
**Universidade do Estado do Rio de Janeiro**

Centro de Tecnologia e Ciências

Faculdade de Engenharia

Tarso Mesquita Machado

# Projeto Dedicado de Redes Neurais Sem Peso Baseadas em Neurônios de Lógica Probabilística Multi-valorada

Rio de Janeiro

2017

Tarso Mesquita Machado

# Projeto Dedicado de Redes Neurais Sem Peso Baseadas em Neurônios de Lógica Probabilística Multi-valorada


Dissertação apresentada, como requisito parcial para obtenção do título de Mestre, ao Programa de Pós-Graduação em Engenharia Eletrônica, da Universidade do Estado do Rio de Janeiro. Área de concentração: Sistemas Inteligentes e Automação.

Orientadora: Prof.ª Dr.ª Nadia Nedjah
Orientadora: Prof.ª Dr.ª Luiza de Macedo Mourelle

Rio de Janeiro
2017

CATALOGAÇÃO NA FONTE
UERJ / REDE SIRIUS / BIBLIOTECA CTC/B

M149 Machado, Tarso Mesquita
Projeto dedicado de redes neurais sem peso baseadas em neurônios de lógica probabilística multivalorada/Tarso Mesquita Machado.- 2017.
156f.

Orientadoras: Nadia Nedjah e Luiza de Macedo Mourelle.
Dissertação (Mestrado) - Universidade do Estado do Rio de Janeiro, Faculdade de Engenharia.

1. Engenharia Eletrônica. 2. Redes neurais(Computação) - Teses. 3. Inteligência artificial - Teses. I. Nedjah, Nadia. II. Mourelle, Luiza de Macedo. III. Universidade do Estado do Rio de Janeiro. IV. Título.

CDU 004.8



Assinatura

Data

Tarso Mesquita Machado

# Projeto Dedicado de Redes Neurais Sem Peso Baseadas em Neurônios de Lógica Probabilística Multi-valorada

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre, ao Programa de Pós-Graduação em Engenharia Eletrônica, da Universidade do Estado do Rio de Janeiro. Área de concentração: Sistemas Inteligentes e Automação.

Aprovado em: 16 de agosto de 2017

Banca Examinadora:

---

Prof.ª Dr.ª Nadia Nedjah (Orientadora)
Faculdade de Engenharia, UERJ

---

Prof.ª Dr.ª Luiza de Macedo Mourelle (Orientadora)
Faculdade de Engenharia, UERJ

---

Prof. Dr. José Gabriel Rodríguez Carneiro Gomes
Programa de Engenharia Elétrica, COPPE/UFRJ

---

Prof.ª Dr.ª Marley Maria Barnardes Rebuzzi Vellasco
Departamento de Engenharia Elétrica, PUC-Rio

Rio de Janeiro
2017

## AGRADECIMENTOS

Agradeço primeiramente aos meus pais, que tiveram todo o trabalho e sacrifício para me educar de modo correto, por investirem em meu futuro e por me ensinarem valores que carregarei comigo por toda a minha vida. Minha mãe, que sempre esteve ao meu lado com braços abertos e seu amor incondicional. Meu pai, que sempre me mostrou, sem mesmo me dizer, como viver a vida e a importância da moral e da ética para o meu crescimento como homem.

Agradeço imensamente às professoras Nadia Nedjah e Luiza de Macedo Mourelle pela paciência, orientação, e pelos ensinamentos dados nas disciplinas cursadas e ao longo do desenvolvido deste trabalho.

Agradeço aos meus colegas de mestrado Alejandra, Alexandre, Luneque, Igor, Joelmir, Pedro e Ramon, pela troca de conhecimentos, pelo companheirismo e pelos momentos de descontração.

Agradeço também ao meu irmão Thiago, pelo companheirismo e pela ajuda nessa jornada. Agradeço aos meus amigos pessoais Bruno, Pedro, Pamela, Tamires, Amanda e João Pedro, pelos ótimos momentos de descontração e pelo apoio.

Agradeço à Kasznar Leonardos Propriedade Intelectual, em especial ao Gustavo Barbosa, pelo incentivo e apoio prestado para a conclusão deste projeto.

Agradeço à FAPERJ e ao PEL-UERJ pela oportunidade e pelos recursos investidos nesta pesquisa.

# RESUMO

Machado, Tarso Mesquita. *Projeto Dedicado de Redes Neurais Sem Peso Baseadas em Neurônios de Lógica Probabilística Multi-valorada.* 2017. 156f. Dissertação (Mestrado em Engenharia Eletrônica) – Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2017.

As Redes Neurais Artificiais (RNA) são modelos computacionais que se espelham no arranjo e na arquitetura do cérebro humano. Para o caso das RNAs convencionais, a informação é propagada entre os neurônios e é ponderada pelos pesos sinápticos entre eles, os quais são ajustados durante a fase de treinamento e refletem o conhecimento adquirido pela rede a respeito de uma aplicação. O problema relacionado ao modelo de RNA convencional reside no fato de que o aprendizado da rede é feito a partir de sucessivas somas e multiplicações. Isto faz com que este modelo de RNA exija grande poder computacional para ser implementado, e ainda faz o aprendizado da rede ser excessivamente lento. Por outro lado, nas Redes Neurais Artificiais Sem Peso (RNSP), a informação aprendida pela rede é armazenada nas memórias RAM de cada um dos neurônios, e o processo de treinamento da mesma consiste geralmente em realizar operações de leitura e escrita. Dessa forma, as RNSPs possuem treinamento mais simples e rápido do que as RNAs convencionais. Entretanto, a grande maioria das discussões a respeito de RNAs é referente àquelas com pesos sinápticos entre os neurônios. Desse modo, existe pouca informação a respeito de implementações de RNSPs, em especial das RNSPs com elementos probabilísticos. O presente trabalho visa analisar o impacto dos parâmetros de projeto de uma RNSP baseada em Neurônios de Lógica Probabilística Multi-valorada (*Multi-Value Probabilistic Logic Node* - MPLN), de modo a estabelecer diretrizes de projeto para a mesma. A solução é obtida por meio da implementação de diversas arquiteturas de RNSP do tipo MPLN em duas aplicações, a saber, o reconhecimento de algarismos manuscritos e a classificação do bit de paridade. Cada uma das arquiteturas implementadas teve um dos parâmetros variado enquanto os demais foram mantidos constantes, de modo a observar o impacto de tal parâmetro na precisão de classificação da rede, épocas de treinamento necessárias para treinar a rede, e o tempo de processamento. O presente trabalho propõe ainda uma modificação na rede MPLN para problemas de múltiplas classes, denominada rede Mod-MPLN. A rede Mod-MPLN é definida por uma mudança no algoritmo de treinamento da rede e pela inclusão de um discriminador específico na saída da rede, mas sem alterar as características intrínsecas da topologia MPLN. De modo a estabelecer as diretrizes de projeto de uma RNSP MPLN e validar a eficácia da rede Mod-MPLN, foram desenvolvidas dez arquiteturas de RNSP para o problema de identificação de algarismos e dez arquiteturas para o problema do bit de paridade, as quais são avaliadas por meio de simulações realizadas no MATLAB®. A partir dos resultados das simulações, são propostas recomendações na escolha dos parâmetros de projeto da rede.

Palavras-chave: Redes Neurais Artificias Sem Peso; Inteligência Artificial; Nó lógico probabilístico; Reconhecimento de Imagens.

## ABSTRACT

Machado, Tarso Mesquita. *Dedicated Project for Neural Networks Based on Multi-valued Probabilistic Logic Node*. 2017. 156f. Dissertação (Mestrado em Engenharia Eletrônica) – Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2017.

Artificial Neural Networks (ANN) are computational models that are mirrored in the arrangement and architecture of the human brain. For the case of conventional ANNs, the information is propagated between neurons and is weighted by the synaptic weights between them, which are adjusted during the training phase and reflect the knowledge acquired by the network regarding the problem to be solved. The problem related to the conventional ANN model lies on the fact that the learning of the network is performed by successive sums and multiplications. This causes such ANN model to require large computational power to be implemented, and causes the learning of the network to be too slow. On the other hand, in the Weightless Neural Networks (WNN), the information learned by the network is stored in the RAM memories of each of the neurons, and the training process of the network usually consists of performing read and write operations. Therefore, WNNs have simpler and faster training than conventional ANNs. However, the vast majority of discussions about ANNs are relative to those with synaptic weights between neurons. Thus, there is little information about the implementations of WNNs, especially WNNs with probabilistic elements. The present work aims to analyze the impact of the project parameters of a WNN based on Multi-valued Probabilistic Logic Neurons (MPLN), in order to establish design project for the same. The solution is obtained through the implementation of several WNN architectures of the MPLN type for two applications, namely the recognition of handwritten numbers and the classification of the parity bit. Each of the implemented architectures had one of the parameters varied while the others were kept constant, in order to observe the impact of such parameter on the classification accuracy, training epochs necessary to train the network and processing time. The present work further proposes a modification in the MPLN network for multi-class problems, defines as the Mod-MPLN network. The Mod-MPLN network is defined by a change in the network training algorithm and by the inclusion of a specific discriminator at the network output, without changing the intrinsic characteristics of the MPLN topology. In order to establish the design guidelines for an MPLN WNN and to validate the effectiveness of the Mod-MPLN network, ten WNN architectures were developed for the handwritten number identification problem and ten architectures for the parity bit problem, which are evaluated by means of simulations performed in MATLAB®. From the results of the simulations, recommendations are proposed for the choice of project parameters of the network.



Keywords: Weightless Neural Network; Artificial Inteligence; Probabilistic Logic Node; Image Recognition.

# LISTA DE FIGURAS

# LISTA DE FIGURAS

# LISTA DE TABELAS

# LISTA DE TABELAS

# LISTA DE ALGORITMOS

# LISTA DE SIGLAS

| | |
|---|---|
| FPGA | *Field Programmable Gate Array* |
| GB | Grupo de imagens de qualidade boa |
| GM | Grupo de imagens de qualidade média |
| GR | Grupo de imagens de qualidade ruim |
| MNIST | *Modified National Institute of Standards and Technology* |
| Mod-MPLN | *Modified Multi-valued Propabilistic Logic Node* |
| MPLN | *Multiple-valued Probabilistic Logic Neuron* |
| PLN | *Probabilistic Logic Node* |
| RAM | *Random Access Memory* |
| RNA | Rede Neural Artificial |
| RNSP | Rede Neural Artificial Sem Peso |
| WiSARD | *Wilkie, Stonham & Aleksander's Recognition Device* |

# SUMÁRIO

# SUMÁRIO

# INTRODUÇÃO

Com o avanço da tecnologia e necessidade de soluções cada vez mais rápidas e completas, os sistemas convencionais começaram a encontrar limitações para atender as necessidades da sociedade moderna. Cada vez mais, surgem situações onde não há um modelo matemático consistente capaz de modelar um problema prático, o que impõe limites óbvios para solucioná-lo, inclusive em soluções computacionais. Para a resolução deste tipo de problemas, a estrutura do cérebro humano demonstra ser mais adequada, pois trabalha de modo paralelo, além da capacidade de aprender através de exemplos (SILVA; SPATI; FLAUZINO, 2010). Esta capacidade que diferencia o cérebro humano do funcionamento dos sistemas eletrônicos conhecidos é chamada de inteligência, e vem cada vez mais tentando ser implementada ou simulada em sistemas eletrônicos, de modo que estes possam vir a ter as capacidades de processamento do cérebro humano. Destas tentativas, surge o conceito de inteligência artificial.

O conceito de inteligência artificial está geralmente relacionado com a capacidade de generalização e aprendizado (BRAGA; CARVALHO; LUDERMIR, 2000), de forma similar ao funcionamento do cérebro humano. Por exemplo, desde criança aprendemos o que é uma cadeira, um objeto geralmente com pernas de apoio, um assento e um encosto o qual é utilizado para uma pessoa sentar. Caso seja apresentado um novo modelo de cadeira, o qual nunca tenha sido visto por uma pessoa, esta pessoa provavelmente será capaz de identificar que aquele objeto é de fato uma cadeira, a partir de uma correspondência das características do objeto observado com aquelas de seu conhecimento a respeito do objeto cadeira. Esta forma de inteligência trata-se de uma generalização do conceito de cadeira para objetos não conhecidos pela pessoa naquele momento. A partir da capacidade de generalização, é possível solucionar problemas complexos através do conhecimento de soluções similares conhecidas para aquele problema, mas sem, de fato, modelar e solucionar o problema em si.

Neste contexto, foi desenvolvido o conceito de redes neurais artificias (RNAs), o qual é um dos mecanismos mais utilizados para solucionar problemas através da generalização e aprendizado. Como o próprio nome diz, uma RNA é um modelo computacional que se espelha no arranjo e na arquitetura do cérebro humano (HAYKIN, 2001). A partir do conhecimento do funcionamento do modelo do neurônio biológico, foi definido para as RNAs um modelo de neurônio artificial, em especial as conexões entre os neurônios que simulam as sinapses. Assim como no cérebro humano, nas RNAs um grande número de neurônios trabalha de forma conexa para processar a informação e fornecer um resultado (HAYKIN, 2001).

Este ramo da computação teve seu primeiro apogeu com a criação das redes Perceptron (ROSENBLATT, 1962; SILVA; SPATI; FLAUZINO, 2010). Entretanto, o baixo poder computacional da época não permitiu a difusão desta técnica no ambiente prático. Somente no final dos anos 1980 é que a área voltou a despertar um interesse significativo nos pesquisadores (HAYKIN, 2001), quando foi descoberto um algoritmo capaz de aprender e solucionar problemas não linearmente separáveis.

Pode-se dizer que redes neurais artificiais consistem em um modo de abordar a solução de problemas de inteligência artificial (BARRETO, 2002). Neste caso, em lugar de tentar programar um computador digital de modo a fazê-lo imitar um comportamento inteligente, procura-se construir um sistema que tenha circuitos modelando os circuitos cerebrais e espera-se ver um comportamento inteligente emergindo, aprendendo novas tarefas, errando, fazendo generalizações e descobertas, e frequentemente ultrapassando seu professor (BARRETO, 2002). Da mesma forma, estes circuitos neurais artificiais poderão se auto-organizar, quando apresentados a ambientes diversos, criando suas próprias representações internas e apresentar comportamentos imprevisíveis.

As RNAs têm sido usadas em inúmeras áreas do conhecimento, como processamento de sinais, análise de imagens médicas, sistemas de diagnóstico e previsões de séries temporais (SILVA; SPATI; FLAUZINO, 2010). Um problema tipicamente solucionável com RNAs são os problemas de reconhecimento de padrões ou de aproximação de uma função. Para os casos de reconhecimento de padrões, uma RNA deve classificar um padrão de entrada como pertencendo a uma determinada classe, mesmo sem nunca ter visto o referido padrão. As principais características e vantagens das RNAs são:

- Adaptação por experiência: é a capacidade de ajuste dos parâmetros internos da rede, tipicamente seus pesos sinápticos, a partir de sucessivas tentativas e erros (SILVA; SPATI; FLAUZINO, 2010). O modo como os pesos sinápticos dos neurônios são ajustados pode variar de acordo com o tipo de implementação e o método de aprendizado utilizado nas redes;

- Capacidade de aprendizado: através da aplicação de um método de treinamento, a rede consegue ajustar seus pesos sinápticos de acordo com os exemplos apresentados à rede até que não haja mais erros, ou seja, a rede se comporte como o sistema desejado (SILVA; SPATI; FLAUZINO, 2010);

- Habilidade de generalização: após o processo de treinamento da rede, esta se torna capaz de resolver diferentes problemas e situações do sistema para o qual foi treinada, sem de fato modelar o problema (HAYKIN, 2001);

- Organização de dados: observando diversos padrões diferentes de entradas e suas particularidades, a rede é capaz de se auto-organizar de modo responder de modos diferentes a estes diversos padrões e particularidades (SILVA; SPATI; FLAUZINO, 2010);

- Tolerância a falhas: devido ao elevado nível de interconexões entre os neurônios artificiais, o erro em um ou em um pequeno número de neurônios pode não trazer um erro propagado suficientemente para comprometer o funcionamento correto do sistema (HAYKIN, 2001), tornando a rede mais robusta que sistemas convencionais;

- Armazenamento distribuído: o conhecimento a respeito do comportamento de um determinado processo dentro da arquitetura da RNA é realizado de forma distribuída entre os diversos neurônios, evitando assim que a falha em um neurônio possa comprometer todo o funcionamento da rede (SILVA; SPATI; FLAUZINO, 2010);

- Facilidade de prototipagem e simulação: a implementação da maioria das arquiteturas neurais pode ser realizada de maneira relativamente fácil em *hardware* ou *software*, pois após o processo de treinamento, os seus resultados são geralmente obtidos através de simples operações matemáticas (SILVA; SPATI; FLAUZINO, 2010).

O problema relacionado ao modelo de Rede Neural Artificial convencional reside no fato de que o aprendizado da rede é feito a partir de sucessivas somas e multipli-

cações, relacionadas a parâmetros do modelo de neurônio artificial, chamados de pesos sinápticos. Isto faz com que este modelo de RNA exija alto poder computacional para ser implementado, e ainda faz o aprendizado da rede ser consideravelmente lento.

Assim, os contratempos do modelo de RNA convencional levaram a atenção dos pesquisadores de volta a outro modelo de Redes Neurais Artificiais, de certo modo esquecido pelas limitações tecnológicas da época em que foi vislumbrado por alguns pesquisadores (ALEKSANDER; MORTON, 1989), a chamada Rede Neural Artificial Sem Pesos (RNSP), ou baseadas em memórias de acesso aleatório (RAM).

As RNSPs tiveram origem nas máquinas de amostragem de n-tuplas de Bledsoe e Browning (BLEDSOE; BROWING, 1959), onde foi proposto que uma entrada de $n$ bits fosse utilizada para formar uma tupla de $n$ bits para endereçar a memória de um neurônio. Igor Aleksander (ALEKSANDER, 1966, 1983) teve um grande interesse em redes de aprendizagem adaptativa usando máquinas de amostragem de n-tuplas, e introduziu o Microcircuito de Lógica Adaptativa Armazenada (*Stored Logic Adaptative Microcircuit - SLAM*), o qual foi produzido especificamente para fins de pesquisa, uma vez que na época não havia memórias em circuito integrado disponíveis para sua implementação.

As RNSPs compartilham das principais características das redes Hopfield (buscar mínimos de energia em tempo real), máquinas de Boltzmann (treinamento de camadas ocultas, escapar de mínimos locais) e retro-propagação de erro (aprendizado a partir de erros) (LUDERMIR; OLIVEIRA, 1994). Além disso, aprender a partir dos erros em RNSPs é mais rápido e mais direto do que com o algoritmo *backpropagation* das RNAs convencionais (MYERS, 1988). Não obstante, as RNSPs são simples de se implementar em *hardware* devido à alta disponibilidade de memórias RAM no mercado.

Grande parte dos trabalhos iniciais sobre RNSPs foi realizado por Aleksander e colaboradores nas Universidades de Canterbury e Brunel no Reino Unido no final dos anos sessenta e setenta (LUDERMIR; OLIVEIRA, 1994). Em (ALEKSANDER; ALBROW, 1968), foi investigada a utilização de redes *feed-forward* de RAM para o reconhecimento de números manuscritos. Em (STONHAM, 1975), também foram utilizadas redes RAM *feed-forward* como extrator de recursos na classificação dos espectros de massa. Em (STONHAM; FAIRHUST, 1976), foi desenvolvido um sistema de classificação para caracteres alfa-numéricos baseado em técnicas de aprendizagem da rede. Nestas aplicações menci-

onadas, um conjunto de redes RAM é normalmente utilizado como discriminadores (ou extratores de características), com um discriminador por classe de resposta.

No final da década de 1970, a construção de uma rede capaz de lidar com imagens de alta qualidade se tornou economicamente viável à medida que as memórias RAMs se tornaram muito mais baratas e de maior capacidade (LUDERMIR; OLIVEIRA, 1994). Em 1979, foi desenvolvida a rede WiSARD (*Wilkie, Stonham & Aleksander's Recognition Device*), sendo que seu protótipo foi concluído em 1981 (ALEKSANDER; THOMAS; BOWDEN, 1984). A rede WiSARD é baseada em $n$ neurônios RAM com entradas de tupla de $k$bits. Durante o funcionamento da rede, uma imagem de entrada é armazenada e digitalizada, sendo que cada *pixel* da imagem é representado por um único bit. Cada uma das $k$ entradas de um neurônio RAM recebe um valor binário de um *pixel* da imagem, utilizando um mapeamento aleatório, porém fixo. Dessa forma, cada um dos $n$ neurônios da rede é responsável por aprender e classificar uma parte da imagem a ser reconhecida. A rede WiSARD é capaz de determinar se uma imagem pertence a uma classe específica através do cálculo de quantos dos $n$ neurônios da rede identificaram as suas respectivas partes da imagem como pertencentes à dita classe.

Em (ALEKSANDER, 1989), foi proposta a introdução de um elemento probabilístico no neurônio RAM, o que resultou no denominado Neurônio Lógico Probabilístico (PLN). A principal característica de um neurônio PLN é a presença de um terceiro estado, o estágio indeterminado $u$, o qual corresponde a uma saída gerada com igual probabilidade entre os níveis lógicos '1' e '0' para representar padrões para os quais a rede ainda não foi treinada.

Para a implementação desta rede, é necessário que cada posição de memória possua dois bits para representar as saídas possíveis: '0', '1' e '$u$'. Assim, o conteúdo da memória representa a probabilidade do neurônio emitir uma saída de nível lógico '1' quando o referido conteúdo de memória é endereçado por uma entrada. As redes PLN obtiveram sucesso notável em diversas áreas de aplicação (LUDERMIR; OLIVEIRA, 1994), e sua principal vantagem sobre as redes RAM é que as redes PLN são iniciadas com o valor '$u$' armazenado na memória, ao invés do valor '0'. Dessa forma, é possível distinguir entre as posições treinadas e não treinadas na memória. Exemplos de algoritmos de treinamento para a rede PLN podem ser encontrados em (ALEKSANDER; MORTON, 1989) e em (AL-ALAWI; STONHAM, 1989).

A partir do modelo de RNSP PLN, em (MYERS, 1989; ALEKSANDER; MORTON, 1989), foi proposto o modelo de RNSP com neurônio lógico probabilístico multi-valorado (*Multiple-valued Probabilistic Logic Neuron* - MPLN), o qual pode ser considerado uma generalização do modelo PLN por dois motivos. Primeiramente, a rede MPLN possui um número de $q$ bits armazenados em cada posição de memória (ao contrário dos dois bits da rede PLN). Este número de $q$ bits irá representar a probabilidade do neurônio emitir uma saída de nível lógico '1' quando a posição de memória é acessada por uma tupla de entrada. A segunda diferença é que, devido à gama de valores possíveis armazenados na posição de memória, a função de ativação é ligeiramente diferente do modelo PLN.

Uma das vantagens do modelo MPLN sobre o PLN é que os neurônios podem armazenar probabilidades mais graduadas com relação ao modelo PLN, o qual armazena apenas as probabilidades 0%, 50% e 100% de uma saída com nível lógico '1'. Além disso, o modelo MPLN possibilita que o treinamento da rede consista em modificações incrementais no conteúdo da memória, o que traz uma certa resiliência no aprendizado de informações pela rede.

Em vista do exposto, nota-se que os sistemas eletrônicos capazes de apresentar um comportamento inteligente vêm sendo objeto de inúmeras pesquisas e aprimoramentos. Nesse contexto, as Redes Neurais Artificiais são uma das metodologias mais conhecidas e bem sucedidas atualmente, com diversas publicações e com aplicações práticas e comercialmente viáveis. Apesar disto, a grande maioria das discussões a respeito de Redes Neurais Artificiais é referente àquelas com pesos sinápticos entre os neurônios. As RNSPs, apesar de seu conceito ter sido desenvolvido há muito tempo atrás, voltaram apenas a ser estudadas nos últimos anos, devido aos avanços na área de computação, principalmente no desenvolvimento das memórias RAM. Ainda assim, existem poucos trabalhos sobre abordagens de RNSPs quando comparadas às RNAs com peso, apesar de a primeira apresentar vantagens significativas frente a última, tal como a velocidade de treinamento e flexibilidade.

Nesse sentido, a partir da literatura estudada, nota-se que poucos autores atentam a prover diretrizes de como projetar uma RNSP, tal como o tamanho da tupla a ser utilizada, taxa de treinamento, algoritmo de treinamento, o tamanho da palavra (para o caso de rede MPLN), etc. Assim, é observado que a literatura a respeito de RNSP é

essencialmente teórica, o que de certa forma afasta a atenção para a mesma sobre novas pesquisas e desenvolvimentos.

Dessa forma, esta dissertação tem como objetivo apresentar uma análise sobre as diferentes topologias de uma RNSP MPLN, analisando os impactos da variação de cada parâmetro na eficácia da rede. O modelo de RNSP MPLN foi escolhido por possuir boa capacidade de generalização e possibilidade de construção e implementação simplificadas. Serão estudadas diversas arquiteturas das redes para diversas aplicações, tal como reconhecimento de imagens, de modo a desenvolver diretrizes no projeto de uma RNSP MPLN para as referidas aplicações estudadas. É ainda outro objetivo desta dissertação analisar as limitações da topologia de RNSP do tipo MPLN, assim como propor modificações na mesma, de modo a melhorar sua eficiência e eficácia nas aplicações abordadas nesta dissertação. O restante desta dissertação está organizado em seis capítulos, cujos conteúdos são brevemente descritos a seguir.

Inicialmente, no Capítulo 1, são apresentados os principais conceitos sobre RNAs e RNSPs, necessário para o entendimento do presente trabalho e para o desenvolvimento do mesmo.

No Capítulo 2 são apresentados diversos trabalhos relacionados ao problema de classificação de imagens através de RNSPs, utilizando as redes WiSARD, PLN, MPLN e outras variações desenvolvidas pela literatura. Nesse capítulo, os trabalhos são organizados segundo a topologia de neurônio RAM utilizada, a saber, se são baseadas no modelo RAM convencional, tal como a rede WiSARD, ou utilizando técnicas probabilísticas, tal como a rede PLN.

O Capítulo 3 discorre sobre os principais aspectos no projeto de uma RNSP, onde são apresentados os principais parâmetros a serem definidos durante o projeto de uma RNSP, bem como seu possível impacto na eficácia da rede. Neste capítulo, é também apresentada a limitação da rede MPLN para o problema de reconhecimento de múltiplas classes, onde é sugerida uma modificação da arquitetura MPLN para mitigar suas deficiências.

No Capítulo 4, são apresentados os aspectos de implementação das arquiteturas de RNSP testadas ao longo deste trabalho. São descritas as arquiteturas de rede testadas, conjunto de dados utilizados, bem como as técnicas de pré-processamento empregadas em tais conjuntos.

O Capítulo 5 apresenta os resultados obtidos através da simulação de todas as arquiteturas de rede, variando os parâmetros de interesse para observar seu impacto na eficiência e eficácia da rede. Os resultados obtidos nas simulações são comparados entre si de modo a determinar quais parâmetros obtiveram melhores resultados. A partir destes resultados, são realizadas as devidas análises de modo a determinar diretrizes de projeto para as RNSPs.

Finalmente, o Capítulo 6 apresenta as conclusões acerca dos testes realizados e sobre a relação dos parâmetros de projeto e a eficiência das arquiteturas de rede, bem como algumas perspectivas e possibilidades de trabalhos futuros.

# Capítulo 1

# REDES NEURAIS ARTIFICIAIS

AS Redes Neurais Artificias (RNAs) são modelos computacionais que se espelham no arranjo e na arquitetura do cérebro humano (SILVA; SPATI; FLAUZINO, 2010). A partir do conhecimento do funcionamento do modelo do neurônio biológico, foi definido para as RNAs um modelo de neurônio artificial, em especial as conexões entre os neurônios que simulam as sinapses. Assim como no cérebro humano, nas RNAs um grande número de neurônios trabalha de forma conexa para processar a informação e fornecer um resultado (HAYKIN, 2001).

As RNAs podem ser classificadas em dois tipos: o primeiro sendo as RNAs com pesos, ou RNAs convencionais, e o segundo sendo as RNAs sem peso (RNSP), as quais são o foco principal desta dissertação. Ambas as RNAs compartilham diversas características relacionadas ao modelo de uma RNA, tal como uma fase de treinamento e uma fase de teste. A principal diferença entre as RNA com peso e sem peso reside na forma como a informação é armazenada na rede, conforme será visto em detalhes ao longo deste capítulo. Na Seção 1.1, são abordados os conceitos principais de uma RNA convencional. Na Seção 1.2, é apresentado o conceito de RNSP e suas principais características. Na Seção 1.3, é apresentada a arquitetura de RNSP WiSARD, uma das primeiras arquiteturas de RNSP propostas e até hoje utilizada em pesquisas e trabalhos publicados, sendo a arquitetura de RNSP com maior número de publicações. Na Seção 1.4, é abordada a arquitetura de RNSP do tipo Nó Lógico Probabilístico (*Probabilistic Logic Node* - PLN), onde é proposto o conceito de um comportamento probabilístico para o neurônio da rede. Ainda, na Seção 1.5, é estudada a Arquitetura de RNSP com Nó Lógico Probabilístico Multi valorado (*Multi-Value Probabilistic Logic Node - MPLN)*, a qual é uma generalização da arquitetura PLN e é a arquitetura utilizada para a análise proposta nesta dissertação. Por fim, na

seção 1.6 são feitas considerações a respeito das arquiteturas de RNA apresentadas, em especial sobre o modelo RNSP MPLN.

## 1.1 Redes neurais artificiais convencionais

As Redes Neurais Artificias tiveram seu primeiro apogeu com a criação das redes *Perceptron* (BRAGA; CARVALHO; LUDERMIR, 2000; ALEKSANDER; MORTON, 1989). Entretanto, o baixo poder computacional da época não permitiu a difusão desta técnica no ambiente prático. Somente no final dos anos 1980 é que a área voltou a despertar um interesse significativo nos pesquisadores (SILVA; SPATI; FLAUZINO, 2010). Foi inclusive nesta época que foi desenvolvido um algoritmo que permitia ajustar os pesos em uma rede com mais de uma camada neural, podendo assim solucionar problemas mais complexos, como o caso da função lógica XOR. Este algoritmo foi denominado *backpropagation* (BRAGA; CARVALHO; LUDERMIR, 2000), e sua praticidade e facilidade de implementação o faz possuir grande aplicabilidade até os dias de hoje.

Como toda RNA, as RNAs convencionais possuem duas fases de operação: a fase de aprendizado ou treinamento, quando os parâmetros da rede são ajustados de acordo com as entradas apresentadas, e a fase de uso ou teste, na qual a rede é utilizada para executar alguma tarefa. Desse modo, a composição final de uma RNA não é pré-determinada, sendo função de suas entradas de treinamento. A sua estrutura é formada por nós, também chamados de neurônios, dispostos em uma ou mais camadas interligadas por conexões denominadas pesos sinápticos. As principais características que levam à solução de problemas através de RNAs são sua capacidade de aprendizagem com um número reduzido de exemplos, e generalização, responsável por a rede responder conforme o esperado a entradas desconhecidas.

### 1.1.1 Estrutura de uma rede neural artificial convencional

De maneira simples, uma rede neural artificial é constituída de unidades elementares conectadas entre si (neurônios), que recebem dados de entrada, trabalham esses dados utilizando diversas operações matemáticas, dependendo de sua arquitetura, e produzem uma saída (SILVA; SPATI; FLAUZINO, 2010). O número de entradas, de neurônios ou de saídas irá depender da complexidade do problema a ser solucionado pela rede. A Figura 1 apresenta um esquema geral de como pode ser representada uma RNA.

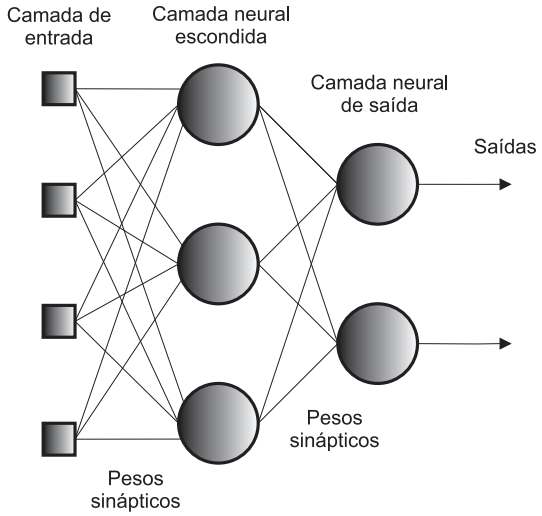



Figura 1: Representação de uma RNA com pesos

Basicamente, uma rede neural artificial pode ser dividida em três partes, denominadas de camadas, as quais são nomeadas da seguinte forma:

- Camada de entrada: é a camada responsável pelo recebimento de informações (dados), sinais característicos ou medições advindas do meio externo. A camada de entrada não possui neurônios artificiais;
- Camadas escondidas, intermediárias, ocultas ou invisíveis: são aquelas que possuem a função de extrair as características associadas ao processo. Quanto maior a quantidade de neurônios nas camadas intermediárias e o número de camadas intermediárias, sistemas mais complexos podem ser implementados pela rede neural;
- Camada de saída: camada também constituída de neurônios, sendo responsável pela produção e apresentação dos resultados finais da rede, que são provenientes dos cálculos efetuados pelos neurônios das camadas anteriores.

### 1.1.2 Neurônio artificial

Um neurônio artificial realiza funções simples, como coletar os sinais existentes em suas entradas, ponderá-los de acordo com sua relevância para o sistema e fornecer saídas de acordo com a função de ativação escolhida. A Figura 2 ilustra a estrutura básica da modelagem de um neurônio artificial.

O neurônio artificial é constituído de sete elementos básicos, os quais são apresentados a seguir:

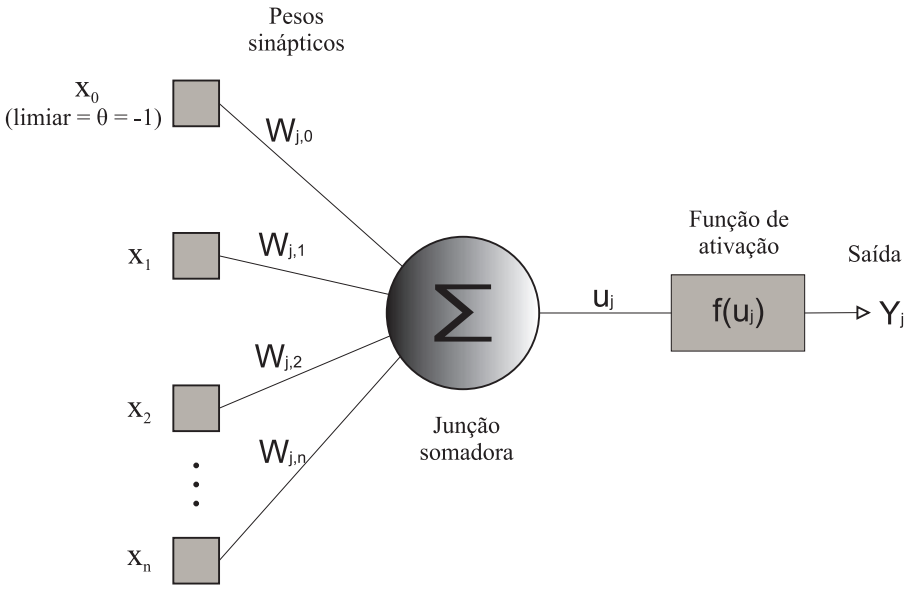



Figura 2: Modelo de neurônio artificial com pesos

- Sinais de entrada $x_1$, $x_2$,..., $x_n$: são os sinais ou medidas provenientes do meio externo, ou seja, a entrada do sistema, que representam os valores assumidos pelas variáveis de um sistema específico;

- Pesos sinápticos $w_{j,1}$, $w_{j,2}$,..., $w_{j,n}$: são os valores que servirão para ponderar o sinal recebido por cada um dos neurônios da rede, permitindo-se comparar a sua relevância em relação a outros neurônios e à própria saída da rede;

- Junção somadora ou Combinador linear $\sum$: sua função é somar todos os sinais que foram ponderados pelos respectivos pesos sinápticos a fim de produzir um valor de potencial de ativação;

- Limiar de ativação (*bias*): é um valor pré-determinado que especifica qual será o valor mínimo produzido pelo combinador linear para que este possa gerar um valor de disparo em direção à saída do neurônio.

- Potencial de ativação $u_j$: é o resultado produzido pela diferença dos valores do combinador linear e do limiar de ativação;

- Função de ativação $f(.)$ ou $g(.)$: sua função é transformar a saída do neurônio em uma forma de onda já conhecida, escolhida de acordo com as especificações de cada

sistema a ser modelado. As funções de ativação mais comuns são o degrau unitário, degrau bipolar, rampa simétrica, função logística e tangente hiperbólica;

- Sinal de saída ($y_j$): consiste do valor final produzido pelo neurônio devido a um determinado conjunto de sinais de entrada, tendo sua forma limita pela função de ativação do neurônio. O sinal de saída de um neurônio pode ainda ser utilizado como entrada para outros neurônios em sistemas mais complexos.

### 1.1.3 Treinamento de uma rede Neural artificial convencional

O treinamento de uma rede neural consiste da aplicação de um conjunto de passos ordenados com o intuito de ajustar os pesos sinápticos e os limiares de seus neurônios. Tal processo de ajuste, também conhecido como algoritmo de aprendizagem, tem como objetivo a generalização de soluções a serem produzidas pelas suas saídas, cujas respostas são representativas do sistema físico em que estas estão mapeando (SILVA; SPATI; FLAUZINO, 2010). Após o treinamento, a rede então será capaz de produzir uma saída próxima daquela esperada a partir de quaisquer sinais inseridos em suas entradas.

Existem dois tipos gerais de treinamento de redes neurais, o primeiro sendo conhecido como treinamento supervisionado, o qual consiste em um método onde, para cada amostra de treinamento, são necessárias as respectivas saídas desejadas. No treinamento supervisionado, para cada entrada, a rede produzirá uma saída, sendo que o valor desta saída será comparado com o valor da saída desejada. A partir desta comparação, será calculado um erro, o qual é utilizado para ajustar os pesos sinápticos da rede. O segundo método é conhecido como treinamento não-supervisionado, onde inexistem as saídas desejadas e, consequentemente, a própria rede deve observar e se auto-organizar em relação às particularidades existentes entre cada elemento do conjunto total de amostras, identificando subconjuntos que contenham similaridades.

### 1.1.4 Funções de ativação de uma rede neural artificial convencional

Conforme explicado na Seção 1.1.2, cada neurônio processa sua informação através de uma soma ponderada de suas entradas. O valor somado deverá ser trabalhado por uma função capaz de normalizar este valor, sendo tal função chamada de função de ativação. As funções de ativação podem ser divididas entre: funções parcialmente diferenciáveis, ou seja, aquelas que possuem pontos onde suas derivadas de primeira ordem não existem,

e funções totalmente diferenciáveis, ou seja, aquelas que possuem derivada de primeira ordem conhecidas em todos os pontos de seu domínio. Dentre as funções de ativação parcialmente diferenciáveis, as mais conhecidas e utilizadas são: a função degrau unitário, a função degrau bipolar e a função rampa simétrica. Dentre as funções de ativação totalmente diferenciáveis, as mais conhecidas e utilizadas são: a função logística ou sigmoide e a função tangente hiperbólica (SILVA; SPATI; FLAUZINO, 2010).

## 1.2 Redes neurais artificiais sem peso

As Redes Neurais Artificias Sem Peso (RNSP) surgem como uma alternativa às RNAs com peso, de modo a apresentar um treinamento mais rápido e que necessita de menor poder computacional. Uma RNSP é baseada em neurônios artificiais que possuem entradas e saídas binárias e sem pesos nas conexões entre seus neurônios. O conhecimento de um neurônio de uma RNSP é armazenadas em uma *look-up table* (tabela verdade), a qual pode ser implementada utilizando memórias de acesso aleatório (*Random Access Memory* - RAM) disponíveis no mercado. Assim, o processamento de informação de um neurônio deste modelo consiste no endereçamento de sua *look-up table* de acordo com um padrão de entrada, e a produção da saída do neurônio com base no respectivo conteúdo da posição da *look-up table* acessada (BRAGA; CARVALHO; LUDERMIR, 2000).

A principal diferença entre as RNSPs e as demais redes neurais consiste na localização da informação aprendida. Enquanto nas RNAs convencionais a mesma se encontra nos pesos das conexões, em RNSPs a informação é armazenada nas *look-up tables*. Além disso, nas RNSPs as entradas são sempre discretas e seus neurônios são capazes de computar todas as funções booleanas de suas entradas digitalizadas. Já as redes neurais com peso possuem entradas e pesos assumindo quaisquer valores reais. Nota-se ainda que um único neurônio com pesos é capaz de calcular apenas problemas com padrões de entrada formando conjuntos linearmente separáveis.

### 1.2.1 O neurônio artificial sem peso ou RAM

Diferentemente dos neurônios artificiais com peso, que calculam sua saída com base na entrada e os pesos entre suas conexões, nos neurônios sem peso o processamento da saída do neurônio depende apenas de sua entrada e dos valores armazenados na RAM corres-

pondente ao neurônio. Ao invés dos pesos sinápticos, os parâmetros ajustáveis em um neurônio RAM consistem nos próprios conteúdos da RAM do neurônio.

A Figura 3 ilustra um neurônio baseado em RAM de $N$ bits de entrada, os quais endereçam a memória RAM do neurônio. Para que o neurônio possua uma posição de memória que possa ser endereçada por todas as combinações dos $N$ bits de entrada, o neurônio deverá possuir uma memória de $2^N$ posições endereçáveis pelas entradas do neurônio. Na ilustração, um sinal binário $I = I_1, I_2, ..., I_N$ na entrada do neurônio será o responsável por endereçar as $M[0]$, $M[1]$ ... $M[2^N - 1]$ posições de memória RAM do neurônio. O conteúdo da memória acessada será então processado pela função de ativação do neurônio RAM de modo a produzir uma saída.

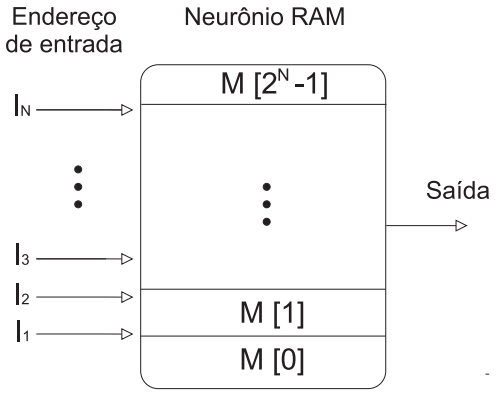



Figura 3: Modelo de um neurônio artificial sem peso

A função de saída $s$ da RNSP geralmente utilizada é a função identidade, podendo ser definida na Equação 1:

$$s = \begin{cases} 0 & \text{se } M[I] = 0 \\ 1 & \text{se } M[I] = 1 \end{cases} \quad (1)$$

Dessa forma, o bit $M[I]$ armazenado na posição de memória endereçada por $I$ representa a saída ou resposta do neurônio à entrada I, isto é, $s = M[I]$. Em outras palavras, a função executada pelo neurônio é determinada pelo conteúdo da memória RAM.

