# Vision Transformers: Um estudo da arquitetura Transformer aplicada à Classificação de Imagens

**Rodrigo Zamboni Silva**


Universidade Federal de Uberlândia
Faculdade de Computação
Programa de Graduação em Ciência da Computação

Uberlândia
2014

**Rodrigo Zamboni Silva**

# Vision Transformers: Um estudo da arquitetura Transformer aplicada à Classificação de Imagens

Trabalho de Conclusão de Curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia como parte dos requisitos para a obtenção do grau de Bacharel em Ciência da Computação.

Área de concentração: Ciência da Computação

Orientador: Prof[a] Dr[a] Rita Maria da Silva Julia

Uberlândia
2014

*Dedico este trabalho à minha família, amigos e a todos que, das formas mais simples e genuínas, acreditaram em mim.*

# Agradecimentos

Agradeço primeiramente a meus pais e minha família, pelo carinho e apoio, ao meu irmão Ricardo, por ser meu absoluto parceiro no dia a dia.

Agradeço aos amigos que ganhei durante minha graduação, em especial aos Paragons, a Lorena 'Lola' Elias e Gabriel Otsuka. Agradeço também aos meus companheiros de ritmo, da Computaria e UFUteria.

Por fim agradeço a cada professor já tive nesses anos de graduação. Em especial a minha orientadora, e minha amiga, Rita Maria da Silva Julia, por todo o carinho e dedicação a mim.

*"A persistência é o caminho do êxito."*
*(Charles Chaplin)*

# Resumo

O presente trabalho tem por objetivo principal estudar e analisar, por meio de experimentos, o funcionamento do modelo de Rede Neural Vision Transformer no processamento e classificação de imagens. Para tanto, para fins comparativos, foram estudados vários outros tipos de modelos de redes neurais que vêm sendo usados com bastante sucesso, tais como: Perceptron Múltiplas Camadas, Redes Convolucionais e as Redes Recorrentes. É importante ressaltar que o primeiro e revolucionário modelo proposto de Rede Transformer foi concebido com a finalidade de lidar com o processamento da linguagem natural. O êxito desse primeiro modelo se deve, essencialmente, ao uso da técnica de *Self-Attention* com o propósito de construir relações entre palavras de uma frase. Em virtude disso, este trabalho também inclui um estudo teórico desse primeiro modelo. Tal inclusão é bastante pertinente, uma vez que a Rede Vision Transformer, a qual é foco principal do presente trabalho, foi produzida, posteriormente, com base nesse primeiro modelo. Isso significa que a Vision Transformer corresponde a uma adaptação desse primeiro modelo ao domínio do processamento de imagens, baseando-se, também na técnica de *Self-Attention*. O modelo Vision Transformer aqui proposto foi implementado a partir do modelo disponível na biblioteca Keras, usando, também, a mesma base de dados. Convém salientar que tal modelo Keras reproduz o modelo original da Vision Transformer. Contudo, a título investigativo, o modelo Keras adiciona ao modelo original a técnica *Data Augmentation*.

**Palavras-chave:** Transformer. Vision Transformer. Aprendizado de Máquina. Redes Neurais Artificiais. Percéptron Múltiplas Camadas. Redes Convolucionais. Redes Recorrentes. Selt-Attention..

# Abstract

The main objective of this work is to study and analyze, through experiments, the functioning of the Vision Transformer Neural Network model in image processing and classification. For comparative purposes, several other types of neural network models that have been successfully used were studied, such as: Multi-Layer Perceptrons, Convolutional Networks, and Recurrent Networks. It is important to highlight that the first and revolutionary Transformer model was conceived to handle natural language processing. The success of this first model is essentially due to the use of the Self-Attention technique to establish relationships between words in a sentence. Consequently, this work also includes a theoretical study of this first model. This inclusion is quite pertinent, given that the Vision Transformer, which is the main focus of this work, was subsequently produced based on this first model. This means that the Vision Transformer corresponds to an adaptation of this first model to the domain of image processing, also based on the Self-Attention technique. The Vision Transformer model proposed here was implemented from the model available in the Keras library, also using the same database. It is worth noting that this Keras model reproduces the original Vision Transformer model. However, for investigative purposes, the Keras model adds the Data Augmentation technique to the original model.

**Keywords:** Transformer. Vision Transformer. Machine Learning. Artificial Neural Networks. Multi-layer Perceptrons. Convolutional Networks. Reccurent Networks. Self-Attention..

# Lista de ilustrações

# Lista de tabelas

# Lista de siglas

**ABNT** Associação Brasileira de Normas Técnicas

**AM** Aprendizado de Máquina

**AP** Aprendizado Profundo

**CNN** Convolutional Neural Network

**DP** Deep Learning

**IA** Inteligência Artificial

**ML** Machine Learning

**MLP** Multi Layer Perceptron

**NBR** Denominação de norma da Associação Brasileira de Normas Técnicas

**PI** Processamento de Imagem

**PLN** Processamento de Linguagem Natural

**PMC** Percéptron Múltiplas Camadas

**RNA** Redes Neurais Artificiais

**RNC** Redes Neurais Convolucionais

**RNR** Redes Neurais Recorrentes

**RNN** Recurrent Neural Network

**ViT** Vision Transformer

**VC** Visão Computacional

# Sumário

# Capítulo 1

# Introdução

A tecnologia está presente em todos os lugares do mundo, do carro elétrico ao aparelho celular, todos estão em constante aprimoramento ao passo da evolução. A tecnologia, cada vez mais, auxilia o ser humano em tarefas do seu cotidiano. Seja para transportar uma pessoa de um lugar a outro, realizar pagamentos em bancos digitais ou simplesmente trocar mensagens de texto com outras pessoas, a tecnologia se faz fundamental para garantir a praticidade e eficácia dessas ações.

Nos anos mais recentes, uma tecnologia vem revolucionando as formas, até então triviais, de se lidar com problemas complexos. O Aprendizado de Máquina (AM), um dos ramos da Inteligência Artificial (IA), faz com que a máquina seja capaz de aprender com informações a ela apresentadas, sendo capaz de produzir conhecimento sobre determinado assunto e então aplicá-lo a problemas relacionados àquele assunto. Outra característica fundamental do AM reside em sua independência e aprimoramento contínuo, ou seja, a máquina é capaz de evoluir, de forma a aprimorar seus conhecimentos e estar mais apta a solução do problema, sem que haja a necessidade da atuação de um agente humano.

Por meio do AM é possível criar agentes inteligentes autônomos (NORVIG; RUSSELL, 2014) capazes de realizar tarefas do mundo moderno (LECUN et al., 1998). Problemas que até então dependem exclusivamente do ser humano para serem resolvidos, passam a ser solucionados por agentes preditivos (VINCENT et al., 2019) (GOLDEN, 2017), que por sua vez vão tomar decisões baseadas em dados coletados ou de experiências vivenciadas, sem uma devida programação prévia para um determinado resultado. Essa abordagem, denominada Aprendizado Profundo (AP) (LECUN; BENGIO; HINTON, 2015), compreende a existência de Redes Neurais, também chamadas de Redes Neurais Artificiais, como sendo as principais ferramentas para resolução de problemas de alta complexidade. Exemplos disso são agentes inteligentes capazes de performar de forma semelhante, em alguns casos até de forma superior, a agentes humanos (SILVER et al., 2017) (NETO; JULIA, 2016) (BERNER et al., 2019).

A evolução das Redes Neurais permitiu uma ramificação enorme de modelos, cada um com características e objetivos diferentes, de forma a criar uma hierarquia de modelos

cada vez mais sofisticados. Dessa forma, o modelo mais ingênuo de uma Rede Neural, o chamado de Perceptron Múltiplas Camadas, foi precursor e eventualmente cedeu espaço a outras arquiteturas mais robustas, como por exemplo Redes Recorrentes e Redes Convolucionais, pois essas se tornaram superiores em inúmeras tarefas propostas às redes neurais.

No topo da hierarquia de evolução de Redes Neurais, estão os modelos chamados Transformers (VASWANI et al., 2017). São eles os principais responsáveis pelas Inteligências Artificiais Generativas do mundo atual, isto é, modelos capazes de codificar uma entrada e produzir uma resposta acertiva no mesmo formato. Como exemplo desses modelos generativos inteligentes baseado em Transformers, destacam-se: 1) os sistemas capazes de produzir texto, seja para uma conversa ou demanda específica, com base em uma dada entrada (requisição) apresentada pelo usuário: ChatGPT (OPENAI, 2024a), Llama (META, 2024), GEMINI (GOOGLE, 2024), etc; 2) sistemas capazes de gerar imagens relativas a um dado contexto requerido pelo usuário: DALL-E2 (OPENAI, 2024b), Deep Dream (MORDVINTSEV, 2024);

Esse modelo se mostrou muito flexível, capaz de assumir tanto um propósito generativo quanto de classificação, ou seja, o modelo Transformer também pode ser utilizado em tarefas de processamento e análise de imagens, como por exemplo reconhecimento facial ou classificação de imagens (DOSOVITSKIY et al., 2020).

Em suma, a Rede Neural Transformer é composta de duas partes: uma, chamada Encoder, responsável por assimiliar e codificar o dado de entrada, repassando esse produto codificado à segunda parte, chamada Decoder, a fim de que ela produza o resultado esperado pelo usuário. Dessa forma, o modelo Transformer que lida com Processamento de Linguagem Natural, que precisa tanto interpretar quanto produzir texto, será composto pelas 2 partes, Encoder e Decoder. Já o modelo Transformer que lida com Processamento de Imagem, por precisar apenas interpretar o dado à ele fornecido, conta apenas com a parte Encoder.

Nesse contexto, o presente trabalho tem como objetivo estudar a evolução dos principais modelos de Redes Neurais, com enfoque particular na Transformer. Para tanto, será implementada, treinada e reproduzida aqui a rede Transformer processadora de imagens acima citada (DOSOVITSKIY et al., 2020).

O atual trabalho foi estruturado de forma a conter 4 Capítulos fundamentais, sendo eles: *Introdução* (Capítulo 1), *Fundamentação Teórica* (Capítulo 2), *Modelo Desenvolvido* (Capítulo 3) e por fim *Experimentos e Resultados* (Capítulo 4). Cada um dos Capítulos estão subdivididos em seções e subseções, organizadas de forma a proporcionar uma leitura intuitiva e de fácil compreensão do trabalho desenvolvido.

## 1.1 Justificativa

Redes Neurais Artificiais, em especial o modelo Transformer, se tornaram ferramentas fundamentais na constituição do atual cenário de Inteligência Artificial. Sua capacidade de abstrair padrões e construir relações entre diferentes dados provocaram significativos avanços nos campos de Processamento de Linguagem Natural (PLN), Visão Computacional (VC) e vários outros domínios da IA.

A tarefa de classificação de imagens pode ter diferentes propósitos. Por exemplo, no âmbito de jogos digitais, pode ser aplicada como ferramenta de detecção de movimentos, ações ou até mesmo eventos do jogo. Isso permitirá,por exemplo, que seja feita uma análise completa a respeito do comportamento do jogador, ou seja, de como é seu processo de tomada de decisão nas variadas situações de jogo.

O modelo Transformer, em particular, foi revolucionário justamente por introduzir o conceito de Atenção, que permitiu com que esse modelo atribuísse valores de relevância distintos para diferentes partes de uma entrada, a fim de elaborar uma predição ou qualquer outro tipo de saída. Diferente de outros modelos, a abordagem utilizada pela Transformer permite o processamento de entradas de variados tamanhos, mantendo sua performance para qualquer que seja o tamanho do dado a ser processado. Essa característica foi fundamental na evolução de tarefas relacionadas a Processamento de Linguagem Natural, tal como tradução de texto (LAKEW; CETTOLO; FEDERICO, 2018), análise sentimental (KOKAB; ASGHAR; NAZ, 2022) e principalmente processamento de imagem (LU et al., 2022) (CHEN et al., 2021).

Além disso, a arquitetura do modelo Transformer permite o processamento paralelizado dos dados de entrada, o que otimiza em grande escala desde o processo de treinamento da rede até a execução do modelo e elaboração de uma saída. O modelo Transformer se mostra adaptável a uma alta variedade de problemas, que ligados ao seu poder de escalonamento, estendem cada vez mais os limites da Inteligência Artificial por quebrar paradigmas até então estabelecidos por modelos de Redes Neurais não tão sofisticados.

## 1.2 Objetivo Geral

O presente trabalho se encaixa no contexto das atividades conduzidas pelo grupo de pesquisa BladeRunner - coordenado por docentes da FACOM e que conta com pesquisadores parceiros de várias instituições de ensino superior nacionais e, outra, interancional - o qual tem por objetivo principal desenvolver sistemas jogadores automáticos inteligentes adaptativos para a Educação e as Ciências da Saúde. Dentre os primeiros resultados obtidos pelo referido grupo, destaca-se o uso da AM baseada em redes neurais profundas (no caso, Redes Convolucionais e Redes Recorrentes) para efetuar tarefa de classificação com rótulo simples (FARIA et al., 2022b) (FARIA et al., 2022a) e com múltiplos rótu-

los (JULIA, 2022). Mais especificamente, os rótulos a que se referem tal classificação correspondem a eventos que ocorrem em vídeos gravados do jogo do Super Mario.

Como o processamento de imagens é muito importante no domínio dos jogos digitais, a fim de tentar contribuir com essas pesquisas conduzidas pelo grupo BladeRunner, o presente trabalho tem por objetivo geral estudar a evolução das redes neurais que mais vêm se destacando no estado da arte relacionado à AM (Percéptron Múltiplas Camadas, Redes Convolucionais, Redes Recorrentes e Transformer), bem como estudar em mais detalhes a rede Transformer processadora de imagens. Em função do destaque da Transformer no contexto atual, pretende-se tambem aqui reproduzir a rede Transformer processadora de imagens proposta em (DOSOVITSKIY et al., 2020), a qual é treinada para efetuar classificação de rótulo simples na Base de Dados CIFAR-100 (KRIZHEVSKY, 2009). A ideia é que, em trabalhos futuros, as redes profundas até então usadas pelo grupo (Convolucionais e Recorrentes) sejam substituídas pela rede Transformer.

## 1.3 Objetivo Específicos

Dentre os objetivos específicos planejados para este trabalho, busca-se realizar um estudo aprofundado dos principais modelos de Redes Neurais contemplados no estado da arte de AM, seguindo a ordem evolutiva desses modelos. Sendo assim, o estudo parte dos conceitos fundamentais acerca de Redes Neurais Artificiais, até o concebimento do modelo mais ingênuo de rede neural, chamado Percéptron Múltiplas Camadas (PMC).

