BlockFace: Um sistema de segurança com BlockChain

Bernardo M. Fernandes[1], **Eric M. F. Guimarães**[1],
Marcos A. Lommez C. R.[1], **Saulo de M. Zandona F.**[1]

[1]Pontifícia Universidade Católica de Minas Gerais (PUC-MG)
Belo Horizonte – MG – Brasil

[2]Instituto de Ciências Exatas e Informática (ICEI) – PUC Minas

Abstract. *With the continuous advancement of surveillance and security technologies, the need to ensure the integrity and security of captured data is increasingly important. This work explores the integration between camera security systems and blockchain technology, with the goal of recognizing, capturing, and securely storing images of individuals who were previously unregistered. By using blockchain as a storage medium, we ensure that data is immutable and auditable, providing greater transparency and trust in the system. This approach aims to offer an innovative solution to improve the effectiveness and security of monitoring systems.*

Resumo. *Com o crescente avanço das tecnologias de vigilância e segurança, a necessidade de garantir a integridade e a segurança dos dados capturados se torna cada vez mais importante. Este trabalho explora a integração entre sistemas de segurança por câmeras e a tecnologia blockchain, com o objetivo de reconhecer, capturar e armazenar de forma segura imagens de indivíduos previamente não registrados. Utilizando a blockchain como meio de armazenamento, garantimos que os dados sejam imutáveis e auditáveis, proporcionando maior transparência e confiança ao sistema. Essa abordagem visa oferecer uma solução inovadora para melhorar a eficácia e a segurança de sistemas de monitoramento.*

1. Introdução

Nos últimos anos, a segurança eletrônica tem se consolidado como uma ferramenta indispensável na proteção de espaços públicos e privados. Os sistemas de câmeras de vigilância, amplamente utilizados, desempenham um papel crucial na dissuasão de crimes e na coleta de evidências visuais. Contudo, com o avanço da tecnologia, novas vulnerabilidades têm surgido, expondo esses sistemas a riscos como invasões cibernéticas, manipulação de dados e perda de integridade das gravações. Estudos destacam que dispositivos conectados, como câmeras IP, frequentemente possuem falhas exploráveis devido a senhas fracas, configurações padrão e falta de atualizações regulares [Stabili et al. 2024]. Essas vulnerabilidades comprometem a integridade dos sistemas de armazenamento e permitem acessos não autorizados, o que torna necessária a implementação de soluções mais robustas e confiáveis [Biondi et al. 2021].

Nesse contexto, a integração do blockchain aos sistemas de câmeras de segurança desponta como uma solução inovadora. O blockchain, uma tecnologia conhecida por sua descentralização, imutabilidade e rastreabilidade, oferece meios para superar limitações dos modelos tradicionais de segurança. Por exemplo, a utilização de blockchain em sistemas de vigilância permite registros imutáveis que garantem a integridade dos dados, conforme demonstrado em [Gedara et al. 2023], que exploraram o potencial dessa tecnologia para rastrear e proteger informações sensíveis. Além disso, o blockchain oferece um histórico auditável, permitindo o monitoramento seguro de acessos e alterações, essencial para prevenir manipulações e aumentar a confiança no sistema.

Este trabalho propõe o desenvolvimento de um sistema de segurança que utiliza blockchain para integrar câmeras de vigilância a um modelo descentralizado, aprimorando a segurança dos dados e mitigando vulnerabilidades. O objetivo é demonstrar como essa tecnologia pode transformar a segurança eletrônica, proporcionando maior confiabilidade, integridade e transparência nos processos de monitoramento e armazenamento.

2. Revisão Bibliográfica

Para o desenvolvimento da BlockChain e das outras funcionalidades necessárias para o projeto, diversas fontes foram utilizadas, elas serviram de inspiração ou até como partes importantes e essenciais do projeto.

O uso de reconhecimento facial em sistemas de vigilância enfrenta uma série de desafios técnicos e éticos. [Cheng et al. 2018a] discute os obstáculos relacionados à precisão, viés algorítmico e privacidade, destacando como fatores ambientais e a qualidade dos dados podem afetar a eficácia dos sistemas. Esses desafios são cruciais para compreender a aplicação prática de tecnologias de reconhecimento facial em cenários reais.

Para superar limitações tradicionais, abordagens inovadoras têm sido propostas. [Ghani et al. 2024] explora a integração de blockchain e redes adversariais generativas (GANs) em sistemas de reconhecimento facial. O uso de blockchain proporciona maior segurança e privacidade ao armazenar dados, enquanto as GANs ajudam a identificar e mitigar possíveis falsificações de identidade.