O neurônio sem peso, ou baseado em RAM, conforme definido acima, pode computar todas as funções binárias de suas entradas, enquanto o neurônio artificial convencional pode apenas computar funções linearmente separáveis. É importante notar que um único neurônio baseado em RAM não possui capacidade de generalização. Foi visto que, em

termos de RNA, a generalização pode ser entendida como a capacidade de prover a saída correta para um padrão de entrada não conhecido com base nos padrões já conhecidos e utilizados para treinar a rede. Os neurônios RAM convencionais produzem a saída correta apenas para os padrões de entrada armazenados na fase de treinamento. Entretanto, sabe-se que as RNSPs compostas por diversos neurônios RAM possuem capacidade de generalização (BRAGA; CARVALHO; LUDERMIR, 2000).

Para uma RNSP com diversos neurônios na camada de entrada, a entrada da rede é dividida em diversas partes, chamadas de tuplas, as quais serão utilizadas para endereçar cada um dos neurônios da rede.

### 1.2.2 Fase de treinamento de uma RNSP

O treinamento mais utilizado para as RNSP é o treinamento supervisionado, o qual é análogo ao treinamento das RNAs com peso explicado na Seção 1.1.3. Para cada amostra de treinamento, existe uma saída desejada para aquela amostra, e os parâmetros da rede são ajustados com base na saída desejada. O treinamento das RNSP tradicionais consiste essencialmente em substituir os valores inicialmente armazenados na memória RAM (*look-up table*) pelos valores correspondentes à saída desejada para o neurônio.

Antes de a rede ter sido treinada, o conteúdo de sua memória irá possuir um valor inicial, o qual pode ser predeterminado ou iniciado aleatoriamente. Para os casos onde o neurônio possui posições da memória compostas de apenas um bit, ou seja, a palavra da memória possui apenas um bit, este deverá ser iniciado com o valor lógico '0'. Durante o treinamento, a cada amostra apresentada para a rede, uma tupla desta amostra será utilizada como entrada de cada um dos neurônios da camada de entrada da rede. A tupla que é apresentada a um neurônio será responsável por endereçar o neurônio, de modo a acessar o conteúdo de memória do neurônio correspondente ao endereço fornecido pela tupla.

Conforme explicado, inicialmente os neurônios terão armazenados na sua memória valores não relacionados ao problema para o qual a rede será treinada para resolver. Assim, para cada amostra de treinamento apresentada à rede, o neurônio irá produzir uma saída, correspondente ao valor armazenado na posição da RAM endereçada pela entrada. Com base no valor esperado para aquela amostra de treinamento, o valor armazenado na referida posição da memória será ajustado.

Em uma RNSP de arquitetura de única camada, os neurônios são ensinados a responder com saída '1' para os padrões do conjunto de treinamento, e apenas para aqueles padrões. Caso mais de duas categorias sejam necessárias, diversas redes RAM devem ser utilizadas em paralelo, cada rede sendo treinada para responder a uma classe do padrão com o nível lógico '1', e para as demais classes com nível lógico '0'.

O treinamento da rede será continuado até que todas as amostras de treinamento sejam apresentadas à rede, e a mesma seja ajustada com base nos valores esperados para cada uma das amostras. O critério de parada do treinamento poderá ser tanto quando a rede atingir um erro abaixo de um valor esperado predeterminado, quanto após um número predeterminado de épocas. Para o caso de uma RNSP convencional, o treinamento da rede consiste apenas em apresentar todas as amostras de treinamento à rede, substituindo as posições de memória acessadas de '0' por '1'. Este treinamento é conhecido por *one shot learning*.

Uma das principais vantagens das RNSPs reside em seu treinamento, o qual é muito mais rápido e flexível em comparação aos algoritmos de treinamento das RNAs convencionais. A velocidade do processo de treinamento nas RNSPs é devido à existência de mútua independência entre os nós quando as entradas da rede são alteradas, o que contrasta com as RNAs com peso (BRAGA; CARVALHO; LUDERMIR, 2000). Nos modelos com peso, o treinamento é mais complexo, uma vez que alterar o valor dos pesos para treinar a rede em vista de um determinado padrão de entrada-saída irá alterar o comportamento do neurônio para outros padrões previamente aprendidos.

### 1.2.3 Fase de teste de uma RNSP

Durante a fase de teste, um padrão desconhecido é apresentado à rede e esta deverá ser capaz de classificar o padrão como pertencendo ou não à classe para a qual a rede foi treinada. Caso a rede possua diversas camadas paralelas para identificar diversas classes, o padrão desconhecido será apresentado a todas as camadas, e espera-se que apenas a camada que foi treinada para reconhecer classe correta identifique o referido padrão desconhecido. Caso o padrão apresentado pertença ao conjunto de treinamento, todos os neurônios da RNSP irão identificar as tuplas do padrão e emitirão saídas de nível lógico '1'. Caso o padrão apresentado seja similar ao padrões do conjunto de treinamento, apenas parte dos neurônios da RNSP irão identificar as tuplas do padrão e emitirão saídas de

nível lógico ‘1’. Dessa forma, para que a RNSP seja capaz de generalizar, a saída de todos os neurônios da rede é somada por um integrador, o qual irá, com base na quantidade de neurônios que emitiu saída ‘1’, determinar se o padrão desconhecido deve ser reconhecido pela rede. Este tipo de RNSP modificada foi utilizada para o desenvolvimento da rede WiSARD (ALEKSANDER; THOMAS; BOWDEN, 1984), uma das aplicações mais conhecidas de RNSP.

## 1.3 A rede WiSARD

Dentre as RNSPs, aquela que possui o maior destaque na literatura é a rede WiSARD (Wilkie, Stonham and Aleksander's Recognition Device) (ALEKSANDER; THOMAS; BOWDEN, 1984). A rede WiSARD é uma rede de reconhecimento de padrões, *feedforward* e de um único estágio de neurônios (BRAGA; CARVALHO; LUDERMIR, 2000). Assim como na concepção inicial de uma RNSP, a rede WiSARD modifica o conteúdo das memórias RAM com o objetivo de acumular aprendizado.

Conforme visto na Seção 1.2.1, um único neurônio RAM não é capaz de generalização, uma vez que ele é capaz apenas de identificar um padrão para o qual ele foi treinado. Entretanto, uma rede composta por vários neurônios RAM seria capaz de generalizar. De modo a ser capaz de generalizar, a rede WiSARD propõe que os neurônios sejam organizados em uma estrutura denominada discriminador.

Um discriminador consiste em uma rede de uma única camada com $M$ neurônios RAM de $N$ entradas cada. Desse modo, um discriminador com $M$ neurônios será capaz de aprender e reconhecer um subconjunto de um padrão com tamanho $N$x$M$. Cada neurônio é conectado a uma parte do padrão de entrada, denominada tupla, aprendendo apenas parte dela durante fase de treinamento. Na fase de teste, a saída de um discriminador para um padrão é um número binário com $M$ bits, o qual corresponde à saída dos $k$ neurônios da rede. Este número de $M$ bits é processado por um dispositivo somador, dando como saída o número dos $M$ neurônios RAM que reconheceram a tupla correspondente àquele padrão de entrada. A Figura 4 ilustra a estrutura básica de uma rede WiSARD.

Na Figura 4, é possível exemplificar o endereçamento dos neurônios na camada de entrada da rede. Observa-se uma imagem 8x8 *pixels*, os quais devem ser *pixels* binários, ou então deverão ser pré-processados antes de endereçar a rede. Cada um dos $M$ neurônios da rede será endereçado por $N$ *pixels* da imagem. Essa palavra de $N$ *pixels* que endereça

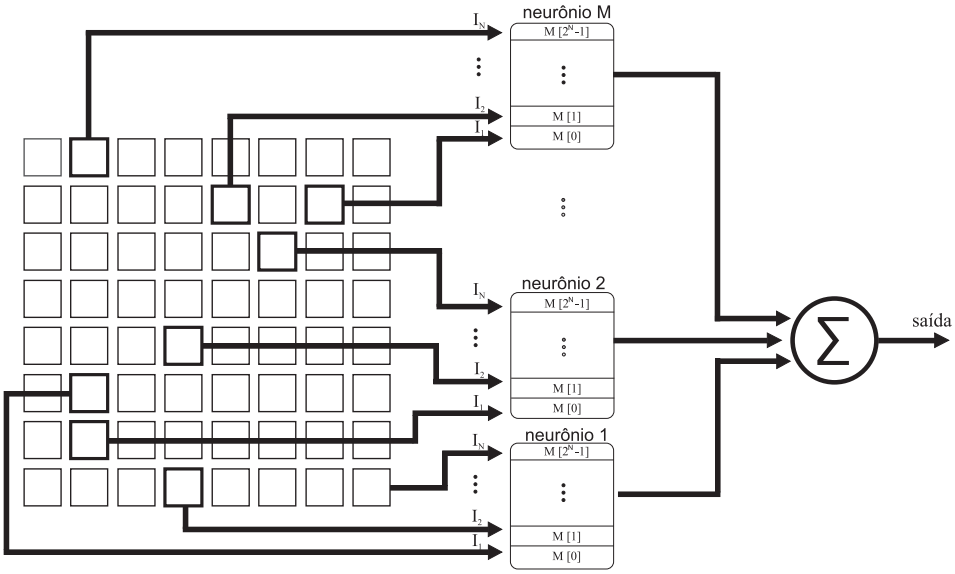



Figura 4: Estrutura da WiSARD

os neurônios é a tupla da rede, a qual deve possuir o mesmo tamanho para todos os neurônios (BRAGA; CARVALHO; LUDERMIR, 2000; ALEKSANDER; MORTON, 1989). Os *pixels* da imagem que formarão as tuplas podem ser escolhidos randomicamente ou de modo que cada neurônios seja responsável por identificar uma parte determinada da imagem. Por exemplo, para uma imagem 8x8 *pixels* e uma tupla de 8 bits, cada neurônios poderia ser responsável por identificar uma linha da matriz 8x8 de *pixels*.

Uma RNSP do tipo WiSARD contendo um único discriminador é capaz de classificar duas classes, isto é, identificar se um padrão pertence à classe para a qual a rede foi treinada (classe 1), ou identificar se o padrão não pertence à classe para a qual a rede foi treinada (classe 2). Para problemas envolvendo mais de duas classes, a rede precisa ser replicada em paralelo em diversas camadas, o que é denominado multi-discriminador (ALEKSANDER; THOMAS; BOWDEN, 1984). Um multi-discriminador consiste em diversos discriminadores combinados em paralelo, sendo cada um treinado para o reconhecimento de uma classe de padrões. Desta forma, uma rede neural WiSARD capaz de reconhecer $C$ classes de padrões, terá em sua arquitetura $C$ discriminadores em paralela.

### 1.3.1 Treinamento da rede WiSARD

De modo semelhante ao treinamento das RNSPs, para o treinamento de uma rede WiSARD deve-se modificar o conteúdo das posições da memória RAM dos neurônios da

rede. Inicialmente o conteúdo das posições de memória é inicializado com nível lógico '0', pois a rede ainda não possui treinamento. Para indicar qual endereço da RAM terá seu conteúdo alterado pelo treinamento, o padrão de entrada é dividido randomicamente em $M$ grupos de $N$ bits, o que é denominado as tuplas da rede. As tuplas da rede são apresentadas como as entradas de cada um dos $M$ neurônios da rede. O conteúdo de cada um dos $M$ neurônios que for endereçado pelos $N$ bits terá o valor modificado para nível lógico '1', uma vez que a rede está sendo treinada para identificar o referido padrão de treinamento. A escolha randômica das tuplas de entrada deve sempre ser a mesma, para todos os discriminadores da WiSARD, assim como para todos os padrões de entrada. Dessa forma é garantida a integridade do treinamento na etapa de reconhecimento.

A rede WiSARD é uma rede neural que possui característica de treinamento supervisionado, ou seja, é necessário informar a qual classe um determinado padrão de treinamento pertence. Para o caso de uma rede WiSARD com múltiplas camadas paralelas (multi-discriminador), apenas a camada responsável por identificar o padrão da amostra de treinamento terá sua memória ajustada. Desse modo, cada amostra de treinamento $k$ possui um valor desejado $d(k)$ que é utilizado para definir de antemão qual camada da rede será modificada para aprender o padrão. O Algoritmo 1 abaixo ilustra o treinamento de uma rede WiSARD:

**Algoritmo 1:** Treinamento da rede WiSARD

```
início
    Inicie a memória de todos os M neurônios com valor lógico '0';
    para i=1: k amostras de treinamento faça
        Divida a amostra k_i em M tuplas de N bits;
        Enderece os M neurônios da camada l com as M tuplas, com base em
        d(k_i);
        Escreva o valor lógico '1' na memória de cada um dos M neurônios
        endereçados pelas tuplas;
    fim
fim
```

A quantidade de neurônios que formam os discriminadores está diretamente relacionada à quantidade de bits que irão endereçar cada neurônio. Em outras palavras, quanto maior a tupla da rede, menos neurônios irão compor os discriminadores. Podemos analisar da seguinte forma a relação entre esses valores: *(i)* utilizando apenas um bit de endereçamento, a quantidade de neurônios por discriminador será igual à quantidade de *pixels* do padrão de entrada, generalizando em excesso o reconhecimento de diferentes

padrões; *(ii)* utilizando a quantidade total de *pixels* para formar o endereço, o discriminador terá apenas um neurônio, impedindo que seja possível reconhecer padrões diferentes daqueles apresentados à rede na fase de treinamento. A relação entre o tamanho da tupla e o número de neurônios no estágio de entrada da rede WiSARD é definida conforme a Equação 2:

$$W = M * N, \tag{2}$$

onde $W$ representa o tamanho em bits de um padrão de entrada, $M$ representa o número de neurônios da rede e $N$ representa o tamanho da tupla para cada neurônio da rede.

### 1.3.2 Fase de teste da rede WiSARD

Quando a fase de treinamento da rede é concluída, espera-se que esta tenha aprendido as características do problema a ser identificado e seja capaz de generalizá-lo a padrões não conhecidos pela rede. Durante a etapa de reconhecimento ou fase de teste da rede WiSARD, um padrão desconhecido é apresentado à rede, mantendo a mesma subdivisão do padrão em tuplas utilizada durante o treinamento da rede. Desse modo, cada tupla do padrão desconhecido irá endereçar os $k$ neurônios da rede, e cada um irá apresentar uma saída com base no valor da memória RAM endereçada. Caso o neurônio produza uma saída com nível lógico '1' para determinada tupla, isso significa que o referido neurônio identificou a tupla daquele padrão como pertencente à classe que a rede foi treinada. Caso a saída possua nível lógico '0', então o neurônio não identificou a tupla daquele padrão como pertencente à classe.

Para saber se a rede como um todo identifica o padrão de entrada, o discriminador irá contabilizar, através de um somador, quantos dos $k$ neurônios apresentaram saída com nível lógico '1', ou seja, identificaram o padrão como pertencente à classe. O esquemático do somador utilizado pode ser observado na Figura 4. Caso o número de neurônios com saídas com nível lógico '1' seja maior que um limiar predeterminado, entende-se que o padrão desconhecido pertence à classe identificada pela rede. Caso o número de neurônios com saídas com nível '1' seja inferior ao limiar predeterminado, entende-se que o padrão desconhecido não pertence à classe identificada pela rede.

Para o caso de uma rede WiSARD com múltiplas camadas paralelas, um padrão desconhecido será apresentado igualmente a todas as camadas da rede. O discriminador de cada camada irá contabilizar a quantidade de neurônios que apresentou saída com

nível lógico ‘1’ para as tuplas do padrão a ser reconhecido. Neste caso, as respostas de cada um dos discriminadores é analisada e comparada, de modo a escolher aquela com o maior número de neurônios de saída com nível lógico ‘1’. A classe do padrão desconhecido será determinada como pertencente à classe do discriminador com o maior número de neurônios com saída ‘1’. Tal configuração é denominada multi-discriminador (ALEKSANDER; MORTON, 1989; ALEKSANDER, 1983).

## 1.4 RNSP com nó lógico probabilístico

A introdução de um elemento probabilístico no neurônio RAM foi proposta por Aleksander (ALEKSANDER; MORTON, 1989), e torna o comportamento da RNSP estocástico, ao invés de um comportamento determinístico como na rede WiSARD. Ao contrário do neurônio ou nó RAM convencional que possuía armazenado em cada posição da memória RAM um único bit representando os valores de ‘0’ ou ‘1’, uma RNSP com nó lógico probabilístico (*Probabilistic Logic Node* - PLN) consiste em um nó RAM que possui mais de um bit armazenado em cada posição da RAM. Esta valor, conhecido como palavra da memória, representa a probabilidade de o nó apresentar uma saída de nível lógico ‘1’ para o padrão de entrada que endereçou a memória.

Para o neurônio probabilístico convencional, o PLN, a palavra armazenada em cada posição de memória possui 2 bits, os quais representam três possibilidades de informação armazenadas na palavra da memória (LUDERMIR, 1994):

- o valor ‘0’, o qual significa 0% de chance do neurônio apresentar saída de nível lógico ‘1’, ou seja, representa a saída do neurônio com nível lógico ‘0’;
- o valor ‘1’, o qual representa 100% de chance do neurônio apresentar saída de nível lógico ‘1’, ou seja, representa a saída do neurônio com nível lógico ‘1’; e
- o valor $u$, ou indeterminado, o qual representa a chance de 50% do neurônio apresentar saída de nível lógico ‘1’, logo o valor denotado por $u$ representa semanticamente um valor indefinido ou desconhecido.

As redes de nós PLN são iniciadas com o valor $u$ em todos os nós. A ideia central do modelo PLN é fortalecer a diferenciação entre os padrões vistos durante a fase de treinamento e os padrões não vistos (BRAGA; CARVALHO; LUDERMIR, 2000). Para o caso da rede WiSARD,

a rede não sabe diferenciar entre uma saída desejada da rede com nível lógico '0' e um endereço não acessado da rede durante a fase de treinamento. Assim, a função de saída $s$, em vez de ser determinística, como no caso das RNSP anteriores, torna-se probabilística (aleatória), podendo ser definida na Equação 3:

$$s = \begin{cases} 0 & \text{se } M[I] = \text{'0'} \\ 1 & \text{se } M[I] = \text{'1'} \\ \text{aleatório} & \text{se } M[I] = \text{'u'}, \end{cases} \tag{3}$$

onde $M[p]$ é o conteúdo da memória endereçada pelo padrão $I$, e aleatório é a função aleatória que fornece 0's e 1's com a mesma probabilidade.

O valor indefinido $u$ em uma célula é usado para identificar uma posição da memória não utilizada durante a fase de treinamento. Assim, se uma entrada que não foi vista durante a fase de treinamento aparece na fase de teste, o nó produz uma saída aleatória que pode possui nível lógico '0' ou '1'. A presença de um elemento indefinido e probabilístico na rede aumenta sua capacidade de generalização e tolerância a ruídos (ALEKSANDER; MORTON, 1989), uma vez que a cada treinamento da rede, seus valores de memória irão ser ajustados de maneira diferente, ao contrário da da rede WiSARD, que apresenta caráter puramente determinístico.

### 1.4.1 Estrutura de uma RNSP PLN

Ao contrário das RNSPs simples, tal como a WiSARD, as quais utilizam uma única camada de neurônios cujas saídas são somadas por um discriminador, uma rede PLN possui mais de uma camada de neurônios, também chamadas de estágios, formando uma arquitetura piramidal. Na arquitetura piramidal, as saídas dos neurônios de uma camada irão endereçar os neurônios da camada subsequente, até que haja apenas um neurônio na chamada camada de saída. A saída do neurônio da camada de saída representa a saída da rede. A Figura 5 ilustra a estrutura básica de uma rede PLN.

Como pode ser observado no exemplo da Figura 5, as tuplas são formadas aleatoriamente a partir de uma imagem. Cada tupla de imagem irá endereçar um dos neurônios do primeiro estágio da RNSP, ou estágio de entrada. O valor da tupla será responsável por acessar uma posição da memória de cada neurônio, a qual armazena a informação referente à saída do neurônio, a saber, nível lógico '1', '0' ou 'u'. Cada um dos neurônios do primeiro estágio irá produzir uma saída com base na posição de memória acessada por sua respectiva tupla, e a saída de cada um dos neurônios do primeiro estágio será

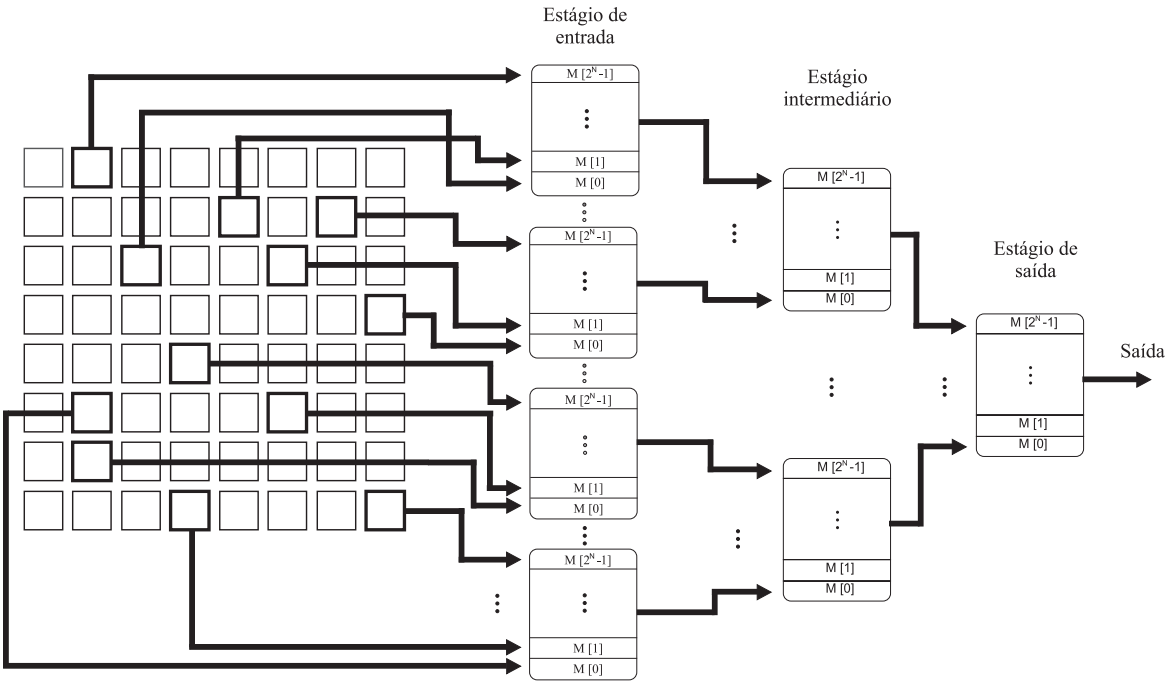



Figura 5: Estrutura de uma PLN

utilizada para endereçar o neurônio do segundo estágio. Assim, as saídas dos neurônios de um estágio serão utilizadas para formar as tuplas do estágio subsequente da rede.

É importante notar que as RNPS podem possuir diversos tamanhos de tupla e números de estágios. O último estágio de uma rede PLN deve apresentar apenas um neurônio, o qual é responsável por produzir a saída da rede. Como a saída dos neurônios da rede PLN é composta por apenas um neurônio, as saídas possíveis da rede consiste em um único bit podendo apresentar nível lógico '0' ou '1'. Assim, de maneira similar à rede WiSARD, uma única rede PLN é capaz de classificar apenas duas classes.

Para que a rede PLN possa identificar mais que duas classes, deve-se configurar múltiplas pirâmides em paralelo para formar uma rede com um dado número de saídas. Neste caso, cada camada paralela é configurada para identificar uma das classes, apresentando valor lógico '1' para os padrões pertences a esta classe, e a não identificar as demais, apresentando valor lógico '0' (LUDERMIR; OLIVEIRA, 1994). Tal arquitetura de pirâmides paralelas é similar ao multi-discriminador da rede WiSARD (AUSTIN, 1994).

### 1.4.2 Treinamento de uma RNSP PLN

O treinamento de uma RNSP do tipo PLN consiste em substituir os valores 'u' inicialmente armazenados na memória RAM *(look-up table)* pelos valores correspondentes à saída desejada para o neurônio (LUDERMIR, 1994). A diferença entre o treinamento da rede WiSARD e da rede PLN reside no fato que a RNSP PLN é inicializada com o valor indefinido $u$, o qual será substituído pelas saídas desejadas para a rede durante o treinamento. Além disso, devido à arquitetura piramidal da rede e seu caráter probabilístico, torna-se necessário verificar se os ajustes realizados pela rede de fato proporcionam a saída desejada para um determinado padrão de treinamento.

Penny (PENNY; STONHAM, 1990) discorre sobre os algoritmos de treinamento para as RNSP do tipo PLN, dentre os quais são apresentados o algoritmo de *reward-penalty* (punição-recompensa) de Aleksander (ALEKSANDER; MORTON, 1989) e o algoritmo de retro-propagação de erro de Al-Alawi (AL-ALAWI; STONHAM, 1989). O algoritmo de treinamento uma rede PLN de arquitetura piramidal escolhido para o presente trabalho é o proposto por Aleksander (ALEKSANDER; MORTON, 1989), e é dado pelas seguintes etapas:

O parâmetro $\beta$ define quantas vezes uma amostra de treinamento será reapresentada à rede antes que seja determinado um conflito de aprendizagem (ALEKSANDER; MORTON, 1989; PENNY; STONHAM, 1990). Conforme mostrado no Algoritmo 2, caso ocorra um conflito de aprendizagem, todos os neurônios da rede terão o conteúdo da memória endereçado pelo padrão de entrada alterado para 'u'. Este procedimento inclui os neurônios dos estágios intermediário e de saída que são endereçados pelos neurônios dos estágios anteriores.

Uma vantagem do modelo PLN é que devido ao valor lógico adicional $u$, as redes PLN são capazes de distinguir entre posições de memória treinadas e não-treinadas. Isto tem efeito direto nos algoritmos de treinamento, os quais poderão trabalhar apenas com as unidades não-treinadas, implicando em mais simplicidade e mais rapidez (BRAGA; CARVALHO; LUDERMIR, 2000). Além disso, usualmente existem várias fontes de ruído na atividade de reconhecimento de padrões em redes neurais. Com a introdução da formulação estocástica da rede PLN, acredita-se que se obteve um modelo mais realista desta atividade (BRAGA; CARVALHO; LUDERMIR, 2000).

Outra vantagem do modelo de RNSP PLN é que, apesar de possuir uma arquitetura com múltiplos estágios (arquitetura em pirâmide), não há necessidade de se calcular um

**Algoritmo 2:** Treinamento da rede PLN

**início**
  Inicie a memória de todos os $M$ neurônios com valor lógico '$u$';
  **para** *i=1: k amostras de treinamento* **faça**
    Divida a amostra $k_i$ em $M$ tuplas de $N$ bits;
    Enderece os $M$ neurônios de entrada com as $M$ tuplas;
    **para** *j=1:* $(e-1)$ *estágios da rede* **faça**
      Calcule a saída $s_{i,j}$ de cada neurônio;
      Forme tuplas com as saídas $s_{i,j}$ para endereçar a camada seguinte;
    **fim**
    Calcule a saída $s_{i,e}$ da camada de saída;
    Calcule o erro $E_k = abs(d(k_i) - s_{i,e})$;
    **se** $E_k = 0$ **então**
      **para** *z=1: M neurônios da rede* **faça**
        $M[I] = s_{i,j}$;
      **fim**
    **fim**
    **senão**
      **repita**
        **para** *j=1:* $e-1$ *estágios da rede* **faça**
          Calcule a saída $s_{i,j}$ de cada neurônio;
          Forme tuplas com as saídas $s_{i,j}$ para a camada seguinte;
        **fim**
        Calcule a saída $s_{i,e}$ da camada de saída;
        Calcule o erro $E_k = abs(d(k_i) - s_{i,e}$;
        $c = c + 1$;
      **até** $E_k = 0$ *ou* $c = \beta$;
      **se** $E_k = 0$ **então**
        **para** *z=1: M neurônios da rede* **faça**
          $M[I] = s_{i,j}$;
        **fim**
      **fim**
      **se** $c = \beta$ **então**
        **para** *z=1: M neurônios da rede* **faça**
          $M[I] = u$;
        **fim**
      **fim**
    **fim**
  **fim**
**fim**

sinal de erro específico para cada um dos neurônios da rede, e também não há necessidade de se calcular o erro para cada neurônio. O ajuste da rede é realizado a partir de um único sinal de erro global, onde todos os neurônios são ajustados com base neste sinal de erro (MYERS, 1989).

As funções de saída de uma RNSP do tipo PLN são descritas em Myers (MYERS, 1989), onde é explicado que as funções mais utilizadas são a função probabilística e sigmoide. A função probabilística é denominada uma função com limitador forte (*hard limiter*), uma vez não possui valores possíveis entre as saídas de nível lógico '0' e '1'. Por outro lado, a função sigmoide é denominada como uma função com limitador fraco (*soft limiter*), uma vez possui infinitos valores possíveis entre as saídas de nível lógico '0' e '1'. Myers conclui que as funções com limitadores forte convergem mais rápido que as de limitadores fracos. Entretanto, a escolha da função é muito dependente do tamanho, arquitetura e complexidade do problema a ser solucionado.

## 1.5 RNSP de nós lógicos probabilísticos de múltiplos valores

A RNSP de nós lógicos probabilísticos de múltiplos valores (*Multi-value Probabilistic Logic Node* - MPLN) é aquela que apresenta posições da memória RAM com uma palavra contendo três ou mais bits para representar a probabilidade do nó produzir uma saída de valor lógico '1' em resposta a uma entrada. Assim como na rede PLN, a rede MPLN apresenta uma arquitetura em pirâmide, de modo que as saídas de um estágio de neurônios sejam utilizadas para formar o novo conjunto de tuplas que irão endereçar os neurônios dos estágio subsequentes, até que seja alcançado o estágio de saída. A saída do neurônio do estágio de saída representa a saída da rede. Assim, uma rede MPLN pode ser representada pela Figura 5. A diferença entre as topologia reside no conteúdo das posições de memória dos neurônios $M[I]$.

A diferença básica entre as redes PLN e MPLN é que, em vez de armazenar apenas as possibilidades '0', '1', '$u$', cada nó da RNSP MPLN armazena uma palavra capaz de representar $2^{\omega}$ valores de probabilidade de o nó apresentar saída '1' quando a célula é endereçada, onde $\omega$ é o número de bits da palavra da RAM. Desse modo, pode-se entender que a rede MPLN é uma generalização da rede PLN para o caso de uma palavra de $\omega$ bits (NEDJAH et al., 2016). Por exemplo, para o caso de $\omega = 4$, cada posição de memória poderia armazenar 16 valores de probabilidade de a rede emitir uma saída de nível lógico '1' para um padrão endereçando a referida posição da memória. Para que haja uma consistência na probabilidade da rede, estes valores de probabilidade devem representar probabilidades com um mesmo passo incremental entre as mesmas, partindo

do valor 0% até o 100%. A Tabela 1 ilustra um exemplo de associação das posições de memória com a probabilidade do neurônios emitir uma saída de nível lógico '1' para o caso onde $\omega = 3$.

Tabela 1: Exemplo de associação das posições de memória com a probabilidade do neurônios emitir uma saída de nível lógico '1'

| **Valor binário** | **Probabilidade** |
|---|---|
| 000 | 0 |
| 001 | 0,167 |
| 010 | 0,333 |
| 011 | 0,5 |
| 100 | 0,667 |
| 101 | 0,833 |
| 110 | 1 |
| 111 | não utilizada |

Caso o número total de valores probabilidades permitidas $2^{\omega}$ não seja capaz de fornecer probabilidades igualmente espaçadas entre 0% e 100%, alguns valores de possibilidade podem ser configurados para serem omitidos, não sendo acessados durante o treinamento e teste da rede. É importante notar que a configuração de valores de probabilidade deve necessariamente possuir um valor correspondente à probabilidade de 0,5, análogo ao valor 'u' da rede PLN, o qual indica uma saída completamente aleatória da rede para um padrão desconhecido. Assim como na rede PLN, o treinamento da rede MPLN é inicializado com todos os valores de memória setados em 0,5.

O algoritmo de treinamento aplicado para a rede MPLN é similar ao da rede PLN. Entretanto, o treinamento da rede MPLN apresenta um caráter iterativo. Os valores armazenados são inicialmente iguais a 0,5, o que representa iguais chances de o neurônio produzir saída '1' ou '0'. Durante o treinamento, o valor em uma posição da RAM é incrementado ou decrementado, podendo chegar a '0' (limite inferior) ou a '1' (limite superior). O valor do conteúdo da palavra é incrementado ou decrementado com base no erro entre a saída da rede e o valor esperado para as amostras de treinamento. Em (ALEKSANDER; MORTON, 1989; BRAGA; CARVALHO; LUDERMIR, 2000), é descrita uma forma de generalizar o algoritmo de treinamento da rede PLN para a rede MPLN. Analogamente, em (PENNY; STONHAM, 1990) é descrito uma forma de generalizar o algoritmo de punição-recompensa de Aleksander (ALEKSANDER; MORTON, 1989) para as redes MPLN. Assim como na rede PLN, o algoritmo de treinamento para uma rede MPLN de arquitetura

piramidal escolhido para o presente trabalho é o proposto por Aleksander (ALEKSANDER; MORTON, 1989; NEDJAH et al., 2016), e é explicado em detalhes no Algoritmo 3 abaixo.

**Algoritmo 3:** Treinamento da rede MPLN

**início**
  Inicie a memória de todos os $M$ neurônios com valor $0,5$;
  **para** *i=1: k amostras de treinamento* **faça**
    Divida a amostra $k_i$ em $M$ tuplas de $N$ bits;
    Enderece os $M$ neurônios de entrada com as $M$ tuplas;
    **repita**
      **para** *j=1:* $(e-1)$ *estágios da rede* **faça**
        Enderece os $M$ neurônios de entrada com as $M$ tuplas;
        Calcule a saída $s_{i,j,c}$ de cada neurônio;
        Forme tuplas com as saídas $s_{i,j,c}$ para a camada seguinte;
      **fim**
      Calcule a saída $s_{i,e,c}$ da camada de saída;
      Calcule o erro $E_{k,c} = abs(d(k_i) - s_{i,e,c})$;
      **se** $E_{k,c} = 0$ **então**
        $acerto = acerto + 1$
      **fim**
    **até** $c = \beta$;
    **se** $acerto = c$ **então**
      nada a fazer, a rede aprendeu o padrão;
    **fim**
    **se** $0 < acerto < c$ **então**
      **para** *z=1: M neurônios da rede* **faça**
        $M[I]_z = M[I]_z + ajuste$ para $c$ onde $E_{k,c} = 0$ (**reward**);
      **fim**
    **fim**
    **se** $acerto = 0$ **então**
      **para** *z=1: M neurônios da rede* **faça**
        $M[I]_z = M[I]_z + ajuste$ (**penalty**);
      **fim**
    **fim**
  **fim**
**fim**

A principal diferença entre o treinamento da rede PLN e da MPLN é que, na primeira, o treinamento consiste apenas em substituir os valores da memória de $u$ para '0' ou '1', ou, em caso de inconsistência do treinamento, reverter este valor para $u$. Por outro lado, o treinamento da rede MPLN consiste em incrementar e decrementar o valor armazenado na posição de memória cada vez que ela é acessada. Logo, não há uma substituição direta, mas sim um processo iterativo com o objetivo de alcançar uma probabilidade de 0% ou 100%. Para obter o novo valor a ser armazenado na palavra de cada

um dos neurônios é utilizada a função descrita pela Equação 4 (LUDERMIR; OLIVEIRA, 1994):

$$M\,[I] = \begin{cases} M[I] + \eta * g(s) & \text{se } \textit{reward} \\ M[I] - \eta * g(s) & \text{se } \textit{penalty}, \end{cases} \tag{4}$$

onde $M[I]$ é o conteúdo atual da célula associada ao padrão $I$; $\eta$ é a taxa de aprendizagem; e $g(s)$ é uma função da saída do neurônio definida pela equação 5, com $s$ sendo a resposta do neurônio para uma determinada amostra de treinamento:

- Caso a rede tenha acertado a resposta para o padrão de treinamento (erro nulo):

$$g\,[s] = \begin{cases} +1 & \text{se } \; s = 1 \\ \\ -1 & \text{se } \; s = 0. \end{cases} \tag{5}$$

- Caso a rede tenha errado a resposta para o padrão de treinamento (erro não-nulo):

$$g\,[s] = \begin{cases} -1 & \textit{se } s = 1 \\ \\ +1 & \textit{se } s = 0. \end{cases} \tag{6}$$

Tal processo de treinamento é conhecido como algoritmo de punição-recompensa, uma vez que a rede é iterativamente “punida”ao errar o valor da saída esperada e “recompensada”ao acertar o valor da saída esperada. Dessa forma, durante o treinamento da rede, todos as posições de memória de todos os neurônios acessados serão ajustadas com base no erro entre a saída final da rede e a saída desejada para a amostrada de treinamento, e também com base na saída do neurônio a ser ajustado. O conteúdo da memória será modificado com o objetivo de encorajar ou desencorajar o neurônio de produzir a mesma saída na próxima vez que esta posição for acessada por uma tupla do padrão de treinamento.

A vantagem da rede MPLN é que, uma vez que é possível armazenar diversos valores de probabilidade na memória, após diversas etapas de modificação no conteúdo da rede em uma direção (por exemplo, incentivando a rede a emitir saída com nível lógico ‘1’), é consideravelmente difícil eliminar o conhecimento adquirido pela rede. Na verdade, será necessário o mesmo número de passos no sentido contrário para retornar o valor de probabilidade armazenado na rede como ‘u’ novamente. Assim, a rede MPLN fornece uma proteção contra erros nas amostras de treinamento, uma vez que uma única amostra errada não seria suficiente para retornar o valor de todos os neurônios acessados para o valor ‘u’ (MYERS, 1989). Adicionalmente, Myers (MYERS, 1989) afirma que as

redes MPLN possuem a capacidade de resolver problemas de aprendizado difíceis (*hard learning problems*), tal como o problema de identificação do bit de paridade, o qual será abordado no Capítulo 4 desta dissertação.

## 1.6 Considerações finais do capítulo

Neste capítulo foram apresentados os conceitos gerais sobre Redes Neurais Artificiais e a sua variante objeto do presente trabalho, as RNSP. Dentre as RNSPs, foram introduzidos os conceitos básicos deste tipo de rede e suas diferenças para o modelo de RNA convencional. Em seguida, foram abordadas as principais arquiteturas RNSP, descrevendo suas vantagens, particularidades, assim como seus algoritmos de treinamento.

# Capítulo 2

# TRABALHOS RELACIONADOS

ESTE capítulo apresenta alguns trabalhos relacionados à metodologia de projeto e treinamento de RNSPs e ao problema do reconhecimento de imagens em geral. Nota-se que existem as mais diversas abordagens de RNSPs para a classificação de imagens, uma vez que este é um problema clássico de sua aplicação. As RNSPs são interessantes na classificação de imagens devido às suas características de aprendizado e generalização. Sendo assim, o trabalho proposto é inspirado nestas metodologias e, com elas, é possível realizar uma análise comparativa mais coerente.

As RNSPs baseadas no modelo WiSARD são aquelas que apresentam neurônios que consistem em uma memória capaz de armazenar um bit em cada posições de memória do neurônio. Estes bits são utilizados para determinar a saída do neurônio quando uma tupla de um padrão de entrada acessa a referida posição de memória do neurônio. Dessa forma, pode-se concluir que RNSPs baseadas no modelo WiSARD armazenam uma saída binária para cada conjunto de entrada. O conjunto de entrada de um neurônio RAM é um conjunto de bits, ou tupla, o qual é utilizado para endereçar a RAM e acessar o conteúdo da memória. O conteúdo da memória é modificado durante o treinamento da rede e utilizado para exibir a resposta do neurônio a um novo padrão durante a fase de teste da rede.

Conforme abordado, o apogeu das RNSPs baseadas em RAM foi o desenvolvimento da rede WiSARD e, até hoje, esta é alvo de estudos em que novas redes são desenvolvidas com base nesta arquitetura. Assim, pode-se dividir os trabalhos relevantes como aqueles baseados no modelo WiSARD, as quais são abordadas na Seção 2.1 e aqueles baseados em outras arquiteturas de neurônio RAM, abordadas na Seção 2.2.

## 2.1 RNSPs baseadas no modelo WiSARD

As RNSPs baseadas no modelo WiSARD são aquelas que operam com um conjunto de neurônios RAM, os quais ão endereçados por um padrão de entrada dividido em tuplas, e as saídas dos neurônios são conectadas em um conjunto de discriminadores, conforme visto na Seção 1.3. Cada discriminador é responsável pela identificação de uma classe do problema a ser resolvido pela rede. Usualmente, as redes WiSARD possuem apenas um estágio de neurônios RAM. Entretanto, como será visto, alguns autores modificaram a rede WiSARD para uma arquitetura em pirâmide, com o intuito de identificar propriedades mais complexas do padrão de treinamento.

### 2.1.1 RNSPs baseadas no modelo WiSARD de única camada

Em (JORGENSEN; CHRISTENSEN; ANDERSEN, 1996), é apresentado um sistema de processamento de imagem para a localização automática de etiquetas de perigo na parte traseira de contêineres. O sistema utiliza redes neurais baseadas no neurônio RAM, especialmente na rede WiSARD, para localizar e classificar etiquetas após uma etapa de pré-processamento envolvendo filtros de borda não-lineares projetados e a conversão de RGB (*Red, Green, Blue*) para HSV (*Hue, Saturation, Value*).

Como as etiquetas de perigo podem estar localizadas em posições diferentes da parte traseira do contêiner, é proposto dividir a tarefa de inspeção em dois passos. O primeiro consiste na localização de possíveis objetos de etiqueta com base na forma do objeto, e o segundo consiste em utilizar as informações de cor e intensidade destes objetos para executar uma classificação final. Após localizada a imagem, esta é filtrada com filtros de borda para que as saídas destes filtros seja adequada para serem inseridas na entrada da rede WiSARD e assim classificar as etiquetas.

Em (GREGORIO; GIORDANO, 2013), é apresentado um sistema baseado na rede WiSARD, denominado CwisarD, para resolver o problema da detecção de mudanças (CD - *Change Detection*) em várias imagens da mesma cena tiradas em diferentes momentos, e, em particular, o movimento em vídeos da mesma posição assumida por uma câmera estática. Na arquitetura da CwisarD, de modo a alimentar os discriminadores com as entradas corretas, a rede cria um discriminador para cada pixel da imagem. As cores RGB do pixel são representadas por uma imagem binária (preto e branco), onde as colunas representam o canal de cor (R, G e B) e as linhas os valor dos canais de cor.

A rede CwisarD precisa ser treinada com um certo número de quadros nos quais nenhum objeto estranho ou externo esteja possivelmente presente. Após esta fase, a rede começa a classificar cada pixel como pertencendo ao ambiente ou como pertencendo a um elemento externo. No primeiro caso, isto significa que a saída do discriminador é superior a um limite $\sigma$. O sistema utiliza os pixels classificados corretamente para adicionalmente treinar os discriminadores associados. As características a serem treinadas *on-line* são uma peculiaridade dos sistemas sem peso. Desse modo, a rede Cwisard é capaz de se adaptar tanto para fundos dinâmicos quanto para mudanças graduais na iluminação.