Em seguida serão abordados os modelos de Redes Neurais Convolucionais, de forma a detalhar suas características e, principalmente, suas diferenças em relação a um PMC, para assim evidênciar o salto evolutivo e técnico dentre esses modelos. O terceiro objeto de estudo serão as Redes Neurais Recorrentes, a fim de, detalhar a nova abordagem proposta por esse modelo, suas aplicações e seus resultados.

Por fim, com o propósito de ser o objeto de estudo principal do atual trabalho, será feita uma pesquisa detalhada a respeito do modelo de redes neurais chamado Transformer, de modo a evidenciar seu funcionamento e os principais componentes de sua arquitetura. Será abordado tanto seu modelo original, proposto para lidar com Processamento de Linguagem Natural, quando sua versão que tem como área de aplicação o âmbito da Visão Computacional, chamado de Vision Transformer.

Com isso, o atual trabalho estende seus objetivos a fim de realizar a implementação prática do modelo Vision Transformer, proposto em (DOSOVITSKIY et al., 2020), definindo seus parâmetros de configuração, possibilitando assim um processo de treinamento intensivo dessa rede. Por fim, os resultados obtidos serão avaliados e concilia-dos conforme o objetivo geral da pesquisa.

# Capítulo 2

# Fundamentação Teórica

Nessa seção são apresentados os principais conceitos teóricos essenciais para o desenvolvimento deste trabalho.

## 2.1 Redes Neurais Artificiais

Uma Rede Neural Artificial (RNA) é uma estrutura computacional que simula a estrutura biológica do cérebro humano, é composta de diversas camadas interligadas de neurônios, a fim de desenvolver um conhecimento elaborado baseado naquilo que lhes foi dado de entrada. Os neurônios são os elementos mais básicos do modelo, são responsáveis por receber e propagar informações, na forma de pulsos elétricos, por meio de conexões. Se tratando de RNAs, essas conexões, que acontecem entre dois neurônios, possuem um peso que altera o valor do pulso enviado pelo neurônio emissor até o neurônio receptor. É dessa forma que o cérebro humano executa o processo de raciocínio, recebendo novas informações e processando-as para então consolidar o conhecimento produzido, que por sua vez terá influência direta na tomada de decisão final do indivíduo.

Um recém nascido não possui a capacidade cognitiva aguçada ao passo de ser concebido ao mundo, ele precisa desenvolver tais habilidades cognitivas durante o decorrer de sua vida, dispondo de diversas técnicas de aprendizado que vão estimular seu desenvolvimento cognitivo nas diversas atividades que um ser humano possa vir a executar. De forma análoga funcionam as RNAs, onde com o propósito de garantir que a rede neural seja"inteligente", ou seja, capaz de raciocinar com alto repertório de conhecimento e então tomar boas decisões, é fundamental que a RNA passe por um processo de *treinamento*.

O treinamento de uma RNA é um processo iterativo que em alguns casos pode se tornar extremamente complexo. No processo de treinamento os pesos dos neurônios da rede são inicializados de forma randômica, e são ajustados durante todas as épocas do treinamento com a ajuda de *Funções de Ativação*, que vão introduzir uma não-linearidade na produção da saída de um neurônio. Além disso, existem outros parâmetros de configuração de uma RNA, tais como, *Taxa de Aprendizado (Learning Rate)* e *Taxa de Perda (Dropout Rate)*

que provocam a alteração direta dos pesos dos neurônios, modicando assim a saída que será repassada adiante na rede. Por fim, existem alguns *Algoritmos Otimizadores* que executam uma varredura e ajuste geral, de forma a balancear dos pesos dos neurônios, durante o processo de treinamento. Todos esses *Hiperparâmetros*, associados a arquitetura proposta da rede e a quantidade de épocas no treinamento, isto é, quantidade de vezes que a rede recebe uma entrada e processa seus dados até chegar em sua última camada e então produzir um resultado final, serão responsáveis por ditar a performace da RNA. O processo de treinamento de uma RNA tem como objetivo de capacitar o modelo para lidar com a maior variedade de situações e entradas possíveis, de forma que ela seja capaz de lidar com precisão em diversas abordagens de aplicação.

Porém, é necessária muita atenção e monitoramento desse processo, pois existem diversos problemas que podem afetar desde o processo de treinamento, deixando-o extremamente longo e sem perspectiva de evolução, até a real efetividade da rede. Dentre os principais obstáculos durante um processo de treinamento de uma RNA, temos:

- Desaparecimento e Explosão de Gradiente: Popularmente conhecidos como *Vanishing Gradient* e *Exploding Gradient*, são duas situações opostas tratando-se do balanceamento dos pesos de neurônios, porém ambas tornam a rede instável. De forma intuitiva, o desaparecimento do gradiente faz com que o coeficiente de alteração do peso do neurônio seja cada vez mais próximo de zero, provocando assim uma estagnação na evolução da rede. Já a Explosão de Gradiente seria o inverso, onde o coeficiente de alteração do peso cresce de forma exponencial, podendo atingir valores incapazes de serem representados (valores não-numéricos, ou NaN).

- Overfitting: Acontece principalmente quando treinamos nossa rede para um conjunto específico de características, fazendo com que a rede consiga interpretar apenas esse seleto grupo de características e deixando de lado todo o resto. Podemos analisar uma ocorrência de Overfitting quando para um conjunto de dados de treinamento o resultado da rede foi satisfatório, porém quando aplicado outro conjunto tido como novo, sua performance atinge valores muito baixos e incondizentes com o treinamento. Podemos observar esse comportamento até mesmo durante o processo de treinamento, se dado um momento a taxa de acerto da rede se mantém estagnado ou um aumento insignificante.

- Dataset Desbalanceado: Dataset se trata do conjunto de dados que se usa para para treinar uma RNA, podendo ser desde um banco de imagens até um banco de textos ou planilhas. Um dataset é particionado em 2 partes a fim de ser utilizado tanto no processo de treinamento da rede quanto no processo de teste e validação da mesma. Quando um Dataset não apresenta uma variação satisfatória dos dados, muito provavelmente a rede não estará preparada para lidar com essa parcela de

casos omissos. Exemplificando, se para uma RNA que classifica imagens de cachorros e gatos, for fornecido um dataset composto 90% de imagens de cachorros, a rede mostrará grande dificuldade em reconhecer imagens de gatos, pois durante seu treinamento ela não teve uma quantidade satisfatória de imagens de gatos para que houvesse o devido aprendizado.

- ❏ Hiperparâmetros Mal Definidos: Consiste na má definição dos hiperparâmetros de configuração da rede, citados previamente. Nos casos onde a Taxa de Aprendizado e/ou Taxa de Perda forem mal definidos no início do processo de treinamento, os pesos dos neurônios irão se alterar de forma descontrolada, causando instabilidade no treinamento e afetando diretamente o desempenho da RNA.

Desde o concebimento do primeiro modelo de RNA, diversos outros modelos com diferentes propósitos e características únicas foram desenvolvidos, de forma que a cada nova proposta de RNA, ela fosse capaz de lidar com problemas que até então não haviam solução, ou aprimorassem de forma significativa uma solução já existente. Tais RNAs poderiam ser aplicadas nas mais diversas áreas de atuação, tais como processamento de dados, predição de resultados e reconhecimento de imagens.

As seções a seguir abordarão os principais modelos de RNAs, de forma a evidenciar a evolução de cada modelo em relação a sua proposta anterior, construindo assim um passo a passo que leve ao modelo principal utilizado no presente trabalho, sendo ele o modelo Transformer. Cabe salientar que as Redes Transformers, que vêm revolucionando a área de Inteligência Artificial nos últimos anos, serão apresentadas, em maiores detalhes, no próximo capítulo (Capítulo 3), uma vez que correspondem ao modelo que inspira o presente trabalho.

## 2.2 Perceptron Múltiplas Camadas

A formação mais ingênua de uma RNA trata-se de um *Perceptron Multi Camadas (PMC)*. Nesse modelo, cada neurônio de uma camada está diretamente conectado com todos os neurônios da camada seguinte. Nas camadas intermediárias da rede, os neurônios contam com uma função de ativação é responsável pela aplicação de uma operação matemática sobre o valor de entrada para então propagá-lo adiante. São esses valores, também conhecidos como pesos de cada neurônio, os responsáveis por efetuar a classificação da entrada, ou seja, responsáveis por produzir a saída da rede. Dessa forma, uma rede devidamente treinada é aquela cujo peso de seus neurônios das camadas intermediárias estão adequadamente balanceados ao problema proposto, sendo acertiva na produção de uma resposta como saída da rede.

Um PMC pode ser dividido em três partes distintas, cada uma com sua função e características bem definidas. São elas:

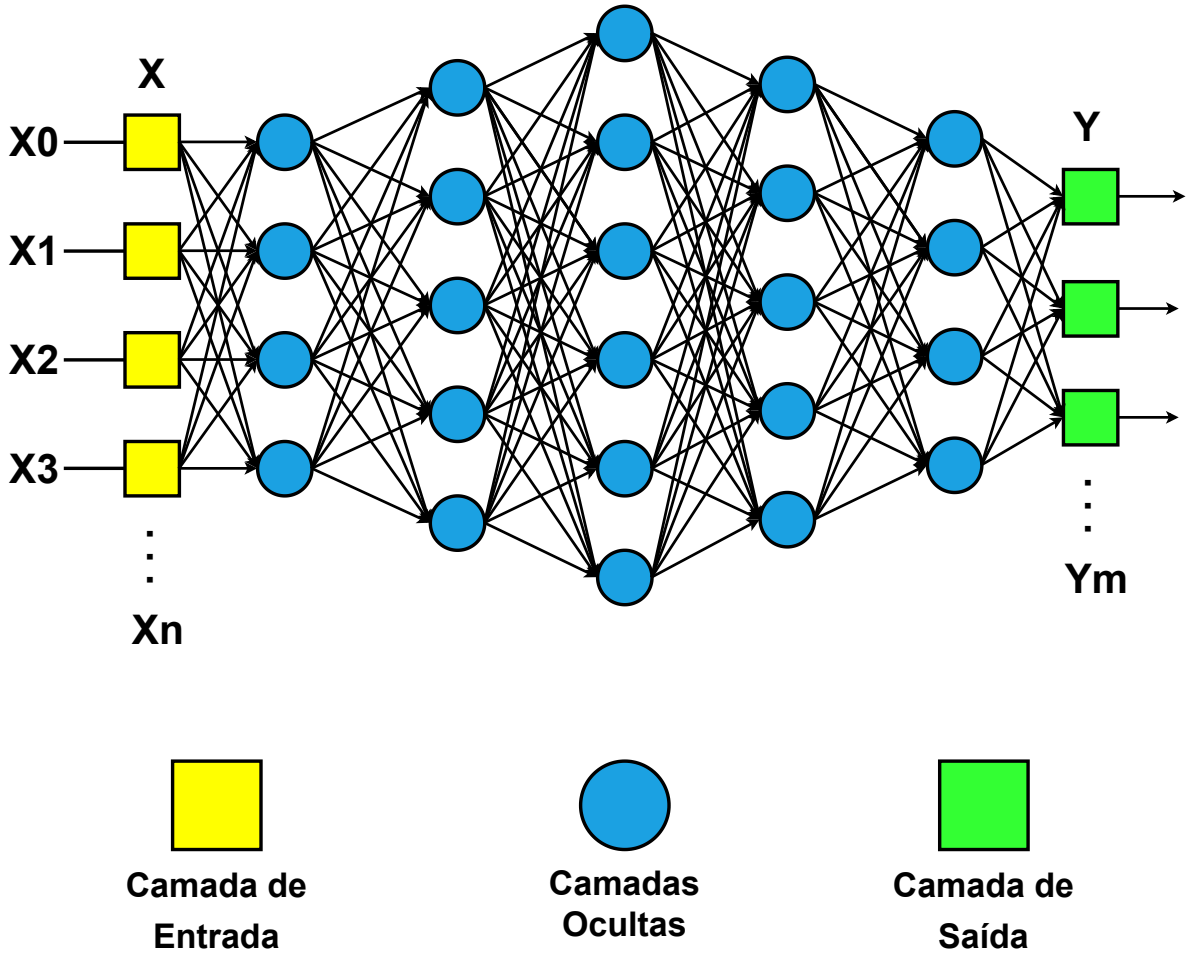



Figura 1 – Arquitetura base de um Perceptron Múltiplas Camadas

- *Camada de Entrada (Input Layer)*: Corresponde a primeira camada da RNA, ilustrada em amarelo na Figura 1, é responsável por receber as instâncias do problema a fim de serem processadas nas camadas intermediárias da rede. O tamanho dessa camada é definido pela representação da instância de entrada, que varia de acordo com o problema modelado. A modelagem das instâncias é fundamental para a obtenção de bons resultados, por exemplo, em uma rede neural PMC, onde o objetivo é classificação de animais, os parâmetros de entrada que caracterizam a instância poderiam ser: *Porte*, *Pelagem*, *Tipo de alimentação*, *Garras*, *Dentes*, etc...

- *Camadas Ocultas (Hidden Layers)*: Correspondem as camadas intermediárias da rede, ilustrada em azul na Figura 1, são responsáveis por efetuar o processamento das instâncias, realizando cálculos e balanceando os valores numéricos dos pesos para que os mesmos estejam ajustados na proporção ideal para a resolução do problema. São nessas camadas que acontecem a extração das características, de forma a constituir o modelo de previsão da rede. Diversas camadas podem compor as Camadas Ocultas de uma RNA, cada camada com seu número de neurônios e pesos em suas conexões com a camada sucessora. A modelagem das camadas ocultas é um fator de suma

importância para o bom desempenho da rede, pois além de ditar a efetividade da predição também influencia o tempo de treinamento da rede.

- *Camada de Saída (Output Layer)*: Corresponde a última camada da RNA, ilustrada em verde na Figura 1, responsável por receber os dados processados por todas as camadas intermediárias da rede para finalmente transformar esses dados em informações a fim de serem interpretadas por seres humanos. Por exemplo, tendo como referência o problema de reconhecimento de dígitos escritos à mão (HE et al., 2016), a camada de saída da RNA proposta será formada por dez neurônios, cada um deles representará um dos dez dígitos possíveis a serem reconhecidos. Ou seja, será um vetor de dez posições onde após a aplicação de uma função SoftMax, que normaliza os valores do vetor entre 0 e 1, cada posição do vetor terá um valor correspondente a probabilidade da imagem de entrada ser identificada como o número que a posição no vetor representa.

