ALUNO: LUIZ MARCIO FARIA DE AQUINO VIANA, M.Sc.
DRE: 120048833
CPF: 024.723.347-10
RG: 08855128-8 IFP-RJ
REGISTRO: 2000103581 CREA-RJ
TELEFONE: (21)99983-7207
E-MAIL: luiz.marcio.viana@gmail.com

DISCIPLINA: CPS730 – INTERNET DAS COISAS
PROFESSOR: CLAUDIO MICELI
DATA DA ENTREGA: 18/07/2022

TRABALHO II - *INTERNET OF THINGS (IoT)*
APLICAÇÕES DA BIBLIOTECA TINYML EM INTERNET DAS COISAS

ÍNDICE

1. MOTIVAÇÕES E OBJETIVOS

Este estudo apresenta as principais pesquisas que foram realizadas com o objetivo de reduzir a complexidade dos modelos de *machine learning (ML)*, possibilitando a aplicação destes modelos em microcontroladores com capacidade limitada de processamento e memória principal.

Aplicando os métodos elaborados de *tiny machine learning (TinyML)*, podemos utilizar modelos de *machine learning (ML)* em microcontroladores com capacidade limitada de processamento e memória.

Seguem dois exemplos de microcontroladores com capacidade limitada de processamento e memória principal, que suportam bibliotecas otimizadas de *tiny machine learning (TinyML)*.

(a) Placa Arduino UNO R3, com microoprocessador ATmega328P, de 8 bits, 16 MHz, 32 KB de memória Flash, 1 KB de memória EEPROM e 2 KB de memória RAM (Figura 1).

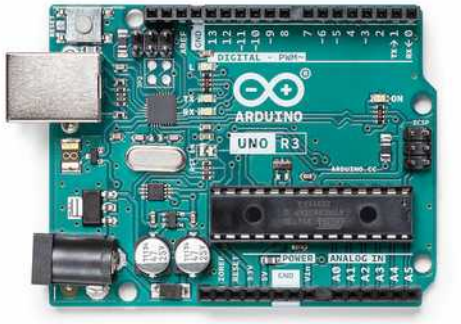

Figura 1: Placa Arduino UNO R3 com Processador ATmega328P, 8 bits, 16 MHz.

(b) Placa WiFi LoRa 868 MHz ESP32 com OLED, com microprocessador Tensilica Xtensa LX6, de 32 bits, 2 núcleos, 240 MHz, 4 MB de memória Flash interna, 1 KB de memória eFuse, 16 KB de memória SRAM temporária, 520 KB de memória SRAM para processamento, e com suporte para até 16 MB de memória Flash externa (Figura 2).

Figura 2: Placa WiFi LoRa 868 MHz ESP32 com OLED, 32 bits, 240 MHz.

1.1. OBJETIVOS

O principal objetivo da Pesquisa para Tese D.Sc., é desenvolver um mecanismo de armazenamento otimizado para suportar grandes volumens de imagens e vídeos, que são capturados por microcontroladores localizados em diversas áreas e que são pré-classificados e enviados para o *data center* (Figura 3).

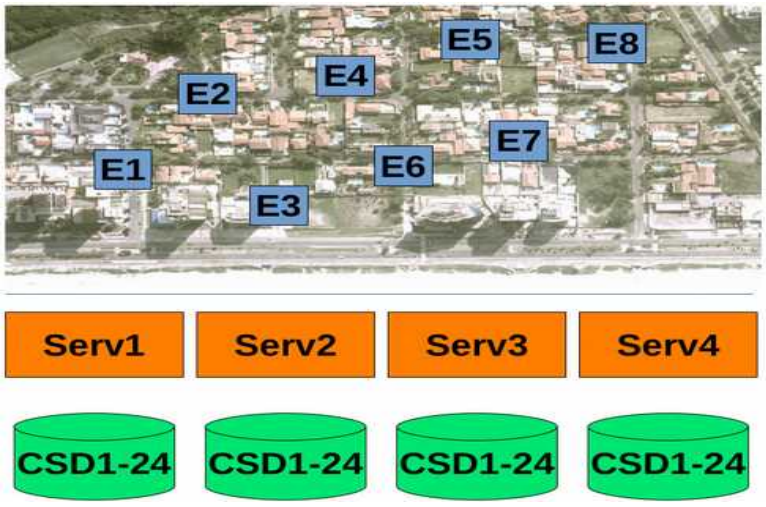

Figure 3: Captura de imagens por câmeras de vigilância, pré-classificação e envio para o *data center*.

Neste processo, a realização da pré-classificação dos dados pelos equipamentos de vigilância é fundamental para a redução do volume total dos dados trafegados pela Internet até o data center.

Estudos realizados, apresentam uma redução de até 20 vezes no volume total de dados trafegados pela Internet, com a pré-classificação dos dados realizada pelos microcontroladores na borda *(edge computing)*.

Desta forma, a utilização dos recursos de *tiny machine learning (TinyML)* pelos microcontroladores na borda, possibilita a utilização de modelos otimizados de redes neurais profundas, que fornecem maior precisão na classificação das imagens.

2. CONCEITOS DE INTELIGÊNCIA COMPUTACIONAL

Podemos definir Inteligência Computacional, como a capacidade de incluir nos sistemas computacionais recursos que possibilitam o aprendizado e classificação de objetos com base na avaliação de resultados anteriores.

Desta forma, a Inteligência Computacional é usada nas situações onde não existe uma expressão matemática direta para resolver um problema, sendo necessário construir um modelo computacional com base nos resultados obtidos do processamento dos dados.

