Estudo Empírico do Aprendizado de Redes Neurais Artificiais com Base no Princípio do Gargalo da Informação

Thiago Vinicius de Sousa Barroso[1], Otávio Noura Teixeira[1]

[1]Faculdade de Engenharia de Computação - Universidade Federal do Pará
Campus Universitário de Tucuruí - PA - Brasil
thiago.barroso@tucurui.ufpa.br, otaviont@ufpa.br

Abstract. *This work examines the behavior of internal representations in Artificial Neural Networks, utilizing information-based techniques. This approach assumes that, during the training process, the model progressively absorbs and compresses information from the input of the data while simultaneously becoming proficient in processing the output, preserving only its relevant aspects. Experiments were conducted with Multilayer Perceptron (MLP) and Convolutional Neural Networks (CNN) models using two distinct datasets. The results reinforce the expected behaviors. Thus, this study contributes to the theoretical understanding of the internal working of deep neural networks, offering insights to make their process more transparent.*

Resumo. *Este trabalho estuda o comportamento das representações internas de Redes Neurais Artificiais utilizando técnicas baseadas em informação. Essa abordagem considera que, durante o processo de treinamento, o modelo absorve e comprime progressivamente a informação dos dados de entrada ao mesmo tempo que se torna competente em processar a saída, preservando apenas seus aspectos relevantes. Foram conduzidos experimentos com modelos do tipo Multilayer Perceptron (MLP) e Convolutional Neural Networks (CNN) utilizando dois conjuntos de dados distintos. Os resultados reforçam os comportamentos esperados. Assim, este estudo contribui para o entendimento teórico do funcionamento interno de redes neurais profundas, oferecendo subsídios para tornar seus processos mais transparentes.*

1. Introdução

O desenvolvimento de modelos inteligentes é mais antigo do que possa parecer. Desde a introdução do conceito do *Perceptron* na década de 1950 [Rosenblatt 1958], inspirado no cérebro humano, os pesquisadores já demonstravam interesse em compreender, estudar e adaptar o aprendizado humano para sistemas computacionais. Entretanto, foi a partir da disponibilidade de grandes conjuntos de dados, técnicas avançadas de processamento paralelo, plataformas de software com integralização em GPUs e algoritmos de otimização mais sofisticados que as Redes Neurais Profundas ganharam novamente seu espaço [Sengupta 2020].

Nos últimos anos, o estudo desses modelos tem ganhado destaque não apenas pela sua eficiência em tarefas complexas como Reconhecimento de Padrões, Visão Computacional e Processamento de Linguagem Natural, mas também pela necessidade crescente de compreender melhor seus processos de aprendizado e funcionamento interno [Goodfellow

2016]. A chamada “caixa-preta” das Redes Neurais levanta desafios teóricos e práticos, ao impulsionar pesquisas voltadas para a interpretabilidade e confiabilidade desses modelos [Lipton 2018], [Doshi-Velez 2017] e [Zhang 2018].

Um melhor entendimento do aprendizado dos modelos pode trazer benefícios significativos, como a redução do alto custo computacional durante o treinamento [Liu 2017], a diminuição da dependência de grandes volumes de dados para obter bons resultados [Rather 2024] e a mitigação de vieses e questões éticas inerentes a modelos complexos [Rudin 2019].

Dessa forma, compreender melhor o seu funcionamento interno, além de impulsionar avanços tecnológicos, também permite o desenvolvimento de soluções mais eficientes, transparentes e responsáveis, o que justifica o desenvolvimento deste estudo. Sendo assim, o objetivo principal deste estudo é investigar empiricamente o comportamento das representações internas de Redes Neurais Profundas sob a perspectiva do Gargalo da Informação.

2. Fundamentação Teórica

2.1. Redes Neurais Profundas e o Gargalo da Informação

O campo da Teoria da Informação foi formalmente estabelecido em [Shannon 1948], esse estudo forneceu ferramentas matemáticas importantes para quantificar a informação. Estas ferramentas se mostram úteis no campo de aprendizado de máquina, provendo soluções alternativas para estudar o comportamento dos modelos.

O estudo das Redes Neurais Artificiais vai além da busca pela maior precisão nos modelos. Um aspecto fundamental é a compreensão de como a informação é processada e transformada ao longo das camadas da rede. Nesse contexto, o Gargalo da Informação [Tishby 1999] surge como uma abordagem que busca explicar como os modelos aprendem representações úteis dos dados de entrada, ao mesmo tempo em que descartam informações irrelevantes.

Baseado na Teoria da Informação, [Tishby 2015] e [Schwartz-Ziv 2017] propõem que durante o treinamento, as redes passam por duas fases principais: extração de características e compressão da informação.

Inicialmente, as camadas ajustam seus pesos para capturar o máximo de padrões nos dados, ampliando a quantidade de informação que retêm. No entanto, com o avanço do treinamento, o modelo começa a descartar detalhes redundantes ou ruidosos, mantendo apenas os aspectos mais relevantes para a tarefa. Esse comportamento pode ser quantificado por meio do cálculo da informação mútua entre as ativações das camadas internas junto dos dados de entrada e saída do *dataset* usado durante o treino.

A aplicação do Gargalo da Informação possui implicações diretas na interpretabilidade, eficiência e confiabilidade dos modelos. Ao entender como e quando ocorre a compressão da informação, é possível desenvolver estratégias que favorecem arquiteturas mais robustas, com menor tendência ao *overfitting* e maior capacidade de

generalização, como em [Kawaguchi 2023] e [Chelombiev 2019]. Além disso, representações mais compactas e informativas tendem a ser mais interpretáveis, o que é desejável em aplicações críticas [Rudin 2019].

2.2. Entropia

Proposta por Claude Shannon na década de 1940 [Shannon 1948], a entropia é uma das medidas fundamentais da Teoria da Informação. Ela quantifica o grau de incerteza ou aleatoriedade de uma variável aleatória, ou seja, mede a quantidade média de informação contida em uma mensagem. Em termos formais, para uma variável discreta X com distribuição de probabilidade $P(x)$, a entropia é dada pela equação 1.

$$H(X) = -\sum_{i=1}^{n} p\left(x_i\right) * log_2\left(p\left(x_i\right)\right) \quad (1)$$

Essa expressão na prática indica que quanto mais imprevisível for a variável X, maior será sua entropia. Por outro lado, se X assume sempre o mesmo valor, sua entropia será zero, pois os dados são previsíveis, logo não existe incerteza.

No contexto deste trabalho, a entropia se mostrou fundamental para calcular medidas mais complexas, como a Entropia Condicional e a Informação Mútua, que são ferramentas importantes para estudar o comportamento das Redes Neurais dentro do contexto de informação. Essas medidas se mostram importantes por oferecer uma forma quantitativa do quanto de Informação Mútua uma camada intermediária qualquer T de um modelo armazena de informação tanto em relação às informações de entrada e saída do *dataset*.

2.3. Entropia Condicional

A Entropia Condicional amplia o conceito de entropia ao considerar uma relação entre duas variáveis aleatórias. Em resumo, ela mede o grau de incerteza que ainda permanece sobre uma variável X, dado que já é conhecido o valor de outra variável Y.

Em outras palavras, indica o quanto de informação em X ainda é necessária para descrever Y, ou, de forma complementar, o quanto conhecer Y ajuda a reduzir a incerteza sobre X. Formalmente a Entropia Condicional é calculada conforme expressa na equação 2:

$$H(X|Y) = -\sum_{j=1}^{m}\sum_{i=1}^{n} p\left(x_i, y_j\right) * log_2\left(p\left(x_i|y_j\right)\right) \quad (2)$$

Essa medida é útil porque permite avaliar a dependência entre variáveis. Se conhecer Y não contribui para conhecer X, então $H(X|Y)$ será alta. Entretanto se Y ajudar a conhecer X, então $H(X|Y)$ se aproximará a zero à medida em que as duas variáveis forem semelhantes.

2.4. Informação Mútua

A Informação Mútua é uma extensão do conceito de entropia, sendo uma medida estatística que quantifica a informação compartilhada entre duas variáveis aleatórias. Em outras palavras, ela indica o quanto do conhecimento sobre uma variável reduz a incerteza sobre outra.

Essa medida é particularmente relevante para o estudo de Redes Neurais Profundas, pois permite analisar como a informação dos dados é propagada e transformada ao longo das camadas do modelo, e é matematicamente definida pela equação 3.

$$I(X;Y) = H(X) - H(X|Y) \tag{3}$$

Onde *H(X)* representa a entropia de X, e *H(X|Y)* é a entropia condicional de *X* dado *Y*. Essa representação expressa o ganho de informação sobre *X* quando se conhece *Y*, ou vice-versa.