## 2.3 Redes Neurais Convolucionais

Redes Neurais Convolucionais (RNC ou CNN) representam um passo importantíssimo na evolução das RNAs, essa nova arquitetura proposta faz uso de uma camada fundamental e característica desse modelo, as chamadas Camadas de Convolução ( *Convolutional Layer*). Essas camadas são compostas por uma série de filtros, que consistem em matrizes quadradas de dimensão menor do que o tamanho da representação das instâncias, iniciados com valores aleatórios que vão ao longo do processo iterativo das camadas de convolução ajustando seus valores a fim de perpetuar características extraídas. Esses filtros percorrem toda a extensão da instância, aplicando operações que visam qualificar as características presentes, essas camadas são fundamentais extratores de características e padrões. Ou seja, diferente de um PMC, onde as características (*Features*), precisam ser definidas manualmente por aquele que opera a rede, uma CNN é capaz de realizar a extração de features de forma automática.

Esse modelo de RNA tem notória relevância quando aplicado a análise e processamento de imagens, precisamente pela ação em conjunto à camadas de Pooling, tais camadas exercem a função de reduzir a dimensão da representação da instância, de forma a propagar somente as características importantes. A Figura 2 mostra a representação da arquitetura de uma CNN, evidenciando suas camadas e operações.

As subseções seguintes apresentarão alguns dos modelos que assumem grande participação nas principais pesquisas no âmbito de aprendizado profundo e visão computacional, descrevendo suas arquiteturas e expondo suas principais características. De modo a ilustrar suas particularidades e evidenciar sua versatilidade e eficácia em diversas aplicações do mundo moderno.

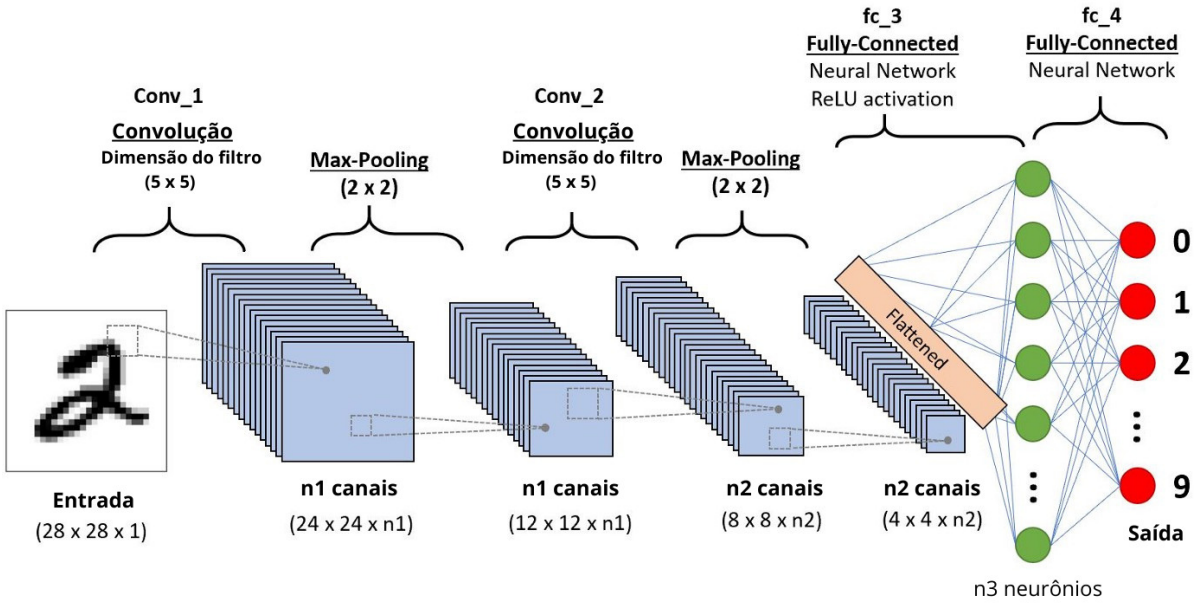



Figura 2 – Representação do modelo da Arquitetura de uma CNN. Adaptado de [ https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 ]

### 2.3.1 AlexNet

AlexNet foi uma CNN criada para uma competição de RNAs de classificação de imagens (KRIZHEVSKY; SUTSKEVER; HINTON, 2012), ela obteve excelentes resultados e chegou a superar os atuais modelos do estado-da-arte. Sua arquitetura é composta por 5 camadas convolucionais, algumas delas sucedidas por camadas de max-pooling e três camadas totalmente conexas, finalizando em uma camada de saída com função softmax. Esse modelo conta com operações ReLU (Rectified Linear Unit) de balanceamento dos pesos das conexões de neurônios, além de operações de Dropout, que "desativam" alguns neurônios que não atingem uma margem de probabilidade. Sua arquitetura está representada na Figura 3.

### 2.3.2 VGG

Foi um modelo de RNA derivado do modelo AlexNet, porém com foco especial em uma característica das RNs, a profundidade. As principais diferenças do modelo VGG para o AlexNet são: redução da dimensionalidade dos filtros das camadas de convolução; introdução de filtros de convolução de dimensão 1x1, auxiliando nos cálculos das funções de aproximação; menor quantidade de parâmetros na RN. Com isso, esse modelo conseguiu superar a AlexNet na tarefa de reconhecimento de imagem, se tornando referência para tarefas dessa natureza (SIMONYAN; ZISSERMAN, 2014).

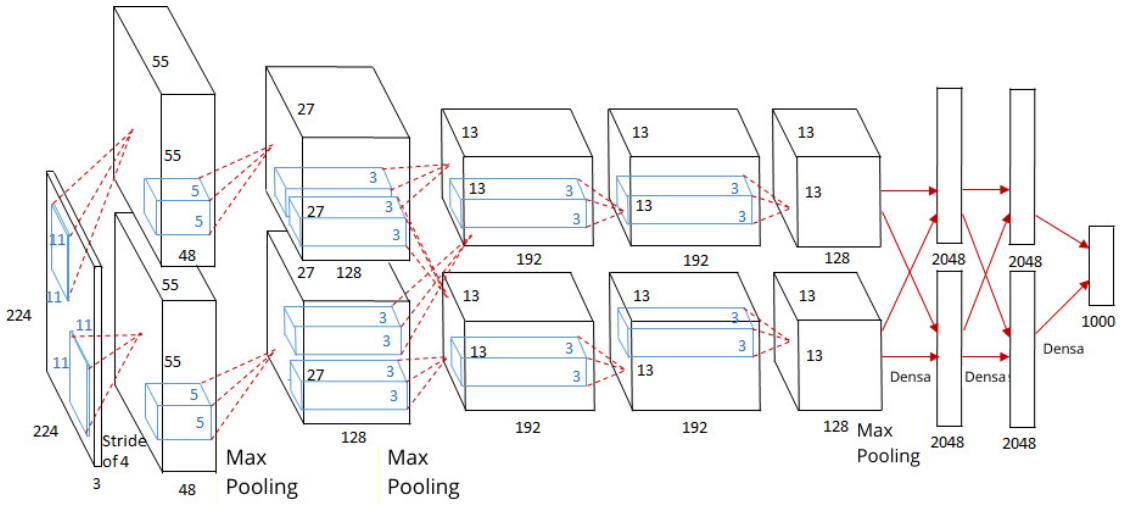



Figura 3 – Arquitetura AlexNet. Adaptado de [https://bainsa.xyz/image-net-classification-with-dcnn/]

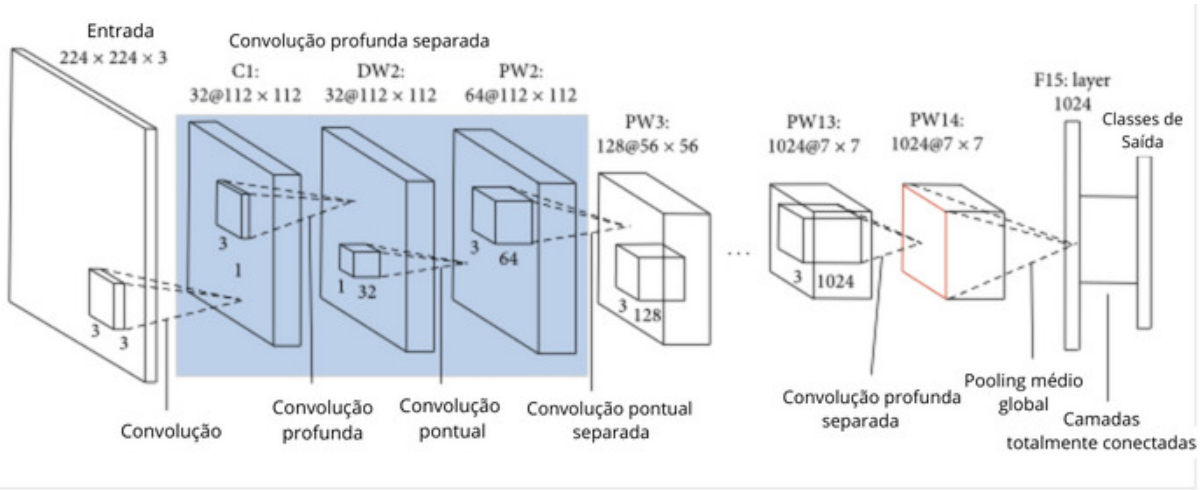



Figura 4 – Arquitetura MobileNet. Adpatado de [https://wikidocs.net/165429]

### 2.3.3 MobileNet

MobileNet foi um modelo de arquitetura de RN que ficou muito famosa por ganhar a ImageNet Challenge: ILSVRC 2012 (RUSSAKOVSKY et al., 2015). A principal ideia da abordagem da arquitetura MobileNet, representada na Figura 4, consiste em separar a parte convolucional em duas camadas separadas. A primeira é chamada de convolução profunda, onde é aplicada uma filtragem leve para cada canal de entrada. A segunda camada, é chamada de convolução pontual, ela é responsável por construir novas features através de combinações lineares vindas do canal de entrada (HOWARD et al., 2017).

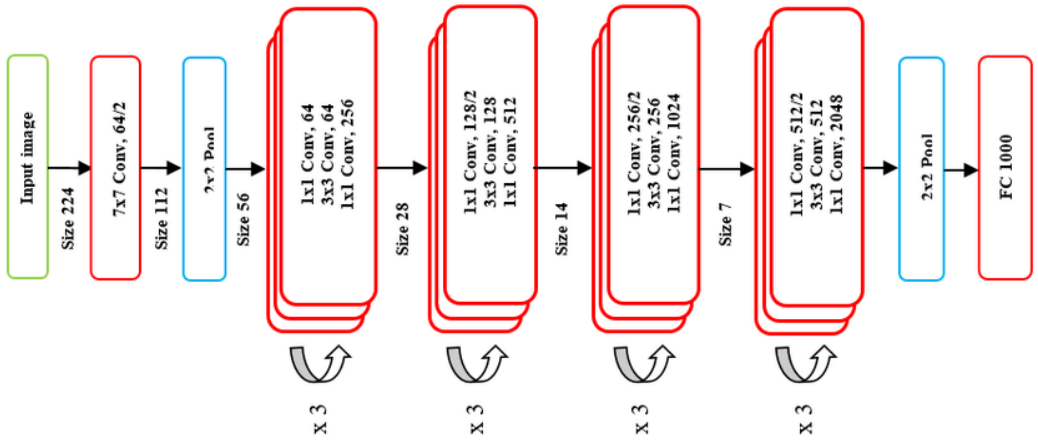



Figura 5 – Arquitetura ResNet50. Adaptado de [https://towardsdatascience.com/the-annotated-resnet-50-a6c536034758]

### 2.3.4 ResNet

A arquitetura do modelo ResNet (Resnet20 e Resnet50), representada na Figura 5, se inspira no modelo das redes VGG (SIMONYAN; ZISSERMAN, 2014), o modelo ResNet possui uma quantidade menor de filtros que consequentemente as torna menos complexas. As camadas convolucionais desse modelo são, em sua maioria, filtros de dimensão 3x3 que seguem o seguinte comportamento: para entradas de tamanho igual à saída, a dimensão da representação das instâncias são mantidas; para as entradas com dimensionalidade previamente reduzida, há uma dobra nessa dimensionalidade, preservando finalmente o valor da dimensão da representação da instância. Esse modelo conta com operações de downsampling posterior as camadas de convolucionais, encerrando em uma camada de pooling geral que é sucedida de uma camada totalmente conexa de tamanho 1000.

## 2.4 Redes Neurais Recorrentes

Redes Neurais Recorrentes (RNR ou RNN) são um modelo de RNA que implementam mais do que apenas o fluxo unidirecional (também conhecido como arquitetura Feed-foward) padrão das arquiteturas ingênuas de RNAs. Nos PMCs existe um conceito de sequencialidade e tempo entre as fases do aprendizado da rede, onde, de forma unidirecional, os ajustes de peso de um neurônio são sempre propagados para as camadas subsequentes. Na arquitetura das RNN esse conceito de unidirecionalidade é quebrado e então as redes passam a ter ligações com neurônios antecessores (este conceito é conhecido como BackPropagation), ou seja, ligações no sentido contrário do fluxo original, que realimentarão a rede com informações em tempo de execução. Dessa forma o ajuste dos pesos de uma determinada iteração do treinamento influencia diretamente no ajuste de pesos da iteração que a sucede, e assim por diante. A Figura 6 ilustra uma arquitetura onde as conexões de BackPropagation estão representadas na cor laranja.

