Avaliação de desempenho da WiSARD utilisando o dataset Kuzushiji-MNIST

Luis Armando Quintanilla Villon[1] **Hendrick Villela Carrijo**[2]

[1]Programa de Engenharia de Sistemas e Computação (PESC) – COPPE/UFRJ
Programa de Engenharia Metalúrgica e de Materiais (PEMM) – COPPE/UFRJ
Redes neurais sem peso (CPS841)

lvillon@cos.ufrj.br hendrickvc@poli.ufrj.br

Resumo. *O classificador WiSARD é um tipo de rede neural muito utilizado para machine learning em processamento de imagens, apresentando como maior diferencial a não utilização de pesos e o armazenamento em memória RAM de imagens mentais aprendidas para classificação. Utilizando o dataset Kuzushiji-MNIST para treinamento do modelo, foi possível aplicar diversas técnicas de binarização de imagens e obter um desempenho com 85,01% de acurácia.*

1. Introdução

Com a enorme quantidade de dados obtidos através de sensores, gravações, prints digitais e digitalizações, a utilização de aprendizado de máquina para descoberta de padrões a partir destes dados está cada vez mais em alta, principalmente no que tange a análise de imagens. O aprendizado supervisionado, caracterizado pelo treino de um modelo através da apresentação de inputs e seus devidos outputs, é uma das formas mais comuns de machine learning.

Usualmente se utiliza aprendizados que possuem pesos para encontrar padrões (ex: Support Vector Machines - SVM e Artificial Neural Networks - ANN), porém também existem redes neurais sem pesos (Weightless Neural Networks - WNN), onde temos como exemplo a WiSARD. Esse classificador se diferencia das redes neurais mais utilizadas por ter uma estrutura onde cada neurônio é baseado em RAM, o que traz como grande vantagem a necessidade de apenas uma iteração de treinamento para cada dado de treino.

Para se treinar e testar um modelo, são utilizados datasets públicos, que usualmente possuem estatísticas de acurácia e erro de outros modelos utilizados, e para o presente trabalho foi utilizado o dataset Kuzushiji-MNIST [Clanuwat et al. 2018], uma variação do dataset MNIST tradicional [LeCun et al. 1998].

Na lista de modelos utilizados com o KMNIST não há nenhum modelo de WNN, portanto este trabalho visa estudar a acurácia que pode ser obtida com o classificador WiSARD e como ele se compara com as redes neurais com peso apresentadas.

Na sequência, é mostrado a estrutura deste relatório. Na seção 2 é apresentado o classificador WiSARD e sua variação ClusWiSARD, na seção 3 apresentamos o dataset KMNIST e as técnicas de pré-processamento utilizadas. Em seguida, na seção 4 é feita uma avaliação de desempenho de cada técnica, na seção 5 apresenta-se a discussão dos resultados encontrados. Finalmente, nas seções 6 e 7, são exibidas as conclusões do trabalho e as sugestões para trabalhos futuros, respectivamente.

2. WiSARD

O classificador WiSARD utiliza-se de princípios neurais para reconhecer padrões a partir de dados, com neurônios baseados em RAM, possuindo como principais componentes um decodificador de endereços, registradores de memória, de dados de entrada e saída. Os dados são armazenados em endereços binários de N entradas, apresentados ao decodificador que possui uma saída com 2^N caminhos possíveis, onde os registradores de entrada e saída de dados são ativados segundo o caminho do decodificador [Costa et al. 2018] [Kappaun et al. 2016].

Ao basear os neurônios em RAM, é estabelecido que para cada padrão de entrada é armazenado um padrão de saída, feito na etapa de treinamento, que podem ser reescritos em outras iterações de treino. Essa definição leva a uma distinção dos neurônios convencionais de ANN por não necessitar de treinamentos complexos, fazendo com que não ocorra generalização na RAM mas as redes de RAMs fazem uma generalização similar a ANN.

Um exemplo de rede de RAM é o discriminador, que consiste em uma camada de T RAMs cada uma com N entradas, mantendo o padrão de armazenamento de 2^N e a camada com padrão de binário T*N bits. Na etapa de treinamento são apresentados os padrões de entrada e armazenadas as saídas, sobrescrevendo os valores iniciais das células (usualmente 0). Com isso, cada RAM armazena parte do padrão de entrada e na etapa de teste a saída do discriminador é o número de RAMs que foram ativadas.