Por exemplo, como podemos reconhecer padrões e identificar números, elementos e rostos em fotos? Para isso, existe somente uma resposta, que é encontrar uma função matemática capaz de fornecer um valor que representa o quanto um rosto em uma foto se assemelha ao rosto de uma pessoa previamente analisada.

Outro exemplo, como uma máquina pode identificar a sua posição atual, reconhecer os objetos em sua volta, e decidir o melhor caminho até chegar em um objeto alvo? Para isso, é necessário coletar a posição atual da máquina, identificar os objetos em sua volta usando um modelo de Inteligencia Computacional para reconhecimento dos objetos, e em seguida calcular uma sequencia de rotas possíveis para se chegar ao objeto alvo. Como existem infinitas alternativas de movimento no plano em R^2, normalmente aplicamos uma Euristica para encontrar uma boa rota em um tempo viável.

Analisaremos a seguir, três modelos de aprendizado de máquina. O modelo linear *(linear model)*, *multi-layer perceptron (MLP)*, e *neural networks (NN)*.

A. *Linear Model*

A Figura 4, apresenta três modelos lineares *(linear models)* de aprendizado de máquina. Nestes modelos, usamos um conjunto de dados para treinamento do modelo, isto é, definir os valores de cada parâmetro do modelo linear.

Podemos classificar os modelos lineares em três grupos (Figura 4): *Perceptron*, *Linear Regression*, *Logistic Regression*.

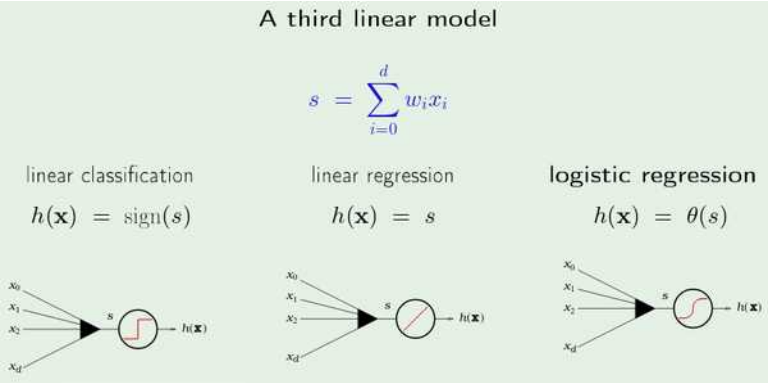

Figure 4: Modelos lineares de aprendizado de máquina.

(a) *Perceptron*: O modelo *perceptron* é usado em situações onde precisamos classificar diretamente um objeto. Por exemplo, podemos usar um modelo *perceptron* para aprovar ou negar credito de uma pessoa.

(b) *Linear Regression*: O modelo de regressão linear é usado em situações onde precisamos encontrar um valor apropriado em função de dados históricos. Por exemplo, podemos usar um modelo de regressão linear para determinar qual o limite de crédito de uma pessoa.

(c) *Logistic Regression*: O modelo de regressão logistica é usado em situações onde precisamos encontrar um valor capaz de fornecer uma ideia de probabilidade de ocorrência de um evento. Por exemplo, podemos usar um modelo de regressão logística para determinar se uma determinada pessoa pode ou não ficar inadiplente após conceder um empréstimo.

B. *Multi-layer Perceptron (MLP)*

O modelo linear pode ser combinado e processado em camadas, onde a camada anterior, fornece os valores de entrada da camada seguinte. Este modelo linear com múltiplas camadas é denominado *multi-layer perceptron* (Figura 5).

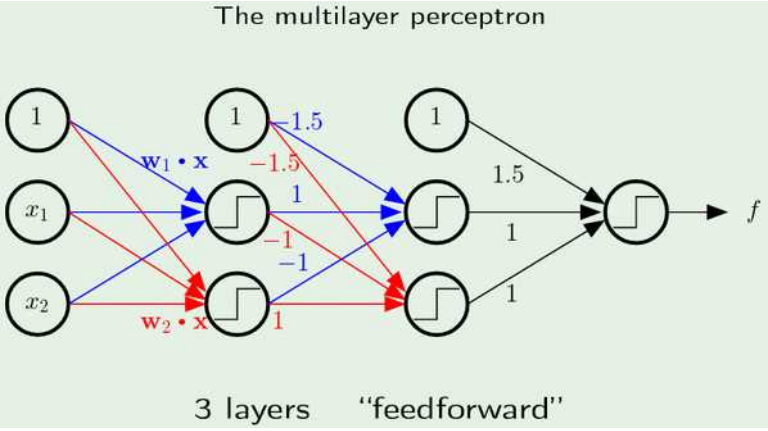

Figure 5: Modelo *multi-layer perceptron*.

O modelo *multi-layer perceptron* é um recurso muito poderoso, capaz de classificar de forma bastante precisa os dados com o crescimento no número de camadas (Figura 6).

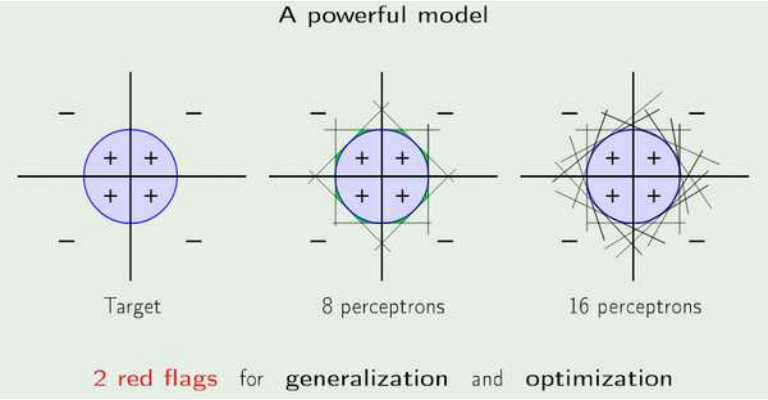

Figure 6: Exemplos de modelo *multi-layer perceptron* com 8 e 16 *perceptrons*.

