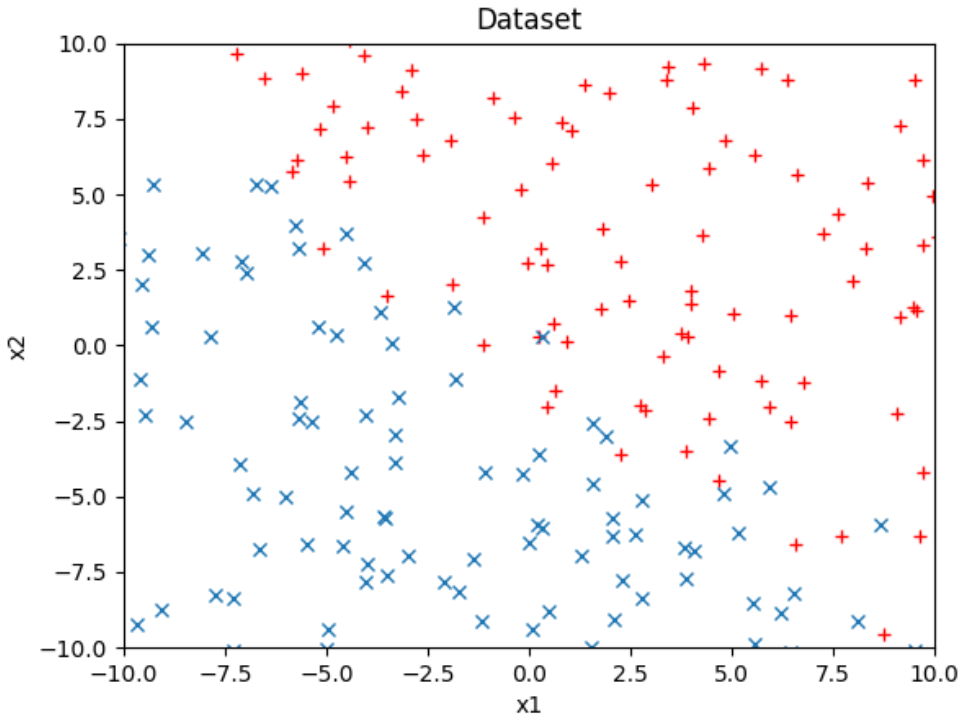

Fig. 4: *Dataset* do conjunto de dados para o treinamento da rede na Função de Classificação *sum_gt_zeros* com regularização

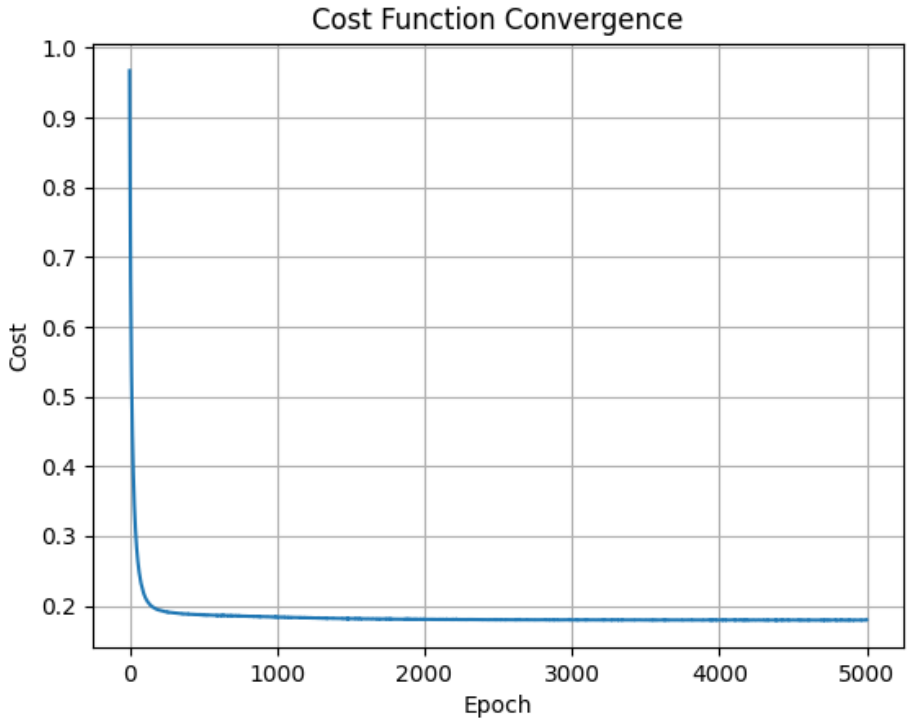

Fig. 5: Convergência da função de custo no treinamento da rede na Função de Classificação *sum_gt_zeros* com regularização