No contexto deste estudo, utiliza-se a informação mútua para medir o quanto uma camada intermediária *T* retém informação sobre os dados de entrada *X* e sobre a saída esperada *Y*, por meio de dois componentes principais:

- **Compressão da Informação - *I(X;T)*:** mede o quanto da informação original dos dados de entrada *X* é preservada em uma camada intermediária *T*. Durante o treinamento, a rede inicialmente armazena muita informação sobre os dados brutos, porém, conforme aprende, começa a descartar detalhes irrelevantes, ao reduzir *I(X;T)*. Esse processo é importante para a generalização do modelo, pois impede que ele memorize características específicas dos dados de treinamento e foque apenas nos aspectos mais relevantes para a tarefa que foi proposta.
- **Previsibilidade da Informação - *I(T;Y)*:** mede o quanto as representações geradas pela camada *T* estão relacionadas com a saída *Y*. Um aumento nesse valor indica que a camada está produzindo representações cada vez mais úteis para a tarefa em questão.

O objetivo do treinamento é maximizar *I(T;Y)* ao mesmo tempo que *I(X;T)* é reduzido de forma controlada. Como resultado, essa análise mostra duas fases distintas durante o treinamento da rede. Sendo essas:

- **Fase de Memorização:** nos estágios iniciais do treinamento, *I(X;T)* aumenta porque a rede está absorvendo uma grande quantidade de detalhes dos dados de entrada. Esta etapa é importante para que a rede tenha acesso às características mais completas e genéricas dos dados, mesmo que isso inclua ruídos e padrões irrelevantes.
- **Fase de Compressão:** conforme a rede aprende, *I(X;T)* diminui, ao indicar que a rede está removendo redundâncias e retendo apenas as informações mais relevantes para prever *Y*. Essa compressão reflete o processo de filtragem da

informação, ao tornar as representações mais abstratas e compactas. Idealmente, essa redução acontece sem prejudicar *I(T;Y)*, ou seja, ao manter a capacidade da rede de prever corretamente a saída com base nas representações aprendidas.

Essa dinâmica pode ser visualizada pelo Plano da Informação, uma ferramenta visual em que é possível observar cada camada da rede representada como um ponto, e sua trajetória ao longo do treinamento revela como o modelo evolui na representação dos dados.

Essa análise tem se mostrado útil para diagnosticar se os treinamentos estão sendo eficientes, identificar camadas que estão redundantes ou pouco utilizadas e também ajudar a guiar a modelagem de arquiteturas mais compactas e interpretáveis.

2.5. Rede Neural Multilayer Perceptron (MLP)

A arquitetura *Multilayer Perceptron* (MLP) [Popescu 2009], foi uma evolução do *Perceptron* que era apenas uma única unidade de processamento isolada. Nesta arquitetura, os neurônios são organizados em uma estrutura sequencial formando camadas, onde cada camada transmite suas saídas para a próxima, formando uma rede densa, conforme é ilustrado na figura 1.

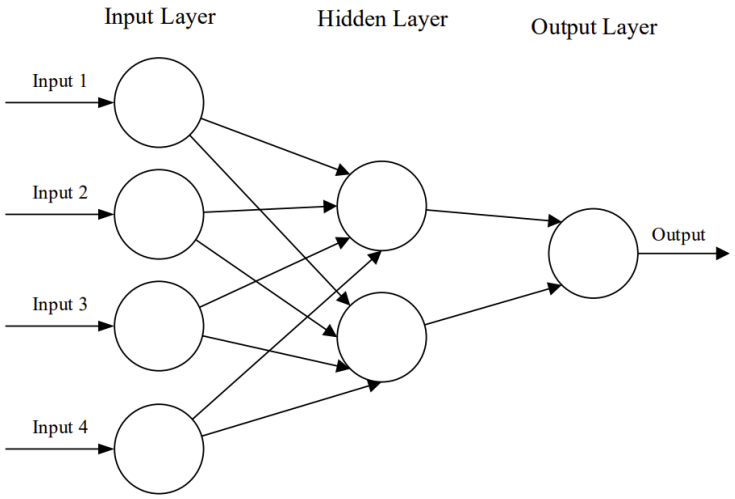

Figura 1. Exemplo de arquitetura de uma MLP (fonte: O'Shea 2015).

Essa arquitetura se consolidou ainda mais com a introdução do Algoritmo de Retropropagação (*Backpropagation*) [Rumelhart 1986] , ao possibilitar um treinamento mais eficiente para estes modelos.

Cada camada realiza uma combinação linear dos valores de entrada, seguida pela adição de um viés interno não relacionado com os dados de treino e, por fim, a aplicação de uma função de ativação não-linear. Esse processo é igualmente repetido em cada camada do modelo, fazendo a informação de entrada fluir até a saída, onde o resultado final é chamado de predição (*Output*) da rede.

A quantidade de camadas é o parâmetro que muitas vezes define uma Rede Neural Profunda. Quanto mais camadas adicionadas, mais capacidade de interpretação dos dados o modelo é capaz de capturar e aprender. Entretanto, como efeito colateral tem-se uma maior demanda de processamento para que os dados de entrada fluam até a saída. E, por consequência, aumenta tanto o tempo de uma simples predição quanto o tempo de todo o seu treinamento.

2.6. Rede Neural Convolucional (CNN)

A Rede Neural Convolucional (CNN), inicialmente proposta em [LeCun 1995] e mais tarde muito popularizada [O'Shea 2015], é frequentemente aplicada em problemas de visão computacional, como classificação de imagens e reconhecimento de padrões.

O sucesso das CNNs modernas foi consolidado em [Krizhevsky 2012], que introduziu a AlexNet e revolucionou a competição ImageNet. A partir de então, diversas arquiteturas foram propostas, como a VGG [Simonyan 2014] e a ResNet [He 2016].

Diferente das MLPs, as CNNs (ilustradas na figura 2) possuem ao menos uma camada convolucional, normalmente no início da arquitetura. Essa camada é responsável por processar dados em formato de grade, podendo ser 2D (imagens em escala de cinza) ou 3D (imagens coloridas) e realizar uma operação que reconhece padrões nas imagens com base em seus parâmetros (Filtros Convolucionais), que extraem padrões relevantes, como bordas, texturas e formas, contribuindo para o processo de reconhecimento de estruturas mais complexas nas camadas seguintes, esse tipo de arquitetura favorece áreas como a Visão Computacional, ao alimentar o modelo com imagens ou vídeos (sequências de imagens).

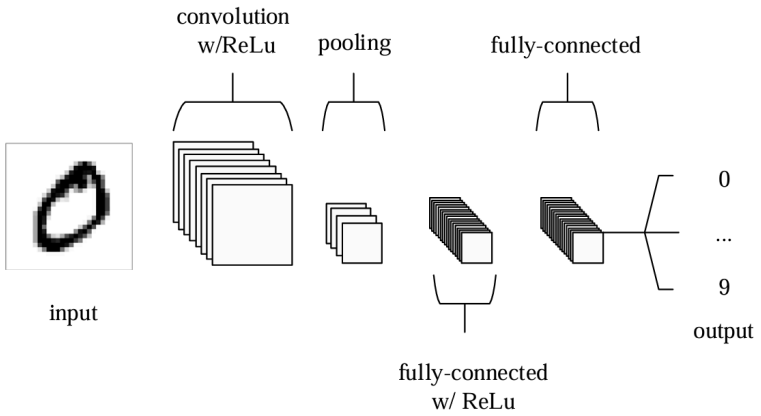

Figura 2. Exemplo de arquitetura de uma CNN (fonte: O'Shea 2015).

3. Metodologia

3.1. Ambiente de experimentos

Todos os testes foram conduzidos com uso da linguagem de programação Python devido à sua ampla adoção na área de Aprendizado de Máquina e à variedade de bibliotecas especializadas disponíveis. Entre as principais bibliotecas utilizadas destacam-se:

- *Keras*: biblioteca de alto nível que faz interface para bibliotecas como *TensorFlow* e *Pytorch*, empregada para a implementação, treinamento e avaliação dos modelos, muito útil por ser mais simples de usar e fácil de escalar;
- *NumPy*: utilizada para manipulação de *arrays* multidimensionais e realizar operações aritméticas;
- *Scikit-learn*: aplicada na etapa de pré-processamento dos dados, especialmente na separação entre conjuntos de treino e teste;
- *Matplotlib*: implementada para a geração de gráficos e representações visuais dos resultados experimentais.

Essas ferramentas, em conjunto, possibilitaram a implementação, execução e análise dos experimentos realizados ao longo deste estudo.

Além do uso dessas bibliotecas, o código responsável pela captura das ativações internas, discretização dos valores, cálculo das medidas de entropia e informação mútua, construção dos planos da informação, foi implementado pelo autor especificamente para este trabalho e está disponível publicamente em repositório do *GitHub*[1], e serve como contribuição prática e como ferramenta de apoio para futuras pesquisas.

3.2. Arquiteturas utilizadas

Na maioria dos experimentos realizados, a arquitetura adotada foi a MLP, devido à sua simplicidade de construção e baixa complexidade computacional, foi possível realizar diversas rodadas de simulações para cada exemplo de conjunto de dados, além de diferentes configurações do mesmo modelo. Mais ao final foram implementados testes com CNNs, utilizando as mesmas configurações e abordagens dos modelos anteriores.