C. *Neural Networks*

Os modelos de redes neurais é um recurso muito mais poderoso que o modelo *multi-layer perceptron*, porque cada nó de uma rede neural representa uma função de transformação *(transformation function)*, capaz de se ajustar de forma muito precisa aos dados (Figura 7).

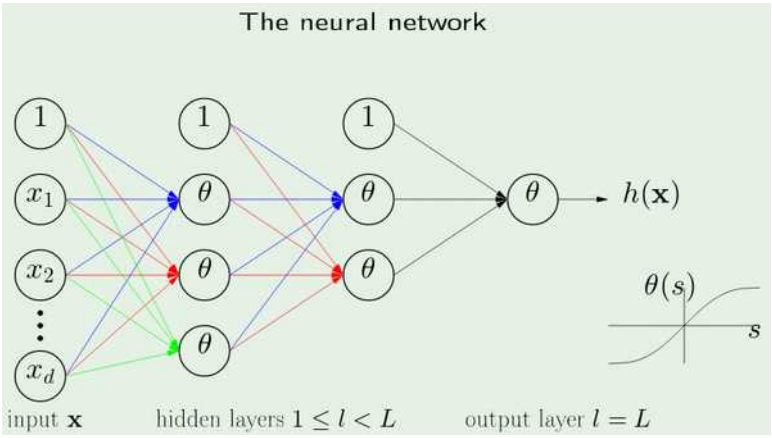

Figure 7: Exemplo de modelo de rede neural com as funções de transformação em cada nó.

Os algoritmos de treinamento de redes neurais efetuam várias iterações, procurando ajustar cada vês mais os parâmetros aos dados de entrada. Esta operação é muito demorada porque é necessário recalcular os todos os parâmetros em cada iteração.

2.1. REDES NEURAIS PROFUNDAS *(Deep Neural Networks - DNN)*

Redes neurais profundas *(deep neural networks - DNN)*, são modelos de redes neurais fortemente conectadas que apresentam uma camada de entrada, duas ou mais camadas ocultas e uma camada de saída. A Figura 8, apresenta um modelo de redes neurais profundas.

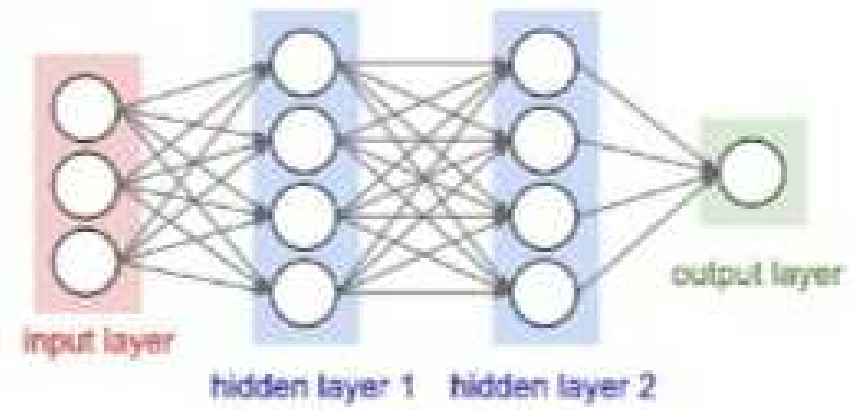

Figure 8: Modelo de redes neurais profundas *(deep learning)*.

2.2. REDES NEURAIS CONVOLUCIONAIS *(Convolutional Neural Networks - CNN)*

As redes neurais convolucionais *(convolutional neural networks - CNN)*, são modelos de redes neurais profundas que são largamente utilizadas na classificação de imagens e áudio. Este modelo de rede neural procura separar em cada camada as características que definem as imagens, permitindo a identificação dos dados. A Figura 9, apresenta um modelo de rede neural convolucional aplicada ao reconhecimento de imagens.

Figure 9: Modelo de redes neurais convolucionais *(convolutional neural networks)*.

A. Redes Neurais Convolucionais

Ao analisarmos imagens de 256 cores usando redes neurais, podemos arrumar os elementos da imagem em um vetor. Desta forma, uma imagem de 28 x 28 pixels e 256 cores, seria representada por um vetor com 784 bytes, e utilizaríamos na camada inicial da rede neural 784 entradas (Figura 10).

Quando estudamos imagens com maior quantidade de cores, como imagens RGB, o tamanho do vetor necessário para representar a imagem é de 2352 bytes. Além disso, ao representar uma imagem por um vetor de dados, perdemos as características associadas a cada profundidade de cor, gerando um modelo com menor taxa de assertividade.

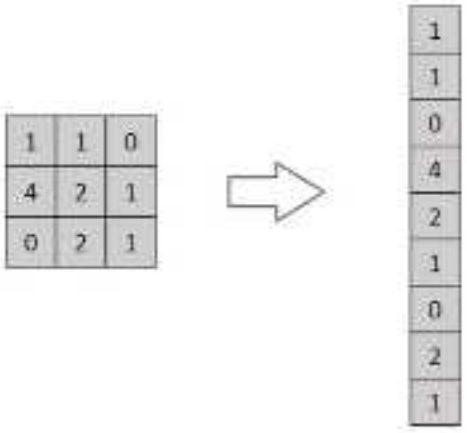

Figure 10: Organização dos *pixels* da imagem em vetor.

