Emergency Sound Detection – WiSARD

Aluno: Luiz Marcio Faria de Aquino Viana, M.Sc.
E-mail: luiz.marcio.viana@gmail.com
Telefone: +55-21-99983-7207
DRE: 120048833

Resumo:

O objetivo deste trabalho é implementar uma aplicacao de detecção de sons de emergencia, usando *smartphones* e servidores na nuvem. Além de estudar os conceitos de *Federated Learning (FL)* e TinyML, que otimizam as implementacoes de algorítimos de aprendizado de máquina em microcontroladores.

1. Introdução

Este estudo apresenta dois novos conceitos de aprendizado de máquina: (i) TinyML e (ii) *Federated Learning*. O conceito de TinyML é fundamental para a implementação de algorítimos de aprendizado de máquina em microcontroladores com baixa capacidade de processamento. O conceito de *Federated Learning* permite distribuir a criação do modelo global de aprendizado de máquina por vários microcontroladores, onde cada equipamento é responsável por contribuir com a criação do modelo global, análisando somente os dados locais.

Neste trabalho foi implementada uma aplicacao de detecção de sons de emergencia, usando *smartphones* e computação em nuvem. A aplicação desenvolvida foi executada em ambiente simulado com múltiplas threads de processamento representando o servidor e os *smartphones*.

Como o objetivo era desenvolver um sistema para *smartphones com* boa capacidade de processamento, mas sem endereço de Internet válidos para comunicação direta entre eles, foi escolhida a implementação de um modelo de aprendizado do tipo *Federated Learning* (FL) com rede neural sem pesos *WiSARD*. Este modelo de aprendizado obteve excelente desempenho na detecção de tiros com armas de fogo.

2. Conceitos Básicos

A Identificação de sons de emergencia é um processo fundamental na análise automatizada de audio. Desta forma, para simplificar a pesquisa sobre identificação de sons de tiros de armas de fogo, usamos o *dataset UrbanSound8K*, que possui mais de 8.732 sons urbanos identificados.

2.1. Definição do Problema

O problema é capturar o som ambiente usando uma aplicação instalada em vários smartphones. Se o som ambiente detectado for classificado como um som de tiro de arma de fogo, uma mensagem com o som e a posição GPS do equipamento é enviada a um servidor na internet que coleta os dados, efetua uma nova analisar mais detalhada, e dependendo da concentração de informações obtidas em uma mesma região da cidade, notifica uma central de emergência para verificar a ocorrência.

Os experimentos deste estudo foram realizados de forma local, em cada nó de processamento simulado. Cada nó de processamento possui uma localização geográfica, e analisa de forma aleatória o *dataset*.

No início da aplicação o equipamento efetua uma autenticação e recebe uma chave de sessão, em seguida solicita o modelo de aprendizado de máquina mais recente. Em seguida, analisa os sons presentes no *dataset* de forma aleatória nas escolhas e no tempo.

DATASET: Kaggle - UrbanSound8K
https://www.kaggle.com/datasets/chrisfilo/urbansound8k

O dataset contém 10 categorias de sons: *air conditioner*, *car horn*, *children playing*, *dog bark*, *drilling*, *engine idling*, *gun shot*, *jackhammer*, *siren*, e *street music*.

Na fase de treinamento inicial realizado no servidor, usamos 30 sons da categoria *gun shot*.

Em seguida, na fase de teste usamos 10 sons da mesma categoria *gun shot*, e estabelecemos os *threshold,* os limites minimo e maximos.

Neste experimento obtivemos 90% de acerto na fase de Teste, mas o processamento não foi executado ate o final, e parou em 1.131 sons analisados, devido a erro de arquivos não encontrados motivados pela base incompleta.

2.2. Federated Learning (FL)

O aprendizado federado, Federated Learning (FL), é um processo de aprendizado de máquina que explora a capacidade de comunicação de dados dos dispositivos remotos e distribui o processamento de dados para construção do modelo de aprendizado.