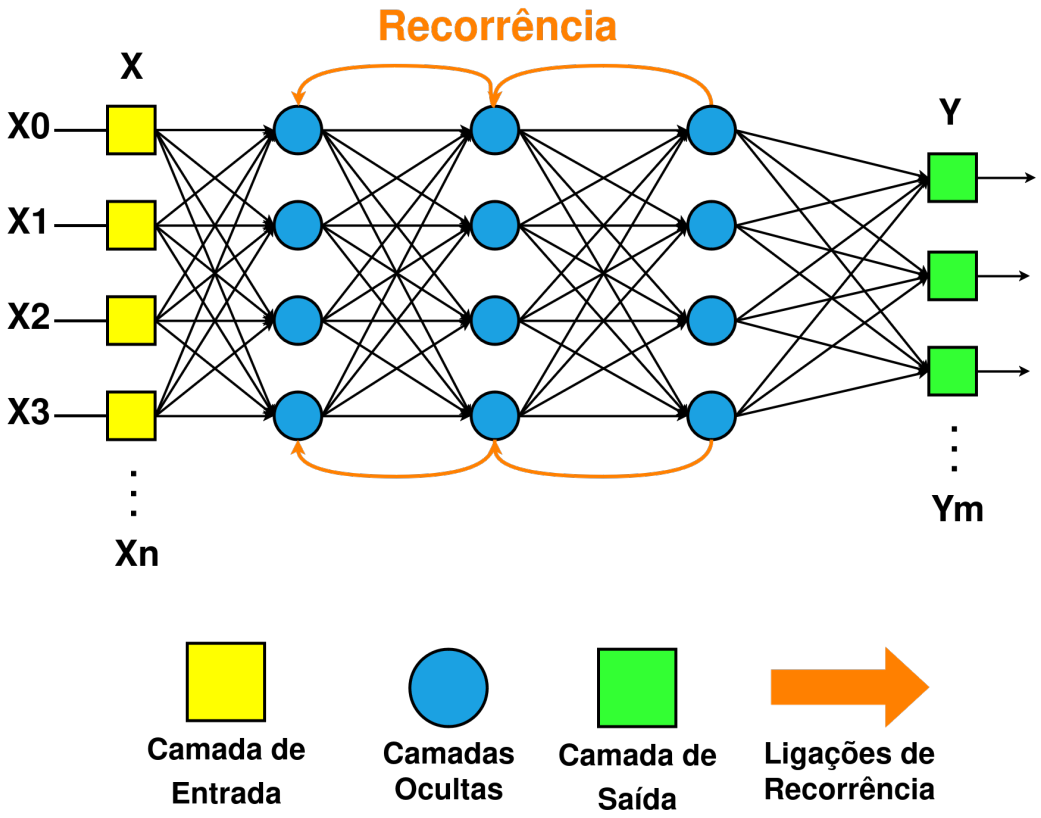



Figura 6 – Arquitetura exemplo de uma Rede Neural Recorrente

Dessa forma, a partir do momento em que uma informação a ser processada chega numa camada com conexões de retroalimentação, o ajuste de peso é repassado a frente e propagado para camadas anteriores a fim de auxiliar no balanceamento dos pesos das entradas que ainda virão. Essa estratégia, aplicada em conjunto ao conceito de FeedBack Loops, que é a utilização de um estado interno que se atualiza a cada nova iteração, o modelo passa a ter condições de administrar a sequencialidade da rede. De tal maneira, além de tornar a rede capaz de reconhecer relações e dependências entre elementos sequenciais da entrada, o cálculo e refinamento dos pesos se torna muito mais preciso, contribuindo categóricamente para a aprimoração do desempenho da rede.

Não apenas conexões com neurônios anteriores, também é póssivel haver conexões de um neurônio com ele mesmo, criando uma ideia de *aninhamento* de neurônios na rede. Fica fácil compreender esse conceito e seu funcionamento se "desenrolar-mos" a rede, conforme ilustrado na Figura 7, a fim de evidenciar como as ligações de recorrência transportam os ajustem de peso e influenciam de forma direta as futuras iterações da rede.

Porém, nas circunstâncias onde os pesos das ligações de feedback estão desproporcionalmente ajustados para valores maiores que 1, e a entrada da rede consiste de uma sequência de dados de tamanho considerável, a execução dos cálculos de ajuste de pesos toma proporções extraordinárias. Ou seja, mais uma vez o problema do gradiente ex-

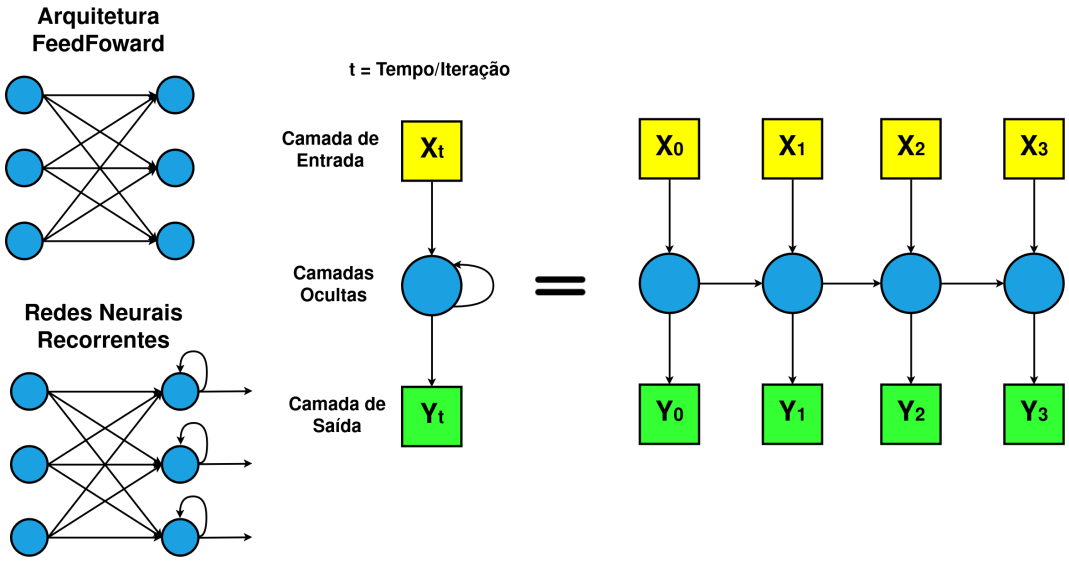



Figura 7 – Comparativo entre Redes Recorrentes e Redes FeedFoward

plosivo seria facilmente reproduzido, e na mesma proporção, caso o ajuste dos pesos das ligações fossem calculados para valores próximos a zero, teríamos o problema de desaparecimento de gradiente.

Essa abordagem se mostrou muito eficaz e trouxe exímios ganhos, tanto de performance quanto em tempo de treinamento, para o campo de pesquisa e aplicações de RNNs. Um exemplo de RNN que vêm se destacando nas pesquisas recentes são as LSTMs, as quais serão introduzidas na próxima subseção.

### 2.4.1 LSTM

Fica cada vez mais notório conceito da cronologia do aprendizado, que está diretamente ligada as instâncias da rede, isso evidência o principal dilema a cerca de redes neurais: *"Como lidar com informações antigas e recentes sendo capaz de balancear suas relevâncias para o processamento da informação atual"*. Nesse sentido, toda informação/conhecimento *antigo* pode ser entendido como dados que foram processados nas instâncias iniciais da rede, ou de considerável distância em relação à instância atual. Do mesmo modo, informações *novas* são aquelas produzidas por instâncias mais recentes ou de considerável proximidade em relação à instância atual de processamento da rede.

Na arquitetura ingênua das Redes Neurais Recorrentes, tanto informações "*antigas*" quanto "*novas*" trafegam pelo mesmo canal, as ligações de recorrência. Ou seja, não existe uma separação para tratamento individual de cada uma delas, não há como determinar a relevância de uma sobre a outra em prol do processamento de uma nova instância.

As redes neurais *LSTM (Long Short Term Memory)* propõem uma arquitetura cujo objetivo é tratar separadamente informações de Longo Prazo (antigas) e informações de Curto Prazo (novas) para então serem capazes de levar em consideração a importância

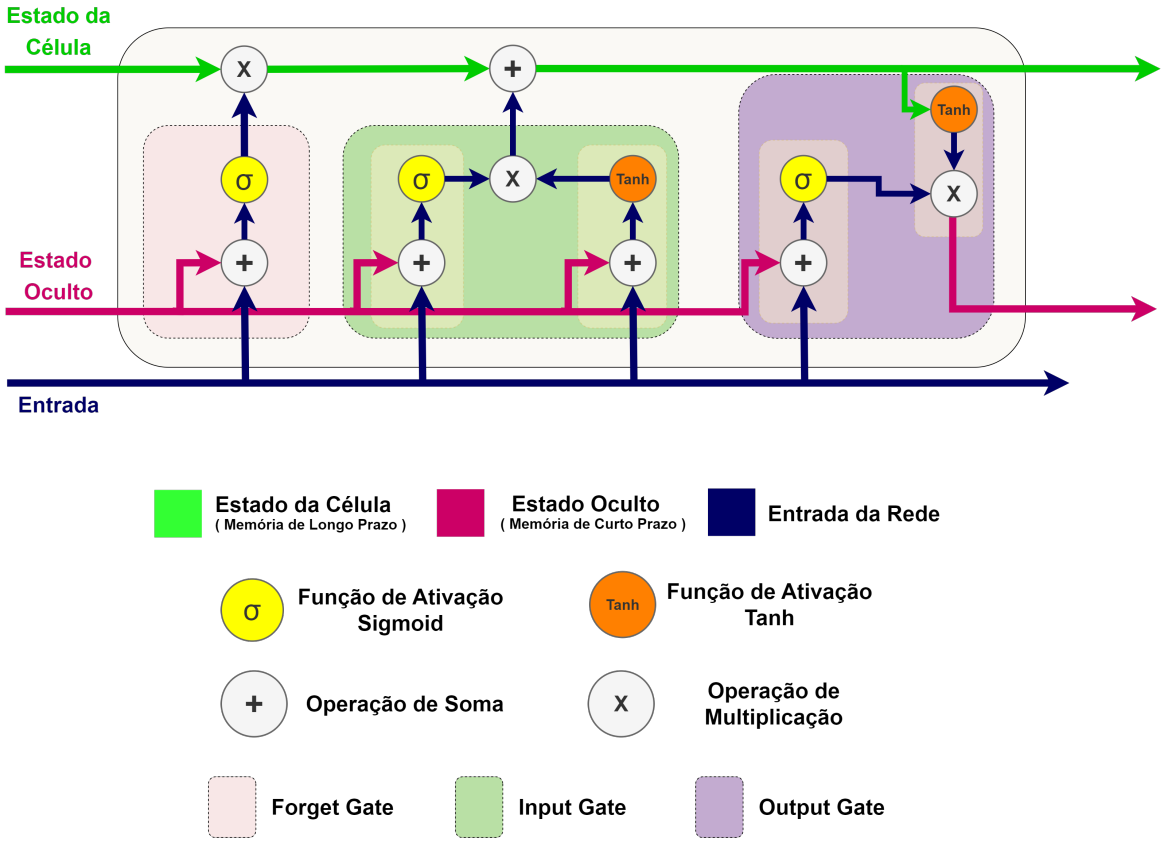



Figura 8 – Arquitetura LSTM

de ambos os tipos de informação no processamento da informação da instância atual. Ou seja, o quão relevante o conhecimento antigo e novo são para o atual momento da rede, esse balanço é fundamental para fazer com que a rede seja capaz de"esquecer" informações irrelevantes e ter influência apenas do o que for importante para o processamento em tempo real.

A Figura 8 ilustra e mostra em detalhes toda a arquitetura do modelo LSTM, evidenciando seu processo de funcionamento e todos os componentes que constituem o modelo.

O canal por onde passa a informação a longo prazo é chamado de *Estado da Célula*, está representado na Figura 8 com a cor verde, e nele não há muita aplicação de ajustes de pesos justamente por se tratar de informações que precisam perdurar por várias iterações da rede. Já o canal por onde percorre a informação a curto prazo, representado na Figura 8 pela cor rosa, é chamado de *Estado Oculto*, nele existem diversos pesos que irão alterar seu valor para cada etapa do processo da rede. Ambos os canais, Estado da Célula e Estado Oculto, são inicializados com valores nulos e serão atualizados conforme o processar das iterações da rede.

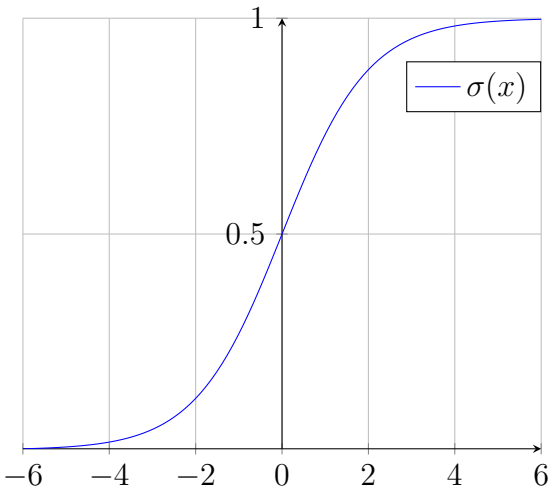
O processamento de uma LSTM se inicia com a junção do valor do Estado Oculto e o valor de entrada da rede, ambos influenciados por pesos de ajuste, que por sua vez será aplicado a uma função Sigmoid , que tem como objetivo converter o valor de entrada para um número entre 0 e 1. O resultado dessa operação determina a *taxa de esquecimento* da

memória a longo prazo, ou seja, quão relevante a informação presente no Estado Ocultado se faz para a entrada a ser processada pela rede na iteração atual. Dessa forma a rede consegue"esquecer" informações irrelevantes, em outras palavras, ela consegue abdicar e fazer com que informações antigas não tenham tanta influência para o estado presente da rede. Essa etapa, que embora tenha como objetivo definir um coeficiente que representa o quanto da memória de longo prazo será conservada, é chamada de *Forget Gate*.

O estágio seguinte é composto por dois blocos, ilustrados na Figura 8, onde o bloco a direita irá combinar o valor presente no Estado Oculto com o valor de entrada da rede, ambos ajustados por pesos próprios, em sequência aplica a função Tanh, que transforma o valor de entrada em um número entre -1 e 1, para enfim gerar uma um coeficiente que represente a *Potencial Memória de Longo Prazo*, do inglês Potential Long Term Memory. Já o bloco à esquerda tem como objetivo definir o quanto dessa Potencial Memória de Longo Prazo será atribuída ao Estado da Célula, ou seja, a combinação desses blocos é responsável por atualizar o valor da Memória de Longo Prazo, incorporando ao Estado da Célula uma porcentagem relativa à Memória de Curto Prazo, para que assim a informação produzida na iteração atual seja propagada e usufruída nas iterações futuras. Essa etapa recebe o nome de *Input Gate*.