Desta forma, sugiram as redes neurais convolucionais, que separam os dados de acordo com a profundidade de cor aplicada as imagens, assegurando um melhor desempenho na classificação das imagens (Figura 11).

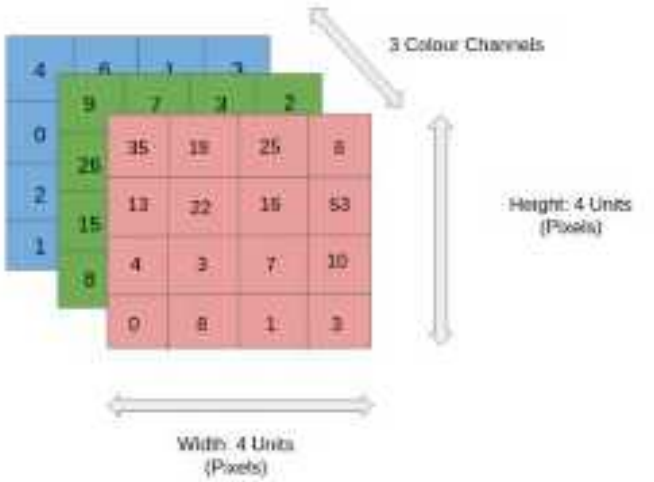

Figure 11: Organização dos *pixels* da imagem por canal de cor.

B. Camadas Convolucionais *(Kernel)*

Cada camada convolucional do modelo de redes neurais implementa uma função de filtro denominada *Kernel*, que procura extrair os detalhes da imagem. Esta operação é realizada na imagem e reduz a quantidade de informações para um número específico de elementos. A Figura 12 apresenta o processo de filtro aplicado por uma camada convolucional.

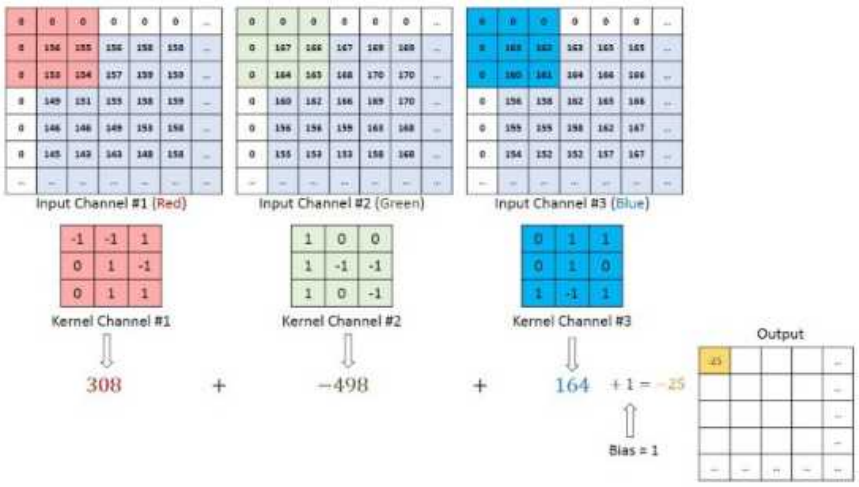

Figure 12: Processo de filtro aplicado pela camada de convolução.

O filtro é aplicado em cada canal de cor, e o resultado obtido da camada de convolução é uma matriz com um número reduzido de elementos. No processo de convolução, as operações mais comuns aplicadas pelos filtros, são operações de identificação de valores médios ou máximos da camada de entrada.

A análise das imagens para construção de um modelo de redes neurais convolucionais (CNN), é um processo que demanda uma grande capacidade de processamento, que não existe nos microcontroladores atuais.

Além disso, este processo é extremamente demorado, e a taxa de erro do modelo reduz com o aumento no número de execuções do algorítmo, processo denominado Épocas de treinamento do algorítmo *(Epoch)*.

A Figura 13, apresenta um gráfico de redução das taxas de erro, obtido usando um modelo com 4 camadas de convolução e com uma saída de 32, 64 e 128 elementos gerados pela operação de convolução (M).

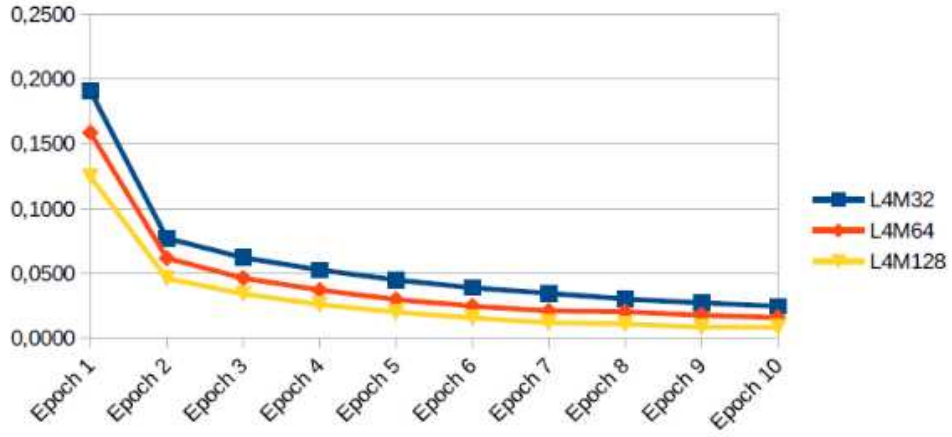

Figure 13: Resultados obtidos com 4 camadas de convolução.

