UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

LUCAS GUSTAVO GOLFETTO LUPSCHINSKI

DETECÇÃO DE AÇÕES DE VIOLÊNCIA EM VÍDEOS POR MEIO DE REDES NEURAIS ARTIFICIAIS

MEDIANEIRA

2023

LUCAS GUSTAVO GOLFETTO LUPSCHINSKI

DETECÇÃO DE AÇÕES DE VIOLÊNCIA EM VÍDEOS POR MEIO DE REDES NEURAIS ARTIFICIAIS

DETECTION OF VIOLENCE ACTIONS IN VIDEOS USING ARTIFICIAL NEURAL NETWORKS

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Ciência da Computação do Curso de Bacharelado em Ciência da Computação da Universidade Tecnológica Federal do Paraná.
Orientador: Prof. Dr. Nelson Miguel Betzek
Coorientador: Prof. Dr. Pedro Luiz De Paula Filho

MEDIANEIRA

2023

LUCAS GUSTAVO GOLFETTO LUPSCHINSKI

DETECÇÃO DE AÇÕES DE VIOLÊNCIA EM VÍDEOS POR MEIO DE REDES NEURAIS ARTIFICIAIS

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Ciência da Computação do Curso de Bacharelado em Ciência da Computação da Universidade Tecnológica Federal do Paraná.

Data de aprovação: 20/junho/2023

_____

Nelson Miguel Betzek
Doutorado
Universidade Tecnológica Federal do Paraná – Campus Medianeira

_____

Hamilton da Silva Pereira
Mestrado
Universidade Tecnológica Federal do Paraná – Campus Medianeira

_____

Juliano Rodrigo Lamb
Doutorado
Universidade Tecnológica Federal do Paraná – Campus Medianeira

MEDIANEIRA

2023

RESUMO

O número de ações consideradas violentas nos últimos anos vem crescendo de forma exponencial. Nas situações mais graves, o uso de algum artefato para ameaçar ainda mais a vítima é utilizado, dificultando alguém conseguir sair ileso da situação. Para diminuir esse problema, espaços públicos e privados optam por aplicar ações que ajudam a promover a segurança, entre eles está a vigilância remota por meio de câmeras estrategicamente posicionadas. Levando em consideração que apenas as câmeras por si não conseguem ser efetivas, métodos para apoiar a vigilância remota são necessários, entre eles consta o objetivo do trabalho, que é a aplicação de uma rede neural artificial (ou RNA), com o intuito de auxiliar agentes de segurança, estes que operam sistemas de vigilância, tanto internos quanto externos. De forma geral, a rede neural deve reconhecer o contato violento entre indivíduos. Essa detecção tem como objetivo de diminuir o tempo de resposta que um profissional teria para se deslocar até o acontecido. Para alcançar resultado proposto, foi realizada a comparação entre duas abordagens – processando o vídeo a partir de uma rede neural que analisa o vídeo de quadro a quadro, e também utilizando uma mescla entre os conceitos de redes neurais convolucionais e a arquitetura de memória de longo prazo, conhecida como LSTM (ou *Long Short-Term Memory*, em inglês), que desempenha um papel crucial no processamento de informações temporais. Os resultados obtidos pela rede neural quadro a quadro forneceu uma taxa de acerto de 87,61% na detecção de ações violentas, enquanto a rede convolucional junto com a arquitetura LSTM retornou 90,02%, validando as duas abordagens.

Palavras-Chave: inteligência artificial; sistemas de segurança; violência urbana.

ABSTRACT

The number of actions considered violent in recent years has grown exponentially. In the most serious situations, the use of some artifact to further threaten the victim is used, making it difficult for someone to get out of the situation unharmed. Thinking about reducing this problem, public and private spaces choose to apply actions that help to promote security, among them is remote surveillance through strategically placed cameras. Bearing in mind that only cameras alone cannot be effective, methods to support remote surveillance are necessary, among them the objective of the work, which is the application of an artificial neural network (or ANN), in order to help security agents, those who operate surveillance systems, both internal and external. In general, the neural network must recognize violent contact between individuals. This detection aims to reduce the response time that a professional would have to move to the event. To achieve the proposed result, a comparison was made between two approaches - processing the video from a neural network that analyzes the video frame by frame, and also using a mix between the concepts of convolutional neural networks and long-term memory architecture. term, known as LSTM, which plays a crucial role in processing temporal information. The results obtained by the frame-by-frame neural network provided an accuracy rate of 87.61% in the detection of violent actions, while the convolutional network together with the LSTM architecture returned 90.02%, validating both approaches.

Keywords: Artificial intelligence; Security systems; Urban violence.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
ConvLSTM	*Convolutional Long Short-Term Memory*
CPU	*Central Processing Unit*
FAIR	*Facebook's Artificial Inteligence Research*
FPS	*Frames per Second*
GPU	*Graphics Processing Unit*
IA	Inteligência artificial
LSTM	*Long Short-Term Memory*
RGB	*Red Green Blue*
RNA	Rede Neural Artificial
RNC	Rede Neural Convolucional
RNR	Rede Neural Recorrente
ReLU	*Rectified Linear Unit*

SUMÁRIO

1 INTRODUÇÃO

Conforme descrito pelo Escritório das Nações Unidas sobre Drogas e Crime (UNODC), o número de crimes violentos é alto em diversos países da América do Sul, como por exemplo no Brasil, com a marca de 22,5 assassinatos para cada 100.000 habitantes em 2021, número esse que coloca o Brasil entre os 15 países mais violentos do mundo (CERQUEIRA, 2022).

Dessa forma, o uso de câmeras de monitoramento remoto pode ser utilizado, afim de que a equipe responsável pelo monitoramento possa identificar situações suspeitas e realizem ações o mais rápido possível. Entretanto o equipamento acaba sendo adquirido sem o devido suporte ou manutenção, e devido a essas situações, a eficácia do sistema acaba sendo influenciada, por conta de as imagens necessitarem da análise constante de profissionais de segurança. Porém o custo acaba se tornando muito alto, tanto pela jornada de trabalho quanto pela quantidade de seres humanos contratados, que dificilmente será o adequado para a quantidade de câmeras e monitores instalados. Dessa forma, a automatização de tal processo se torna algo de suma importância, afim de diminuir os possíveis erros humanos que inevitavelmente vão acontecer.

O trabalho apresenta uma solução para a detecção de ações de movimento em vídeo, alertando a presença de violência para os profissionais competentes, afim de diminuir o tempo de resposta dos mesmos e aumentar a eficácia na proteção da população. A partir do exposto, o presente trabalho oferece ao leitor uma visão geral sobre conceitos importantes na detecção de ações de movimento, como as redes neurais desempenham o papel principal na análise de dados, assim como quais ferramentas melhor se destaca o objetivo principal do estudo. Da mesma forma, são abordados quais métodos de análise e também quais bases de dados mais se encaixam no treinamento das redes neurais escolhidas – análise de movimento quadro a quadro e detecção por métricas temporais. É discutido também os resultados obtidos e como o trabalho pode ser incrementado para situações específicas no futuro.

1.2 OBJETIVOS

A presente seção apresenta os objetivos, geral e específicos, propostos pelo trabalho.

1.2.1 Objetivo Geral

O trabalho tem como objetivo comparar os resultados entre duas abordagens de detecção de movimento em vídeo, dividindo as ações desempenhadas pelos indivíduos em classes específicas, a fim de reconhecer a ação violenta com o máximo de acurácia permitida.

1.2.2 Objetivos Específicos

- Pesquisar entre a bibliografia disponível métodos de reconhecimento de ações de movimento em vídeos utilizando-se de redes neurais artificiais;
- Obter bases de dados viáveis e condizentes com o escopo do trabalho para o treinamento dos algoritmos;
- Implementar os algoritmos escolhidos (detecção de movimento com uma arquitetura quadro a quadro e identificação de movimento por rede *ConvLSTM*) no ambiente de desenvolvimento para treiná-los em detecção de ação violenta;
- Realizar a comparação de acurácia e perda durante os testes entre as duas abordagens, assim como a análise gráfica e matriz de confusão dos melhores resultados obtidos.

1.3 Justificativa

De acordo com os dados levantados pela UNODC (apud CERQUEIRA, 2022), índice de violência ainda está muito acima do ideal, principalmente nos países da

América do Sul, levantando discussões a respeito de novas e eficientes maneiras para combater tais atos. Sistemas de vigilância, especificamente câmeras de segurança, formam uma maneira de monitorar espaços públicos e privados. Entretanto, sua funcionalidade se limita a capacidade do profissional humano operando esse sistema e, por consequência, enfraquece e cria brechas na detecção de possíveis atos violentos. Portanto, uma forma de automatizar esse processo é extremamente necessário, fornecendo autonomia ao sistema desenvolvido e agilidade ao profissional para se deslocar até o local do acontecido.

O presente trabalho busca realizar a comparação entre duas abordagens utilizando conceitos de aprendizado profundo em redes neurais artificiais existentes, com o objetivo de identificar qual se desempenha melhor na situação de análise de ações de violência em vídeos.

2 REFERENCIAL TEÓRICO

Nesta seção, vários conceitos importantes sobre o tema são descritos, fornecendo explicações e funcionalidades para servir de base literária no estudo.

2.1 Imagem Digital

Conforme descrito por Solomon e Breckon (2011), uma imagem digital pode ser demonstrada como uma representação discreta de diversos dados com informações específicas de identidade e posições espaciais, configurando a disposição das cores e dos dados da imagem. Foi estipulado também pelos autores que, como uma convenção, o início da imagem será no seu canto superior esquerdo, portanto, uma imagem digital bidimensional é o conjunto de informações que um sensor (biológico ou artificial) consegue obter, de acordo com as posições cartesianas da cena. Cada elemento da imagem pode ser representado pela anotação *I(m, n)*, no qual *m* indica a posição da linha e *n* da coluna. Esses elementos que juntos formam uma imagem digital é comumente conhecido como *pixel*.

A palavra *pixel* é derivada do inglês *picture element,* ou em português elemento de imagem. Esse conceito refere a unidade mais baixa de uma imagem digital, composto por um único valor de coloração, seguindo a ordem do tipo de escala utilizada pela imagem, como preto e branco, RGB, escala de cinza, entre outras. Esse valor numérico obtido pela análise do *pixel* é parte essencial no processamento de imagens, pois é possível treinar um programa para comparar se naquela área da imagem pode conter o objeto previamente estipulado, graças as informações inseridas no conjunto de *pixels* que formam tal objeto (SOLOMON *et al.* 2011).

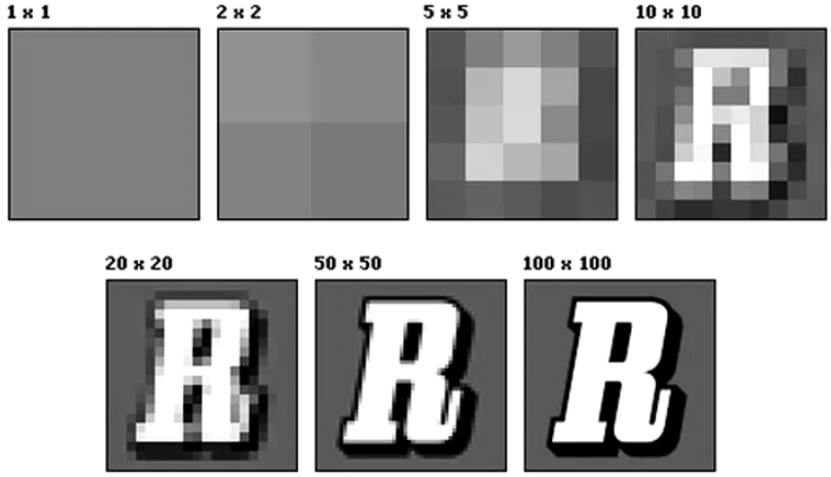
2.2 Resolução das imagens digitais

Uma das formas mais utilizadas para mensurar a resolução de uma imagem é pela quantidade de *pixels* que ela possui e, consequentemente, pelo valor de intensidade que ele apresenta. Essa abordagem de análise é denominada resolução

espacial (PEDRINI; SCHWARTZ, 2007). O formato de apresentação de tal método é fornecido como a quantidade de colunas e de linhas que a imagem tem, normalmente expressa como *m* (Linhas) x *n* (Colunas), onde definem o valor total dos pixels.

Para que uma imagem possa ser compreendida em sua totalidade, uma resolução razoável se faz necessária, pois quanto mais *pixels* se fazem presentes na imagem, mais detalhes da cena se tornam possíveis de analisar. Vale a pena ressaltar que, uma imagem possuindo uma resolução grande (com uma quantia alta de *pixels*), é gerado um arquivo que requer mais poder de processamento para ser apresentado. A Figura 1 demonstra os efeitos que diversas resoluções espaciais tem sobre a mesma imagem, fomentando o conceito de quanto mais *pixels*, maior a resolução e consequentemente, melhor a compreensão da imagem.

Figura 1 - Aplicação de diversas resoluções em uma mesma imagem - I(m, n)



Fonte: Adaptado de Wikimedia Commons (2006).

Para representar o conceito de resolução em vídeos, o método resolução temporal é o mais utilizado, o qual expressa por meio da quantidade de imagens capturadas em um determinado intervalo de tempo. Uma das unidades de medida mais empregadas nessa categoria é a *frames* por segundo (quadros por segundo) ou apenas FPS.

Na maioria dos locais, tanto locais públicos quanto privados, que optam pelo uso de um sistema de câmeras de segurança são orientados a configurar a exibição para 7,5 FPS (ALMEIDA, 2018). Essa informação é relevante pelos seguintes pontos:

- Capacidade de reconhecer praticamente todos os movimentos orgânicos capturados no vídeo;

- Redução drástica no tamanho dos arquivos de vídeo;
- Possibilidade de aumentar a resolução de entrada, devido à baixa taxa de quadros;
- Diminuição na largura de banda necessária em mais de quatro vezes, para fazer o compartilhamento dessas imagens.

Entretanto, existem situações em que uma taxa de quadros maior é necessária, como casas de jogo, para que nenhum movimento brusco ou rápido demais se perca entre um quadro e outro. Entretanto esse sistema requer uma arquitetura de funcionamento mais avançada, investindo em armazenamento e manutenção constante, junto com a qualidade do equipamento (ALMEIDA, 2018).