3.3. Conjunto de dados

Na primeira fase experimental deste estudo, utilizou-se o *dataset* sintético proposto por [Schwartz-Ziv 2017]. O *dataset* é composto por 12 variáveis de entrada binárias, que representam pontos uniformemente distribuídos sobre uma esfera 2D. Como cada variável pode assumir apenas os valores 0 ou 1, com o total de combinações possíveis sendo de 2^{12}, o conjunto de dados inteiro totaliza 4096 amostras.

[1] https://github.com/thag0/Information-Bottleneck

Esse cenário extenso e variado é ideal para observar com maior clareza como a informação é processada e comprimida nas diferentes camadas do modelo além de ser simples de ser reproduzido e expandido, sua estrutura pode ser melhor entendida visualmente conforme aponta a figura 3.

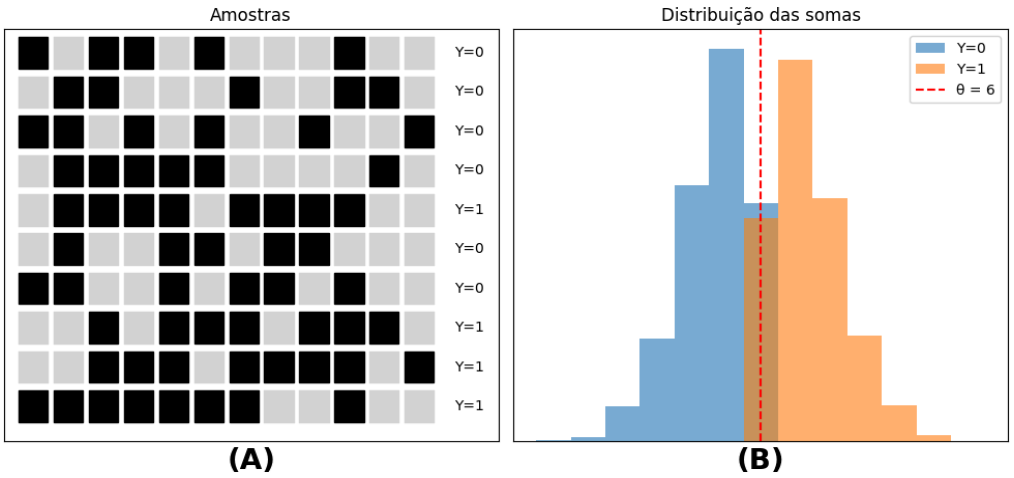

Figura 3. Exemplo do conjunto de dados sintéticos, onde (A) representa o conjunto de entrada e saída, e (B) demonstra como os dados estão distribuídos.

Conforme proposto em [Schwartz-Ziv 2017], os dados de entrada são variáveis binárias geradas de forma aleatória (figura 3A); essas variáveis são somadas e, caso o valor ultrapasse um limiar predefinido (representado pela linha pontilhada em vermelho, na figura 3B), o seu resultado será 1, caso contrário será 0.

Mais adiante utilizou-se o *dataset* MNIST, um conjunto amplamente conhecido que contém imagens de dígitos manuscritos de 0 a 9 em escala de cinza, conforme ilustrado na figura 4. Cada amostra é uma imagem em escala de cinza com uma dimensão de 28×28 *pixels*, onde cada *pixel* é representado por um valor contínuo entre 0 e 255, e indica a intensidade da cor (0 correspondente ao preto e 255 correspondente ao branco). Esse conjunto foi escolhido por ser um problema de classificação visual realista, permitindo avaliar a aplicabilidade da análise de informação mútua em um cenário mais complexo e real.

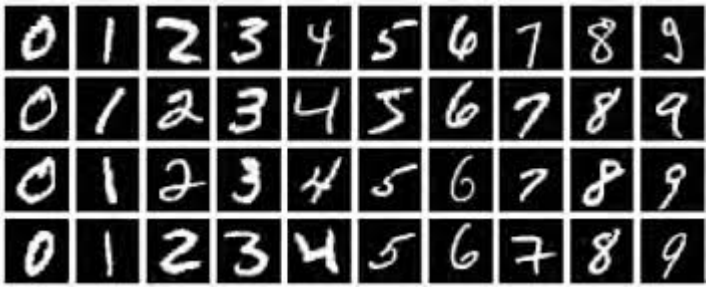

Figura 4. Exemplo de dados do conjunto MNIST.

3.4. Escolha de hiperparâmetros

Todos os testes realizados foram programados com uma pequena quantidade de épocas, visto que as bibliotecas atuais são otimizadas para que os modelos aprendam com qualidade e em tempo reduzido.

O otimizador escolhido foi o *Stochastic Gradient Descent* (SGD) com taxa de aprendizado e *momentum* (uma variável adicional opcional) alterados propositalmente com valores pequenos com o objetivo de acompanhar suavemente a evolução do aprendizado dos modelos.

As camadas ocultas foram configuradas com a função de ativação Tangente Hiperbólica que possui um comportamento semelhante à *Sigmoid*, e permite a adição de mais camadas sem sofrer tanto com o *Vanish Gradient* (figura 5). Isto é um problema comum que ocorre durante o processo de retropropagação onde os gradientes usados para atualizar os pesos da rede se tornam muito pequenos e não conseguem aplicar suas correções apropriadamente, principalmente nas camadas iniciais do modelo [Goodfellow 2016].

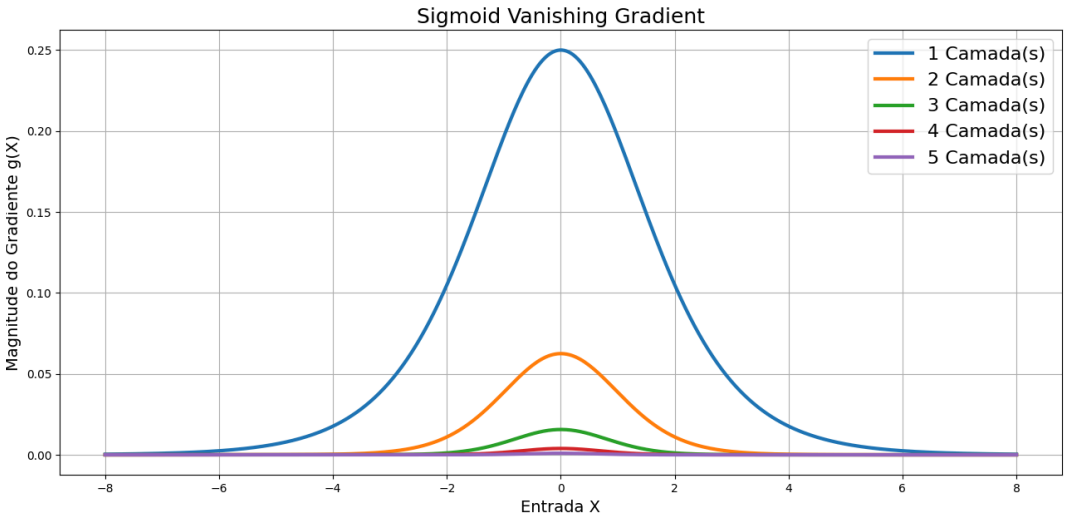

Figura 5. Comparativo do comportamento dos gradientes da função Sigmoid quando camadas são empilhadas em um modelo (fonte: Autor).

Mais adiante no avanço dos experimentos a função de ativação das camadas ocultas foi alterada pela ReLU (Rectified Linear Unit), que possui comportamento simples, caso o valor resultante da operação interna do neurônio seja maior que zero, a saída será o próprio valor resultante, caso contrário será zero. Na figura 6 é possível observar um comparativo entre as das funções utilizadas nas camadas intermediárias.