No aprendizado federado, uma rede de sensores coleta os dados e encaminha para um microcontrolador que concentra os dados recebidos dos sensores.

Nestes dispositivos remotos, um processo atualiza periodicamente um modelo parcial de aprendizado de máquina, com base nos dados coletados por cada microcontrolador.

Após a atualização do modelo parcial, os parâmetros são compartilhados entre todos os dispositivos e são utilizados para aprimorar a precisão do modelo global de aprendizado de máquina.

Deste modo, o processo de criação do modelo global é distribuído entre os dispositivos remotos, reduzindo a carga de trabalho realizada por cada dispositivo.

O aprendizado federado é excelente na teoria, porque na prática não tem como implementar, porque os *smartphones* possuem IPv4 e IPv6 inválidos. Desta forma para *smartphones* conectados a internet a solução é sincronizar o modelo com um servidor na nuvem. A Figura 1, apresenta a arquitetura do modelo de aprendizado federado usando um servidor centralizado.

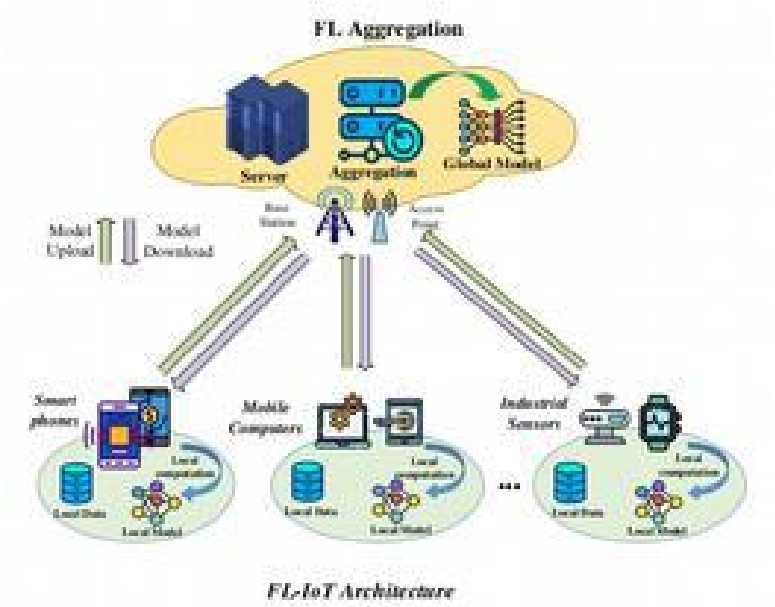

Figura 1: arquitetura do modelo de aprendizado federado.

As bibliotecas TinyML possibilitam a aplicação do aprendizado federado em qualquer dispositivo remoto, como computadores, tablets, smartphones e microcontroladores de baixo custo. A Figura 2, apresenta dois microcontroladores de baixo custo, o Arduino e o ESP32, Figura 2.

Estes equipamentos utilizam redes WiFi e podem ser usados em áreas industriais, tirando proveito do conceito de aprendizado federado, de acordo com a definição primordial.

Figura 2: microcontroladores de baixo custo Arduino e ESP32.

2.3. TinyML

As redes neurais convolucionais usadas na classificação de elementos nas imagens, ocupam grande espaço em memória, e para serem usadas em microcontroladores, é necessário efetuar a otimização do modelo de redes neurais. A Tabela 3, apresenta as características dos microcontroladores ATmega328P e PIC16F873A.

ATMega328 . Low Power 8-Bit Microcontroller . Advanced RISC Architecture – 32 x 8 General Purpose Working Registers – Up to 20 MIPS Throughput at 20 MHz – On-chip 2-cycle Multiplier . High Endurance Non-volatile Memory Segments – 4/8/16/32K Bytes of In-System Self-Programmable Flash program memory – 256/512/512/1K Bytes EEPROM – 512/1K/1K/2K Bytes Internal SRAM . Price: US$ 1.11/unit to US$ 1.21/unit . Reference: Alibaba.com in 4th September, 2022.	Microchip PIC16F873A . Low Power 8-Bit Microcontroller . Instruction Set - 35 Instructions . Operating Frequency - DC – 20 MHz . Flash Program Memory (14-bit words) – 4 KB . Data Memory - 192 bytes . EEPROM Data Memory - 128 bytes . Price: US$ 2.00/unit to US$ 5.00/unit . Reference: Alibaba.com in 4th September, 2022.