2.3 Digitalização de imagem

A digitalização é uma forma de obter valores contínuos com o intuito de formar a imagem em um outro meio, geralmente computacional. Esse processo é dividido em duas etapas importantes: a amostragem e a quantização. A amostragem é a elaboração de uma matriz bidimensional a partir das coordenadas cartesianas da imagem obtida e a quantização consiste na atribuição de um valor inteiro para a intensidade dos *pixels* (PEDRINI; SCHWARTZ, 2007).

O valor que a quantização gera é analisado em uma escala de classificação. Para imagens monocromáticas, é utilizada a escala de cinza, no qual o seu parâmetro vai do 0 (preto) ao 255 (branco), dessa forma cada pixel se apresentará de uma forma a fim de construir a imagem final. Já para imagens coloridas, utiliza-se um vetor triplo, o qual cada posição mostra a intensidade da combinação entre as três cores: vermelho, verde e azul. Esse formato comumente é denominado de escala *RGB*, do inglês *red, green and blue* (SOLOMON; BRECKON, 2011).

2.4 Redes Neurais Artificiais

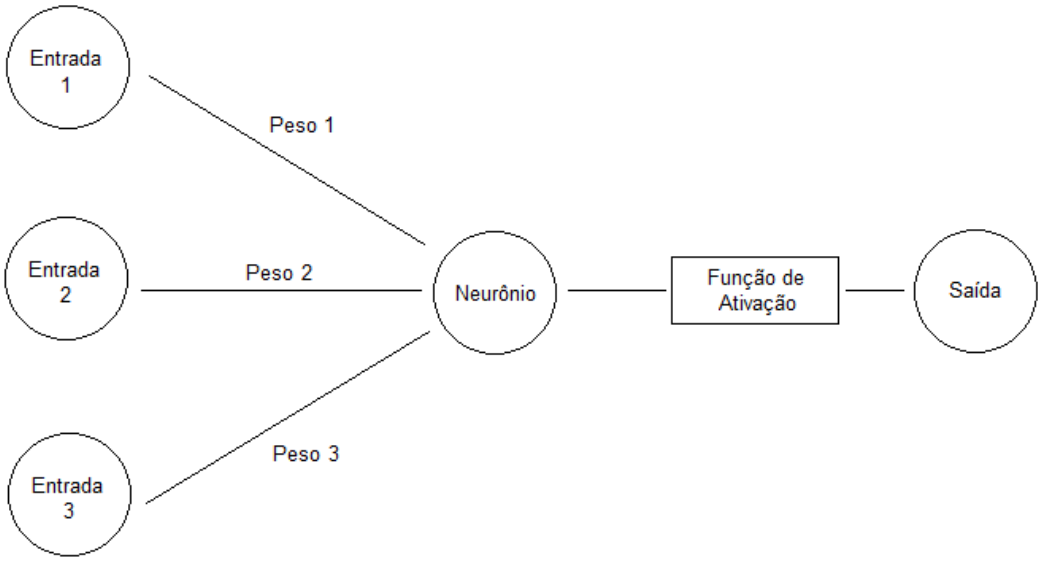
Como o próprio nome implica, uma rede neural artificial tem como objetivo reproduzir a sequência de atividades de acordo com o funcionamento do cérebro

humano. O cérebro pode ser denominado um sistema de processamento de dados muito complexo, não linear e paralelo, por conta da disposição da suas unidades de processamento mais básicas, os neurônios. Ele é capaz de organizar a linha de raciocínio afim de conseguir processar certos tipos de informações muito mais rápido do que qualquer máquina já construída (HAYKIN, 2007). Baseado nesse conceito, é assimilada a ideia já existente no âmbito orgânico para conseguir aplicar no mundo computacional, se tornando base para o aprendizado de máquina.

Para entender a dinâmica das redes neurais, foram destacados importantes pontos para se atentar: As informações serão processadas nos neurônios, que por sua vez são conectados entre si para troca e passagem de dados, mas nem toda a conexão tem a mesma força. Os neurônios tem estados internos pré-estabelecidos por conexões diretas com outros neurônios, e eles apresentam funções de ativação que ditarão o valor de saída. Em outras palavras, cada neurônio recebe um valor de entrada, que é a soma ponderada das saídas de outros neurônios diretamente conectados com ele, produzindo uma saída que pode ou não ser a entrada do neurônio seguinte (VASILEV *et al*. 2019).

Dessa forma, o neurônio artificial é constituído pela atribuição computacional dos conceitos anteriormente abordados. Os valores de entrada são multiplicados por um peso sináptico, afim de mensurar a importância de cada entrada para o processamento do neurônio. Após esse processamento, a informação passa por uma função de ativação, que é responsável por limitar a amplitude do valor de saída do neurônio (HEATON, 2015). A Figura 2 ilustra a estrutura básica de um neurônio artificial, conforme descrito no texto.

Figura 2 - Estrutura de um neurônio artificial



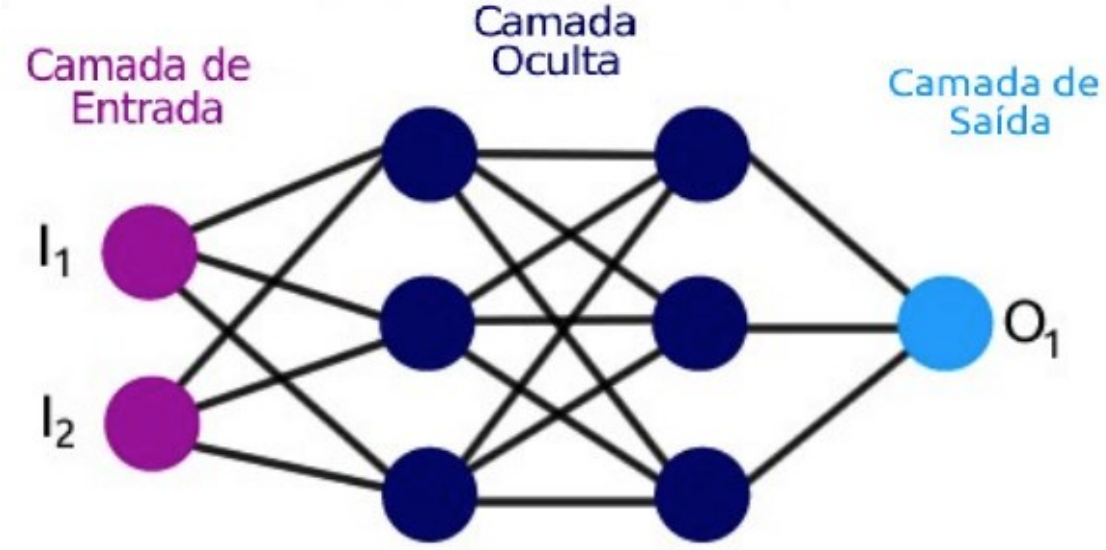
Fonte: Adaptado de Haykin (2007).

O processamento do neurônio artificial se dá pela Equação 1, que para alcançar o valor de saída *y*, deve ser feito o somatório das entradas dos neurônios anteriores x_i, realizado a multiplicação pelo seu peso condizente w_i e somado com um peso especial *b*, também chamado de *bias*, que tem como objetivo centralizar a curva da função de ativação. Esse resultado obtido será então aplicado na função de ativação que melhor se encaixa no problema, totalizando o valor de saída, que pode ser utilizado pelo próximo neurônio da cadeia (VASILEV *et al*. 2019).

$$\boldsymbol{y = f(\sum_i x_i \cdot w_i + b)} \qquad (1)$$

Já em uma rede neural multicamada, é necessário que todas as camadas se comuniquem entre si. Por conta da similaridade com o cérebro, uma rede neural artificial pode apresentar uma quantidade enorme de neurônios. Na Figura 3, é possível observar a relação entre dois valores de entrada, I_1 e I_2, duas camadas ocultas e apenas um valor de saída O_2.

Figura 3 - Topologia de uma Rede Neural Multicamada



Fonte: Adaptado de Grubler (2018).

No qual:

- $I = [I_1, I_2]$ são vetores de entrada;
- $O = [O_1]$ é o vetor de saída.

2.5 Funções de Ativação

A função de ativação é responsável por determinar se o resultado do cálculo daquele neurônio é condizente com o espectro total da rede neural, se ele pode ser considerado válido, ou se ele será desligado para não influenciar nos demais resultados posteriores da cadeia. Dada tamanha importância, a escolha de uma função de ativação que encaixe no problema proposto é essencial, levando em consideração que mais de uma função pode ser aplicada em neurônios diferentes (DATA SCIENCE ACADEMY, 2022).

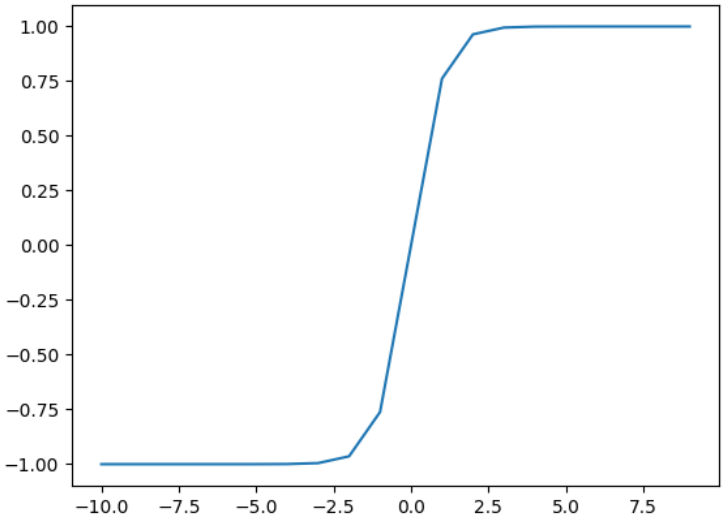
Uma função de ativação mais utilizada atualmente pelas redes neurais artificiais é a tangente hiperbólica ou *tangent hyperbolic* (tanh), pelo seu caráter não linear e diferenciável, se tornando essencial para calcular os gradientes durante o treinamento da rede neural. Ela possui a característica de mapear os valores na escala entre -1 e 1, propriedade essa que facilita as instruções do neurônio serem compreendidas pela máquina (DATA SCIENCE ACADEMY, 2022). Na Equação 2 é

apresentada a fórmula matemática da função *tanh*, sendo *x* o valor de trabalho do neurônio.

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \quad (2)$$

Por ser uma função simples, seu comportamento será uma hipérbole linear a partir do valor de *x* resultando a um valor de *y* variando de -1 até o 1, conforme a Figura 4, facilitando a compreensão se o neurônio conseguiu produzir um valor condizente com o escopo da rede neural.

Figura 4 - Comportamento gráfico da função *tanh*



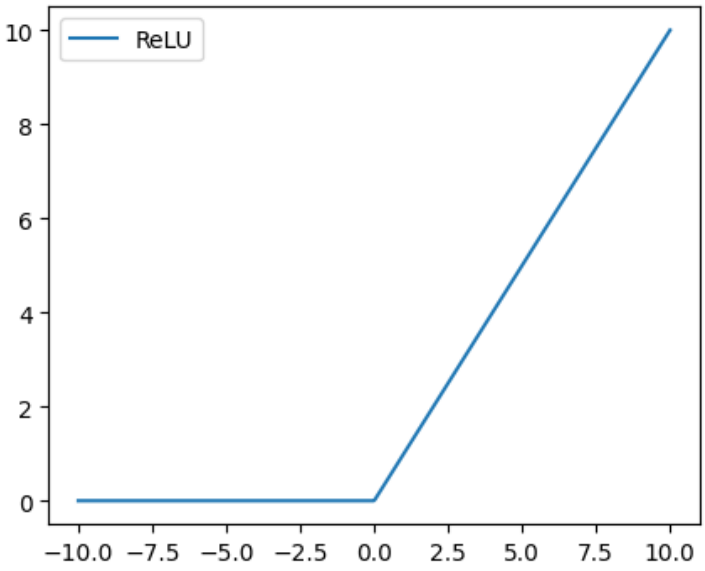
Fonte: Autoria própria (2023).

Outra função de ativação importante é a unidade linear retificada ou *Rectified Linear Unit* (*ReLU*), pelo seu caráter não linear e a propriedade de não precisar ativar todos os neurônios da rede simultaneamente para realizar seus cálculos, facilitando as instruções do neurônio serem compreendidas pela máquina (DATA SCIENCE ACADEMY, 2022). A Equação 3 é apresentado a fórmula da função *ReLU*, sendo *x* o valor resultante de trabalho do neurônio artificial.

$$f(x) = \begin{cases} x, x \geq 0 \\ 0, x < 0 \end{cases} \quad (3)$$

Por também ser uma função simples, seu comportamento será uma reta crescente e linear a partir do eixo *x*, conforme a Figura 5, facilitando a compreensão se o neurônio conseguiu produzir um valor condizente com o escopo da rede neural.

Figura 5 - Comportamento gráfico da função *ReLU*



Fonte: Autoria própria (2023).

Outra função de ativação utilizada nos problemas de multiclasse é a chamada *softmax*. Ela converte um vetor de números reais em um vetor de probabilidades, no qual cada elemento representa a chance de pertencer à determinada classe. Como a *softmax* faz a normalização das instâncias, o seu resultante são valores inteiros iguais a 1, ideal para utilizar no final da rede neural, descrita pela Equação 4 (DATA SCIENCE ACADEMY, 2022).

$$(\boldsymbol{z})_{\boldsymbol{j}} = \frac{\boldsymbol{e}^{\boldsymbol{z_j}}}{\sum_{\boldsymbol{k=1}}^{\boldsymbol{K}} \boldsymbol{e}^{\boldsymbol{z_k}}} \quad (4)$$

No qual:

- z_j é o valor de entrada para a função *softmax* associado à classe *j*;
- *e* é a base do logaritmo natural;
- E o somatório de todos os valores exponenciais dos valores de entrada z_k para todas as classes.