Durante a Pesquisa para Tese D.Sc., estou analisando soluções de Inteligência Computacional usando redes neurais profundas *(deep neural networks - DNN)*, para reconhecimento de padrões nas imagens.

Além de outras soluções com objetivo de detectar mudanças e classificar texturas em imagens geradas por sistemas de monitoramento por satélite e drones (Figura 14).

Figura 14: Exemplos de classificação de texturas, detecção de mudanças e reconhecimento de padrões usando Inteligência Computacional.

A utilização de modelos otimizados de aprendizado de máquina, *tiny machine learning (TinyML)*, permite que microcontroladores com pouca capacidade de processamento e memória, como as Placas Arduino UNO R3 (Figura 1), e WiFi LoRa 868 MHz ESP32 com OLED (Figura 2) efetuem a pré-classificação dos dados antes de enviar para o *data center*, reduzindo em até 20 vezes o volume de dados trafegados pela Internet.

2.3. REDES NEURAIS PARA MICROCONTROLADORES *(TinyML)*

As redes neurais convolucionais usadas na classificação de elementos nas imagens, ocupam grande espaço em memória, e para usar estes modelos em microcontroladores, é necessário efetuar a otimização do modelo de redes neurais.

O processo de otimização das redes neurais profundas (DNN) para permitir o seu uso em microcontroladores é denominado *tiny machine learning (TinyML)*, e envolve algumas etapas,

ETAPA 1: Identificação dos termos do modelo de redes neurais que possui maior importancia na classificação dos dados, processo denominado de identificação e aprimoramento *(knowledge distillation)*.

ETAPA 2: Poda do modelo *(prunning)*, isto é, eliminação dos termos de menor importância na classificação dos dados.

ETAPA 3: Normalização dos parâmetros *(quantization),* permitindo a representação dos termos por valores inteiros de 8, 16 ou 32 bits. A utilização de valores inteiros no lugar dos valores de ponto flutuante, otimiza os cálculos durante o processo da classificação dos dados.

ETAPA 4: Agrupamos os padrões de bits iguais *(encoding)*, reduzindo o tamanho do modelo representado em memória. Padrões de bits iguais são representados por seu código interno, obtido da tabela de padrões de bits previamente identificados no modelo.

ETAPA 5: Escrevemos o modelo em uma linguagem de alto nível *(compilation)* como C/C++, Java ou Python, para poder inserir no código da aplicação que irá executar no microcontrolador.

3. PROCESSOS DE OTIMIZAÇÃO DE REDES NEURAIS

Descreveremos a seguir o processo de otimização de redes neurais profundas (DNN), detalhando as 5 etapas deste processo: *knowledge distillation*, *prunning*, *quantization*, *encoding* e *compilation*.

3.1. APRIMORAMENTO DA REDE NEURAL *(Knowledge Distillation)*

Após o treinamento de uma rede neural, seus parâmetros são calculados, e para o aprimoramento do modelo da rede neural, cada termo é analisado para verificar a contribuição dele no resultado da classificação de uma base de teste.

Caso o termo não contribua de forma significativa no resultado final, ele é marcado como um termo possível de ser removido do modelo. A remoção de termos do modelo de redes neurais que não influenciam muito no resultado final da classificação de uma base de teste reduz a quantidade termos calculados e otimiza o processamento do modelo.

3.2. SIMPLIFICAÇÃO DO MODELO *(Prunning)*

Após a fase de identificação dos termos que podem ser removidos do modelo de redes neurais, *knowledge distillation*, efetuamos a remoção destes termos, e geramos um novo modelo simplificado *(prunning)*.

O modelo simplificado, possui uma quantidade menor de termos que precisam ser calculados, otimizando o processamento do modelo de redes neurais.

3.3. NORMALIZAÇÃO DOS PARÂMETROS (Quantization)

O processo de normalização dos parâmetros, faz com que os valores dos parâmetros sejam convertidos em valores inteiros de 8, 16 ou 32 bits, com o objetivo de otimizar o cálculo usando números inteiros no lugar do cálculo com valores de ponto flutuante.

Por exemplo, em uma Placa Arduino UNO R3, que usa o processador ATmega328P, uma operação de multiplicação usando registradores com números inteiros de 8 bits, e terminando em um valor inteiro de 16 bits *(Unsignet Integer Multiply - MUL)*, consome 1 ciclo de processamento.

Enquanto uma operação de multiplicação usando registradores com números em ponto flutuante de 8 bits, e terminando em um valor em ponto flutuante de 16 bits *(Fractional Unsignet Integer Multiply - FMUL)*, consome 2 ciclos de processamento.

Como os registradores do microprocessador ATmega328P são de 8 bits, as operações de multiplicação com valores em ponto flutuante de 16 e 32 bits, consomem muitos ciclos de processamento em relação as mesmas operações com números inteiros.

Desta forma, o processo de normalização do modelo de redes neurais, convertendo os parâmetros de ponto flutuante para números inteiros, é fundamental para reduzir o tempo de processamento na classificação dos dados.

3.4. COMPACTAÇÃO DO MODELO *(Encoding)*

O processo de compactação do modelo de redes neurais *(encoding)*, procura identificar a ocorrência de sequencias de bits iguais no vetor de parâmetros do modelo.

A substituição do padrão de bits por seu código interno, obtido da tabela de padrões de bits previamente identificados no modelo, reduz o tamanho dos dados armazenados.

3.5. CONVERSÃO DO MODELO *(Compilation)*

O processo de conversão do modelo de redes neurais *(compiling)*, converte o modelo de redes neurais em uma linguagem de alto *nível como C/C++, Java ou Python*, com o objetivo de permitir a inserção do modelo no código da aplicação que irá executar no microcontrolador.