Tabela 3: Lista das características dos microcontroladores ATmega328P e PIC16F873A.

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 possuem 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. Solução usando *WiSARD*

A identificação de padrões em sons de tiros de armas de fogo diretamente nos equipamentos de monitoramento, reduz de forma significativa o volume de dados trafegados pela rede.

Neste estudo, foi implementado um mecanismo de armazenamento otimizado dos modelos de aprendizado de redes neurais aplicado na classificação de sons, explorando a capacidade de processamento d*smartphones*. Este trabalho avalia o uso das redes neurais sem peso *WiSARD* no processo de classificação de sons.

A principal características da rede neural *WiSARD* é a rapidez de processamento da fase de aprendizado e classificação do modelo. Porém, estes modelo de redes neurais consomem muita memória quando aplicado no processamento de imagens rasters.

As redes neurais sem peso realizam o mapeamento dos bits do arquivo de audio em um vetor de endereçamento de memória, que armazena o aprendizado obtido na fase de treinamento da rede neural.

A rede neural sem pesos *WiSARD* procura identificar a frequência de ocorrência dos padrões do som. Para realizar o aprendizado de maquina e obter o melhor desempenho na identificação do som, implementamos os seguintes passos:

i) O dataset contém 10 categorias de sons: *air conditioner*, *car horn*, *children playing*, *dog bark*, drilling, *engine idling*, *gun shot*, *jackhammer*, *siren*, *street music*

ii) Na fase de Treinamento usamos 30 sons da categoria *gun shot*.

iii) Na fase de Teste usamos 10 sons da categoria *gun shot* e estabelecemos os *threshold,* os limites minimos e maximos.

iv) Nivelamos os arquivos de som para valores inteiros entre 0 a 256.

v) Em seguida, separamos em 32 faixas, e medimos a frequencia de acesso de cada faixa pelo arquivo de som.

vi) Desta forma,btivemos 90% de acerto na fase de Teste.

vii) O processamento não foi executado ate o final, por limitação dos arquivos presentes no *dataset*.

WiSARD

Na classificação da WiSARD, os endereços para os nós RAM são extraídos das diversas n-uplas de pixels que compõem a partição da imagem, da mesma maneira como é feito no caso do treinamento, e um contador em cada neurônio é zerado no início da operação de classificação da imagem. Para cada n-upla onde o endereço gerado aponta para um valor 1, este contador é incrementado.

O valor final desse contador constitui a resposta do neurônio a imagem apresentada, e vence o neurônio que apresentar a resposta mais alta, ou seja, o neurônio cujo contador tiver a soma mais alta.

A Figura 3, apresenta a arquitetura de classificação da rede neural sem pesos *WiSARD*, onde o contador de valor mais alto determina o neurônio com a resposta mais provável.

Figura 3: apresenta a arquitetura de classificação da rede neural sem pesos WiSARD

3.1. Implementação Própria da *WiSARD*

AudioPattern, é a implementação da Rede Neural sem Pesos *WiSARD.* A implementação da rede neural sem peso *WiSARD* foi realizada usando um mapa de memória. Os dados de entrada são preprocessadas para pegar os 4096 primeiros valores, e agrupar em 32 faixas.

Este experimento analisou o desempenho do modelo de aprendizado das redes neurais sem peso *WiSARD*, e obteve excelente desempenho.

Na fase de Treinamento da rede neural sem pesos WiSARD usamos 30 sons de tiros com armas de fogo. Em seguida, aplicamos o modelo na classificação de 10 sons separados para uso na fase de Teste.