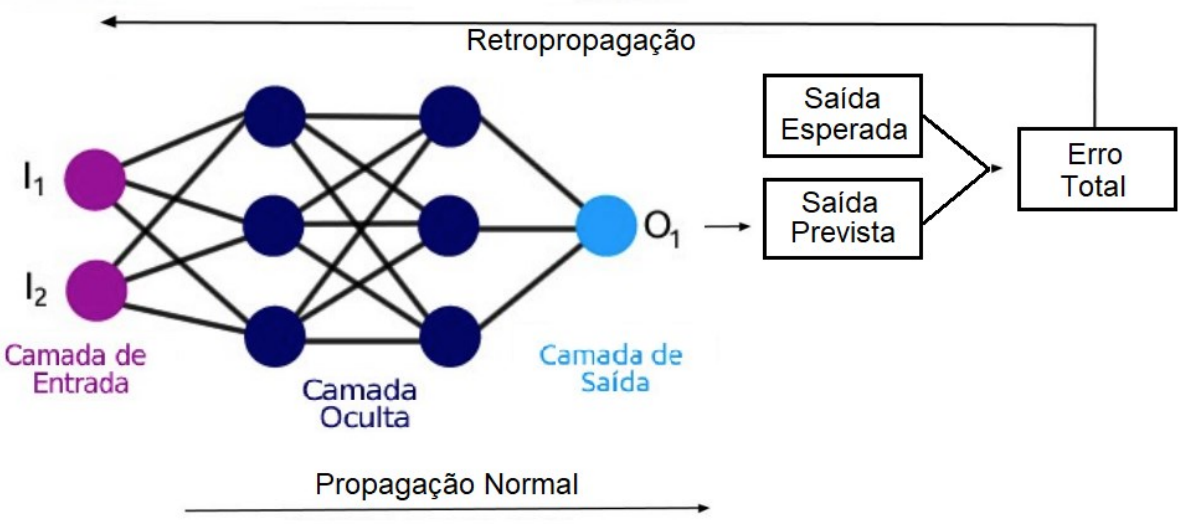
2.6 *Backpropagation* em Redes Neurais

Para treinar uma rede neural artificial, é necessário alterar levemente o valor dos pesos que cada entrada terá, para que assim os neurônios consigam repetir os cálculos com os valores atualizados, otimizando o resultado final esperado. Entre as diversas formas de otimização das redes neurais, uma que acaba se destacando é a técnica de *backpropagation*, ou retropropagação em português. Essa ideia acaba sendo um tipo especial do algoritmo de gradiente descendente (HEATON, 2015).

Após a predição ser realizada pela rede, o erro entre o valor esperado e o obtido é utilizado com o gradiente da saída gerada pela camada final. Então, por meio da regra da cadeia, os pesos e *bias* das camadas anteriores são atualizados. Essa calibração dos pesos e *bias* formam o aprendizado na rede (DATA SCIENCE ACADEMY, 2022).

Na Figura 5, é possível observar o funcionamento normal da rede e, após a comparação final do neurônio, entre o resultado esperado com o resultado obtido, a fase *backpropagation* se inicia para confirmar se existe a necessidade de atualizar o *bias* ou não.

Figura 6 - Funcionamento da fase *backpropagation*



Fonte: Adaptado de Grubler (2018).

O erro calculado na fase de propagação normal é utilizado na segunda fase, por meio de um cálculo de gradiente, valor esse que é calculado pela derivada da função de erro, aplicando cada um dos pesos da rede. Por meio desse gradiente, é

possível averiguar quando é necessário modificar o valor do peso, afim de melhorar o erro total (TAYLOR, 2017).

2.7 *Deep Learning* em Redes Neurais

Redes neurais são compostas geralmente por três camadas: uma camada de entrada, uma camada de saída e uma camada intermediária, essa responsável pelo processamento dos dados. Logo uma rede neural com camadas profundas (*deep learning*) possui múltiplas camadas intermediárias. O termo profundo geralmente se refere ao número de camadas intermediárias da rede, uma vez que tradicionalmente redes neurais comuns tem de uma até três camadas intermediárias, entretanto, com uma rede no estilo *deep learning* é possível chegar a mais de 150 camadas. O aprendizado profundo consiste em programar um sistema para ele conseguir aprender sozinho, obtendo conhecimento de maneira automática, sem a interação de um ser humano (GOODFELLOW, *et al*. 2016).

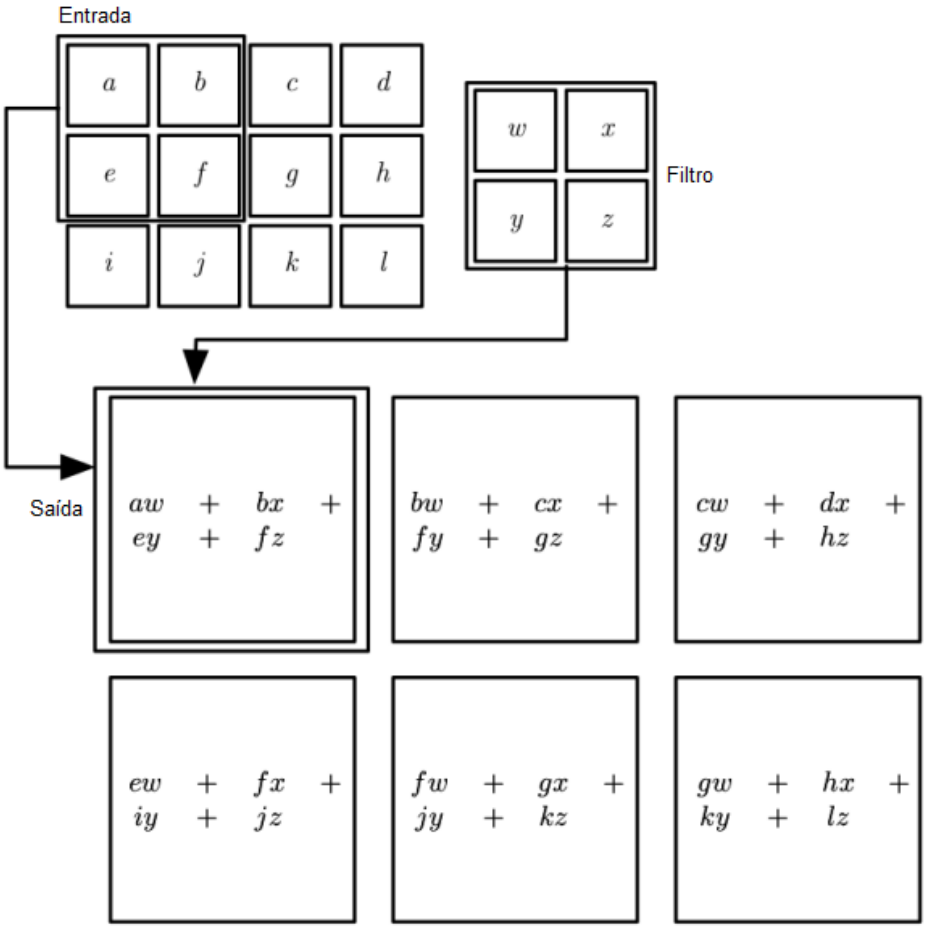
Esse conhecimento consiste em detectar padrões em diversos conjuntos de dados. Em outras palavras, a entrada do algoritmo é o amontoado de dados, que passará pela análise de sua rede neural, e retorna um programa capaz de desempenhar uma função específica, como conseguir detectar objetos em vídeos (GLOROT, *et al*. 2010).

2.8 Redes Neurais Convolucionais

Redes neurais convolucionais (RNCs) ou do inglês *Convolutional Neural Network* (CNN) são um conjunto de redes neurais que tem se destacado no quesito aprendizado profundo, principalmente em áreas relacionadas a reconhecimentos de imagem e classificação. O termo convolucional é pelo fato da rede neural utilizar a convolução no lugar da multiplicação da matriz, geralmente em uma de suas camadas para uma rede neural convolucional funcionar de acordo, ela conta com algumas camadas comportando funções especiais, como a camada convolucional e a camada de *pooling*, ou agrupamento (GOODFELLOW, *et al*. 2016).

A camada convolucional utiliza-se de filtros previamente treinados pela rede para captar características da imagem, na qual cada filtro gera uma nova matriz, que ressalta as características procuradas por cada filtro, como cores, pontas de objetos e outros padrões treinados (LAWRENCE *et al*. 1997). Na Figura 6 é possível entender o processo de convolução, que destaca a multiplicação de cada região da matriz pelo filtro convolucional.

Figura 7 - Exemplo do processo de convolução

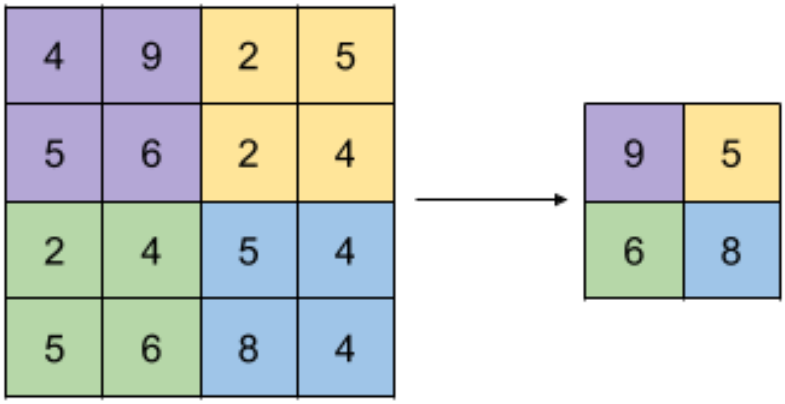


Fonte: Adaptado de GoodFellow *et al*. (2016).

Já camada de *pooling* é responsável por diminuir o tamanho das matrizes geradas pelo algoritmo da camada anterior, entretanto essa compressão da matriz deve ser feita sobressaindo ainda mais as características capturadas pelos filtros anteriores. Outra característica importante da camada de *pooling* é a capacidade de ignorar os possíveis ruídos que a imagem possa ter, facilitando a procura dos elementos requeridos na imagem. Entre os diversos algoritmos de *pooling*, o *max pooling* é o mais usual por conta de preservar o neurônio com o maior sinal de ativação

de cada uma das regiões de entrada (VASILEV, *et al*. 2019). A Figura 7 destaca o processo do *max pooling* de 4 valores genéricos de altura por 4 comprimento, para uma matriz de 2 por 2.

Figura 8 - Exemplo de *max pooling* para uma matriz 2 x 2

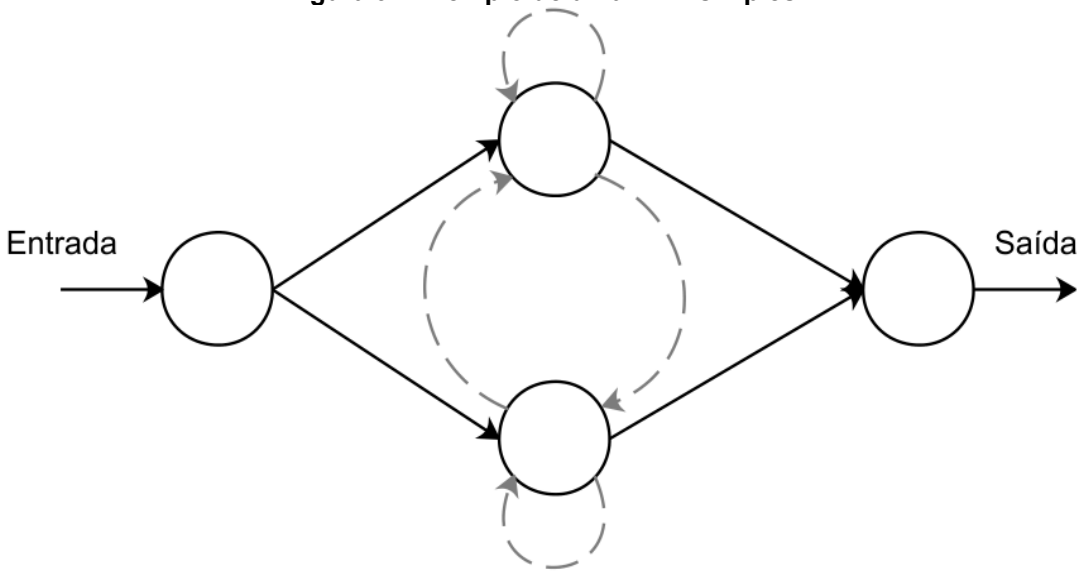


Fonte: Adaptado de Pokhrel (2019).

2.9 Redes Neurais Recorrentes

As redes neurais recorrentes (ou RNR) podem ser interpretadas como uma rede neural de propagação normal, entretanto o diferencial é um incremento na memória, ou seja, não é necessário depender apenas da entrada atual, mas também do resultado que está guardado na memória dos neurônios anteriores. Portanto, a rede neural recorrente é muito utilizada para reconhecimento de ações que já estão sendo treinadas por algum tempo, como reconhecimento de fala, de escrita manual, predição de próxima palavra em teclados de celular, com base nas letras já inseridas e nas palavras frequentemente utilizadas (DATA SCIENCE ACADEMY, 2022). Conforme esquematizado na Figura 8, a RNR possui as setas tracejadas indicando os valores “conversando” entre os neurônios, permitindo que as informações persistam na rede.

Figura 9 - Exemplo de uma RNR simples



Fonte: Bitaraes, *et al*. (2019).

A arquitetura *Long Short-Term Memory* (LSTM) ou memória longa de curto prazo, em português, é baseada nas redes neurais recorrentes, que classifica, processa e pode prever séries temporais sem estar presas a um determinado intervalo de tempo, se diferenciando da aplicação simples da RNR convencional. Uma arquitetura LSTM consiste em várias células interconectadas. Cada célula possui quatro componentes principais – As portas de entrada, as portas de esquecimento, a unidade de memória e as portas de saída. Esses componentes trabalham juntos para permitir que a LSTM compreenda o escopo do processo e mantenha as informações por longo prazo (KANG, 2017).

As portas de entrada determinam quais informações devem ser armazenadas na célula de memória, enquanto as portas de esquecimento controlam quais informações devem ser descartadas. A unidade de memória é responsável por armazenar e lembrar informações relevantes. Ela pode armazenar e atualizar seu estado ao longo de todo o tempo da execução. Por fim, as portas de saída decidem quais informações devem ser transmitidas para as camadas subsequentes da rede (KANG, 2017).

3 MATERIAIS E MÉTODOS

Neste capítulo são apresentados os materiais necessários para realizar os treinamentos e os testes propostos neste trabalho. Na sequência, são definidos os métodos que serão utilizados para atingir os objetivos propostos.