Uma análise abrangente das metodologias modernas e desafios do reconhecimento facial é apresentada no estudo de [Oloyede et al. 2020]. Este trabalho revisa os avanços em aprendizado profundo, destacando tanto os progressos em precisão quanto as limitações, como variações de iluminação, expressões faciais e questões relacionadas à proteção de dados pessoais.

A convergência entre blockchain e reconhecimento facial é ainda explorada em [Zhang 2022], que propõe um sistema de registro de presença baseado nessas tecnologias. O artigo demonstra como o blockchain pode ser utilizado para criar registros imutáveis e confiáveis, enquanto o reconhecimento facial assegura a identificação precisa dos participantes, eliminando fraudes.

Esses estudos demonstram a relevância e as possibilidades de integração entre reconhecimento facial e blockchain, ao mesmo tempo em que evidenciam os desafios a serem superados. O presente trabalho busca expandir essas discussões, aplicando tais tecnologias para a criação de um sistema de segurança robusto e confiável.

3. Descrição de arquitetura da Blockchain

A blockchain foi utilizada para salvar dados de um sistema de segurança distribuído, com o objetivo de mantê-los invioláveis e garantir sua integridade. O foco principal é garantir uma arquitetura segura e com tolerância a falhas.

A blockchain de produção própria do grupo, chamada de **Nether** utiliza de um modelo híbrido de consenso que combina **Proof of Work (PoW)** e **Proof of Authority (PoA)**, garantindo segurança e mitigando problemas de eficiência de modelos PoW puros, assim como a liberdade de mudança de líderes que o PoA não permite facilmente.

Figura 1. Arte promocional da Nether

3.1. Topologia Geral

- Líderes estão organizados em uma topologia de **malha parcial**.
- Quando considerados como uma unidade, os demais nós se ligam aos lideres em uma topologia de **estrela**.
- Os nós assim mantém apenas uma lista de líderes e subordinados, com as conexões feitas via **P2P**, de maneira a mitigar processamento desnecessário deixando-os para os nós com mais poder computacional/tempo ocioso.

3.1.1. Nós Líderes

- São eleitos via PoW.
- Permanecem como líderes por um tempo relativamente longo (até serem desligados ou até atingir um tempo X, quando novas eleições serão realizadas).
- Mantém salvos todos os dados históricos da blockchain durante o período em que forem líderes.
- Formam uma topologia de **malha parcial**, em que os líderes têm conexões diretas uns com os outros.

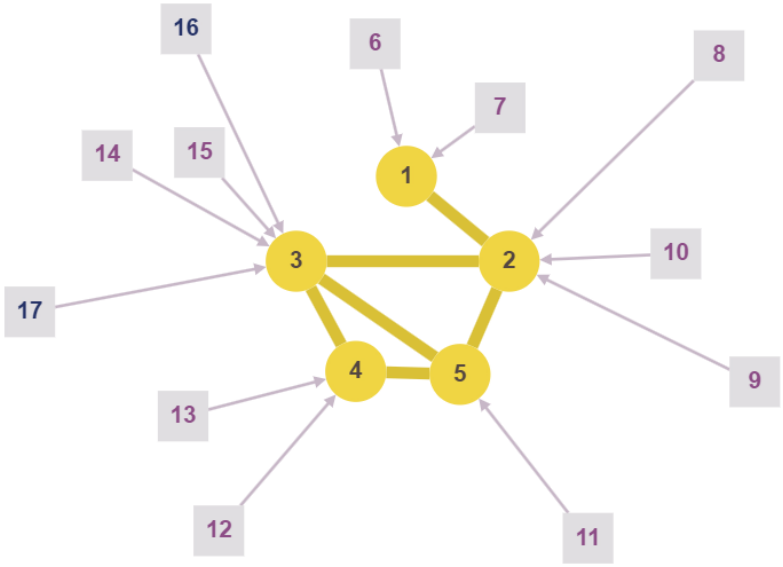

Figura 2. Topologia do Nether
Nós lideres em amarelo e nós subordinados em cinza

3.1.2. Eleições dos Nós Líderes

- As eleições serão iniciadas pelos líderes atuais, e os nós interessados poderão se candidatar avaliando em seu contexto se querem ou não participar como lideres, seja por não terem poder computacional ou estarem ocupados com outras tarefas.
- O primeiro nó a responder será eleito como líder.
- Para evitar condições de corrida, o primeiro líder a responder notifica os outros sobre o encerramento da eleição. Em casos de empate, a decisão é tomada de forma arbitrária pelos lideres.