4. TEMA DE ESTUDO E PESQUISA NO DOUTORADO

O tema de Pesquisa para Tese D.Sc. é a avaliação das unidades de armazenamento de grandes volumes de dados e com capacidade de processamento *in-situ*, *computational storage devices (CSDs).*

A pesquisa, tem como objetivo resolver o problema de captura, classificação, pesquisa e processamento de grandes volume de imagens geradas por câmeras de vigilância, sistemas de monitoramento por satélites e drones.

4.1. *COMPUTATIONAL STORAGE DEVICES (CSDs)*

O conceito de *computational storage devices (CSDs)* vem sendo amplamente estudado com a aplicação dos recursos de processamento diretamente na unidade de armazenamento, *in-situ processing*, permitindo que diversos serviços suportados anteriormente pelo sistema operacional ou pela camada da aplicação estejam disponíveis na unidade de armazenamento, reduzindo o trabalho da CPU e proporcionando maior velocidade na transferência dos dados entre as unidades de armazenamento e memória.

A Figura 15, apresenta a unidade do tipo *computational storage devices (CSDs)*, Newport CSD de 16 TB, fabricada pela empresa NGD SYSTEMS. Na Figura 15, podemos ver a imagem de um servidor hospedeiro com 24 unidades CSDs, totalizando 1536 TB de dados.

Para otimizar a captura, classificação, pesquisa e processamento de grandes volume de imagens, foi desenvolvido uma ferramenta para realização dos experimentos, denominada Horus Image Server v2.0.

Figura 15: *Computational storage devices (CSDs)* - Newport CSD.

Figura 16: Servidor hospedeiro com 24 unidades de armazenamento CSDs.

4.2. *CLOUD COMPUTING*

O conceito de computação em nuvem, *cloud computing*, permite que os recursos de transferência de dados, armazenamento e processamento, seja compartilhado por diversar empresas, em um serviço de nuvem pública, como Amazon AWS, Microsoft Azure, Oracle Cloud, Google Cloud, e etc, ou por diversos projetos de uma grande empresa, que compartilham uma nuvem privada, como ocorre na Petrobrás, e outras grandes empresas.

4.3. *EDGE COMPUTING*

O surgimento de tecnicas de otimização do modelo e aprendizado de máquina, *tiny machine learning*, favorecem a implementação de soluções de inteligência computacional na borda, *edge computing, e* permite uma pré-classificação das imagens pelos equipamentos remotos.

Estudos mostram que o trafego total de dados entre os equipamentos remotos e o *data center*, pode ser reduzido em até 20 vezes com o uso de computação na borda.

4.4. *INTERNET OF THINGS (IoT)*

Atualmente, os dispositivos portateis do tipo *smartphones* geram um grande volume de informações que são coletadas automaticamente pelos sistemas operacionais Android e iOS.

Esta quantidade gigantesca de informações é enviada automaticamente a diversos servidores, para armazenar vídeos, fotos, audios, mensagens, informações de navegação, informações de usabilidade, erros do sistema, dados de posicionamento geográfico, dados da rede celular, dados das redes WiFi e Bluetouth que estes equipamentos identificam a todo instante.

Este gigantesco conjunto de informações são armazenados, jamais apagados, e analisados constantemente para fornecer um perfil dos usuários e prover um serviço melhor.

Deste modo, a Infraestrutura necessária neste processo inclui unidados de armazenamento capazes de processar os dados *in-situ*, e fornecer os resultados das consultas de forma simplificada, e no formato solicitado pelo usuário. Afim de reduzir o volume de dados trafegados entre cada unidade de armazenamento e o servidor hospedeiro.

Outros equipamentos que também geram grandes volumes de dados, são os equipamentos de vigilância e monitoramento com câmeras. Estes equipamentos registram imagens que são armazenadas constantemente, e na ocorrência de um evento, precisam ser consultadas de forma fácil e rápida para auxiliar as equipes de segurança na resolução de um problema.

Além dos *smartphones* e equipamentos de vigilância e monitoramento com câmeras, podemos destacar os sistemas de supervisão e controle, comuns nas empresas de utilities, como as empresas de Telecomunicações, Energia, Saneamento, e Óleo e Gás.

Estas empresas coletam constantemente informações dos equipamentos instalados em suas redes, para poder prover um serviço mais eficiênte. Estes dados são armazenados e muitas ações automatizadas são realizadas por sistemas especialistas, e em situações criticas de forma manual pelos operadores.

As informações coletadas, são armazenadas e analisadas constantemente, com o objetivo de melhorar a ação dos sistemas especialistas e auxiliar as equipes na resolução de problemas de emergência que precisam ser resolvidor de forma manual pelos operadores.

5. HORUS IMAGE SERVER v2.0

Para elaboração dos experimentos da Pesquisa para Tese de D.Sc., foi desenvolvido um servidor de banco de dados distribuído, que coordena o processo de armazenamento, classificação, pesquisa e processamento das imagens armazenadas nas unidades CSDs.

Soluções de geoprocessamento acessam grandes volumes de dados compostos por múltiplos arquivos de imagens de grandes dimensões, com mais de 400 MB, que precisam ser frequentemente processadas para conversão de formatos, alteração do sistema de coordenadas e projeção, criação de cache com diferentes níveis de precisão, comparação e identificação de mudanças, reconhecimento de padrões e etc.

Deste modo, distribuíndo o processamento entre as unidades de armazenamento, obtemos maior agilidade nas execução destas tarefas. Além de efetuar o treinamento de modelos de Redes Neurais Profundas, *Deep Neural Networks (DNN)*, e a classificação das imagens, usando um processo em execução nas unidades CSDs.