3.1 Equipamentos a Serem Utilizados

Para realizar o treinamento e os testes, a opção utilizada é fornecida pela Google, chamada de Google *Colaboratory*, ou simplesmente Google *Colab*, é um ambiente online e interativo que permite aplicar códigos na linguagem *Python* no próprio navegador, que será compilado pelos servidores e retornado os resultados (COLAB, 2022). No momento inicial do trabalho, comparando o custo benefício da plataforma *Colab* para o escopo do trabalho proposto, não foi necessário adquirir a versão paga da plataforma, justamente pela base de dados não ser extremamente extensa, nem os algoritmos complexos o bastante, sem contar que o *hardware* adicional não é garantido mesmo realizando o pagamento, justamente pela alta demanda dos demais usuários.

Para realizar as configurações, programação e compilação de código, bem como a edição das imagens e vídeos, o *laptop* Lenovo G50 foi utilizado conforme as seguintes especificações técnicas:

- Processador: Intel Core i5 5200U:
    - Núcleos físicos: 2;
    - Núcleos lógicos: 4;
- Memória: 8 GB RAM;
- Placa de vídeo: AMD Radeon HD 8500M, 2 GB de VRAM.

3.2 Ferramentas e Bibliotecas (*Python*)

O *Python* é uma linguagem de programação comumente escolhida para desenvolver e aplicar conceitos de inteligência artificial e ciência de dados,

principalmente aprendizado de máquina. Suas características de alto nível e multiplataforma, somadas ao fato de ser uma linguagem de código aberto, fornecem uma série de bibliotecas de código prontas, muitas delas criadas pela comunidade de programação para os mais diferentes segmentos de implementações (PYTHON, 2022). Por conta dessas características positivas, a linguagem *Python* foi escolhida para aplicar as abordagens de análise de vídeo.

As bibliotecas do *Python* fornecem ideias previamente implementadas já pensadas para facilitar a criação e programação de algoritmos, justamente por absorver muito bem diversos segmentos de aplicação em um pacote simples de fácil importação. Entre as diversas bibliotecas utilizadas, as mais importantes:

- Tensorflow: Lançado ao público em fevereiro de 2017, *tensorflow* é uma biblioteca de código aberto criada pela equipe *brain* da *Google* utilizada para computação numérica e *machine learning* (aprendizado de máquina). A biblioteca agrupa uma grande quantidade de modelos e algoritmos para *machine learning* e *deep learning* (aprendizado profundo) e o disponibiliza de um modo abstraído, permitindo maior facilidade para o desenvolvimento. *Tensorflow* possui uma aplicação para a programação do usuário em *python*, porém a ferramenta em si é executada na linguagem c++, para que seja possível uma melhor performance de execução (TENSORFLOW, 2022);
- Keras: O *tensorflow* adotou a ferramenta *keras* como uma aplicação de alto nível desde suas primeiras versões. O intuito do *keras* é criar uma plataforma de fácil compreensão, para diminuir o tempo de aprendizado e proporcionar treinamentos com grandes quantidades de dados sem complicar a programação. A ferramenta *keras* não possui uma aplicação para realizar operações em baixo nível, como produto de matrizes, tensores e convoluções, por isso necessita de ferramentas que farão essa compilação de baixo nível, e a principal e padrão utilizada pelo *keras* é o *tensorflow* (KERAS, 2023);
- Pytorch: O *pytorch* é uma biblioteca criada em 2016 e desenvolvida pela *FAIR* (*Facebook's AI Research*), foi escrita baseada em *python*, *c++* e *cuda*. Assim como o *tensorflow*, o *pytorch* tem objetivos similares – facilitar o desenvolvimento de redes neurais e sua aplicação em diversos escopos. O *pytorch* vem ganhando bastante espaço, principalmente pelo seu suporte ao uso de placas gráficas para treinamento das redes neurais, ponto

extremamente importante quando se possui grandes quantidades de dados a serem analisados (PYTORCH, 2022);

- Numpy: O *numpy* é uma biblioteca *open-source* amplamente utilizada para realizar cálculos numéricos e científicos em *python*. Ele oferece suporte a *arrays* multidimensionais e fornece uma ampla gama de funções matemáticas e ferramentas de manipulação de dados (NUMPY, 2023);
- Matplotlib: O *matplotlib* é uma biblioteca de visualização de dados em *python*, amplamente utilizada para criar gráficos e visualizações de alta qualidade (MATPLOTLIB, 2023);
- Opencv: O *opencv* é uma biblioteca de código aberto que oferece suporte ao processamento de imagens e visão computacional. Ele foi desenvolvido com o objetivo de fornecer ferramentas e mais de 2500 algoritmos para realizar tarefas relacionadas à interpretação e análise de imagens (OPENCV, 2023);
- Scikit-image: O *scikit-image* é uma biblioteca de processamento de imagens baseada em *python*, que oferece uma ampla gama de funcionalidades para manipulação, análise e processamento de imagens. Ele é construído em cima da biblioteca *numpy* e fornece algoritmos pré-programados para realizar diversas tarefas relacionadas ao processamento de imagens (SCIKIT, 2023).

3.3 Trabalhos Correlatos

Para desenvolver uma aplicação que ajude a automatizar a detecção dessas ações violentas, é necessário compreender como é feita a análise de imagens. Para alcançar tal resultado, diversos trabalhos relacionados serviram de base para o estudo, sendo os mais similares os seguintes:

- (ABDALI; AL-TUMA, 2019): Utilização de RNC e LSTM na detecção de violência em vídeo;
- (BABA; GUI; CERNAZANU; PESCARU, 2019): Uma abordagem de rede de sensores para detecção de violência em cidades inteligentes usando *deep learning*.

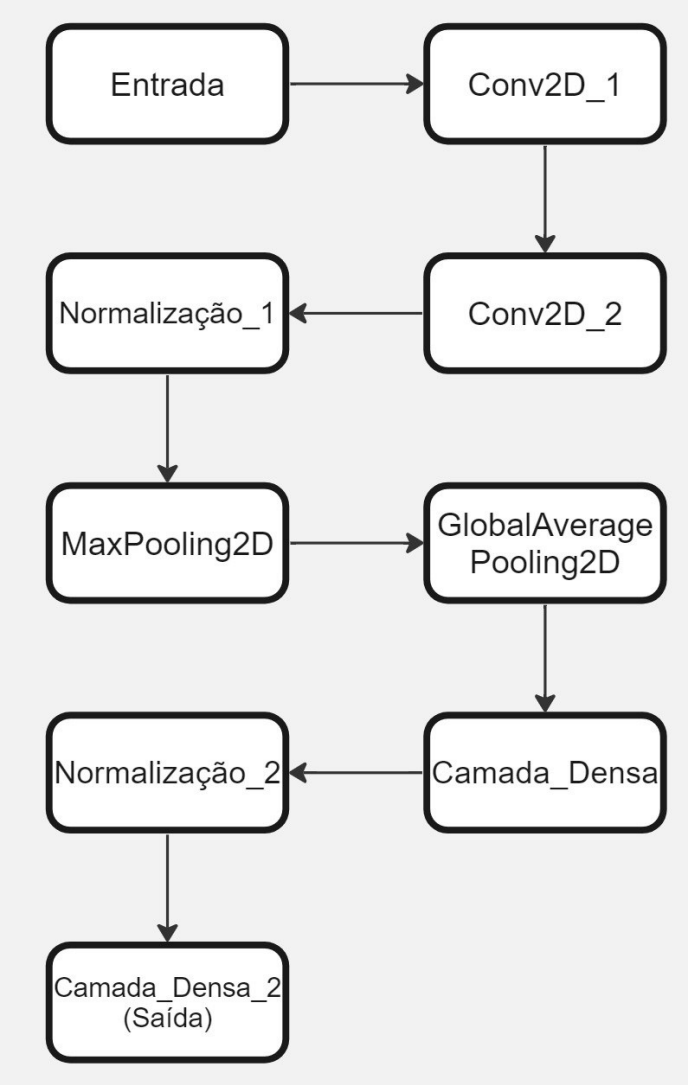
3.4 Redes Neurais Utilizadas

Para a implementação da abordagem quadro a quadro, foi escolhido como base o algoritmo disponibilizado pela plataforma *Learn OpenCV* (2021), que apresenta uma arquitetura de nove camadas, no qual é aplicado os frames extraídos do *dataset* de forma sequencial, essa extração é feita a partir da busca entre todos os vídeos compostos naquela classe especificada. Após a obtenção dessas imagens, as instancias são transmitidas para uma camada convolucional 2D, por trabalhar apenas com os valores de altura, largura e taxa de cores. A rede então aplicará a função de ativação *ReLU* e a próxima camada fará a normalização desses valores, sendo posteriormente aplicados em uma camada de *max pooling*.

A próxima etapa é utilizar a função de *global average pooling*, que fará a transformação da altura e largura da imagem em uma classificação só, facilitando a etapa seguinte, chamada de camada densa, conectando todas as camadas anteriores, aplicando o parâmetro 256, por ser o resultado de 64 vezes 4, que é o número de classes escolhido, e a camada será ativada com a função *ReLU* (KERAS, 2023).

Para a oitava camada, será normalizado novamente o conjunto de dados e finalmente aplicada a função de ativação *softmax*, a fim de retornar vetores de valor 1 na classe com mais acerto e 0 nas demais. Toda a construção do algoritmo é feita com base na biblioteca *keras*, que inclui as funções utilizadas. A Figura 10 a seguir retrata a arquitetura do modelo de análise quadro a quadro, conforme especificado acima.

Figura 10 - Arquitetura do Modelo Quadro a Quadro



Fonte: Autoria própria (2023).

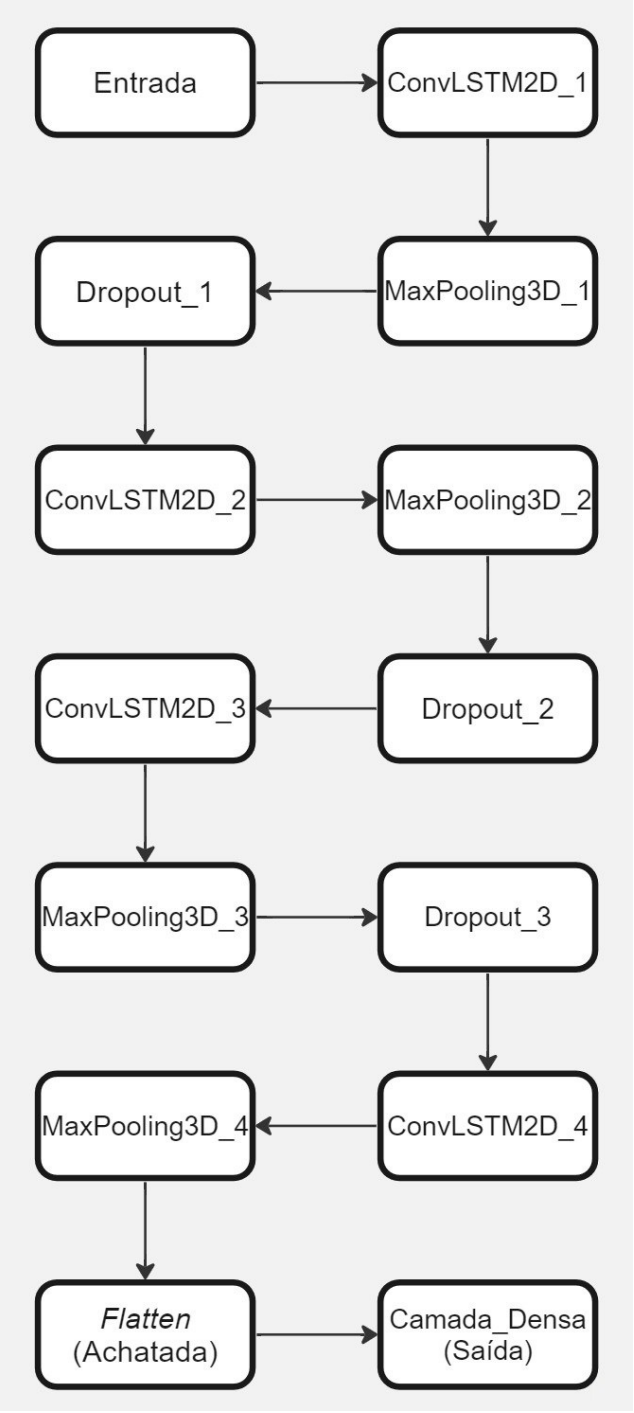
Para a segunda rede neural a ser implementado, foi possível utilizar como referência o algoritmo exposto pela plataforma *BleedAI* (2021), que possibilitou a escolha de uma arquitetura que mescla dois formatos descritos neste trabalho, a rede *ConvLSTM*. Essa rede tem como função aplicar uma RNC para extrair as características espaciais da sequência de frames do vídeo e aplicar logo na arquitetura LSTM com a peculiaridade de substituir o processamento padrão da LSTM por convoluções, que se torna capaz de manter as identidades espaciais da imagem e, ao mesmo tempo, identificar as relações temporais de tal intervalo de frames extraídos do vídeo.

Em comparação com a arquitetura quadro a quadro, esta arquitetura possui quatorze camadas, que se inicia com a função *ConvLSTM2D*, contida na biblioteca do *keras*, que por sua vez são retratados os valores de quantidade de *frames*, altura e

largura dos mesmos, e ativada com a função *tanh*, que posteriormente é aplicada na camada de *max pooling* 3D, que diferente da 2D, tem-se um atributo adicional sendo a sequência temporal de frames. Esta função tem como objetivo diminuir a dimensionalidade das imagens para prevenir trabalho computacional desnecessário, juntamente com a função *dropout*, que por sua vez atribui valores iguais a zero aleatoriamente para prevenir o *overfitting* (quando o modelo acaba “decorando” os dados de treinamento, o incapacitando de desempenhar com qualidade em classes diferentes) (KERAS, 2023).

Essa sequência entre a função *ConvLSTM*, *max pooling* 3D e *dropout* acontece três vezes na rede neural afim de diminuir os dados e focar naqueles que contém as informações requeridas. No final do terceiro *max pooling* 3D, os dados serão submetidos a camada *flatten*, que tem como função compactar todos os valores (taxa de frames, altura, largura e taxa de cores) em um vetor de uma dimensão apenas, utilizando a multiplicação entre eles para alcançar o resultado (KERAS, 2023). Por fim, esse valor unitário é submetido a camada densa que passará pela função de ativação *softmax*, retornando o vetor de 1 para a classe escolhida e zero para as restantes. As quatorze camadas presentes na arquitetura da rede neural *ConvLSTM* podem ser observadas na Figura 11.

Figura 11 - Arquitetura do Modelo *ConvLSTM*



Fonte: Autoria própria (2023).