Em (GREGORIO; GIORDANO, 2014), é apresentada uma metodologia de RNSP com base em pixels para enfrentar o problema de detecção de mudanças no campo de visão de uma câmera, conhecida como a rede CwisardDH. A rede CwisarDH supera uma deficiência da rede CwisardD, onde uma pequena modificação no ambiente era identificada como um objeto estranho, mesmo no caso onde este objeto passava a fazer parte do ambiente. A rede CwisarDH propõe um *buffer* para armazenar um número consecutivo de quadros de uma imagem. Quando o *buffer* está cheio, as informações armazenadas no mesmo são utilizadas para treinar a rede *online*. Assim, um objeto que passa a integrar a imagem pode vir a ser considerado como parte da imagem, caso este seja mantido imóvel durante todos os pixeis do *buffer*. Isto melhora o desempenho da rede no caso de objetos intermitentes, os quais passam a fazer parte ou deixam de fazer parte da imagem para a qual a rede foi treinada. As principais características da rede CwisarDH são: 1) a capacidade de adaptação dinâmica às mudança de fundo devido ao modelo de RNSP adotado, e 2) a introdução de históricos de cores de pixel para melhorar o comportamento do sistema em vídeos onde ocorre (des)aparecimento de objetos na cena de vídeo e/ou mudanças bruscas de iluminação, forma e brilho do fundo.

Nota-se que as aplicações relacionadas ao reconhecimento de imagens geralmente requerem uma boa técnica de pré-processamento das imagens, uma vez que a informação de uma imagem precisa ser convertida em um formato binário antes de ser alimentada a uma RNSP. Além disso, muitas vezes o processamento da imagem pode auxiliar a RNSP a obter um melhor treinamento e melhor precisão durante a fase de teste. Em (MOREIRA et al., 2013), é proposto um método de rastreio de alvos de superfície submarina em vídeo utilizando uma RNSP baseada no modelo WiSARD. O autor chama a atenção para o fato de que o rastreamento de objetos em vídeo é uma tarefa importante e desafiadora em

diversas aplicações, e ainda que dificuldades podem surgir devido a condições climática, trajetória e aparência alvo, oclusões, condições de iluminação e ruído. O rastreamento é uma aplicação de alto nível e requer a localização do objeto quadro por quadro em tempo real. Em cada quadro, um rastreador baseado em detecção por segmentação executa três etapas principais: detecção, rastreio e análise das características do objeto. Estas etapas dependem da qualidade da segmentação e o rastreio executado pela RNSP WiSARD depende da qualidade da binarização da imagem. O trabalho propõe ainda uma binarização rápida híbrida (detecção de bordas) em modelo de cor *YcbCr* e modos para configurar uma rede neural WiSARD para melhorar a eficiência quando o erro de binarização ocorrer.

Em (GARCIA, 2003), é apresentado um estudo sobre o uso de métodos de otimização global, tais como algoritmos genéticos, *simulated annealing* e *tabu search*, aplicados para auxiliar a escolha do tamanho da tupla nas RNSPs baseadas em multi-discriminadores. O tamanho da tupla, definido pelo autor como conectividade da RNSP, consiste no número de bits de uma imagem que endereçam cada neurônio. Uma baixa conectividade pode significar que nem todos os bits de uma imagem são endereçados a um neurônio da rede. Por outro lado, uma alta conectividade pode significar que um bit da imagem pode ser endereçado a mais de um neurônio da rede. O trabalho investiga como tais métodos influenciam no desempenho dos multi-discriminadores em tarefas complexas, como o reconhecimento de caracteres numéricos manuscritos. Segundo o autor, é preciso decidir o tamanho da tupla ideal para o problema, sendo que esta não deve ser muito grande, pois, além de aumentar exponencialmente a memória, o neurônio acaba se especializando apenas nas características aprendidas no conjunto de treinamento. Todos os algoritmos são testados em uma RNSP WiSARD utilizada para reconhecer caracteres numéricos manuscritos. No geral, o trabalho conclui que métodos de otimização não são eficientes para a escolha do padrão de conectividade de RNSPs, pois os métodos causaram excessivo *overfiting* na rede, adaptando da melhor forma possível a conectividade para o padrão de treinamento.

### 2.1.2 RNSPs baseadas no modelo WiSARD de múltipla camada

Apesar de a RNSP WiSARD possuir originalmente apenas uma camada de neurônios, alguns autores expandiram o modelo para topologias de múltiplas camadas, com o objetivo de identificar características intrínsecas e complexas da imagem. Em (BANDEIRA, 2010), é

proposta uma arquitetura de uma RNSP utilizando a hierarquia multi-camadas baseada no modelo WiSARD e o *Neocognitron*, a rede sendo denominada como NC-WiSARD. Segundo o autor, a rede foi desenvolvida utilizando as principais características do *Neocognitron* (FUKUSHIMA, 2003), da rede WiSARD e da rede AUTOWiSARD (WICKERT; FRANÇA, 2001).

Por meio da hierarquia multi-camadas, a rede neural NC-WiSARD é capaz de identificar simples características de padrões nos estágios iniciais e, através da concatenação destes, a rede adquire capacidade cognitiva para responder a classes complexas. Proveniente da estrutura da arquitetura da rede, não são necessários ajustes das imagens de entrada (pré-processamento) para que a rede consiga responder corretamente a padrões redimensionados e/ou reposicionados. O referido trabalho aborda a aplicação do reconhecimento de caracteres da base de dados *MNIST*, logo sua comparação com o presente trabalho se torna interessante.

## 2.2 RNSPs baseadas em outras arquiteturas de neurônios RAM

Após o sucesso da topologia de RNSP WiSARD, o tema de RNSP conseguiu relevante atenção dos pesquisadores e diversos outros modelos buscaram melhorar seu desempenho. A partir do modelo de um neurônio RAM, autores desenvolveram novas redes capazes de, por exemplo, armazenar mais de um bit no endereço de memória, e ainda apresentar uma saída do neurônio probabilística com base em um valor armazenado na palavra do neurônio RAM.

A grande diferença entre a rede WiSARD e suas variações é que nas últimas busca-se dar um tratamento estocástico ao neurônio. Para as RNSPs com neurônio estocástico, seus neurônios possuem processamento e treinamento baseados em função de probabilidade. Em geral, estas redes possuem um neurônio com posições de memória de mais um bit. O conteúdo de uma posição da memória exprime uma probabilidade do neurônio apresentar uma saída de valor lógico '1' para uma determinada entrada que endereça o neurônio.

Em (HOWELLS; FAIRHURST; BISSET, 1995), é descrita uma RNSP de memória de acesso aleatório probabilística (pRAM) para a classificação de objetos em uma sequência de vídeo de imagens FLIR (*forward looking infra-red*) em duas classes: alvo e desordem.

Uma RNSP do tipo pRAM é semelhante ao modelo MPLN, no qual um valor de probabilidade é armazenado nas posições da memória. A diferença é que na RNSP pRAM probabilidades contínuas podem ser armazenadas na memória do neurônio. É utilizado um sensor de calor que usualmente detecta o motor dos veículos, o que torna possível extração de uma característica relativamente precisa do objeto a ser detectado, mesmo em uma resolução baixa da imagem. A extração de características é realizada nos *patches* da imagem em torno dos pontos de calor, sendo então pre-processadas e utilizadas para treinar a rede. Estes recursos serviram como entrada para uma RNSP pRAM com aprendizagem por reforço treinada para produzir valores de (1 0) para alvos e (0 1) para a desordem. Novamente, observa-se a importância do pré-processamento de imagens para que estas possam ser utilizadas para endereçar uma RNSP.

Em (CARVALHO; FAIRHURST; BISSET, 1994), é apresentada uma nova arquitetura para RNSP que, embora retenha as propriedades de aprendizagem e generalização possuídas por arquiteturas de rede já existentes, permite padrões alvo arbitrários para as classes de padrão com forte propriedade de convergência. A arquitetura de rede fornece a base para um sistema de reconhecimento de padrões capazes de aplicação em um ambiente prático. A rede apresentada é denominada Rede de Convergência Booleana (BCN - *Boolean Convergency Network*), sendo similar à rede RNSP PLN, onde os valores '0', '1' e 'u' podem ser armazenados nas posições de memória. A diferença entre a rede PLN e a BCN é que, na RNSP BCN, o próprio neurônio pode produzir uma saída de valor 'u' ou receber uma entrada de valor 'u', ou seja é dado um tratamento determinístico do valor indefinido 'u'. Durante o treinamento, caso uma posição da memória deva ser substituída por um valor oposto, ela é substituída por 'u' ao invés. Caso um endereço possua 'u's, todas as combinações possíveis são testadas. Caso as posições acessadas contenham apenas 'u' ou um valor definido, a saída do neurônio será o valor definido. Caso as posições acessadas contenham apenas ambos os valores definidos '0' e '1', a saída do neurônio será 'u'.

A rede BCN compreende três grupos sequenciais de neurônios rotulados *Pre*, *Main* e *Post*, e estes possuem neurônios com valores de conectividade (tupla) distintos. Os valores de conectividade devem ser escolhidos como um compromisso entre os neurônios de conectividade baixa, que proporcionam um número excessivo de *matches*, fornecendo informações potencialmente contraditórias, e neurônios de alta conectividade, em que a probabilidade de quaisquer *matches* é inaceitavelmente baixa.

Em (RAMANAN et al., 1995), é apresentada uma abordagem integrada para a classificação de padrões, onde uma arquitetura de RNSP booleana auto-organizável (treinamento não-supervisionado) é utilizada como um processador *front-end* para uma arquitetura neural *feedforward* baseada em princípios de busca de objetivo, denominada RNSP GSN(*Goal Seeking Neuron*). O desempenho da arquitetura integrada é ilustrado através de um exemplo de aplicação em um problema de reconhecimento de caracteres. Similar à rede BCN, a rede GSN consiste de uma rede multi-camadas, onde cada neurônio pode receber como entrada, armazenar e produzir valores do tipo '0', '1' e 'u'.

Em (CARVALHO; FAIRHURST; BISSET, 1994), é proposta uma rede chamada de SOFT (*Self Organizing Feature exTractor*), a qual possui uma arquitetura onde os neurônios são agrupados em blocos, e onde cada bloco é inicialmente uma grade de $N$ camadas de neurônios. Cada bloco cobre uma área diferente da imagem de entrada e seu mapeamento de conexão irá fornecer mais de um caminho a partir de um dado pixel na entrada a um dado neurônio de saída. Isto garante a migração dos dados através da rede e, por sua vez, permite a extração de características preenchendo o espaço de entrada sem um alto nível de conectividade dos nós.

A rede SOFT utiliza um algoritmo de treinamento não-supervisionado, onde os blocos da rede podem ser treinados em paralelo. As camadas de cada bloco de neurônios podem ser treinadas em *pipeline* paralelo, começando com os neurônios na primeira camada e, utilizando a saída produzida pelas camadas treinadas anteriormente, as camadas subsequentes podem ser treinadas. Cada neurônio da rede SOFT pode receber como entradas, armazenar e gerar os valores 'u', '0' e '1'.

A primeira camada é treinada apresentando todo o conjunto de treinamento da rede para seus neurônios e calculando a frequência de acesso para cada conteúdo de memória de cada neurônio. Cada camada sucessiva é treinada com as saídas produzidas pelas camadas anteriores quando todo o conjunto de treinamento é apresentado e propagado através da rede. A rede possui similaridades com a Rede *Neocognitron* (FUKUSHIMA, 2003), uma vez que ambas utilizam uma estrutura de múltiplas camadas. As camadas são treinadas sequencialmente com neurônios e, nas camadas mais profundas, os neurônios respondem a características mais complexas e cobrem uma área maior da imagem de entrada.

Em (JúNIOR et al., 2014), é apresentado um sistema de Localização Global baseado na Imagem (*Image-Based Global Localization*) - VibGL) que usa RNSP de Memória de

Acesso Aleatório de Generalização Virtual (RNSP VG-RAM). Para o mapeamento da imagem, é empregada uma RNSP VG-RAM que aprende as posições do mundo associadas com as imagens capturadas ao longo de uma trajetória. A rede G-RAM (*Generalizing Ramdon Access Memory*) consiste em uma variação do modelo RAM onde o conteúdo da memória, modificado durante o treinamento, é espalhado entre posições vizinhas. O espalhamento auxilia a generalização da rede para padrões não conhecidos, com base na assertiva de que estes padrões serão semelhantes. O espalhamento é feito com base na distância de *hamming* entre os endereços. Já na rede VG-RAM, não há memória alocada nas posições não acessadas durante o treinamento da rede. Durante a fase de teste, caso uma posição nova seja acessada, a saída do neurônio será calculada *on-line* com base nas informações armazenadas nas posições de memória vizinhas.

Durante a localização, as novas imagens da trajetória são apresentadas para a RNSP VG-RAM, que gera as suas posições no mundo. O trabalho apresenta experimentos com o sistema VibGL aplicado ao problema da localização de um carro autônomo. Os resultados do trabalho mostram que o sistema é capaz de aprender grandes mapas (vários quilômetros de comprimento) de ambientes do mundo real e realizar localização global com precisão média de cerca de 3m. Considerando-se uma tolerância de 10m, a rede VibGL é capaz de localizar o carro 95 por cento do tempo.

Em (FORECHI et al., 2016), é apresentado um aperfeiçoamento na rede VG-RAM de (JúNIOR et al., 2014). É descrito que o tempo de teste da RNSP VG-RAM para aplicações que exigem muitas amostras de treinamento pode ser grande, uma vez que ele aumenta com o tamanho da memória de cada neurônio. No trabalho, os autores apresentam uma nova arquitetura, a *Fat-Fast* RNSP VG-RAM. As redes *Fat-Fast* VG-RAM empregam *hashing* encadeados multi-índices para uma rápida busca na memória do neurônio. Esta técnica de *hashing* encadeada aumenta o consumo de memória VG-RAM (por isso, o nome*fat*), mas reduz o tempo de teste substancialmente (por isso, o nome*fast*), mantendo a maior parte de seu desempenho de aprendizagem de máquina.

Para resolver o problema do consumo de memória, é proposta uma técnica de agrupamento de dados para reduzir o tamanho global da memória do neurônio. Isto pode ser conseguido através da substituição de grupos de memória de neurônios pelos seus respectivos valores de centroide. Com esta abordagem, é possível reduzir o tempo

de teste da RNSP VG-RAM e o consumo de memória, mantendo um desempenho de aprendizagem de máquina elevado e aceitável.

O trabalho apresenta resultados com a *Fat-Fast* VG-RAM aplicada a dois problemas de reconhecimento de imagem. A primeira sendo para reconhecimento de dígitos manuscritos e a segunda para reconhecimento de sinais de trânsito. Os resultados experimentais mostram que, em ambos os problemas de reconhecimento, a nova abordagem de RNSP VG-RAM é capaz de funcionar três ordens de grandeza mais rápido e consumir duas ordens de grandeza menos memória do que a VG-RAM padrão, enquanto experimenta apenas uma pequena redução no desempenho do reconhecimento.

Em (MYERS, 1989), é discutido um estudo das funções de saída para as redes PLN e MPLN, assim como uma análise da importância da resolução da memória $\omega$. Inicialmente, é realizada uma breve introdução às RNSP com nó lógico probabilístico. Em seguida, é apresentado um estudo matemático do parâmetro $\omega$. Myers aponta que valores altos de $\omega$ protegem a rede de ruídos que podem ocorrer durante o treinamento. Por outro lado, é dito que utilizar valores altos de $\omega$ aumenta o custo de memória da rede e dificulta o processo de retornar a memória da rede a estágio 'u', quando necessário. Testes foram realizados com a aplicação de classificação do bit de paridade em sequências de 3 e 7 bits.

Em (NEDJAH et al., 2016), é proposta uma arquitetura em *hardware* reconfigurável de uma RNSP do tipo MPLN utilizando ainda o recurso de paralelismo em *pipeline*. A rede proposta utiliza a arquitetura piramidal, usual do modelo MPLN, e tem o objetivo de reconhecimento de imagens. O trabalho utiliza como conjunto de dados os algarismos da base de dados *NMIST*, mas limitada a um reduzido número de caracteres por questão de memória do *hardware*. A rede proposta utiliza do artificio de camadas paralelas dedicadas ao reconhecimento de cada uma das classes a ser identificada pela rede. O trabalho apresenta resultados do reconhecimento de caracteres tanto em ambiente de simulação quanto em síntese de *hardware*.

## 2.3 Considerações finais do capítulo

Neste capítulo, foram apresentados trabalhos relacionados ao problema de classificação e reconhecimento de imagem utilizando RNSPs. Inicialmente, foram descritos os trabalhos desenvolvidos com base na topologia WiSARD, a qual, apesar de antiga, ainda tem inspirado diversas otimizações muito eficazes no problema de classificação de imagens. O

problema da rede WiSARD é que esta apresenta perda de capacidade de generalização quando o problema a ser resolvido possui classes muito semelhantes, onde uma pequena variação no padrão o faz pertencer a uma nova classe. Apesar de muitos trabalhos apresentarem aperfeiçoamentos ao modelo WiSARD, estes, em essência, também carecem de capacidade de generalização em aplicações onde existem classes com características muito semelhantes.

Com relação às variações de RNSP diferentes do modelo WiSARD, as vantagens e desvantagens são inerentes às suas características. Por exemplo, a RNSP do tipo VG-RAM apresenta excelentes resultados no reconhecimento de imagens, entretanto sua aplicação depende de características difíceis de implementar em *hardware*, tal como o espalhamento em memória, o que torna essa aplicação menos interessante quando comparada a outras implementações de RNSP. Com relação às redes estocásticas, estas apresentam boa capacidade de generalização e tolerância a ruídos, mas podem apresentar um treinamento mais lento frente a modelos determinísticos, tal como a WiSARD, os quais usualmente possuem um treinamento de uma única época (*one-shot learning*).

Desse modo, nota-se que existem diversos trabalhos anteriores que abordam o reconhecimento de imagens a partir de RNSP. Entretanto, os trabalhos são focados em comprovar a eficácia de uma concepção teórica em RNSP, sendo que nenhum dos trabalhos aborda a questão de prover uma RNSP otimizada para uma determinada aplicação. Nota-se que os autores usualmente escolhem os parâmetros da rede de forma empírica, e não fornecem ao leitor qualquer base para que ele seja capaz de projetar uma RNSP otimizada.

Conforme visto anteriormente, o presente trabalho é focado em RNSP do tipo MPLN, pois esta possui boa capacidade de generalização e fácil implementação em *hardware*. Assim, o presente trabalho possui o objetivo de apresentar um estudo sobre a variação dos parâmetros intrínsecos de uma RNSP e propor diretrizes de projeto para um RNSP do tipo MPLN, de modo a auxiliar o leitor a escolher os parâmetros de uma RNSP de uma maneira que atenda às necessidades de seu projeto. Serão estudadas arquiteturas das redes para diversas aplicações, tal como reconhecimento de imagens, de modo a desenvolver diretrizes no projeto de uma RNSP MPLN para as referidas aplicações estudadas.

Adicionalmente, serão estudadas variações da topologia MPLN tradicional com o objetivo de aumentar a precisão da rede. Além disso, o presente trabalho visa propor

uma solução capaz de apresentar boa capacidade de generalização e tolerância a falhas, assim como um treinamento estável e que possua boa escalabilidade com o conjunto de treinamento e número de classes a serem reconhecidas.

# Capítulo 3

# CARACTERÍSTICAS DE PROJETO DE UMA REDE MPLN

As RNSPs são de fato interessantes para a solução de diversos problemas e possuem vantagens sobre as RNAs convencionais. Entretanto, assim como nas RNAs convencionais, a eficácia de uma RNSP se dá fortemente devido à escolha dos parâmetros de seu projeto, sendo que, para certas arquiteturas, a rede pode sequer convergir devido à inconsistência dos dados na fase de treinamento da rede. Este capítulo apresenta uma discussão a respeito dos parâmetros de projeto de uma RNSP.

Na Seção 3.1, discutem-se os parâmetros mais importantes para o projeto de uma RNSP, e o impacto de sua variação na rede. Na Seção 3.2, são discutidas as limitações da rede MPLN para o problema de classificação com múltiplas classes, onde podem ocorre conflitos de treinamento. Por fim, na Seção 3.3, é apresentada a modificação na rede MPLN proposta no presente trabalho: a rede MPLN modificada, ou Mod-MPLN , a qual visa solucionar as limitações da rede MPLN e ainda otimizar a fase de treinamento da rede.

## 3.1 Os parâmetros de projeto de uma MPLN

Os parâmetros de projeto de uma RNSP são decisivos para que a rede apresente alta eficiência e eficácia. A escolha dos parâmetros é um desafio para o projetista de uma RNSP e, muitas vezes, são escolhidos empiricamente ou a partir de tentativas e erros, não havendo um consenso estabelecido para a escolha dos mesmo. A seguir, serão discutidos os principais parâmetros que devem ser definidos no projeto de uma RNSP e seu impacto no funcionamento da rede.

### 3.1.1 Tamanho da tupla e número de neurônios

Conforme visto na Seção 1.3, em uma RNSP, um padrão de entrada binário de $W$ bits é dividido em $M$ partes de $N$ bits, as quais serão utilizadas para endereçar os $M$ neurônios do estágio de entrada. As $M$ partes de $N$ bits em que o padrão de entrada é dividido são chamadas de tuplas, conectividade ou *fan-in*. O tamanho da tupla deve ser escolhido com cuidado, uma vez que a memória utilizada cresce exponencialmente com o aumento do tamanho da tupla. Por exemplo, para uma tupla de 8 bits, a memória utilizada por cada neurônio RAM da rede é de $2^8$ bits (256 bits), enquanto que para uma tupla de 16 bits, a memória utilizada por cada neurônio RAM da rede é de $2^{16}$ bits (65.536 bits). Em (BRAGA; CARVALHO; LUDERMIR, 2000), é descrito que as RNSP geralmente possuem tupla na faixa de 2 a 10), devido ao aumento exponencial da memória utilizada com o aumento do tamanho da tupla.

Além disso, uma tupla grande prejudica a capacidade de generalização da rede, uma vez que os padrões aprendidos por cada neurônio são muito específicos (GARCIA, 2003). Desse modo, durante a fase de teste da rede, uma tupla com apenas um bit de diferença de uma tupla aprendida pela rede não seria identificada por esta. Isto pode ser entendido como uma perda de generalização, pois a rede não foi capaz de identificar um padrão (conjunto de tuplas) que era consideravelmente semelhante a um padrão conhecido pela RNSP. Tal fato se torna ainda mais agravado quando o tamanho da tupla é muito grande, conforme explicado em (ALEKSANDER; MORTON, 1989; BRAGA; CARVALHO; LUDERMIR, 2000; GARCIA, 2003). Esta limitação pode ser entendida como um *overfitting* da rede causado por um tamanho excessivo da tupla da rede.

Por outro lado, caso a tupla de bits seja muito pequena, um neurônio da rede iria identificar diversos padrões como pertencendo a uma determinada classe, deixando de ser capaz de distinguir as diferenças entre um padrão e outro em nível de tupla. Consequentemente, durante o treinamento da rede, uma posição da memória do neurônio seria modificada diversas vezes, em consequência de uma tupla de um padrão ser reconhecida diversas vezes ao longo das amostras de treinamento. Para o caso da rede WiSARD, onde cada posição da memória dos neurônios possui apenas um bit, isto levaria a uma rápida saturação da rede, pois uma posição é modificada com o valor '1' independentemente se ela corresponde a uma ou cem tuplas de amostras de treinamento. Assim, a rede perde a capacidade classificar padrões, uma vez que ocorre uma saturação da mesma. Segundo

(GARCIA, 2003; STONHAM, 1986), uma tamanho de tupla maior é necessário em casos onde os padrões de uma determinada classe forem muito similares, os dados forem suscetíveis a ruído, ou exista uma grande diversidade de padrões dentro de cada classe.

Para mitigar estes problemas na rede WiSARD, foi proposto o método de *bleaching* na rede (SOARES; SILVA; GREGORIO, 1998; BANDEIRA, 2010), onde cada posição de memória dos neurônios pode possuir mais de um bit, de modo que sejam capazes de distinguir a quantidade de vezes que uma determinada posição de memória do neurônio foi acessada durante o treinamento da rede. Para o caso de uma rede MPLN, este problema pode ser mais facilmente mitigado ao aumentar a quantidade de bits que compõem uma posição de memória de cada neurônio. Entretanto, tal técnica, assim como o *bleaching*, aumenta o consumo de memória da rede.

### 3.1.2 Número de estágios

O número de estágios é uma característica particular das redes em arquitetura em pirâmide, tal como as redes PLN e MPLN. Conforme visto na Seção 1.4, nas redes de arquitetura em pirâmide é proposta uma arquitetura semelhante à arquitetura multicamadas das RNAs convencionais, onde a saída dos neurônios de um estágio irá endereçar os neurônios do próximo estágio. Segundo (ALEKSANDER; MORTON, 1989), o tamanho da tupla deveria ser fixo ao longo de todos os estágios, mas, após pesquisa bibliográfica, não se encontrou um real motivo que impedisse a utilização de uma tupla variável ao longo dos estágios da rede. De todo modo, considerando um tamanho fixo do padrão de entrada e um tamanho fixo do tamanho da tupla ao longo dos estágios, o número de estágios da rede irá depender diretamente do tamanho da tupla. Especificamente, para um padrão de tamanho $W$ de bits e para um tamanho $N$ de tuplas, o número de estágios $D$ pode ser calculado de acordo com a Equação 7:

$$W = N^D. \tag{7}$$

Seguindo a recomendação de manter o tamanho da tupla fixo em todos os estágios (ALEKSANDER; MORTON, 1989; BRAGA; CARVALHO; LUDERMIR, 2000), o número de estágios se torna uma mera consequência do tamanho da tupla, conforme estabelece a equação 7. Entretanto, o presente trabalho propõe uma análise sobre outras arquiteturas de RNSP do tipo MPLN, onde o tamanho da tupla varia ao longo dos estágios, de modo

a observar a eficácia das redes e o impacto causado por não obedecer a equação ensinada por Aleksander.

Desse modo, surge a necessidade da escolha de um novo parâmetro, a saber o número de estágios que uma RNSP do tipo MPLN irá possuir, assim como o tamanho da tupla para cada um dos estágios. A grande vantagem de uma estrutura em pirâmide com mais de um estágio é que a rede é capaz de aprender características intrínsecas do padrão de entrada. Esta arquitetura é conhecida como redes profundas na literatura de redes neurais convencionais e (BANDEIRA, 2010) demonstra que ela é similarmente aplicável para RNSPs.

### 3.1.3 Resolução das posições de memória

A resolução das posições de memória é uma característica particular das redes MPLN, assim como de outras RNSPs com mais de um bit por posição de memória não abordadas em detalhe nesta dissertação. Conforme visto na Seção 1.5, na rede MPLN os bits de cada posição de memória de um neurônio são capazes de determinar uma probabilidade do referido neurônio emitir uma saída de valor lógico '1' quando a referida posição da memória é acessada pela tupla que endereça o neurônio. Assim, pode-se dizer que um neurônio MPLN apresenta um conjunto discreto de probabilidades de produzir uma saída de nível lógico '1', sendo este conjunto dependente da quantidade de bits nas posições de memória. Por exemplo, para o caso onde 3 bits são utilizados, cada posição da memória pode armazenar $2^3$ valores possíveis de probabilidade de o neurônio apresentar uma saída de nível lógico '1'.

Conforme adicionalmente visto na Seção 1.5, o treinamento de uma rede MPLN é geralmente realizado com base no algoritmo *reward-penalty*, logo existe um limite até onde o valor armazenado em uma dada posição de memória pode ser incrementado ou decrementado durante o treinamento. Em geral, estes valores correspondem a 0, representando 0% de probabilidade do neurônio emitir uma saída com nível lógico '1', e o valor '1', representando 100% de probabilidade do neurônio emitir uma saída com nível lógico '1'. Um valor pequeno de bits nas posições de memória pode causar a saturação da rede, uma vez que o valor de probabilidade máximo seria alcançado e a rede não seria capaz de se ajustar a novos padrões de entrada que endereçassem aquela posição da memória, similar à saturação da rede WiSARD. Por outro lado, um excessivo número de bits por

posição da memória irá aumentar substancialmente a memória consumida pelo sistema e o tempo necessário para treinar a rede.

Considera-se que uma RNSP MPLN converge quando todas as posições da memória de todos os seus neurônios possuem probabilidades de emitir uma saída de nível lógico '1' de 0 % ou 100 %, ou então nunca são acessadas durante o treinamento. Um valor diferente destes causaria um comportamento probabilístico na saída da rede, o que não é desejável.

A vantagem da rede MPLN com muitos bits em cada endereço de memória é que, durante treinamento, após diversos incrementos no valor armazenado em uma posição de memória, por exemplo na direção de encorajar uma saída de nível lógico '1', é muito difícil apagar este conhecimento sobre o padrão aprendido. Na verdade, para que o conhecimento acumulado por uma determinada posição da memória de um neurônio seja apagado, será necessário o mesmo número de ajustes na direção contrária à direção para a qual o neurônio estava sendo previamente ajustado. Desse modo, a RNSP MPLN apresenta uma proteção contra erros advindos de, por exemplo, amostras ruidosas durante o treinamento.

É importante notar que, para o caso da RNSP MPLN, apenas o erro global da rede é considerado durante o possível ajuste da mesma, logo uma parte ruidosa da imagem endereçando qualquer neurônio da rede pode causar erro na saída e, consequentemente, um ajuste incorreto de todos os neurônios da rede. Neste caso, um ajuste incorreto da rede iria penalizar a mesma apenas em um valor incremental, ou passo, determinado pela taxa de aprendizado, conforme a Equação 4. Entretanto, no caso da RNSP MPLN tal ajuste não será suficiente para levar a rede de volta para a incerteza (valor 'u'). Assim, a probabilidade da rede emitir uma saída de nível lógico '1' será pouco alterada em um único ciclo de ajuste.

Por outro lado, para o caso de uma rede PLN, um único erro resultante de uma inconsistência em qualquer neurônio da rede resultaria em uma posição da memória de cada neurônio sendo modificada para o nível lógico 'u'. Dessa forma, um único erro apagaria qualquer conhecimento prévio da rede representado pelo valor previamente armazenado, e independentemente de quantos outros padrões de treinamento haviam reafirmado aquele valor prévio. Tal fato reforça a importância de uma certa resiliência no ajuste do valor de probabilidade armazenado em cada posição de memória dos neurônios. Esta resiliência

é determinada essencialmente pela quantidade de bits em cada posição de memória dos neurônios.

O aumento da quantidade de bits por posição de memória da RNSP MPLN irá, além do aumento da memória RAM utilizada, causar o aumento da dificuldade de retornar um neurônio para a posição 'u', quando isto for necessário. Este recomeço do treinamento pode ser necessário em caso de dados ruidosos ou devido a um mal ordenamento dos padrões de treinamento, os quais levariam posições de memória para longe de uma saída de nível lógico 'u'. Assim, uma alta resiliência da rede ao aprendizado pode também ser prejudicial durante o treinamento. Sendo assim, o número de bits por posição de memória $\omega$ deverá ser escolhido para balancear a proteção da rede contra ajustes errôneos (resiliência do aprendizado) contra a quantidade de memória disponível e razoável para uma implementação viável da RNSP.

Myers (MYERS, 1989) apresenta um estudo onde, para o exemplo de treinamento e classificação do problema da paridade de bits, contata-se que o valor $\omega$ de probabilidades possíveis em uma posição de memória de um neurônio deve ser $5 \leq \omega \leq 15$, especificamente $\omega = 11$. Nota-se que os valores de probabilidade possíveis em uma posição de memória estão diretamente relacionados com o tamanho de bits em cada posição de memória, através da relação $w = 2^f$, onde $f$ é o tamanho em bits de cada posição em memória.

### 3.1.4 Taxa de aprendizado do algoritmo de treinamento *reward-penalty*

Para as RNSPs do tipo MPLN, o algoritmo de treinamento *reward-penalty* inclui um elemento correspondente à taxa de aprendizado das RNAs convencionais, a saber, o parâmetro $\eta$ definido na Equação 4. Especificamente, a equação do *reward-penalty* possui um fator multiplicador, o qual irá definir o quanto a RNSP será encorajada (*reward*) ou desencorajada (*penalty*) a produzir uma resposta apresentada para um determinado padrão do conjunto de treinamento. De modo geral, um valor baixo para a taxa de aprendizado não compromete a capacidade de aprendizado da rede, mas torna o aprendizado da RNSP excessivamente lento e com alto custo computacional para o sistema. Por outro lado, uma alta taxa de aprendizado pode causar a saturação da memória da RNSP MPLN, ou seja, o valor armazenado nas posições de memória irá atingir seu valor máximo ou mínimo rapidamente. Ao atingir o valor máximo ou mínimo de probabilidade, a rede não seria mais

capaz de ajustar a probabilidade daquela posição de memória na direção destes valores para outras amostras de treinamento.

É importante observar que a taxa de aprendizado do algoritmo de aprendizado e a resolução da posição de memória são parâmetros correspondentes. Isso pode ser afirmado uma vez que a resolução da memória determina quantos passos (ajustes) são necessário para alcançar o valor máximo inferior ou superior da probabilidade do neurônio emitir uma saída de nível lógico '1'. Por outro lado, a taxa de aprendizado define quantos passos serão "percorridos"durante um ajuste do valor de memória da rede. A Tabela 2 mostra um comparativo entre diversas resoluções de memória e dois valores de taxa de aprendizagem, de modo a observar a relação entre ambos os parâmetros.

Tabela 2: Relação entre a resolução da memória e a taxa de aprendizagem

| $\omega$ | 'u' | $\eta$ | Número de ajustes |
|---|---|---|---|
| **5 posições (0 a 4)** | 2 | 1 | 2 |
| | | 2 | 1 |
| **7 posições (0 a 6)** | 3 | 1 | 3 |
| | | 3 | 1 |
| **9 posições (0 a 8)** | 4 | 1 | 4 |
| | | 2 | 2 |
| **11 posições (0 a 10)** | 5 | 1 | 5 |
| | | N/A | N/A |
| **13 posições (0 a 12)** | 6 | 1 | 6 |
| | | 2 | 3 |
| **15 posições (0 a 14)** | 7 | 1 | 7 |
| | | N/A | N/A |

Nota-se que é necessário o mesmo número de ajustes para o caso de uma resolução de memória de 7 posições, com uma taxa de aprendizagem de 1, e para o caso de uma resolução de memória de 13 posições, com taxa de aprendizagem de 2. Dessa forma, pode-se definir a relação entre a taxa de aprendizagem e a resolução da memória, conforme a Equação 8:

$$\eta = \frac{\omega - 1}{2 * ajuste}, \tag{8}$$

onde $\eta$ é a taxa de aprendizagem da rede, $\omega$ é a resolução da memória e *ajuste* é a quantidade de ajustes iterativos na memória para atingir um valor máximo a partir do valor inicial 'u'.

Assim, nota-se que a taxa de aprendizagem possui os mesmos impactos na rede do que a resolução da memória. Uma taxa de aprendizado muito baixo irá aumentar o

tempo de treinamento da rede, uma vez que serão necessários mais ajustes até que a rede atinja os valores desejados limites de probabilidade 0% e 100%. Por outro lado, uma taxa de aprendizagem muito alta pode causar a rápida saturação da rede, atingindo os valores limite de probabilidade muito rápido e não sendo capaz de se ajustar na mesma direção para novas amostras. Adicionalmente, o valor da taxa de aprendizagem está relacionado com a facilidade que a rede é capaz de "esquecer"o conhecimento do problema, ou seja, retornar seus conteúdos de memória para o valor 'u'. Conforme visto na Seção 3.1.3, retornar o conteúdo da memória para 'u' pode ser interessante no caso de dados ruidosos ou mal ordenamento dos padrões de treinamento.

## 3.2 Limitações da MPLN com múltiplas classes

Conforme visto nas Seções 1.4 e 1.5, uma rede MPLN de uma única camada é capaz de aprender e classificar um padrão entre duas classes, a primeira como pertencendo ao padrão para o qual a rede foi treinada, geralmente indicado por uma saída de nível lógico '1', e a segunda como não pertencendo ao padrão para o qual a rede foi treinada, geralmente indicado por uma saída de nível lógico '0'. Para problemas onde existem mais de duas classes a serem identificadas, deve-se utilizar diversas redes paralelas, também chamadas de camadas paralelas, sendo cada camada treinada para identificar uma determinada classe (nível lógico '1') e não identificar padrões pertencentes às demais classes (nível lógico '0').

Para o caso de uma RNSP do tipo WiSARD, esta estratégia não afeta o treinamento da rede, uma vez que todas as posições de memória são iniciadas com nível lógico '0' e o treinamento da rede consiste em substituir os '0's por '1's apenas para a camada da rede para a qual o padrão de treinamento pertence. A aferição de qual camada da RNSP deverá ser ajustada de acordo com um novo padrão de treinamento é possível uma vez que o treinamento da rede WiSARD é do tipo supervisionado. Assim, cada amostra de treinamento $k$ possui um valor desejado $d(k)$ que é utilizado para definir de antemão qual camada da rede será modificada para aprender o padrão. Desse modo, para o caso de uma RNSP capaz de identificar 10 classes, a rede deverá apresentar 10 camadas paralelas. Por exemplo, para uma determinada amostra de treinamento pertencendo a uma classe 3, apenas a terceira camada da RNSP seria endereçada e treinada para aquele padrão de treinamento, enquanto as demais camadas permaneceriam inalteradas.

Por outro lado, para a rede MPLN, todas as posições de memória são iniciadas com iguais probabilidades da rede emitir uma saída com nível lógico de '0' ou '1'. Após o treinamento da RNSP MPLN, caso uma posição da memória possua valor correspondente a uma probabilidade diferente de 0% ou 100%, o neurônio irá produzir uma saída com base em um valor de probabilidade, o que não é desejável para uma rede com alto valor de precisão. Tal situação se agrava ainda mais no caso onde o neurônio possui valor 0,5 armazenado em uma posição de memória acessada, valor este inicialmente atribuído a todas as posições de memória de todos os neurônios. Durante a fase de teste, caso uma posição de memória contendo o valor inicial de 50% ('u') seja acessada, a saída do neurônio será completamente aleatória. Assim, é necessário que a rede seja treinada tanto para incrementar a probabilidade das posições de memória que devem emitir uma saída com nível lógico '1', bem como decrementar a probabilidade das posições de memória que devem emitir uma saída com nível lógico '0'.

Além disso, ao contrário da rede WiSARD, na RNSP MPLN todas as camadas da rede devem ser treinadas com todos os padrões de treinamento. Nesse caso, para todos os padrões de treinamento que a camada da rede deve identificar, a saída desejada (treinamento supervisionado) para o referido padrão será '1', enquanto a saída da camada para todos os outros padrões de treinamento será '0'. Desse modo, espera-se que a camada treinada seja ajustada para emitir uma saída de nível lógico '1' para os padrões pertencentes à classe para a qual a camada foi treinada e '0' para todas as demais classes. Para o exemplo de uma RNSP do tipo MPLN adaptada para reconhecer 10 classes, para um determinado padrão pertencendo a umas das 10 classes que a rede foi treinada, apenas uma das camadas da rede deverá apresentar uma saída de nível lógico '1', enquanto as demais nove camadas deverão apresentar nível lógico '0'.

O problema que surge com essa metodologia é que as camadas da RNSP são treinadas consideravelmente mais para identificar padrões como não-pertencentes à classe que a camada deve identificar do que para a classe que a camada efetivamente deve identificar. Para o exemplo de uma RNSP com 10 camadas responsáveis por identificar 10 classes, considerando que o conjunto de treinamento seja igualmente distribuído entre todas as classes, cada camada da rede será treinada nove vezes mais para identificar um padrão de treinamento como não-pertencente à classe identificada pela rede (saída com nível lógico '0') do que para identificar o padrão como pertencente à classe identificada pela rede (saída

com nível lógico ‘1’). Este desbalanceamento entre as classes individuais de cada camada (classe ‘0’ ou classe ‘1’) pode comprometer a convergência da rede. Caso as amostras de treinamento pertencentes a classes diferentes sejam muito semelhantes, algumas posições da memória dos neurônios serão endereçadas tanto por padrões pertencentes a classe a ser reconhecida quanto pelos demais padrões. Entretanto, a rede será modificada muito mais vezes com o objetivo de emitir uma saída de nível lógico ‘0’ do que para emitir um nível lógico ‘1’.

Outro problema que surge com a RNSP MPLN para problemas de múltiplas classes é a possibilidade de garantir que a rede apresentará apenas uma saída com nível lógico ‘1’ para um determinado padrão a ser reconhecido, o que é desejável, uma vez que determinado padrão deve ser identificado como pertencendo a apenas uma classe. Como a RNSP MPLN possui um conjunto de bits que expressam a probabilidade de um neurônio emitir uma saída com nível lógico ‘1’, é possível que, após o treinamento da rede, nem todas as posições de memória de todos os neurônios tenham sido ajustados para emitir os valores ‘1’ e ‘0’ com 100 % de probabilidade (comportamento determinístico). Desse modo, caso uma determinada camada da RNSP possua uma neurônio com o valor de probabilidade 10% armazenado em uma posição de sua memória, uma camada da rede pode vir a emitir uma saída de nível lógico ‘1’ (neste caso, com 10 % de chance), enquanto este neurônio deveria, na verdade, emitir saída com nível lógico ‘0’ (neste caso, com 90 % de chance).

Quando este erro ocorre no estágio de saída de uma camada da rede, a rede irá produzir uma saída errada a respeito de um determinado padrão a ser reconhecido. Para o exemplo de uma rede com 10 camadas responsáveis por identificar 10 classes diferentes, e considerando que cada padrão pertence a apenas uma classe, a probabilidade de ocorrer um erro na rede e ela produzir mais de uma saída com nível lógico ‘1’ é considerável.

Na rede WiSARD, foi visto na Seção 1.3.2 que um problema análogo onde mais de uma camada poderia atingir o valor de semelhança mínimo pré-estabelecido pelo discriminador é facilmente resolvido através de uma modificação no discriminador. Esta modificação, denominada multi-discriminador (ALEKSANDER; THOMAS; BOWDEN, 1984), é configurada para identificar um padrão como pertencente à classe correspondente à camada com maior semelhança. Em outras palavras, o padrão irá pertencer à classe da camada com maior número de neurônios identificando as tuplas do padrão apresentado como pertencente à sua classe. Desse modo, o discriminador é capaz de garantir que a

rede identifique um padrão a ser reconhecido como pertencente a uma única classe, independentemente se mais de uma camada da rede atingiu o valor de semelhança (limiar) pré-estabelecido pela rede.

Para o caso de uma rede MPLN, uma solução possível é implementar um discriminador na saída das camadas da rede com o objetivo de processar a saída das camadas e necessariamente classificar o padrão como pertencente a apenas uma classe. Por exemplo, é proposto um discriminador para identificar o padrão como pertencendo a classe que tenha sido produzido com o maior valor de certeza da rede. Em outras palavras, o discriminador é configurado para identificar o padrão como pertencente à classe cuja camada possui neurônio no estágio de saída com maior valor de probabilidade armazenado na posição de memória acessada. Por exemplo, caso a camada 3 da rede emitiu uma saída de nível lógico '1' a partir de um valor de probabilidade 1,0 (100 % de probabilidade) e a camada 7 da rede emitiu uma saída de nível lógico '1' a partir de um valor de probabilidade 0,3 (30 % de probabilidade), o discriminador irá considerar que o padrão apresentado à rede pertence à classe 3.