Os resultados iniciais apresentaram um ganho em *speedup* de até 2,41, na execução do sistema de banco de dados distribuído, Horus Image Server v2.0, em um ambiente simulado com 4 unidades de armazenamento do tipo CSDs, com capacidade de processamento embutido, *in-situ processing*, quando comparado com o ambiente de referência, que utiliza 1 unidade de armazenamento do tipo SSD, tradicional e sem capacidade de processamento embutido.

5.1. IMPLEMENTAÇÃO PROPOSTA DE REDES NEURAIS OTIMIZADAS *(TinyML)*

A implementação de redes neurais otimizadas usando t*iny machine learning*, permite usar modelos de redes neurais em microcontroladores com pouca capacidade de processamento e armazenamento.

Neste cenário, o modelo de redes neurais é criado em um servidor e instalado no *data center*, removendo a necessidade de processamento e armazemamento da rede neural.

5.2. TREINAMENTO E DISTRIBUIÇÃO DO MODELO

O processo de treinamento do modelo de aprendizado de máquina, é realizado pelos servidores localizados no *data center*, e posteriormente instalados no microcontrolador.

Esta sequencia de processamento é necessária devido as limitações de processamento e memória dos microprocessadores que utilizam a Placa Arduino UNO R3 e a Placa WiFi LoRa 868 MHz ESP32 com OLED.

5.3. BENEFICIOS ESPERADOS

Entre os benefícios da implementação da biblioteca *tiny macrine learning (TinyML)*, podemos destacar: (a) a redução do volume de imagens enviadas para o *data center,* e (b) a otimização no tempo de processamento de cada *frame* de vídeo, como as principais vantagens obtidas pela aplicação dos métodos de *tiny machine learning (TinyML)*.

6. CONCLUSÕES

A Pesquisa para Tese D.Sc., tem como objetivo verificar o desempenho das unidades de armazenamento com capacidade de processamento, *computational storage devices (CSDs)*, que possuem suporte para processamento embutido, *in-situ processing*, e que podem executar tarefas, como a classificação e o processamento de imagens diretamente nas unidades de armazenamento.

As unidades de armazenamento do tipo CSDs, oferecem uma solução bastante eficiente de armazenamento, classificação, indexação e processamento dos dados obtidos de sistemas de vigilância, sistemas de monitoramento por satélites, e de sistemas de monitoramento por drones.

Para auxiliar na elaboração da Pesquisa para Tese D.Sc., foi desenvolvido um sistema de banco de dados distribuído, denominado Horus Image Server v2.0, que utiliza a capacidade de processamento das unidades CSDs.

Em uma próxima versão do servidor Horus Image Server v2.0, o servidor poderá receber uma extensão que permite o processamento de modelos *tyne machine learning (TinyML)*, com objetivo de otimizar a classificação dos dados coletados pelas câmeras de vigilância, redes de satélites e drones.

8. BIBLIOGRAFIA

[1] L. M. F. A. VIANA, Memorização Dinâmica de Traces com Reuso de Valores de Instruções de Acesso à Memória [Rio de Janeiro] 2002, XIT, 118 p. 29,7 cm (COPPE/UFRJ, MSc., Engenharia de Sistemas e Computação, 2002), Tese - Universidade Federal do Rio de Janeiro, COPPE, 1. Reuso Dinâmico de Traces, 2. Arquitetura de Processador, I. C0PPEíWR.J 11. Título (série).

[2] Anonymous; "I/O and Energy Efficient Large-scale Image Similarity Search Using Computational Storage Devices", USENIX FAST.

[3] D. Jaeyong, C. F. Victor, B. Hossein, T. Mahdi, R. Siavash, H. Ali, Heydarigorji, S. Diego, F. G. Bruno, S. Leandro, K. S. Min, M. V. L. Priscila, M. G. F. França, A. Vladimir; "Cost-effective, Energy-efficient, and Scalable Storage Computing for Large-scale AI Applications"; ACM Transactions on Storage, Vol. 16, No. 4, Article 21. Publication date: October 2020.

[4] C. Wei, L. Yang, C. Zhushi, Z. Ning, L. Wei, W. Wenjie, O. Linqiang,W. Peng, W. Yijing, K. Ray, L. Zhenjun, Z. Feng, Z. Tong; "POLARDB Meets Computational Storage: Efficiently Support Analytic Workloads in Cloud-Native Relational Database"; 18th USENIX Conference on File and Storage Technologies; pages 29-41; 2020.

[5] T. Surat, M. Bradley, H. T. Kung; "Distributed Deep Neural Networks over the Cloud, the Edge and End Devices"; Naval Postgraduate School Agreements No. N00244-15-0050 and No. N00244-16-1-0018. 2015.

[6] Z. Li, W. Hao, T. Radu, and H. C. David Du; "Distributing Deep Neural Networks with Containerized Partitions at the Edge"; 2019.

[7] C. Sandeep, C. Eyal, P. Evgenya, C. Thianshu, K. Sachin; "Neural Networks Meet Physical Networks: Distributed Inference Between Edge Devices and the Cloud"; 2018.

[8] H. Ali, T. Mahdi, R. Siavash, B. Hossein; "STANNIS: Low-Power Acceleration of Deep Neural Network Training Using Computational Storage Devices"; 2020.

[9] L. Chun-Yi, B. K. Jagadish, J. Myoungsoo, T. K. Mahmut; "PEN: Design and Evaluation of Partial-Erase for 3D NAND-BasedHigh Density SSDs"; ISCA 2020.