Para o WiSARD em específico, utiliza-se um método de multi-discriminadores, com cada individual treinado para uma classe específica. Na etapa de classificação, o padrão de saída é determinado como o discriminador que apresentar a maior quantidade de RAMs ativadas para um dado de entrada.

2.1. ClusWiSARD

Para evitar a saturação de discriminadores, utilza-se a extensão ClusWiSARD, que consiste na criação de da vários discriminadores para uma mesma classe. Seu funcionamento classificatório é similar ao do WiSARD, com o padrão de saída do discriminador sendo a quantidade de RAMs ativadas pelos dados de entrada.

A principal característica para alívio de RAM está no aprendizado de inputs que não possuem muita semelhança com classes já estabelecidas. Esse processo é realizado em discriminadores diferentes dos principais de cada classe. Desta forma, criam-se subclasses e novos discriminadores à medida que os padrões de entrada forem muito distintos ao já determinados para classificação.

3. Dados de Entrada e técnicas de binarização

Na literatura existem vários datasets inspirados no MNIST [Xiao et al. 2017]. Em particular, o KMNIST é composto por 10 classes de caracteres japoneses de textos antigos, onde cada classe possui 60000 imagens de treino e 10000 imagens de teste, cada imagem sendo composta por 28x28 pixels em escala de cinza abrangendo 10 classes diferentes (Figura 1).

O primeiro desafio importante neste trabalho consistiu em encontrar os métodos apropriados para binarizar as imagens do dataset. Para tanto, consideraram-se em utilizar

as seguintes técnicas de binarização[1]:

1. Thresholding simples: esta técnica consiste na aplicação de um threshold binário, onde para uma determinada imagem em escala de cinza cada pixel é submetido ao mesmo tratamento, em que se o valor do pixel for menor que o valor do threshold determinado então a ele é atribuído o valor 0. Caso contrário é atribuído um valor máximo determinado pelo usuário, que neste trabalho foi 1 (255 em escala de cinza).

2. Thresholding adaptativo: neste caso há uma variação do algoritmo de thresholding simples. Para cada pixel da imagem, é realizado uma análise dos pixels vizinhos, onde o tamanho da vizinhança é determinado pelo usuário, com o objetivo de utilizar o melhor valor de threshold. Após a análise da vizinhança, o valor final de corte é obtido através de duas formas: subtraindo uma constante C (definida pelo usuário) da média de valores dos pixels ao redor, ou subtraindo uma constante C da soma ponderada gaussiana de valores de pixels vizinhos.

3. Método de Otsu: ao utilizar este método, o usuário não precisa escolher o valor de threshold para pré-processamento de imagem, sendo isso feito de forma automática pelo próprio algoritmo. A determinação do valor é feito pelo histograma de escala de cinza da imagem, onde o algoritmo procura o valor de threshold que minimiza a variação ponderada dentro da classe.

4. Temômetro simples: nesta técnica, cada pixel é codificado como um vetor de tamanho d, definido pelo usuário, em que cada posição apresenta o valor de 1 ou 0 a partir do threshold estabelecido para esta posição. Por exemplo, um termômetro que possui um vetor de tamanho 5, terá os thresholds de 51, 102, 153, 204 e 255, de dessa forma que um pixel de intensidade de 145 ao ser codificado será representado pelo vetor 11100.

Adicionalmente, com o objetivo de reduzir os ruídos presentes na imagem, foi empregado um filtro gaussiano, cuja eficacia foi estudada em trabalhos anteriores [Mu and Gilmer 2019] [Finn]

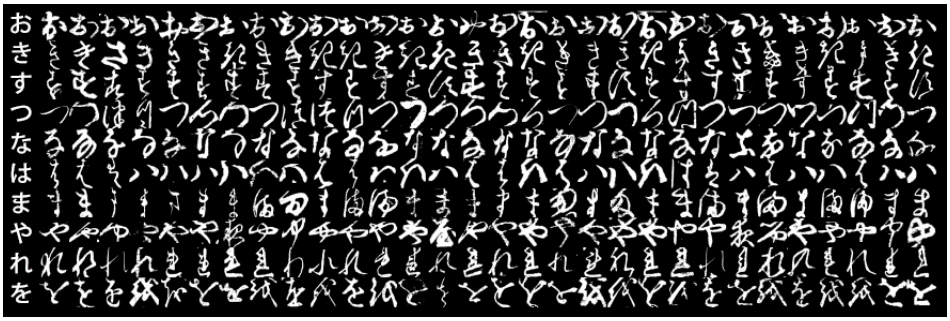


Figura 1. Alguns exemplos do KMNIST, com as respectivas classes por cada fila

[1] https://docs.opencv.org/4.5.2/d7/d4d/tutorial_py_thresholding.html

4. Avaliação de desempenho

Para analisar a acurácia fornecida pela WiSARD, o treinamento e classificação foi realizada em um computador de escritório de propósito geral (Tabela 1). O programa foi implementado na linguagem python 3.9.5 e a versão da biblioteca wisardpkg considerada foi a 1.6.3.