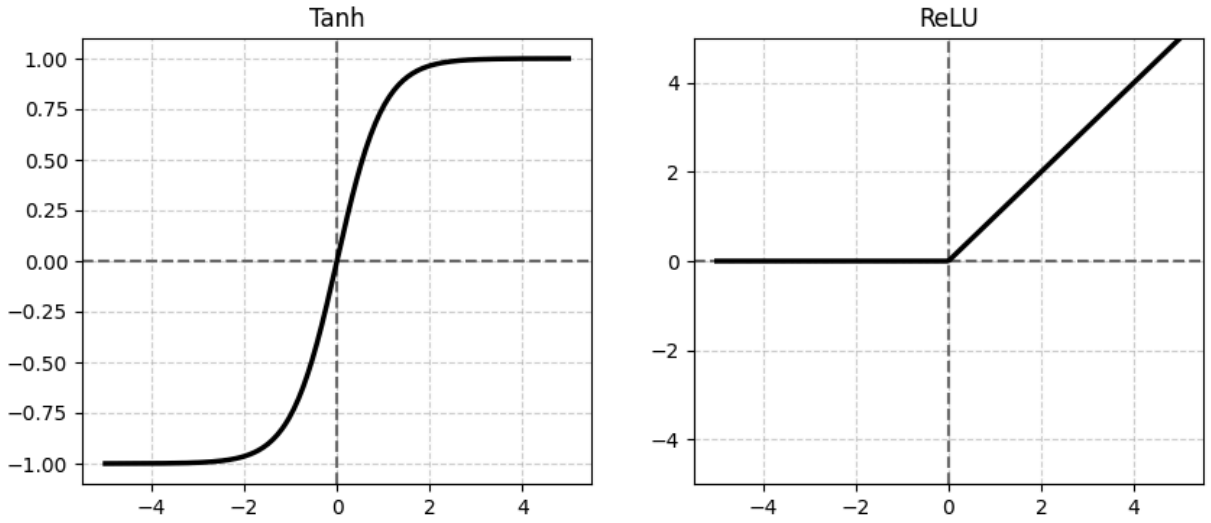

Figura 6. Comparação entre o comportamento a função de ativação TanH junto da função ReLU.

Por fim, na camada de saída foi configurada a função de ativação *Sigmoid para os modelos MLP* por ser prática e fácil de interpretação dos resultados, além da função de ativação Softmax nos modelos CNN para transformar as predição em distribuições de probabilidade.

3.5. Treinamento e captura de informações

Os modelos utilizados foram treinados por meio do Aprendizado Supervisionado [Nasteski 2017], uma abordagem utilizada em tarefas de classificação e regressão. Nesse tipo de treinamento, o modelo recebe exemplos compostos por entradas junto de suas respectivas saídas (ou rótulos), permitindo aprender a mapear corretamente os dados de entrada para as respostas desejadas. O seu objetivo é ajustar os parâmetros do modelo de forma a minimizar os erros de predição, processo guiado por uma função de perda ou função de custo.

A função de perda é responsável por quantificar o erro cometido pelo modelo ao comparar suas previsões com os rótulos corretos, quanto maior o erro, maior o valor retornado por ela. Esse valor é utilizado pelo algoritmo de otimização, no caso o SGD, para atualizar os parâmetros da Rede Neural, de modo a reduzir o erro ao longo das iterações do treinamento.

Além disso, também é possível monitorar a acurácia do modelo. Essa métrica indica o percentual de acertos nas previsões realizadas e pode ser representada graficamente ao longo do tempo. Espera-se observar o oposto do comportamento da função de perda, uma curva crescente de acurácia, conforme o treinamento avança, ao indicar que a rede está com uma taxa de acerto cada vez maior ao longo do processo.

Para a geração do Plano da Informação, inicia-se com a captura das ativações de cada camada do modelo em cada época do treinamento. Em seguida, transformam-se os valores

contínuos das ativações em valores finitos discretizados por meio de uma técnica de discretização por *bins* (*Binning*), que divide o intervalo de valores em faixas iguais e substitui cada valor pelo índice da faixa correspondente.

Essa abordagem permite transformar ativações contínuas em dados categóricos, o que facilita o cálculo de medidas de entropia e informação mútua, além de reduzir a influência de valores extremos. Com a lista de ativações discretizadas, são calculadas, para cada época e para cada camada, duas medidas de informação mútua: uma entre as ativações e os dados de entrada, e outra entre as ativações e os rótulos de saída.

Esse cálculo permite quantificar o quanto cada camada retém informação sobre a entrada e sobre a saída desejada ao longo do treinamento. O resultado é organizado no plano da informação, no qual cada ponto representa a relação entre compressão da informação e previsibilidade em cada camada e época, ao oferecer uma visão detalhada da dinâmica de aprendizado da rede.

Em todos os experimentos, os modelos foram treinados enquanto informações consideradas relevantes eram coletadas, tais como: a arquitetura utilizada, o número de épocas de treinamento, o tamanho do lote, os valores de perda e a precisão ao longo das épocas, além das ativações das camadas internas em cada lote processado.

A coleta sistemática dessas métricas também permitiu acompanhar a evolução do aprendizado ao longo do tempo, identificar o ponto de transição entre as fases de memorização e compressão, e comparar o comportamento de diferentes arquiteturas sob o mesmo critério.

4. Experimentação e resultados

O primeiro modelo testado possui uma arquitetura apresentada no artigo de Schwartz-Ziv, sendo um MLP de 12-10-8-6-4-2-1 neurônios em cada camada utilizando o *dataset* dos pontos uniformemente distribuídos numa esfera 2D, com uma quantidade de 300 épocas de treino, 64 amostras por lote e uma discretização feita utilizando 30 *bins*. Na figura 7 é apresentado o plano da informação obtido como resultado.

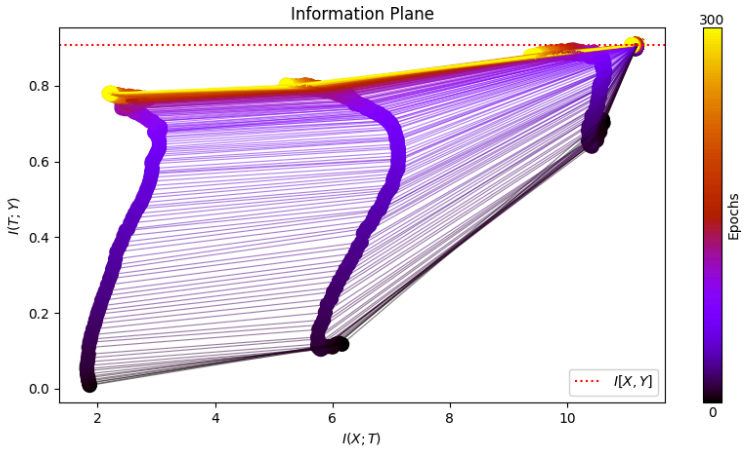

Figura 7. Plano da informação com uma MLP de arquitetura 12-10-8-6-4-2-1.

Com este resultado é possível observar como o modelo captura rapidamente as informações relevantes do conjunto de dados já nas primeiras épocas de treinamento, reforçado pelas colunas crescentes quase completamente representadas na cor roxa.

O valor de *I(X;T)* inicia alto nas camadas mais próximas aos dados de saída, e vai diminuindo lentamente conforme as camadas aprendem a descartar informações irrelevantes. Ao mesmo tempo, *I(T;Y)* aumenta, indicando que as representações intermediárias estão se tornando mais úteis para a tarefa de predição.

Essa dinâmica é compatível com a teoria do o Gargalo da Informação, evidenciando a transição entre as fases de memorização e compressão. Camadas mais profundas apresentam maior compactação da informação de entrada, mas ainda retêm conteúdo essencial para prever a saída, o que reforça a evolução da eficácia do modelo analisado. Essas propriedades também são visualizadas de forma mais sutil no gráfico de acurácia do modelo durante o treino, como mostrado na figura 8.

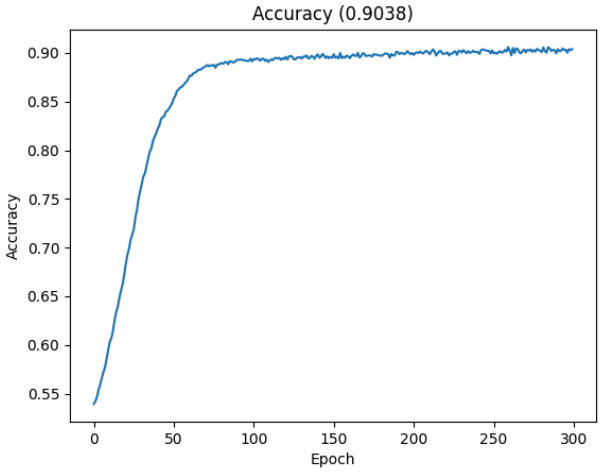

Figura 8. Crescimento da curva de acurácia do modelo ao longo do treino.

É possível observar a fase de memorização no começo do treinamento, até aproximadamente a época 50, pelo fato de a acurácia do modelo crescer drasticamente (evidenciado pela maior parte do aprendizado ter acontecido em menos de 20% do total de épocas do treino) até chegar a um limiar de saturação. Logo após, a partir da época 50, é observada a fase de compressão, onde começam a ser descartadas informações irrelevantes sem comprometer seu desempenho para as predições, com o refino de seus pesos e, como consequência, o aumento da sua acurácia até o limite da arquitetura proposta.

O mesmo ocorre, mas de forma oposta, ao observar o gráfico de perda (ou erro) do modelo durante o treinamento, como mostrado na figura 9.