Entretanto, esta estratégia pode ainda ser ineficiente no caso onde um erro ocorre nos estágios anteriores da rede. Neste caso, devido à arquitetura em pirâmide da rede, uma saída errônea de um estágio anterior endereçaria um neurônio indesejável do estágio seguinte e este estágio poderia não representar a saída da rede para um determinado padrão. Neste caso, o discriminador para uma rede MPLN não seria eficaz em filtrar este erro, uma vez que o grau de certeza do último estágio da rede independe do grau de certeza do estágio anterior, onde ocorreu o erro. Sendo assim, torna-se necessária uma nova forma de garantir que apenas a camada da RNSP mais adequada para um determinado padrão, e apenas esta camada, identifique o referido padrão como pertencente a uma classe da rede.

## 3.3 Modificação proposta: RNSP MPLN modificada

Tendo em vista as limitações apresentadas acima para uma RNSP do tipo MPLN para um problema de classificação com múltiplas classes, o presente trabalho propõe uma modificação na arquitetura da RNSP do tipo MPLM de modo a sanar estas limitações. A rede MPLN modificada, doravante definido como RNSP Mod-MPLN, utiliza conceitos de uma rede WiSARD e se diferencia de uma rede MPLN em dois aspectos. O primeiro se refere

ao treinamento da rede e o segundo diz respeito à implementação de um discriminador para selecionar, a partir dos resultados apresentados por cada camada paralela da rede, a qual classe pertence um determinado padrão a ser reconhecido.

O treinamento da rede Mod-MPLN se assemelha ao da rede WiSARD no sentido de que cada camada é treinada apenas com os padrões de treinamento que ela deve identificar. Por exemplo, a camada responsável por identificar o algarismo 3 apenas será somente treinada com padrões de treinamento que representem o número 3, da forma como ocorre com a rede WiSARD. Desse modo, cada camada da rede Mod-MPLN terá conhecimento a respeito de identificar com saída de nível lógico '1' todos os padrões de treinamento pertencentes à sua classe, mas não será treinada para identificar com nível lógico '0' os demais padrões não-pertencentes a sua classe. Após treinada, a rede apresentará saídas próximas do nível aleatório (probabilidade de 50 %) para os padrões diferentes daqueles para os quais ela foi treinada para identificar, o que demonstra a ignorância da rede para os demais padrões.

O lado positivo deste ajuste se reflete na velocidade de convergência do treinamento, o qual é devido a dois principais motivos. O primeiro é devido à quantidade de amostras que são apresentadas a cada camada. Na rede MPLN, todas as amostras de treinamento de todas as classes são apresentadas a cada camada da rede, enquanto na rede Mod-MPLN apenas as amostras de treinamento pertencentes à classe de uma camada é que são apresentadas à mesma. Desse modo, para um problema de classificação contendo 10 classes distintas e com quantidade de amostra balanceada, a quantidade de amostras que será apresentada a uma camada da rede Mod-MPLN será 10 vezes menor do que para a rede MPLN. O segundo motivo é por conta do algoritmo de treinamento *reward-penalty* ser iterativo, ao contrário do *one-shot learning* da rede WiSARD. Isto significa que o treinamento da rede MPLN pode levar diversas épocas, principalmente pelo fato de que uma mesma posição de memória de um determinado neurônio pode ser endereçada por padrões de treinamento pertencentes a classes diferentes. Isto causará um incremento e decremento iterativo do valor de probabilidade naquela posição de memória, até que a própria rede se ajuste para superar esta inconsistência. Por outro lado, a rede Mod-MPLN supera estes conflitos na fase de treinamento uma vez que cada camada é apenas ajustada com o objetivo de reconhecer o padrão de treinamento para a qual a referida camada está sendo treinada. Em outras palavras, as posições de memória dos

neurônios são ajustadas no sentido de produzir um valor de nível lógico '1' para os padrões de treinamento que ela visa identificar. Não há um ajuste com relação aos padrões de treinamento não pertencentes à classe objeto da camada da rede. Como o ajuste da rede Mod-MPLN é sempre no sentido de emitir uma saída de nível lógico '1', não há conflitos de treinamento e o treinamento da rede se torna consideravelmente mais rápido.

O treinamento da rede Mod-MPLN arquitetura piramidal proposto no presente trabalho se difere do MPLN principalmente na aplicação de um problema de classificação de múltiplas classes. Especificamente, o treinamento da rede Mod-MPLN é dado a seguir:

**Algoritmo 4:** Treinamento da rede Mod-MPLN

**início**
  Inicie a memória de todos os $M$ neurônios com valor $0,5$;
  **repita**
    **para** *i=1: k amostras de treinamento* **faça**
      Divida a amostra $k_i$ em $M$ tuplas de $N$ bits;
      Enderece os $M$ neurônios de entrada da camada $l$ com as $M$ tuplas, com base em $d(k_i)$;
      **para** *j=1:* $(e-1)$ *estágios da rede* **faça**
        Enderece os $M$ neurônios de entrada com as $M$ tuplas;
        Calcule a saída $s_{i,j,c}$ de cada neurônio;
        Forme tuplas com as saídas $s_{i,j,c}$ para a camada seguinte;
      **fim**
      Calcule a saída $s_{i,e,c}$ da camada de saída;
      Calcule o erro $E_{k,c} = abs(d(k_i) - s_{i,e,c})$;
      **se** $E_{k,c} = 0$ **então**
        $M[I]_z = M[I]_z + ajuste$ para $c$ onde $E_{k,c} = 0$ (**reward**);
      **fim**
      **senão**
        $M[I]_z = M[I]_z + ajuste$ (**penalty**);
      **fim**
    **fim**
  **até** *Condição de parada*;
**fim**

Durante a fase de teste, a rede Mod-MPLN precisa garantir que apenas uma camada paralela da rede identifique o padrão de treinamento. Desse modo, é proposto um discriminador capaz de selecionar a camada paralela que foi melhor ajustada para identificar este padrão. O discriminador utilizado na rede Mod-MPLN tem o objetivo de solucionar a deficiência da rede MPLN em eliminar os elementos probabilísticos que podem influenciar negativamente a rede durante a fase de reconhecimento. Além disso, o discriminador também garante que o treinamento da rede Mod-MPLN, o qual é realizada

apenas com os padrões respectivos às classe que devem ser identificadas, não impacte negativamente na eficácia da rede. Assim como na rede WiSARD, o discriminador utilizado tem o objetivo de selecionar, dentre as camadas paralelas da rede Mod-MPLN, aquela que apresenta a resposta identificando o padrão com o maior grau de confiança.

A diferença do discriminador da rede Mod-MPLN é que este considera o valor de probabilidade armazenado em todos os neurônios acessados por um padrão a ser reconhecido. Especificamente, o discriminador da rede Mod-MPLN realiza um somatório do valor de probabilidade identificado pelos bits das posições de memória de cada um dos neurônios endereçados pelo padrão a ser reconhecido, desde o estágio de entrada até o estágio de saída da rede. Tal implementação se fundamenta nos ensinamentos da literatura de RNSP (BRAGA; CARVALHO; LUDERMIR, 2000; ALEKSANDER; MORTON, 1989), os quais mostram que, nas RNSPs, todo o conhecimento da rede reside nas memórias do neurônio. Desse modo, o discriminador da rede Mod-MPLN tem o objetivo de utilizar toda a informação aprendida pela rede. A Figura 6 ilustra o discriminador da rede Mod-MPLN para uma única camada, de modo a facilitar a visualização.

O discriminador da rede irá somar o grau de certeza do valor emitido por cada um dos neurônios da rede, com base na saída apresentada pelo neurônio e no valor de probabilidade armazenado na posição de memória endereçada. O grau de certeza de cada neurônio consiste em um número que irá variar de acordo com os possíveis valores de probabilidade que podem ser expressos por um neurônio e que corresponde ao quão precisa foi a resposta da rede. O cálculo do grau de certeza para uma rede com neurônios com posições de memória com 10 valores de probabilidade é definida pela Equação 9:

$$C\left[i\right] = \begin{cases} P(i) & \text{se } s(i) = 1 \\ 1 - P(i) & \text{se } s(i) = 0, \end{cases} \tag{9}$$

onde: $s(i)$ é a saída de um neurônio $i$ da rede; $C(i)$ é o grau de certeza do neurônio $i$; e $P(i)$ é a probabilidade associada à posição de memória do neurônio.

O algoritmo 5 descreve as etapas realizadas pelo discriminador da rede Mod-MPLN para selecionar, dentre as camadas paralelas da rede, aquela que melhor identifica o padrão desconhecido a ser classificado.

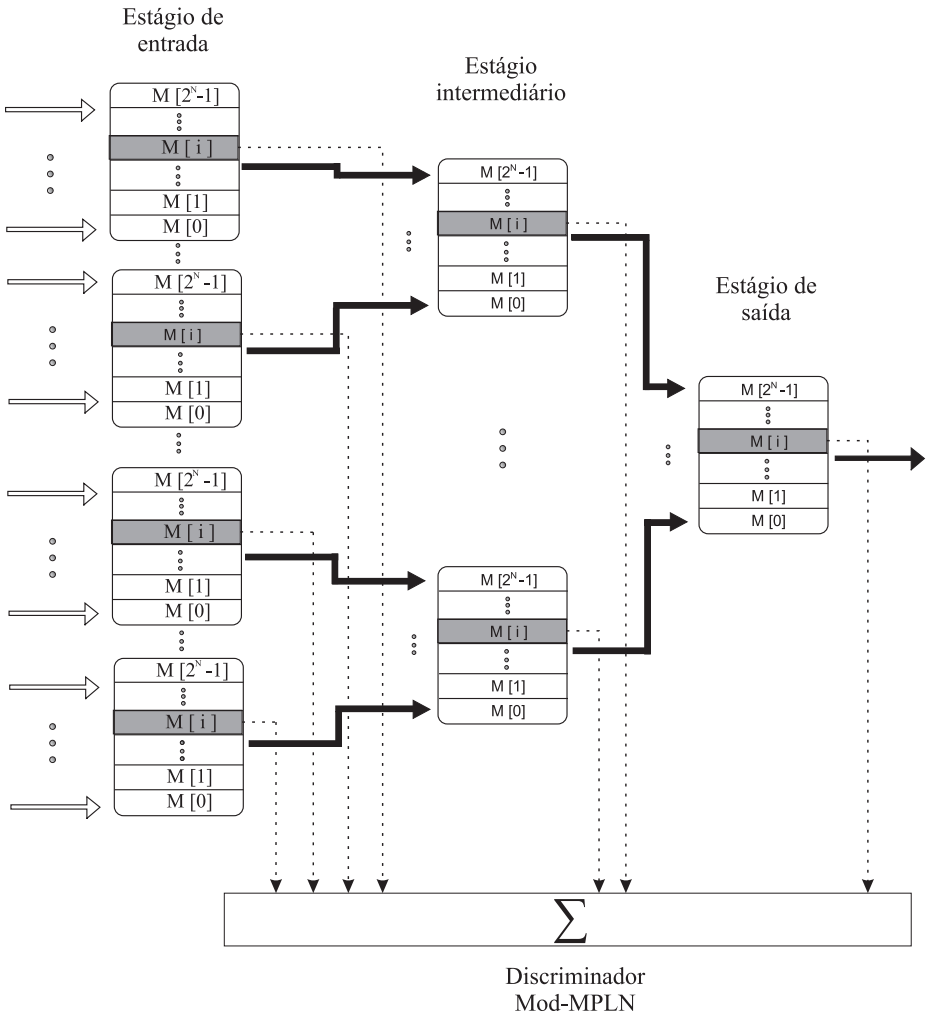



Figura 6: Discriminador para a RNSP mod-MPLN

**Algoritmo 5:** Cálculo da saída da rede Mod-MPLN de acordo com o discriminador

**início**
  **para** *j=1: l camadas da rede* **faça**
    **para** *i=1: M amostras de treinamento* **faça**
      Calcular C(i) conforme 9;
      $soma(j) = soma(j) + C(i)$;
    **fim**
  **fim**
  Calcule o maior índice de $soma(j)$;
  Identifique o padrão como pertencente à classe do maior índice de $soma(j)$;
**fim**

A grande vantagem da rede Mod-MPLN é que a mesma utiliza todo o conhecimento armazenado pela rede durante a fase de treinamento. Conforme visto na Seção 1.2.2, o aprendizado das RNSPs é dado pela escrita na memória RAM dos neurônios. Para as redes

do tipo MPLN, este aprendizado é realizado pelo ajuste iterativo dos valores nas posições de memória através do algoritmo de treinamento *reward-penalty* (MYERS, 1988). Sendo assim, ao aferir o grau de certeza da rede, que pode ser entendido como o quão bem a rede foi ajustada para identificar um padrão como pertencendo a uma classe determinada, é desejável que o discriminador utilize todo o conhecimento adquirido pela rede ao escolher a qual classe o padrão a ser reconhecido pertence.

## 3.4 Considerações Finais do Capítulo

Neste capítulo foram discutidos os principais desafios no projeto de uma RNSP, em especial uma RNSP do tipo MPLN. Conforme visto, os parâmetros da rede definem a sua estrutura e impactam na sua eficiência e eficácia, onde comummente deve-se fazer um *trade-off* entre eficácia da rede, alto custo de processamento e consumo de memória. Adicionalmente, foi visto que a rede MPLN apresenta limitações para problemas de classificação de múltiplas classes, uma vez que cada camada paralela da rede deve ser treinada para não identificar todas as classes, com exceção daquela que ela deve reconhecer. Por fim, foi apresentada a modificação proposta pelo presente trabalho à arquitetura MPLN: a rede Mod-MPLN, a qual visa solucionar as limitações da rede MPLN e ainda otimizar a fase de treinamento da mesma.

# Capítulo 4

# ASPECTOS DE IMPLEMENTAÇÃO

NESTE capítulo, são apresentadas as particularidades das simulações realizadas no presente trabalho. O principal objetivo da dissertação é realizar um estudo da RNSP do tipo MPLN, apresentar resultados comparativos entre as arquiteturas de RNSP estudadas e simuladas e, a partir dos resultados obtidos, propor diretrizes de projeto para uma RNSP, sejam elas para a rede WiSARD, para a rede MPLN ou para a modificação da rede MPLN proposta, a rede Mod-MPLN. Todas as RNSPs foram implementadas em ambiente de simulação, utilizando o *Matlab*®. A aplicação utilizada para avaliar a capacidade de aprendizado e eficiência das redes foi o conjunto de dados de algarismos manuscritos MNIST e o problema de classificação do bit de paridade, os quais serão brevemente descritos neste capítulo.

Na Seção 4.1, é apresentado o conjunto de dados da aplicação escolhida para as simulações, o qual contém os dados que serão utilizados para formar um dos conjuntos de dados para endereçar a rede. Na Seção 4.1.1, é apresentada a técnica de pré-processamento utilizada no conjunto de dados obtidos, de modo a avaliar o desempenho das redes para um conjunto de dados melhor ajustado. Na Seção 4.1.2, descreve-se a técnica de separação dos conjuntos de entradas, de modo a facilitar a classificação do desempenho das redes para conjuntos de entrada de qualidade boa, média ou ruim. Além disso, na Seção 4.1.3, é apresentada a arquitetura proposta para treinamento e teste da rede. Na Seção 4.2, é apresentada a segunda aplicação estudada na dissertação, a classificação do bit de paridade em sequências de $n$ bits. Por fim, na Seção 4.4, são descritas as arquiteturas de RNSP do tipo MPLN e Mod-MPLN que foram implementadas na presente dissertação para ambas as aplicações.

## 4.1 Aplicação em classificação de imagens

A aplicação em classificação de imagens deste trabalho se limita à classificação de imagens binárias de algarismos. Para isto, foi utilizada a base de dados de imagens de algarismos manuscritos de 0 a 9 do conjunto de dados MNIST (HULL, 1994), o qual faz parte do conjunto NIST (*National Institute of Standards and Technology*). A base de dados MNIST compreende 60.000 amostras para o conjunto de treinamento e 10.000 amostras para o conjunto de teste, as quais possuem seu respectivo *label* em algarismo arábico. A base de dados foi escolhida por possuir fácil implementação em RNSPs e por ser uma base de dados amplamente utilizada, possibilitando comparações entre os resultados obtidos. Para que as imagens de dados fossem utilizadas para endereçar os neurônios do estágio das redes implementadas, foi necessário binarizar estas imagens, convertendo-as em formato preto e branco.

Nota-se que a base de dados MNIST pode ser muito penosa para as RNAs, dado que certos algarismos manuscritos são muito diferentes do que se espera para determinado algarismo, alguns sendo difíceis até mesmo para identificação a olho nu. Além disso, muitos algarismos estão deslocados do centro da quadro da imagem, o que dificulta a classificação correta das mesmas. A seguir, são propostas duas formas de contornar este problema e melhor avaliar a capacidade de reconhecimento das redes.

### 4.1.1 Pré-processamento da imagem

Com o objetivo de tornar o conjunto de dados mais uniforme, é proposta uma técnica de pré-processamento. A partir da visualização das imagens do conjunto de dados MNIST, observou-se que muitas das mesmas se encontravam descentralizadas no quadro da imagem. Esta descentralização cria espaços não ocupados pelos algarismos (espaços em branco) na imagem, os quais poderiam endereçar neurônios diversas vezes, causando o aprendizado de características não correspondentes aos algarismos. Por outro lado, os espaços em branco internos das imagem, os que delimitam a forma dos algarismos, são muito importantes para a imagem e devem ser preferencialmente preservados.

A Figura 7 mostra exemplos dos algarismos manuscritos da base de dados MNIST sem qualquer técnica de pré-processamento, onde os espaço em preto representam o valor lógico ‘0’ e os espaço em branco representam o valor lógico ‘1’. Desse modo, o pré-processamento proposto consiste em remover as linhas e colunas inteiramente pretas da

imagem, ou seja, as linhas e colunas sem qualquer parte do algarismo. A eliminação destas linhas irá diminuir o tamanho da imagem. Em seguida, a imagem de tamanho reduzido é redimensionada para seu tamanho original utilizando uma interpolação bicúbica (KEYS, 1981; AKYAMA, 2010) através do Matlab. A função capaz de realizar este redimensionamento é uma função padrão conhecida do Matlab, a saber, a função *imresize*. A nova imagem criada irá possuir as mesmas características da imagem original, mas o algarismo ilustrado na imagem irá preencher pelo menos um *pixel* das quatro extremidades do quadro da imagem. A figura 8 ilustra exemplos de algarismos manuscritos após a realização do pré-processamento.

Figura 7: Exemplos de imagens do conjunto MNIST sem pré-processamento

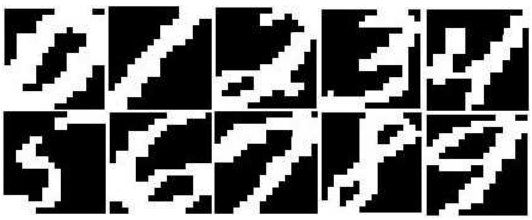


Figura 8: Exemplos de imagens do conjunto MNIST com pré-processamento

Nota-se que as imagens preprocessadas ficam centralizadas no quadro da imagem, o que facilita a rede extrair características dos algarismos. Conforme a figura ilustrada acima, as imagens processadas estão melhor alinhadas ao centro do quadro da imagem, e também possuem menos espaços em branco, o que facilita o aprendizado e reconhecimento das redes.

### 4.1.2 Separação de conjuntos de imagem

Apesar de o preprocessamento elencado acima auxiliar no treinamento e classificação das redes, ainda assim algumas imagens ilustram algarismos muito diferentes do esperado. Logo, para estas amostras é possível que a rede continue a falhar no reconhecimento das

imagens. Como o conjunto de dados MNIST possui diversas imagens, estas imagens de qualidade inferior impedem a rede de alcançar bons resultados em termos de precisão.

De modo a contornar isto, propõe-se separar os conjuntos de treinamento e teste em três grupos de acordo com a qualidade das imagens. O critério para avaliar a qualidade das imagens foi o coeficiente de correlação de *Pearson* (PEARSON, 1895). Para isto, o coeficiente de correlação de *Pearson* foi comparado entre as próprias imagens da mesma classe, tirando uma média com objetivo de encontrar a imagem com maior grau de similaridade entre as demais e, consequentemente, a que melhor represente o modelo do algarismo pertencente a uma determinada classe. Encontradas as imagens modelo para cada uma das classes, estas foram utilizadas para classificar as demais.

Especificamente, cada uma das imagens do conjunto de treinamento e teste, dependendo de seu valor esperado $d(k)$ ou etiqueta *label*, é comparada com a respectiva imagem modelo para determinar seu coeficiente de correlação de *Pearson*. Caso o coeficiente de correlação seja igual ou superior a 0,6, ou seja, $p \geq 0,6$, esta imagem é dita como de boa qualidade e integra o grupos das imagens de boa qualidade. Caso o coeficiente de correlação seja superior a 0,6 e inferior a 0,4, ou seja, $0,4 < p < 0,6$, esta imagem é dita como de média qualidade e integra o grupos das imagens de média qualidade. Por fim, caso o coeficiente de correlação seja igual ou inferior a 0,4, ou seja, $p < 0,4$, esta imagem é dita como de qualidade ruim e integra o grupo das imagens de baixa qualidade.

Assim, o conjunto de algarismos da base MNIST foi dividido em três grupos, a saber, o grupo de imagens de boa qualidade, denominado grupo GB, o grupo de imagens de qualidade média, denominado GM, e o grupo de imagens de qualidade ruim, denominado GR. Ao separar as imagens em três grupos, é possível observar de forma mais correta as propriedades de cada rede estudada, uma vez que estas não terão seu desempenho mascarado por imagens de baixa qualidade.

### 4.1.3 Estrutura da rede RNSP para classificação de imagens

Conforme discutido anteriormente, todas as arquiteturas de RNSP foram implementadas em *software*, especificamente pela plataforma de simulação MATLAB®. A estrutura de projeto proposta possui algumas peculiaridades intrínsecas ao *software* de simulação, em especial o *script* elaborado para simular o treinamento e teste das redes. As peculiaridades do *software* desenvolvido não são interessante para a análise do presente trabalho e,

consequentemente, não serão discutidas. De qualquer modo, a estrutura proposta pode ser descrita de modo simplificado conforme o diagrama de blocos da Figura 9 abaixo.

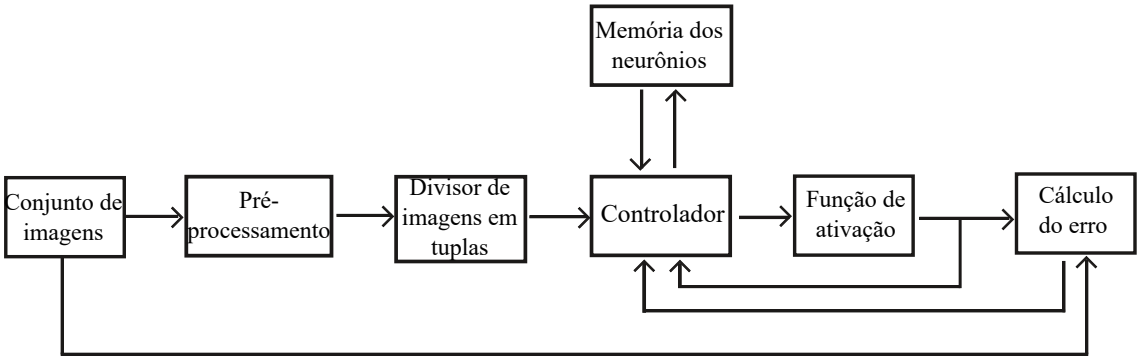



Figura 9: Diagrama de blocos para a implementação de reconhecimento de algarismos

Conforme ilustrado na Figura 9, antes de as imagens endereçarem a RNSP, estas passam pelo bloco de pré-processamento das imagens. Conforme discutido anteriormente, muitas aplicações de RNAs para problemas de reconhecimento de imagens utilizam de técnicas de pré-processamento da imagem antes que esta seja alimentada à rede. O pré-processamento das imagens é totalmente opcional, e foi utilizado apenas em parte das simulações realizadas na presente dissertação. No caso do presente trabalho, decidiu-se por utilizar um pré-processamento simples devido ao baixo custo computacional e facilidade de implementação. A descrição detalhada da técnica de preprocessamento utilizada é abordada na Seção 4.1.1.

Após o pré-processamento, as imagens são separadas em subconjuntos de acordo com as etiquetas (*labels*) das imagens. O conjunto de imagens da base de dados MNIST não apresenta qualquer separação das amostras com relação a sua classe: elas são, provavelmente de modo intencional, misturadas entre si. O bloco de separação tem a finalidade de separar este conjunto de imagens em 10 subconjuntos (tendo em vista as 10 classes), cada um contendo apenas elementos pertencentes a uma determinada classe. Assim, durante o treinamento da RNSP, cada camada paralela da rede receberá apenas os elementos do subconjunto de treinamento pertencentes à classe à qual a camada deve ser treinada para reconhecer. Este bloco é válido para as redes WiSARD e Mod-MPLN.

O próximo bloco consiste na formação de tuplas a partir do conjunto de entrada, as quais serão utilizadas para endereçar o estágio de entrada da rede. Para o caso de uma RNSP com arquitetura em pirâmide, a configuração das tuplas para os estágios seguintes

também é decidida por este bloco, com base na configuração do estágio anterior a este. O tamanho da tupla possui relação direta com o número de neurônios de cada estágio na rede, conforme mostrado na Equação (7). Pode-se entender este bloco como parte do controlador da rede, mas este foi mantido separado para fins didáticos.

O bloco correspondente ao controlador da rede é responsável pelo acesso à memória RAM da RNSP, ou seja, ao conteúdo das memórias dos neurônios, onde todo conhecimento adquirido pela rede é armazenado. Durante o treinamento da rede, o controlador é responsável por extrair, a partir de um determinado elemento do conjunto de dados de entrada, os bits que formarão a tupla de bits que irá endereçar cada um dos neurônios do estágio de entrada da rede. Além disso, o controlador recebe os bits correspondentes à saída de cada um dos neurônios da rede, de modo a extrair as novas tuplas de bits que irão endereçar os estágios subsequentes. Adicionalmente, o controlador recebe o sinal correspondente ao erro de classificação da rede durante o treinamento, o qual é utilizado para modificar o conteúdo da memória dos neurônios que foram endereçados pelo referido padrão de treinamento. A modificação (escrita) da memória é realizada de acordo com o algoritmo *reward-pentalty* discutido na Seção 1.5.

O bloco seguinte corresponde à função de ativação da rede. De forma similar às RNAs convencionais, as RNSPs utilizam uma função de ativação para trabalhar a saída da rede. No presente trabalho, a rede WiSARD utiliza uma função degrau, enquanto as redes PLN, MPLN e Mod-MPLN utilizam uma função probabilística.

Por fim, o último bloco corresponde ao bloco de cálculo do erro. Este bloco é responsável por calcular o erro entre a saída produzida pela rede para uma amostra de treinamento determinada e a saída desejada para a mesma amostra de treinamento. O erro da rede é utilizado para determinar o ajuste que será realizado na memória dos neurônios, e também como critério de parada durante a fase de treinamento.

## 4.2 Aplicação na identificação de paridade

No contexto de sistemas digitais e telecomunicações, a paridade refere-se ao número de bits ‘1’ de um determinado número binário. Para assinalar a paridade, é adicionado, no final ou no início de uma sequência binária, um dígito binário de paridade. A paridade é muito utilizada para detectar erros nas transmissões, já que o seu cálculo é extremamente simples. De maneira simples, no momento da transmissão de um sinal, são contabilizados

o números de bits ‘1’ no sinal, e é acrescido um bit de paridade com base na quantidade de bits ‘1’. Quando este sinal for recebido por um receptor, os número de bits ‘1’ são novamente contabilizados e devem coincidir com o bit de paridade do sinal. Caso o número de bits ‘1’ não coincida com o bit de paridade, será identificada a ocorrência de perda de dados durante a transmissão.

Existem dois tipos de código de paridade: a paridade par e a paridade ímpar. Caso o sinal possua paridade par, deseja-se que o número de bits de valor ‘1’ do sinal seja par. Dessa forma, quando o número de bits de valor ‘1’ for ímpar, é adicionado um bit de valor ‘1’ ao início ou final do dado, tornando o número de bits de valor ‘1’ par. Caso o número de bits de valor ‘1’ seja par, então o bit de paridade adicionado terá valor ‘0’. Por outro lado, caso o sinal possua paridade ímpar, deseja-se que o número de bits de valor ‘1’ do sinal seja ímpar. Neste caso, quando o número de bits de valor ‘1’ for par, é adicionado um bit de valor ‘1’ ao início ou final do dado, tornando o número de bits ímpar. Caso o número de bits de valor ‘1’ seja ímpar, então o bit de paridade adicionado terá valor ‘0’. O presente trabalho utilizou a configuração de paridade par.

O problema da paridade de $n$ bits é muito frequente em sistemas digitais. Circuitos que utilizam paridade de $n$ bits são muito utilizados para detecção e correção de erros. O problema de classificação do bits de paridade em sequências de $n$ bits é considerado um problema de difícil aprendizado (*hard learning problem*), uma vez que nem todos os sistemas digitais são capazes de implementá-lo (WILAMOWSKI; HUNTER; MALINOWSKI, 2003). No problema da paridade, a variação de um único bits no sinal de entrada irá modificar a saída esperada para a entrada.

Em (BRAGA; CARVALHO; LUDERMIR, 2000), é explicado que a rede WiSARD não é capaz de solucionar o problema da paridade de $n$ bits, enquanto a rede MPLN seria capaz de solucionar este tipo de problema. Dessa forma, o problema da paridade de n bits é selecionado como a segunda aplicação do presente trabalho, de modo a comprovar as afirmações de (BRAGA; CARVALHO; LUDERMIR, 2000), e ainda propor diretrizes de projeto para uma RNSP com objetivo de solucionar o problema de paridade de $n$ bits.

## 4.3 O discriminador da rede MPLN

Conforme foi discutido na Seção 1.3, o discriminador da rede WiSARD é utilizado para determinar se um padrão desconhecido pertence à classe da rede, através da soma das

respostas dos neurônios endereçados pelo padrão desconhecido. Para o caso de um multi-discriminador, a resposta de cada camada paralela é comparada, de modo que a rede identifique o padrão desconhecido como pertencendo à classe da camada com maior número de neurônios que identificaram uma tupla do padrão. Para o caso da rede MPLN, a arquitetura em pirâmide não estabelece uma relação entre a saída das camadas paralelas da rede. Assim, cada camada irá identificar o padrão desconhecido como pertencente à sua classe de forma independente.

Dessa forma, um método intuitivo para descobrir a qual classe o padrão pertence é observar quais camadas paralelas apresentaram saída com valor lógico '1', o que representa que a camada identificou o padrão. Entretanto, como os neurônios da rede MPLN apresentam comportamento probabilístico, é possível que mais de uma camada paralela apresente saída com nível lógico '1' para um determinado padrão desconhecido. Isto ocorre porque o padrão desconhecido pode acessar uma posição de memória do neurônio de saída com valor de probabilidade diferente de zero, mesmo que este seja um valor baixo e tolerável para a convergência da rede durante o treinamento. Caso mais de uma camada paralela identifique o padrão desconhecido, não será possível determinar a qual classe o padrão pertence, o que diminui a eficácia da rede.

Conforme visto na Seção 3.2, uma solução possível para este problema é selecionar, dentre todas as camadas paralelas que emitiram uma saída de nível lógico '1', aquela que apresenta maior probabilidade de emitir uma saída com nível lógico '1'. Dessa forma, é possível eliminar a ambiguidade da hipótese onde uma camada que possuía uma probabilidade muito baixa de produzir uma saída de nível lógico '1' apresentar saída de nível lógico '1'. Entretanto, caso o erro ocorra em uma camada intermediária da rede, é possível que um neurônio da camada de saída com alta probabilidade de produzir um saída de nível lógico '1' seja acessado erroneamente e este seria possivelmente selecionado pelo discriminador, o que levaria ao erro da rede.

Dessa forma, é proposto um novo discriminador para a rede MPLN tradicional, o qual consiste no mesmo discriminador da rede Mod-MPLN. Conforme visto na Seção 3.3, o discriminador da rede Mod-MPLN contabiliza o grau de certeza de cada neurônio acessado pelo padrão a ser reconhecido. O grau de certeza de cada neurônio consiste em um número que irá variar de acordo com os possíveis valores de probabilidade que podem ser expressos por um neurônio e que corresponde ao quão precisa foi a resposta da rede.

O cálculo do grau de certeza para uma rede de neurônios com posições de memória com 10 valores de probabilidade é definido na Equação 9.

A grande vantagem de contabilizar o grau de certeza de todos os neurônios acessados, e não apenas do neurônios de saída, consiste no fato de que o discriminador utiliza toda a informação aprendida pela rede ao decidir a qual classe um determinado padrão pertence. Como mais informação é utilizada, um erro em um único neurônio não afetaria muito a determinação da classe da rede, uma vez que todos os outros neurônios conseguiriam contrabalancear este erro. Desse modo, entende-se que o novo discriminador proposto é mais robusto e tolerante a falhas do que os métodos conhecidos da literatura.

## 4.4 As arquiteturas implementadas

Um dos objetivos do presente trabalho é analisar o impacto da variação dos parâmetros em uma RNSP do tipo MPLN. Para que isto seja possível, é preciso implementar diversas arquiteturas de RNSP, tanto para a aplicação de reconhecimento de caracteres manuscritos quanto na classificação de testes de paridade. Isto se deve pois parâmetros como o tamanho da tupla e número de estágios da rede são dependentes do tamanho de bits do conjunto de entrada.

### 4.4.1 Arquiteturas implementadas para o reconhecimento de imagens

Para que diversas configurações de redes possam ser testadas e analisadas, diferentes conjuntos de entrada foram criados a partir do conjunto de dados MNIST. As imagens do conjunto MNIST foram redimensionadas de modo a obter diferentes tamanhos de imagem, e assim formar os novos grupos. Os tamanhos de imagem utilizados no presente trabalho são 16x16 pixels/bits, 24x24 pixels/bits e 32x32 pixels/bits. Os tamanhos específicos foram escolhidos uma vez que possuem diversos múltiplos, facilitando a formação de diversas arquiteturas de RNSP. Além disso, yais tamanhos foram selecionados de modo a possibilitar uma análise com relação à escalabilidade da rede MPLN. Para cada um dos três conjuntos, foram ainda gerados conjuntos utilizando o pré-processamento descrito na Seção 4.1.1, totalizando seis conjuntos de dados utilizados.

A primeira imagem testada possui 16 x 16 *pixels*, totalizando 256 *pixels*, os quais serão tratados como bits para endereçar o estágio de entrada da RNSP. Para esta imagem, foram propostas três arquiteturas: a primeira, denominada Arq1, mantém uma tupla fixa

de 2 bits, e possui 128 neurônios no primeiro estágio, 64 neurônios no segundo estágio, 32 neurônios no terceiro estágio, 16 neurônios no quarto estágio, 8 neurônios no quinto estágio, 4 neurônios no sexto estágio, 2 neurônios no sexto estágio e 1 único neurônio no estágio de saída. A segunda arquitetura, denominada Arq2, mantém uma tupla fixa de 4 bits, e possui 64 neurônios no primeiro estágio, 16 neurônios no segundo estágio, 4 neurônios no terceiro estágio, e 1 único neurônio no estágio de saída. A terceira arquitetura proposta para uma imagem 16x16, denominada Arq3, possui uma tupla variável de 8 e 4 bits. No primeiro estágio, a rede possui 32 neurônios com uma tupla de 8 bits, os quais endereçam o segundo estágio com 4 neurônios e uma tupla de 8 bits. Para o estágio de saída, a rede possui uma tupla de apenas 4 bits, contrariando ao que ensina Aleksander em (ALEKSANDER; MORTON, 1989).

Como o tamanho em bits da tupla e o número de estágios da rede estão diretamente relacionados ao tamanho em bits da entrada, o presente trabalho analisou também aplicações com imagens de algarismos com resolução de 24x24 *pixels* e de 32x32 *pixels*. Para as imagens de algarismos com resolução de 24x24 *pixels*, foram desenvolvidas cinco arquiteturas, enquanto que foram desenvolvidas três arquiteturas para as imagens de algarismos com resolução de 32x24 *pixels*. A Tabela 3 abaixo apresenta a configuração de números de estágios, números de neurônios por estágio e tamanho de tupla por estágio para cada uma das dez arquiteturas de rede simuladas no presente projeto.

Tabela 3: Arquiteturas propostas para o reconhecimento de caracteres - células indicando o número de neurônios / tamanho da tupla

| | | **Estágios** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | **Arq.** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| **16** | **Arq1** | 128/2 | 64/2 | 32/2 | 16/2 | 8/2 | 4/2 | 2/2 | 1/2 | - | - |
| **x** | **Arq2** | 64/4 | 16/4 | 4/4 | 1/4 | - | - | - | - | - | - |
| **16** | **Arq3** | 32/8 | 4/8 | 1/4 | - | - | - | - | - | - | - |
| **24** | **Arq4** | 192/3 | 64/3 | 32/2 | 16/2 | 8/2 | 4/2 | 2/2 | 1/2 | - | - |
| **x** | **Arq5** | 144/4 | 36/4 | 9/4 | 3/3 | 1/3 | - | - | - | - | - |
| **24** | **Arq6** | 96/6 | 16/6 | 4/4 | 1/4 | - | - | - | - | - | - |
| | **Arq7** | 72/9 | 8/9 | 8/8 | - | - | - | - | - | - | - |
| **32** | **Arq8** | 512/2 | 256/2 | 128/2 | 64/2 | 32/2 | 16/2 | 8/2 | 4/2 | 2/2 | 1/2 |
| **x** | **Arq9** | 256/4 | 64/4 | 16/4 | 4/4 | 1/4 | - | - | - | - | - |
| **32** | **Arq10** | 128/8 | 16/8 | 4/4 | 1/4 | - | - | - | - | - | - |

Nota-se os parâmetro números de estágios, números de neurônios por estágio e tamanho de tupla por estágio foram selecionados de forma a implementar as dez arquite-

turas com diferentes configurações entre, de modo a possibilitar a análise do impacto de cada parâmetro na eficiência e eficácia da rede.

### 4.4.2 Arquiteturas implementadas para a paridade de N bits

Para o caso da paridade de $n$ bits, os conjuntos de dados foram criados através de um *script* específico do Matlab, o qual é configurado para gerar todas as cominações possíveis de sequências de $n$ bits, com a sua correspondente saída desejada. Estes conjuntos de dados foram separados em dois conjuntos de igual tamanho: o conjunto de treinamento e o conjunto de teste. A divisão das amostras entre os dois conjuntos foi realizada de forma aleatória.

Os conjuntos de dados gerados para o problema da paridade de N bits consistem no grupo de dados para a paridade de 6, 8, 10 e 12 bits. Para cada um dos tamanhos de sequência, foram desenvolvidos conjuntos de dados para todas as possibilidades de combinação de bits. Cada conjunto de dados possui $2^n$ amostras, onde $n$ é o número de bits da sequência. Como o problema da paridade possui apenas duas saídas possíveis, ou seja, determinar qual o valor lógico '0' ou '1' do bit a ser incorporado na sequência, a RNSP implementada possui apenas uma camada. Neste caso, a saída da rede corresponde ao valor lógico do bit de paridade que deve ser incluído na sequência a ser transmitido. A Tabela 4 especifica a configuração das 10 arquiteturas de rede simuladas no presente projeto.

Tabela 4: Arquiteturas propostas para o problema da paridade: número de neurônios/tamanho da tupla

| | | **Estágios** | | | |
|---|---|---|---|---|---|
| **Paridade** | **Arq.** | **1** | **2** | **3** | **4** |
| **6** | **1** | 3/2 | 1/3 | - | - |
| | **2** | 2/3 | 1/2 | - | - |
| **8** | **3** | 4/2 | 1/4 | - | - |
| | **4** | 2/4 | 1/2 | - | - |
| | **5** | 4/2 | 2/2 | 1/2 | - |
| **10** | **6** | 5/2 | 1/5 | - | - |
| | **7** | 2/5 | 1/2 | - | - |
| **12** | **8** | 6/2 | 1/6 | - | - |
| | **9** | 2/6 | 1/2 | - | - |
| | **10** | 4/3 | 2/2 | 1/2 | - |

## 4.5 Inserção de ruído no conjunto de treinamento

Conforme visto na Seção 3.1.3, um dos parâmetros a ser escolhido no projeto de uma RNSP do tipo MPLN é a resolução na memória $\omega$, ou seja, a quantidade de valores para representar a probabilidade do neurônio emitir uma saída com nível lógico '1'. Uma resolução de memória elevada faz com que seja mais difícil apagar o conhecimento acumulado pela rede, o que torna a mesma mais resistente a amostras ruidosas durante o treinamento. Por outro lado, o aumento da resolução da memória irá causar o aumento do tamanho da memória necessária para implementar a rede, e, consequentemente, maior poder computacional necessário para implementar a mesma. Adicionalmente, o aumento da resolução da memória dos neurônios também aumenta o tempo de convergência da rede durante o treinamento, uma vez que serão necessárias mais épocas de treinamento para alcançar os valores de probabilidade de 100 % e 0 %.

De modo a analisar o impacto da variação da resolução da memória dos neurônios, é necessário inserir ruídos nos conjuntos de treinamento utilizados para treinar as RNSP do tipo MPLN. Assim, é proposto que o ruído inserido nas imagens consista em alterar as saídas desejadas para uma parte do conjunto de treinamento utilizado. Por exemplo, uma amostra correspondente ao algarismo 3 teria sua resposta espera alterada para o valor 5. Neste caso, a rede será treinada de forma errada para esta amostra de treinamento ruidosa. Dessa forma, será possível verificar se valores altos da resolução da memória do neurônios serão eficazes em diminuir o impacto das amostras ruidosas na eficácia da rede.

Para a análise, são utilizadas as imagens 16x16 *pixels*, uma vez que estas demandam menor poder computacional das simulações. São propostos três conjuntos de dados com imagens ruidosas a partir das imagens 16x16 *pixels*. O primeiro conjunto, denominado conjunto 1, consiste no conjunto de imagens 16x16 com 10 % das imagens possuindo valores esperados incorretos, ou seja, 10 % de ruído nas imagens. O segundo conjunto, denominado conjunto 2, consiste no conjunto de imagens 16x16 com 15 % das imagens possuindo valores esperados incorretos. Por fim, o terceiro conjunto, denominado conjunto 3, consiste no conjunto de imagens 16x16 com 20 % das imagens possuindo valores esperados incorretos.

Com relação aos conjuntos de dados para a aplicação da classificação do bit de paridade, os conjuntos também foram modificados para incluir ruído nos mesmos. A partir de simulações preliminares, notou-se que a rede MPLN apresentou resultados considera-

velmente ruins para amostras com ruído de 10 % e 15 %. Dessa forma, ao invés de utilizar também amostras com 20 % de ruído, decidiu-se utilizar amostras com 5 %, de modo a observar de forma mais gradual o impacto do ruído e sua realização com o parâmetro $\omega$.