4. Análise dos Experimentos

Os experimentos deste estudo foram realizados de forma simulada em 1 computador local, representando um único nó de processamento. Este experimento pode ser facilmente estendido para ambientes com múltiplos nós de processamento (*smartphones*).

HARDWARE: Notebook com Processador Intel i5-4210U; 1,7 GHz; 2 Núcleos; 4 Threads; 12 GB RAM;

SOFTWARES: Linux CentOS 7; Kernel 3.10; Java 1.8.0.292-b10; Eclipse Java; PostgreSQL;

DATASET: Kaggle - UrbanSound8K
https://www.kaggle.com/datasets/chrisfilo/urbansound8k

O dataset contém 10 categorias de sons: *air conditioner*, *car horn*, *children playing*, *dog bark*, *drilling*, *engine idling*, *gun shot*, *jackhammer*, *siren*, e *street music*.

Neste experimento usamos o *dataset UrbanSound8K*, onde na fase de Treinamento usamos 30 sons da categoria *gun shot*. Na fase de Teste usamos 10 sons da categoria *gun shot* e estabelecemos os *thresholds*.

Obtivemos 90% de acerto na fase de Teste. O processamento NAO foi executado ate o final porque a base de dados está incompleta e possui somente 1.131 arquivos de audios.

As Tabelas 2 e 3, apresentam os resultados respectivamente do processamento do Treinamento e dos Testes.

-- TRAIN --	
0 - gun_shot - 7060-6-0-0.wav	15 - gun_shot - 7067-6-0-0.wav
1 - gun_shot - 7060-6-1-0.wav	16 - gun_shot - 7068-6-0-0.wav
2 - gun_shot - 7060-6-2-0.wav	17 - gun_shot - 23161-6-0-0.wav
3 - gun_shot - 7061-6-0-0.wav	18 - gun_shot - 23161-6-1-0.wav
4 - gun_shot - 7062-6-0-0.wav	19 - gun_shot - 24631-6-0-0.wav
5 - gun_shot - 7063-6-0-0.wav	20 - gun_shot - 24632-6-0-0.wav
6 - gun_shot - 7064-6-0-0.wav	21 - gun_shot - 24632-6-1-0.wav
7 - gun_shot - 7064-6-1-0.wav	22 - gun_shot - 24652-6-0-0.wav
8 - gun_shot - 7064-6-2-0.wav	23 - gun_shot - 25037-6-0-0.wav
9 - gun_shot - 7064-6-3-0.wav	24 - gun_shot - 25037-6-1-0.wav
10 - gun_shot - 7064-6-4-0.wav	25 - gun_shot - 25038-6-0-0.wav
11 - gun_shot - 7064-6-5-0.wav	26 - gun_shot - 25038-6-1-0.wav
12 - gun_shot - 7065-6-0-0.wav	27 - gun_shot - 25039-6-0-0.wav
13 - gun_shot - 7066-6-0-0.wav	28 - gun_shot - 25039-6-1-0.wav
14 - gun_shot - 7066-6-1-0.wav	29 - gun_shot - 34708-6-0-0.wav

Tabelas 2: Resultados do processamento do Treinamento.

```
-- TEST --
0 - gun_shot - 34708-6-1-0.wav 8 - gun_shot - 37236-6-0-0.wav
Score: 11.727783203125         Score: 1.496337890625
1 - gun_shot - 34708-6-2-0.wav 9 - gun_shot - 46654-6-0-0.wav
Score: 11.727783203125         Score: 9.52587890625
2 - gun_shot - 34708-6-3-0.wav
Score: 11.727783203125
3 - gun_shot - 34708-6-4-0.wav
Score: 11.727783203125
4 - gun_shot - 34708-6-5-0.wav
Score: 11.727783203125
5 - gun_shot - 35799-6-0-0.wav
Score: 10.4833984375
6 - gun_shot - 35800-6-0-0.wav
Score: 10.408447265625
7 - gun_shot - 36403-6-0-0.wav
Score: 9.4599609375
```

Tabelas 3: Resultados do processamento dos Testes.