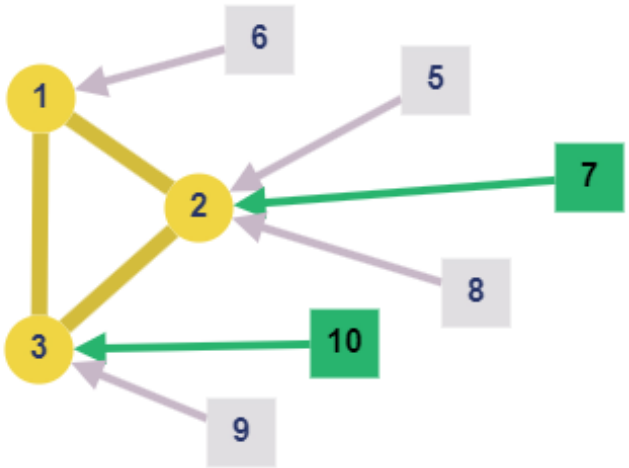

Figura 3. Processo de eleição, candidatos em verde e líderes em amarelo

3.1.3. Nós Subordinados

- Se conectam a um único nó líder.
- Mantém um array contendo dados de outros líderes caso percam a conexão do líder atual,
- Não possuem a obrigação de manter uma cópia completa da blockchain. Eles armazenam apenas a parte que lhes é relevante e o último nó atualizado.

3.2. Segurança e Criptografia

Assinatura Digital e Validação

- Cada bloco é assinado digitalmente pelo líder que o pedir para adicionar à blockchain, utilizando o algoritmo **ECDSA (Elliptic Curve Digital Signature Algorithm)**.
- Cada nó possui uma **chave privada** escolhida durante o registro. A sua **chave pública** resultante é seu próprio nome na rede e **ID**.
- Quando um bloco é inserido na blockchain, o líder responsável assina o bloco digitalmente mantendo sua chave publica no bloco. Os outros líderes validam essa assinatura antes de adicionar o bloco. Em caso de invalidação o líder é automaticamente desconectado.

3.3. Conexão de Nós

Primeira Inicialização de uma Nova Blockchain

- Um nó pode inicializar diretamente como líder para criar uma nova blockchain no primeiro momento de operação da rede.
- Uma rede deve possuir um nome/id único, para evitar desastres em caso de um nó de uma rede se conectar ao nó de outra blockchain compatível. Logo toda conexão inicial verifica a blockchain alvo.

Primeira Conexão de um Nó

- Um nó, ao se conectar pela primeira vez, precisa fornecer manualmente o endereço de outro dispositivo da rede, o qual irá redirecionar o novo nó a um líder caso o mesmo não seja.
- A blockchain se baseia em conexões **IPV6** para conexão sem necessidade de TURN/STUN ou um servidor central.
- Um nó mantém uma lista de nós em stand-by, que ficam aguardando conexão oficial na rede.
- O nó conectado envia uma lista de líderes ao novo nó, que se conecta ao líder com o menor valor de **[tempo de resposta + carga de trabalho]**.

Lista de Seeds

- Após a primeira conexão, o nó mantém em memória primária e secundária uma lista de **seeds** para reconexão, evitando a necessidade de repetir o processo manual.
- Caso a conexão com o líder seja perdida, basta se conectar a outro usando o mesmo critério.

Processo de Recuperação e Reconciliação

- Para garantir a resiliência da rede, cada nó mantém sempre o último bloco válido pois o mesmo possui o dado do último líder, além de manter em memória secundária sua lista de peers e hierarquia, assim possibilitando a recuperação da blockchain em caso de *shutdown* parcial ou completo.
- Após um *shutdown*, um nó tenta se conectar inicialmente ao líder salvo no ultimo bloco inserido de sua cópia da base de dados.
- Em caso de *shutdown* de um líder, o mesmo, ao reconectar, perde o status de líder.
- Caso um líder em reconexão não encontre outros lideres no processo de recuperação a rede permanece em status de espera. Se, após um tempo, um líder não seja encontrado, é realizado um consenso para retomada dos lideres anteriores ativos.
- Após recuperação da conexão é iniciado um protocolo de **reconciliação de dados incoerentes**, garantindo que os dados perdidos ou divergentes sejam resolvidos.
- Se durante o processo de reconexão de um nó for detectado a existência de dados que não estão presentes nos lideres a reconciliação ocorre da seguinte forma:
  - A rede de maior tamanho é mantida.
  - Os nós "perdidos"são analisados e reincorporados à rede maior.