3.5 Bases de Dados

Para treinar o reconhecimento de ações de movimento em vídeos, o *dataset* (ou base de dados) escolhido para tal função é o denominado UCF50, disponibilizado para fins acadêmicos pela Universidade Central da Flórida. Essa base de dados

fornece 50 classes de ações diferentes, como ciclismo, corrida de cavalos e jogo de bilhar (KISHORE *et al*. 2012). Em um primeiro momento, é interessante trabalhar com essas ações mais genéricas, para averiguar se as redes neurais estão configuradas da forma correta. Alguns exemplos de imagens extraídas de vídeos são ilustrados na Figura 12.

Figura 12 – Exemplos de Quadros das Classes Escolhidas do *dataset* UCF50

Fonte: Autoria própria (2023).

Foram escolhidas quatro classes de ações de movimento para realizar os testes preliminares, tais sendo:

- *Fencing* (111 vídeos): Classe de dois esgrimistas duelando entre si, relevante pelo contato de duas pessoas com um espaço entre elas, utilizando de um objeto longo (espada);
- *TaiChi* (100 vídeos): Uma pessoa apenas durante a execução do vídeo realizando movimentos de artes marciais, porém com vagarosidade entre uma ação e outra;
- *Drumming* (161 vídeos): Uma única pessoa tocando bateria na cena, que condiz com movimentos rápidos de braços e mãos, tal rapidez nos movimentos é geralmente encontrada em brigas;
- *HorseRace* (127 vídeos): Corrida de cavalos, que são ações de movimento ao longe entre vários elementos móveis, diferenciando das outras três classes.

Para o treinamento com ações de violência, diversas bases de dados disponíveis foram analisadas, priorizando vídeos provenientes de ações de violência reais, portanto o *Real Life Violence Situations Dataset,* idealizado por um grupo egípcio, se sobressaiu em relação aos demais analisados, se tornando o escolhido para o presente trabalho. Esta base de dados possui duas classes *Violence* e *NonViolence*, esta última contendo ações aleatórias não condizentes com a categoria violência, conforme demonstrado na Figura 13 (SOLIMAN, 2019).

Figura 13 - Ações de Violência e Não Violência presentes do dataset *Real Life Violence Situations*.

Fonte: Autoria própria (2023).

3.6 Pré-processamento e estipulação de métricas

Para fins computacionais, o pré-processamento utilizado para inserir nas redes neurais se baseou em diminuir a resolução de entrada dos quadros extraídos entre todos os vídeos utilizados da base de dados, a resolução escolhida para tal foi de 64 *pixels* de altura e 64 *pixels* de largura. Outra etapa foi a normalização dos *pixels* de cada imagem extraída, utilizando a divisão dos mesmos por 255, valor este condizente com a escala de cores RGB que cada *pixel* pode assumir, resultando em valores entre 0 e 1 para cada elemento.

Em relação ao ambiente de teste *Google Colab*, foi importado e extraído o *dataset* UCF50 primeiro em uma máquina física particular, para então fazer a importação no site, afim de evitar possíveis alterações por parte de seus idealizadores durante todo o processo do trabalho.

A mesma estratégia de armazenar uma cópia estática foi aplicada com o *Real Life Violence Situations dataset*. Entretanto, por limitações da plataforma *Google Colab*, não foi possível realizar os testes com os 1000 vídeos totais de cada classe, *Violence* e *NonViolence*, pois o ambiente desconectava sozinho por uso excessivo de memória. Portanto, o ambiente permitiu a entrada de 500 vídeos de cada classe sem apresentar nenhum erro, do início ao fim dos testes. A métrica utilizada para remover 500 vídeos do *dataset* sem perder a representatividade, foi subtrair o intervalo de

vídeos entre o denominado NV_251 até o NV_750 para a classe *NonViolence*, e da mesma forma, remover do V_251 até o V_750 da classe *Violence*.

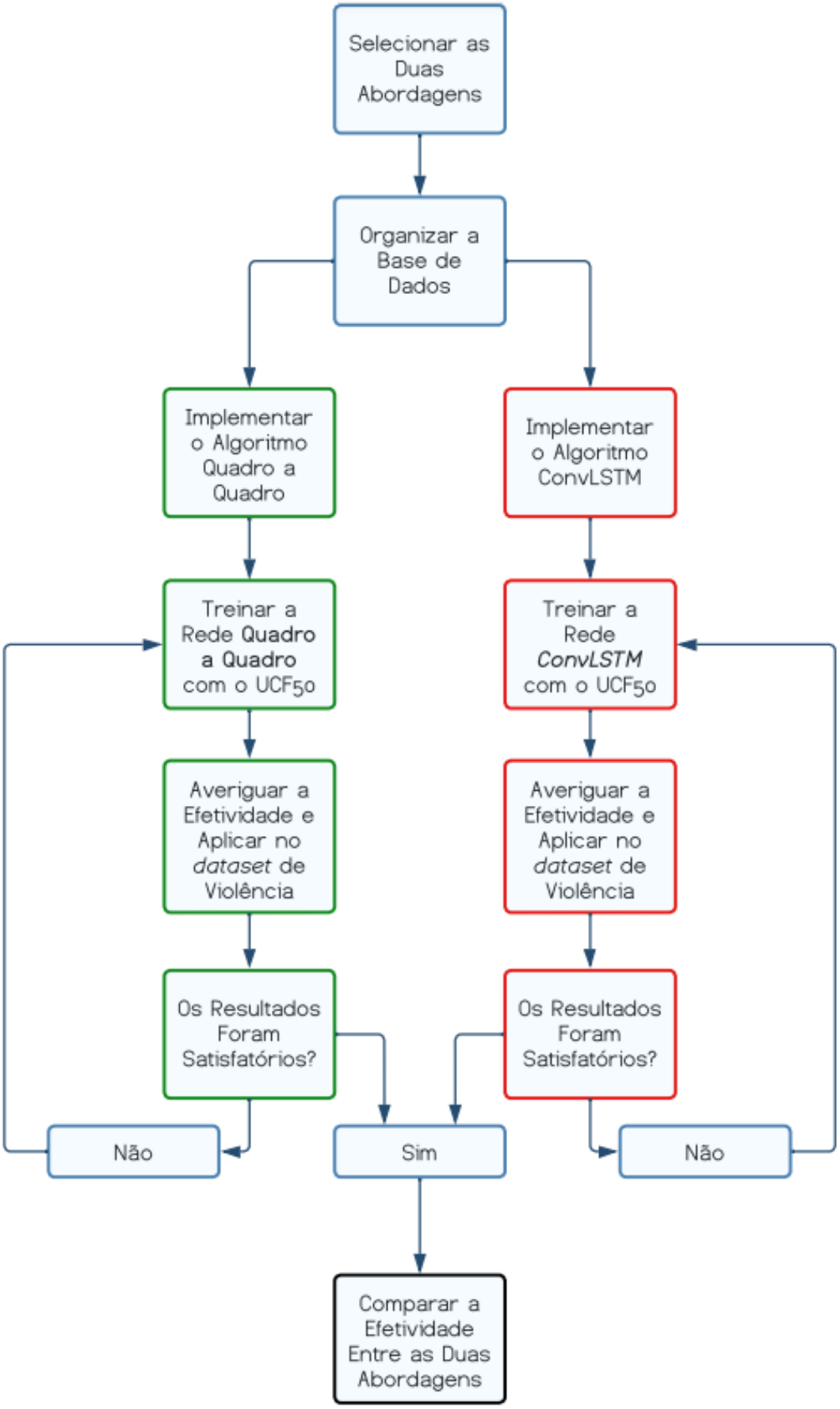
Para o treinamento e teste, foram empregadas 50 épocas de execução do modelo, ou seja, 50 vezes que o algoritmo vai analisar a entrada e fornecer um resultado. A quantidade 50 é uma métrica condizente com o treinamento necessário que o algoritmo precisa para retornar uma acurácia alta, além de prevenir o *overfitting* dessa execução. A função denominada *early stopping* também foi inserida para ajudar nessa prevenção, fornecendo a interrupção do treinamento e validação quando uma variável está estagnada. As métricas fornecidas para realizar essa parada precoce foram baseadas no '*val_perda'*, que é o valor de perda de análise entre as imagens no processo de validação da rede neural. Por conta disso, o '*val_perda'* é constantemente analisado para encontrar o seu menor número e, quando o atual é menor que o anterior, ele possui uma margem fixa épocas subsequentes para conseguir um menor. Caso não consiga, a execução é interrompida. Essa margem para alcançar um '*val_perda'* menor é chamada de paciência (ou *patience*, em inglês).

3.7 Roteiro dos Experimentos

Conforme a Figura 14 demonstra, os experimentos se resumem em implementar e treinar as duas abordagens de reconhecimento de movimento nas bases de dados, afim de reconhecer que tipo de ação está sendo realizada no vídeo, como um indivíduo esticar o braço e ir em direção a outra pessoa, ou sacar uma arma e apontar, independente de quem ou o que. Posteriormente, o treinamento das mesmas em situações que previamente se tem conhecimento, afim de descobrir se o vídeo possui ou não uma ação de violência.

Após a realização dos testes, será obtido os resultados por meio das métricas fornecidas pelos próprios algoritmos, a fim de analisar e comparar as particularidades de um com o outro, para discernir qual o mais indicado para o escopo do trabalho.

Figura 14 - Fluxograma de Organização do Estudo.



Fonte: Autoria própria (2023).

3.7.1 Primeira etapa: Selecionar as duas abordagens

A primeira abordagem escolhida para ser trabalhada é o algoritmo RNC para a análise quadro a quadro do vídeo, conforme descrito na seção 2.2, extraindo os frames dos vídeos e realizando a análise juntamente com os conceitos de *pooling*. Já para a segunda abordagem, foi aplicado os conhecimentos de RNC em junção com a abordagem LSTM no qual, conforme especificado nas seções 2.2 e 2.3, desempenham o papel de extrair valores espaciais das imagens nos vídeos, e aplicar a sequência de aprendizado com persistência, respectivamente.

3.7.2 Segunda etapa: Aplicar as abordagens no UCF50

As bases de dados selecionadas serão divididas em que tipo de ação está sendo desempenhada na imagem, os nomes dessas ações serão os nomes das classes que os algoritmos escolhidos vão analisar e gravar na sua memória. Será então feito o pré-processamento, mencionado na seção 3.4, das imagens extraídas e feito o treinamento com a primeira base de dados UCF50, essa parte do treinamento servirá para habituar e reconhecer se a aplicação dos algoritmos está ocorrendo de forma positiva ou necessita de ajustes.

3.7.3 Terceira etapa: Aplicar as abordagens no *dataset* de violência e comparar os resultados

Nesta etapa, após o treinamento satisfatório dos dois algoritmos, será feita a implementação dos algoritmos na base de dados *Real Life Violence Situations*, afim de observar o comportamento dos algoritmos em cenas reais de violência, e se eles são capazes de compreender o ocorrido. Após o sucesso individual em classificar essa ação violenta, é feita a comparação entre as duas abordagens, a fim de averiguar qual obteve mais sucesso em compreender o que se passava, para poder aprimorar e realizar os testes novamente.

4 RESULTADOS E DISCUSSÃO

A presente seção aborda os resultados obtidos no treinamento e fomenta a discussão dos mesmos.

4.1 Abordagem Quadro a Quadro

Nos testes da abordagem quadro a quadro, foram experimentados alguns fatores de métricas além dos padronizados anteriormente que surtiram efeitos diversos nos resultados finais. Os primeiros testes realizados com o *dataset* UCF50, podem ser visualizados conforme a Tabela 1, utilizando as quatro classes previamente comentadas.

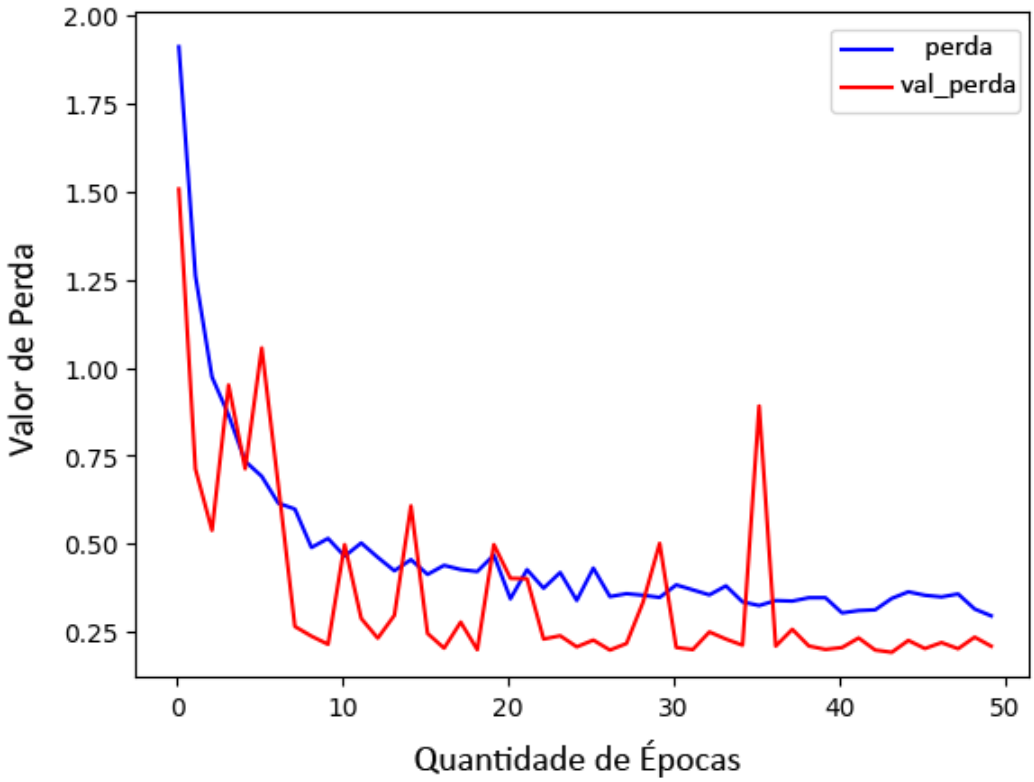
Tabela 1 - Resultados da Rede Quadro a Quadro com o *dataset* UCF50

paciência	Número de *Frames*	*val_perda*	val_acuracia
15	10	0,6567	0,8840
20	10	0,4220	0,9084
15	50	0,1767	0,9677
20	**50**	**0,1041**	**0,9875**

Fonte: Autoria própria (2023).