As Listagens 1 e 2, apresentam exemplos da classificação do *dataset*..

```
-- CLASSIFY ALL DATASET --
1004 - dog_bark - 344-3-5-0.wav
Score: 11.727783203125
[GUN_SHOT]
*** Next audio in 3 seconds...
<?xml version="1.0" encoding="ISO-8859-1"?><Request> <RequestId><![CDATA[20220926_160724_244070]]></RequestId> <Command><!
[CDATA[REQMSG_SENDDATA]]></Command> <Params>  <SessionUUID><![CDATA[f84c073363194ef095fb427aff427a8f]]></SessionUUID>
<FileName><![CDATA[344-3-5-0.wav]]></FileName> <FileSize><![CDATA[-1]]></FileSize> <FileDataBase64><![CDATA[-1]]></FileDataBase64>
</Params>  <Position>  <Latitude><![CDATA[22.756206758268867]]></Latitude>  <Longitude><![CDATA[44.64076474481096 6]]></Longitude>
</Position></Request>
BASEMSG - MessageId: 20220926_160724_244070,MessageTypeId: -1,MessageType: REQMSG_SENDDATA,statusCode: -1,statusMessage: ?
SendData - Request FileName: 344-3-5-0.wav;Latitude: 22.756207;Longitude: 44.640765
<?xml version="1.0" encoding="ISO-8859-1"?><Response><ResponseId><![CDATA[20220926_160724_244089]]></ResponseId>  <Response><!
[CDATA[RESPMSG_SENDDATA]]></Response>  <Status>  <StatusCode><![CDATA[-1]]></StatusCode>  <StatusMessage><!
[CDATA[?]]></StatusMessage>  </Status>  <ResponseData>  <SessionUUID><![CDATA[f84c073363194ef095fb427aff427a8f]]></SessionUUID>
<FileUUID><![CDATA[03912ef12436408599119 1b55c20e8a8]]></FileUUID>  </ResponseData></Response>
BASEMSG - MessageId: 20220926_160724_244089,MessageTypeId: -1,MessageType: RESPMSG_SENDDATA,statusCode: -1,statusMessage: ?
SendData - Response FileUUID: 03912ef12436408599119 1b55c20e8a8
-- CLASSIFY ALL DATASET --
1005 - drilling - 518-4-0-0.wav
Score: 4.0380859375
[?]
```

FALHA!
IDENTIFICACAO DE LATIDO
COMO TIRO

SUCESSO!
IDENTIFICACAO DE LATIDO
COMO NAO TIRO

Listagem1: Resultados da classificação do *dataset*, com falhas.

Listagem2: Resultados da classificação do *dataset*, com Sucessos.

5. Conclusão

Este estudo apresentou o uso da rede neural sem pesos *WiSARD* na deteccao de sons de emergencia.

A rede neural *WiSARD* foi armazenada em um vetor de 4096 valores Inteiros de 32 bits, porem, nos experimentos podemos verificar que pode ser usada vetores menores e com menor precisao, como vetores de 1024 valores de 8 bits (byte) – seguindo um padrão proposto pelo TinyML.

Nesta implementacao pudemos analisar os detalhes por tras de uma solucao para monitoramento de emergencia, como por exemplo a identificação precisa dos sons ambientes e os valores de coordenadas GPS, que podem ser usados para validar uma ocorrencia em funcao da concentracao de dados em uma mesma regiao.

Referências

[1] BURGER, D., AUSTIN T. M., "The Simplescalar Tool Set, Version 2.0", Technical Report 1342, Computer Science Departament of University of Wisconsin.

[2] CALDER, B., GRUNWALD, D., “Next cache line and set prediction”, In Proceedings of the 22nd Annual International Symposium on Computer Architecture, pp 287-296, Jun-1995.

[3] COSTA, A. T., FRANÇA, F. M. G., CHAVES, E. M. F., "The Reuse Potential of Trace Memoization", Technical Report ES-498/99, COPPE/UFRJ, Rio de Janeiro, May-1999.