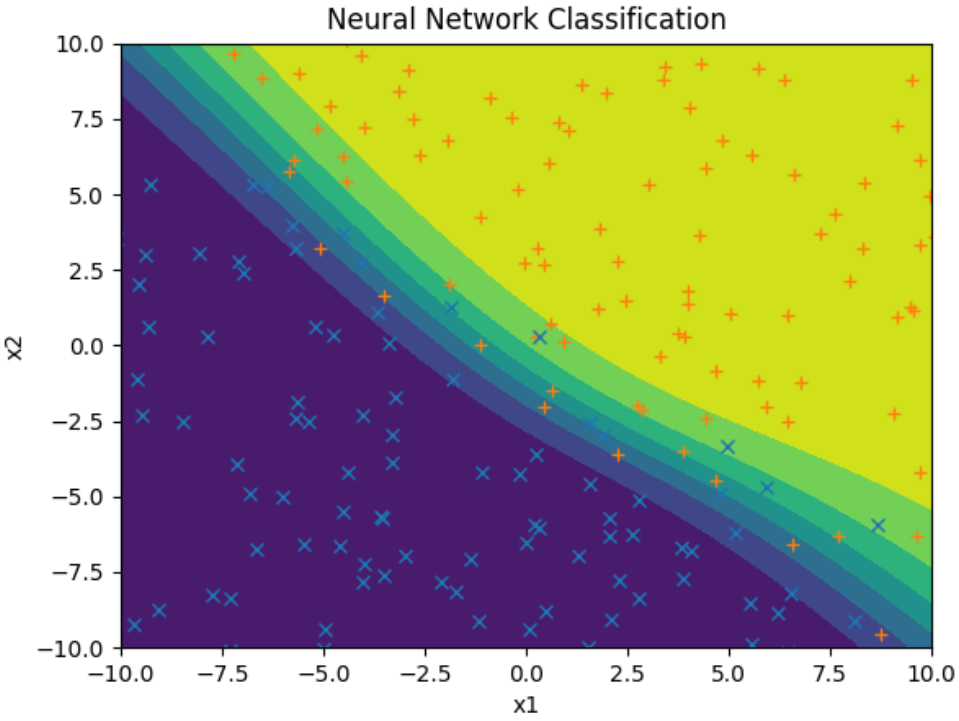

Fig. 6: Resultado do treinamento com regularização

2.2 Função de Classificação *XOR*

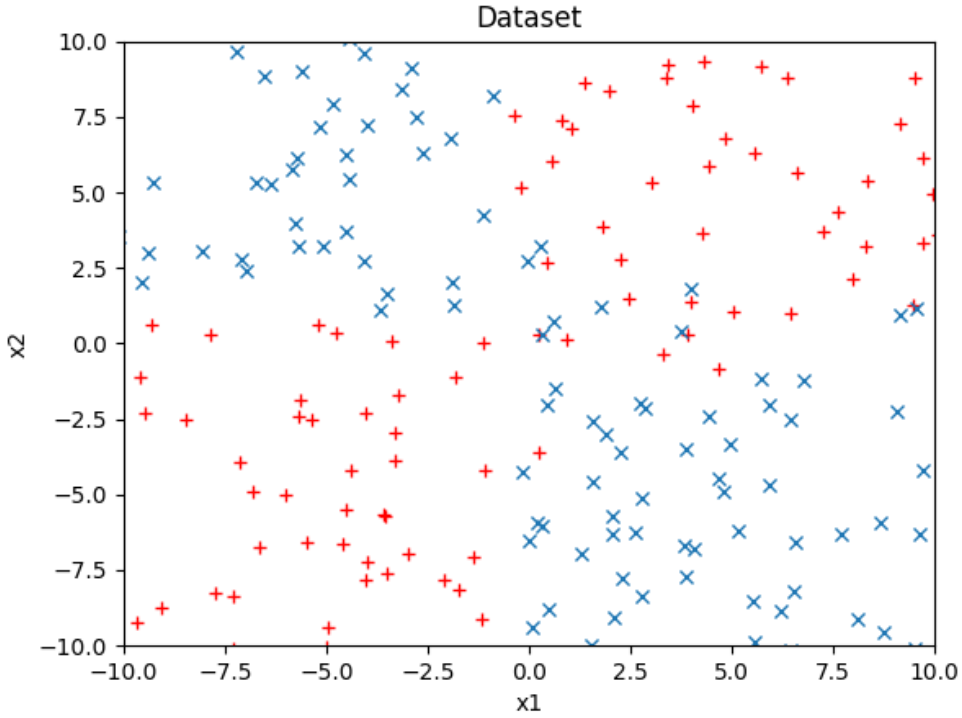

Fig. 7: *Dataset* do conjunto de dados para o treinamento da rede na Função de Classificação *XOR* sem regularização