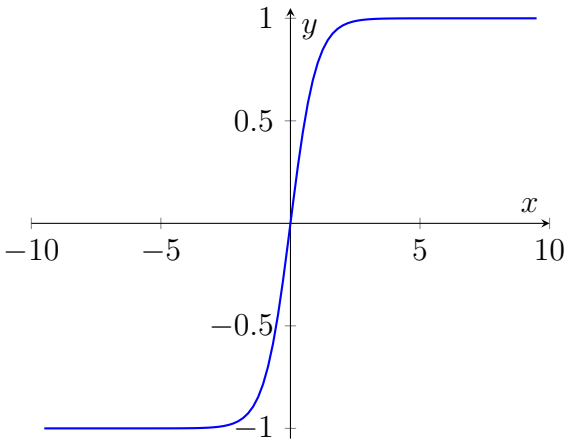
A Função de Ativação Sigmoid é definida da seguinte maneira:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$




A Função de Ativação Tanh é definida da seguinte maneira:

$$tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} = \frac{1 - e^{-2x}}{1 + e^{-2x}}$$




Por fim, a etapa final do processamento de uma célula LSTM, chamada de *Output Gate*, também possui 2 blocos, similares ao estágio anterior. O bloco a direita combina o valor do Estado da Célula aplicado a função de ativação Tanh, chamado até então de *Potencial Memória de Curto Prazo*, com o valor produzido pela célula a direita, um valor que representa a taxa de quanto essa Potencial Memória a Curto Prazo será propagada, processo similar ao cálculo do novo valor para Estado da Célula no estágio anterior. Sendo assim, sua saída corresponde ao novo valor para Memória de Curto Prazo que será atribuída ao Estado Oculto da rede. Tal valor representa a saída da célula LSTM, que por sua vez pode alimentar uma outra célula LSTM, criando assim uma rede profunda capaz de processar inúmeras instâncias, sempre mantendo em sua conjuntura informações de Curto e Longo prazo.

Uma característica fundamental da arquitetura LSTM se dá ao fato de durante todas as iterações de seu processamento, não há operações de ajuste de pesos excessivos na Memória de Longo Prazo, o que possibilita a preservação de informações importantes mesmo que tenham sido processadas em inúmeras instâncias anteriores, além de colaborar para que não haja nem o desaparecimento ou explosão do gradiente, problema que até então outros modelos de RNAs sofrem gravemente. Dessa forma, o cálculo da Memória a Curto Prazo da rede sofre influência direta não só dos pesos aplicados a ela, mas também da Memória de Longo Prazo, que foi devidamente tratada para que seu valor não extrapole um valor real e possa ser utilizado em todas as iterações da rede.

Capítulo 3

# Modelo Desenvolvido

A sequencialidade das entradas é um fator obrigatório nas RNAs, de forma a demandar uma entrada por iteração e estabelecendo, assim, uma relação de ordem e ligação entre as entradas. Exemplificando, a fim de processar em uma RNA informações climáticas do mês de fevereiro, deve-se alimentar a rede inicialmente com informações relacionadas ao dia 1º de fevereiro, seguido do dia 2º de fevereiro e assim por diante. No contexto de processamento de linguagem natural, a manipulação de uma frase é dada palavra por palavra, de forma que o processamento e cálculo de variáveis que ditariam seu contexto teriam apenas as palavras anteriores à aquela processada como referência. A língua portuguesa conta com diversos recursos onde o contexto de uma palavra pode ser desconhecido se as palavras que a sucedem forem desconsideradas durante sua análise. Para que seja inferido de forma adequada o contexto de uma palavra em uma frase qualquer, se faz necessário analisar a frase como um todo, desde sujeito, predicado, até quaisquer outros elementos da linguagem presentes na frase.

A arquitetura de Redes Neurais chamada Transformer foi revolucionária justamente por se propor a resolver este desafio, tal abordagem permite a inserção e processamento de múltiplas entradas simultâneas em uma única iteração, sendo capaz de atribuir relação de ordem entre elas sem que haja a necessidade de iteração sequencial e individual das entradas. Essa arquitetura sofisticada permite o processamento de entradas de tamanho expressivo, sem prejudicar o desempenho do treinamento e eficiência geral da rede, se apoiando apenas no poder de processamento computacional para sua execução. Poder Computacional não é um problema para as grandes empresas do mundo atual, os equipamentos tecnológicos são excepcionalmente evoluídos e contam com um extremo poderio de processamento.

Esse modelo de RNA lida de forma eficaz os desafios comuns de arquiteturas passadas, sua abordagem de processamento simultâneo permite um relevante ganho de desempenho no processo de treinamento da rede, além de ser capaz de reter informações a longo prazo e manter suas relações com outros dados a serem processadas. Essa arquitetura Transformer que lida com Processamento de Linguagem Natural (a partir daqui referenciada por

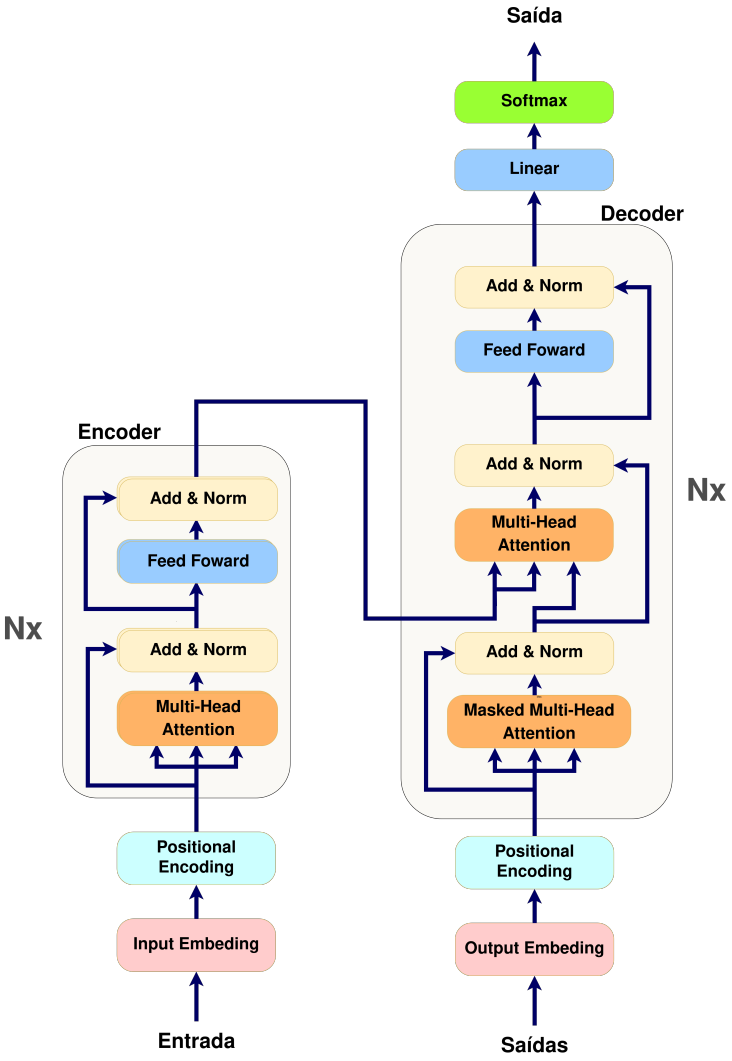



Figura 9 – Arquitetura da RT-PLN

RT-PLN) foi inicialmente proposta em (VASWANI et al., 2017) e se tornou precursora dos principais modelos processadores de linguagem natural generativos, tais como *BERT* (DEVLIN et al., 2019), *GPT-3* (OPENAI, 2024a), *T5* (RAFFEL et al., 2023), por justamente contar com 2 módulos principais: *Encoder* (módulo de mapeamento de palavras, interrelacionando-as) e *Decoder*(módulo generativo, apto a produzir texto).

A Figura 9 ilustra a arquitetura do modelo RT-PLN, expondo sua estrutura e a composição de seus módulos, evidenciando todo o processo de funcionamento da rede, diferenciando a entrada e operações características de cada um dos módulos Encoder e Decoder, além de expor a forma como eles se relacionam a fim de colaborar para a geração de uma saída coesa da rede.

O módulo Encoder recebe como entrada uma sequência de dados, no caso da RT-PLN uma sequência de palavras que formam uma frase, para então processá-las por diversas camadas a fim de construir relações entre elas e assim produzir represetações ricas em contexto. Tais representações serão utilizadas pelo módulo Decoder para produção de

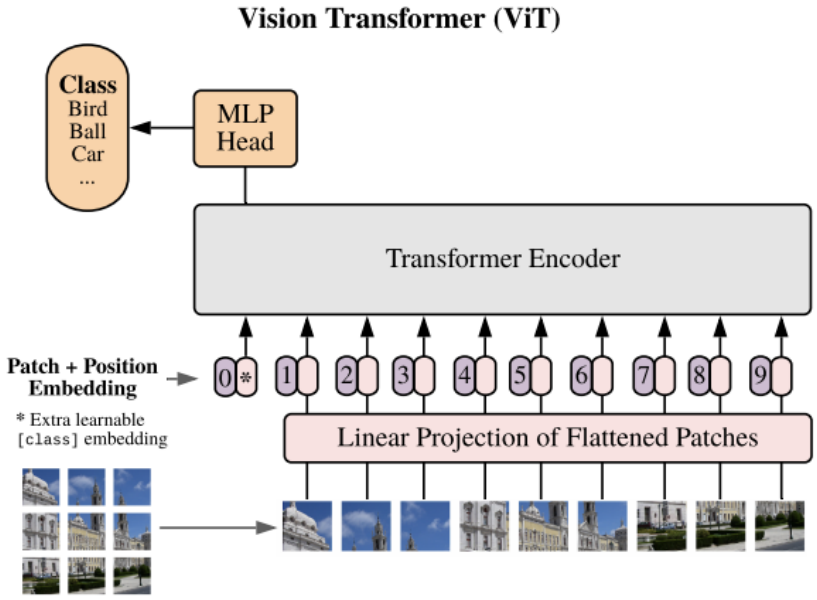



Figura 10 – Arquitetura do modelo RT-PI. Adaptado de [https://arxiv.org/pdf/2010.11929]

uma nova frase que servirá como saída, ou seja, uma resposta que faça sentido perante à aquela dada como entrada. A coesão da frase gerada é garantida pela utilização nos mecanismos de *Self-Attention* e redes FeedFoward, que permitem a captura de contexto entre palavras mesmo que muito distantes, presentes na arquitetura do modelo.

Inspirados no modelo processador de Linguagem Natural apresentado em (VASWANI et al., 2017), em (DOSOVITSKIY et al., 2021) os autores propõem uma arquitetura simplificada da rede Transformer capaz de efetuar Processamento de Imagem (aqui referenciada como RT-PI). No caso, tal arquitetura requer apenas o módulo Encoder, o qual será utilizado para analisar e correlacionar cada parte de uma dada imagem, de forma a tornar a rede apta a clasificar os objetos nela presente. As imagens utilizadas em (DOSOVITSKIY et al., 2021) correspondem a Datasets públicos de média/larga escala, desenvolvidos para outras treinamento e teste de diversos modelos diferentes RNAS, tais como ImageNet e ResNet.

Assim sendo, considerando o fato de que este trabalho se encaixa no contexto de projetos de pesquisa que usam games como estudo de caso para desenvolvimento e aplicações de técnicas de Aprendizado de Máquina, bem como o fato de que, em games, o processamento de imagens é um desafio crucial, este trabalho tem por objetivo replicar a arquitetura do modelo RT-PI, chamado de *Vision Transformer*, proposto em (DOSOVITSKIY et al., 2020), o qual é ilustrado na Figura 10.

No modelo RT-PI, a rede recebe uma imagem como dado de entrada e executa algumas operações de"tratamento" a fim de preparar a entrada para seu processamento na rede. Assim como no modelo RT-PLN, o módulo Encoder recebe uma sequência de dados que

caracterizam a imagem de entrada, operando sobre eles e assim construindo uma estrutura que represente da imagem. Essa representação elaborada pelo módulo Encoder serve de entrada para um módulo de Fully Connected Layers irão por fim caracterizar e classificar a entrada em alguma das classes previamente modeladas pelo usuário da rede.

Para tanto, o banco de imagens do presente trabalho, chamado de CIFAR-100 (KRIZHEVSKY, 2009), foi extraído de um acervo público que é amplamente utilizado por diversos cientistas e pesquisadores de todo o mundo. Esse banco de imagens é composto por 60.000 imagens, separadas entre imagens para treinamento e imagens para teste, onde cada uma delas recebe um rótulo que descreve qual o seu conteúdo, ou seja, a qual classe pertence aquela imagem, exemplo: *"castor"*, *"golfinho"*, *"lontra"*, *"foca"*, *"baleia"*, etc.

Neste contexto, considerando a relevância das redes Transformer no cenário atual da Inteligência Artificial, o presente capítulo tem por objetivo apresentar, em um primeiro momento, o módulo *Encoder* no qual se baseia a RT-PI aqui implementada (Seção 3.1), e, na sequência , o módulo *Decoder* usado na arquitetura completa da RT-PLN (Seção 3.2).

A título de completude, como o módulo Encoder é comum tanto à RT-PI quanto à RT-PLN, e como as entradas de tal módulo, em cada uma dessas redes, diferenciarem-se entre si em função do papel que cada uma exerce (ou seja, texto, no caso da RT-PLN ou imagem, no caso da RT-PI), a apresentação da entrada de tal módulo será subdividida em duas subseções distintas. Além disso, ainda a título de completude, apesar da RT-PI implementada neste trabalho não requerer o uso do módulo Decoder, na Seção 3.2 será apresentada uma descrição desse módulo, uma vez que ele é fundamental no contexto das redes Transformer.

## 3.1 Encoder

Na frase *"O gato comeu o rato."*, a ordem das palavras na frase constrói uma relação clara a respeito de quem executa tal ação sobre quem, o que torna possível sua interpretação e atribuição de sentido conforme a da cadeia predatória desses dois animais no mundo real. O modelo deve ser capaz de fazer a mesma interpretação, distinguindo com exatidão a correlação entre as palavras da entrada. Tal dinâmica acontece em virtude do inovador mecanismo de *Self-Attention* implementado no módulo Encoder, que para cada palavra, calcula um coeficiente de similaridade com todas as outras palavras da frase, incluindo ela mesma. Esses valores irão auxiliar a rede a codificar suas respectivas palavras, tendo como base a relação entre elas, a fim de que a rede consiga dar *"atenção"* somente a palavras com maior coeficiente de similaridade e interrelancionando-as de forma coerente.