Dessa maneira, os melhores resultados obtidos foram com a sequência de 50 *frames* de cada vídeo analisado, juntamente com a margem de 20 épocas como paciência. Retornando a perda média da análise de 0,1041 e uma acurácia média de 98,75%. A métrica chamada perda, é uma medida da discrepância entre as validações realizadas pelo algoritmo e seus valores reais baseados no treinamento. O objetivo do treinamento é fazer a rede neural aprender a minimizar esse erro encontrado, fazendo ajustes conforme o necessário (KERAS, 2023). Na Figura 15 é possível compreender a curva do '*val_perda'*, em vermelho, se caracterizando pela perda na hora da validação do algoritmo, e a *perda*, em azul, condiz com a etapa do treinamento. O gráfico da Figura 15 ilustra o melhor resultado obtido, no qual o eixo horizontal é caracterizado pelas épocas, e o eixo vertical o valor *perda* total.

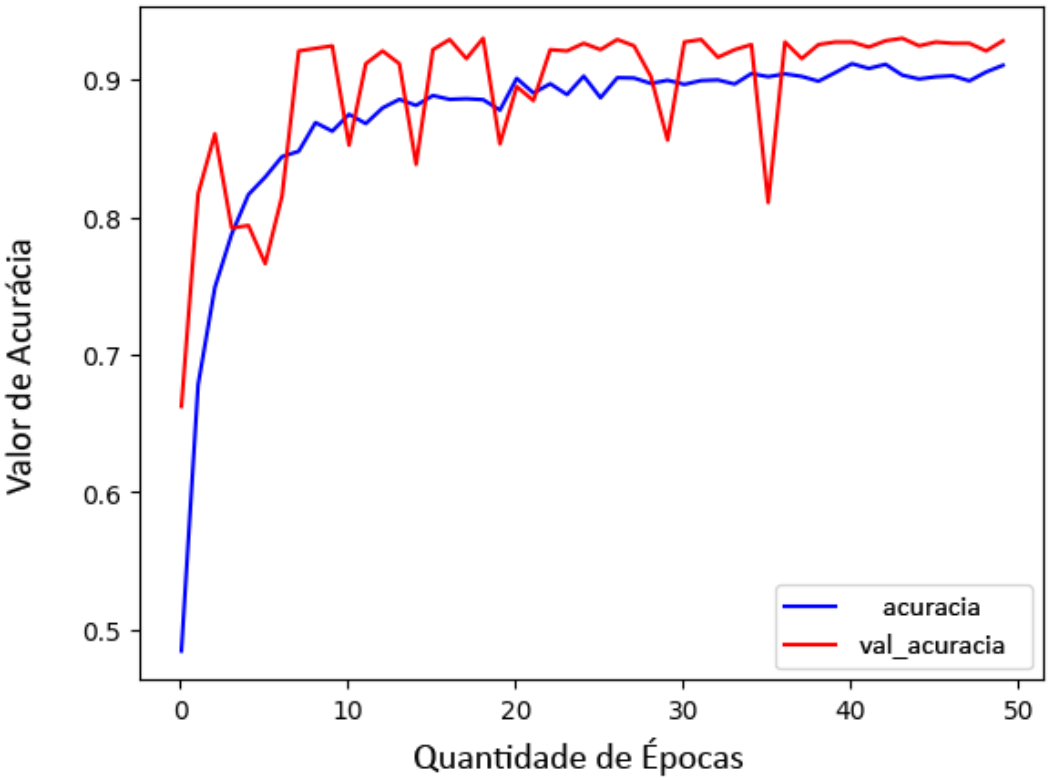
Figura 15 - *val_perda* vs. *perda* da Rede Quadro a Quadro – UCF50



Fonte: Autoria própria (2023).

Já na Figura 16 é apresentado o gráfico da função *acurácia*, em azul, que comporta a acurácia fornecida pelo treinamento, assim como o *val_acuracia*, em vermelho, sintetizando a acurácia alcançada pela validação da rede. O eixo horizontal se mantém descritas as épocas totais realizadas, e o eixo vertical o valor total da acurácia alcançada, em uma escala de 0 a 1, sendo 0 o menor valor possível, e 1 equiparado a 100%.

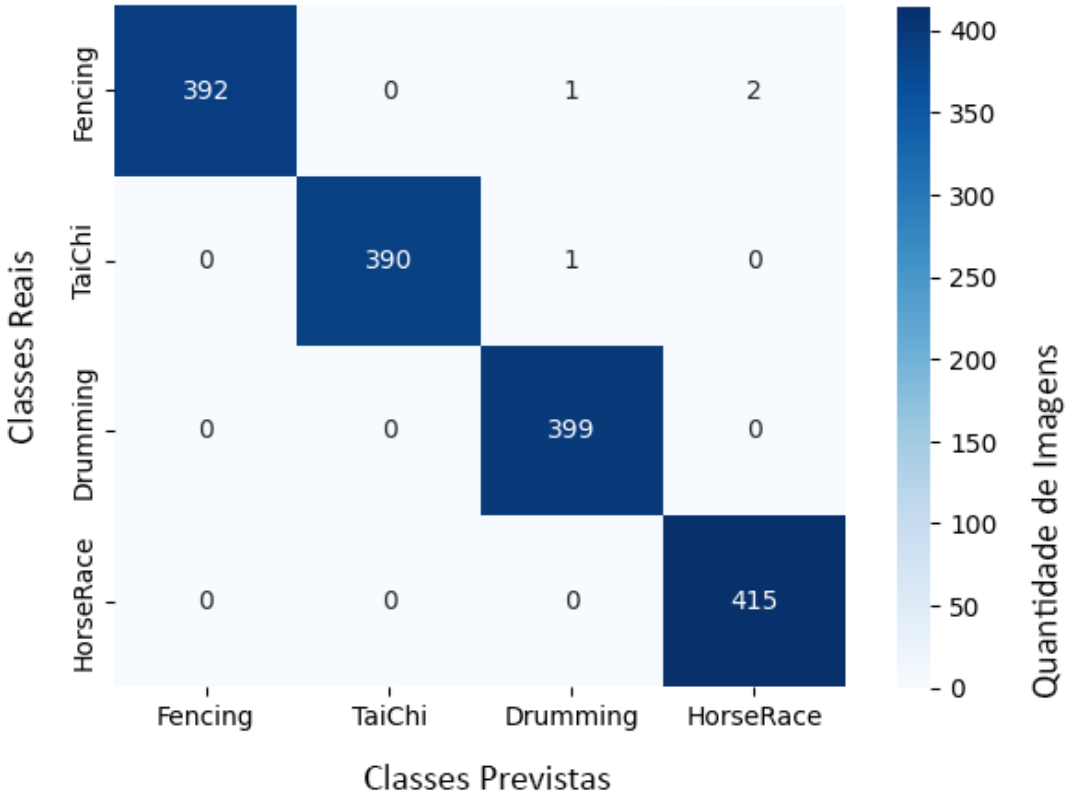
Figura 16 - *val_acuracia* vs. *acuracia* da Rede Quadro a Quadro – UCF50



Fonte: Autoria própria (2023).

Na matriz de confusão gerada pelo teste de maior sucesso, conforme ilustrado na Figura 17, é possível observar a ausência de *overfitting*, devido a taxa de verdadeiros positivos em suas respectivas categorias. Da mesma forma, os valores contidos dentro de cada célula em sua respectiva linha ou coluna, retornam a quantidade de quadros analisados em cada classe durante todo o processo de treinamento e validação.

Figura 17 - Matriz de Confusão da Rede Quadro a Quadro – UCF50



Fonte: Autoria própria (2023).

A partir das métricas e resultados obtidos pelo teste com o UCF50, foi possível aplicar o *Real Life Violence Situations dataset*, utilizando o pré-processamento de remoção, conforme especificado na seção 3.5. A Tabela 2 fornece os resultados das mesmas métricas de testes realizadas no processo anterior, retornando valores significativos e similares.

Tabela 2 - Resultados da Rede Quadro a Quadro com o *Real Life Violence Situations dataset*

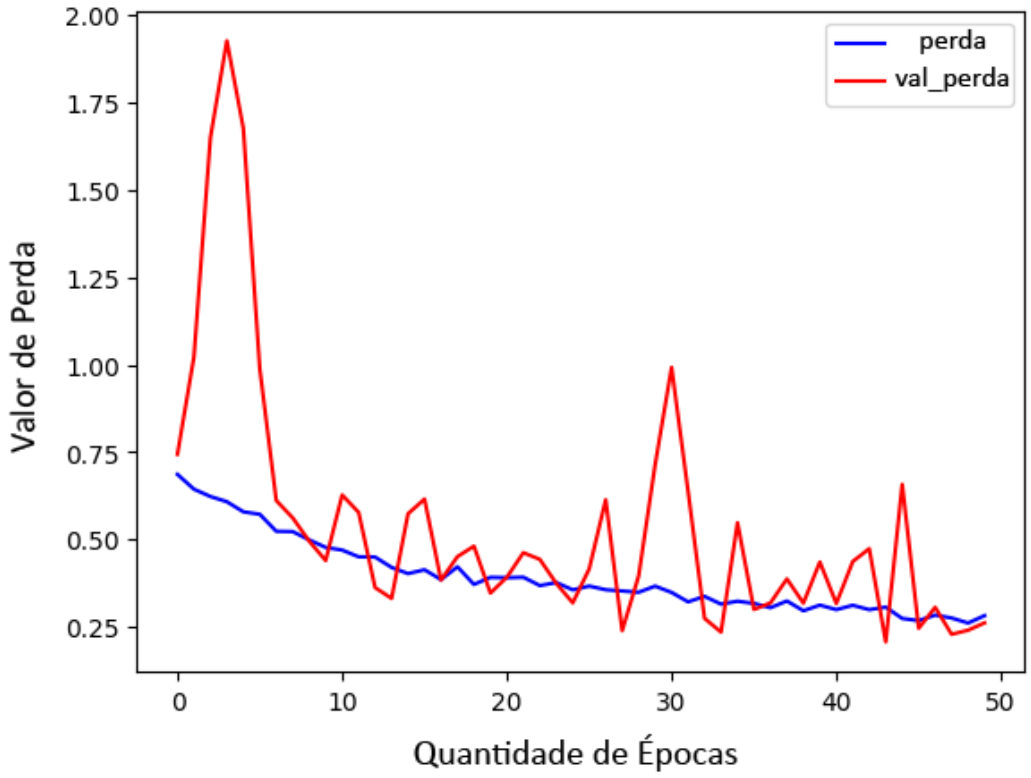
paciência	Número de *Frames*	*val_perda*	val_acuracia
15	10	0,9832	0,7742
20	10	0,8263	0,7854
15	50	0,6198	0,8465
20	**50**	**0,4567**	**0,8761**

Fonte: Autoria própria (2023).

Conforme os resultados obtidos, é possível dizer que a rede neural quadro a quadro precisou de uma extração de 50 imagens por vídeo para chegar ao resultado de 87,61%, fornecendo uma sequência mais detalhada da ação sendo desenvolvida

na cena em questão. A Figura 18 ilustra a curva do *val_perda* vs. *perda*, utilizando as mesmas configurações de desenho dos gráficos anteriores.

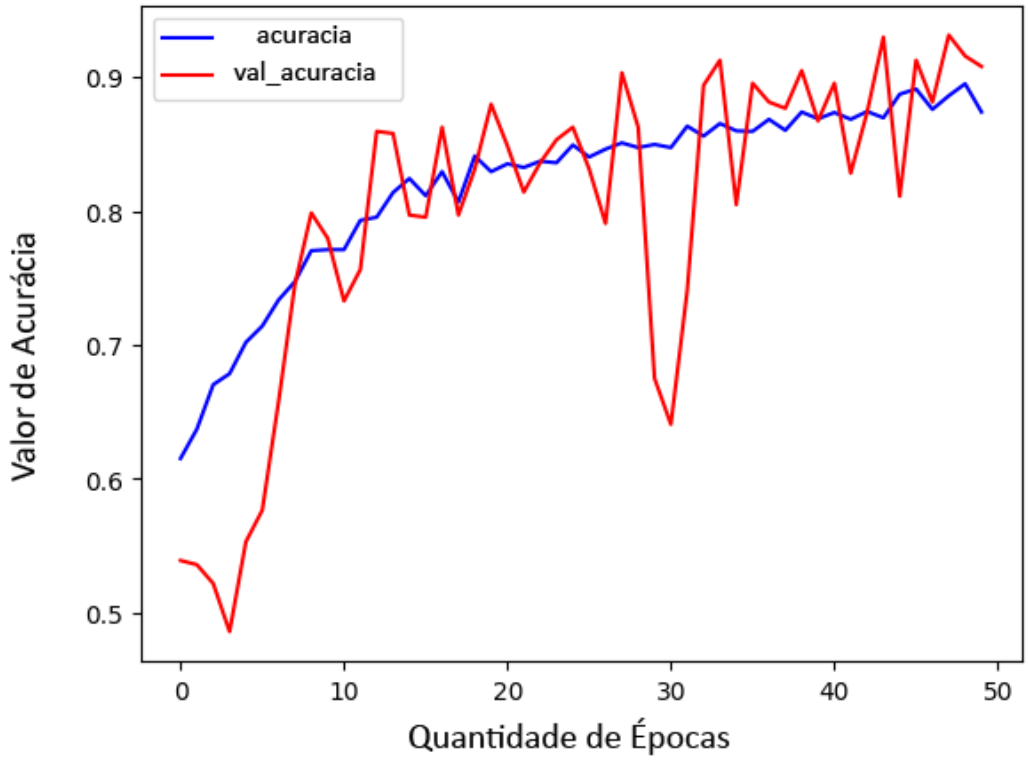
Figura 18 - *val_perda* vs. *perda* da Rede Quadro a Quadro – *Real Life Violence Dataset*



Fonte: Autoria própria (2023).

Seguindo os mesmos preceitos, a Figura 19 ilustra a taxa de *val_acuracia* juntamente com o *acuracia* da abordagem quadro a quadro para os vídeos contidos nas classes *Violence* e *NonViolence*.