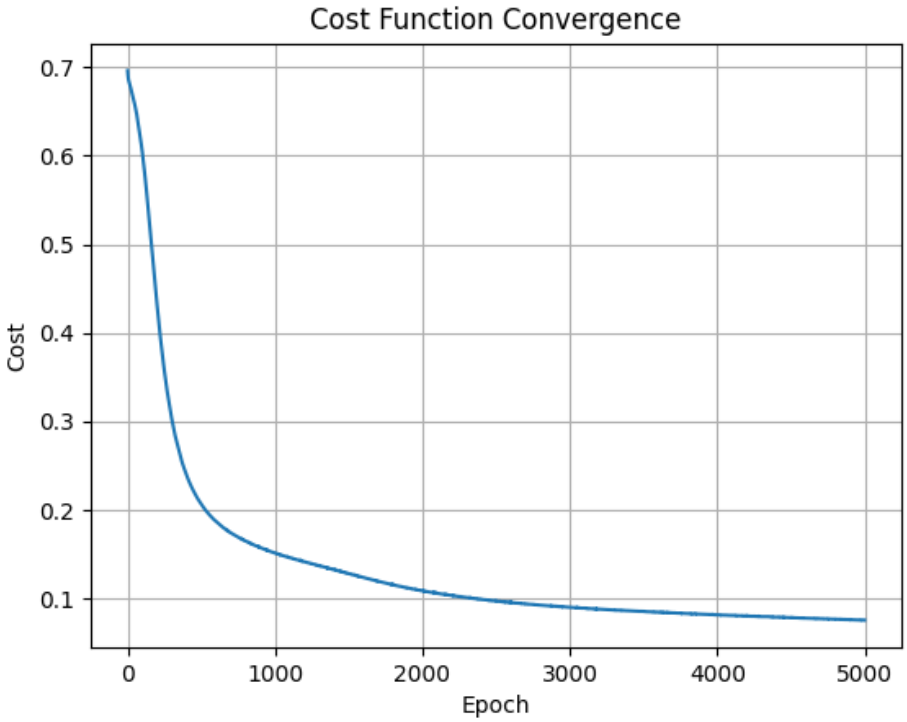

Fig. 8: Convergência da função de custo no treinamento da rede na Função de Classificação *XOR* sem regularização