4. Desenvolvimento

O trabalho foi dividido em 3 (três) etapas. Na primeira etapa foi a idealização e formulação do projeto e testes para prova de conceito. Na segunda etapa foram realizadas as implementações da blockchain separadamente e o fine-tuning do reconhecimento facial do FaceNet com o Qmul-Survface. Na terceira e última etapa foi realizada a integração total do projeto e os testes de escalabilidade.

4.1. Etapa 1: Idealização e Formulação do Projeto

Nesta etapa, o foco foi compreender e definir claramente o problema a ser resolvido e as metas do projeto. A fase de idealização incluiu:

- **Identificação de Necessidades**: O ponto de partida foi entender o contexto de segurança e como uma solução baseada em blockchain poderia trazer mais confiança e eficiência ao sistema de câmeras. Realizamos uma análise das necessidades do mercado e de possíveis usuários, identificando as principais vulnerabilidades de sistemas de segurança tradicionais.
- **Pesquisa de Tecnologias**: Exploramos diferentes tecnologias que poderiam ser utilizadas. Consideramos a viabilidade de utilizar blockchain, avaliando suas vantagens para armazenamento seguro e gestão de registros. Também foram discutidas opções de integração com sistemas de câmeras, considerando padrões de comunicação e criptografia.
- **Definição de Escopo e Objetivos**: A idealização culminou na definição do escopo do projeto, estabelecendo as funcionalidades principais, como registro imutável de eventos e auditoria transparente. Além disso, definimos as métricas de sucesso, como segurança dos dados e facilidade de integração

4.2. Etapa 2: Implementação da Blockchain e Fine-Tuning do Reconhecimento Facial

Nesta etapa, os esforços foram concentrados em duas frentes principais: a implementação da blockchain e o ajuste fino (fine-tuning) do modelo de reconhecimento facial.

- **Implementação da Blockchain:**
  - Foi desenvolvida uma blockchain híbrida chamada *Nether*, combinando os modelos de consenso *Proof of Work (PoW)* e *Proof of Authority (PoA)*.
  - A arquitetura foi desenhada com foco na segurança e tolerância a falhas, garantindo:
    - Imutabilidade dos dados por meio de hashs encadeadas dos blocos
    - Assinaturas digitais dos dados inseridos (*ECDSA*).
    - Topologia apropriada para um cenário corporativo privado.
    - Eleições de líderes para assegurar descentralização e distribuição de tarefas computacionais.
- **Fine-Tuning do Reconhecimento Facial:**
  - O modelo *FaceNet* foi ajustado utilizando a base de dados *QMUL-SurvFace*.
  - As camadas iniciais do modelo foram congeladas para preservar os pesos pré-treinados.

4.3. Etapa 3: Integração e Testes de Escalabilidade

A etapa final consistiu na integração das tecnologias desenvolvidas e nos testes de escalabilidade do sistema.

- **Integração:**
  - A blockchain foi integrada ao sistema de câmeras, permitindo o registro seguro e imutável de eventos detectados.
  - O sistema foi configurado para capturar imagens em tempo real, processar os rostos detectados com *YOLO* e *FaceNet*, e armazenar as informações na blockchain.
- **Testes de Escalabilidade:**
  - **Escalabilidade Forte:** Fixou-se o número de operações e o tamanho do problema. A quantidade de threads foi aumentada gradativamente para avaliar o desempenho e identificar o ponto de saturação.
  - **Escalabilidade Fraca:** Incrementou-se simultaneamente o número de threads e o tamanho do problema, garantindo uma análise da eficiência do sistema em diferentes configurações.

5. Prova de Trabalho (Proof of Work)

O desafio do **Proof of Work** é a mineração de um valor **nonce** em que ao concatenar a uma mensagem pre definida deve gerar uma hash de **sha256** que deve ter 'n' zeros a esquerda. Sendo o sha256 projetada para gerar uma distribuição pseudoaleatória, de números na hash de maneira irreversível.