Tabela 1. Informações do computador utilizado

Processador	Intel(R) Core(TM) i3 10100
Número de núcleos	4
Número de threads (Hyper-threading)	8
Clock Base	3.60 GHz
Clock Turbo Máximo	4.30 GHz
Cache L1	128 KB
Cache L2	1MB
Cache L3	6MB
Memoria RAM	16GB DDR4-2666

Como foi mencionado anteriormente, foi utilizado um filtro gaussiano como preprocessamento. Em particular, neste trabalho se utilizou um kernel de tamanho (9,9). No que concerne à binarização, é importante destacar que no caso do Thresholding simples, o Thresholding utilizado foi 1, visto que mostrou melhor desempenho para o cálculo da acurácia em relação a outros valores.

Usando a WiSARD, o maior valor da acurácia obtida com cada técnica foi calculado à medida que o valor de tupla cresce (Figura 2a) atingindo valores máximos próximos a 85% (Tabela 2). No caso da ClusWiSARD, os valores maiores de acurácia encontrados seguem um padrão similar à calculada a WiSARD (Tabela 3). É importante destacar que os dados obtidos representam a média de 5 experimentos (treinamentos e classificações) para cada rede neural. Complementarmente, analisou-se o valor da acurácia para várias resoluções do termômetro, conforme aumenta o tamanho de tupla (Figura 2b). Visto que o aumento na resolução do termômetro significa maior consumo de memória, o máximo valor da resolução empregada foi de 32.

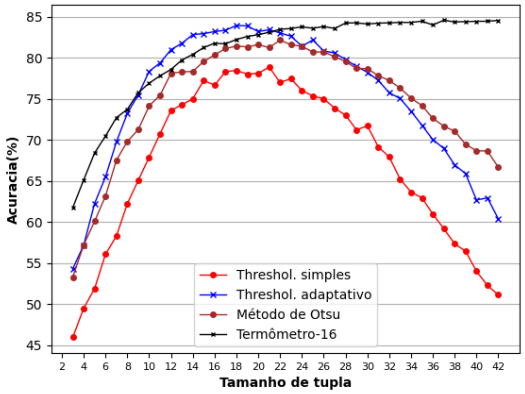
Tabela 2. Informações da acurácia e desempenho - WiSARD

Método	**Thresh. simples**	**Thresh. adaptativo**	**Otsu**	**Termômetro - 16**
Acurácia (%)	78.87	83.95	82.16	84.97
Tamanho de tupla	21	18	22	36
Tempo de treinamento (s)	1.24	1.26	1.14	41.59
Tempo de classificação (s)	0.98	1.10	0.93	15.88
Tempo total (s)	2.22	2.36	2.07	57.47

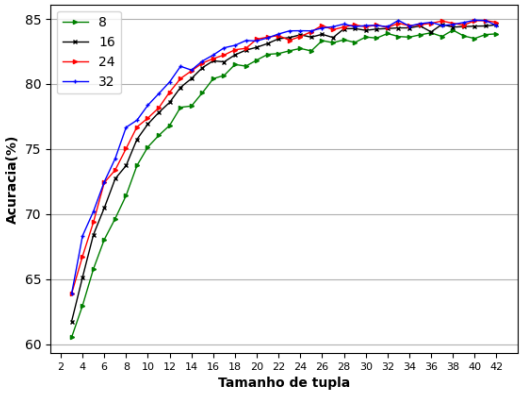
Com o objetivo de visualizar a forma em que a WiSARD treinou e classificou o KMNIST, optou-se por gerar a imagem mental do dataset (Figura 3b) e compara-lo com algumas imagens originais antes do treinamento (Figura 3a).

5. Conclusões

Pelos resultados obtidos, observa-se que a acurácia atingida depende muito do método de binarização utilizado e do tamanho de tupla considerada.