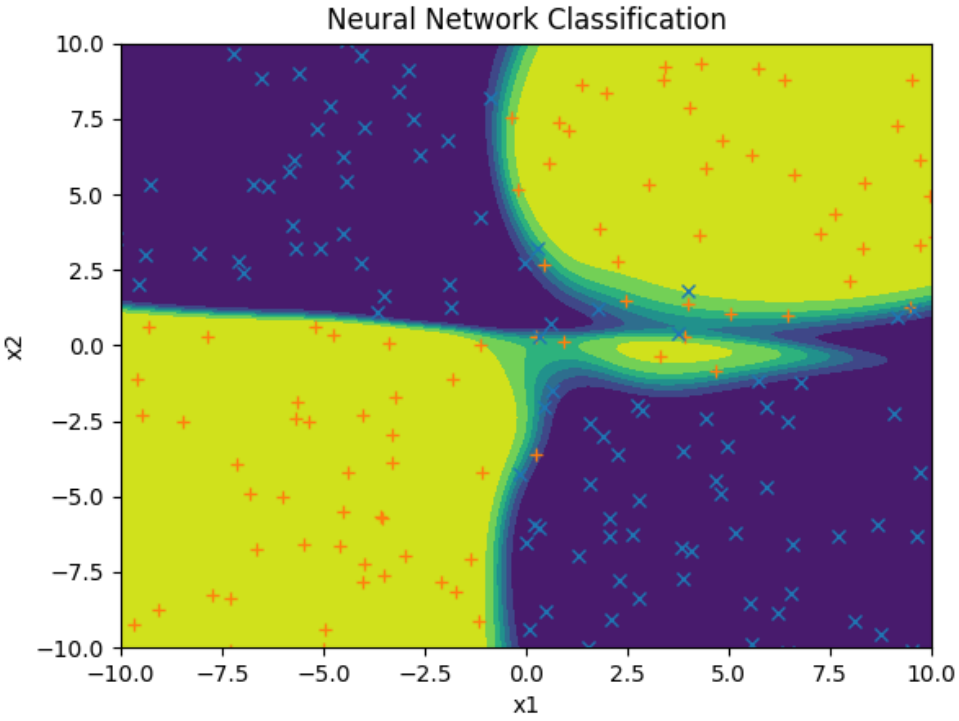

Fig. 9: Resultado do treinamento sem regularização

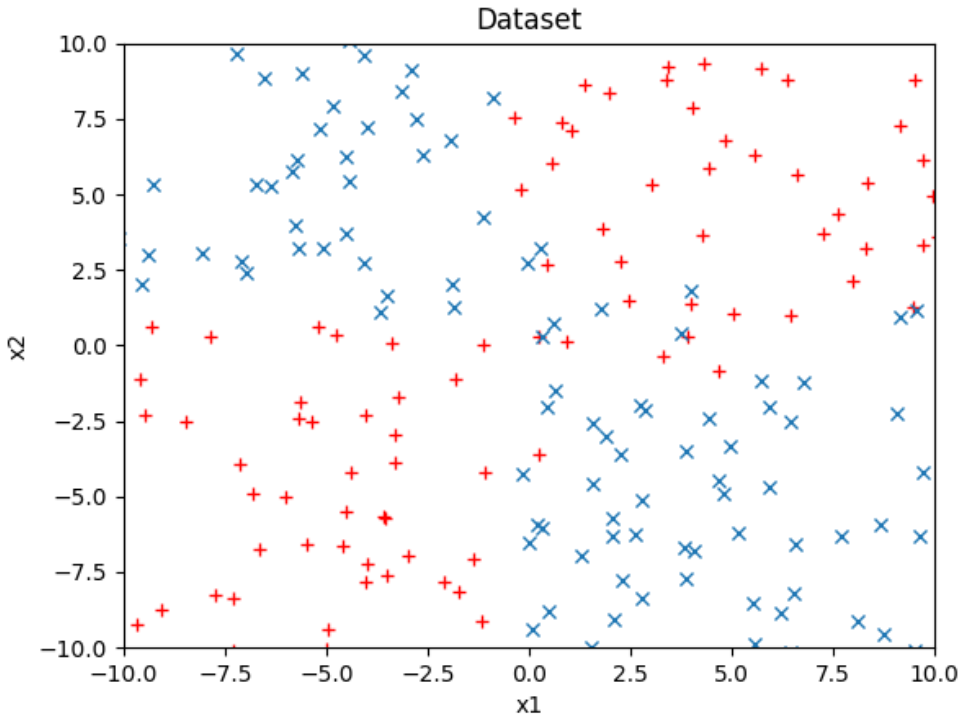

Fig. 10: *Dataset* do conjunto de dados para o treinamento da rede na Função de Classificação *XOR* com regularização

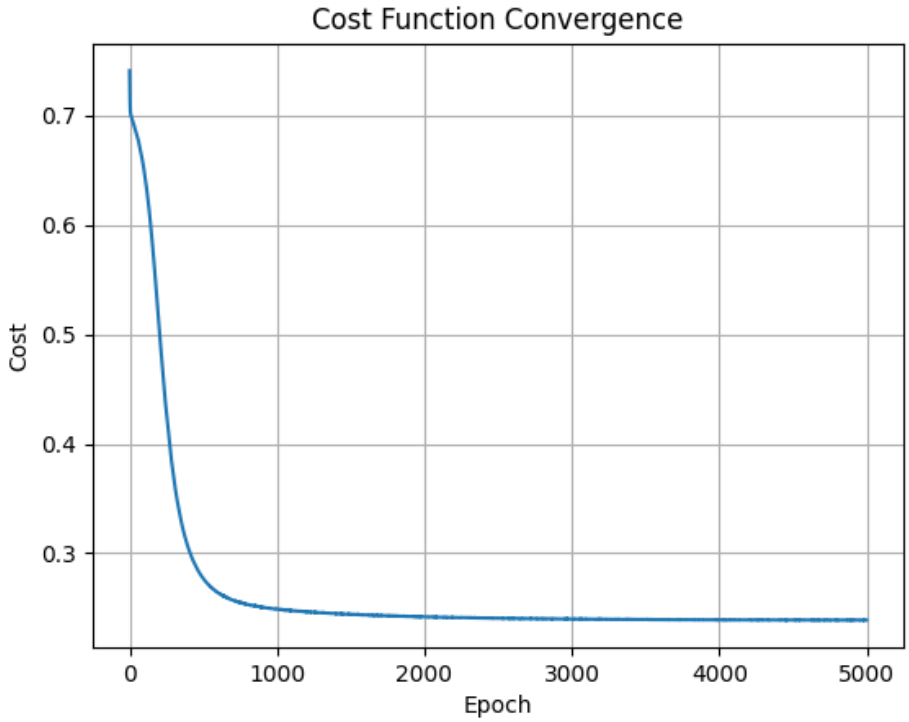

Fig. 11: Convergência da função de custo no treinamento da rede na Função de Classificação *XOR* com regularização