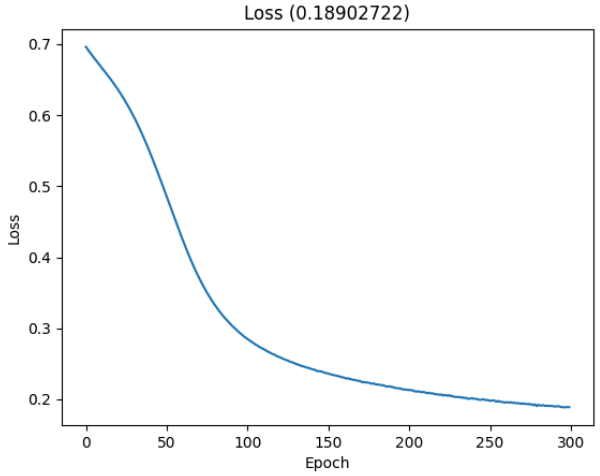

Figura 9. Decaimento da função de perda do modelo ao longo do treino.

É notável como a perda diminui drasticamente até uma certa quantidade de épocas semelhante à ilustrada pelo crescimento da acurácia, agora por volta da época 100, e conforme o treinamento avança, a fase de compressão refina os parâmetros do modelo para melhorar seus acertos sem comprometer o que já foi aprendido anteriormente.

Um segundo experimento foi realizado com dados idênticos ao exemplo anterior, mas com a adição de uma nova camada oculta à arquitetura, configuradas com 12-10-7-5-4-3-2-1 neurônios em cada camada. As demais configurações, tais como: quantidade de épocas, taxa de aprendizado e funções de ativação, foram mantidas as mesmas do experimento anterior

O objetivo dessa adição foi investigar como o aumento na profundidade das camadas do modelo afeta o comportamento do aprendizado no decorrer do treinamento, conforme apresentado na figura 10.

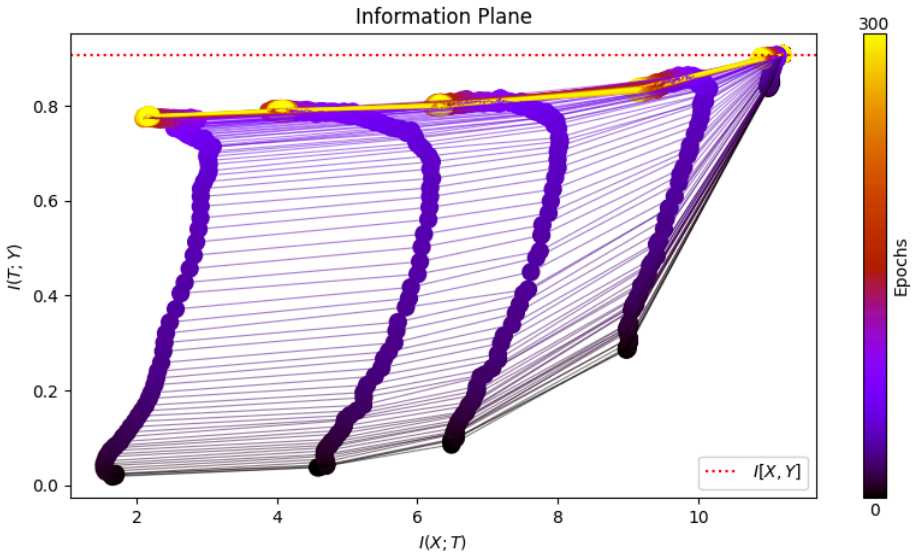

Figura 10. Plano da informação com uma MLP de arquitetura 12-10-7-5-4-3-2-1.

É possível notar que a trajetória das camadas no plano da informação mantém o mesmo padrão geral: os pontos se deslocam inicialmente lentamente para a direita, aumento de *I(X;T)*, o que indica memorização; e rapidamente para cima, aumento de *I(T;Y)*, o que indica aprendizado útil para a tarefa.

No entanto, com a adição da nova camada, observa-se que a compressão da informação ao longo das camadas intermediárias se torna mais tardia, conforme observado na figura 11.

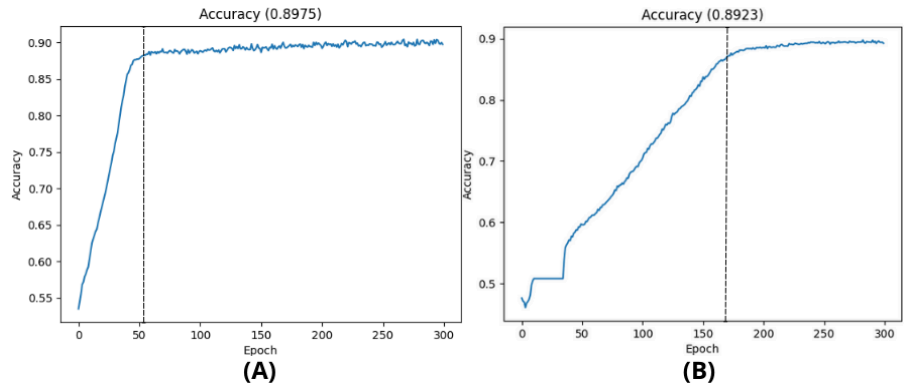

Figura 11. Comparação da acurácia do modelo 12-10-8-6-4-2-1 (A) com o modelo 12-10-7-5-4-3-2-1 (B).

A comparação entre os dois experimentos sugere que a adição de uma camada oculta aumenta a capacidade do modelo de realizar compressão, sem comprometer a performance, ao favorecer a generalização. Entretanto, o primeiro modelo (12-10-8-6-4-2-1) possuía um

total de 313 parâmetros, já o segundo modelo (12-10-7-5-4-3-2-1) contava com 297 parâmetros, uma redução de aproximadamente 6% em comparação ao modelo inicial.

Mesmo com essa pequena redução na quantidade de parâmetros, a adição de uma camada se mostrou ter mais relevância para o retardamento do processo de memorização, ao evidenciar que a complexidade de abstração do modelo está mais ligada à sua profundidade do que a quantidade de neurônios em si.

Na sequência dos experimentos foi utilizado o *dataset* MNIST. Esse conjunto de dados introduz uma complexidade ainda maior em relação ao *dataset* anterior, ao exigir do modelo a capacidade de extrair características úteis a partir de uma representação visual em uma grade de *pixels*.

O modelo mantido para este experimento foi um MLP composto por 784-8-8-8-8-10 neurônios, com um total de 6.586 parâmetros, adaptado para receber entradas vetorizadas a partir das imagens 28x28 (784 *pixels*). Os valores dos *pixels* foram normalizados entre 0 e 1 para facilitar a captura de informações pelo modelo. A arquitetura testada foi ajustada para lidar com essa dimensionalidade mais elevada, e a análise no plano da informação foi realizada da mesma forma: acompanhando *I(X;T)* e *I(T;Y)* ao longo das épocas para cada camada intermediária, evidenciado na figura 12.

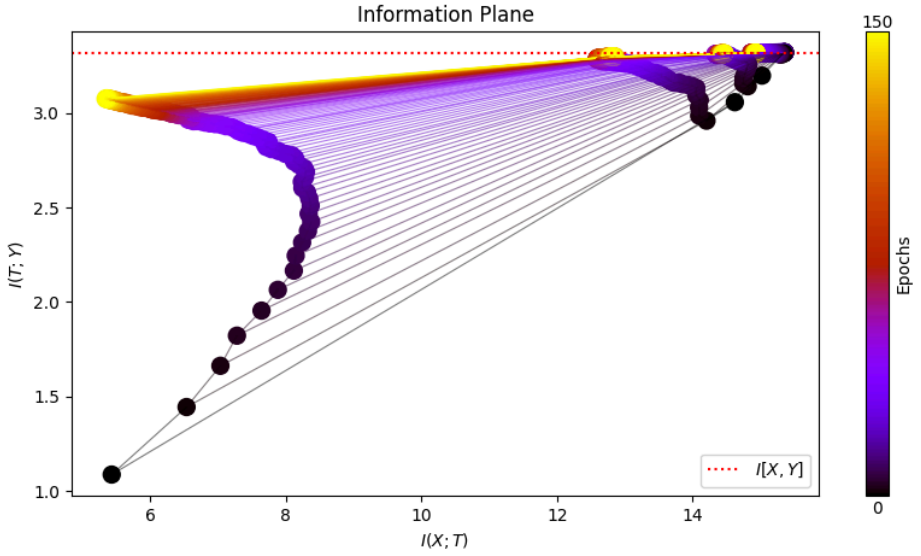

Figura 12. Plano da informação com uma MLP de arquitetura 784-8-8-8-8-10 com o dataset MNIST.

A imagem do plano da informação obtida neste experimento mostra um comportamento muito semelhante ao observado nos testes anteriores, mas com algumas diferenças importantes.

Nota-se que as curvas se iniciam em valores mais altos e apresentam menor variação a partir da segunda camada, ao indicar que a informação da entrada é, em grande parte, trabalhada na camada inicial, com uma compressão mais gradual nas camadas seguintes.