[4] COSTA, A. T., FRANÇA, F. M. G., “Process of Formation Memorization and Reuse, in Execution Time, of Sequences of Dynamic Instructions in Computers”, International Patent, number WO 01/04746 A1, Patent Cooperation Treaty (PCT), Jul-1999.

[5] COSTA, A. T., FRANÇA, F. M. G., CHAVES, E. M. F., "Evaluating DTM in a Superscalar Processor Architecture", Technical Report ES-538/00, COPPE/UFRJ, Rio de Janeiro, Aug-2000.

[6] COSTA, A. T., FRANÇA, F. M. G., CHAVES, E. M. F., "The Dynamic

Trace Memoization Reuse Technique", In Proceedings of the International Conference on Parallel Architecture and Compiler Techniches (PACT2000), Oct-2000.

[7] COSTA, A. T., FRANÇA, F. M. G., CHAVES, E. M. F., "Exploiting Reuse with Dynamic Trace Memoization: Evaluating Architectural Issues", In Proceedings of the 12th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD), pp. 163-172, Oct-2000.

[8] COSTA, A. T., “Explorando Dinamicamente o Reuso de Traces em Nível de Arquitetura de Processador”, D.Sc. dissertation, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, 2001.

[9] FRANKLIN, M., SOHI, G. S., “ARB: A Hardware Mechanism for Dynamic Reordering of Memory References”, In Proceedings of the IEEE Transaction Computers v.45, n. 5, pp. 552-571, May-1996.

[10] GABBAY, F., MENDELSON, A., "Speculative Execution Based on Value Prediction", Technical Report EE-TR 1080, Technion – Israel Institute of Technology, Nov-1996.

[11] GABBAY, F., MENDELSON, A., “Can Program Profiling Support Value Prediction?”, In Proceedings of the 30 th Annual Symposium on Microarchitecture, pp. 270-280, Dec-1997.

[12] GABBAY, F., MENDELSON, A., “The Effect of Instruction Fetch Bandwidth on Value Prediction”, In Proceedings of the 25th Annual International Symposium on Computer Architecture, pp. 272-281, 1998.

[13] GONZALEZ, A., TUBELLA, J., MOLINA, C., “Trace-Level Reuse”, In Proceedings of the International Conference on Parallel Processing, pp 30-37, Japan, Sep-1999.

[14] HEIL, T. H., SMITH, Z., SMITH J. E., “Improving Branch Predictors by Correlating on Data Values”, In Proceedings of the 32 th International Symposium on Microarchitecture, pp. 28-37, Nov-1999.

[15] HENNESSY, J., PATTERSON, D., Computer Architecture: A Quantitative Approach, Morgan-Kaufmman, pp 29-38 e pp 76-83, 1997.

[16] HUANG, J., LILJA, D.J., "Exploiting Basic Block Value Locality with

Block Reuse", In Proceedings of the 5th High Performance Computer Architecture (HPCA), pp 106-114, Jan-1999.

[17] HUANG, J., LILJA, D.J., "Exploiring Sub-Block Value Reuse for Superscalar Processors", In Proceedings of the 2000 International Conference on Parallel Architecture and Compiler Techniches (PACT2000), Oct-2000.

[18] JACOBSEN, E., ROTENBERG, E., SMITH, J. E., “Assign Confidence to Conditional Branch Predictions”, In Proceedings of the 29th Annual IEEE/ACM International Symposium on Microarchitecture, pp. 142-152, Dec-1996.

[19] LAM, M. S., WILSON, R. P., “Limits of Control Flow on Parallelism”, In Proceedings of the 19th International Symposium on Computer Architecture ACM/IEEE, pp. 46-57, Jul-1992.

[20] LIPASTI, M. H., WILKERSON, C. B., SHEN, J. P., "Value Locality and Load Value Prediction", In Proceedings of the 7th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), pp 138-147, Oct-1996.