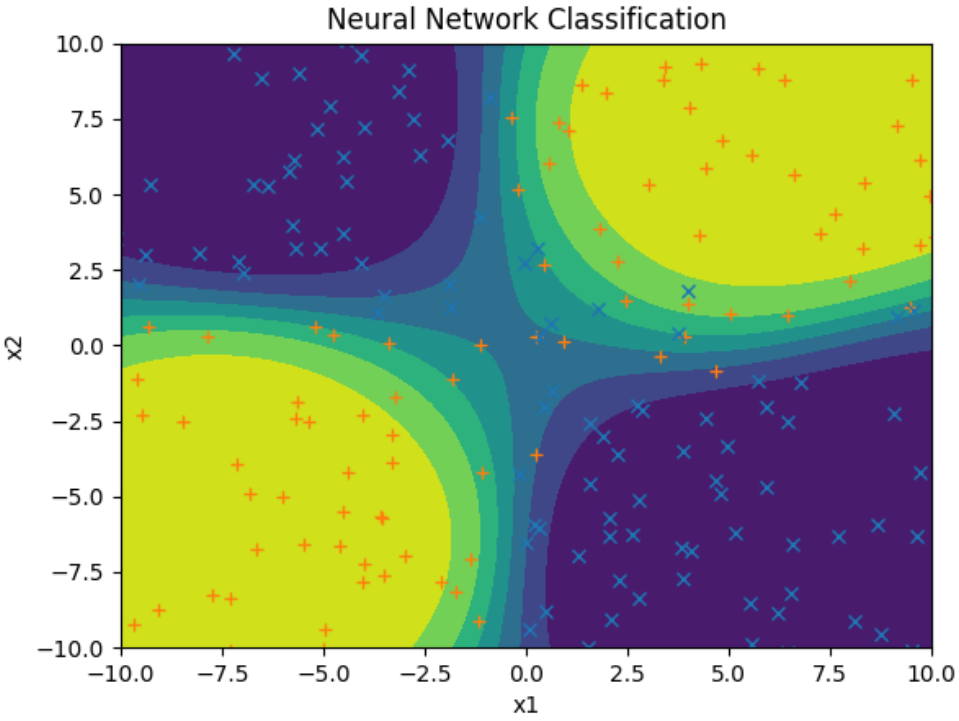

Fig. 12: Resultado do treinamento com regularização

2.3 *Imitation Learning*

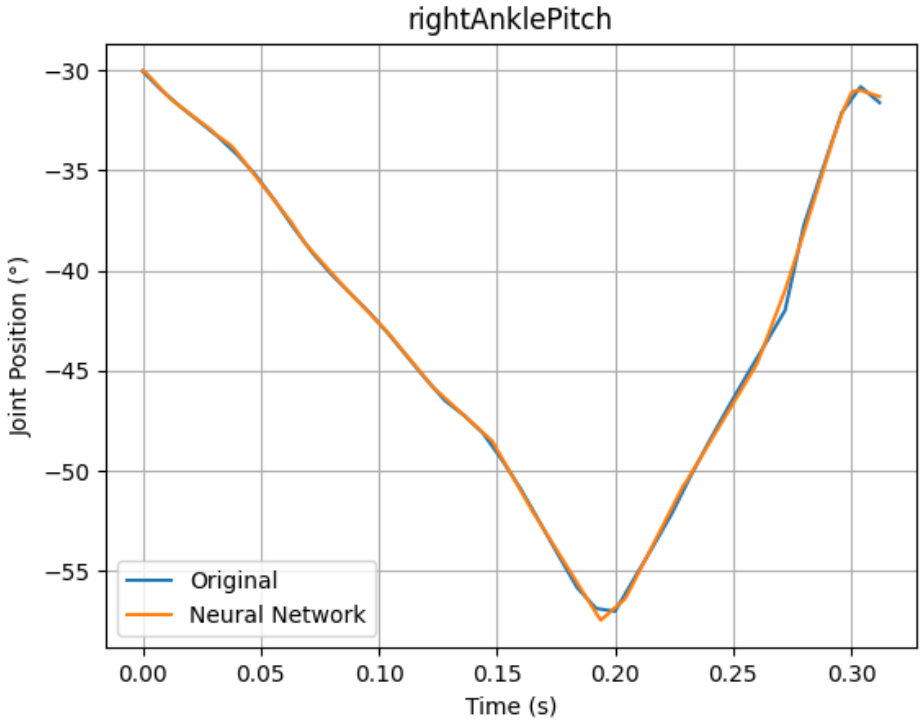

Fig. 13: Gráfico comparativo: inclinação do tornozelo direito

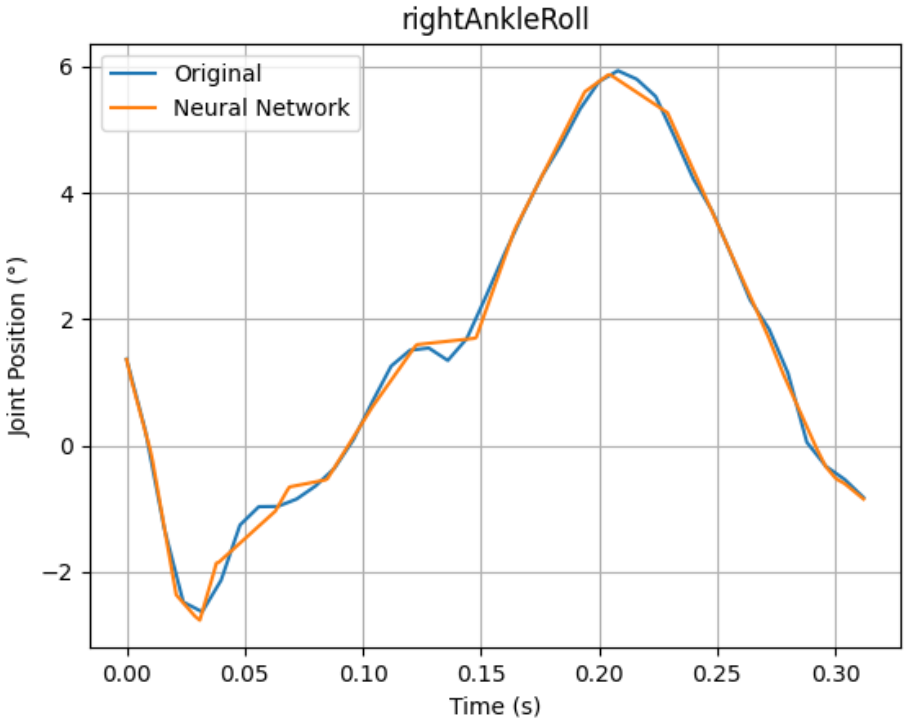

Fig. 14: Gráfico comparativo: rotação do tornozelo direito

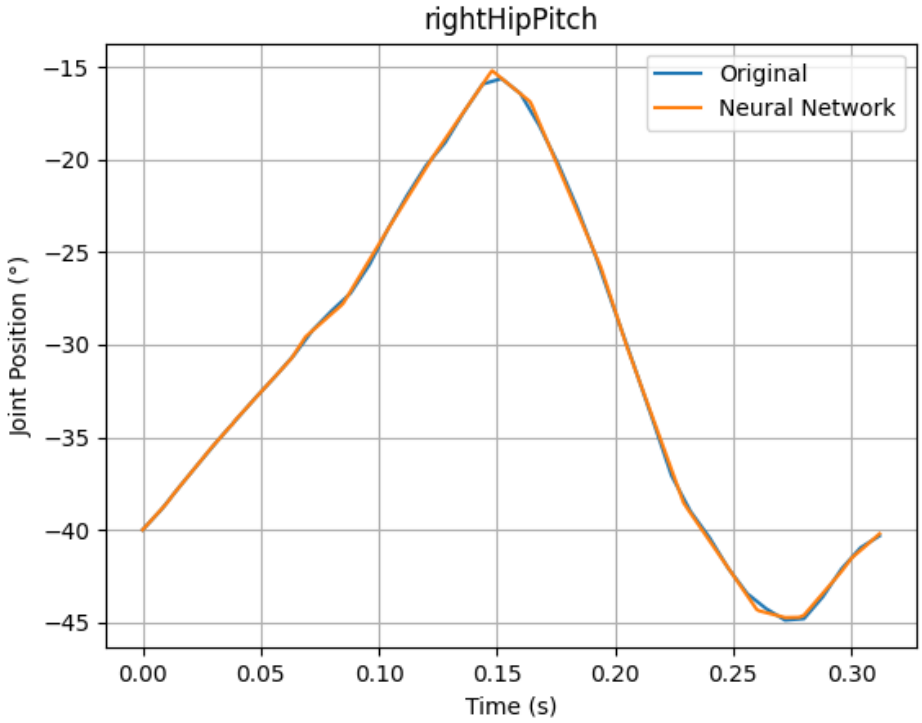

Fig. 15: Gráfico comparativo: inclinação do quadril direito