## 4.6 Considerações Finais do Capítulo

Neste capítulo foram apresentados os conceitos particulares à implementação do presente trabalho. Foram discutidos os conjuntos de dados utilizados e como estes foram manipulado de forma a possibilitar uma melhor análise do desempenho das RNSP, bem como o impacto da variação dos parâmetro das redes. Em seguida, foi apresentado o diagrama de como as redes foram implementadas, ou seja, os elementos capazes de realizar os modelos matemáticos refentes às RNSPs. Por fim, foram apresentadas as arquiteturas de RNSP que serão implementadas. No capítulo seguinte são apresentados os resultados de simulação das arquiteturas implementadas.

# Capítulo 5

# ANÁLISE DOS RESULTADOS

NESTE capítulo, são apresentados e analisados os resultados alcançados para as RNSPs do tipo MPLN e Mod-MPLN descritas nos Capítulos 1 e 3, com base nos aspectos de implementação abordados no Capítulo 4. A avaliação do método proposto é feita por meio de simulações conduzidas no MATLAB®.

Conforme visto no Capítulo 4, para cada tamanho de imagem ou tamanho de sequência de bits, diferentes arquiteturas foram desenvolvidas. Além disso, para o caso das imagens de algarismos manuscritos, as amostras do conjunto de imagem foram divididas em três grupos com base na qualidade das imagens. Como as redes MPLN e Mod-MPLM são rede probabilísticas, cada simulação obtida irá apresentar resultados diferentes um dos outros. Assim, todas as simulações das redes MPLN e Mod MPLN foram realizadas 50 vezes, extraindo a média e o desvio padrão das mesmas. As subseções a seguir irão apresentar os resultados obtidos para cada uma das configurações de rede. Na Seção 5.1, são apresentados os resultados para a aplicação proposta na classificação do bit de paridade em sinais de $n$ bits, assim como é analisado o impacto da variação do tamanho da tupla, número de estágios e resolução da memória na eficiência e eficácia da rede. Analogamente, na Seção 5.2, são discutidas as particularidades da aplicação no reconhecimento de algarismos manuscritos, onde comprova-se a ineficiência da rede MPLN em problemas de classificação com múltiplas classes. Ainda na Seção 5.2, são apresentados os resultados obtidos para a rede Mod-MPLN, os quais são comparados com a rede WiSARD, e também verifica-se o impacto da variação dos parâmetros de projeto na rede Mod-MPLN.

As simulações foram realizadas utilizando cinco computadores, o primeiro consistindo em um Windows® 10, Intel(R) Core(TM) i5-3230M CPU @2.60 GHz quad-core,

com 8 GB de memória RAM e placa de vídeo dedicada com 4 GB de memória RAM, utilizando o software Matlab® R2016a. Os demais computadores utilizados possuem configuração de *hardware* similar, com sistemas operacionais Windows®, processadores de múltiplos núcleos, e 8 GB de memória RAM. Dessa forma, espera-se que ocorra certa variação no tempo de processamento das simulações realizadas em computadores diferentes. Entretanto, devido à similaridade das configurações, entende-se que estas pequenas variações não irão afetar as análises realizadas no presente trabalho.

## 5.1 Problema da paridade

A primeira aplicação abordado é o problema de atribuição e verificação do bit de paridade de sequências de bits, o qual foi brevemente explicado na Seção 4.2. Adicionalmente, foi visto na Seção 4.4.2 que, para o problema da paridade de $n$ bits, foram desenvolvidos 4 conjuntos de dados: conjunto de sequências de 6, 8, 10 e 12 bits. Tais conjuntos de dados foram criados a partir de um *script* do Matlab. O tamanho de cada conjunto de dados é dependente do tamanho de bits do sinal, uma vez que $n$ bits permite $2^n$ variações de sinal.

Para cada um dos conjuntos de imagem, diversas arquiteturas desenvolvidas foram desenvolvidas, conforme ilustrado na Tabela 4. A seguir, será analisado o impacto da variação dos parâmetros da rede MPLN no problema da paridade, de modo auxiliar o projetista na escolha dos mesmos na hora de implementar uma RNSP desta topologia.

### 5.1.1 Impacto da variação da tupla e número de estágios

Para analisar os impactos da variação de parâmetros na rede MPLN, alguns parâmetros da rede foram mantidas constantes, variando apenas a arquitetura das redes. Conforme ilustrado na Tabela 4, cada arquitetura da rede representa um configuração de números de estágios, neurônios em cada estágio, e o tamanho de tupla dos neurônios em cada estágio. A Tabela 5 ilustra os parâmetros mantidos constantes na simulação.

Conforme visto na Seção 1.5, o parâmetro $\omega$ representa a resolução da memória RAM dos neurônios, o parâmetro $\beta$ representa a quantidade de vezes que uma amostra é reapresentada a rede antes de se determinar um conflito de aprendizado, e o parâmetro $\eta$ representa a taxa de aprendizagem da rede. Para a análise do tamanho da tupla, o parâmetro $\omega$ foi mantido constante com valor 11, conforme proposto em (MYERS, 1989). O

Tabela 5: Parâmetros mantidos constantes para a análise do impacto da tupla

| **Parâmetro** | **Valor** |
|---|---|
| $\omega$ | 11 |
| $\beta$ | 4 |
| $\eta$ | 0,1 |
| $e_1$ | 0,9 |
| $e_2$ | 1000 |

parâmetro $\omega$ foi mantido em 4 de modo a prover um treinamento estável na rede, mas sem comprometer muito o tempo de treinamento da mesma. O parâmetro $n$, correspondente ao parâmetro $\omega$, conforme visto na Seção 3.1.4, foi mantido com valor 0,1. Os parâmetros $e_1$ e $e_2$ correspondem às duas condições de parada utilizadas no treinamento, sendo que $e_1$ corresponde à acurácia obtida durante o treinamento, e $e_2$ corresponde ao número máximo de épocas para a fase de treinamento.

É importante notar que as arquiteturas de rede são adaptadas para diferentes tamanhos de bits na classificação do bit de paridade. Nesse sentido, conforme visto na Seção 4.4.2, cada conjunto de dados possui $2^n$ amostras igualmente distribuídas em conjunto de treinamento e conjunto de teste. Dessa forma, para analisar os conjuntos de dados conjuntamente, foram propostos alguns valores de $k$ amostras, conforme a Tabela 6.

Tabela 6: Grupos de amostras formados a partir das amostras de treinamento

| **Paridade** | **Arq.** | **# amostras treinamento** | **# amostras em G1** | **# amostras em G2** | **# amostra em G3** |
|---|---|---|---|---|---|
| 6 bits | Arq1 | 32 | 10 | 20 | 32 |
| | Arq2 | 32 | 10 | 20 | 32 |
| 8 bits | Arq3 | 128 | 10 | 50 | 100 |
| | Arq4 | 128 | 10 | 50 | 100 |
| | Arq5 | 128 | 10 | 50 | 100 |
| 10 bits | Arq6 | 512 | 10 | 100 | 1000 |
| | Arq7 | 512 | 10 | 100 | 1000 |
| 12 bits | Arq8 | 2048 | 10 | 100 | 1000 |
| | Arq9 | 2048 | 10 | 100 | 1000 |
| | Arq10 | 2048 | 10 | 100 | 1000 |

Dessa forma, o grupo G1 representa uma situação onde estão disponíveis poucas amostras de treinamento, o grupo G2 representa uma situação onde estão disponíveis razoáveis amostras de treinamento, e o grupo G3 representa a situação onde estão disponíveis diversas amostras de treinamento para treinar a rede. Separar os conjuntos de

treinamento é necessário para possibilitar que todos sejam analisados sob a mesma perspectiva, apesar destes possuírem tamanhos distintos.

Ao longo das simulações, foi possível observar que, durante o treinamento das RNSP para a classificação do bit de paridade, estas acabavam convergindo para mínimos locais. De forma a aliviar isto, foi proposto um critério de parada adicional através da inclusão de um contador de épocas. Quando o contador de épocas da rede atinge um valor limite de 100 épocas, a memória dos neurônios acessada é modificada para seu valor inicial 0,5. Dessa forma, a memória dos neurônios poderia se reorganizar a partir do valor intermediário '$u$' durante o treinamento.

A Figura 10 mostra o desempenho das 10 arquiteturas testadas em termos da acurácia durante a fase de teste. Os valores representam a média e o desvio padrão de 50 simulações para cada uma das arquiteturas, utilizando o três grupos G1, G2 e G3 das amostras de treinamento. A configuração do tamanho da tupla por estágio para cada arquitetura está definida para cada caso.

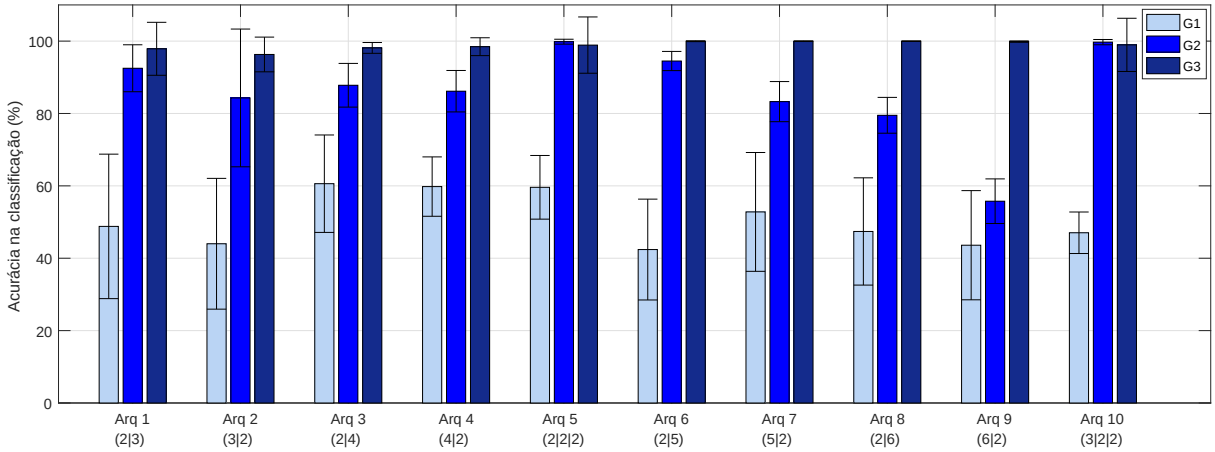



Figura 10: Acurácia obtida pelas arquiteturas para os grupos de amostras de treinamento

Primeiramente, é importante notar que, para o grupo e poucas amostras G1, todas as redes obtiveram acurácia na faixa, de 40% a 60%, com alto desvio padrão. Nesse sentido, nota-se que o problema da paridade apresenta apenas 2 classes, e o completo desconhecimento da solução representa uma probabilidade de 50% de classificar o bit de paridade. Dessa forma, nenhuma das arquiteturas foi capaz de classificar corretamente o bit de paridade a ser incluído na sequência de bits.

Para o segundo grupo de amostras G2, todas as arquiteturas obtiveram acurácia acima de 80%, com exceção da Arq9, a qual é destinada a classificar o bit de paridade

em sinais de 12 bits. É interessante observar que as arquiteturas com menores valores de tupla no estágio de entrada obtiveram os melhores resultados no grupo G2 de amostras. Especificamente, para o prolema da paridade de 6 bits, classificado pelas arquiteturas Arq1 e Arq2, a primeira obteve melhor desempenho. Para o problema da paridade de 8 bits, classificado pelas arquiteturas Arq3, Arq4 e Arq5, a arquitetura com tupla fixa de 2 bits em 3 estágios (Arq5) obteve resultados consideravelmente superiores. Já para o problema da paridade de 10 bits, a arquitetura 6 obteve melhores resultados que a Arq7. Por fim, para o problema da paridade de 12 bits, a arquitetura 10, com 3 estágios, obteve resultados consideravelmente superiores que a Arq8. A arquitetura Arq9, conforme mencionado, não foi capaz de classificar o bit de paridade de forma satisfatória.

Para o grupo de amostras G3, todas as arquiteturas obtiveram acurácia acima de 95%. Assim, para o grupo com muitas amostras de treinamento, todas as arquiteturas foram capazes de classificar o bit de paridade de forma satisfatória. Tais resultados vão de encontro com (GARCIA, 2003; STONHAM, 1986), onde é descrito que um tamanho de tupla menor é mais indicado no caso onde exitem poucas amostras de treinamento. Para os casos onde existem poucas amostras, por exemplo os grupos G1 e G2, as arquiteturas com menor tamanho de tupla obtiveram os melhores resultados.

A seguir, será analisado o impacto do tamanho de tuplas nas épocas de treinamento necessárias para que o treinamento da rede seja concluído. A Figura 11 ilustra o número médio de épocas de treinamento necessárias para cada arquitetura completar a fase de treinamento.

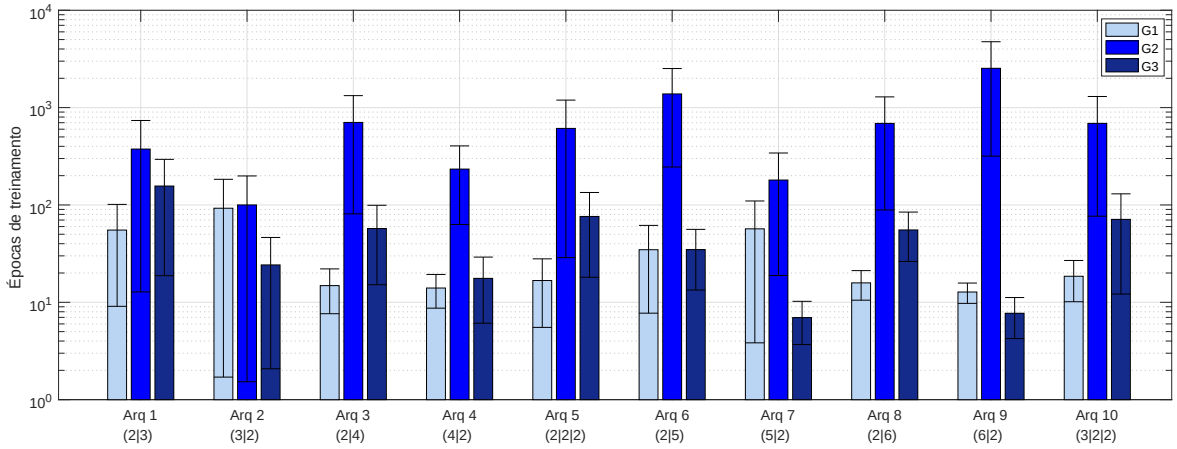



Figura 11: Total de épocas utilizadas pelas arquiteturas durante o treinamento da rede para os grupos de amostras de treinamento

Em um primeiro olhar na Figura 11, é possível observar que os valores de desvio padrão são muito elevados para quase todas as arquiteturas. Isto se deve ao fato de que as redes ocasionalmente não convergiam nas 1000 épocas de treinamento. Consequentemente, com o auxilio do contador de épocas, a memória dos neurônios acessados era retornada ao seu valor 0.5. Entretanto, as épocas totais do treinamento foram contabilizadas em sua totalidade. Assim, é possível observar que o número de épocas utilizado foi muito variável durante o treinamento, o que mostra que todas as redes são consideravelmente suscetíveis a não convergir devido a mínimos locais. A alta variação na velocidade de convergência da rede pode ainda ser atribuída ao fato das tuplas serem formadas aleatoriamente a partir dos bits de entrada e pelo fato de as amostras serem ordenadas de forma aleatória a cada simulação.

Voltando-se à Figura 11, é possível observar que o grupo de poucas amostras G1 obteve um tempo de simulação próximo ao grupo de muitas amostras G3, enquanto o grupo de amostras G2 levou muito mais tempo para convergir do que os demais. É importante lembrar que, conforme visto na Figura 10, nenhuma das arquiteturas foi capaz de classificar corretamente o bit de paridade. Dessa forma, o número de épocas reduzido se deve apenas ao fato de que existem menos amostras a serem apresentadas à rede. Com relação ao grupo de amostras G2, os resultados se mostram interessantes uma vez que seria intuitivo esperar um número de épocas de treinamento inferior ao grupo G3, dado que o grupo possui menos amostras a serem apresentadas às redes. Dessa forma, concluiu-se que o grupo G2 não possui amostras suficientes para que a rede aprenda o problema de forma suficiente. Nestes casos, a rede precisou ser reiniciada diversas vezes de acordo com o contador de épocas, aumentando as épocas necessárias para o fim do treinamento. No entanto, apesar da instabilidade durante o treinamento, as arquiteturas foram capazes de obter resultados satisfatórios após a convergência do treinamento, conforme mostrado na Figura 10.

Para o caso da paridade de 6 bits, a qual é classificada pelas Arq1 e Arq2, nota-se que a tupla de maior tamanho no estágio de entrada consome, em média, menos épocas de treinamento que a arquitetura de menor tupla. Entretanto, tal observação se torna de baixa confiabilidade devido ao alto desvio padrão para estas arquiteturas. Com relação à paridade de 8 bits, novamente, os neurônios de maior tupla no estágio de entrada utilizaram menos épocas de treinamento. O mesmo ocorre para o caso da paridade de

10 e 12 bits, onde as arquiteturas Arq7 e Arq9 utilizaram menos épocas de treinamento, respectivamente.

Assim, pode-se concluir que, para problemas de difícil aprendizado, tal como o problema de classificação do bit de paridade, uma tupla de maior tamanho no estágio de entrada torna a rede com maior capacidade de aprendizado do problema. Isto é corroborado em (GARCIA, 2003), onde é dito que tamanhos de tupla mais elevados são mais indicados no caso onde existem diversas amostras de treinamento. Apesar de (ALEKSANDER; MORTON, 1989) estabelecer uma relação entre o tamanho da tupla e o aumento do consumo da memória, existem casos onde um tupla de maior tamanho maior irá fazer a rede convergir mais rápido. Dessa forma, nestes casos, incluindo o problema da classificação da paridade, um tamanho maior de tupla aumenta a velocidade de convergência da rede.

A Figura 12 ilustra o tempo médio gasto por cada arquitetura para completar a fase de treinamento e teste.

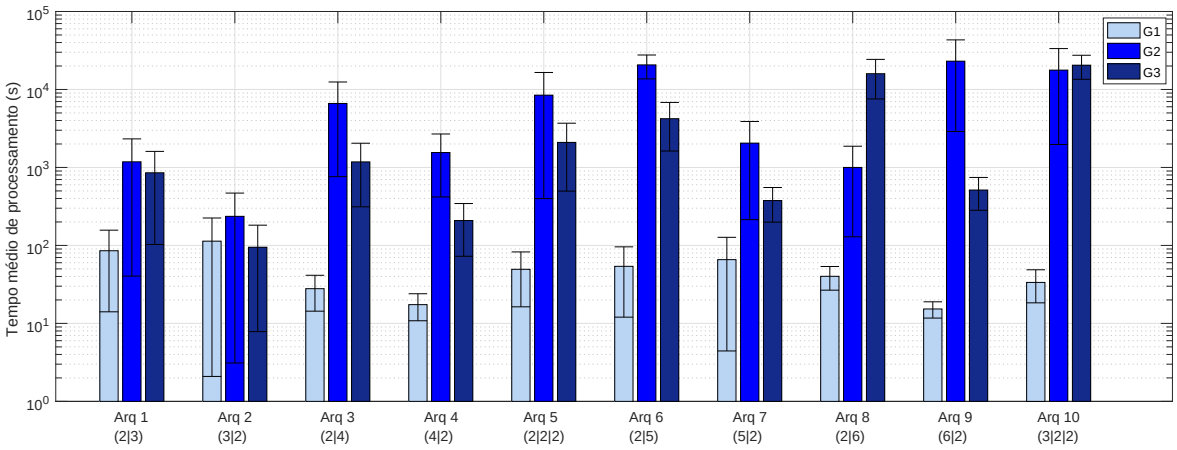



Figura 12: Tempo médio necessário para cada arquitetura finalizar o treinamento e teste utilizando os grupos de amostras de treinamento

Como era esperado, o gráfico de tempo acompanha o gráfico de épocas da 11, uma vez que quanto mais épocas de treinamento forem utilizadas, maior o tempo consumido pela rede durante o treinamento da mesma. Dessa forma, a mesma variação do número de épocas pode ser observada com o alto desvio padrão, em especial para o caso da paridade de 6 bits.

A principal diferença entre o consumo de épocas de treinamento e o consumo de tempo pode ser observada no grupo de muitas amostras G3, uma vez que o tempo de

processamento obviamente aumenta com o número de amostras. Adicionalmente, como o tempo de processamento computa tanto a fase de treinamento quanto a fase de teste, o tempo tende a aumentar com o número de amostras.

Quanto ao impacto da tupla no tempo de processamento, os resultados são condizentes com os resultados de épocas de treinamento da Figura 11. As arquiteturas de tamanho de tupla maior no estágio de entrada apresentaram tempos de processamentos menores. Uma análise destes resultados leva à recomendação de utilizar tuplas maiores no estágio de entrada da rede quando existem suficientes amostras de treinamento. Tal recomendação seria justificada pelo fato de que o estágio de entrada é aquele que recebe diretamente os bits dos padrões de entrada, enquanto os estágios subsequentes são endereçados pelas saídas dos estágios anteriores, identificando sub-combinações do padrão de entrada (BANDEIRA, 2010). Assim, pode-se concluir que o estágio de entrada é aquele que precisa possuir maior capacidade de aprendizado, uma vez que é diretamente endereçado pelos padrões de entrada.

A partir da análise das 10 arquiteturas, configuradas para solucionar o problema de classificação do bit de paridade em sequências de 6, 8, 10 e 12 bits, pode-se observar que as arquiteturas com maior tamanho de tupla obtiveram melhores resultados. Para o caso do grupo de amostras G2, as arquiteturas com menor tamanho de tupla obtiveram acurácia consideravelmente superior, apesar da instabilidade no treinamento. Entretanto, para o grupo de amostras G3, as arquiteturas de maior tupla no estágio de entrada apresentaram não apenas acurácia igual ou superior às demais arquiteturas, mas também possibilitam um treinamento consideravelmente mais estável. Consequentemente, as arquiteturas de maior tupla convergiram em menor tempo e número de épocas.

### 5.1.2 Impacto da resolução da memória na classificação do bit de paridade

Para analisar o impacto da variação da resolução da memória dos neurônios da rede, é preciso variar o parâmetro $\omega$. Conforme visto na Seção 3.1.3, o aumento de $\omega$ torna mais difícil de uma posição de memória do neurônio retornar para a posição 'u'. Dessa forma, a rede se torna mais resiliente a ruídos, uma vez que uma única amostra ruidosa não será capaz de eliminar completamente o conhecimento adquirido pelo neurônio a respeito do problema a ser solucionado. Por outro lado, o aumento de $\omega$ aumenta o consumo de

memória da rede, uma vez que serão necessários mais bits para representar cada posição de memória dos neurônios.

Conforme conhecido da literatura (MYERS, 1989), o principal interesse em utilizar um valor de $\omega$ elevado reside na tolerância a ruídos na amostra de treinamento. Dessa forma, os conjuntos de sequências de bits foram modificados para inserir ruídos nos mesmos, conforme explicado na Seção 4.5. As simulações realizadas para aferir o impacto da variação de $\omega$ foram realizadas para as arquiteturas 1 a 5 da Tabela 4, e para o grupo de treinamento de muitas amostras G3 da Tabela 6. A Tabela 7 ilustra os parâmetros mantidos constantes na simulação.

Tabela 7: Parâmetros constantes durante a variação da resolução da memória $\omega$

| **Parâmetro** | **Valor** |
|---|---|
| $k$ | G3 |
| $\beta$ | 4 |
| $\eta$ | 0,1 |
| $e_1$ | 0,9 |
| $e_2$ | 1000 |

As simulações para a variação da resolução da memória $\omega$ também utilizam o contador de épocas, com objetivo de auxiliar a convergência da rede. Novamente, quando o contador de épocas da rede atinge um valor limite de 100 épocas, a memória dos neurônios acessada é modificada para seu valor inicial 0,5. Dessa forma, a memória dos neurônios poderia se reorganizar durante o treinamento.

A Figura 13 mostra o desempenho das arquiteturas testadas em termos da acurácia durante a fase de teste. Os valores representam a média e o desvio padrão de 50 simulações para cada uma das cinco arquiteturas utilizadas na classificação do bit de paridade para sinais de 6 e 8 bits. Ambos os conjuntos de treinamento de 6 e 8 bits foram modificados para incluir 5% de amostras com valor esperado incorreto. As simulações foram realizadas utilizando o grupo G3 das amostras de treinamento.

Conforme pode ser observado, todas as arquiteturas apresentaram resultados satisfatórios para os valores de $\omega$ testados. Para as arquiteturas 1 e 2, responsáveis pela classificação do bit de paridade em sinais de 6 bits, a acurácia obtida pela rede aumenta com o aumento de $\omega$. Tal resultado condiz com os ensinamentos de (MYERS, 1989), os quais afirmam que maiores tamanhos de $\omega$ aumentam a tolerância da rede a ruídos. Para Arq3, Arq4 e Arq5, responsáveis pela classificação do bit de paridade em sequência de 8

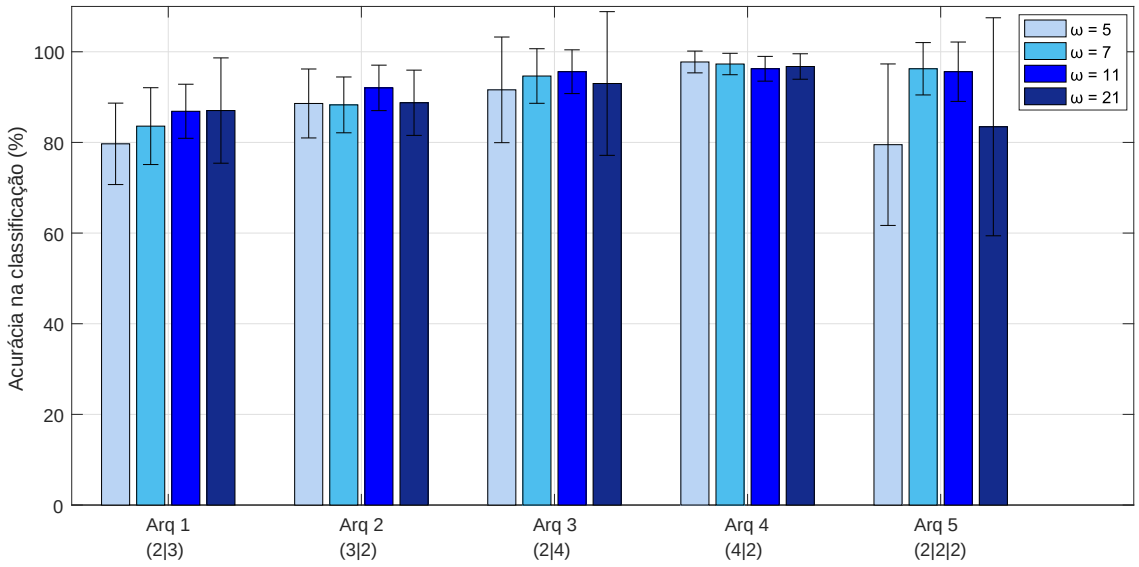



Figura 13: Acurácia obtida pelas arquiteturas com diferentes valores de $\omega$ em amostras com ruído de 5%

bits, nota-se que as arquiteturas com $\omega = 7$ e $\omega = 11$ obtiveram resultados ligeiramente superiores. Dessa forma, observa-se que a acurácia da rede não aumenta indefinidamente com o aumento de $\omega$.

Em geral, as arquiteturas com $\omega = 11$ obtiveram os melhores resultados para a classificação do bit de paridade. Nota-se ainda que as arquiteturas com valor de $\omega = 5$ obtiveram os piores resultados. Dessa forma, pode-se concluir que uma resolução de memória $\omega = 5$ torna a rede mais suscetível a ruídos que as demais.

A Figura 14 mostra o desempenho das arquiteturas testadas em termos do número de épocas de treinamento necessário para concluir o treinamento da rede.

O alto desvio padrão observado nas arquiteturas se deve ao fato de que diversas vezes a rede não convergiu durante as 1000 épocas de treinamento. Para Arq1 e Arq2, nota-se que o número de épocas de treinamento necessárias aumenta com o aumento de $\omega$, onde há uma notável instabilidade pra $\omega = 11$ e $\omega = 21$. Resultados similares podem ser observados para Arq3, Arq4 e Arq5, com a exceção das arquiteturas com $\omega = 5$. As arquiteturas obtiveram os priores resultados em todas as arquiteturas para $\omega = 21$. É importante observar que o aumento das épocas de treinamento com o aumento de $\omega$ é um resultado esperado (MYERS, 1989), uma vez que mais ajustes são necessários até que a rede alcance os valores de probabilidade de 0% e 100%.

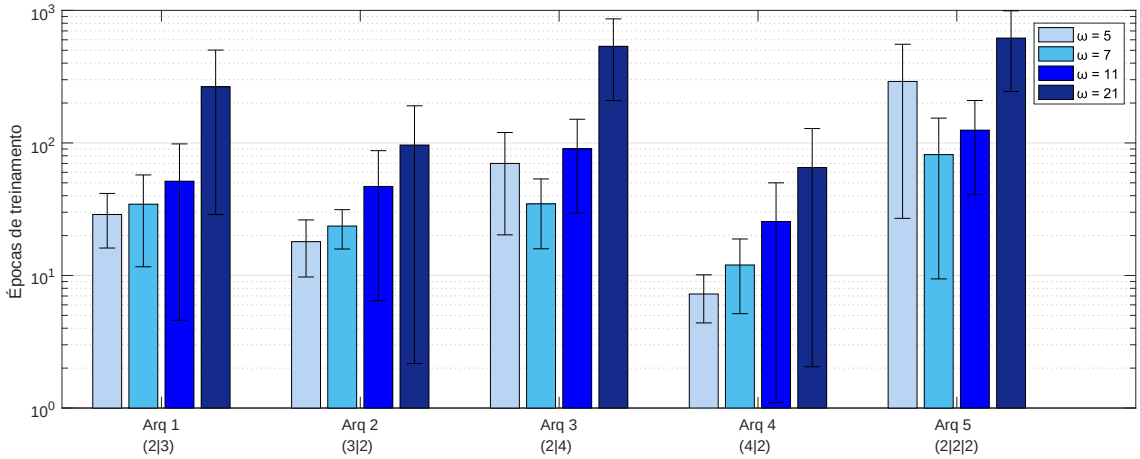



Figura 14: Épocas de treinamento necessárias para concluir o treinamento da rede para s arquiteturas com diferentes valores de $\omega$ em amostras com ruído de 5%

A Figura 15 mostra o desempenho das arquiteturas testadas em termos do número de épocas de treinamento necessário para concluir o treinamento da rede.

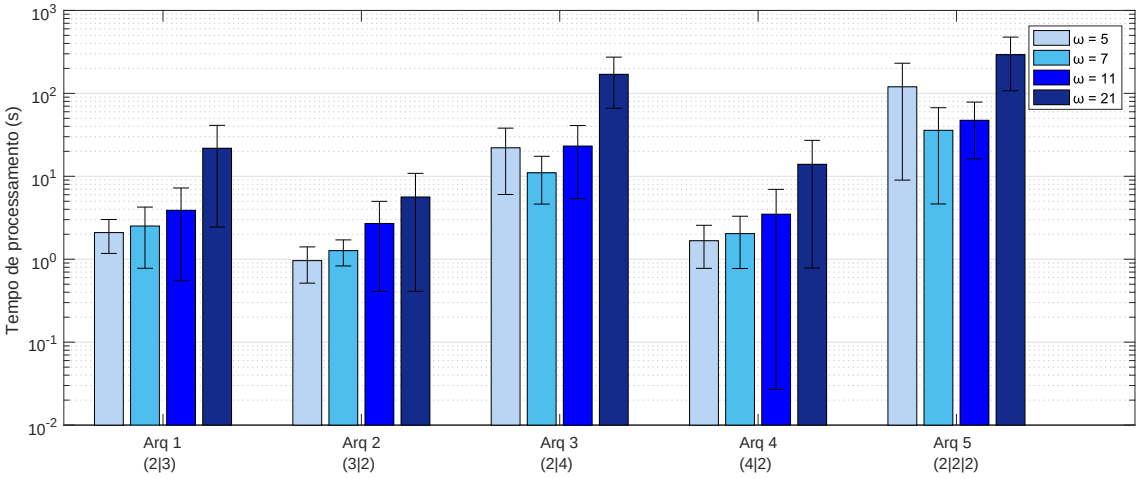



Figura 15: Tempo de processamento necessário para concluir o treinamento e teste da rede para as arquiteturas com diferentes valores de $\omega$ em amostras com ruído de 5%

De modo similar ao ocorrido para a análise das épocas de treinamento, o baixo índice de convergência das arquiteturas prejudica a análise da relação entre o valor de $\omega$ e o tempo de processamento das arquiteturas. Para Arq1 e Arq2, nota-se que o tempo de processamento aumenta com o aumento de $\omega$. Tal valor é esperado, uma vez que uma alta resolução da memória faz que com mais ajustes em seu conteúdo sejam necessários para alcançar os valores de probabilidade 0% e 100%. Analogamente ao número de épocas de treinamento, as arquiteturas obtiveram os piores resultados em todas as arquiteturas

para $\omega = 21$. A Tabela 8 apresenta uma análise comparativa entre os resultados obtidos pelas arquiteturas para diferentes $\omega$ com ruído de 5%.

Tabela 8: Comparativo entre as arquiteturas para diferentes $\omega$ para paridade com ruído de 5%

| | | **Resolução da memória** | | | |
|---|---|---|---|---|---|
| **Arq.** | **Resultado** | $\omega = 5$ | $\omega = 7$ | $\omega = 11$ | $\omega = 21$ |
| **Arq1** | **Acurácia** | pior | bom | melhor | melhor |
| | **Épocas** | melhor | regular | regular | pior |
| | **Tempo** | melhor | bom | bom | pior |
| **Arq2** | **Acurácia** | bom | bom | melhor | bom |
| | **Épocas** | melhor | bom | regular | pior |
| | **Tempo** | melhor | bom | regular | pior |
| **Arq3** | **Acurácia** | regular | bom | melhor | regular |
| | **Épocas** | regular | melhor | regular | pior |
| | **Tempo** | regular | melhor | regular | pior |
| **Arq4** | **Acurácia** | melhor | melhor | bom | bom |
| | **Épocas** | melhor | bom | regular | pior |
| | **Tempo** | melhor | bom | regular | pior |
| **Arq5** | **Acurácia** | pior | melhor | melhor | regular |
| | **Épocas** | regular | melhor | bom | pior |
| | **Tempo** | regular | bom | bom | pior |

Dessa forma, nota-se que as arquiteturas com $\omega = 7$ e $\omega = 11$ obtiveram os melhores valores de acurácia durante a classificação, enquanto as arquiteturas com $\omega = 7$ obtiveram em geral os melhores resultados em termos de número de épocas de treinamento e tempo de processamento. Assim, pode-se concluir que para um conjunto de treinamento com 5% de ruído, o valor mais indicado para a resolução da memória seria $\omega = 7$. Apesar de valores de $\omega$ maiores aumentarem a tolerância da rede a ruídos (MYERS, 1989), entende-se que, para apenas 5% de ruído, $\omega = 7$ já proporciona tolerância suficiente. Os valores maiores de $\omega$ apresentam pouca melhoria em termos de acurácia quando comparados ao aumento no número de épocas e tempo de processamento consumidos.

De modo a avaliar o impacto do parâmetro $\omega$ em conjuntos de treinamento com maior ruído, foi inserido ruído de 10% nos conjuntos de dados de algarismos manuscritos. A Figura 16 mostra o desempenho das arquiteturas testadas em termos da acurácia durante a fase de teste. Ambos os conjuntos de treinamento de 6 e 8 bits foram modificados para incluir 10% amostras com valor esperado incorreto. As simulações foram realizadas utilizando o grupo G3 das amostras de treinamento.

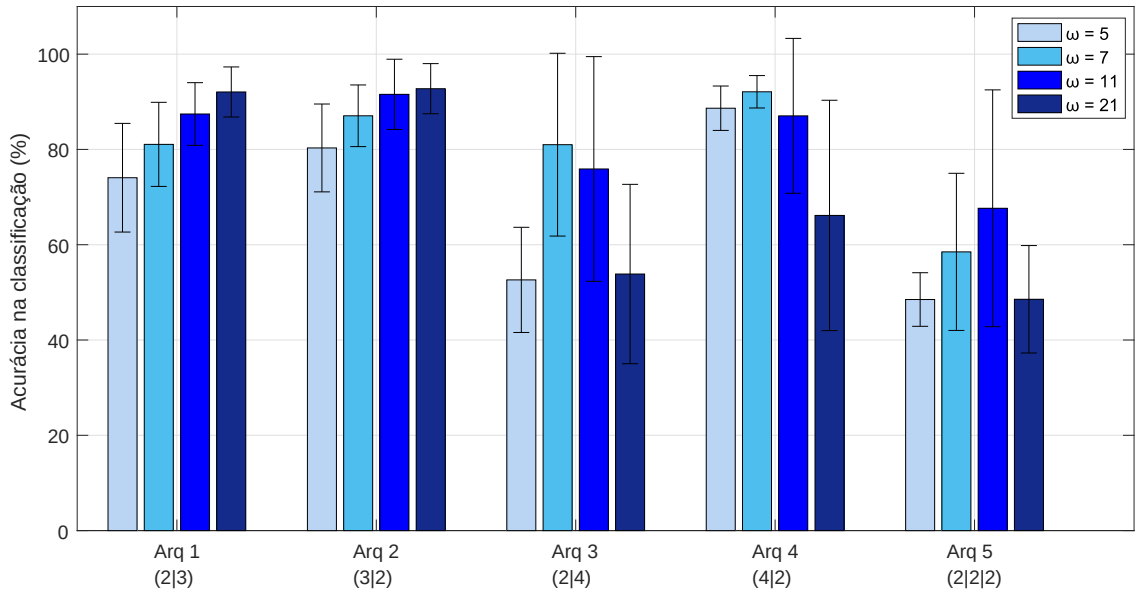



Figura 16: Acurácia obtida pelas arquiteturas com diferentes valores de $\omega$ em amostras com ruído de 10%

Para as arquiteturas Arq1 e Arq2, responsáveis pela classificação do bit de paridade em sinais de 6 bits, a acurácia obtida pela rede aumenta com o aumento de $\omega$. Tal resultado condiz com os ensinamentos de (MYERS, 1989), a qual afirma que maiores tamanhos de $\omega$ aumentam a tolerância da rede a ruídos. Para Arq3, Arq4 e Arq5, responsáveis pela classificação do bit de paridade em sinais de 8 bits, pode-se considerar que os valores obtidos foram inconclusivos. Para os valores de $\omega = 7$ e $\omega = 11$, foi possível observar um alto desvio padrão na acurácia, o que significa que o treinamento da rede possivelmente não convergiu dentro das 1000 épocas de treinamento.

Em geral, as arquiteturas com $\omega = 7$ e $\omega = 11$ obtiveram os melhores resultados para a classificação do bit de paridade em sinais de 8 bits. Nota-se ainda que as arquiteturas com valor de $\omega = 21$ obtiveram os piores resultados. É importante notar que o problema de classificação do bit de paridade se torna cada vez mais complexo com o aumento do tamanho de bits do sinal. Dessa forma, é possível que um valor alto de $\omega$ tenha dificultado a rede de atingir os valores de acurácia de 0% e 100% antes das 1000 épocas de treinamento.

A Figura 17 mostra o desempenho das arquiteturas testadas em termos do número de épocas de treinamento necessário para concluir o treinamento da rede.

O alto desvio padrão observado nas Arq1 e Arq2 se deve ao fato de que diversas vezes a rede não convergiu durante as 1000 épocas de treinamento. Para as Arq3, Arq4

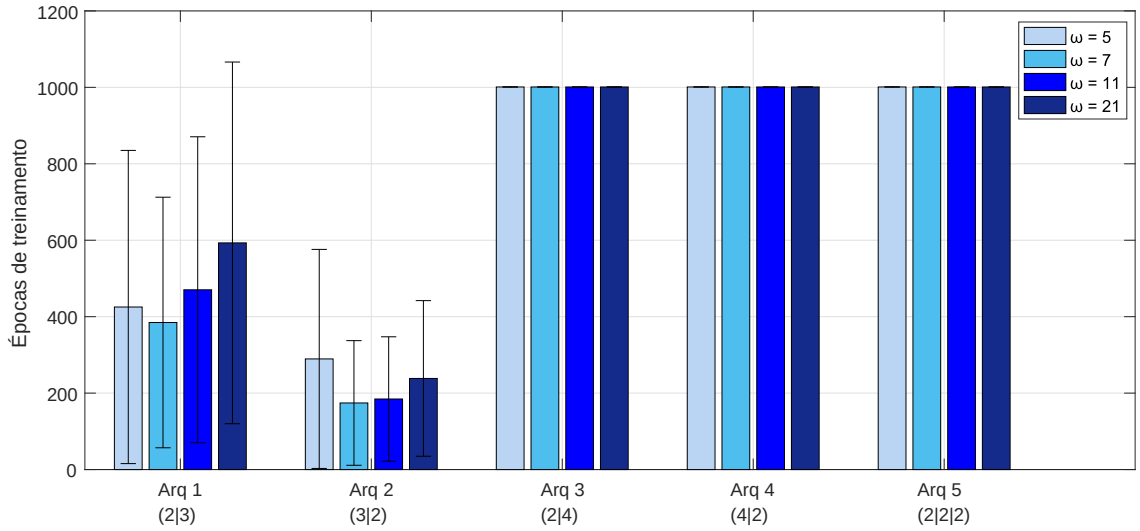



Figura 17: Épocas de treinamento necessárias para concluir o treinamento da rede para as arquiteturas com diferentes valores de $\omega$ em amostras com ruído de 10%

e Arq5, nota-se que a rede não convergiu sequer uma vez durante as 1000 épocas de treinamento. Dessa forma, não foi possível estabelecer uma relação entre o valor de $\omega$ e o número de épocas durante o treinamento para um conjunto de treinamento com ruído de 10%.

A Figura 18 mostra o desempenho das arquiteturas testadas em termos de tempo de processamento necessário para concluir o treinamento e teste da rede.

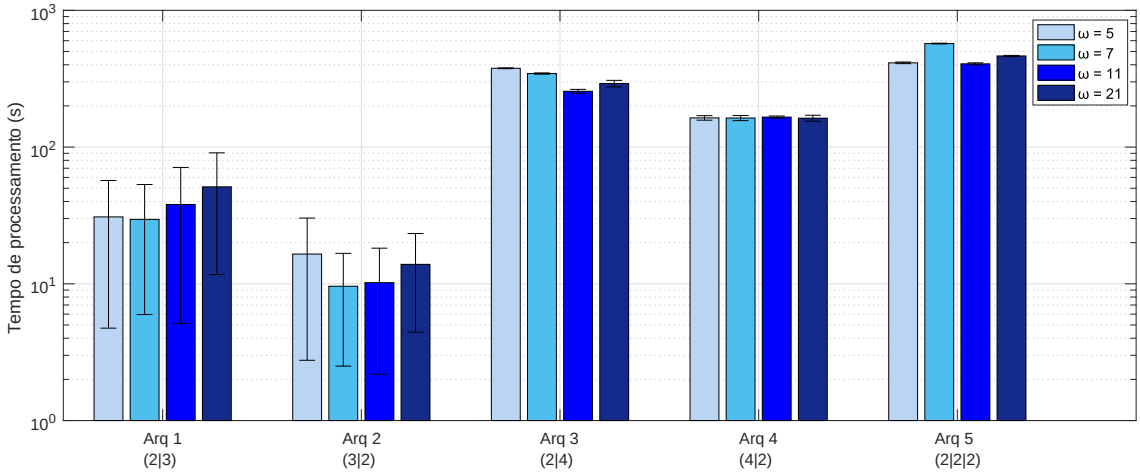



Figura 18: Tempo de processamento necessário para concluir o treinamento e teste da rede para as arquiteturas com diferentes valores de $\omega$ em amostras com ruído de 10%

De modo similar ao ocorrido para a análise das épocas de treinamento, o alto índice de não convergência das arquiteturas prejudica a análise da relação entre o valor

de $\omega$ e o tempo de processamento das arquiteturas. Para as Arq1 e Arq2, nota-se que a rede obteve os maiores tempos de processamento para $\omega = 21$. Tal valor é esperado, uma vez que uma alta resolução da memória faz que com mais ajustes em seu conteúdo sejam necessários para alcançar os valores de probabilidade 0% e 100%. Para Arq3, Arq4 e Arq5, os altos valores de tempo são resultado do fato de que as redes não convergiram durante as 1000 épocas de treinamento, tal como observado na Figura 17. A Tabela 8 apresenta uma análise comparativa entre os resultados obtidos pelas arquiteturas para diferentes $\omega$ com ruído de 10%.