[21] LIPASTI, M. H., SHEN, J. P., "Exceeding the Dataflow Limit Via Value Prediction", In Proceedings of the 29th Annual ACM/IEEE International Symposium and Workshop on Microarchitecture, pp 226-237, Dec-1996.

[22] MACFARLING, S., “Combining Branch Predictors”, Technical Report TN-36, DEC Western Research Laboratory, Jun-1993.

[23] MARTIN, M. M., ROTH, A., FISCHER, C. N., “Exploiting Dead Value Information”, In Proceedings of the 30 th Annual Symposium on Microarchitecture, pp. 128-135, Dec-1997.

[24] MOSHOVOS, A., SOHI, G., “Streamlining Inter-operation Memory Comunication via Data Dependence Prediction”, In Proceedings of the 30th Annual ACM/IEEE International Symposium and Workshop on Microarchitecture, pp 235-245, Dec-1997.

[25] PAN, S.-T., SO, K., RAHMEH, J. T., “Improving the Accuracy of Dynamic Branch Prediction Using Branch Correlation”, In Proceedings of

the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS V), pp. 76-83, Oct-1992.

[26] REBELLO, V. E. F., “Neurocom Project”, Technical Report ProTem-II CC, CNPQ, Brasil, May-1997.

[27] ROTENBERG, E., BENNETT, S., SMITH, J. E., “Trace Cache: a Low Latency Approach to High Bandwidth Instruction Fetching”, In Proceedings of the 29th International Symposium on Microarchitecture, pp. 24-34, Dec-1996.

[28] RYCHLIK, B., FAISTL J. W., J., KRUG, B. P., KURLAND, A. Y., SUNG, J. J.,VELEK, M. N., SHEN, J. P., “Efficient and Accurate Value Prediction Using Dynamic Classification”, Technical Report of Microarchitecture Research Team in Dept. of Electrical and Computer Engineering, Carnegie Mellon University, 1998.

[29] RYCHLIK, B., FAISTL, J., KRUG, B., SHEN, J. P., “Efficacy and Performance Impact of Value Prediction”, In Proceedings of the International Conference on Parallel Architectures and Compilation Techniques, Oct-1998.

[30] SPARC International, “The SPARC Architecture Manual, Version 7”, Sun Microsystems Inc. Oct-1987.

[31] SAZEIDES, Y., SMITH, J. E., “The Predictability of Data Values”, In Proceedings of the 30th International Symposium on Microarchitecture, pp. 248-258, Dec-1997.

[32] SMITH, J. E., SOHI, G. S., “The Microarchitecture of Superscalar Processors”, In Proceedings of the IEEE Transaction Computers v.83, n. 12, pp. 1609-1624, Dec-1995.

[33] SODANI, A., SOHI, G. S., "Dynamic Instruction Reuse", In Proceedings of the 24th International Symposium on Computer Architecture (ISCA), pp 194-205, Jun-1997.

[34] TOMASULO, R., “An Efficient Algotithm for Exploiting Multiple Arithmetic Units”, IBM Journal of Research and Development v. 11, n. 1,

pp. 25-33, Jan-1967.

[35] YEH, T.-Y., MARR, D. T., PATT, Y. N., “Increasing the Instruction Fetch Rate via Multiple Branch Prediction and a Branch Address Cache”, In Proceedings of the 7th ACM International Conference on Supercomputing, pp. 67-76, Jul-1993.

[36] YEH, T.-Y., PATT, Y. N., “Two-Level Adaptive Branch Prediction”, In Proceedings of the 24 th Annual ACM/IEEE International Symposium and Workshop on Microarchitecture, pp. 51-61, Nov-1991.

[37] YEH, T.-Y., PATT, Y. N., “A Comparison of Dynamic Branch Predictors that Use Two Levels of Branch History”, In Proceedings of the International Symposium on Computer Architecture, pp. 257-267, May-1993.

[38] TYSON, G. S., AUSTIN, T. M., “Improving the Accuracy and Performance of Memory Communication Through Renaming”, In Proceedings of the 30 th Annual Symposium on Microarchitecture, pp. 218-227, Dec-1997.