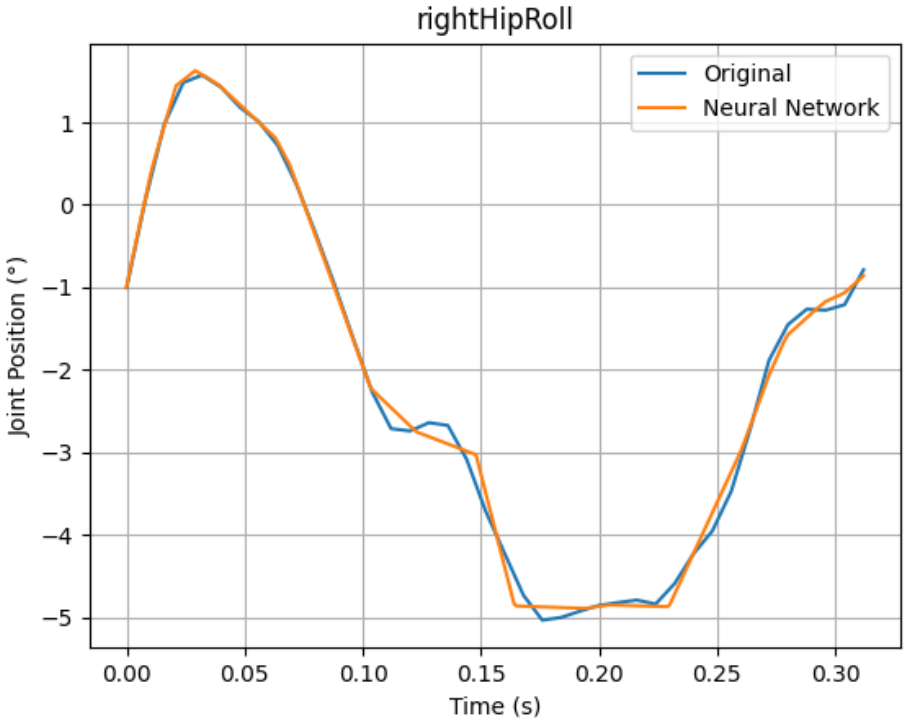

Fig. 16: Gráfico comparativo: rotação do quadril direito

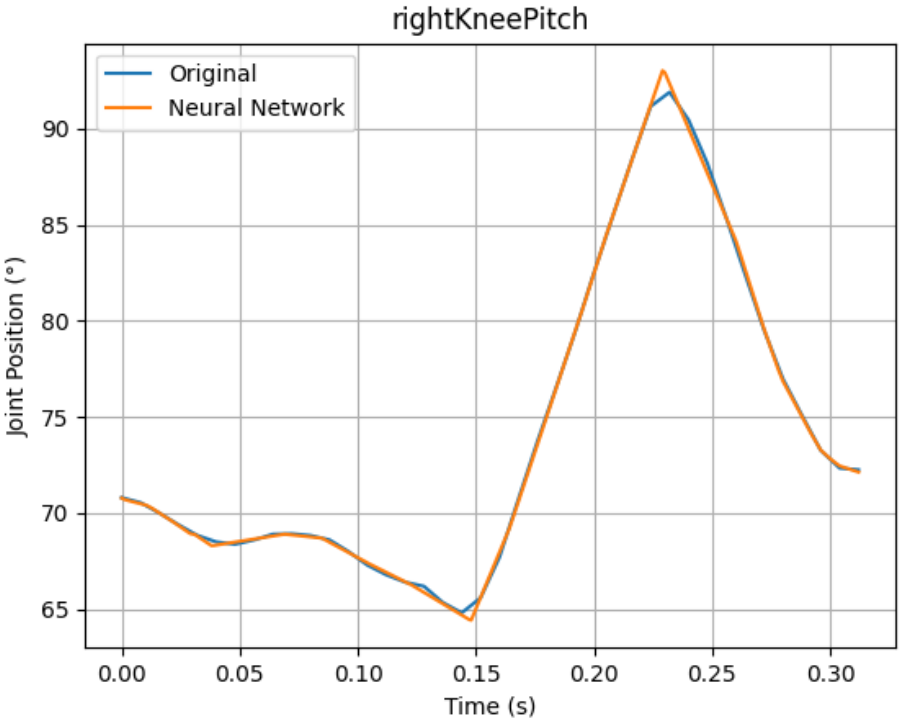

Fig. 17: Gráfico comparativo: inclinação do joelho direito

3 Discussões

3.1 Função de Classificação *sum_gt_zeros*

A função *sum_gt_zeros* apresenta um cenário de classificação relativamente simples, pois é linearmente separável. Isso se reflete claramente na rápida convergência da função de custo durante o treinamento da rede neural, tanto com regularização quanto sem esta. Observa-se, nas Figuras 2 e 5, que o valor da função de custo decresce rapidamente nas primeiras centenas de épocas, atingindo uma região de estabilização com valores abaixo de 0,2 em ambos os casos. Esse comportamento também foi observado no lab anterior, também de redes neurais.