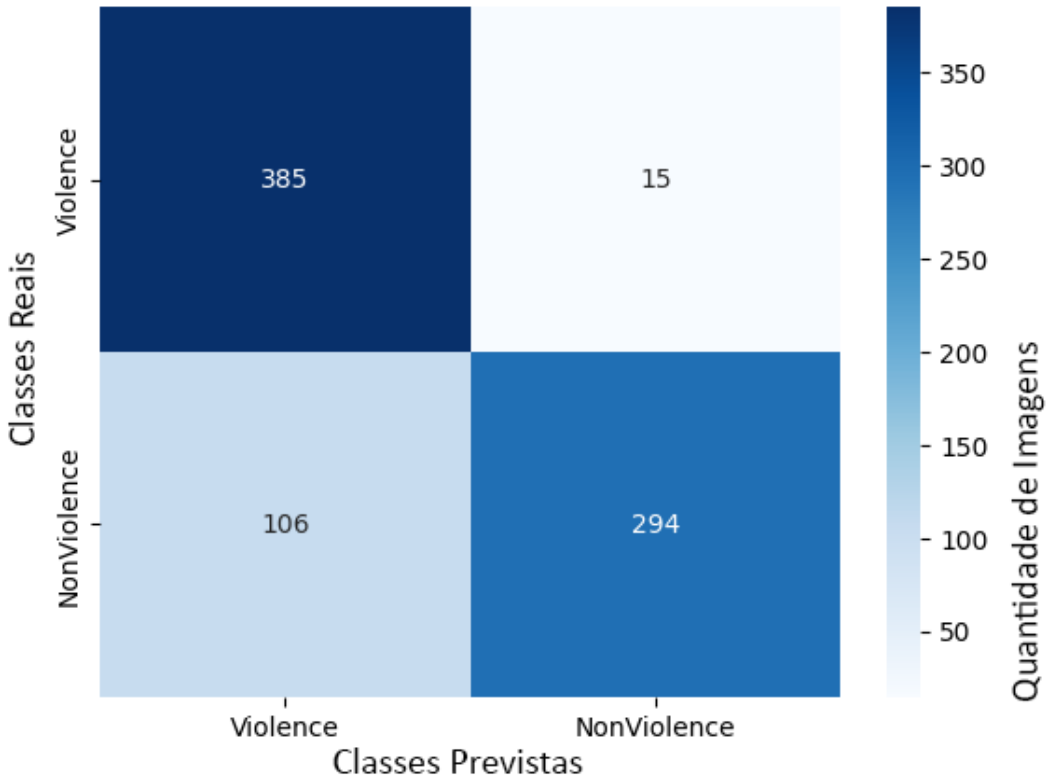
Figura 19 - *val_acuracia* vs. *acuracia* da Rede Quadro a Quadro – *Real Life Violence Situations Dataset*



Fonte: Autoria própria (2023).

De acordo com os gráficos do processo de treinamento e validação, observou-se que a rede desempenhou durante as 50 épocas estabelecidas, provando que a detecção de imagens de violência fornece um desafio para a rede quadro a quadro, perceptível pelo caráter não linear da curva. No qual os picos e quedas do *val_perda* demonstram uma resistência na validação de novas imagens. A seguir na Figura 20, a matriz de confusão demonstra onde a dificuldade da classificação está.

Figura 20 - Matriz de Confusão da Rede Quadro a Quadro – *Real Life Violence Situations dataset*



Fonte: Autoria própria (2023).

Conforme demonstrado na matriz, teve cerca de 25% da classe *Violence* classificada como *NonViolence* no terceiro quadrante, denominado de falso positivo, que acontece quando o elemento treinado classificou como negativo, mas o algoritmo valida como positivo. Isso geralmente acontece quando as características das instâncias são compreendidas como similares entre si, neste caso, imagens que correspondem a vídeos classificados como não violentos, possuem elementos semelhantes com a violência compreendida pelo algoritmo (MEDIUM, 2023).

4.2 Abordagem *ConvLSTM*

Para o treinamento e validação da abordagem *ConvLSTM*, conforme descrito anteriormente, foram utilizadas as mesmas quatro classes do *dataset* UCF50 e mantido a taxa máxima de 50 épocas totais, juntamente com a função *early stopping call-back* fazendo referência ao menor valor do *val_perda* adquirido durante o

processo, a fim de realizar a interrupção quando alcançar as épocas determinadas. Os resultados obtidos podem ser verificados a partir da Tabela 3.

Tabela 3 - Resultados da Rede *ConvLSTM* com o *dataset* UCF50

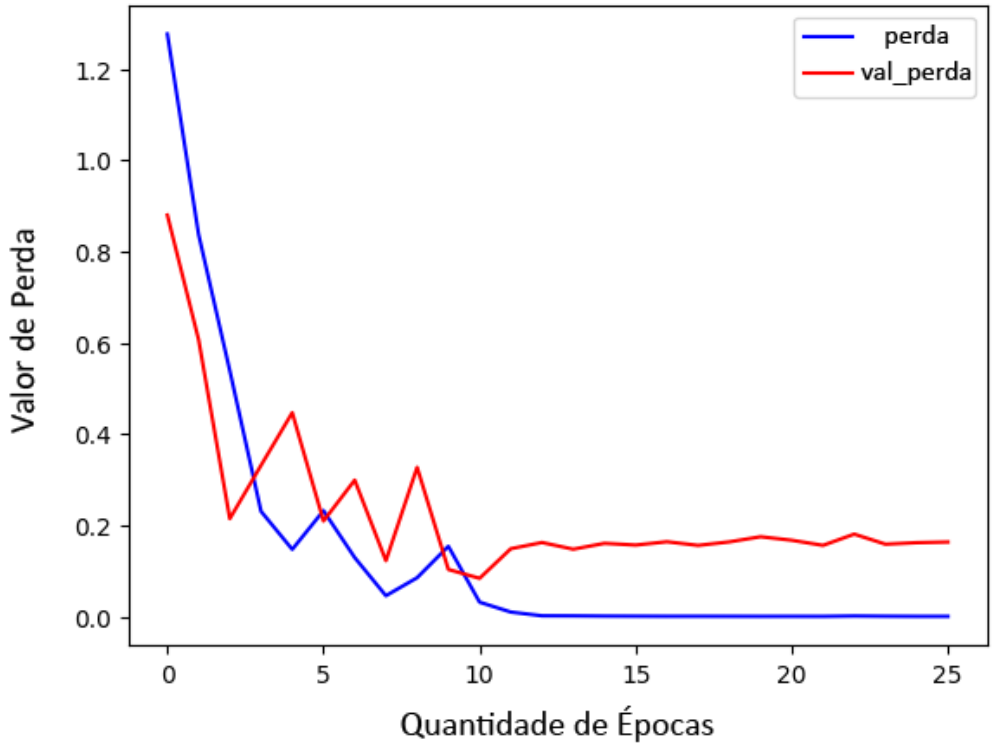
paciência	Número de *Frames*	*val_perda*	val_acuracia
15	10	0,5567	0,9310
20	10	0,3410	0,9376
15	**50**	**0,1998**	**0,9446**
20	50	0,2206	0,9407

Fonte: Autoria própria (2023).

Os resultados de todos os processos feitos com o *dataset* UCF50 surtiram efeitos muito positivos e semelhantes entre si, concretizando rede neural como válida. O resultado que levemente se sobressaiu mediante os demais, foi utilizando uma sequência de 50 *frames* por vídeo de cada classe, fomentando a entrada temporal da arquitetura LSTM, e a paciência de 15 épocas, contabilizando 94,46% de acurácia e uma perda na validação de 0,1998. Na Figura 21 e Figura 22 é possível observar uma certa linearidade quando é alcançado o valor máximo, demonstrando o limite capaz possível da rede *ConvLSTM* para as quatro classes escolhidas da base de dados.

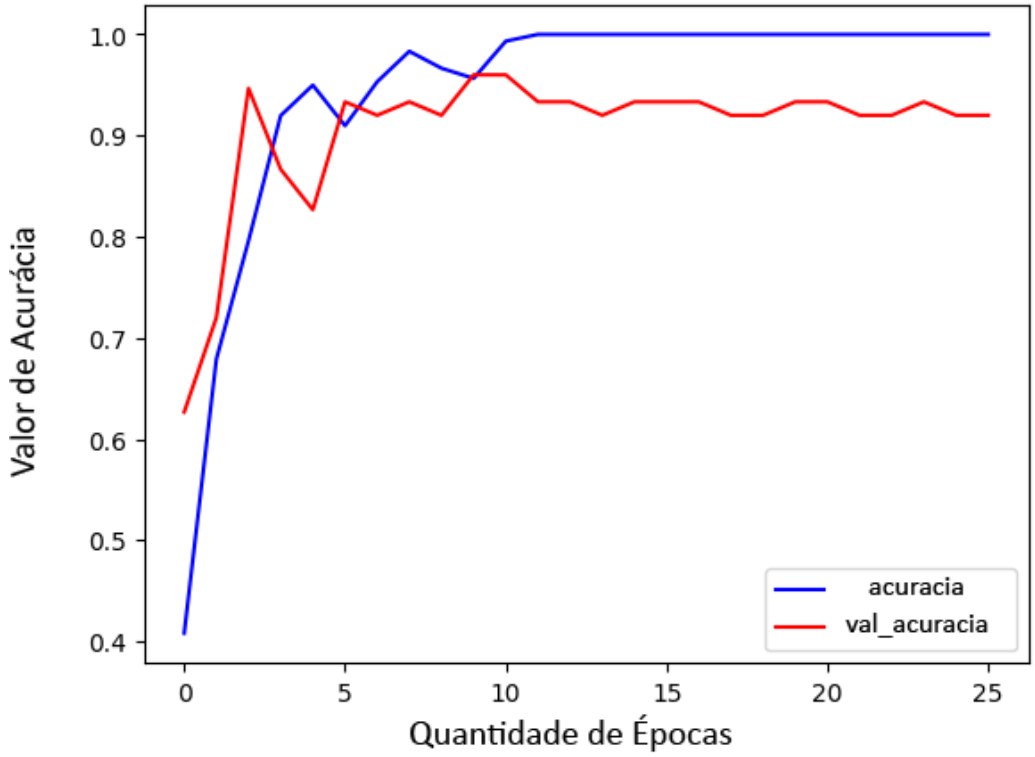
Essa estabilidade é possível de se alcançar quando a rede neural se aproxima de uma configuração ótima, em que a perda é minimizada e a acurácia é maximizada. Um dos motivos de se alcançar tais métricas, é devido à baixa quantidade de entradas da rede, ou seja, o algoritmo conseguiu aprender com facilidade o proposto, sendo necessário a interrupção do processo para prevenir o *overfitting* (KERAS, 2023).

Figura 21 - *val_perda* vs. *perda* da Rede *ConvLSTM* – UCF50



Fonte: Autoria própria (2023).

Figura 22 - *val_acuracia* vs. *acuracia* da Rede *ConvLSTM* – UCF50



Fonte: Autoria própria (2023).

Contabilizando os resultados por meio da matriz de confusão na Figura 23, os valores numéricos contidos em cada célula, diferente da rede quadro a quadro, são os conjuntos de imagens de cada vídeos, ao invés delas separadas. Isso demonstra a capacidade da rede de classificar o vídeo por completo em uma das classes.

Figura 23 - Matriz de Confusão da Rede *ConvLSTM* - UCF50

Fonte: Autoria própria (2023).

Novamente, após o sucesso dos testes com o *dataset* UCF50, teve início o treinamento e a validação da rede neural *ConvLSTM* com o *Real Life Violence Situations dataset*. Os testes realizados por meio das mesmas métricas retornaram os valores demonstrados pela Tabela 4.

Tabela 4 - Resultados da Rede *ConvLSTM* com o *Real Life Violence Situations dataset*

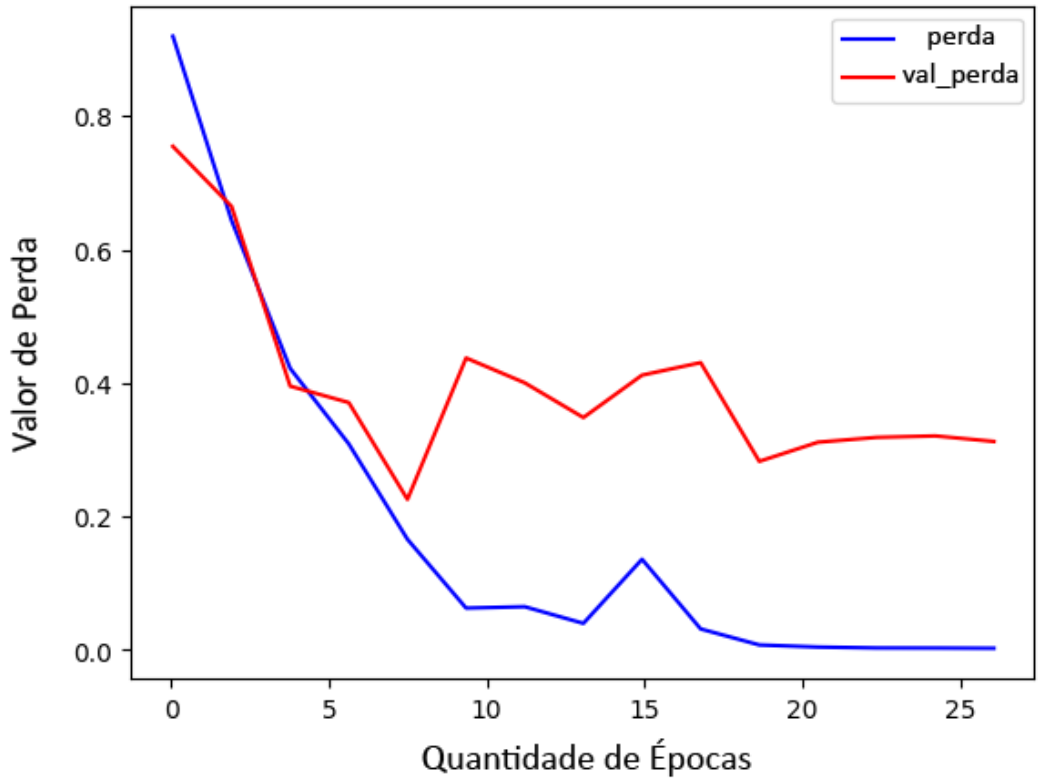
paciência	Número de *Frames*	*val_perda*	val_acuracia
15	10	0.6397	0,8423
20	10	0,6230	0,8412
15	50	0.4486	0.8945
20	**50**	**0.3953**	**0.9002**

Fonte: Autoria própria (2023).