O valor de *I(T;Y)* cresce rapidamente nas primeiras épocas e se estabiliza próximo ao valor de *I(X;Y)*, representado pela linha pontilhada vermelha, ao sinalizar que o modelo conseguiu aprender representações eficazes para a tarefa de classificação. A curva mais suave sugere uma transição menos abrupta entre as fases de memorização e compressão, característica comum em modelos treinados com dados mais ricos e complexos.

Esse resultado reforça a hipótese de que, mesmo em problemas mais desafiadores, a dinâmica do Gargalo da Informação permanece válida, com o ajuste do modelo em suas representações internas de forma progressiva, ao compactar a informação da entrada enquanto retém o que é relevante para a saída.

De forma similar ao realizado no teste com o *dataset* anterior, o segundo experimento com o *dataset* MNIST utilizou uma arquitetura com duas camadas ocultas adicionais, sendo 784-8-8-8-8-8-8-10 neurônios e com 6.730 parâmetros, sem alteração nas funções de ativação nem na quantidade de épocas, com alteração apenas na profundidade do modelo.

Essa modificação teve um impacto direto na dinâmica de aprendizado observada no plano da informação, destacado na figura 13.

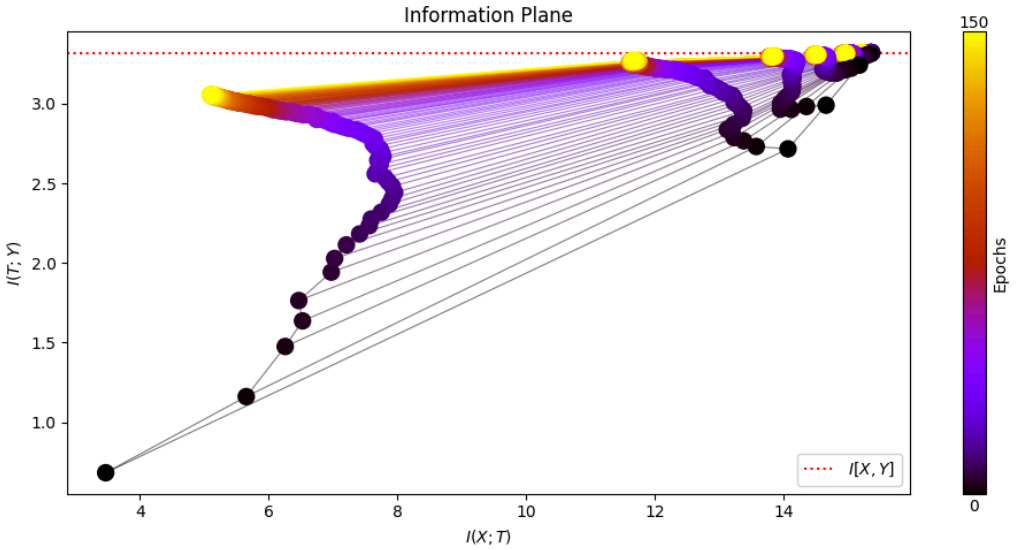

Figura 13. Plano da informação com uma MLP de arquitetura 784-8-8-8-8-8-8-10 com o dataset MNIST.

O resultado obtido para este segundo experimento revela curvas de *I(X;T)* ainda mais espalhadas comparadas ao exemplo da figura 12, com maior variação entre as camadas. As camadas iniciais ainda preservam grande parte da informação da entrada, enquanto as camadas mais profundas realizam uma compressão menos agressiva.

Apesar disso, o valor de *I(T;Y)* mantém-se elevado ao longo do treinamento, ao aproximar-se novamente do limite de *I(X;Y)*, o que indica que o modelo continua a aprender representações eficazes para a tarefa de classificação.

Comparado ao modelo anterior, com menos camadas, o novo modelo apresenta uma dinâmica mais complexa, com curvas mais fragmentadas e uma transição menos uniforme entre memorização e compressão. Essa complexidade pode estar relacionada ao maior número de parâmetros e ao aumento da profundidade, que exige um tempo maior de ajuste para estabilizar as representações internas. Ainda assim, o comportamento geral permanece coerente com o esperado.

Esse experimento demonstra que a adição de camadas aumenta a flexibilidade da rede, ao permitir uma representação mais rica das informações de entrada. No entanto, isso também mostra como o processo de treinamento fica mais sensível e sujeito a variações no comportamento das camadas intermediárias.

O impacto da adição de mais camadas pode ser melhor visualizado na figura 14, onde é possível acompanhar a trajetória do plano da informação em quatro cenários: um modelo com arquitetura 784-8-8-8-8-10 (base), arquitetura 784-8-8-8-8-8-8-10 (duas camadas adicionais), arquitetura 784-8-8-8-8-8-8-8-8-10 (quatro camadas adicionais) e arquitetura 784-8-8-8-8-8-8-8-8-8-8-10 (seis camadas adicionais).

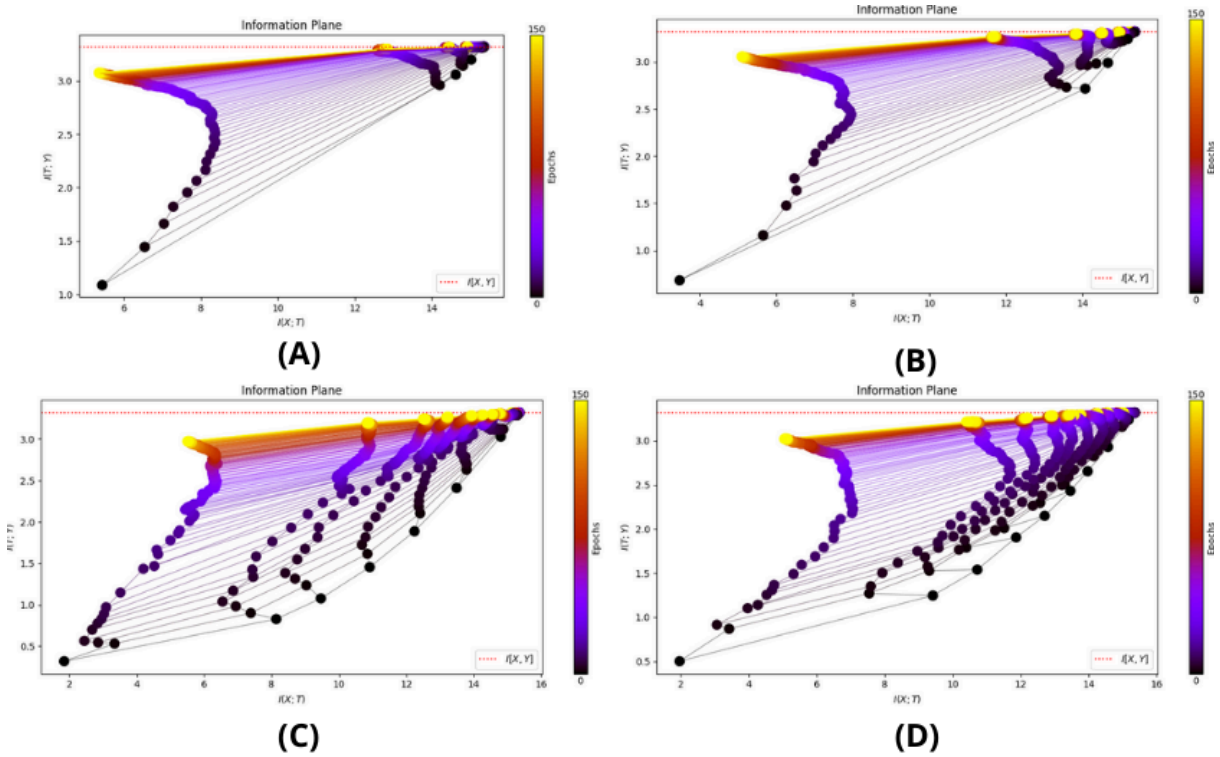

Figura 14. Plano da informação com modelos MLP utilizando arquitetura 784-8-8-8-8-10 (A), 784-8-8-8-8-8-8-10 (B), 784-8-8-8-8-8-8-8-8-10 (C) e 784-8-8-8-8-8-8-8-8-8-8-10 (D), junto ao *dataset* MNIST.

Essa nova comparação agora evidencia a tendência da fase de compressão ir desaparecendo à medida que o modelo se torna mais complexo. Com mais parâmetros para ajustar, o modelo passa a maior parte do treinamento capturando informações, e a medida que

sua profundidade cresce, o comportamento de *overfitting* se manifesta, visto o modelo começa a se tornar competente para a tarefa (aumento de *I(T;Y)*) mas também memoriza muitos dados de entrada (aumento de *I(X;T)*);

A comparação entre os quatro modelos reforça que há um equilíbrio a ser buscado entre profundidade e eficiência, e que arquiteturas mais profundas, embora potencialmente mais poderosas, exigem maior cuidado na regularização e no monitoramento do aprendizado.

Na sequência foi realizado um teste utilizando o *dataset* sintético da esfera no plano 2D, com uma única alteração, a função de ativação das camadas ocultas que antes era a Tangente Hiperbólica foi substituída pela ReLU.