Entretanto, a aplicação da regularização L2 com $\lambda = 0{,}002$ demonstra um pequeno impacto para melhor na curva de convergência, evitando oscilações e flutuações localizadas, além de garantir um custo global menor, conforme mostram os gráficos. Apesar de o efeito não ser fortemente perceptível neste problema (dado seu baixo nível de complexidade), a regularização contribui para tornar o modelo mais robusto, restringindo o crescimento excessivo dos pesos.

A visualização da superfície de decisão obtida (Figuras 3 e 6) reforça essa análise: em ambos os casos, a fronteira de decisão é bem definida e se adapta suavemente ao conjunto de dados. A superfície obtida com regularização,, todavia, é mais suave. Portanto, para a função *sum_gt_zeros*, ainda que a regularização traga resultados um pouco melhores, ela apresenta benefícios marginais, dado que a rede já é capaz de aprender a tarefa de forma eficaz com poucas amostras. Sendo assim, cabe uma ponderação por parte do arquiteto da rede se vale a pena o

custo computacional a mais dado pela regularização em relação ao resultado obtido com ela.

3.2 Função de Classificação *XOR*

A função *XOR* representa um desafio clássico para redes neurais, por não ser linearmente separável (também vista no laboratório anterior). Como consequência, a rede exige maior complexidade na sua estrutura e maior esforço computacional para alcançar um bom desempenho.

Observa-se, nas Figuras 8 e 11, que a convergência da função de custo ocorre de maneira mais lenta em comparação à função *sum_gt_zeros*. Além disso, os valores finais da função de custo permanecem mais altos, especialmente na versão sem regularização. Isso demonstra a maior dificuldade da rede em encontrar um mínimo adequado para o erro, dada a natureza não-linear da tarefa.

A introdução da regularização L2 também mostra efeitos distintos neste cenário: ao passo que ela ajuda a suavizar o treinamento e evitar overfitting local, ela também pode restringir a capacidade de classificação da rede em tarefas mais complexas como essa. A superfície de decisão obtida (Figuras 9 e 12) confirma essa observação, sendo mais suave com regularização, mas em alguns pontos menos aderente aos dados em regiões altamente não-lineares.

Comparativamente, a função *sum_gt_zeros* é mais facilmente aprendida, enquanto a função *XOR* demanda mais ajustes de arquitetura e parâmetros para um desempenho ideal, sendo mais impactada pela regularização.

3.3 *Imitation Learning*

Na parte de *imitation learning*, a rede neural foi treinada com 30.000 épocas, utilizando apenas 40 amostras extraídas de movimentos reais, conforme o comentário do arquivo contendo a pré-implementação indicava. O treinamento foi realizado com o conjunto completo de dados em cada iteração, sem a utilização de mini-batches.

As Figuras 13 até 17 mostram comparações entre os dados originais (extraídos do movimento do robô) e os valores estimados pela rede neural para diferentes articulações. Em geral, observa-se uma boa aproximação da rede em relação aos dados originais, especialmente em movimentos suaves e contínuos, como nas articulações *rightAnklePitch* e *rightHipPitch*.

Entretanto, em articulações como *rightAnkleRoll* e *rightHipRoll*, a rede neural apresenta maior dificuldade em capturar as variações rápidas e os picos abruptos presentes nos dados, indicando sensibilidade a ruídos e pqeuenas oscilações. Isso evidencia uma limitação típica de redes com poucos dados e sem reforço de regularização: a tendência ao superajuste em padrões suaves e dificuldade em generalizar movimentos mais complexos ou ruidosos (repentinos). Essa técnica, conforme visto nas videoaulas, é ainda mais prejudicada quando o sistema é em tempo real, dado o atraso na captura dos dados para processamento, etc.

Ademais, o uso de todo o conjunto de dados em cada iteração sem variação (sem mini-batch) pode ter limitado a capacidade da rede de escapar de mínimos locais, além de reduzir a robustez frente a variações nos dados. É bem provável que, se houvesse um *dataset* com muito mais dados, o treinamento seria melhor otimizado, talvez também com o uso da regularização, a fim de aplicar melhores ajustes nos parâmetros e evitar o *overfitting*. Ainda assim, considerando o pequeno número de amostras e a natureza regressiva da tarefa, os resultados são satisfatórios

para uma abordagem inicial de *imitation learning*. Não obstante, o número elevado de iterações (30 mil épocas), ajudou a rede neural a contornar os poucos dados existentes, garantindo um melhor ajuste ao movimento original, conforme visto nos gráficos comparativos.