De forma sucinta, este é o principal objetivo do módulo Encoder, mapear os elementos de uma entrada e criar relações entre eles por meio do mecanismo de Atenção proposto por (VASWANI et al., 2017). Cabe salientar que a arquitetura do módulo Encoder perma-

nece igual para ambos os propósitos, seja Processamento de Imagem ou Processamento de Linguagem Natural, havendo sutis diferenças apenas nas etapas de entrada e saída do módulo. Tais divergências serão abordadas em detalhes nas subseções de Entrada, Processamento e Saída ( Subseções 3.1.1, 3.1.2 e 3.1.3 respectivamente).

### 3.1.1 Entrada do Encoder

Conforme Figura 10, o Encoder corresponde ao módulo da RT-PI que recebe a imagem a ser classificada pela rede.

Para que o módulo Encoder consiga atuar sobre um determinado dado de entrada, seja ele no formato de texto ou imagem, é necessário que haja um tratamento desse dado na etapa que antecede seu acesso ao módulo Encoder. Uma vez que redes neurais não são capazes de utilizar letras para a execução de suas operações aritméticas, é fundamental transformar esse dado em algo que seja legível para a rede em baixo nível, ou seja, o formato da entrada deve ser convertido em uma estrutura que seja viável e passível de ser processada pela rede. Existem diversas maneiras de realizar essa conversão de formato, cada RNA a realiza de forma a melhor se adaptar ao seu modelo e propósito. Essa etapa de transformação do formato da entrada acontece na camada denominada como *Input Embeding*.

Além disso, embora um dos grandes avanços dessa arquitetura seja a entrada simultânea de várias informações, isso não significa que a ordem delas não seja importante. No processamento de linguagem natural, na maioria dos casos a ordem em que as palavras estão dispostas afeta diretamente o sentido da frase. A título de exemplo, na língua portuguesa, a frase *"Rita adora viajar!"* perde completamente seu sentido se a ordem das palavras for alterada para *"Viajar adora Rita!"*, sendo assim, deve-se respeitar a ordem das palavras a fim de que o contexto possa ser precisamente compreendido e a rede não seja enviesada por entradas incondizentes. A operação de identificação e consideração de posicionamento de diferentes"pedaços" da entrada é chamada de *Positional Encoding* e acontece de forma similar entre os modelos RT-PLN e RT-PI, essa informação é incorporada a representação da entrada de forma a ter influência nos cálculos da rede.

As subseções a seguir apresentarão em detalhes as características de todo o processo de entrada de dado no módulo Encoder, a Subseção 3.1.1.1 abordará a entrada relativa ao modelo RT-PLN enquanto a Subseção 3.1.1.2 abordará a entrada relativa ao modelo RT-PI. Vale salientar que, pelo fato do presente trabalho ter enfoque no modelo RT-PI, o processo de entrada ao módulo Encoder do modelo RT-PI será detalhado de forma profunda na Subseção 3.1.1.2.

#### 3.1.1.1 Entrada do Encoder da RT-PLN

No modelo RT-PLN, a conversão da entrada para um formato numérico compreensível pela rede é realizado pelo método conhecido como *Word Embeding*, um dos mais utilizados no ramo de redes neurais, nele é possível representar uma palavra utilizando apenas números, convertendo uma sequência de letras em uma estrutura composta de um vetor de números reais. Com isso, a entrada da rede que antes estava configurada em um formato irreconhecível pela rede, agora se torna familiar e está apta a ser processada pelos demais componentes da RT-PLN.

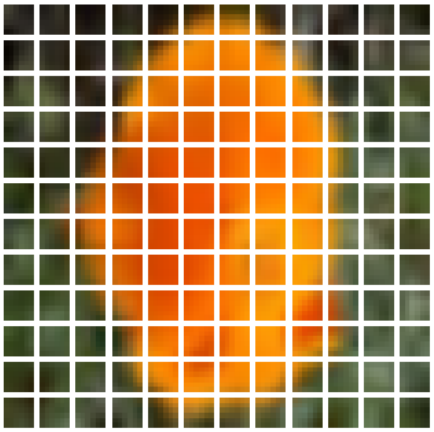
Para realizar a operação de Positional Encoding, é adicionado à representação numérica de cada palavra um coeficiente relativo à sua posição. O valor desse coeficiente é calculado utilizando a mescla de diversas funções de *seno* e *cosseno*, que irão garantir que os valores que representam uma determinada posição sejam característicos dela mesma e não possa se repetir para uma outra posição independente de qual for o tamanho da entrada. Com essas conversões devidamente aplicadas, a entrada está pronta para ser aplicada ao módulo Encoder.

#### 3.1.1.2 Entrada do Encoder da RT-PI

Da mesma forma que se fez necessário converter palavras em uma representação numérica que seja legível pela rede, o mesmo deve acontecer quando trata-se de processamento de imagens. De forma análoga, uma rede neural não é capaz de receber puramente uma imagem sem que haja sua devida transformação em um objeto passível de ser processado pela rede neural em questão.

Uma imagem pode ser entendida como uma sequência de pixels, porém se essa sequência for utlizada de forma análoga à palavras de uma frase, o calculo de atenção do modelo deverá ser feito comparando cada unidade de píxel com todos os outros da mesma imagem. Dessa forma, o modelo se restringe a aceitar apenas imagens de baixa resolução, caso contrário a quantidade de cálculos comprometeriam severamente a complexidade do modelo, tornado-o extremamente lento e ineficiente. Por isso, de forma semelhante ao processamento de texto, onde a entrada é constituída por uma frase com várias palavras, o Vision Transformer adota a estratégia de fracionar a imagem de entrada em diversas *sub-imagens*, ou "pedaços", a fim de facilitar e paralelizar seu processamento.

Nessa operação de fragmentação da imagem em diversos pedaços de tamanho predefinido, o usuário da rede precisa definir, por meio dos Hiperparâmetros, o tamanho dos pedaços, no inglês chamados de *Patches*, nos quais a imagem original será subdividida, o que influenciará diretamente na quantidade de pedaços subdivididos. Esse processo é chamado de *Image Patching* e é ilustrado na Figura 3.1.1.2.



Em seguida, de forma similar ao processamento de palavras, cada um dos *Patches* precisa passar pelo processo de transformação a fim de se adequar e ser legível a nível da rede neural. A transformação de imagens, mais comum no ramo da Inteligência Computacional, consiste da conversão para uma matriz com valores que corresponder os Valores RGB da imagem. De tal maneira, esse processo chamado de *Linear Projection*, resulta em uma representação numérica da imagem de entrada que é de fácil interpretação para o modelo.

Além de uma representação numérica, é de suma importância que seja incorporada a ela a informação de posição do *Patch* em relação aos outros. Pois se a imagem de entrada original for "montada" de forma incorreta a imagem deixa possuir sentido e se torna incoerente para a proposta do modelo. Esse mapeamento acontece em sequência do processo de conversão da mesma forma como é executado no modelo de processamento de texto, onde a informação de posicionamento é introduzida em conjunto a representação numérica da entrada. Dessa forma, devido essa etapa conhecida como *Positional Encoding*, o modelo tem referência de localização de cada um dos *Patches* originados da imagem de entrada, permitindo a coesão do processamento mesmo com a fragmentação da informação.

Para complementar a representação da imagem de entrada antes que ela seja de fato processada pelo modelo ViT, é associada a ela um elemento fundamental para classificação final da imagem. Uma tática desenvolvida e implementada originalmente no modelo BERT (DEVLIN et al., 2019), trata-se da utilização de um token especial chamado de CLS, esse token tem como propósito auxiliar e carregar consigo a informação que diz respeito a classificação da entrada, que no caso do modelo ViT, representa a classificação final da imagem. Esse token assume as mesmas características da representação de um *Patch*, isso acontece para que esse token de classificação consiga ser processado simultaneamente com os outros *Patches* da imagem original e possibilitando assim a classificação acurada da entrada. Finalmente, os dados estão devidamente preparados e aptos a serem processados pelo módulo Encoder que sucede as etapas de Linear Projection, Positional Encoding e Class Learning Token.

É necessário destacar que as imagens de entrada do modelo RT-PI serão devidamente detalhadas na Seção 4.2, de forma a elucidar a natureza de suas características ao passo de acesso ao modelo.

### 3.1.2 Processamento do Encoder RT-PLN e RT-PI

Com os dados de entrada devidamente formatados para o processamento da rede, não se faz necessário nenhum tipo de alteração na estrutura do módulo Encoder. Dessa forma, tanto o modelo RT-PLN quanto o RT-PI utilizam a mesma arquitetura do módulo Encoder, não se fazendo necessária a divisão em subseções dististas conforme feito com a entrada. Na prática, ao ingressar no módulo Encoder, a entrada é distribuída em três Fully Connected Layers distintas a fim de produzir os vetores que terão um papel fundamental para o cálculo da similaridade das palavras, são eles: *Query*, *Key* e *Value*. Num primeiro momento, os vetores *Query* e *Key* passam por uma operação de multiplicação de matriz de produto escalar, obtendo assim uma *Score Matrix*, i.e. uma matriz com valores que representam quanto foco cada uma palavra deve ter sobre as outras, dessa forma cada palavra apresentará um valor de correlação sobre as demais. Essa matriz passa então por uma etapa de dimensionalização, que auxiliará no controle aritmético dos valores, para que então seja aplicada uma função Softmax, que irá redistribuir os valores existentes para valores entre 0 e 1 de acordo com sua porcentagem relativa. No final desse processo, tem-se uma matriz com valores devidamente calculados que irão representar, para todas as palavras da entrada, quanta relevância ela deve ter para com as demais. Em seguida, essa matriz de atenção passa por outra operação de multiplicação, só que dessa vez com o vetor *Value*, e tem como resultado um vetor onde o valores acentuados da matriz vão fazer com que suas palavras correspondentes se sobressaiam em relação a palavras cujo valor é próximo de zero. Dessa forma o modelo aprende quais palavras e correlações são importantes. Por fim, esse novo vetor passa por uma Fully Connected Layer para que então essas informações possam ser processadas e o modelo consiga de fato aprender tal *"conhecimento"* produzido, constituindo assim a estrutura de uma instância chamada *Attention Head*, ilustrada na Figura 11.

Uma característica importante para a arquitetura Transformer, e essencial para a eficácia desse modelo, é o fato do procedimento de cálculo de coeficientes de atenção realizado por uma Attention Head poder ser feito inúmeras vezes de forma paralelizada, formando assim o submódulo chave chamado *Multi-Headed Attention*. Dessa forma, cada Attention Head produz um vetor diferente que por fim serão concatenados e processados pela Fully Connected Layer que as sucedem.

O vetor de saída do módulo Multi-Headed Attention é então acrescido ao vetor original, i.e. aquele livre de alterações provocadas pelo módulo Multi-Headed Attention, que então passa pela operação de normalização, para novamente tratar possíveis valores irregulares presentes no vetor. Logo após, o vetor será processado pelo segundo submódulo do módulo

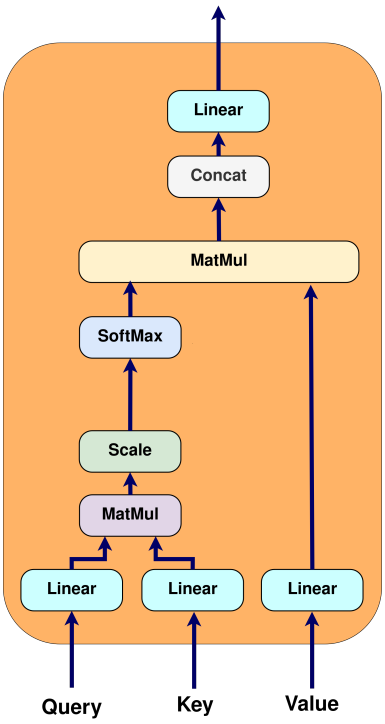



Figura 11 – Arquitetura de uma Attention Head

Encoder, que consiste basicamente de um bloco de Fully Connected Layers, seguidos da concatenação com o vetor pré-processamento e normalização, processo similar ao que acontece na saída do submódulo Multi-Headed Attention.

### 3.1.3 Saída do Encoder

#### 3.1.3.1 Saída do Encoder da RT-PLN

No modelo RT-PLN, como o propósito da rede é processamento de linguagem natural, ou seja, produzir uma saída textual coerente com a que foi dada na entrada da rede, as relações de similaridade entre as palavras produzidas pelo módulo Encoder vão alimentar e servir como base para a geração de texto no módulo Decoder.

#### 3.1.3.2 Saídas do Encoder e do PMC da RT-PI

Conforme mostrado na Figura 10, as redes RT-PI são consitituídas do módulo Encoder, seguido de um PMC (*Prediction Head*. O papel do Encoder é receber a imagem original e produzir automaticamente, em sua saída, uma representação sintética e precisa da imagem recebida (ou seja, uma representação onde são mantidas apenas as principais características da imagem, sem que isso comprometa sua identificação). O papel do PMC é receber essa representação produzida pelo Encoder é produzir, na saída, um vetor de predições de pertinência a uma dada classe. Mais especificamente, a dimensão de tal vetor corresponde ao número de classes do problema tratado, enquanto que cada posição dele

armazena o valor da probabilidade (entre 0 e 1) de a imagem apresentada na entrada do Encoder pertencer à classe representada por aquela posição.

## 3.2 Decoder

O módulo Decoder corresponde ao componente autoregressivo da rede, recebe informações tanto da entrada do modelo quanto da saída produzida módulo Encoder, para então processar tais dados e com eles decodificar e produzir uma resposta condizente com a entrada e a proposta da rede. Por exemplo, no contexto de um Transformer de tradução da língua portuguesa para língua inglesa, caso a entrada seja *"Eu gosto de viajar"* a rede deve ser capaz de processar seus conhecimentos e decodificar uma mensagem que faça sentido como resposta da entrada, tal como *I like to travel*. A composição do módulo Decoder contém elementos muito similares aos presentes no módulo Encoder, porém, para que a decodificação da saída seja feita com precisão, o módulo conta com algumas modificações específicas para atender ao objetivo de geração de saída. A composição do módulo Decoder, assim como a de seus submódulos, serão devidamente evidenciados nas subseções a seguir.

Vale salientar que, como o propósito do módulo Decoder é produzir uma saída baseada na interrelação das entradas construída pelo módulo Encoder, o modelo RT-PI não faz uso do módulo Decoder por justamente não necessitar uma elaboração, e de fato geração, de saída. No contexto do RT-PI as relações produzidas pelo módulo Encoder, que enventualmente serão processadas um PMC, são suficientes para construir uma resposta para o problema de Processamento de Imagem.