Essa simples alteração de comportamento muda completamente o funcionamento interno do modelo. Mesmo que as quantidades de neurônios e de camadas sejam idênticas, o seu funcionamento interno, desde a forma como a saída é calculada e, principalmente, como os gradientes são repassados na fase de treinamento, são totalmente diferentes, podendo ser visualizado na figura 15.

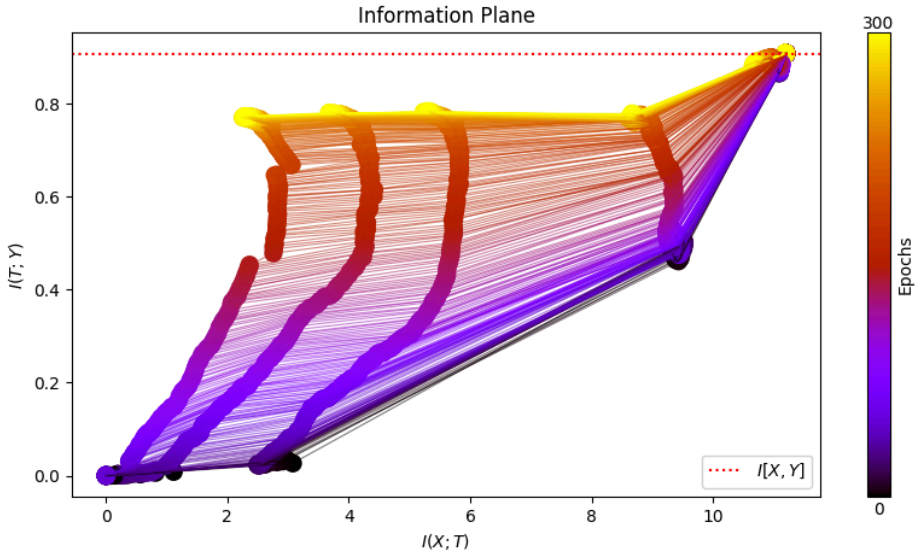

Figura 15. Plano da informação com dataset sintético, arquitetura 12-10-7-5-4-3-2-1 e função de ativação ReLU.

A sutil mudança na função de ativação trouxe uma nova percepção visual para contribuir com novas discussões.

É correto afirmar que agora o modelo teve mais dificuldade em absorver informação dos dados de treino, fato que é evidenciado pela coloração dos pontos no plano da informação, que agora são visivelmente mais avermelhados, além de observar uma menor tendência na fase de compressão, indicando que o modelo acumulou menos informação em relação aos dados de entrada por cada camada. Esse comportamento fica mais evidente quando é observado o gráfico de acurácia desse mesmo modelo nos diferentes cenários (figura 16), reforçando o atraso ocorrido durante o aprendizado.

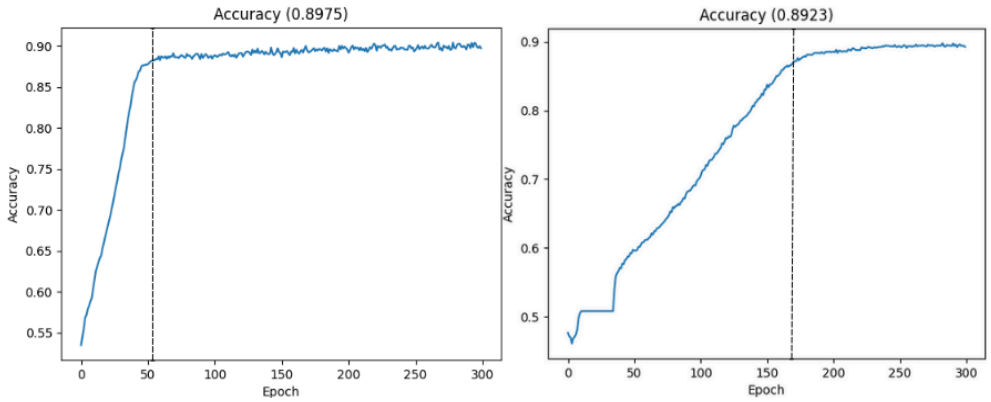

Figura 16. Comparação da curva de acurácia do modelo 12-10-7-5-4-3-2-1 com a função de ativação Tangente Hiperbólica (esquerda) e ReLU (direita).

Com os resultados dos novos testes, tem-se uma melhor percepção do impacto da alteração feita, pois a função ReLU se mostrou mais ineficiente para captura dos dados de treino na tarefa dada, pois demora mais épocas para ter o mesmo desempenho de aprendizado que a Tangente Hiperbólica proporcionou ao mesmo modelo inicialmente.

Algo curioso é que o comportamento das fases de memorização e compressão ainda se mostraram presentes, mesmo com uma curva menos angular, ainda, é possível observar o comportamento "natural" de absorção máxima dos dados para generalização e refinamento dos parâmetros do modelo para maior desempenho nas predições.

Na sequência dos experimentos, foi utilizado um modelo Convolucional junto ao conjunto de dados MNIST. Entretanto, devido a natureza dos experimentos em que é necessário capturar cada ativação, de cada camada e de cada época de treinamento, em modelos MLP esse uso não era necessariamente um problema pois existem poucas ativações nesses modelos.

Já em modelos com camadas Convolucionais, tem-se uma saída tridimensional (uma lista de imagens 2D equivalente às predições de cada filtro da camada), nesse cenário foi necessário reduzir a quantidade de filtros das camadas convolucionais além de aumentar o tamanho do filtro de *pooling* das camadas subsequentes.

Outros parâmetros também precisaram ser alterados, como a quantidade de épocas de treinamento para reduzir o tamanho das capturas das ativações; e, uma pequena otimização se fez necessária: por padrão os modelos utilizam um formato *float32* para os dados (com alocação de 32 bits de memória), e, assim, foi necessário converter as ativações capturadas para *float16*, que utiliza metade do espaço de memória do formato anterior.

Para efeito de comparação, o modelo utilizado no primeiro teste com MNIST com a maior quantidade de camadas (784-8-8-8-8-8-8-10) possui apenas 58 ativações para serem capturadas, já a CNN utilizada, mesmo com as reduções de dimensionalidade citadas, conta

com 6.462 ativações para serem capturadas, todas em sua maior parte concentradas na primeira camada convolucional.

Com todas estas alterações realizadas, a arquitetura no experimento seguinte é composta por duas camadas convolucionais, cada uma com 8 filtros de tamanho 3x3 (Altura x Largura) seguidas por camadas de pooling para redução de dimensionalidade dos dados calculados, acompanhadas de uma camada de achatamento e, por fim, quatro camadas densas (equivalentes a um modelo MLP). A função de ativação nas camadas convolucionais usada foi a ReLU, nas camadas Densas (exceto a última) foi utilizada a função Tangente Hiperbólica e na camada de saída foi utilizada a função *Softmax*.

No total, este modelo conta com 5.454 parâmetros, uma redução de aproximadamente 19% em relação ao modelo mais complexo utilizado neste mesmo *dataset*. Os resultados podem ser visualizados nas figuras 17, 18 e 19, respectivamente.

```
model =  Sequential([
    Input(input_shape),
    Conv2D(8, (3, 3), activation = 'relu'),
    MaxPooling2D((3, 3)),
    Conv2D(8, (3, 3), activation = 'relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(40, activation = "tanh"),
    Dense(20, activation = "tanh"),
    Dense(20, activation = "tanh"),
    Dense(20, activation = "tanh"),
    Dense(10, activation = "softmax"),
])
```

Figura 17. Arquitetura da CNN utilizada junto com o dataset MNIST.

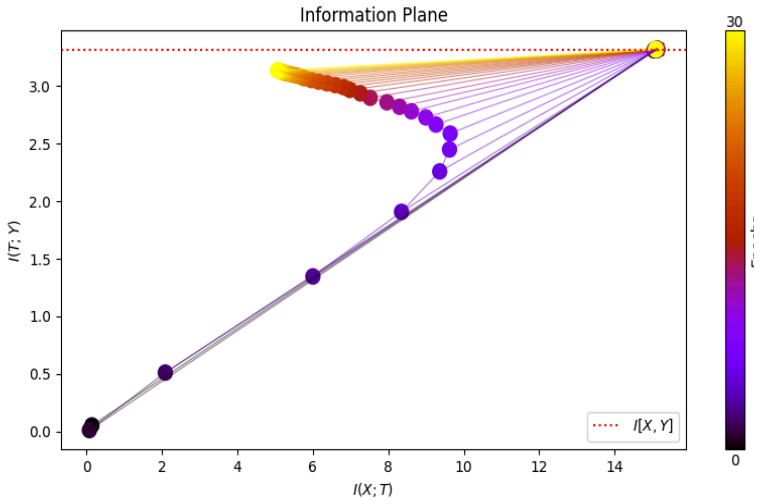

Figura 18. Plano da Informação da CNN com o dataset MNIST.