(a) Acurácia usando diferentes métodos de binarização

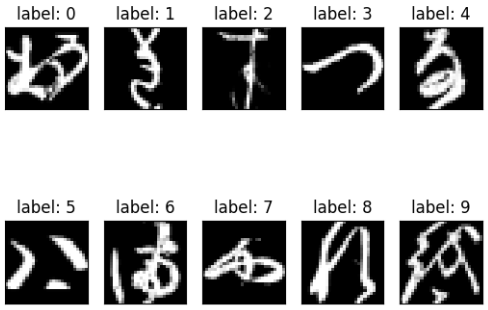


(b) Acurácia utilizando o termômetro para diferentes resoluções

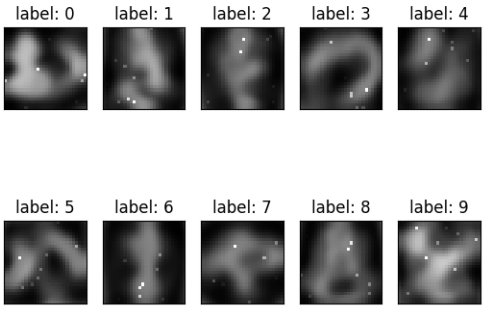
Figura 2. Acurácia conseguida usando a WiSARD

Tabela 3. Informações da acurácia e desempenho - ClusWiSARD

Método	Thresh. simples	Thresh. adaptativo	Otsu	Termômetro - 16
Acurácia (%)	79.60	84.02	82.58	85.01
Tamanho de tupla	20	18	20	36
minScore	0.5	0.5	0.5	0.5
threshold	85	50	50	50
discriminatorLimit	2	5	2	5
Tempo de treinamento (s)	1.65	2.69	1.61	65.52
Tempo de classificação (s)	1.67	4.15	1.73	53.29
Tempo total (s)	3.32	6.84	3.34	118.81



(a) Alguns exemplos do conjunto de treinamento



(b) Imagens Mentais realizadas pela WiSARD

Figura 3. Esboço da imagem mental

Há uma diferença substancial no comportamento da acurácia à medida que o tamanho de tupla se incrementa (Figura 2). Utilizando o método de Otsu e as outras duas técnicas de Thresholding, as curvas encontradas atingem uma acurácia máxima entre 18 e 22 de tamanho de tupla, para depois decrescer. Por outro lado, a curva que representa o uso de termômetro, manifesta um convergência para 85% de acurácia aproximadamente.

Observou-se que incrementar a resolução do termômetro não constitui um aumento significativo da acurácia (Figura 2b) para resoluções maiores de 16. Além disso, o tempo de treinamento e classificação, utilizando o termômetro, é maior em relação aos outros métodos, notavelmente com um aumento maior que 1000% de tempo total do modelo em relação ao Thresholding adaptativo com ClusWiSARD (Tabela 2). Assim, em alguns casos poderia ser mais conveniente utilizar a segunda técnica com a melhor acurácia, que neste caso é o Thresholding adaptativo. Uma situação similar foi encontrado quando utilizado a ClusWiSARD (Tabela 3), onde o aumento de acurácia não é mais de 1% em relação a WiSARD e acrescenta, na melhor das hipóteses testadas, mais de 49% de tempo total do modelo, que talvez inviabilize seu uso em determinadas situações.

6. Trabalhos Futuros

Pretende-se utilizar outras técnicas de pre-processamento e binarização para comparar com os melhores resultados obtidos neste trabalho. Ademais, outras abordagens para complementar este estudo são:

- Realizar estudos visando outras métricas que justifiquem ou descartem os modelos observando a relação entra ganho de acurácia e aumento no tempo total do modelo.
- Analisar o desempenho da WiSARD utilizando programação paralela mediante as diferentes APIs, por exemplo OpenMP e MPI.

Referências

Clanuwat, T., Bober-Irizar, M., Kitamoto, A., Lamb, A., Yamamoto, K., and Ha, D. (2018). Deep learning for classical japanese literature.

Costa, S., Almeida, T., Gramacho, W., and Carvalho, R. (2018). Avaliação da rede neural sem peso wisard na base de dados mnist.

Finn, L. Collaborative filter pre-processing for improved corrupted image classification.

Kappaun, A., Camargo, K., Rangel, F., Firmino, F., Lima, P., and Oliveira, J. (2016). Evaluating binary encoding techniques for wisard. In *2016 5th Brazilian Conference on Intelligent Systems (BRACIS)*, pages 103–108, Los Alamitos, CA, USA. IEEE Computer Society.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Mu, N. and Gilmer, J. (2019). Mnist-c: A robustness benchmark for computer vision. *arXiv preprint arXiv:1906.02337*.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.