[39] YUNG, R., “Design Decisions Influencing the UltraSPARC’s Instruction Fetch Architecture”, In Proceedings of the 29th Annual IEEE/ACM International Symposium on Microarchitecture, pp. 178-190, Dec-1996.

[40] YOUNG, C., GLOY, N. C., SMITH, M. D., “A Comparative Analysis of Schemes for Correlated Branch Prediction”, In Proceedings of the 22nd Annual International Symposium on Computer Architecture, pp. 276-286, Dec-1995.

[41] 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).

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

[43] 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.

[44] 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.

[45] 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.

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

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

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

[49] 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.

[50] 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.

[51] 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.

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

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

[54] 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.

[55] 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.

[56] 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.

[57] D. G. Massimo, G. Maurizio; “WiSARD rp for Change Detection in Video Sequences”; ESANN 2017.

[58] 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.

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

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

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

[62] Y. LeCun, “Learning Process in an Asymmetric Threshold Network”, 1986.

[63] Y. LeCun, “A Theoretical Framework for Back-Propagation”, 1988.

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

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

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

[67] 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.

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

[69] 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.

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

[71] 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.

[72] 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.

[73] 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.

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

[75] 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.

[76] 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.

[77] 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.

[78] Lima, José de Jesus Botelho; Sistema Antibloqueio (ABS) para Freios Eletromecânicos utilizando Controle por Modos Deslizantes [Rio de Janeiro] 2005 xiv, 124p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia Elétrica, 2005); Tese – Universidade Federal do Rio de Janeiro, COPPE; 1. Controle por modos deslizantes; 2. Sistema Antibloqueio (ABS) para veículos de quatro rodas; 3. Controle de Freio Eletromecânico; 4. Observadores não lineares; 5. Ajuste do deslizamento desejado, busca da força de atrito ótima, busca extremal; I. COPPE/UFRJ II. Título (série).

[79] Trimble Navigation Limited; Mobile Positioning and Communications; CrossCheck GSM 1900 with IQ Event Engine; Part Number: 43458-00 – Revisão: C – Data: Fevereiro, 2001, https://www.trimble.com.

[80] Trimble Navigation Limited; Mobile Positioning and Communications; CrossCheck GSM/GPRS 850/1900 Installation Manual; Part Number: 47770-10-ENG – Revisão: A - Data: Janeiro, 2004, https://www.trimble.com.

[81] Rosa, Marcelo Carvalho; Sistema de Direção Elétrica Assistida para Veículos Elétricos e Híbridos Utilizando Motor de Indução [Curitiba] 2015 (UTFPR, Especialização em Sistemas Embarcados para Industria Automotiva, 2015); Trabalho de Conclusão de Curso; Universidade Tecnológica Federal do Paraná – UTFPR; 1. Direção elétrica assistida; 2. Veículos híbridos; 3. Eficiência na emissão de gases.

[82] Fornasari, Adriano José; Biondo, Diego; Roani, Luan Saldanha; Sistema de Controle de Temperatura de Ar Condicionado Automotivo [Pato Branco] 2013 (UTFPR, Tecnologia em Automação Industrial, 2013); Trabalho de Conclusão de Curso; Universidade Tecnológica Federal do Paraná – UTFPR.

[83] Israel de Andrade; “Como Funcionam os Carros Autonomos”; https://medium.com/brasil-ai/como-funcionam-os-carros-autônomos-parte-1-sensoriamento-e-visão-computacional.

[84] C. E. Milhor, L. C. Passarini; “Estado da Arte do Controle Eletrônico dos Motores de Combustão Interna”; II Congresso Nacional de Engenharia Mecânica; João Pessoa – PB; Agosto, 2002.

[85] E. Passos; “Tesla agora sabe ultrapassar sozinho e dirigir na completa escuridão”; Revista Quatro Rodas; https://quatrorodas.abril.com.br/noticias/tesla-agora-sabe-ultrapassar-sozinho-e-dirigir-na-completa-escuridao/.