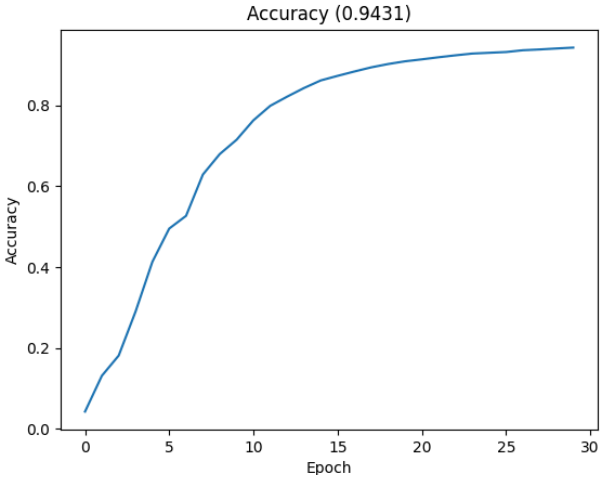

Figura 19. Curva de acurácia da CNN com o dataset MNIST.

Com estes resultados é possível concluir mais uma vez que o comportamento das fases de memorização e compressão se mantém até mesmo em camadas Convolucionais, destacando a forma como a informação evolui nelas.

Desta vez, observa-se uma grande evolução no valor de *I(X;T)* maior do que nos testes anteriores com as MLP, indicando a maior facilidade das camadas convolucionais em capturar informações dos dados de entrada dos *datasets*.

Entretanto os demais pontos das camadas seguintes estão todos saturados (acima e à direita), possivelmente pela natureza do modelo em possuir poucas camadas somado aos poucos parâmetros presentes, tanto nas camadas Convolucionais, quanto nas camadas Densas, o que demonstra uma maior evolução do aprendizado ocorrido apenas na primeira camada.

5. Considerações finais

Este estudo teve como objetivo analisar o processo de aprendizado em Redes Neurais Artificiais sob a perspectiva da Teoria da Informação, utilizando o princípio do Gargalo da Informação como ferramenta de interpretação.

Através da quantificação da informação mútua entre as ativações internas e os dados de entrada e saída, foi possível observar visualmente as fases de memorização e compressão ao longo do treinamento dos modelos ao utilizar abordagens baseadas em entropia e informação mútua.

Os experimentos realizados com os modelos MLPs e CNNs, independente do *dataset* utilizado, evidenciaram a dinâmica teórica proposta por [Schwartz-Ziv 2017] ao mostrar que, após uma fase inicial de acumulação de informação, ocorre uma compressão controlada nas camadas internas, o que favorece a generalização e o refinamento dos parâmetros do modelo. Essa observação reforça a hipótese de que a compactação da informação irrelevante é uma etapa importante para o bom desempenho no ajuste fino dos parâmetros dos modelos.

A aplicação prática do plano da informação mostrou-se eficaz para visualizar a evolução das representações internas, ao oferecer um diagnóstico interpretável que auxilia na compreensão do comportamento das camadas e na identificação de possíveis redundâncias e ineficiências na arquitetura, e por colaborar principalmente na redução do tempo de treinamento e na otimização das arquiteturas.

Apesar dos resultados consistentes, este estudo apresentou limitações fortes em modelos convolucionais, mesmo de pequeno porte, pois o alto uso de memória para capturar todas as ativações 3D produzidas pelas camadas mostrou ser uma grande barreira, o que indica a necessidade de abordagens mais eficientes e otimizadas para modelos maiores ou com arquiteturas mais complexas.

Além da contribuição teórica, este trabalho também oferece uma implementação prática capaz de capturar e visualizar as ativações internas de diferentes arquiteturas, o que pode ser útil como ferramenta de apoio a pesquisas futuras para interpretabilidade de Redes Neurais Artificiais.

Como ideias futuras para expansão deste trabalho, sugere-se a extensão da análise a CNNs mais complexas e Redes Neurais Recorrentes (RNNs), implementação de técnicas para redução e otimização do uso de memória durante a captura das ativações, e o estudo da influência de diferentes funções de perda e otimizadores na dinâmica de compressão e memorização, além de explorar as diversas funções de ativação que as bibliotecas modernas oferecem. Estes esforços podem contribuir para um entendimento mais abrangente do aprendizado profundo e para o desenvolvimento de modelos mais eficientes e interpretáveis.

Referências

Chelombiev, I.; Houghton, C.; O'Donnell C. (2019) Adaptive Estimators Show Information Compression in Deep Neural Networks. arXiv preprint arXiv:1902.09037.

Doshi-Velez, F.; Kim, B. (2017) Towards A Rigorous Science of Interpretable Machine Learning. arXiv preprint arXiv:1702.08608.

Goodfellow, I.; Bengio, Y; Courville, A. (2016) Deep Learning. Cambridge, MA: MIT Press. ISBN 9780262337373.

He, K.; Zhang, X.; Ren, S.; Sun, J. (2016) Deep Residual Learning for Image Recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778. https://doi.org/10.1109/CVPR.2016.90.

Kawaguchi, K.; Deng, Z.; Ji, X.; Huang, J. (2023) How Does Information Bottleneck Help Deep Learning? In: International conference on machine learning. PMLR, p.16049-16096.

Krizhevsky, A.; Sutskever, I.; Hinton, G. (2012) ImageNet Classification with Deep Convolutional Neural Networks. Advances in Neural Information Processing Systems (NeurIPS), pp. 1097–1105. https://doi.org/10.1145/3065386.

LeCun, Y.; Bengio, Y. (1995) Convolutional Networks for Images, Speech, and Time-Series. The Handbook of Brain Theory and Neural Networks. MIT Press, pp. 255–258.

Lipton, Z. (2018) The Mythos of Model Interpretability: In machine learning, the concept of interpretability is both important and slippery. Queue, v. 16, n. 3, p. 31-57.

Liu, Z.; Li, J.; Shen, Z.; Huang, G.; Yan, S.; Zhang, C. (2017) Learning Efficient Convolutional Networks through Network Slimming. Proceedings of the IEEE International Conference on Computer Vision (ICCV), pp. 2736–2744. https://doi.org/10.1109/ICCV.2017.298.

Nasteski, V. (2017) An Overview of the Supervised Machine Learning Methods. Horizons. b, v. 4, n. 51-62, p. 56.

O'Shea, K.; Nash, R. (2015) An Introduction to Convolutional Neural Networks. arXiv preprint arxiv:1511.08458.

Popescu, M.; Balas, V.; Popescu-Popescu, L.; Mastorakis, N. (2009) Multilayer Perceptron and Neural Networks. WSEAS Transactions on Circuits and Systems, v. 8, n. 7, p. 579-588.

Rather, I.; Kumar, S.; Gandomi, A. (2024) Breaking the data barrier: a review of deep learning techniques for democratizing AI with small datasets. Artificial Intelligence Review, v. 57, n. 9, p. 226.

Rosenblatt, F. (1958) The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. Psychological Review, 65(6), 386–408. https://doi.org/10.1037/h0042519.

Rudin, C. (2019) Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead. Nat Mach Intell 1, 206–215. https://doi.org/10.1038/s42256-019-0048-x.

Rumelhart, D.; Geoffrey, E.; Hinton, G.; Williams, R. (1986) Learning representations by back-propagating errors. Nature 323, 533–536. https://doi.org/10.1038/323533a0.

Schwartz-Ziv, R.; Tishby, N. (2017) Opening the Black Box of Deep Neural Networks via Information. arXiv preprint arxiv:1703.00810.

Sengupta, S.; Basak, S.; Saikia, P.; Paul S.; Tsalavoutis, V.; Atiah, F.; Ravi, V.; Peters, R. (2020) A Review of Deep Learning with Special Emphasis on Architectures, Applications and Recent Trends. Knowledge-Based Systems, v. 194, p. 105596.

Simonyan, K.; Zisserman, A. (2014) Very Deep Convolutional Networks for Large-Scale Image Recognition. ArXiv preprint arXiv:1409.1556, 2014.

Shannon, C. (1948) A Mathematical Theory of Communication. Reprinted with corrections from The Bell System Technical Journal, Vol. 27, pp. 379–423, 623–656, July, October.

Tishby, N.; Pereira, F. C.; Bialek, W. (1999) The Information Bottleneck Method. in *Proc. Allerton Conf. Commun. Control Comput.*, Monticello, IL, USA, pp. 368–377.

Tishby, N.; Zaslavsky, N. (2015) Deep learning and the Information Bottleneck Principle. *IEEE Information Theory Workshop (ITW)*, Jerusalem, Israel, pp. 1-5. https://doi.org/10.1109/ITW.2015.7133169.

Zhang, Q.; Zhu, S. (2018) Visual Interpretability for Deep Learning: a survey. *Frontiers Inf Technol Electronic Eng* 19, 27–39. https://doi.org/10.1631/FITEE.1700808.