Tabela 9: Comparativo entre as arquiteturas para diferentes $\omega$ para paridade com ruído de 10%

| | | **Resolução da memória** | | | |
|---|---|---|---|---|---|
| **Arq.** | **Resultado** | $\omega = 5$ | $\omega = 7$ | $\omega = 11$ | $\omega = 21$ |
| **Arq1** | **Acurácia** | pior | bom | melhor | melhor |
| | **Épocas** | regular | bom | regular | pior |
| | **Tempo** | bom | bom | regular | pior |
| **Arq2** | **Acurácia** | bom | bom | melhor | melhor |
| | **Épocas** | pior | melhor | bom | pior |
| | **Tempo** | pior | bom | bom | regular |
| **Arq3** | **Acurácia** | pior | melhor | bom | pior |
| | **Épocas** | pior | pior | pior | pior |
| | **Tempo** | pior | regular | melhor | bom |
| **Arq4** | **Acurácia** | bom | melhor | bom | pior |
| | **Épocas** | pior | pior | pior | pior |
| | **Tempo** | bom | bom | bom | bom |
| **Arq5** | **Acurácia** | pior | regular | melhor | pior |
| | **Épocas** | pior | pior | pior | pior |
| | **Tempo** | melhor | pior | melhor | regula |

Por fim, será analisado o impacto da variação da resolução da memória utilizando conjuntos de treinamento com ruído de 15%. A Figura 19 mostra o desempenho das arquiteturas testadas em termos da acurácia durante a fase de teste. Ambos os conjuntos de treinamento de 6 e 8 bits foram modificados para incluir 15% amostras com valor esperado incorreto. As simulações foram realizadas utilizando o grupo G3 das amostras de treinamento.

Conforme pode ser observado, para Arq1 e Arq2, responsáveis pela classificação do bit de paridade em sinais de 6 bits, a acurácia obtida pela rede aumenta com o aumento de $\omega$. Tal resultado condiz com os resultados obtidos para os conjuntos de treinamento com ruído de 5% e 10%. Para Arq3, Arq4 e Arq5, responsáveis pela classificação do bit

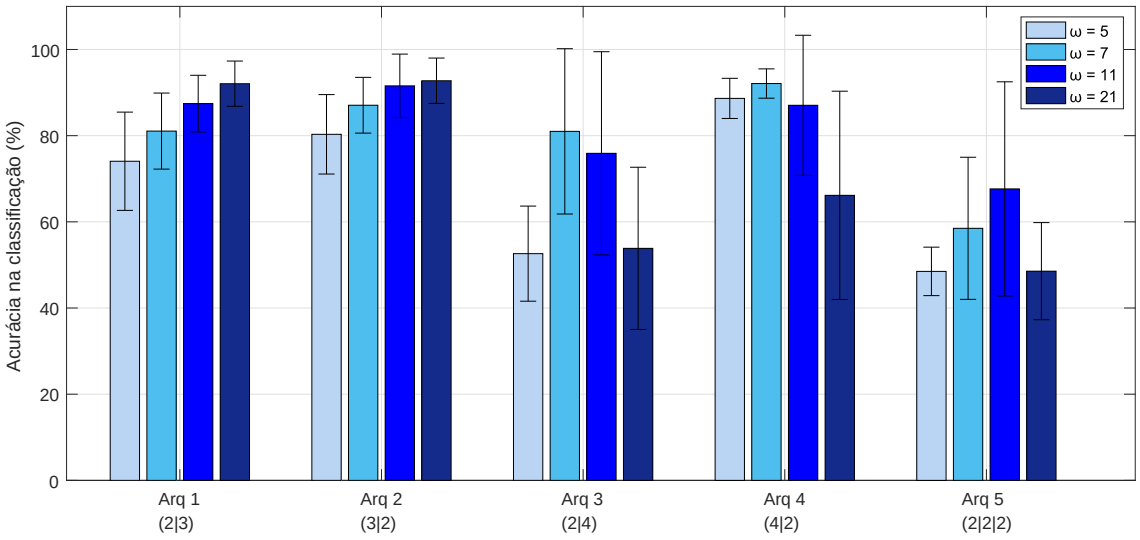



Figura 19: acurácia obtida pelas arquiteturas com diferentes valores de $\omega$ em amostras com ruído de 15%

de paridade em sinais de 8 bits, as arquiteturas obtiveram melhores resultados para $\omega = 7$ e $\omega = 11$, apesar do alto desvio padrão.

Com relação às épocas de treinamento necessárias para concluir o treinamento da rede, foi observado que nenhuma das arquiteturas foi capaz de concluir o treinamento antes das 1000 épocas de treinamento. O fato das redes não terem o treinamento concluído justifica o alto desvio padrão na acurácia.

A Figura 20 mostra o desempenho das arquiteturas testadas em termos de tempo de processamento necessário para concluir o treinamento e teste da rede.

Dentre as arquiteturas testadas, aquelas com $\omega = 7$ obtiveram os melhores resultados em termos de tempo de processamento, em especial para a Arq2. Nota-se ainda que, na arquitetura Arq5, valores de $\omega$ muito altos ou muito baixos levam à instabilidade no treinamento. Conforme explicado em (MYERS, 1989), valores muito baixos de $\omega$ fazem com que a rede perca facilmente o conhecimento armazenado do problema na presença de amostras com ruído ou corrompidas. Por outro lado, valores muito altos de $\omega$ tornam necessários muitos ajustes na rede até que a rede atinja valores de probabilidade de 0% e 100%. A Tabela 8 apresenta uma análise comparativa entre os resultados obtidos pelas arquiteturas para diferentes $\omega$ com ruído de 15%.

Assim, assim como observado para os conjuntos de treinamento com 5% e 10%, as arquiteturas com $\omega = 7$ e $\omega = 11$ obtiveram os melhores valores de acurácia durante a classificação, enquanto as arquiteturas com $\omega = 7$ obtiveram em geral os melhores

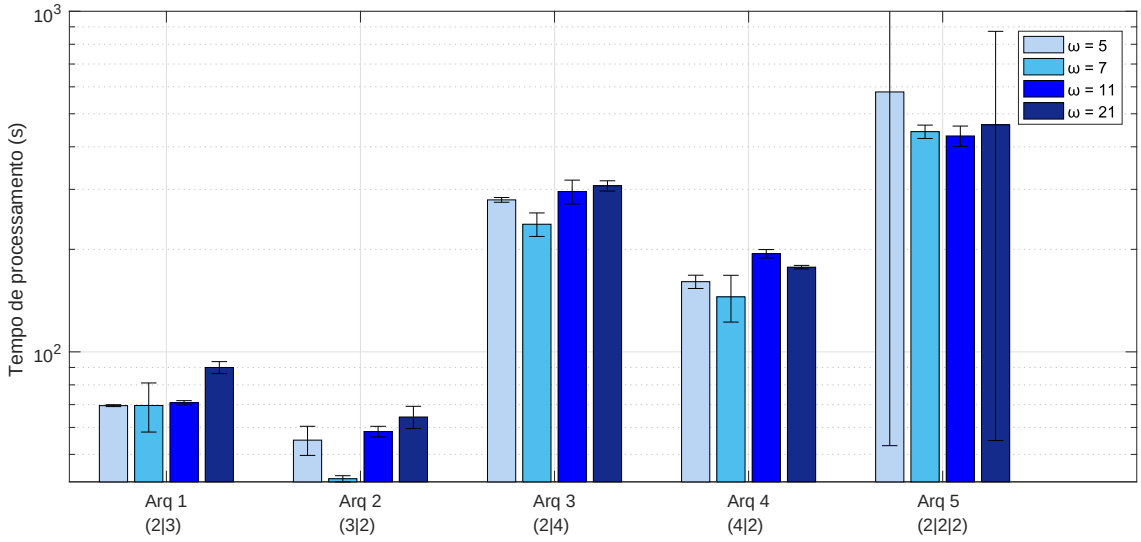



Figura 20: Tempo de processamento necessário para concluir o treinamento e teste da rede para as arquiteturas com diferentes valores de $\omega$ em amostras com ruído de 15%

Tabela 10: Comparativo entre as arquiteturas para diferentes $\omega$ para paridade com ruído de 15%

| | | **Resolução da memória** | | | |
|---|---|---|---|---|---|
| **Arq.** | **Resultado** | $\omega = 5$ | $\omega = 7$ | $\omega = 11$ | $\omega = 21$ |
| **Arq1** | **Acurácia** | pior | regular | bom | melhor |
| | **Épocas** | pior | pior | pior | pior |
| | **Tempo** | regular | regular | regular | pior |
| **Arq2** | **Acurácia** | pior | bom | melhor | melhor |
| | **Épocas** | pior | pior | pior | pior |
| | **Tempo** | regular | melhor | regular | pior |
| **Arq3** | **Acurácia** | bom | melhor | bom | pior |
| | **Épocas** | pior | pior | pior | pior |
| | **Tempo** | regular | melhor | regular | pior |
| **Arq4** | **Acurácia** | bom | melhor | bom | pior |
| | **Épocas** | pior | pior | pior | pior |
| | **Tempo** | regular | melhor | pior | regular |
| **Arq5** | **Acurácia** | pior | regular | melhor | pior |
| | **Épocas** | pior | pior | pior | pior |
| | **Tempo** | pior | regular | regular | regular |

resultados em termos de número de épocas de treinamento e tempo de processamento. Pode-se concluir que para o problema de classificação do bit de paridade com presença de ruídos, o valor mais indicado para a resolução da memória seria $\omega = 7$.

## 5.2 Problema de reconhecimento dos algarismos

Nesta seção, serão discutidos os resultados obtidos para a segunda aplicação implementada nesta dissertação: o reconhecimento de algarismos manuscritos. As arquiteturas descritas na Seção 4.4.1 foram implementadas em ambiente de simulação de modo a analisar as particularidades do problema de classificação com múltiplas classes e o impacto da variação dos parâmetros no projeto de uma RNSP no reconhecimento de imagens.

### 5.2.1 Avaliação do desempenho da rede MPLN modificada

Primeiramente, serão discutidos os principais aspectos referentes à rede Mod-MPLN proposta nesta dissertação. Será comprovada a assertiva de que a rede MPLN tradicional não apresenta resultados satisfatórios para problemas de classificação envolvendo múltiplas classes. Em seguida, os resultados da rede Mod-MPLN serão apresentados, de modo a validar a modificação proposta.

#### 5.2.1.1 Impacto do número de classes na rede MPLN

Conforme discutido na Seção 3.2, a rede MPLN não seria eficaz para problemas envolvendo múltiplas classes, uma vez que cada camada da rede tem seu valor iniciado em 'u', devendo ser ajustada para todas as amostras de todas as classes. Durante o treinamento da rede, todos os valores de 'u' devem ser modificados para probabilidades 0% e 100% do neurônio emitir uma saída de nível lógico '1', de modo a evitar o comportamento probabilístico da rede. O que ocorre é que, para problemas de múltiplas classes, a rede MPLN necessitará se ajustar para não reconhecer todas as classes exceto a classe para qual a rede é treinada para reconhecer. Assim, entende-se que o treinamento da rede em excesso para identificar diversos padrões diferentes como pertencendo a uma mesma classe, a saber, a classe "não pertence", pode comprometer a convergência da rede.

De modo a corroborar esta assertiva, foram simuladas algumas arquiteturas de RNSP MPLN para o reconhecimento de algarismos com classes reduzidas. O primeiro teste realizado consiste em reconhecer apenas dois caracteres, a saber, os algarismos 0 e 1. Dessa forma, a RNSP projetada possui apenas uma camada, similar às redes utilizadas na classificação do bit de paridade.

Para estes testes, foram utilizadas arquiteturas com tamanho de tupla de 4 e 8 bits, e treinadas com conjuntos de treinamento contendo 10, 50 e 100 amostras de cada classe.

Além disso, conforme visto na Seção 4.1.2, cada conjunto de imagem foi subdividido em três conjuntos: o grupo de imagens de boa qualidade GB, o grupo de imagens de média qualidade GM, e o grupo de imagens de qualidade ruim GR. A Figura 21 ilustra os resultados obtidos para a rede MPLN configurada para classificar os algarismos 0 e 1.

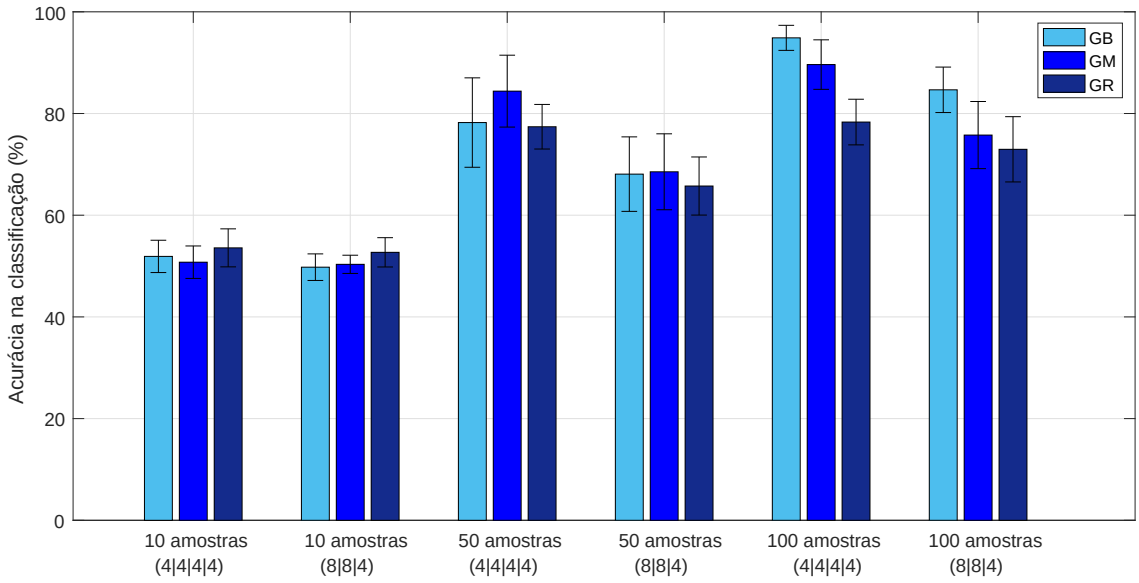



Figura 21: Acurácia obtida pelas arquiteturas da MPLN na classificação de duas classes de algarismos manuscritos

Nota-se que a acurácia da rede aumenta com o número de amostras de treinamento para as duas configurações de tupla testadas. Especificamente, a rede MPLN apresenta resultados satisfatórios para os grupos GB, GM e GR a partir de 100 amostras de treinamento por classe. Sendo assim, pode concluir que a rede MPLN é capaz de classificar entre as duas classes algarismos manuscritos testadas.

As mesmas arquiteturas de rede MPLN serão implementadas para a classificação de quatro classes de algarismos manuscritos, a saber, os algarismos 0, 1, 2 e 3. Tendo em vista que os ensinamentos sobre a rede MPLN (ALEKSANDER; MORTON, 1989; BRAGA; CARVALHO; LUDERMIR, 2000; NEDJAH et al., 2016) não especificam um elemento capaz de processar a saídas das camadas paralelas da rede MPLN, foi utilizado um discriminador tal como proposto na Seção 4.3. A Figura 22 ilustra os resultados obtidos para a rede MPLN configurada para classificar as classes descritas acima.

É possível notar que a rede MPLN não apresenta resultados satisfatórios em qualquer das arquiteturas e utilizando qualquer dos três conjuntos de treinamento. Isto se deve ao fato de que a rede é excessivamente ajustada para responder com nível lógico '0'

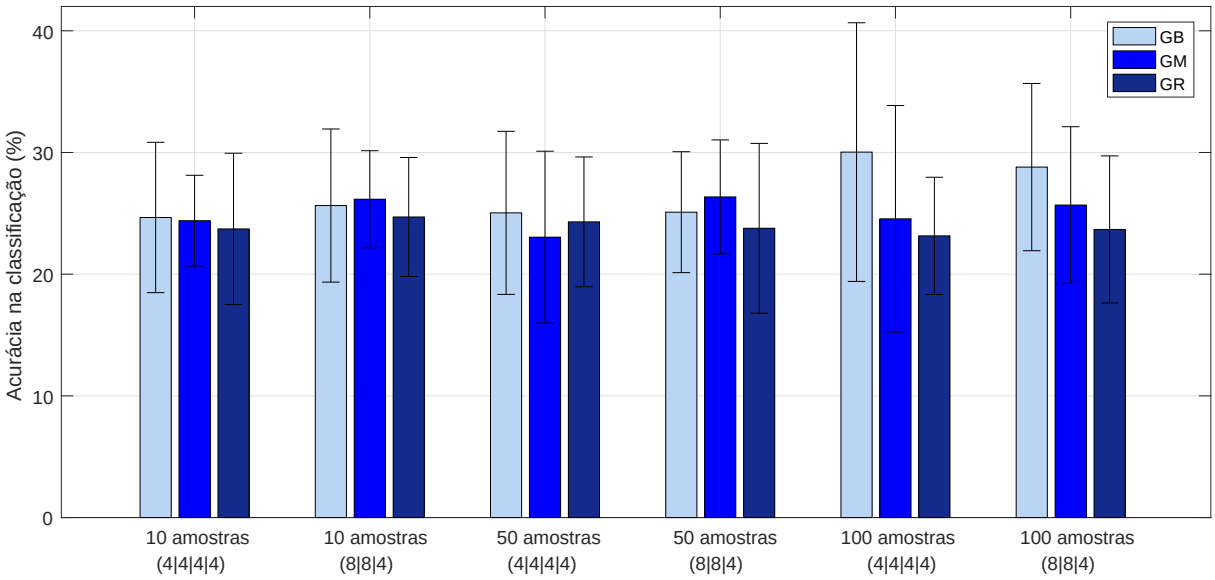



Figura 22: Acurácia obtida pelas arquiteturas da MPLN na classificação de quatro classes de algarismos manuscritos

aos padrões de treinamento. Especificamente para os testes com quatro classes, os neurônios da rede MPLN são treinados três vezes mais para responder com saída '0' do que para responder com saída '1'. Além disso, a rede acaba sendo treinada para identificar padrões diferentes como pertencentes a uma mesma classe '0'. Assim, pode-se concluir que estes dois contratempos no treinamento da rede MPLN impedem que a mesma consiga aprender corretamente o problema a ser solucionado.

De modo a solucionar os problemas da rede MPLN para aplicações com múltiplas classes, o presente trabalho propôs a rede Mod-MPLN, conforme descrito na Seção 3.3. Na rede Mod-MPLN, as camadas paralelas da rede são apenas treinadas com o padrão para qual elas devem ser configuradas para identificar. Como as posições de memória não acessadas durante o treinamento permaneceram com valor 'u', é proposto um discriminador na saída da rede para processar a resposta de todas as camadas paralelas e produzir uma resposta unificada.

Sendo assim, como a rede MPLN se mostrou ineficaz no problema da classificação de imagens com múltiplas classes, a aplicação de reconhecimento de algarismos manuscritos será realizada utilizando a RNSP do tipo Mod-MPLM. Entretanto, antes de analisar o impacto da variação dos parâmetros na rede Mod-MPLN, faz-se necessário apresentar os resultados para o problema de classificação de algarismos manuscritos, de modo a validar a modificação proposta.

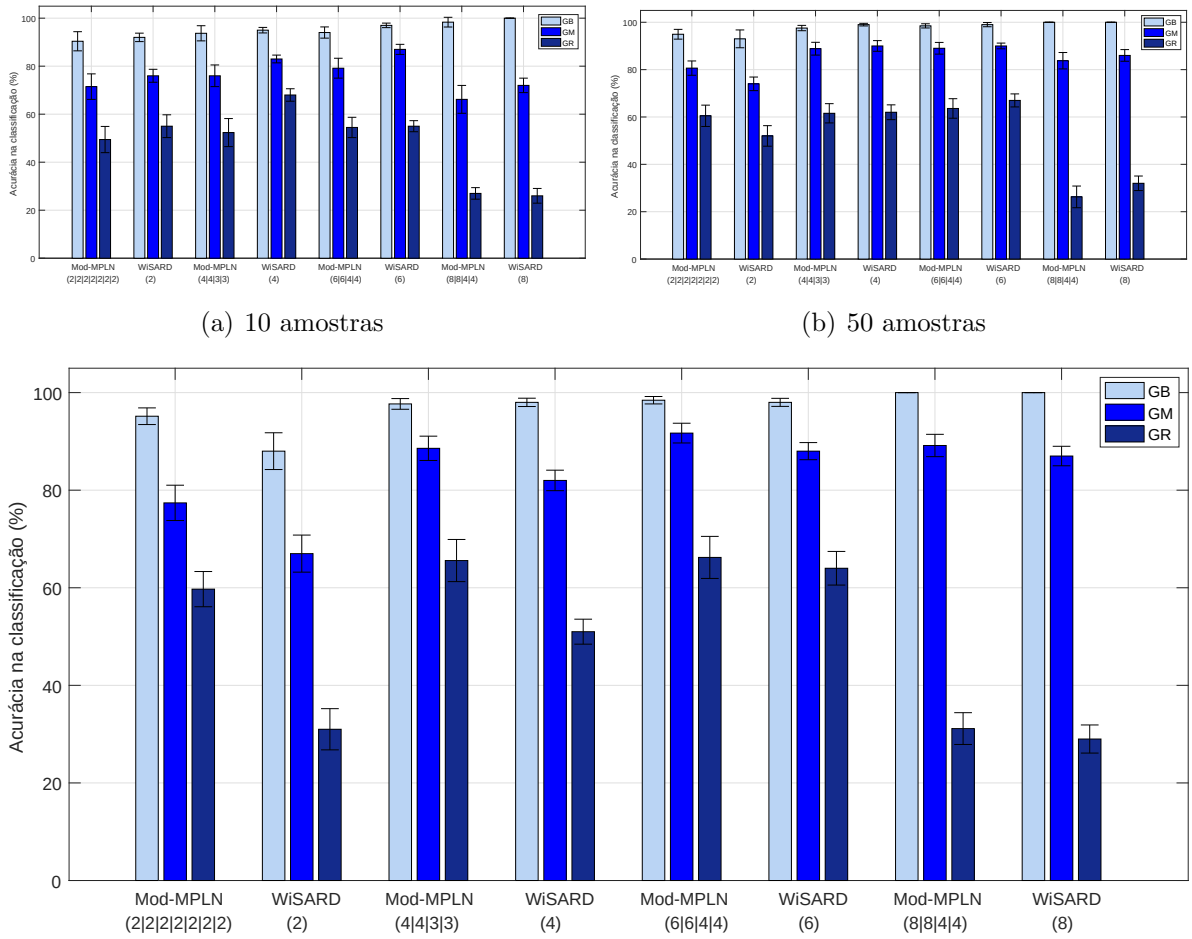
#### 5.2.1.2 Comparativo da acurácia das redes Mod-MPLN e WiSARD

De modo a validar a eficácia da rede Mod-MPLN, a mesma foi implementada para o reconhecimento de caracteres manuscritos de 0 a 9, totalizando dez classes. Os resultados da rede são comparados com os resultados da rede WiSARD (ALEKSANDER; THOMAS; BOWDEN, 1984), a qual é uma rede especialmente adaptada para o problema de reconhecimento de imagens. Conforme visto na Seção 4.1.2, o conjunto de caracteres foi dividido em três grupos de acordo com o o coeficiente de correlação de *Pearson*, a saber, o grupo das imagens de boa qualidade (GB), o grupo das imagens de média qualidade (GM) e o conjunto de imagens com qualidade ruim (GR). As arquiteturas testadas possuem tuplas de 2, 4, 6 e 8 bits. Para a rede Mod-MPLN, todos os testados foram realizados utilizando o valor de $\omega = 11$, conforme recomendação de (MYERS, 1989). A Figura 23(a) ilustra os resultados obtidos pelas arquiteturas das RNSPs Mod-MPLN e WiSARD para o conjunto de dados de caracteres manuscritos contendo 10 amostras de treinamento por classe, e 100 amostras de teste.

Nota-se que ambas as arquiteturas obtiveram resultados satisfatórios para o grupo GB, resultados aceitáveis para o grupo GM, e resultados consideravelmente ruins para o grupo GR. Os resultados comprovam a importância de separar o conjunto de treinamento da base de dados MNIST, uma vez que este possui imagens das mais variadas qualidades.

A partir da Figura 23(a), é possível verificar que a rede WiSARD obteve resultados ligeiramente superiores à rede Mod-MPLN. Acredita-se que isto se deve ao fato da rede Mod-MPLN possuir uma arquitetura em pirâmides (BRAGA; CARVALHO; LUDERMIR, 2000). Conforme visto na Seção 1.4, na arquitetura em pirâmide a informação é propagada a partir dos neurônios do estágio de entrada até o neurônio do estágio de saída. Dessa forma, para apenas 10 amostras de treinamento por classe, é possível que a rede Mod-MPLN não consiga processar corretamente as saídas do estágio de entrada de modo a aprender características das imagens. Por outro lado, a rede WiSARD possui apenas um estágio de neurônios, o qual é diretamente endereçado pelos padrões de treinamento.

A Figura 23(b) ilustra os resultados obtidos pelas arquiteturas das RNSPs Mod-MPLN e WiSARD para os conjuntos de dados contendo 50 amostras por classe, totalizando 500 amostras de treinamento. Nota-se que, para o conjunto de dados com 50 amostras por classe, as redes WiSARD e Mod-MPLN obtiveram desempenhos semelhantes, sendo que a rede Mod-MPLN obteve resultados superiores para a arquitetura com




(a) 10 amostras

(b) 50 amostras

(c) 100 amostras

Figura 23: Acurácia obtida por Mod-MPLN *vs.* WiSARD para treinamentos usando diferentes números de amostras por classe

tupla de 2 bits. Isso se deve ao fato de a rede WiSARD ser suscetível a saturação em casos de muitas amostras de treinamento, em especial para arquiteturas de tupla muita pequena (ARAúJO, 2011). Dessa forma, a rede Mod-MPLN possui maior tolerância à saturação do que a rede WiSARD.

Por fim, a Figura 23(c) ilustra os resultados obtidos pelas arquiteturas das RNSPs Mod-MPLN e WiSARD para os três conjuntos de treinamento utilizando 100 amostras por classe, totalizando 1000 amostras de treinamento. Nota-se que a rede Mod-MPLN apresenta resultados consideravelmente superiores à rede WiSARD para o conjunto de 100 amostras de treinamento. Conforme falado, a rede WiSARD pode saturar caso muitas amostras de treinamento enderecem as mesmas posições de memória dos neurônios. A rede WiSARD tradicional não é capaz de diferenciar uma posição de memória endereçada apenas uma vez de outra endereçada diversas vezes. Por outro lado, a rede MPLN e sua

variante Mod-MPLN possui posições de memória com resolução $\omega$, a qual torna a rede capaz de diferenciar as posições de memória mais acessadas durante o treinamento.

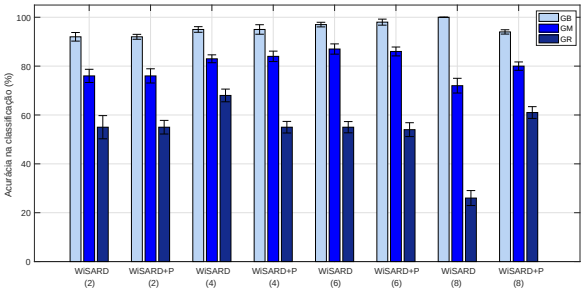
A partir das Figuras 23(a), 23(b) e 23(c), pode-se concluir que o desempenho da rede WiSARD cai com o aumento das amostras nos casos onde a tupla da rede possui apenas 2 bits, e se mantém essencialmente constante para as demais arquiteturas. Por outro lado, a rede Mod-MPLN apresenta melhores resultados com o aumento no número de amostras de treinamento. Dessa forma, pode-se concluir que a rede Mod-MPLN não só apresenta acurácia satisfatória para os problemas de classificação envolvendo múltiplas classes, como também pode ser mais indicada que a rede WiSARD em aplicações com muitas amostras de treinamento.

#### 5.2.1.3 Análise do impacto do preprocessamento

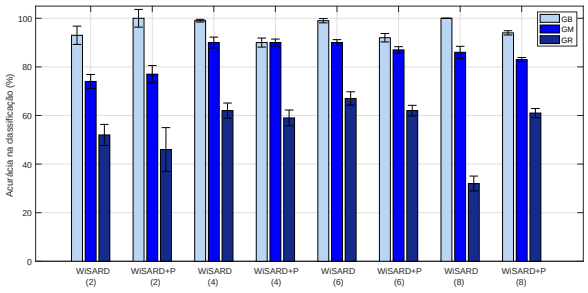
De modo a analisar o impacto da técnica de pré-processamento aplicada no desempenho da rede, os resultados dos conjuntos de imagens com e sem pré-processamento foram comparados entre si. A técnica de pré-processamento utilizada foi descrita na seção 4.1.1. A Figura 24 ilustra os resultados para as arquiteturas da rede WiSARD com e sem o bloco de pré-processamento.

Conforme pode ser observado na Figura 24(a), as arquiteturas com o bloco de pré-processamento apresentaram resultados ligeiramente superiores, em especial para o GR nas arquiteturas de tupla de 8 bits. Para o grupo de amostras contendo 50 amostras por classe, ilustrado na Figura 24(b), as arquiteturas da rede WiSARD apresentaram resultados similares, com a exceção do grupo GR da arquitetura de tupla de 8 bits. Já no grupo de amostras contendo 100 amostras por classe, ilustrado na Figura 24(c), as arquiteturas com o bloco de pré-processamento apresentaram em geral resultados superiores para todos os conjuntos de treinamento.

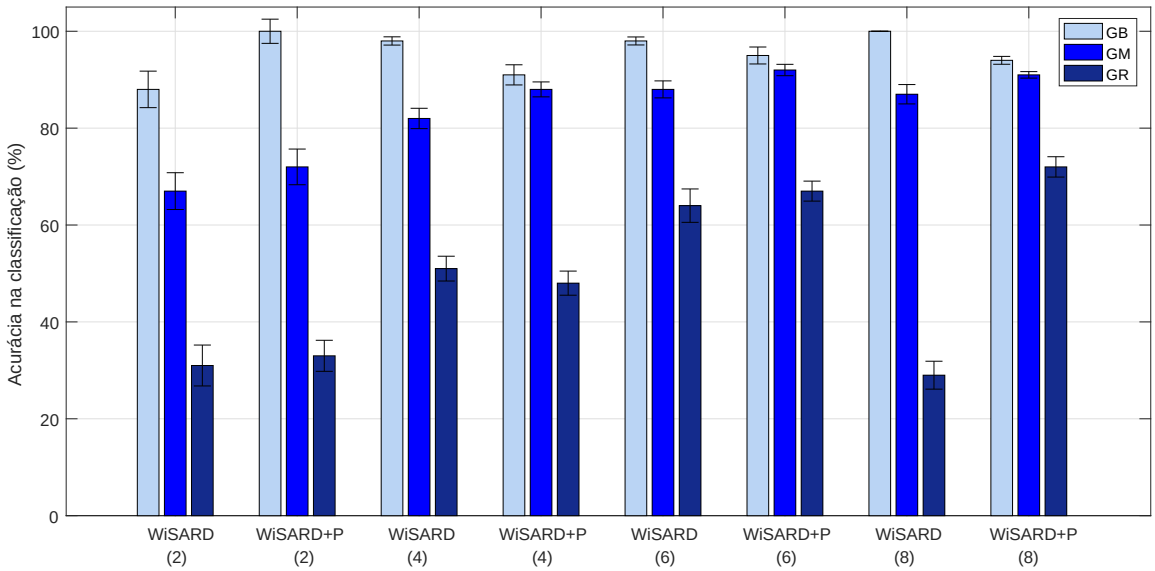
De forma geral, a rede WiSARD apresenta resultados melhores com o uso de um pré-processamento para centralizar os algarismos em um local comum da imagem. Isto é importante pois a rede WiSARD reconhece as imagens a partir do reconhecimento de partes individuais da imagem (tuplas), logo é importante que as partes das imagens sejam partes comuns do algarismo para todas as imagens. Pode-se observar ainda que o bloco de pré-processamento é especialmente importante na rede WiSARD para os grupos de imagem com qualidade inferior. Isto ocorre porque as imagens de qualidade inferior são muito diferentes entre si, e tendem a acessar diferentes posições de memória da rede. Como




(a) 10 amostras




(b) 50 amostras



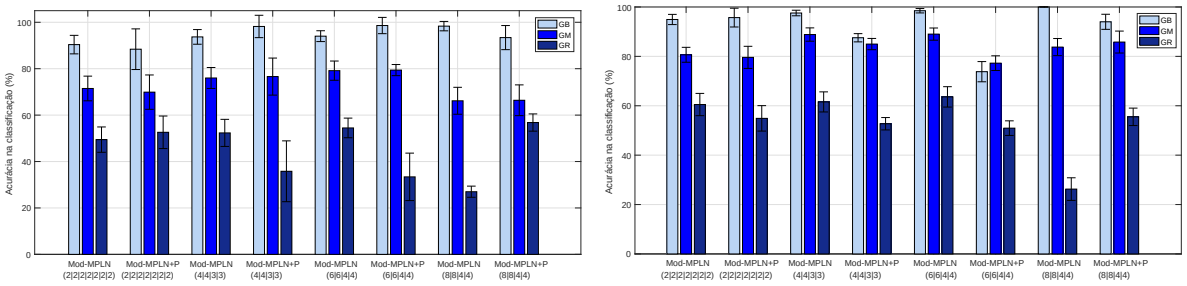

(c) 100 amostras

Figura 24: Acurácia obtida por WiSARD com e sem o pré-processamento para treinamentos usando diferentes números de amostras por classe

a rede WiSARD convencional não possui um mecanismo para contabilizar os acessos em suas posições de memória, ela não saberá diferenciar posições acessadas ocasionalmente pelas imagens com baixa qualidade daquelas que são comuns à classe a ser classificada.

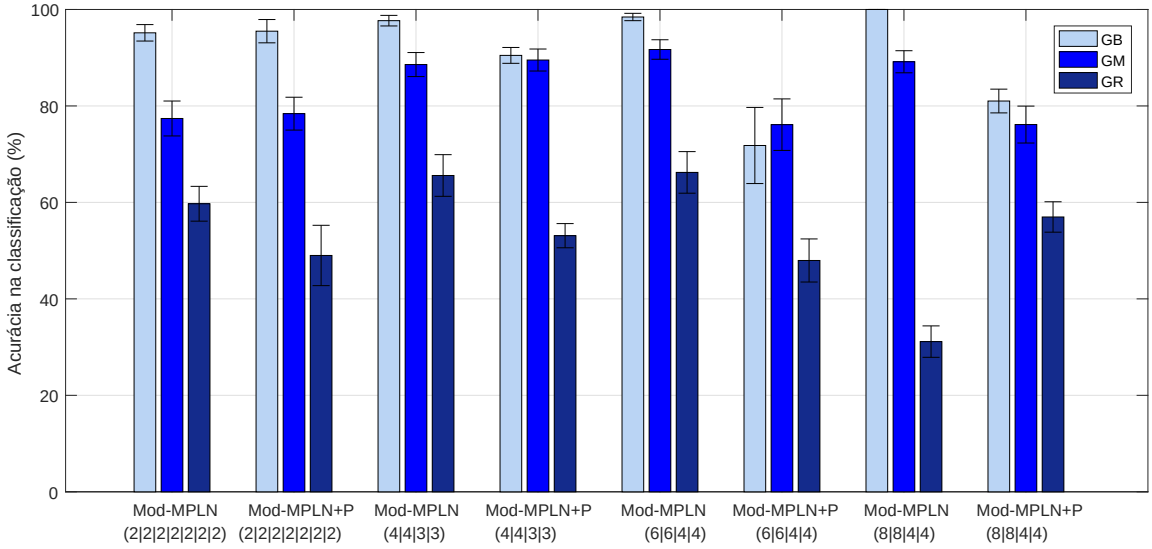
A seguir, serão analisados os resultados entre as arquiteturas da rede Mod-MPLN com e sem o bloco de pré-processamento. A Figura 25(a) mostra os resultados obtidos para conjuntos de treinamento com 10 amostras por classe, totalizando 100 amostras. Pode-se notar que a inclusão do bloco de pré-processamento tanto aumenta a acurácia para certos casos, como também diminui em outros. A diferença mais significante observada ocorre no grupo GR para as arquiteturas de tupla de 8 bits, onde a configuração com pré-processamento obteve resultados consideravelmente superiores.

A Figura 25(b) mostra os resultados obtidos para conjuntos de treinamento com 50 amostras por classe, totalizando 500 amostras. Nota-se que a inclusão do pré-processamento




(a) 10 amostras

(b) 50 amostras




(c) 100 amostras

Figura 25: Acurácia obtida por Mod-MPLN com e sem o pré-processamento para treinamentos usando diferentes números de amostras por classe

diminuiu a acurácia da rede para todos os conjuntos de treinamento nas arquiteturas com tamanho de tupla de 2, 4 e 6 bits. Para a arquitetura de tamanho de tupla de 8 bits, nota-se que o bloco de pré-processamento proporciona melhores resultados para os grupos GM e GR. O mesmo pode ser analisado na Figura 25(c), onde o bloco de pré-processamento apenas proporcionou resultados melhores para o grupo GR da arquitetura de tupla de 8 bits.

Sendo assim, pode-se concluir que a rede Mod-MPLN possui menos dependência de um pré-processamento do que a rede WiSARD. Isso se deve ao fato de que a rede Mod-MPLN é uma rede de múltiplos estágios. Assim como nas RNAs convencionais, os estágios mais profundos da rede são capazes de aprender características mais intrínsecas do problema a ser resolvido (BANDEIRA, 2010). Por outro lado, a WiSARD meramente contabiliza a quantidade de neurônios que foram previamente acessados por um padrão

de treinamento, e classifica o padrão como pertencente à classe reconhecida por mais neurônios.

É importante notar que não é o objetivo do presente trabalho propor técnicas de pré-processamento para serem utilizadas em RNSPs. Conforme visto na Seção 4.1.1, o pré-processamento utilizado no presente trabalho é relativamente simples, e possivelmente existem técnicas mais robustas capazes de possibilitar resultados superiores tanto para a rede WiSARD quanto para a rede Mod-MPLN. Contudo, o estudo apresentado nesta seção tem apenas o objetivo de comparar a dependência das redes WiSARD e Mod-MPLN de um pré-processamento dos conjuntos de dados alimentados às rede.

### 5.2.2 Avaliação do impacto da configuração dos parâmetros

A seguir, será analisado o impacto das variações dos parâmetros para a rede Mod-MPLN, similarmente ao que foi discutido nas Seções 5.1.1 e 5.1.2. Inicialmente, serão apresentados resultados de simulação para a análise do impacto do tamanho da tuplas dos neurônios e o número de estágios da rede. Para isto, diversas arquiteturas de rede Mod-MPLN foram testadas, sendo que cada arquitetura possui um número de estágio, número de neurônios por estágio e tamanho da tupla por neurônio.

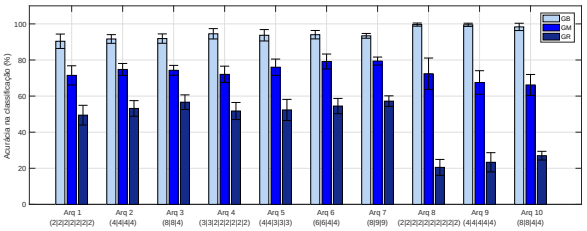
Em seguida, será analisado o impacto da resolução da memória $\omega$ na eficiência e eficácia das redes. Nesta análise, diversas arquiteturas da rede Mod-MPLN foram testadas utilizando diversos valores de $\omega$, sendo que as redes são treinadas com conjuntos de dados contendo ruído, de modo a relacionar o valor de $\omega$ e a tolerância ao ruído das redes.

#### 5.2.2.1 Impacto do tamanho da tupla para a rede Mod-MPLN

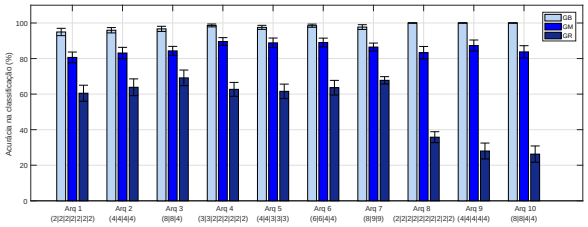
Conforme visto nas Seções 3.1.1 e 3.1.2, a escolha do tamanho da tupla e o número de estágios da rede MPLN são muito importantes no projeto de uma RNSP, uma vez que estes parâmetros impactam diretamente no tamanho da memória utilizada, capacidade de generalização e capacidade de aprendizado da rede (ALEKSANDER; MORTON, 1989; MYERS, 1989).