[10] T. Arash, G. L. Juan, S. Mohammad, G. Saugata, M. Onur; "MQSim: A Framework for Enabling Realistic Studies of Modern Multi-Queue SSD Devices"; ISCA 2020.

[11] A. Bulent, B. Bart, R. John, K. Mathias, M. Ashutosh, B. A. Craig, S. Bedri, B. Alper, J. Christian, J. S. Willian, M. Haren, and W. Charlie; "Data Compression Accelerator on IBM POWER9 and z15 Processors"; ISCA 2020.

[12] B. Eunjin, K. Dongup, and K. Jangwoo; "A Multi-Neural Network Acceleration Architecture"; ISCA 2020.

[13] B. Summet, G. Jayesh, S. Zeev, R. Lihu, Y. Adi, S. Sreenivas; "Focused Value Prediction"; ISCA 2020.

[14] K. Jeremy, A. Willian, B. Willian, B. Nadya, B. Robert, B. Chansup, C. Gary, G. Kenneth, H. Matthew, K. Jonathan, M. Andrew, M. Peter, P. Andrew, R. Albert, R. Antonio, Y. Charles; "Dynamic Distributed Dimensional Data Model (D4M) Database and Computation System"; 2012.

[15] C. M. Rodrigo, O. T. Giovane, L. P. Maurício, L. P. Laércio, T. C. Amarildo, M. G. F. Felipe; "Value Reuse Potential in ARM Architectures"; IEEE 28th International Symposium on Computer Architecture and High Performance Computing; 2016.

[16] L. P. Maurício, R. C. Bruce, T. C. Amarildo, M. G. F. Felipe, O. A. N. Philippe, "A Speculative Trace Reuse Architecture with Reduced Hardware Requirements", 2006.

[17] D. G. Massimo, G. Maurizio; "WiSARD rp for Change Detection in Video Sequences"; ESANN 2017.

[18] Y. S. Abu-Mostafa, M. Magdon-Ismail, H. Lin Authors, "e-Chapter 7 – Neural Networks" in Learning From Data a Short Course, Country: Passadena, CA, USA, 2012, ch. 7.

[19] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011, https://scikit-learn.org/stable/user_guide.html.

[20] Keras: The Python Deep Learning library, F. Chollet et al, 2015, https://keras.io/.

[21] TensorFlow, J. Dean, G. Corrado, A. Ng., 2011, https://www.tensorflow.org/.

[22] Y. LeCun, "Learning Process in an Asymmetric Threshold Network", 1986.

[23] Y. LeCun, "A Theoretical Framework for Back-Propagation", 1988.

[24] D. Eigen, J. Rolfe, R. Fergus, Y. LeCun, “Understanding Deep Architectures using a Recursive Convolutional Network”, 2014.

[25] D. G. Massimo, G. Maurízio; “Change Detection with Weightless Neural Networks”; CVPR 2014.

[26] C. Chunlei, C. Li, Z. Lei, Z. Xiaoyun, G. Zhiyong; “RCDFNN: Robust Change Detection Based on Convolutional Fusion Neural Network”; 2018.

[27] Y. S. Abu-Mostafa, M. Magdon-Ismail, H. Lin Authors, “e-Chapter 8 – Support Vector Machines” in Learning From Data a Short Course, Country: Passadena, CA, USA, 2012, ch. 8.

[28] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011, https://scikit-learn.org/stable/user_guide.html.

[29] C. G. B. Caio; “An Adaptation of the Data Flow Library Sucuri Static Scheduler for In-Situ Computing”; Dissertation presented to COPPE/UFRJ in March, 2018.

[30] M. Bjørling, J. González and P. Bonnet; “LightNVM: The Linux Open-Channel SSD Subsystem”; 15th USENIX Conference on File and Storage Technologies, 2015.

[31] J. Huang, S. Nath, A. Badam, S. Sengupta, B. Sharma, L. Caulfield and M. K. Qureshi; “FlashBlox: Achieving Both Performance Isolation and Uniform Lifetime for Virtualized SSDs”; 15th USENIX Conference on File and Storage Technologies, 2015.

[32] S. Yan, H. Li, M. Hao, M. H. Tong, S. Sundararaman, A. A. Chien and H. S. Gunawi; “Tiny-Tail Flash: Near-Perfect Elimination of Garbage Collection Tail Latencies in NAND SSDs”; 15th USENIX Conference on File and Storage Technologies, 2015.

[33] F. Douglis, A. Duggal, P. Shilane, T. Wong, S. Yan and F. Botelho; “The Logic of Physical Garbage Collection in Deduplicating Storage”; 15th USENIX Conference on File and Storage Technologies, 2015.

[34] W. He and D. H. C. Du; “SMaRT: An Approach to Shingled Magnetic Recording Translation”; 15th USENIX Conference on File and Storage Technologies, 2015.

[35] Y. Li, H. Wang, X. Zhang, N. Zheng, S. Dahandeh and T. Zhang; “Facilitating Magnetic Recording Technology Scaling for Data Center Hard Disk Drives through Filesystem-level Transparent Local Erasure Coding”; 15th USENIX Conference on File and Storage Technologies, 2015.

[36] R. G. Tinedo, J. Sampé, E. Zamora, M. S. Artigas and P. G. López; “Crystal: Software-Defined Storage for Multi-tenant Object Stores”; 15th USENIX Conference on File and Storage Technologies, 2015.

[37] S. K. Lee, K. H. Lim, H. Song, B. Nam, S. H. Noh; “WORT: Write Optimal Radix Tree for Persistent Memory Storage Systems”; 15th USENIX Conference on File and Storage Technologies, 2015.