Algoritmo do Problema de Hash Invertida

```go
func ProofOfWork(message string, zeros int) int {
    prefix := strings.Repeat("0", zeros)
    nonce := 0
    for {
        msg := fmt.Sprintf("%s%d", message, nonce))
        hash := sha256.Sum256([]byte(msg))
        if strings.HasPrefix(fmt.Sprintf("%x", hash), prefix) {
            return nonce
        }
        nonce++
    }
}
```

5.0.1. Custo computacional

O algoritmo do **Problema de Hash Invertida** é análogo a problemas NP-Completos pois possui complexidade similar $O(n^2)$ para descoberta de números de zero a esquerda em uma hash qualquer.

O algoritmo de hash gera uma sequencia de 36 bytes, equivalente a 256 bits, o que equivale a um combinatório de 2^{256} valores possíveis. Dessa maneira a quantidade de números existentes com 1 zero a esquerda equivale a 2^{255} números possíveis, e assim sucessivamente como:

Zeros à Esquerda	Espaço de Busca
1	2^{255}
2	2^{254}
3	2^{253}
$\vdots$	$\vdots$
20	2^{236}
$\vdots$	$\vdots$
n	2^{256-n}

Dessa maneira podemos calcular que para um número aleatório qualquer de 256 bits a chance do mesmo possuir algum zero a esquerda é inverso a quantidade de números disponíveis neste espaço com zeros a esquerda. Desta maneira podemos formular que:

Zeros à Esquerda (n)	Números disponíveis	Chance (%)
1	2^1	50%
2	2^2	25%
3	2^3	12.5%
4	2^4	6.25%
5	2^5	3.125%
$\vdots$	$\vdots$	$\vdots$
20	2^{20}	0.000095%
$\vdots$	$\vdots$	$\vdots$
n	2^n	$\frac{100}{2^n}\%$

Desta maneira se torna fácil visualizar a natureza de crescimento exponencial do problema o qual é não determinístico devido a distribuição de valores dentro da hash do sha256. Enquanto por outro lado o processo de testagem para um valor qualquer se dá em tempo $O(1)$, ou seja pode ser testado em tempo P.

5.1. Escalabilidade e poder de paralelismo

A prova por trabalho é facilmente paralelizada por não existir dependência de dados entre as iterações, sendo assim foi atribuído várias threads para a procura de uma string que satisfaça o problema dos zeros, o qual é chamado de **nonce**.

5.1.1. Greenthreads

Devido a natureza da linguagem onde o projeto foi implementada (Golang) o algoritmo foi paralelizado utilizando greenthreads em vez de threads tradicionais. O qual pode ser definida como:

Green threads são threads gerenciadas em nível de usuário, não pelo sistema operacional, permitindo agendamento mais leve e eficiente, mas com limitações no uso de múltiplos núcleos de CPU.

Desta maneira é possível realizar o agendamento de grande número de threads, onde golang possui a capacidade de gerar ate 100.000 threads sem gerar gargalo para o computador.

5.1.2. Ambiente de teste

O ambiente de testes foi configurado com as seguintes especificações:

Componente	Especificação
Sistema Operacional	Linux Mint 21.1
CPU	QUAD CORE MODEL - Intel i5 - 11300H
GPU	RTX 3050
Memória RAM	16GB

5.1.3. Escalabilidade Forte

Para o teste de escalabilidade forte foi fixado o número de zeros procurados em um tamanho que se garante uma baixíssima probabilidade de se encontrar e juntamente fixando o número de operações totais executadas. Desta maneira podemos garantir determinismo na medição de operações por segundo, enquanto o número de threads é aumentado. Após testes os resultados são demonstrados na figura 4.