O melhor resultado retornado pela rede *ConvLSTM* foi de 90,02% de acurácia e um valor 0,3953 de *val_perda*. Analisando o gráfico apresentado pela Figura 24, é possível observar uma certa oscilação entre cada época na linha do *val_perda*, mas normalizando na sequência.

Essa discrepância, conforme mencionado anteriormente, deve-se a dificuldade da rede neural de se adaptar com os dados nas primeiras épocas, de conseguir diferenciar entre as duas classes. Entretanto, após um certo ponto foi possível diminuir a perda na validação, resultando em um valor aceitável.

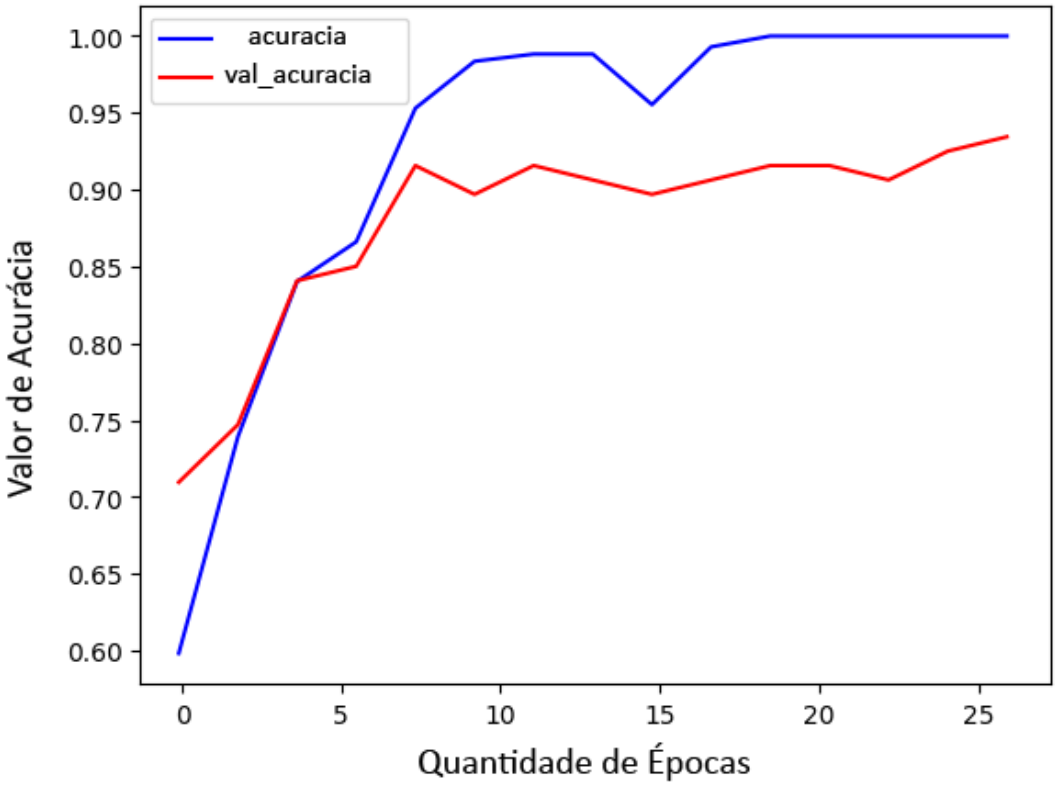
Figura 24 - *val_perda* vs. *perda* da Rede *ConvLSTM – Real Life Violence Situations dataset*



Fonte: Autoria própria (2023).

Já a Figura 25 ilustra o gráfico correspondente a acurácia, mostrando que o *acuracia*, em azul, oscilou junto com a perda na época de número 15, mas logo na sequência conseguiu retornar a sua crescente. Em relação ao *val_acuracia,* não foi afetado de forma significativa por conta dessa oscilação de perda na validação, por conta dessa oscilação ocorrer em valores muito pequenos para influenciar na acurácia total da rede (KERAS, 2023).

Figura 25 - *val_acuracia* vs. *acuracia* da Rede *ConvLSTM – Real Life Violence Situations dataset*



Fonte: Autoria própria (2023).

Para finalizar a análise, a matriz de confusão ilustrada pela Figura 26 mostra uma boa validação por parte da rede *ConvLSTM*, se assemelhando consideravelmente com a rede quadro a quadro.

Figura 26 - Matriz de Confusão da Rede *ConvLSTM – Real Life Violence Dataset*

Fonte: Autoria própria (2023).

A matriz forneceu um número maior de erros no terceiro quadrante (ou quadrante dos falsos positivos), semelhante a abordagem quadro a quadro, conforme descrito anteriormente. Por conta dos resultados obtidos, é válido afirmar que a rede *ConvLSTM* obteve sucesso na classificação de ações de violência.

4.3 Limitações

É importante ressaltar que os erros de classificação podem ser atribuídos à complexidade inerente das ações de violência, que muitas vezes envolvem movimentos rápidos que podem ser desafiadoras até mesmo para um observador humano. Ainda assim, a capacidade da rede *ConvLSTM* em identificar corretamente uma grande parte das ações violentas é notável. Da mesma forma, o desempenho desses modelos está intrinsecamente ligado à qualidade e à representatividade dos dados utilizados no treinamento.

Portanto, a coleta de conjuntos de dados diversificados e de alta qualidade, que representem ampla gama de cenários e condições adversas, é essencial para garantir a robustez e a generalização dos modelos.

5 CONSIDERAÇÕES FINAIS

Tanto a rede neural quadro a quadro quanto a *ConvLSTM* resultaram em valores eficazes na classificação, fornecendo boa capacidade de identificar ações violentas em vídeos. Entretanto, por meio dos gráficos e métricas, foi possível verificar que *ConvLSTM* possui mais espaço para melhorias, tanto em treinamentos mais extensos, quanto com mais dados de entrada ou épocas, o que infelizmente não foi realizado por limitação física do ambiente utilizado.

A habilidade desses modelos em reconhecer e distinguir com precisão as ações violentas reflete a eficácia dos algoritmos de aprendizado de máquina aplicados. Além disso, a obtenção de valores eficazes na classificação evidencia a importância de uma abordagem diferenciada na análise de vídeos, considerando não apenas os quadros individuais, mas também a sequencialidade temporal das informações.

5.1 Conclusão

Os resultados obtidos pela abordagem quadro a quadro ultrapassou 98% de acurácia com o *dataset* UCF50 para as quatro classes escolhidas, enquanto a rede *ConvLSTM* chegou a 94% utilizando cinco épocas de paciência a menos. Este resultado sugere melhor adaptabilidade do algoritmo perante os dados de entrada. Tal comportamento foi semelhante quando aplicado no *Real Life Violence dataset*, que por sua vez retornou 90% para a *ConvLSTM*, ultrapassando os 87% do método quadro a quadro, proporcionando a melhor abordagem do presente estudo.

5.2 Trabalhos Futuros

O treinamento e validação requerem um hardware muito acima do convencional, fornecido pela versão gratuita do *Google Colab*, uma próxima etapa seria obter acesso a máquinas especializadas para tarefas de aprendizado profundo com imagens e vídeos, como servidores, e realizar novamente o treinamento e

validação dos algoritmos, possibilitando o aumento do número e qualidade das entradas da base de dados, assim como a quantidade de processos simultâneos.

Construir uma rede mais robusta, com mais funcionalidades distintas seria uma forma de estabilizar a oscilação das métricas encontradas, finalizando com a aplicação em um sistema de câmeras de segurança em tempo real.

6 BIBLIOGRAFIA

ABDALI. M. R.; R. F. AL-TUMA. **Robust Real-Time Violence Detection in Video Using CNN And LSTM**. 2019 2nd Scientific Conference of Computer Sciences (SCCS). 2019. pp. 104-108, DOI: 10.1109/SCCS.2019.8852616.

ALMEIDA, Claudio de. **Qual é a melhor taxa de frames? 7,5, 15 ou 30 fps?.** 2018. Disponível em: http://www.institutocftv.com.br/qual-a-melhor-taxa-de-frames.html. Acesso em 14 out. 2022.

BABA, M.; GUI, V.; CERNAZANU, C.; PESCARU, D. **A Sensor Network Approach for Violence Detection in Smart Cities Using Deep Learning**. 2019, 19, 1676. https://doi.org/10.3390/s19071676. Acesso em: 30 ago. 2022.

BITARAES, S.; *et al*. 2019. **Previsão de séries temporais em processos de mineração utilizando redes neurais recorrente es e séries temporais nebulosas**. 10.13140/RG.2.2.21361.02407.

CERQUEIRA, Daniel *et al*. **Armas de fogo e homicídios no Brasil 2022**. Rio de Janeiro: [s.n.], 2022. Disponível em: https://forumseguranca.org.br/wp-content/uploads/2022/09/informe-armas-fogo-homicidios-no-brasil.pdf. Acesso em: 30 ago. 2022.

CHENG, M. *et al*. Rwf-2000: **An open large scale video database for violence detection**. 2020 25th International Conference on Pattern Recognition (ICPR). IEEE, 2021.

COLAB. **Conheça o Colab**. 2022. Disponível em: https://colab.research.google.com/ Acesso em: 14 out. 2022.

COMMONS, Wikimedia. **Resolution Illustration**. 2006. Disponível em: https://commons.wikimedia.org/wiki/File:Resolution_illustration.png. Acesso em: 31 ago. 2022.

DATA SCIENCE ACADEMY. **Deep Learning Book**. 2022. Disponível em: http://deeplearningbook.com.br. Acesso em: 30 ago. 2022.

GLOROT, X.; BENGIO, Y. **Understanding the difficulty of training deep feedforward neural networks**. Journal of Machine Learning Research - Proceedings Track, v. 9, p. 249–256, 01 2010.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [S.l.]: MIT Press, 2016. Disponível em: http://www.deeplearningbook.org. Acesso em: 31 ago. 2022.

GRUBLER, M. **Entendendo o funcionamento de uma Rede Neural Artificial**. 2018. Disponível em: https://medium.com/brasil-ai/entendendo-o-funcionamento-de-uma-rede-neural-artificial-4463fcf44dd0. Acesso em: 07 set. 2022.

HAYKIN, S. **Redes Neurais: Princípios e Prática**. Artmed, 2007. ISBN 9788577800865.Disponível em: https://books.google.com.br/books?id=bhMwDwAAQBAJ. Acesso em: 07 set. 2022.

HEATON, J. **Artificial intelligence for humans**. 1. ed. Clarkston, Georgia: Heaton Research, 2015. v. 3. ISBN 978-1505714340.

KERAS. **Keras**. 2022. Disponível em: https://keras.io/. Acesso em: 15 mai. 2023.

KISHORE, K. *et al***. Recognizing 50 Human Action Categories of Web Videos, Machine Vision and Applications Journal** (MVAP), September, 2012. Disponivel em: www.crcv.ucf.edu/data/UCF50_files/MVAP_UCF50.pdf. Acesso em: 07 set. 2022.

LAWRENCE, S. *et al*. **Face recognition**: a convolutional neural-network approach. IEEE Transactions on Neural Networks, v. 8, n. 1, p. 98–113, 1997.

MATPLOTLIB. **Matplotlib: Visualization with Python**. 2022. Disponível em: https://matplotlib.org/. Acesso em 13 fev. 2023.

MEDIUM. **Entendendo o que é Matriz de Confusão com Python**. 2023. Disponível em: https://medium.com/data-hackers/entendendo-o-que-%C3%A9-matriz-de-confus%C3%A3o-com-python-114e683ec509 Acesso em: 01 mai. 2023.

NUMPY. **The fundamental package for scientific computing with Python**. 2023 Disponível em: https://numpy.org/. Acesso em 13 fev. 2023.

ODSC - Open Data Science. **Overview of the YOLO Object Detection Algorithm**. 2018. Disponível em: https://odsc.medium.com/overview-of-the-yolo-object-detection-algorithm-7b52a745d3e0. Acesso em 14 out. 2022

OPENCV. **About OpenCV**. Disponível em: https://opencv.org/about/. 2023. Acesso em 13 fev. 2023.

PEDRINI, H.; SCHWARTZ, W. R. **Análise de imagens digitais: princípios, algoritmos e aplicações.** 2007. São Paulo: Cengage Learning.

POKHREL, Sabina. **Beginners guide to convolutional neural networks**. 2019. Towards data science. Disponível em: https://towardsdatascience.com/beginners-guide-to-understanding-convolutional-neural-networks-ae9ed58bb17d. Acesso em 30 ago. 2022.

PYTHON. **Python 3.11.0 documentation**. 2022. Disponível em: https://docs.python.org/3/. Acesso em 14 out. 2022.

PYTORCH. **PyTorch features**. 2022. Disponível em: https://pytorch.org/features/. Acesso em 14 out. 2022.

REDMON, J. *et al*. **You Only Look Once**: Unified, Real-Time Object Detection. 2015. Cite arxiv:1506.02640. Disponível em: http://arxiv.org/abs/1506.02640. Acesso em 30 ago. 2022

SCIKIT. **Image processing in Python**. Disponível em: https://scikit-image.org/. Acesso em 13 fev. 2023.

SOLIMAN, M. *et al*. **Violence Recognition from Videos using Deep Learning Techniques**, Proc. 9th International Conference on Intelligent Computing and Information Systems (ICICIS'19), Cairo, pp. 79-84, 2019.

SOLOMON, C.; BRECKON, T. **Fundamentals of digital image processing**: a practical approach with examples in matlab. 1. ed. Chichester: Wiley-Blackwell, 2011. ISBN 9780470844724.

SPARK. **Spark Guide**. 2022. Disponível em: https://spark.apache.org/docs/latest/ml-guide.html. Acesso em 14 out. 2022.

TAYLOR, M. **Neural Networks**: A Visual Introduction for Beginners. [S.l.]: Blue Windmill Media, 2017. ISBN 9781549869136.

TENSORFLOW. **TensorFlow Federated**. 2022.
Disponível em: https://www.tensorflow.org/federated. Acesso em 14 out. 2022.

VASILEV, I. *et al*. **Python deep learning**: exploring deep learning techniques and neural network architectures with PyTorch, Keras, and TensorFlow. 2. ed. Birmingham: Packt Publishing, 2019. ISBN 9781789348460.

VIJEIKIS, R. *et al*. **Efficient Violence Detection in Surveillance**. Sensors 2022, 22, 2216. Disponível em: https://doi.org/10.3390/s22062216. Acesso em 30 ago. 2022