### 3.2.1 Entrada do Decoder

O acesso ao módulo Decoder acontece tanto pela inserção direta de um dado como entrada da rede, quanto pela etapa seguinte da saída do módulo Encoder. No primeiro caso, de forma análoga a entrada de um novo dado no módulo Encoder, a etapa de pré-acesso ao módulo Decoder também conta com operações de Word Embeding, seguidas da aplicação de uma operação de Positional Encoding. Porém neste caso, por se tratar do módulo que irá produzir a saída da rede, esse processo conversão da entrada para uma estrutura válida para a rede recebe o nome de *Output Embeding*.

### 3.2.2 Processamento do Decoder

O primeiro sub-módulo do módulo Decoder se trata do módulo *Multi-Headed Attention* já conhecido, porém com uma pequena diferença em relação ao existente no módulo Encoder. No processo de geração de resposta, seja para tradução de texto ou geração de linguagem, é necessário respeitar a ordem em que as palavram estão dispostas. Dessa

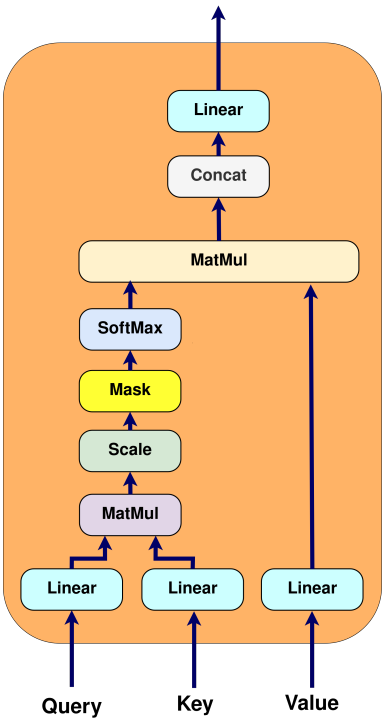



Figura 12 – Arquitetura do Módulo Masked Multi-Headed Attention

| | | | |
|---|---|---|---|
| 0,7 | 0,1 | 0,1 | 0,1 |
| 0,1 | 0,6 | 0,2 | 0,1 |
| 0,1 | 0,3 | 0,6 | 0,1 |
| 0,1 | 0,3 | 0,3 | 0,3 |

+

| | | | |
|---|---|---|---|
| 0 | - inf | - inf | - inf |
| 0 | 0 | - inf | - inf |
| 0 | 0 | 0 | - inf |
| 0 | 0 | 0 | 0 |

=

| | | | |
|---|---|---|---|
| 0,7 | - inf | - inf | - inf |
| 0,1 | 0,6 | - inf | - inf |
| 0,1 | 0,3 | 0,6 | - inf |
| 0,1 | 0,3 | 0,3 | 0,3 |

Figura 13 – Aplicação da Máscara sobre Matriz de Valores de Ligação

forma, ao se criar pesos para as relações da linguagem de saída, diferente do módulo Encoder, uma determinada palavra não pode ter relação com suas subsequentes pelo simples fato delas ainda não “existirem". Para se implementar tal estratégia, o módulo Multi-Headed Attention conta com a aplicação de uma *“Máscara”* que irá anular todos os valores acima da diagonal principal da *Score Matrix*, conforme ilustrado pela Figura 13, desconsiderando assim a relações com palavras futuras. Similar ao módulo Encoder, pode-se paralelizar quantas execuções for necessário do sub-módulo agora chamado de *Masked Multi-Headed Attention*, ilustrado pela Figura 12, com as mesmas finalidades do Multi-Headed Attention normal.

Com a Matriz de valores que delimita quais relações entre palavras podem ser feitas, o segundo sub-módulo Multi-Headed Attention exerce a função de conectar os dois módulos primordiais do arquitetura Transformer a fim de relacionar a entrada com as relações construídas pelo módulo Encoder. Essa convergência de dados contece da seguinte forma,

o vetores Query e Key tem como origem a saida elaborada pelo módulo Encoder, enquanto o vetor Value provém da Masked Multi-Headed Attention. Nesse passo, as relações produzidas pelo módulo Enconder são aplicadas à entrada do módulo Decoder, de forma a efetivamente produzir um resposta válida correspondente a entrada inserida na rede.

Em seguida, a saída produzida pelo último módulo Multi-Headed Attention passa por um bloco de Fully Connected Layers, onde seu processamento será realizado e seus pesos serão calculados e ajustados de acordo com a resposta produzida.

### 3.2.3 Saída do Decoder

Por fim, a etapa final do processamento de uma arquitetura Transformer recai sobre uma camada Linear que é responsável por fazer a classificação da resposta, similar ao processo que acontece em modelos de redes neurais já consolidadas, onde uma operação de SoftMax será aplicada ao vetor final, que contém as palavras que serão utilizadas para formar a saída da rede, e assim o modelo saber quais dessas palavras tem a maior correspondência com a entrada da rede. A título de exemplo, fazendo uso do modelo Transformer de tradução de linguagem, é nessa etapa onde são escolhidas as palavras da linguagem de entrada do módulo Decoder que fazem corretamente a tradução das palavras de entrada do módulo Encoder. Tendo assim ao final do processo, a tradução a frase para além do calculo de relações entre tais palavras que permanecerá conhecido pela rede neural.

# Capítulo 4

# Experimentos e Resultados

O modelo RT-PI implementado no presente trabalho foi treinado utilizando uma abordagem padrão de aprendizado sob treinamento supervisionado, característico de trabalhos de classificação de imagens. De forma que o Dataset utilizado no treinamento contenha não apenas as imagens de entrada, mas também um rótulo indicando a qual classe tal imagem pertence, para que dessa forma seja possível calcular métricas essenciais que indicarão a eficácia e o bom desempenho da rede. Além disso, os parâmetros que compõem uma RNA são outro fator fundamental no processo de treinamento, pois quando são definidos de forma planejada para melhor atender o propósito do problema, são responsáveis por atuar diretamente no bom desempenho da rede.

A implementação do modelo Vision Transformer foi feita através da linguagem de programação Python (versão 3.10.12), e conta com a utilização das bibliotecas: Keras, Numpy e Tensorflow. A máquina utilizada para realização dos experimentos do atual trabalho corresponde à um notebook Samsung, processador Intel I5 7ª geração, 1 TB de Memória de Disco e 8 Gb de memóriam RAM.

Por fim, as imagens utilizadas para o treinamento e teste da RT-PI desenvolvida provém de uma dataset público, disponibilizado em (KRIZHEVSKY, 2009), que é amplamente utilizado em diversos trabalhos relacionados a Aprendizado de Máquina e Processamento de Imagem. Sua popularidade se dá pelo fato de ser um Dataset de média escala, de bom balanceamento e com imagens de classes distintas muito bem preparadas e balanceadas para o processamento em uma RNA.

Este capítulo será dividido em três seções distintas, onde cada uma irá abordar de forma detalhada todas as características que compõem cada etapa do processo de treinamento de uma rede neural. Dessa forma, as seções a seguir tem como objetivo: descrever os parâmetros de configuração da rede (Seção 4.1); elucidar o processo e técnicas utilizadas para o treinamento da rede, além detalhar de forma mais profunda o dataset utilizado (Seção 4.2); e por fim apresentar métricas que representem o resultado a cerca do treinamento realizado e do estudo desenvolvido no presente trabalho (Seção 4.3).

## 4.1 Parâmetros dos Experimentos

Além da arquitetura do modelo, uma RNA conta com diversos parâmetros de configuração que precisam ser definidos pelo usuário da rede antes mesmo de seu funcionamento. Esses parâmetros, como por exemplo *Taxa de Aprendizado*(Learning Rate), *Taxa de Evasão*(Dropout Rate) e *Épocas*(Epocs), ditam comportamentos indispensáveis do treinamento da rede. A definição adequada desses parâmetros colabora não só para a excelente performance da rede, mas também inibem gargalos de treinamento que poderiam provocar a estagnação da eficácia do modelo.

O modelo aqui implementado reproduz aquele proposto no RT-PI original (3). Assim sendo, foram usados aqui os mesmos valores de parâmetros e a mesma configuração usada no trabalho original, tal como, por exemplo, o número de submódulos *MultiHeaded Attention* paralelizados.

Os parâmetros de configuração da RT-PI são apresentados na Tabela 1 e devidamente descritos abaixo:

- *Learning Rate*: Hiperparâmetro usado para controlar o ritmo no qual um algoritmo atualiza ou aprende os valores de uma estimativa de parâmetro presente na arquitetura do modelo.
- *Weight Decay*: Hiperpâmetro usado para causar um desconto no valor dos pesos do modelo a fim de controlar e eventualmente inibir prejuízos causados por overfitting.
- *Epocs*: Hiperparâmetro que representa quantas iterações de treinamento tal rede terá, ou seja, quantas vezes o modelo passa pelo processo de receber uma, ou mais, instâcias de entrada para então processa-la em sua arquitetura.
- *Image Size*: Dimensão da Imagem de entrada do modelo, dada em pixels.
- *Batch Size*: Representa a quantidade de Imagens de entrada que serão dadas como entrada na RNAs ao passo de uma interação de treinamento.
- *Patch Size*: Dimensão de um pedação da imagem de entrada produzido durante o processo de Image Patching.
- *MultiHeaded Attention Heads*: Quantidade de módulos MultiHeaded Attention paralelizados na composição do módulo Encoder da arquitetura.
- *Transformer Layers*: Quantidade de camadas, ou seja, repetições aninhadas do módulo Encoder, presentes na implementação do modelo.

Durante os experimentos realizados, todos os parâmetros de configuração da RT-PI, com excessão do parâmetro Epocs, se manteram livres de alteração. Isso se deve pela falta de tempo hábil para realização de experimentos mais diversos, além do poder computacional ser limitado, o que torna o processo de treinamento não tão otimizado.

Tabela 1 – Configuração dos hiperparâmetros utilizados na implementação do Vision Transformer

| Hiperparâmetros | | |
|---|---|---|
| (1) | Learning Rate | 0.0001 |
| (2) | Weight Decay | 0.0001 |
| (3) | Epocs | 20/30/50/100 |
| (4) | Image Size size | 72 |
| (5) | Batch Size | 128 |
| (6) | Patch Size | 6 |
| (7) | MultiHeaded Attention Heads | 4 |
| (8) | Transformer Layers | 8 |

## 4.2 Treinamento da Rede

Além da definição dos Hiperparâmetros da rede, o próprio Dataset utilizado no treinamento é um fator de suma relevância que impacta diretamente o desempenho de uma RNA. De forma que, se o banco de dados não possui imagens de qualidade, devidamente rotuladas, preparadas e igualmente balanceadas no dataset como um todo, o treinamento do modelo será prejudicado e produzirá um resultado inconsistente e de baixa qualidade.

Conforme mencionado no cabeçalho do Capítulo 3, o Dataset CIFAR-100 (KRIZHEVSKY, 2009), utilizado no atual trabalho, é composto por 60.000 imagens, de dimensão 32x32 pixels, igualmente divididas entre 100 classes previamente modeladas. Dessa forma, as 60.000 imagens são repartidas em dois grupos distintos, onde um deles é composto por 50.000 imagens devidamente balanceadas e tem como objetivo serem utilizadas no processo de treinamento da rede, enquanto o outro grupo, composto pelas outras 10.000 imagens restantes, é aplicado a operações de teste e avaliação do modelo.

Cada imagem desse Dataset é, por natureza, um frame que comporta um único objeto, ou seja, na imagem está representado apenas o objeto cujo seu rótulo identifica. No caso do Dataset CIFAR-100, conforme mencionado no parágrafo anterior, as imagens estão divididas em 100 rótulos (classes) diferentes, de forma a compreender os mais diversos objetos presentes no mundo real, como por exemplo: “urso”, “abelha”, “maçã”, “bicicleta”, etc..

### 4.2.1 Pré Treinamento

Além da organização e preparação de um bom Dataset, existem outras operações que podem ser aplicadas numa etapa que antecede o treinamento da rede. Tais operações colaboram ainda mais para a otimização do processo de treinamento, tanto no quesito de tempo de execução quanto em evolução das métricas de avaliação. Dessa forma, o processo de treinamento resulta em uma RNA ainda mais apurada, que devido seu processo de treinamento aprimorado, irá, por fim, contar com métricas de avaliação ainda

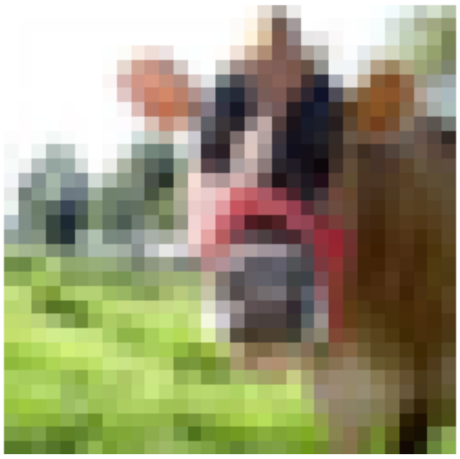


Figura 14 – Imagem Original

Figura 15 – Imagem pós Data Augmentation

Figura 16 – Aplicação da técnica Data Augmentation

mais desenvolvidas. Com esse propósito, o atual trabalho, além de desenvolver a implementação do modelo RT-PI, também realiza a técnica de *Data Augmentation* no dados de entrada do modelo, que consiste na aplicação de diversas operações de manipulação das imagens de entrada, a fim de diversificar ainda mais a ampla gama de imagens existentes no Dataset utilizado. Ou seja, no modelo desenvolvido pelo atual trabalho, antes mesmo das imagens do dataset serem submetidas ao treinamento da rede, elas passam pela seguinte sequência de operações: Normalização, redimensionamento, rotação 180º no sentido horizontal, rotação circular e por fim ampliação. Dessa maneira, o dataset de treinamento conta com entradas ainda mais distintas e diversificadas quando comparado com o dataset originalmente disponibilizado, o que colabora significativamente para um treinamento mais profundo e completo da rede. Com isso, o resultado final do processo de treinamento trata-se de uma rede mais robusta e consolidada, com uma eficácia consideravelmente aprimorada devido seu extenso processo de treinamento.