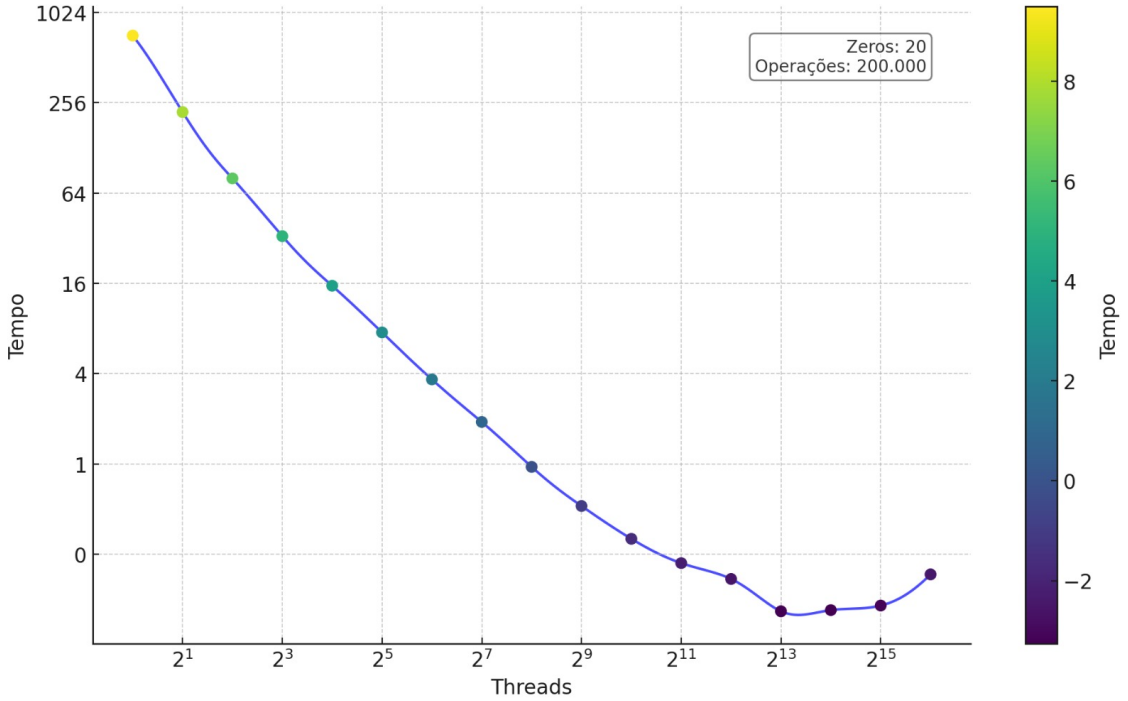

Figura 4. Testes de Escalabilidade Forte

Podemos perceber na figura 4 que de acordo com que o número de threads cresce exponencialmente o tempo de processamento também cai exponencialmente, até que ao chegar em 2^{14} ou 16.384 threads o overhead começa a se tornar mais pesado do que o ganho obtido na paralelização.

Ganhos absurdos como este mesmo com grande quantidade de threads pode ser explicado por 2 fatores:

1. **Degradação do loop:** O processamento contínuo do loop sofre degradação ao longo do tempo, tornando-se mais lento. Como a inicialização do loop é de alta velocidade, paralelizar permite que esta inicialização ocorra repetidamente, mitigando o overhead causado pelo tempo de vida útil do loop.
2. **Baixo overhead das greenthreads:** Greenthreads possuem baixíssimo overhead, pois não geram trocas de contexto ou ações diretamente no sistema operacional.

Com isso conseguimos verificar um speedup de $3917,639$ em comparação de 1 e 16.384 threads. O qual é calculado utilizando a seguinte fórmula

$$\text{Speedup} = \frac{T_1}{T_p}$$

onde:

T_1 é o tempo de execução em um único núcleo/thread.
T_p é o tempo de execução com múltiplos núcleos/threads.

5.1.4. Escalabilidade Fraca

Para o problema da escalabilidade fraca é necessário realizar o crescimento do numero de threads juntamente com o tamanho do problema, e para isso faz-se necessário uma analise mais profunda da probabilidade de se encontrar um número na prova por trabalho.

A probabilidade de sucesso em uma tentativa aleatória é dada por:

$$P = \frac{1}{2^n}$$

Onde:

- n: Número de zeros à esquerda.

A partir de um conjunto de tentativas aleatórias precisamos fixar uma probabilidade fixa de dado um número de operações de encontrar a hash alvo

A fórmula para calcular o número esperado de operações (E) para alcançar uma probabilidade p é dada por:

$$E = \frac{\ln(1-p)}{\ln\left(1-\frac{1}{2^n}\right)}$$

Onde:

- p: Probabilidade alvo (ex.: 5%, 80%).
- n: Número de zeros à esquerda.

Assim após decidir arbitrariamente por uma probabilidade, podemos calcular o número de threads necessárias para realizar a escalabilidade fraca. O número de threads necessárias (N_t) é dado por:

$$N_t = \left\lceil \frac{E}{T} \right\rceil$$

Onde:

- E: Número esperado de operações.
- T: Operações realizadas por thread.