A Figura 26(a) ilustra os resultados obtidos para as 10 arquiteturas simuladas, utilizando apenas 10 amostras de treinamento por classe. Conforme visto na Seção 4.4.1, as 10 arquiteturas propostas têm o objetivo de identificar os três conjuntos de imagem utilizados. Especificamente, as arquiteturas Arq1, Arq2 e Arq3 identificam os algarismos do conjunto de 16x16 *pixels*, as arquiteturas Arq4, Arq5, Arq6 e Arq7 identificam os

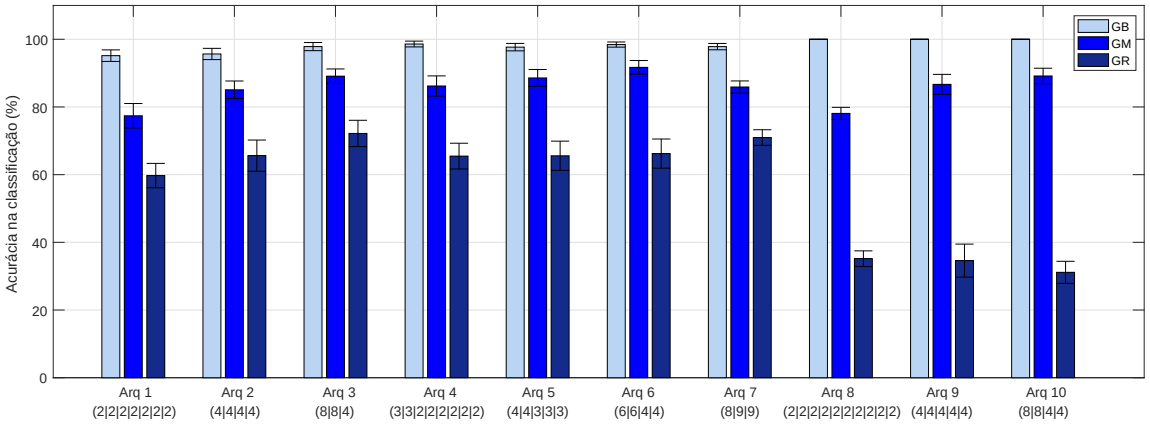
algarismos do conjunto de 24x24 *pixels*, e as arquiteturas Arq8, Arq9 e Arq10 identificam os algarismos do conjunto de 32x32 *pixels*. Além disso, cada conjunto de imagem foi subdividido em três conjuntos: o grupo de imagens de boa qualidade GB, o grupo de imagens de média qualidade GM, e o grupo de imagens de qualidade ruim GR. Pode-se observar na Figura 26(a), entre parênteses, o tamanho da tupla por estágio da rede. Os valores representam a média e o desvio padrão de simulações para cada uma das três arquiteturas utilizadas na classificação dos algarismos manuscritos da base de dados MNIST. A configuração do tamanho da tupla para cada arquitetura está definida na forma de: (tupla no estágio 1 | tupla no estágio 2 | tupla no estágio 3).




(a) 10 amostras




(b) 50 amostras




(c) 100 amostras

Figura 26: Acurácia obtida por Mod-MPLN para treinamentos usando diferentes números de amostras por classe

Com relação à acurácia obtida pelas arquiteturas, nota-se que, conforme esperado, todas as arquiteturas obtiveram resultados melhores para os grupos de imagens de melhor qualidade. Com relação às arquiteturas Arq1, Arq2 e Arq3, responsáveis por identificar os algarismos de 16x16 *pixels*, a Arq3 obteve resultados ligeiramente superiores, em especial para o grupo GR. Para as arquiteturas Arq4, Arq5, Arq6 e Arq7, responsáveis por identificar os algarismos de 24x24 *pixels*, as Arq6 e Arq7 obtiveram os melhores resultados. Para os grupos GB e GM, ambas as Arq6 e Arq7 obtiveram acurácia igual, enquanto a

Arq7 obteve resultados ligeiramente superiores para o grupo GR. Por fim, com relação às arquiteturas Arq8, Arq9 e Arq10, responsáveis por identificar os algarismos de 32x32 *pixels*, as Arq9 e Arq10 obtiveram os melhores resultados. Para os grupos GB e GM, ambas as Arq9 e Arq10 obtiveram acurácia igual, enquanto a Arq10 obteve resultados consideravelmente superiores para o grupo GR.

Na Figura 26(b), são mostrados os resultados obtidos para as 10 arquiteturas simuladas, utilizando 50 amostras de treinamento por classe. De modo simular à Figura 26(a) é possível observar que todas as arquiteturas obtiveram melhores resultados em termos de acurácia para os grupos de imagens de melhor qualidade.

Com relação às arquiteturas responsáveis por identificar os algarismos de 16x16 *pixels*, a Arq3 obteve resultados ligeiramente superiores, em especial para o grupo GR. Para as arquiteturas responsáveis por identificar os algarismos de 24x24 *pixels*, as arquiteturas Arq4, Arq5, Arq6 e Arq7 obtiveram resultados muito semelhantes. Por fim, com relação às arquiteturas responsáveis por identificar os algarismos de 32x32 pixels, a Arq8 obteve os melhores resultados para o grupo GR, enquanto a Arq9 obteve os melhores resultados para o grupo GM.

Dessa forma, para os três conjuntos de imagens testados, as arquiteturas com maior tamanho de tupla obtiveram os melhores resultados em termos de acurácia na classificação. Tais resultados são condizentes com o observado para o problema da paridade de $n$ bits discutido na Seção 5.1.1.

Na Figura 26(c), são mostrados os resultados obtidos para as 10 arquiteturas simuladas, utilizando apenas 100 amostras de treinamento por classe. Dentre as arquiteturas responsáveis por identificar os algarismos de 16x16 *pixels*, a Arq3 obteve resultados consideravelmente superiores para os três grupos. Para as arquiteturas responsáveis por identificar os algarismos de 24x24 pixels, a Arq6 obteve os melhores resultados para os grupos GB e GM, enquanto a Arq7 obteve os melhores resultados para o grupo GR. Por fim, com relação às arquiteturas responsáveis por identificar os algarismos de 32x32 pixels, a Arq10 obteve os melhores resultados, com exceção do grupo GR.

Dessa forma, a partir da análise das Figuras 26(a), 26(b) e 26(c), pode-se concluir que em geral as arquiteturas com maior tamanho de tupla obtiveram os melhores resultados de acurácia. A única arquitetura que apresentou resultados abaixo das de menor

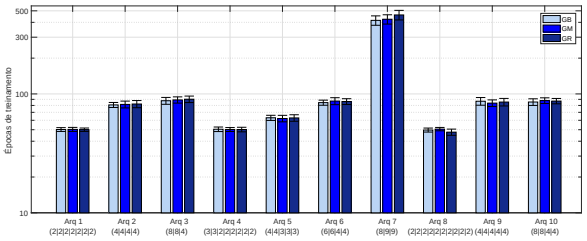
tupla foi a arquitetura Arq7, a qual possui tupla maior nos estágios das camadas mais próximas da saída.

Adicionalmente, pode-se observar as arquiteturas de tamanho maior de tupla se tornam mais interessantes para os conjuntos com mais amostras de treinamento. Isto ocorre pois as arquiteturas com tamanhos de tupla menores possuem menos posições de memória, uma vez que cada neurônio possui $2^N$ posições de memória. Dessa forma, para uma resolução de memória $\omega$ constante, estas arquiteturas apresentam maior tendência à saturação. Os resultados obtidos são condizentes com (GARCIA, 2003), onde o autor aponta que as configurações com maior tamanho de tupla são indicadas para conjuntos de treinamento com mais amostras. Por outro lado, para os tamanhos de tupla selecionados nesta dissertação, pode-se afirmar que tamanhos de tupla maiores apresentam maior capacidade de aprendizado e generalização. Como (GARCIA, 2003) afirma que uma tupla grande prejudica a capacidade de generalização da rede, entende-se que estes tamanhos devem ser consideravelmente superiores aos tamanhos utilizados nesta dissertação.

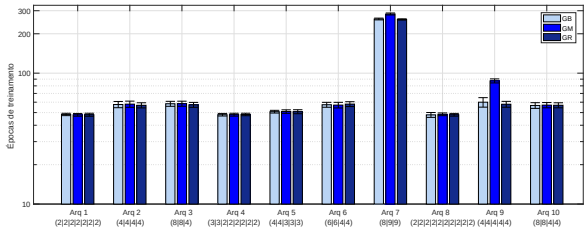
A seguir, será analisado o impacto do tamanho de tupla e estágios nas épocas de treinamento necessárias para que o treinamento da rede seja concluído. De forma análoga ao impacto das imagens na acurácia da rede, foram realizados testes para os conjuntos de dados contendo 10, 50 e 100 amostras por classe. A Figura 27(a) ilustra o número médio de épocas de treinamento necessárias para cada arquitetura completar a fase de treinamento utilizando os conjuntos de amostras com 10 amostras por classe.

Primeiramente, pode-se notar que, ao contrário da rede MPLN, a rede Mod-MPLN apresenta baixo desvio padrão para as épocas de treinamento. Isto ocorre pois, no treinamento da rede Mod-MPLN, dificilmente ocorrem conflitos, uma vez que apenas as amostras que a rede pretende identificar são apresentadas a suas respectivas camadas da rede. Dessa forma. Como a rede Mod-MPLN tem uma grande tendência a convergência durante o treinamento, não há necessidade da utilização do contador de épocas, da forma como foi feita para a rede MPLN.

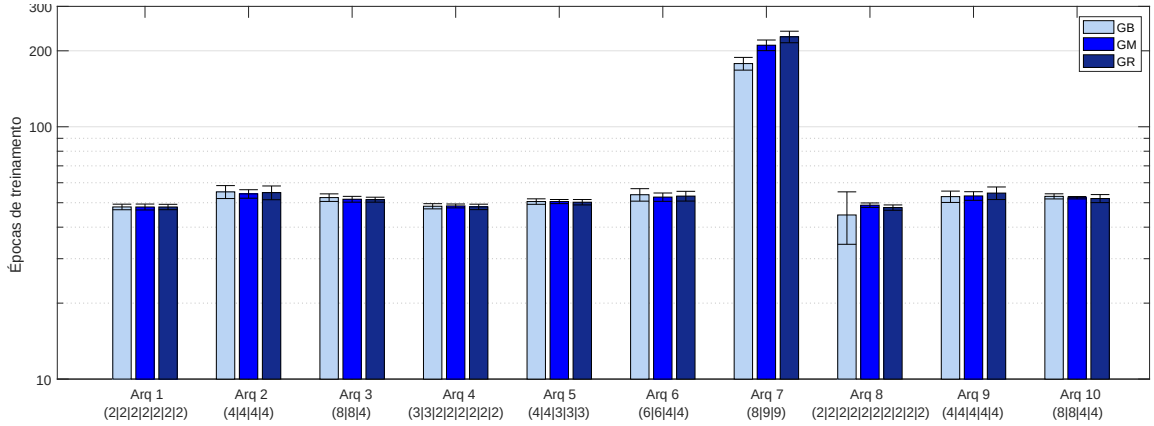
Com relação ao desempenho das arquiteturas, é possível observar que as arquiteturas com maior tamanho de tupla consomem um número de épocas ligeiramente superior às arquiteturas com menor tamanho de tupla. Isto se torna claro para o caso da arquitetura Arq7, a qual consumiu um número muito maior de épocas que as demais arquiteturas.




(a) 10 amostras




(b) 50 amostras



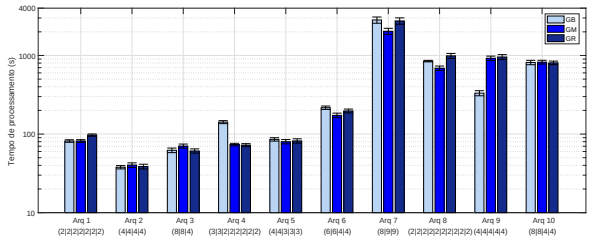

(c) 100 amostras

Figura 27: Total de épocas no treinamentos usando diferentes números de amostras por classe

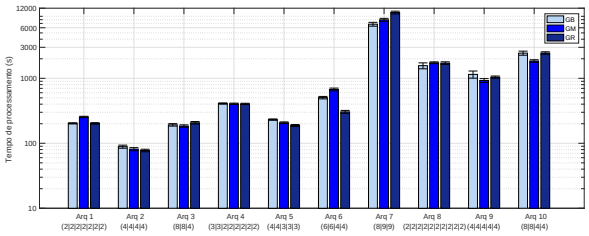
As Figuras 27(b) e 27(c) ilustram o número médio de épocas de treinamento necessárias para cada arquitetura completar a fase de treinamento utilizando os conjuntos de amostras com 50 e 100 amostras por classe, respectivamente.

Observando as Figuras 5.27(b) e 5.27(c), nota-se que o número de épocas necessário para treinar a rede pouco varia com o aumento do número de classes. Adicionalmente, o número de épocas é bastante similar para as arquiteturas simuladas, com exceção da arquitetura Arq7. Assim, além de apresentar uma acurácia relativamente baixa quando comparada às demais arquiteturas simuladas, a arquitetura Arq7 apresentou número de épocas significativamente maior. Dessa forma, para uma rede Mod-MPLN utilizada no reconhecimento de imagens de múltiplas classes, recomenda-se utilizar tuplas maiores no estágio de entrada e tuplas menores nos estágios mais profundos.

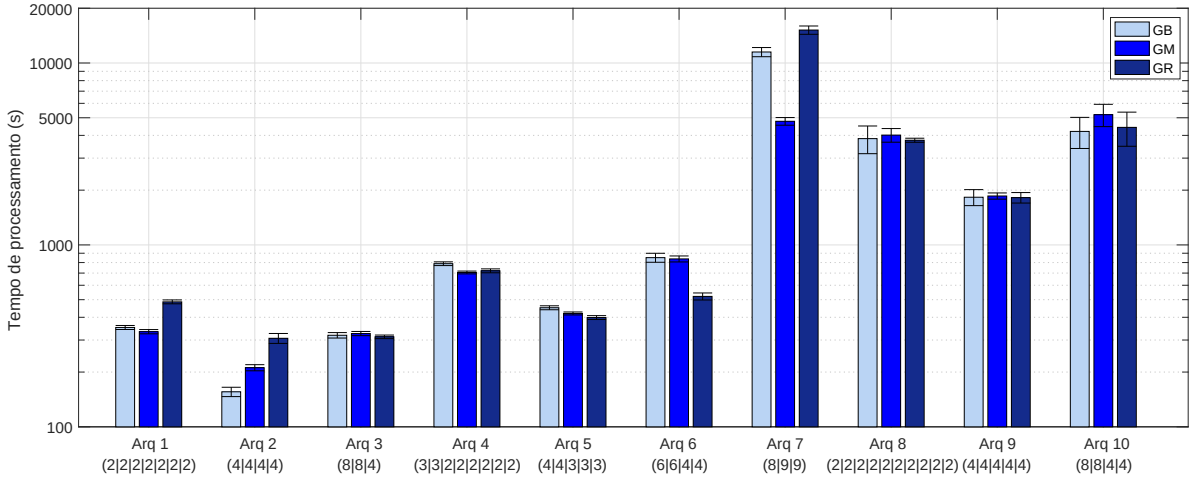
Por fim, será analisado o tempo médio gasto pelas arquiteturas para realizar o treinamento e classificação, utilizando os conjuntos de treinamento de 10, 50 e 100 amostras por classe. A Figura 28(a) ilustra o tempo médio gasto por cada arquitetura para completar a fase de treinamento e teste para conjuntos de treinamento com 10 amostras por classe.




(a) 10 amostras




(b) 50 amostras




(c) 100 amostras

Figura 28: Tempo médio de treinamento e teste usando diferentes números de amostras por classe

Nota-se que o tempo de processamento é substancialmente maior nas arquiteturas configuradas para identificar imagens com maior tamanho em *pixels*, uma vez que é necessário maior poder computacional nas simulações. Com relação às arquiteturas configuradas para endereçar a imagem com 16x16 *pixels*, a arquitetura Arq1 obteve o maior tempo de processamento. Apesar de Arq1 ser a arquitetura que requer menos memória RAM, para efeitos de simulação, esta é aquela que requer o maior poder de processamento. Isso se deve ao fato de a rede possuir muitos neurônios, sendo que para cada neurônio, o *script* do MATLAB® desenvolvido utiliza funções para realizar conversão decimal-binário e binário-decimal, as quais exigem considerável poder computacional.

Com relação às arquiteturas configuradas para endereçar a imagem de 24x24 *pixels*, o gráfico de tempo aumenta com o tamanho da tupla, em especial para a arquitetura Arq7. Entende-se que a arquitetura Arq7 apresentou um tempo de processamento muito superior às demais porque, além de possuir um tamanho de tupla superior, tal arquitetura levou consideravelmente mais épocas de treinamento, conforme pôde ser observado na Figura 27.

As Figuras 28(b) e 28(c) ilustram o tempo médio gasto por cada arquitetura para completar a fase de treinamento e teste com conjuntos de treinamento contendo 50 e 100 amostras por classe, respectivamente. Tais resultados são condizentes com os resultados de épocas de treinamento da Figura 27. Assim, pode concluir que, para problemas de reconhecimento de algarismos manuscritos, uma tupla de maior tamanho faz com que a rede consuma maior tempo de processamento. Isto é corroborado em (ALEKSANDER; MORTON, 1989; BRAGA; CARVALHO; LUDERMIR, 2000), onde é dito que tamanhos de tupla mais elevados aumentam o custo de memória e o poder computacional necessário para implementar a rede.

A partir da análise das 10 arquiteturas, configuradas para solucionar o problema de algarismos manuscritos, pode-se observar que as arquiteturas com maior tamanho de tupla obtiveram melhores resultados em termos de acurácia na classificação. Entretanto, estas mesmas arquiteturas necessitam de maior poder computacional para sua implementação. Dessa forma, a escolha do tamanho da tupla deve ser realizada levando em conta a acurácia requerida e a disponibilidade de recursos para a implementação da RNSP.

Adicionalmente, a análise destes resultados leva à recomendação de utilizar tuplas maiores no estágio de entrada da rede. Tal recomendação seria justificada pelo fato de que o estágio de entrada é aquele que recebe diretamente os bits dos padrões de entrada, enquanto os estágios subsequentes são endereçados pelas saídas dos estágios anteriores, identificando sub-combinações do padrão de entrada (BANDEIRA, 2010). Dessa forma, os resultados mostram que o estágio de entrada é aquele que precisa possuir maior capacidade de aprendizado, uma vez que é diretamente endereçado pelos padrões de entrada.

#### 5.2.2.2 Impacto da resolução da memória para rede Mod-MPLN

Da mesma forma como realizado na Seção 5.1.2, para analisar o impacto da variação da resolução da memória dos neurônios da RNSP Mod-MPLN, é preciso variar o parâmetro $\omega$. Conforme visto na Seção 3.1.3, o aumento de $\omega$ torna mais difícil de uma posição de memória do neurônio retornar para a posição 'u'. Dessa forma, a rede se torna mais resiliente a ruídos, uma vez que uma única amostra ruidosa não será capaz de eliminar completamente o conhecimento adquirido pelo neurônio a respeito do problema a ser solucionado. Por outro lado, o aumento de $\omega$ aumenta o consumo de memória da rede, uma vez que serão necessários mais bits para representar cada posição de memória dos neurônios.

Conforme conhecido da literatura (MYERS, 1989), o principal interesse em utilizar um valor de $\omega$ elevado reside na tolerância a ruídos na amostra de treinamento. Dessa forma, os conjuntos de sinais de bits foram modificados para inserir ruídos nos mesmos, conforme explicado na Seção 4.5. As simulações realizadas para aferir o impacto da variação de $\omega$ foram realizadas para as arquiteturas 1 a 3 da Tabela 3, e para os grupos de treinamento GB, GM e GR contendo 100 amostras por classe. A Tabela 11 ilustra os parâmetros mantidos constantes na simulação.

Tabela 11: Parâmetros constantes durante a variação das arquiteturas

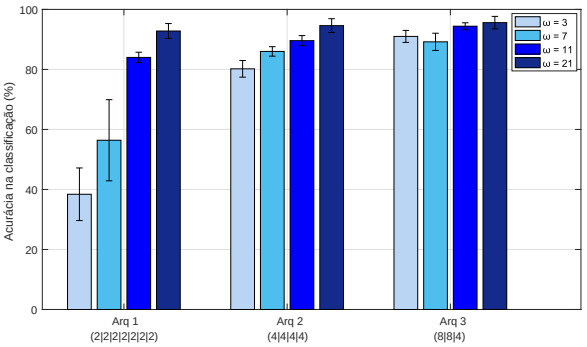
| **Parâmetro** | **Valor** |
|---|---|
| $k$ | 1000 |
| $\eta$ | 0,1 |

A Figura 29(a) mostra o desempenho das arquiteturas testadas em termos da acurácia durante a fase de teste utilizando o grupo de imagens de qualidade boa GB.

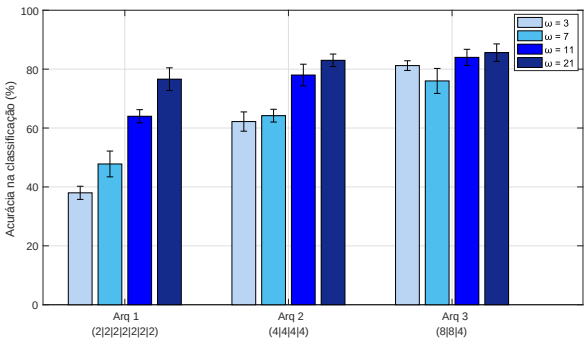
Conforme pode ser observado, a acurácia da rede aumenta com o aumento de $\omega$. Isto é principalmente observado na arquitetura Arq1, a qual possui tupla de 2 bits em todos os estágios. Isto ocorre uma vez que um tamanho de tupla menor tem por consequência neurônios com poucas posições de memória para guardar informações a respeito do problema. Assim, estas poucas posições de memória serão acessadas diversas vezes durante o treinamento, e o neurônio não será capaz de discriminar a quantidade de acessos em cada posição de memória, uma vez que todas terão sido ajustadas para os valores máximos de probabilidade 0% ou 100%. Tal efeito indesejado é análogo à saturação da rede WiSARD.

Por outro lado, a resolução da memória $\omega$ mitiga o problema de saturação nas redes MPLN e Mod-MPLN. Um valor alto de $\omega$ permite que cada neurônio consiga discriminar entre as posições de memória que foram acessadas diversas vezes durante o treinamento daquelas que foram acessadas apenas ocasionalmente. Tal característica é especialmente importante para rede Mod-MPLN, uma vez que o discriminador da rede considera o valor de probabilidade associado ao conteúdo da memória de cada um dos neurônios acessados pela padrão a ser classificado.

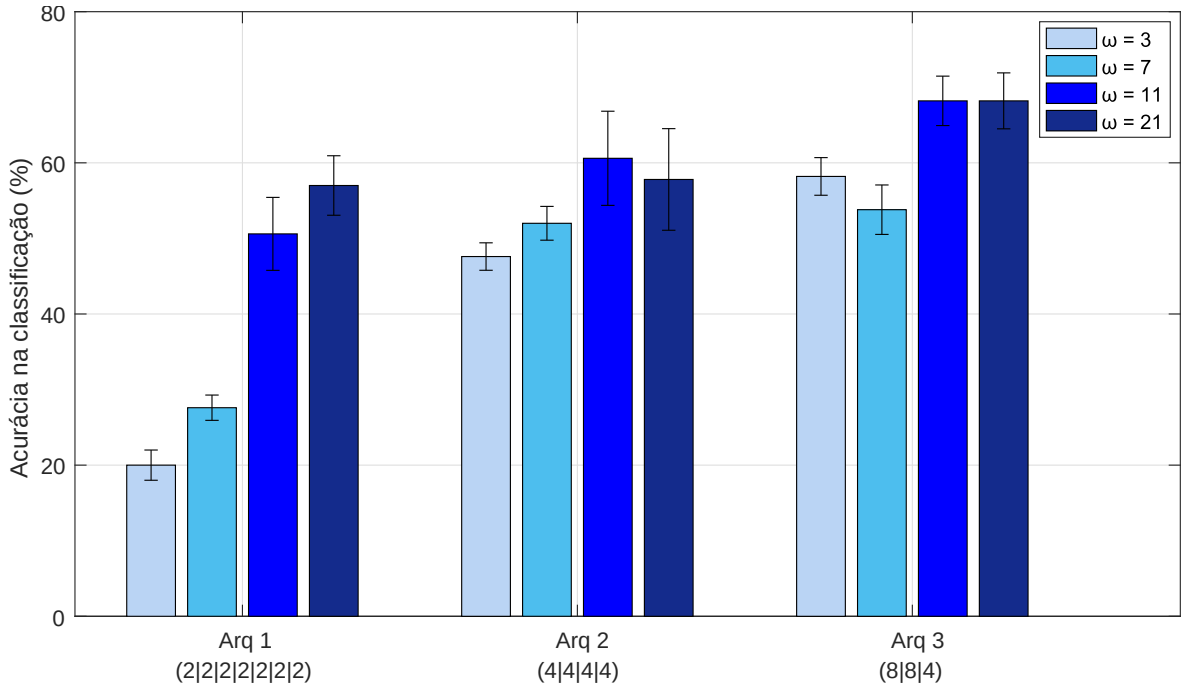
Na Figura 29(b), é mostrado o desempenho das arquiteturas testadas em termos da acurácia durante a fase de teste utilizando o grupo de imagens de qualidade média GM. Observa-se que os resultados são consistentes com os resultados da Figura 29(a). No




(a) Grupo de amostras GB




(b) Grupo de amostras GM




(c) Grupo de amostras GR

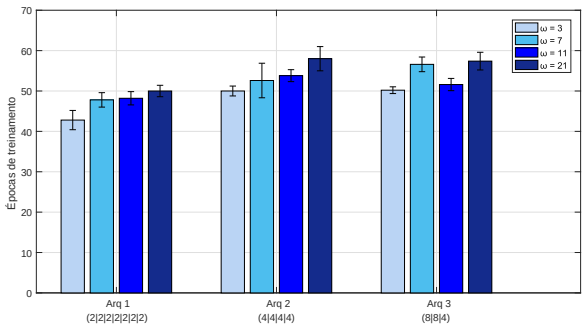
Figura 29: Acurácia na classificação para as arquiteturas com diversos $\omega$ para os diferentes grupos com 100 amostras por classe e 10% de ruído

entanto, para imagens de pior qualidade a rede apresenta resultados resultados consideravelmente piores para valores de $\omega = 3$ e $\omega = 7$ na arquitetura Arq2.

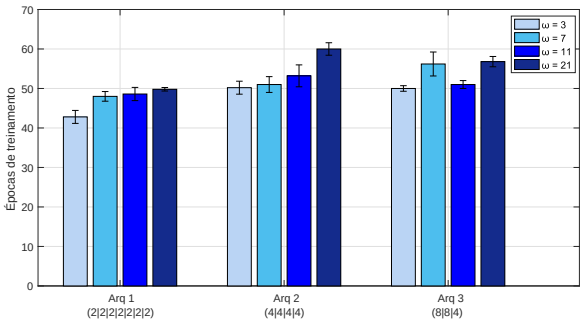
Já na Figura 29(c), é mostrado o desempenho das arquiteturas testadas em termos da acurácia para o grupo de imagens de qualidade ruim GR. Para o conjunto de imagens com a pior qualidade, é possível observar que até mesmo a arquitetura com tamanho de 8 bits apresenta resultados consideravelmente inferiores para valores baixos de $\omega$. Dessa forma, pode-se concluir que, em problemas onde há presença de ruídos na imagem, a tendência da rede a saturação é influenciada por dois fatores principais: o tamanho da

tupla e a resolução da memória $\omega$. Além disso, quanto pior a qualidade das imagens, pior é o desempenho das arquiteturas com baixa resolução da memória.

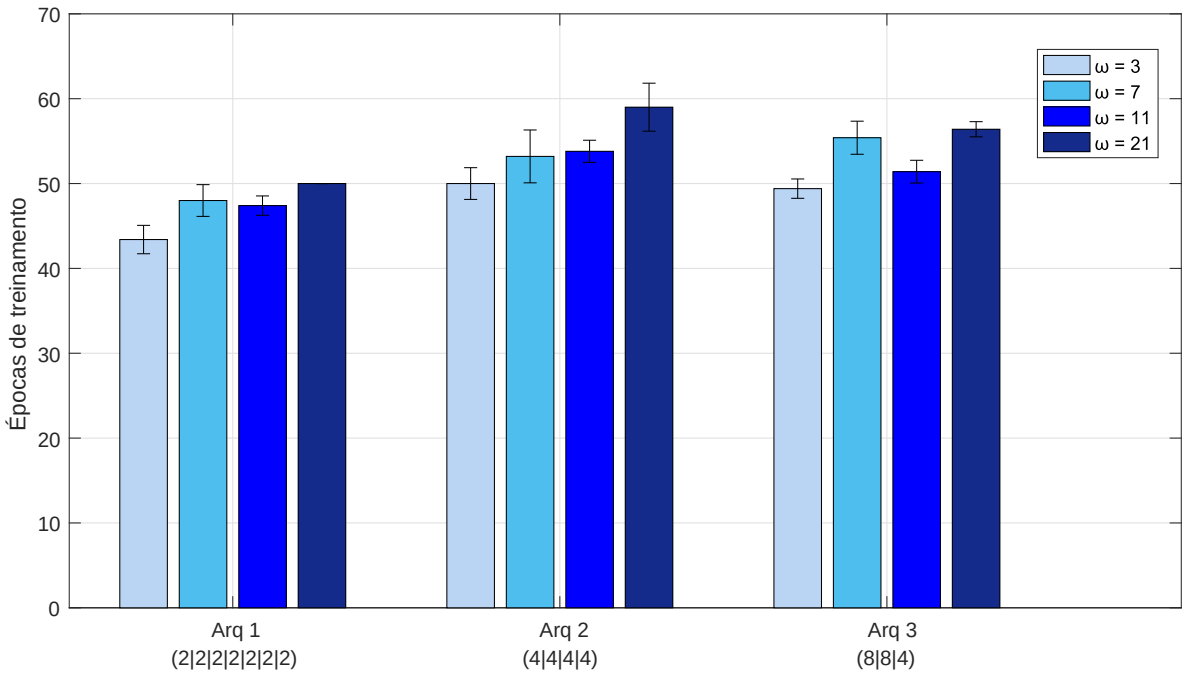
A seguir, será analisado o impacto da resolução da memória $\omega$ em termos de épocas de treinamento necessárias para a convergência da rede, utilizando os três grupos de imagem de qualidades distantes GB, GM e GR. A Figura 30(a) mostra o desempenho das arquiteturas testadas em termos de épocas de treinamento para o grupo de amostras GB.




(a) Grupo de amostras GB




(b) Grupo de amostras GM




(c) Grupo de amostras GR

Figura 30: Número de épocas de treinamento com diversos $\omega$ para os diferentes grupos com 100 amostras por classe e 10% de ruído

É possível observar que o número de épocas de treinamento aumenta ligeiramente com o aumento da resolução da memória $\omega$. Tal resultado é esperado uma vez que,

mantendo a taxa de aprendizado da rede $\eta$ constante, quando maior $\omega$, mais iterações serão necessárias até que as posições de memória acessadas atinjam os valores de probabilidade 0% ou 100%.

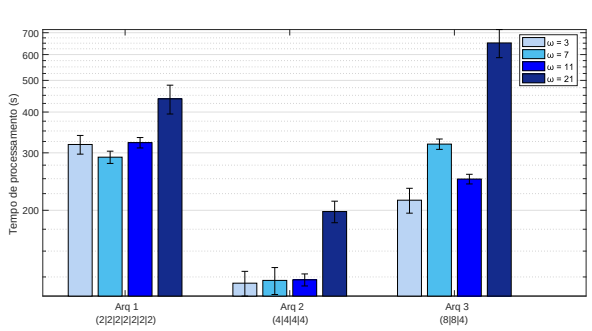
As Figuras 30(b) e 30(c) mostram o desempenho das arquiteturas testadas em termos de épocas de treinamento para os grupos GM e GR, respectivamente. É possível observar que a qualidade das imagens de treinamento possui pouco impacto no número de épocas de treinamento necessárias para a convergência da rede. Além disso, nota-se que a resolução da memória $\omega$ apresentou impacto pouco significativo no número de épocas de treinamento das arquiteturas da rede.

Por fim, será analisado o impacto da resolução da memória $\omega$ em termos de tempo de processamento durante a fase de treinamento e teste das arquiteturas da rede. A Figura 31(a) mostra o desempenho das arquiteturas testadas em termos do tempo de processamento.

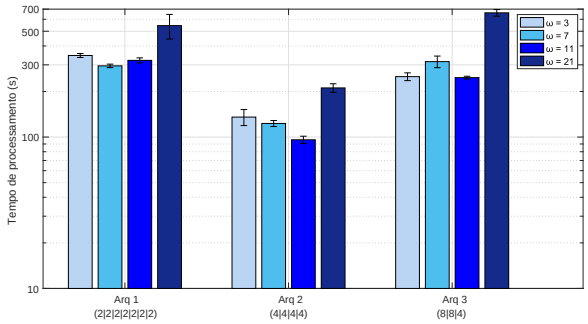
Conforme pode ser observado, a resolução da memória possui impacto significativo no tempo de processamento da rede, em especial para os valores de $\omega = 21$. Nota-se que a arquitetura Arq1 obteve os maiores tempos de simulação para os valores baixos de $\omega$, ao passo que obteve valores mais equilibrados. Isso se deve ao fato de a rede possuir muitos neurônios, sendo que para cada neurônio, o *script* do MATLAB® desenvolvido utiliza funções para realizar conversão decimal-binário e binário-decimal, as quais exigem considerável poder computacional. Assim, em termos de simulação, as arquiteturas com muitos neurônios apresentaram tempo de simulação muito altos. Para o caso de uma implementação em *hardware*, como o processamento das informações seria realizado em paralelo por cada neurônio, tais tempos de processamento seriam muito menores.

As Figuras 31(b) e 31(c) mostram o desempenho das arquiteturas testadas em termos de tempo de processamento para o grupo GM e GR, respectivamente. Pode-se notar que o tempo de processamento é pouco influenciado pela qualidade dos conjuntos de dados. Desse modo, para estes casos o tempo de processamento também aumenta com o aumento de $\omega$. A Tabela 12 mostra uma análise comparativa entre o desempenho das arquiteturas para os diferentes valores de $\omega$ para o caso de amostras com ruído de 10%.

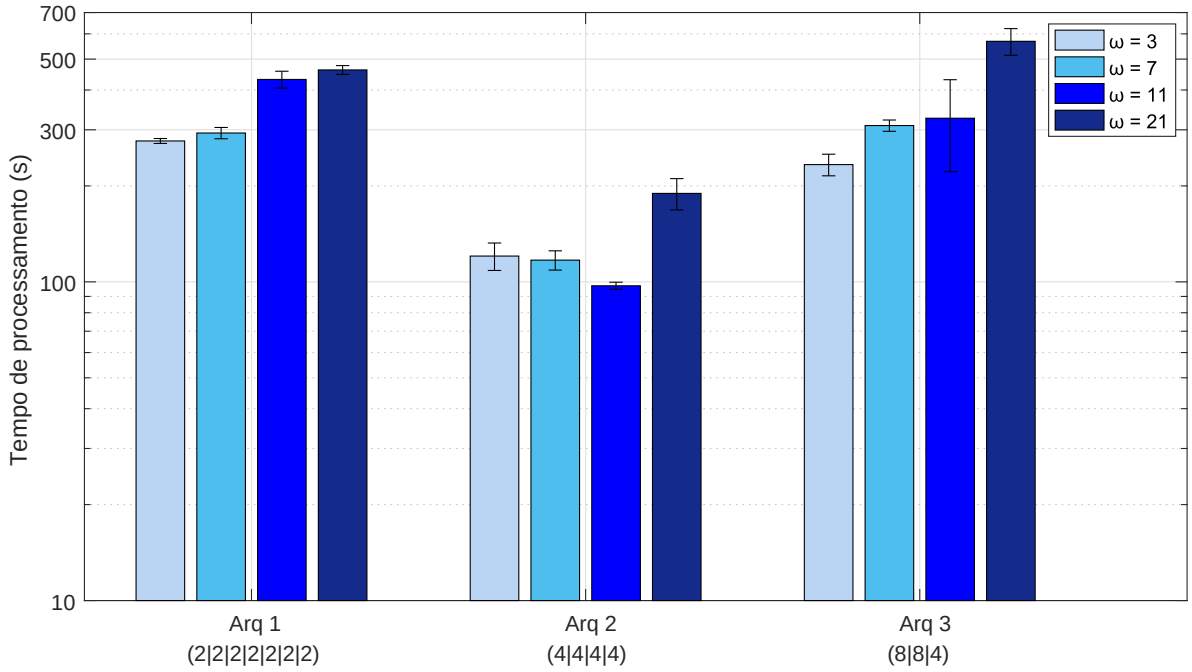
Sendo assim, pode-se concluir que, para valores de ruído de 10% nos conjuntos de entrada GB, GM e GR, a acurácia na classificação da rede tende a aumentar com o aumento de $\omega$. Entretanto, os resultados para $\omega = 21$ foram apenas ligeiramente superiores




(a) Grupo de amostras GB




(b) Grupo de amostras GM




(c) Grupo de amostras GR

Figura 31: Tempo de processamento médio com diversos $\omega$ para os diferentes grupos com 100 amostras por classe e 10% de ruído

que os resultados para $\omega = 11$. Por outro lado, o tempo de processamento para $\omega = 21$ foi consideravelmente superior ao tempo de processamento das arquiteturas com $\omega = 11$. Assim, para o problema de classificação de imagens com ruído de 10%, o valor de $\omega = 11$ torna-se o mais recomendado. Este valor é condizente com a recomendação de (MYERS, 1989), a qual afirma que maiores tamanhos de $\omega$ aumentam a tolerância da rede a ruídos, e cita que o valor de $\omega = 11$ apresenta resultados satisfatórios.

A seguir, os mesmos testes em termos de análise do impacto da resolução da memória $\omega$ serão realizados para o conjunto de dados com ruído de 15%. Isto significa que cada um dos grupos GB, GM e GR apresenta 15% de amostras pertencentes à classe

Tabela 12: Comparativo entre as arquiteturas para diferentes valores de $\omega$ com ruído de 10%

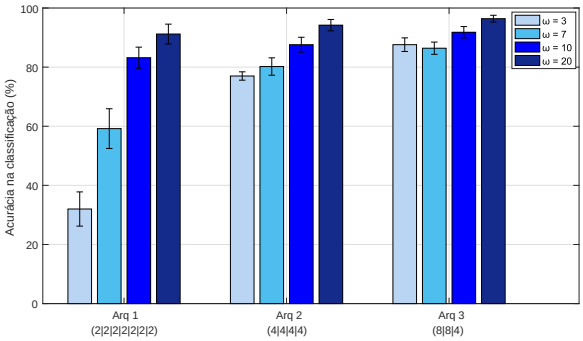
| | | Resolução da memória | | | |
|---|---|---|---|---|---|
| **Arq.** | **Resultado** | $\omega = 3$ | $\omega = 7$ | $\omega = 11$ | $\omega = 21$ |
| **Arq1** | **Acurácia** | pior | regular | bom | melhor |
| | **Épocas** | melhor | bom | bom | pior |
| | **Tempo** | bom | bom | regular | pior |
| **Arq2** | **Acurácia** | pior | regular | bom | melhor |
| | **Épocas** | melhor | bom | bom | pior |
| | **Tempo** | regular | regular | bom | pior |
| **Arq3** | **Acurácia** | bom | pior | melhor | melhor |
| | **Épocas** | melhor | pior | bom | pior |
| | **Tempo** | melhor | regular | bom | pior |

incorreta. A Figura 32(a) mostra o desempenho das arquiteturas testadas em termos da acurácia durante a fase de teste utilizando o grupo de imagens de qualidade boa GB.

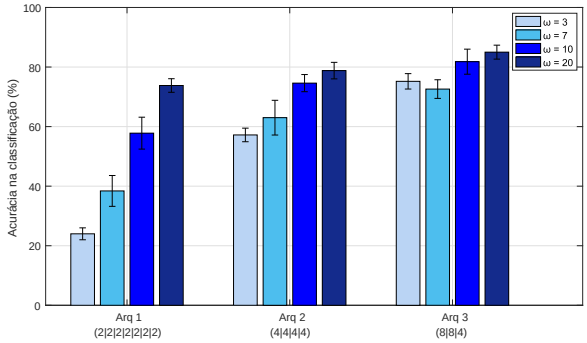
Analogamente aos resultados obtidos para os conjuntos de treinamento com 10% de ruído, a acurácia da rede aumenta com o aumento de $\omega$. Isto é principalmente observado na arquitetura Arq1, a qual possui tupla de 2 bits em todos os estágios. Conforme discutido, um tamanho de tupla menor tem por consequência neurônios com poucas posições de memória para guardar informações a respeito do problema, o que causa a saturação da rede. É possível observar ainda que a rede Mod-MPLN obtém resultados satisfatórios para todos os valores de $\omega$ nas arquiteturas Arq2 e Arq3.

A Figura 32(b) mostra o desempenho das arquiteturas testadas em termos da acurácia durante a fase de teste para o grupo GM com 15% de amostras ruidosas. Observa-se que os resultados são consistentes com os resultados da Figura 29. Quanto pior a qualidade das imagens, mais dependente de um valor alto de $\omega$ a rede se torna.

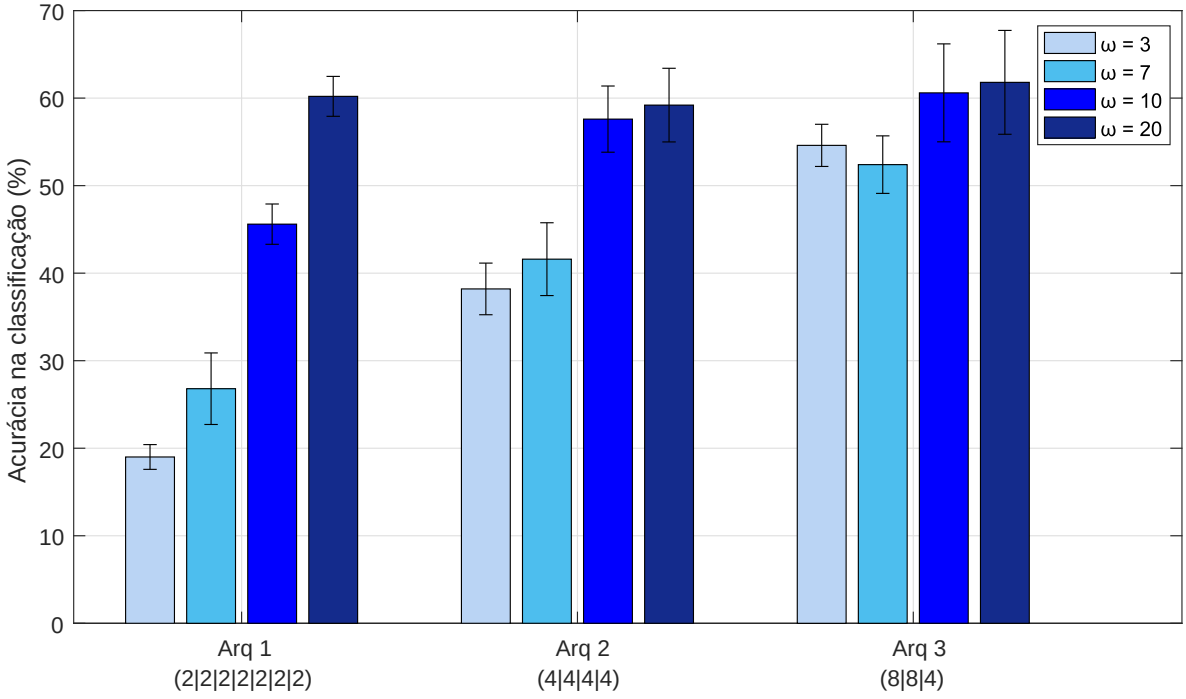
Já a Figura 32(c) mostra o desempenho das arquiteturas testadas em termos da acurácia durante a fase de teste para o grupo GR com 15% de amostras ruidosas. Os resultados para o conjunto de imagens com a pior qualidade são consistentes com os resultados anteriores na análise do impacto da resolução da memória. Valores maiores de $\omega$ tornam a rede mais resistente a amostras ruidosas. Nota-se que até mesmo a arquitetura com tamanho de 8 bits apresenta resultados consideravelmente inferiores para valores baixos de $\omega$. Observa-se novamente que, em problemas onde há presença de ruídos na imagem, a tendência da rede a saturação é influenciada por dois fatores principais: o tamanho da tupla e a resolução da memória $\omega$.