### 4.2.2 Treinamento

No processo de treinamento do modelo RT-PI, a cada iteração de treinamento, um conjunto de dados, chamado de *Batch*, é dado como instância de entrada da rede. Tal conjunto é então processado e sobre ele calculado métricas parciais a respeito do desempenho da rede para aquela determinada entrada. Dessa forma, com o passar das iterações e o processamento de novas instâncias de entrada, a rede seja capaz de melhor ajustar seus pesos, de forma progressiva, para que ao fim do treinamento a acurácia, métrica utilizada para medir a média de acerto da rede, seja a maior possível.

Foram performados 4 experimentos distintos no atual trabalho, onde a composição da arquitetura e os parâmetros de configuração permanecessem os mesmos, variando apenas o número de *Épocas* de treinamento do modelo. Dessa forma, será possível observar a

Tabela 2 – Experimentos desenvolvidos

| Experimentos | |
|---|---|
| Experimento | Épocas |
| 1 | 20 |
| 2 | 30 |
| 3 | 50 |
| 4 | 100 |

relevância desse parâmetro para o bom desempenho do modelo, além de evidenciar a evolução da métrica de avaliação da rede dada a profundidade do seu treinamento. A configuração dos Hiperparâmetros da rede, atrelada a composição de sua arquitetura, obtiveram como tempo médio computacional gasto para o processamento de 1 Época o valor de, aproximadamente, 30 minutos.

A Tabela 2 a seguir enumera os experimentos realizados conforme a quantidade de Épocas utilizadas no treinamento.

## 4.3 Resultados

Nesta seção serão apresentados os resultados relativos aos experimentos realizados no atual trabalho, além de evidenciar o uso da métrica *Acurácia* e apontar algumas considerações em torno dos experimentos realizados.

A métrica Acurácia é calculada com base no número de imagens corretamente classificadas em relação ao número total de imagens utilizadas no treinamento, obtendo assim uma porcentagem de acerto médio da rede. A utilização dessa métrica é comum em trabalhos de classificação de imagens, na maioria dos casos ela consegue avaliar de forma concisa a performance do modelo. Porém, é necessário extrema cautela com a utilização dessa métrica nos casos onde o Dataset de treinamento não está devidamente balanceado, nesses casos a acurácia pode mascarar o real desempenho da rede. A título de exemplo, em um Dataset com 80% de suas imagens pertencentes a classe A e apenas 20% a classe B, uma rede ingênua, que consegue classificar corretamente apenas images da classe A, vai obter um alto valor de acurácia mesmo sendo incapaz de classificar imagens pertencentes a classe B. Em tais casos, outras métricas, como por exemplo precisão e F1, são mais recomendadas para avaliar a performace do modelo.

Portanto, pelo fato do Dataset (KRIZHEVSKY, 2009) ser um banco de dados devidamente balanceado e já consolidado no âmbito de Processamento de Imagem, a métrica Acurácia se torna adequada para avaliar o desempenho do modelo RT-PI desenvolvido. A Tabela 3 apresenta os resultados obtidos pelos experimentos realizados, expressando não apenas o valor de acurácia geral do experimento, mas também o valor médio das 5 melhores acurácias obtidas durante o processo de treinamento do experimento.

Tabela 3 – Resultados dos experimentos

| Experimento | Acurácia Total | Top 5 Acurácia |
|---|---|---|
| 1 | 43.21% | 73.91% |
| 2 | 46.91% | 76.10% |
| 3 | 50.23% | 78.01% |
| 4 | 52.63% | 79.77% |

Os resultados dos experimentos indicam que o modelo RT-PI desenvolvido não alcançou um valor elevado de acurácia, apesar das técnicas aplicadas e do intenso treinamento, dado ao poder computacional existente, o modelo demonstrou dificuldade em aprimorar seu desempenho a partir de um determinado momento do processo de treinamento. Indicando assim, uma possível estagnação do modelo, que pode ser observada nos últimos estágios do treinamento, onde o progresso do modelo se torna insignificante ao passo das subsequentes iterações de treinamento.

O modelo RT-PI desenvolvido, originado do modelo RT-PLN proposto no artigo "Attention is all you need" (VASWANI et al., 2017), utiliza diversas operações de escalonamento para manter controle do gradiente de seus parâmentros, evitando assim os clássicos problemas a cerca de gradiente existentes nos modelos mais ingênuos de RNAs.

As causas estagnação dos experimentos podem estar atreladas a diversos fatores, alguns já abordados no presente trabalho, como por exemplo a definição não otimizada dos parâmetros de configuração da rede, poder computacional incompatível com o modelo ou até mesmo a ação das conexões residuais presentes na arquitetura do modelo.

(LIU et al., 2020) realiza uma pesquisa que busca compreender os principais gargalos existentes no processo de treinamento da Rede Neural Transformer, estudando condutas teóricas e empíricas relacionadas ao modelo, que evidenciam a forte dependência das conexões residuais como principal causador da instabilidade do processo de treinamento, o que potencializa a variação dos parâmetros da rede. Porém, o mesmo autor, propõe uma possível solução para lidar com tais desafios, trata-se de um método de inicialização adaptivo de parâmetros, que prevê o controle dos parâmetros fundamentais de treinamento do modelo até que ele se estabilize.

Já (ZHUANG et al., 2023) reforça a importância da inicialização apropriada dos parâmetros associada da adoção de paradigmas de otmização que busquem acelerar a taxa de convergência do modelo, além apontar a utilização de hardware apropriado em conjunto de técnicas de uso eficiente de memória como abordagens de significativa relevância para o bom desempenho da rede neural.

A carência na evolução do desempenho do modelo evidencia o desafio de treinar RNAs para performarem tarefas complexas, tal como processamento e classificação de imagens.

Os resultados elucidam a necessidade de um estudo mais detalhado a respeito, tanto do modelo desenvolvido quanto das técnicas de refinamento aplicadas, para então alcançar níveis mais satisfatórios de acurácia e desempenho geral da rede.

## 4.4 Conclusão e Trabalhos Futuros

### 4.4.1 Conclusão

O estudo realizado no presente trabalho forneceu informações valiosas a respeito do ramo de Aprendizado de Máquina, bem como as características fundamentais e pontos positivos e negativos de diversos modelos de RNAs, com foco especial no objeto de estudo do trabalho, ou seja, a arquitetura Transformer para Processamento de Imagens.

A estagnação do aprendizado do modelo e os baixos níveis de acurácia obtidos nos experimentos evidenciam uma possível falha na definição dos parâmtros de configuração da rede, ou talvez a técnica de Data Augmentation aplicada não foi suficiente para colaborar de forma efetiva com a conversão da rede. Ainda que o modelo Transformer seja amplamente reconhecido por desempenhar bem em diversos problemas do mundo atual, seu processo de treinamento é intenso e nada é trivial, requer um alto poder computacional e demanda a utilização de diversas técnicas de preparação de dados e configuração de parâmetros, além de outros procedimentos que auxiliarão no processo de conversão do modelo.

Apesar dos desafios a cerca do modelo Transformer, os resultados desse trabalho comprovam a importância do modelo no avanço no campo da Inteligência Artificial, de modo a contribuir com sistemas cada vez mais robustos e eficientes, que por sua vez irão beneficiar a população em diversos aspectos. O conhecimento obtido ao desenvolver desse trabalho será utilizado de base para futuras pesquisas e eventual aprimoramento do modelo desenvolvido.

### 4.4.2 Trabalhos Futuros

Para trabalhos futuros, será realizado um estudo mais aprofundado do modelo Vision Transformer, além da exploração de novas técnicas e refinamento das já aplicadas no atual trabalho, de forma a aprimorar o desempenho do modelo RT-PI desenvolvido na execução da tarefa de processamento e classificação de imagens. Cientes dos principais desafios e gargalos do processo de treinamento de uma rede neural Transformer, propõe-se a resolução deles por meio da exploração e aplicação das técnicas citadas brevemente na Seção 4.3, em conjunto com a exploração de novas abordagens, para que então se obtenha um nível satisfatório das métricas de avaliação de desempenho.

Além disso, pretende-se aplicar o modelo RT-PI como ferramenta de classificação de eventos e ações no jogo Super Mario Bros, tarefa essa que o autor, em conjunto com

sua orientadora e os demais integrantes do grupo de estudos BladeRunner, desempenha a um certo tempo. Espera-se que a arquitetura Transformer traga ganhos significativos de desempenho quando comparado aos trabalhos já realizados ( (FARIA et al., 2022b), (JULIA, 2022) e (FARIA et al., 2022a)), colaborando assim para o propósito maior do projeto de aplicar Aprendizado Profundo no desenvolvimento de habilidades cognitivas.

# Referências

BERNER, C. et al. Dota 2 with large scale deep reinforcement learning. **ArXiv**, abs/1912.06680, 2019. Disponível em: <https://api.semanticscholar.org/CorpusID:209376771>.

CHEN, H. et al. Pre-trained image processing transformer. In: **2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)**. Los Alamitos, CA, USA: IEEE Computer Society, 2021. p. 12294–12305. Disponível em: <https://doi.ieeecomputersociety.org/10.1109/CVPR46437.2021.01212>.

DEVLIN, J. et al. Bert: Pre-training of deep bidirectional transformers for language understanding. In: **North American Chapter of the Association for Computational Linguistics**. [s.n.], 2019. Disponível em: <https://api.semanticscholar.org/CorpusID:52967399>.

DOSOVITSKIY, A. et al. An image is worth 16x16 words: Transformers for image recognition at scale. **CoRR**, abs/2010.11929, 2020. Disponível em: <https://arxiv.org/abs/2010.11929>.

_____. An image is worth 16x16 words: Transformers for image recognition at scale. In: **International Conference on Learning Representations**. [s.n.], 2021. Disponível em: <https://openreview.net/forum?id=YicbFdNTTy>.

FARIA, M. et al. Proposing an automatic system for generating super mario bros datasets. In: **Anais Estendidos do XXI Simpósio Brasileiro de Jogos e Entretenimento Digital**. Porto Alegre, RS, Brasil: SBC, 2022. p. 325–330. ISSN 0000-0000. Disponível em: <https://sol.sbc.org.br/index.php/sbgames_estendido/article/view/23668>.

FARIA, M. P. P. et al. Investigating the performance of various deep neural networks-based approaches designed to identify game events in gameplay footage. **Proc. ACM Comput. Graph. Interact. Tech.**, Association for Computing Machinery, New York, NY, USA, v. 5, n. 1, may 2022. Disponível em: <https://doi.org/10.1145/3522624>.

GOLDEN, J. A. Deep Learning Algorithms for Detection of Lymph Node Metastases From Breast Cancer: Helping Artificial Intelligence Be Seen. **JAMA**, v. 318, n. 22, p. 2184–2186, 12 2017. ISSN 0098-7484. Disponível em: <https://doi.org/10.1001/jama.2017.14580>.

GOOGLE. **Gemini**. 2024. <https://gemini.google.com/>.

HE, K. et al. Deep residual learning for image recognition. In: **2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2016. p. 770–778.

HOWARD, A. G. et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications. **ArXiv**, abs/1704.04861, 2017. Disponível em: <https://api.semanticscholar.org/CorpusID:12670695>.

JULIA, E. **A deep learning system to perform multi-instance multi-Label event classification in video game footage.** 2022. Disponível em: <https://doi.org/10.14393/ufu.di.2022.562>.

KOKAB, S.; ASGHAR, S.; NAZ, S. Transformer-based deep learning models for the sentiment analysis of social media data. **Array**, v. 14, p. 100157, 04 2022.

KRIZHEVSKY, A. **CIFAR-100 Official Website**. 2009. <https://www.cs.toronto.edu/~kriz/cifar.html>. Accessed: 2023-04-20.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: PEREIRA, F. et al. (Ed.). **Advances in Neural Information Processing Systems**. Curran Associates, Inc., 2012. v. 25. Disponível em: <https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf>.

LAKEW, S. M.; CETTOLO, M.; FEDERICO, M. A comparison of transformer and recurrent neural networks on multilingual neural machine translation. **arXiv preprint arXiv:1806.06957**, 2018.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **nature**, Nature Publishing Group, v. 521, n. 7553, p. 436, 2015.

LECUN, Y. et al. Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, v. 86, p. 2278 – 2324, 12 1998.

LIU, L. et al. Understanding the difficulty of training transformers. In: **Conference on Empirical Methods in Natural Language Processing**. [s.n.], 2020. Disponível em: <https://api.semanticscholar.org/CorpusID:215814515>.

LU, M. et al. Transformer-based image compression. In: **2022 Data Compression Conference (DCC)**. [S.l.: s.n.], 2022. p. 469–469.

META. **LLaMA**. 2024. <https://llama.meta.com/>.

MORDVINTSEV, A. **Deep Dream**. 2024. <https://deepdreamgenerator.com/>.

NETO, H.; JULIA, R. Ls-draughts: Um sistema de aprendizagem para damas com geração automática de características. **VIII Congresso Brasileiro de Redes Neurais (CBRN)**, p. 1–6, 04 2016.

NORVIG, P.; RUSSELL, S. **Inteligência artificial: Tradução da 3a Edição**. Elsevier Brasil, 2014. ISBN 9788535251418. Disponível em: <https://books.google.com.br/books?id=BsNeAwAAQBAJ>.

OPENAI. **ChatGPT**. 2024. <https://openai.com>.

_____. **DALL-E2**. 2024. <https://openai.com/dall-e-2>.

RAFFEL, C. et al. **Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer**. 2023.

RUSSAKOVSKY, O. et al. Imagenet large scale visual recognition challenge. **International Journal of Computer Vision**, v. 115, n. 3, p. 211–252, Dec 2015. ISSN 1573-1405. Disponível em: <https://doi.org/10.1007/s11263-015-0816-y>.

SILVER, D. et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. **arXiv preprint arXiv:1712.01815**, 2017.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. **CoRR**, abs/1409.1556, 2014. Disponível em: <https://api.semanticscholar.org/CorpusID:14124313>.

VASWANI, A. et al. Attention is all you need. In: GUYON, I. et al. (Ed.). **Advances in Neural Information Processing Systems**. Curran Associates, Inc., 2017. v. 30. Disponível em: <https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf>.

VINCENT, D. R. et al. Sensors driven ai-based agriculture recommendation model for assessing land suitability. **Sensors**, v. 19, n. 17, 2019. ISSN 1424-8220. Disponível em: <https://www.mdpi.com/1424-8220/19/17/3667>.

ZHUANG, B. et al. A survey on efficient training of transformers. In: **Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence**. [s.n.], 2023. (IJCAI '23). ISBN 978-1-956792-03-4. Disponível em: <https://doi.org/10.24963/ijcai.2023/764>.