Assim definimos valores constantes e calculamos para n variável, $p = 5\%$, e $T = 100.000$, chegando a seguinte tabela:

Zeros (n)	Operações	Threads
16	6723	1
17	13446	2
18	26892	3
19	53784	6
20	107569	11
21	215139	22
22	430279	44
23	860558	87
24	1721117	173
25	3442234	345
26	6884469	689
27	13768938	1377
28	27537877	2754
29	55075755	5508

Agora fixado o número de threads juntamente com o número de operações podemos calcular a escalabilidade fraca que se mostra na figura 5.

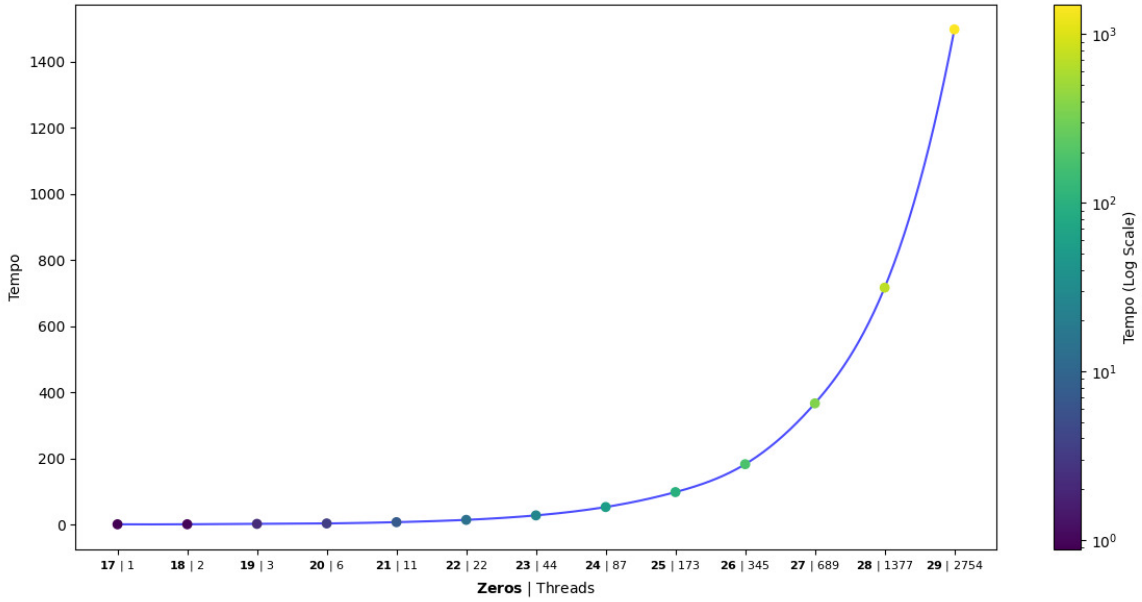

Figura 5. Testes de Escalabilidade Fraca

6. Processamento e Análise de Imagens

A aplicação realiza a detecção de rostos em tempo real utilizando a Rede Neural Convolucional (CNN) YOLO, que divide a imagem em regiões e prevê caixas delimitadoras e classes simultaneamente[V7 Labs 2024]. Aplicada juntamente dos algoritmos de NMS para boxes duplicados e DeepSorte para tracking do mesmo rosto de acordo com a passagem do rosto pelos frames.

Após a detecção, a imagem é recortada e enviada ao *FaceNet*, um modelo de rede neural do Google para reconhecimento facial. O *FaceNet* cria embeddings de rostos, posicionando rostos semelhantes próximos no espaço vetorial. Utilizando a técnica *triplet loss*, ele minimiza a distância entre rostos iguais e maximiza entre diferentes, proporcionando alta precisão no reconhecimento[Schroff et al. 2015].

O modelo *FaceNet* foi fine-tuned usando a rede Inception-ResNetV1 e a biblioteca PyTorch, congelando as primeiras camadas para preservar os pesos do modelo pré-treinado. O ajuste foi realizado com a base de dados *QMUL-SurvFace* para melhorar o desempenho em cenários de vigilância

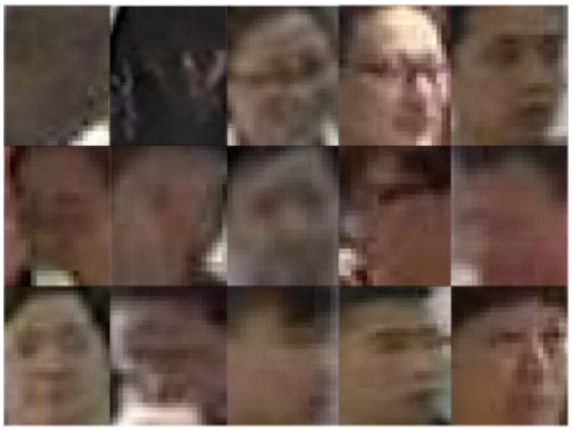

Figura 6. Exemplo de imagens do QMUL-SurvFace

Dados sobre o modelo pré-treinado FaceNet

O modelo *FaceNet* foi treinado inicialmente em um conjunto de dados contendo aproximadamente **200 milhões de imagens de 8 milhões de indivíduos**, utilizando uma rede neural do tipo *Inception-ResNetV1*. O treinamento foi realizado em múltiplas bases de dados, incluindo *CASIA-WebFace* e *VGGFace2*, com o objetivo de obter representações vetoriais de alta qualidade para reconhecimento facial.

O modelo obteve resultados impressionantes no benchmark *LFW* (Labeled Faces in the Wild), um conjunto de dados que contém faces de pessoas em condições variadas de iluminação, expressão e pose. O FaceNet alcançou uma acurácia de **99,63%** no LFW, uma das mais altas já registradas para essa tarefa.

Resultados de Acurácia do FaceNet:

- **Base de dados de treinamento:** *CASIA-WebFace*, *VGGFace2*.
- **Acurácia no LFW (1:1): 99,63%**.
- **Número de imagens de treinamento:** 200 milhões.
- **Número de indivíduos treinados:** 8 milhões.

Fine-tuning com QMUL-SurvFace

O *fine-tuning* do modelo FaceNet foi realizado utilizando a base de dados *QMUL-SurvFace* [Cheng et al. 2018b]., composta por **463.507 imagens de 15.573 indivíduos**. Esta base de dados contém imagens de rostos capturadas em cenários reais e não planejados, com variações de iluminação, ângulos e expressões faciais. Para o fine-tuning,

as primeiras camadas do modelo FaceNet foram congeladas para preservar os pesos pré-treinados, enquanto as camadas superiores foram ajustadas para melhorar a precisão do reconhecimento facial em condições de vigilância.

Após o processo de *fine-tuning*, o modelo obteve uma **acurácia de 100%** no conjunto de dados de teste que possuía 46.351 imagens.

7. Conclusão e Trabalhos Futuros

O trabalho alcançou com satisfação todas as propostas apresentadas e obteve resultados ótimos tanto em acurácia de fine-tuning do modelo quanto na escalabilidade do código de prova por trabalho fornecido. Com uma blockchain que atende o ambiente corporativo com eficácia para o problema proposto.

7.1. Trabalhos Futuros

Futuras melhorias podem incluir:

- Implementação de mais funcionalidades da blockchain.
- Testes de invasão e resiliência da rede distribuída.
- Adicionar mais possibilidades de conexão além de IPV6

Referências

Biondi, P., Bognanni, S., and Bella, G. (2021). Vulnerability assessment and penetration testing on ip camera. In *2021 8th International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*. IEEE.

Cheng, Z., Zhu, X., and Gong, S. (2018a). Surveillance face recognition challenge.

Cheng, Z., Zhu, X., and Gong, S. (2018b). Surveillance face recognition challenge. *arXiv preprint arXiv:1804.09691*.

Gedara, K., Nguyen, M., and Yan, W. (2023). *Enhancing Privacy Protection in Intelligent Surveillance: Video Blockchain Solutions*, pages 42–51.

Ghani, M. A. N. U., She, K., Rauf, M. A., Khan, S., Khan, J. A., Aldakheel, E. A., and Khafaga, D. S. (2024). Enhancing security and privacy in distributed face recognition systems through blockchain and gan technologies. *Computers, Materials and Continua*, 79(2):2609–2623.

Oloyede, M., Hancke, G., and Myburgh, H. (2020). A review on face recognition systems: recent approaches and challenges. *Multimedia Tools and Applications*, 79.

Schroff, F., Kalenichenko, D., and Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, page 815–823. IEEE.

Stabili, D., Bocchi, T., Valgimigli, F., and Marchetti, M. (2024). Finding (and exploiting) vulnerabilities on ip cameras: the tenda cp3 case study.

V7 Labs (2024). Yolo object detection: The most popular deep learning algorithm for real-time object detection. Acesso em: 26 nov. 2024.

Zhang, Q. (2022). Attendance system based on blockchain and face recognition. In *2022 International Conference on Smart Applications, Communications and Networking (SmartNets)*, pages 1–6.