(a) Grupo de amostras GB




(b) Grupo de amostras GM

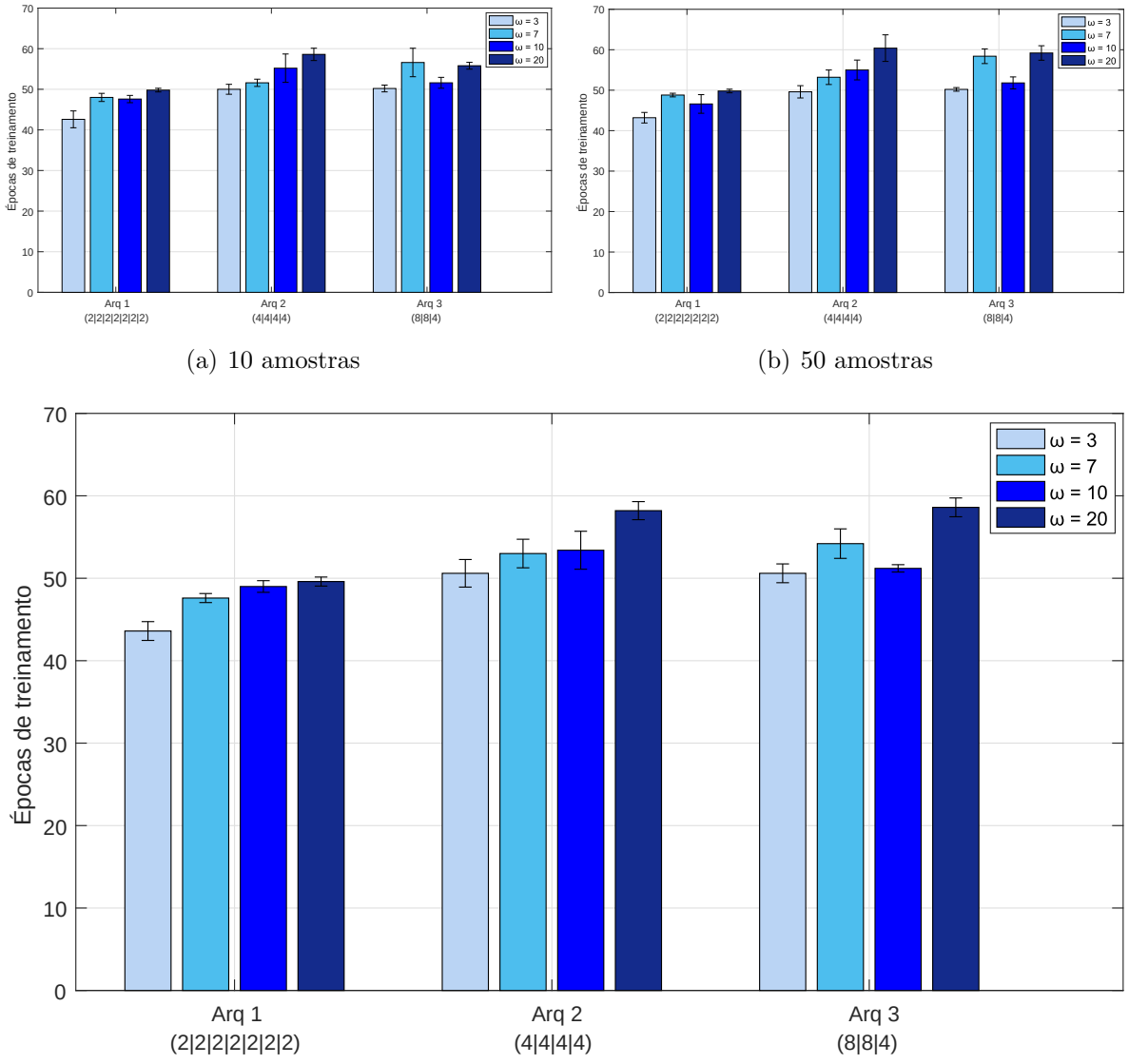



(c) Grupo de amostras GR

Figura 32: Acurácia na classificação para as arquiteturas com diversos $\omega$ para os diferentes grupos com 100 amostras por classe e 15% de ruído

A seguir, será analisado o impacto da resolução da memória $\omega$ em termos de épocas de treinamento necessárias para a convergência da rede, utilizando os três grupos de imagem de qualidades distantes GB, GM e GR. A Figura 33(a) mostra o desempenho das arquiteturas testadas em termos de épocas de treinamento para o grupo GB com 15% de amostras ruidosas.

Assim como na análise do números de épocas de treinamento para amostras com 10 % de ruído, o número de épocas de treinamento aumenta ligeiramente com o aumento da resolução da memória $\omega$. Para valores maiores $\omega$, mais iterações serão necessárias até




(a) 10 amostras

(b) 50 amostras

(c) 100 amostras

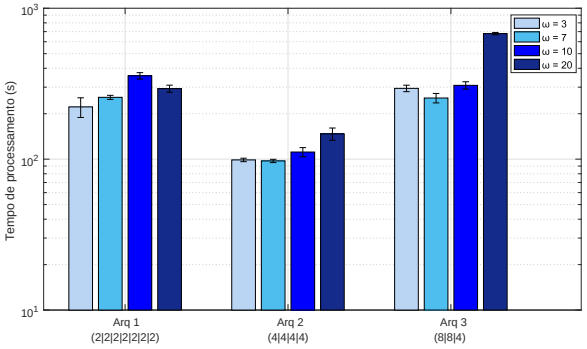
Figura 33: Número de épocas de treinamento com diversos $\omega$ para os diferentes grupos com 100 amostras por classe e 15% de ruído

que as posições de memória acessadas atinjam os valores de probabilidade 0% ou 100%, o que aumenta as épocas de treinamento utilizadas durante o treinamento da rede.

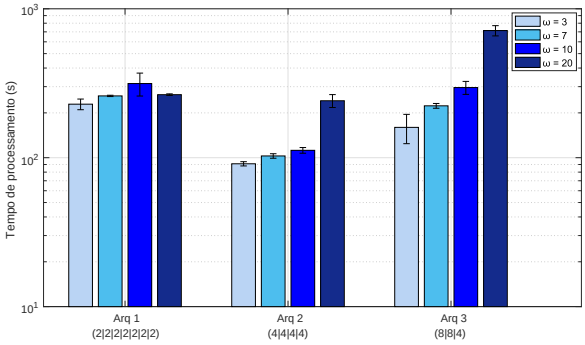
As Figuras 33(b) e 33(c) mostram o desempenho das arquiteturas testadas em termos de épocas de treinamento para o grupo GM e GR com 15% de amostras ruidosas, respectivamente. Assim como para o caso das amostras com 10% de ruído, a qualidade das imagens de treinamento e a resolução da memória $\omega$ possuem pouca influência no número de épocas de treinamento da rede Mod-MPLN.

Por fim, será analisado o impacto da resolução da memória $\omega$ em termos de tempo de processamento durante a fase de treinamento e teste das arquiteturas da rede. A

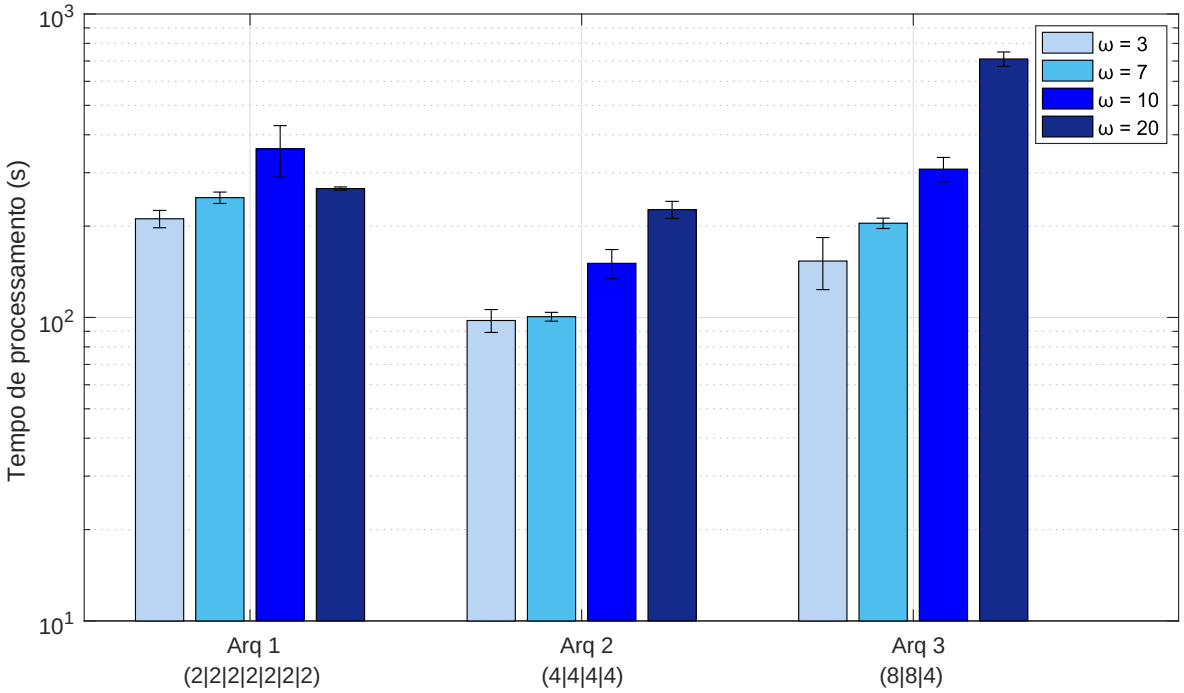
Figura 34(a) mostra o desempenho das arquiteturas testadas em termos de tempo de processamento necessário para concluir fase de treinamento e teste para o grupo GB com 15% de amostras ruidosas.




(a) 10 amostras




(b) 50 amostras




(c) 100 amostras

Figura 34: Tempo de processamento médio com diversos $\omega$ para os diferentes grupos com 100 amostras por classe e 15% de ruído

Assim como discutido nas análises do impacto de $\omega$ no tempo de processamento para amostras com 10% de ruído, a resolução da memória possui impacto considerável no tempo de processamento da rede para os valores de $\omega = 21$. Além disso, assim como visto na Seção 5.2.2.1, o tamanho da tupla também impacta de forma diretamente proporcional ao tempo de processamento nas fases de treinamento e teste.

Nas Figuras 34(b) e 34(c), pode-se observar o desempenho das arquiteturas testadas em termos de tempo de processamento necessário para concluir fase de treinamento e teste com 15% de amostras ruidosas para os grupos GM e GT, respectivamente. Pode-se notar que o tempo de processamento é pouco influenciado pela qualidade dos conjuntos de dados. Desse modo, para estes casos o tempo de processamento também aumenta com o aumento de $\omega$ e com o tamanho da tupla. Novamente, a arquitetura Arq1 apresentou tempo de simulação elevado devido a questões de simulação. A Tabela 13 mostra uma análise comparativa entre o desempenho das arquiteturas para os diferentes valores de $\omega$ para o caso de amostras com ruído de 15%.

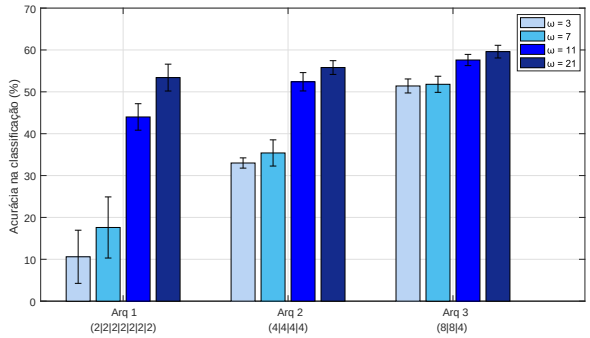
Tabela 13: Comparativo entre as arquiteturas para diferentes valores de $\omega$ com ruído de 15%

| | | **Resolução da memória** | | | |
|---|---|---|---|---|---|
| **Arq.** | **Resultado** | $\omega = 3$ | $\omega = 7$ | $\omega = 11$ | $\omega = 21$ |
| **Arq1** | **Acurácia** | pior | regular | bom | melhor |
| | **Épocas** | melhor | bom | bom | pior |
| | **Tempo** | melhor | bom | pior | regular |
| **Arq2** | **Acurácia** | pior | regular | bom | melhor |
| | **Épocas** | melhor | bom | bom | pior |
| | **Tempo** | melhor | bom | regular | pior |
| **Arq3** | **Acurácia** | bom | pior | melhor | melhor |
| | **Épocas** | melhor | pior | bom | pior |
| | **Tempo** | melhor | bom | regular | pior |

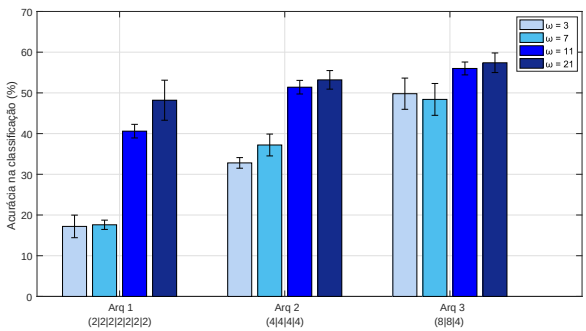
A última etapa das simulações foi realizada para analisar o impacto da resolução da memória $\omega$ para o conjunto de dados com ruído de 20%. Isto significa que cada um dos grupos GB, GM e GR apresenta 20% de amostras pertencentes à classe incorreta. A Figura 35(a) mostra o desempenho das arquiteturas testadas em termos da acurácia durante a fase de teste utilizando o grupo de imagens de qualidade boa GB.

Assim como nos demais casos, a acurácia da rede aumenta com o aumento de $\omega$. Isto é principalmente observado na arquitetura Arq1, a qual possui tupla de 2 bits em todos os estágios. Para valores ainda maiores de ruído, tamanhos de tupla menores também resultam em neurônios com poucas posições de memória para guardar informações a respeito do problema.

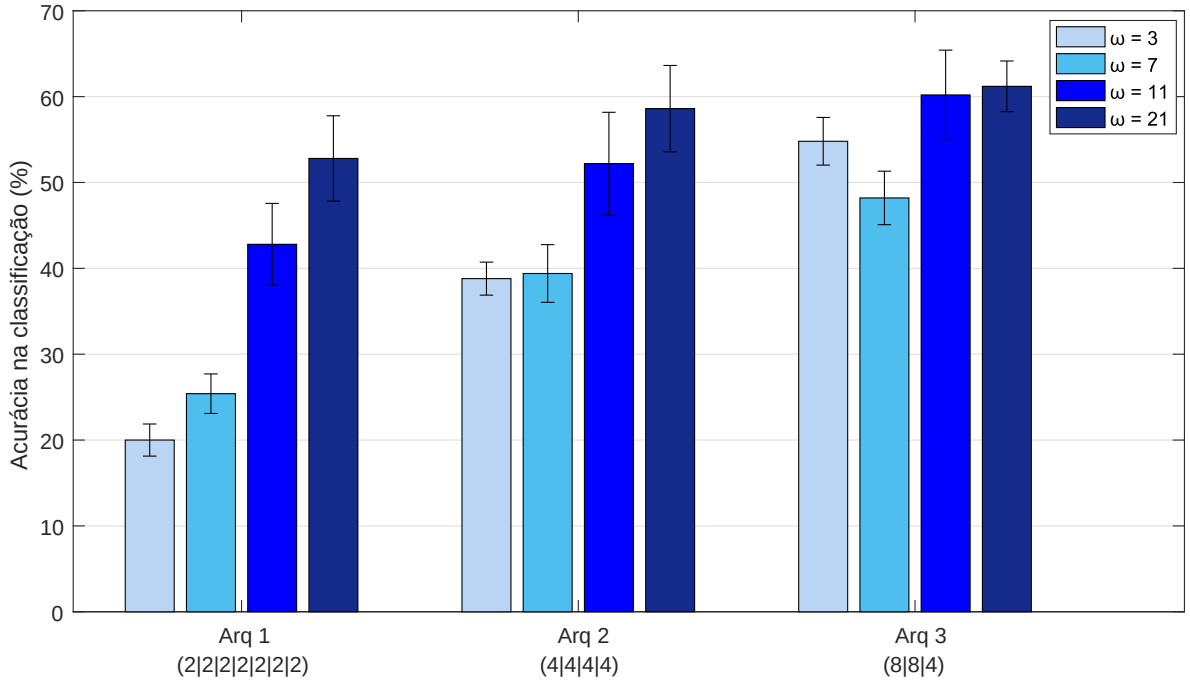
Na Figura 35(b), é mostrado desempenho das arquiteturas para o grupo GM. Observa-se que os resultados são consistentes com os resultados da Figura 35(a). Quanto pior a qualidade das imagens, mais dependente de um valor alto de $\omega$ a rede se torna.




(a) Grupo de amostras GB




(b) Grupo de amostras GM




(c) Grupo de amostras GR

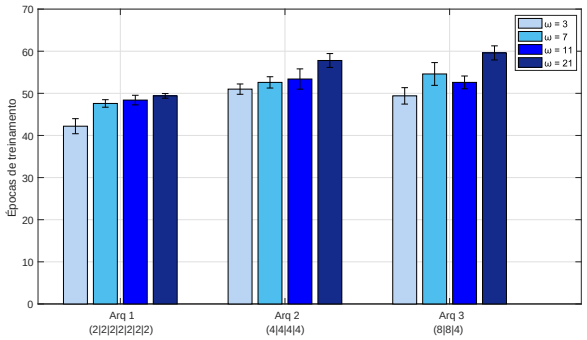
Figura 35: Acurácia na classificação para as arquiteturas com diversos $\omega$ para os diferentes grupos com 100 amostras por classe e 20% de ruído

Já a Figura 35(c) mostra o desempenho das arquiteturas testadas em termos da acurácia durante a fase de teste para o grupo GR. Os resultados para o conjunto de imagens com a pior qualidade são consistentes com os resultados anteriores na análise do impacto da resolução da memória. Valores maiores de $\omega$ tornam a rede mais resistente a amostras ruidosas. Nota-se que até mesmo a arquitetura Arq3 com tamanho de 8 bits apresenta resultados consideravelmente inferiores para valores baixos de $\omega$.

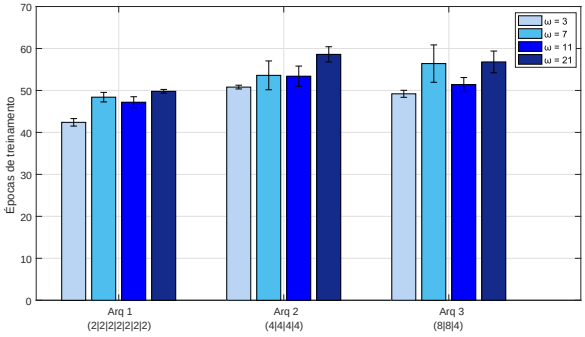
Para a análise da acurácia com amostras de treinamento com 20% de ruído, as arquiteturas com maior tamanho de tupla também são mais tolerantes a ruídos. Dessa forma, pode-se concluir que, em todos os testes onde há presença de ruídos na imagem,

a tendência da rede a saturação é influenciada por dois fatores principais: o tamanho da tupla e a resolução da memória $\omega$.

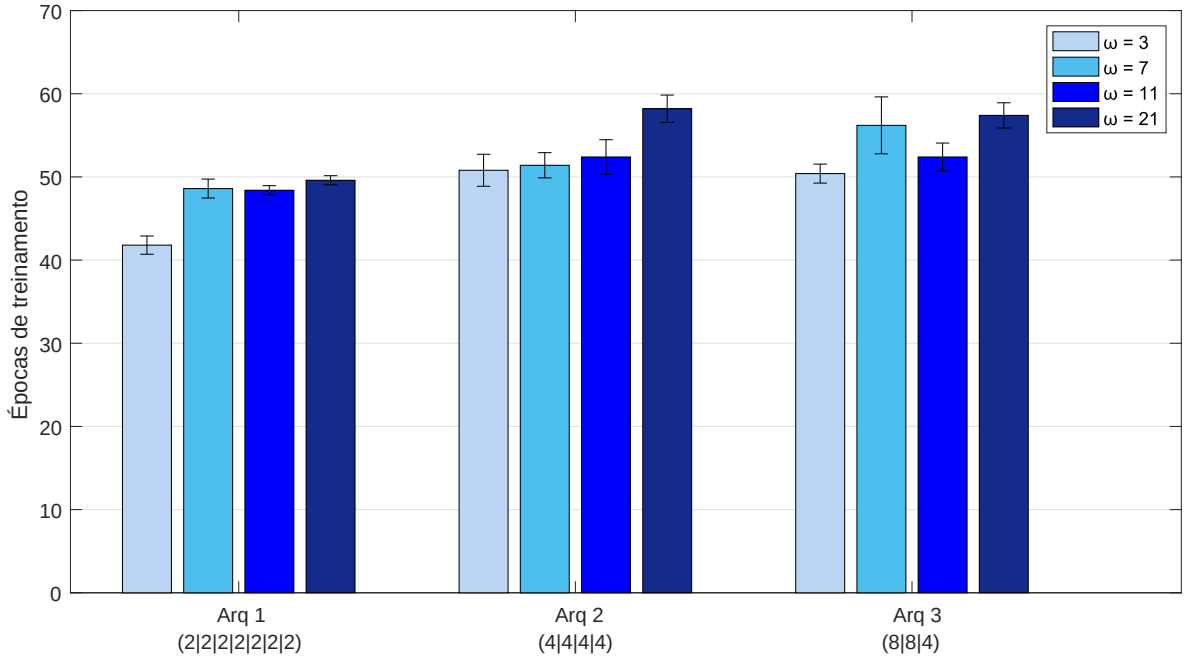
A seguir, será analisado o impacto da resolução da memória $\omega$ em termos de épocas de treinamento necessárias para a convergência da rede, utilizando os três grupos de imagem de qualidades distantes GB, GM e GR. As Figuras 36(a), 36(b) e 36(c) mostram o desempenho das arquiteturas testadas em termos de épocas de treinamento para os grupos GB, GM e GR com 20% de amostras ruidosas, respectivamente.




(a) Grupo de amostras GB




(b) Grupo de amostras GM



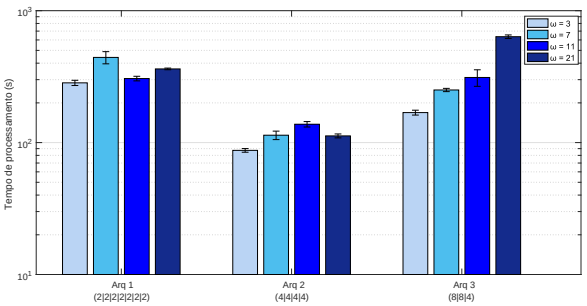

(c) Grupo de amostras GR

Figura 36: Número de épocas de treinamento com diversos $\omega$ para os diferentes grupos com 100 amostras por classe e 20% de ruído

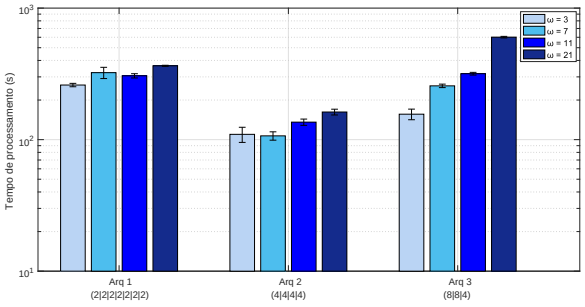
Assim como para o caso das amostras com 10% e 15% de ruído, o aumento da resolução da memória $\omega$ possui pouco impacto no número de épocas de treinamento. O

pequeno aumento é justificado uma vez que, quanto maior $\omega$, mais ajustes nos pesos de memória são necessários para modificar as probabilidades de resposta do valor 'u' para 0% e 100% .

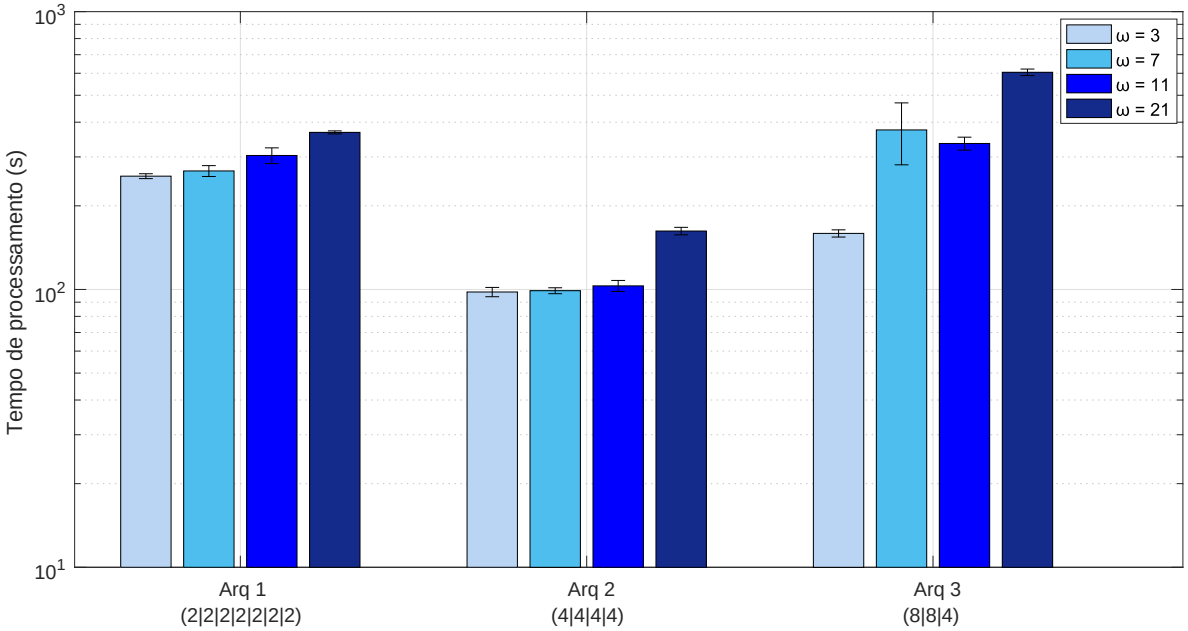
Por fim, será analisado o impacto da resolução da memória $\omega$ em termos de tempo de processamento durante a fase de treinamento e teste das arquiteturas da rede. A Figura 37(a) mostra o desempenho das arquiteturas testadas em termos de tempo de processamento necessário para concluir fase de treinamento e teste para o grupo GB com 20% de amostras ruidosas.




(a) Grupo de amostras GB




(b) Grupo de amostras GM




(c) Grupo de amostras GR

Figura 37: Tempo de processamento médio com diversos $\omega$ para os diferentes grupos com 100 amostras por classe e 20% de ruído

Analogamente aos testes em amostras com 10% e 15% de ruído, a resolução da memória possui impacto considerável no tempo de processamento da rede para os valores

mais altos de $\omega$. Além disso, assim como visto na Seção 5.2.2.1, o tamanho da tupla também impacta de forma diretamente proporcional no tempo de processamento nas fases de treinamento e teste.

Adicionalmente, ao observar as Figuras 37(b) e 37(c), pode-se notar que o tempo de processamento é pouco influenciado pela qualidade dos conjuntos de dados. Desse modo, para estes casos, o tempo de processamento também aumenta com o aumento de $\omega$ e com o aumento do tamanho da tupla. A Tabela 14 mostra uma análise comparativa entre o desempenho das arquiteturas para os diferentes valores de $\omega$ para o caso de amostras com ruído de 20%.

Tabela 14: Comparativo entre as arquiteturas para diferentes valores de $\omega$ com ruído de 20%

| | | **Resolução da memória** | | | |
|---|---|---|---|---|---|
| **Arq.** | **Resultado** | $\omega = 3$ | $\omega = 7$ | $\omega = 11$ | $\omega = 21$ |
| **Arq1** | **Acurácia** | pior | pior | bom | melhor |
| | **Épocas** | melhor | bom | bom | bom |
| | **Tempo** | melhor | bom | bom | pior |
| **Arq2** | **Acurácia** | pior | pior | bom | melhor |
| | **Épocas** | melhor | bom | bom | pior |
| | **Tempo** | melhor | bom | regular | pior |
| **Arq3** | **Acurácia** | regular | pior | melhor | melhor |
| | **Épocas** | melhor | regular | bom | pior |
| | **Tempo** | melhor | bom | regular | pior |

Conclui-se que, analogamente aos resultados para valores de ruído de 10% e 15% nos conjuntos de entrada GB, GM e GR, a acurácia na classificação da rede tende a aumentar com o aumento de $\omega$. Entretanto, os resultados para $\omega = 21$ foram apenas ligeiramente superiores aos resultados para $\omega = 11$, enquanto que o tempo de processamento para $\omega = 21$ foi consideravelmente superior ao tempo de processamento das arquiteturas com $\omega = 11$. Assim, para o problema de classificação de imagens com ruído de 20%, o valor de $\omega = 11$ torna-se o mais recomendado. Este valor é condizente com a recomendação de (MYERS, 1989), a qual afirma que maiores tamanhos de $\omega$ aumentam a tolerância da rede a ruídos, e cita que o valor de $\omega = 11$ apresenta resultados satisfatórios.

## 5.3 Considerações Finais

Neste capítulo, foi avaliado o impacto dos parâmetros de uma RNSP na acurácia da rede. Foram utilizados seis conjuntos de treinamento para avaliar os tamanhos de tupla

e arquitetura da rede, assim como foram utilizadas duas aplicações bastante distintas na avaliação das redes: o reconhecimento do bit de paridade em sequências de $n$ bits e o problema de reconhecimento de imagens de algarismos manuscritos.

Com relação ao problema da paridade de $n$ bits, a rede MPLN tradicional apresentou resultados satisfatórios para todas as arquiteturas testadas, contanto que haja um número de suficiente de amostras de treinamento. A partir da análise das 10 arquiteturas, configuradas para solucionar o problema de classificação do bit de paridade em sequências de 6, 8, 10 e 12 bits, foi possível observar que as arquiteturas com maior tamanho de tupla obtiveram melhores resultados para o grupo com diversas amostras de treinamento G3. Por outro lado, as arquiteturas com menor tamanho de tupla no estágio de entrada obtiveram melhores resultados para os grupo com amostras de treinamento limitadas G1 e G2.

Além disso, foi introduzido ruído nas amostras para observar o impacto da resolução da memória $\omega$ na capacidade da rede tolerar a amostras com ruído. A partir dos testes realizados, foi possível observar que os valores de $\omega = 7$ e $\omega = 11$ obtiveram os melhores resultados, conforme recomendação de (MYERS, 1989).

Adicionalmente, foram discutidas as limitações da rede MPLN tradicional para o problema de classificação com diversas classes, onde foi comprovado que a RNSP MPLN não apresenta resultados satisfatórios para problemas com mais de duas classes. Nesse sentido, a presente dissertação propôs uma modificação na forma de treinamento e reconhecimento da rede: a rede Mod-MPLN. A rede Mod-MPLN consiste em uma modificação da rede MPLN para problemas de múltiplas classes, através da mudança no algoritmo de treinamento da rede e pela inclusão de um discriminador específico na saída da rede, mas sem alterar as características intrínsecas da topologia MPLN. Foi comprovado que a rede Mod-MPLN apresenta resultados comparáveis à da rede WiSARD, os quais são superiores para conjuntos de treinamento com muitas amostras.

Com relação ao impacto dos parâmetros na rede Mod-MPLN, os maiores tamanhos da tupla apresentam melhor acurácia na rede. Entretanto, conforme já havia sido discutido em (ALEKSANDER; MORTON, 1989) e (BRAGA; CARVALHO; LUDERMIR, 2000), quanto maior a tupla, maior o consumo de memória e poder computacional necessário para implementação da rede. Dessa forma, o projetista deverá escolher um número de tupla dentro das limitações de projeto. No presente trabalho, onde imagens de tamanho

16x16 a 32x32 *pixels* foram utilizadas, uma tupla de 8 bits obteve, no geral, os melhores resultados em termos de acurácia de classificação.

É possível observar ainda que as redes MPLN e Mod-MPLN podem ser implementadas com tuplas de tamanho variável ao longo dos estágios da rede. Para os casos onde a tupla é variável ao longo dos estágios, idealmente as maiores tuplas devem estar nos estágios de entrada.

Não obstante, foi introduzido ruído nas amostras de algarismos manuscritos para observar o impacto da resolução da memória $\omega$ na capacidade da rede Mod-MPLN tolerar a presença amostras com ruído. A partir dos testes realizados, foi possível observar que os valores de $\omega = 11$ obtiveram os melhores resultados, conforme recomendação de (MYERS, 1989).

# Capítulo 6

# CONCLUSÕES E TRABALHOS FUTUROS

NESTE capítulo são apresentadas as principais conclusões acerca do impacto na variação de parâmetros de projetos de uma RNSP do tipo MPLN, bem como a modificação proposta para esta topologia de RNSP, a rede Mod-MPLN. Adicionalmente, são apresentadas algumas possíveis melhorias, assim como algumas possibilidades de trabalhos futuros.

## 6.1 Conclusões

Esta dissertação abordou o problema da escolha de parâmetros durante o projeto de uma RNSP do tipo MPLN. O trabalho desenvolvido implementou, por meio de simulação, diversas arquiteturas de RNSP do tipo MPLN e Mod-MPLN de modo a analisar o impacto dos parâmetros da rede na eficiência e eficácia da rede.

Para tal, foram utilizadas duas aplicações para treinamento e teste das RNSP MPLN: a classificação do bit de paridade e a identificação de algarismos manuscritos. Para a aplicação de classificação do bit de paridade, foram desenvolvidos 4 conjuntos de dados e 10 arquiteturas da rede MPLN, de modo a analisar o impacto da variação do tamanho da tupla de bits e do número de estágios da rede nas redes. Adicionalmente, foi introduzido ruído nos conjuntos de treinamento, de modo a analisar o impacto da variação da resolução da memória $\omega$ em cada endereço de memória dos neurônios.

Adicionalmente, foram discutidas as limitações da rede MPLN tradicional para o problema de classificação com diversas classes, onde foi comprovado que a rede MPLN não apresenta resultados satisfatórios para problemas com mais de duas classes. Nesse sentido, a presente dissertação propôs uma modificação na forma de treinamento e reco-

nhecimento da rede: a rede Mod-MPLN. A rede Mod-MPLN consiste em uma modificação da rede MPLN para problemas de múltiplas classes, através da mudança no algoritmo de treinamento da rede e pela inclusão de um discriminador específico na saída da rede, mas sem alterar as características intrínsecas da topologia MPLN.

Não obstante, os resultados da rede MPLN para a aplicação de reconhecimento de algarismos foram comparados com os resultados obtidos por uma rede WiSARD convencional, de modo a analisar a eficiência e eficácia da rede proposta no problema de classificação de algarismos numéricos. Por fim, foi realizado uma análise entre os resultados obtidos pelas redes Mod-MPLN e WiSARD para os conjuntos de dados com e sem pré-processamento, de modo a analisar a dependência de ambas as redes de um pré-processamento prévio do conjunto de entrada.

Para a aplicação de reconhecimento de algarismos, foram desenvolvidos 6 conjuntos de dados e 10 arquiteturas da rede Mod-MPLN, de modo a analisar o impacto da variação do tamanho da tupla de bits e do número de estágios da rede nas redes. Adicionalmente, de forma análoga à aplicação do bit de paridade, foi introduzido ruído nos conjuntos de treinamento, de modo a analisar o impacto da variação da resolução da memória $\omega$ em cada endereço de memória dos neurônios.

Conforme visto no Capítulo 5, as arquiteturas de RNSP do tipo MPLN e Mod-MPLN com os maiores tamanhos de tupla obtiveram os melhores resultados de precisão na classificação tanto no problema de classificação de bit de paridade quanto na classificação dos algarismos manuscritos. Por outro lado, foi possível observar que, quanto maior o tamanho da tupla, maior o tempo de processamento durante a fase de treinamento e teste da rede.

Dentre os conjuntos de dados de algarismos numéricos, para as resoluções de 16x16, 24x24 e 32x32 *pixels*, todas as arquiteturas com tuplas variando de 2 a 9 bits obtiveram resultados satisfatórios. Estes resultados corroboram os ensinamentos de (BRAGA; CARVALHO; LUDERMIR, 2000), onde é descrito que as RNSP geralmente possuem tupla na faixa de 2 a 10. De todo modo, foi possível observar que as arquiteturas de RNSP com tupla de 8 bits apresentaram em geral os melhores resultados. Assim, pode-se concluir que os tamanhos de tupla 2 a 8 bits apresentam resultados satisfatórios na classificação de imagens de algarismos manuscritos. A escolha do tamanho da tupla dentre estes valores

deve ser feita em vista da precisão necessária para o problema a ser resolvido, assim como em vista do poder de processamento e memória disponíveis para a implementação da rede.

Além disso, foi possível validar que arquiteturas de RNSP do tipo MPLN e Mod-MPLN apresentam resultados satisfatórios mesmo quando o tamanho da tupla é variável ao longo dos estágios da rede. Portanto, pode-se concluir que a recomendação em (ALEKSANDER; MORTON, 1989) a respeito da Equação 7 não precisa ser necessariamente seguida.

Adicionalmente, a partir dos resultados da aplicação de classificação do bit de paridade, foi possível observar que, nos casos onde a tupla de bits é variável ao longo dos estágio da rede, a rede apresenta melhores resultados quando os menores tamanhos de tupla estão nos estágios de entrada. Conforme discutido na Seção 5.1.1, isto se deve ao fato de que o estágio de entrada recebe diretamente os bits dos padrões de entrada, enquanto os estágios subsequentes são endereçados pelas saídas dos estágios anteriores, identificando sub-combinações do padrão de entrada.

Por fim, na análise do impacto da presença de ruídos no conjunto de treinamento e sua relação com a resolução da memória $\omega$, foi possível comprovar que valores altos $\omega$ proporcionam uma tolerância a ruídos para a rede, conforme afirmado em (MYERS, 1989). Foi possível concluir ainda que, em todos os testes onde há presença de ruídos na imagem, a tendência da rede a saturação é influenciada por dois fatores principais: o tamanho da tupla e a resolução da memória $\omega$. Entretanto, valores altos de $\omega$ aumentam consideravelmente o tempo de processamento da rede. Para as duas aplicações testadas no presente trabalho, foi possível observar que os valores de $\omega = 7$ e $\omega = 11$ apresentam tolerância a ruídos satisfatória enquanto não impactam de forma muito prejudicial o tempo de processamento da rede. Portanto, no momento de projetar uma RNSP do tipo MPLN ou Mod-MPLN, sugere-se que tais valores sejam considerados.

Salienta-se ainda que as recomendações feitas na presente dissertação foram feitas com base nos resultados obtidos para as duas aplicações testadas: o problema da identificação do bit de paridade e o reconhecimento de algarismos numéricos. O primeiro é conhecido por ser um problema de difícil aprendizado, e possui apenas duas classes. O segundo, apesar de ser solucionável por RNSPs mais simples, como a rede WiSARD, é um problema que pode possuir muitas classes, e exige alto poder computacional. Dessa forma, as duas aplicações foram escolhidas por apresentarem características bastante distintas entre si.

Os resultados e conclusões obtidas foram comparadas com assertivas de trabalhos relacionados, de modo a subsidiar as recomendações já aferidas pela literatura. No entanto, é possível que para outras aplicações práticas, outros valores de parâmetros sejam mais indicados no projeto de uma rede MPLN ou Mod-MPLN. Dessa forma, a principal contribuição deste trabalho reside na análise do impacto dos principais parâmetros de projeto destas redes, provendo diretrizes de projeto a serem seguidas, ou que sejam o ponto de partida do projetista ao se deparar com um problema a ser solucionado por uma RNSP MPLN.

## 6.2 Trabalhos Futuros

A fim de contribuir com a análise realizada nesta dissertação, sugere-se que sejam analisadas outras aplicações para verificar se as conclusões tiradas nesta dissertação são válidas a qualquer aplicação. Adicionalmente, sugere-se realizar testes com outros tamanhos de tupla, em especial tamanhos superiores a 10 bits, os quais não foram testados neste trabalho devido a uma limitação de poder de processamento e memória dos computadores utilizados.

Adicionalmente, sugere-se que as topologias de RNSP do tipo MPLN sejam sintetizadas em *hardware* configurável, tal como uma FPGA (*Field Programmable Gate Array*). A partir da análise dos resultados em ambiente real, seria possível verificar se os resultados obtidos em simulação condizem com a realidade.

Como trabalho futuro, sugerimos que a análise realizada na presente dissertação seja ampliada a outras arquiteturas de RNSPs, tal como as RNSP do tipo pRAM, VG-RAM e GSN brevemente apresentadas como trabalhos relacionados no Capítulo 2. Assim, seria possível analisar o impacto da variação dos parâmetros de projeto destas configurações de RNSP, e possivelmente compará-los com as conclusões deste trabalho. Tal análise possibilitaria verificar a correspondência do impacto dos parâmetros comuns a ambas as configurações de RNSP.

# REFERÊNCIAS


AKYAMA, M. T. *Interpolação de imagens baseada em Clustering*. Dissertação (Mestrado) — Universidade Tecnológica Federal do Paraná, 2010.

AL-ALAWI, R.; STONHAM, T. A training strategy and functionality analysis of multi-layer boolean neural networks. *Brrunel University Preprint*, 1989.

ALEKSANDER, I. Self-adaptive universal logic circuits. *IEE Electronic Letters*, v. 2, p. 321, 1966.

ALEKSANDER, I. Emergent intelligent properties of progressively structured pattern recognition nets. *Pattern Recognition Lett.*, v. 1, p. 375 – 384, 1983.

ALEKSANDER, I. The logic of connectionist systems. *Neural Computers*, Springer, Berlin, v. 41, p. 189 – 197, 1989.

ALEKSANDER, I.; ALBROW, R. C. Microcircuit learning nets: Some test with hand-written numerals. *Electronic Lett.*, v. 4, p. 408, 1968.

ALEKSANDER, I.; MORTON, H. *An Introduction to Neural Computing*. London: Chapman and Hall, 1989.

ALEKSANDER, I.; THOMAS, W. V.; BOWDEN, P. A. Wisard a radical step forward in image recognition. *Emerald Insight*, 1984.

ARAúJO, L. A. de. *RWiSARD: Um modelo de rede neural sem peso para reconhecimento e classificação de imagens em escala de cinza*. Dissertação (Mestrado) — Universidade Federal do Rio de Janeiro, COPPE/UFRJ, 2011.

AUSTIN, J. A review of ram based neural networks. In: *Microelectronics for Neural Networks and Fuzzy Systems*. Turin, Italy: IEEE, 1994. ISBN 0-8186-6710-9.

BANDEIRA, L. C. *NC-WiSARD: Uma interpretação sem pesos do modelo neural*. Dissertação (Mestrado) — Universidade Federal do Rio de Janeiro, 2010.

BARRETO, J. M. *Introdução às Redes Neurais*. http://www.inf.ufsc.br/ j.barreto/tutoriais/Survey.pdf, 2002.

BLEDSOE, W.; BROWING, I. Pattern recognition and reading by machine. In: *Proc Eastern Joint Computer Conf. Boston*. Boston, Massachusetts: ACM New York, USA, 1959. p. 225 – 232.

BRAGA, A. de P.; CARVALHO, A. P. de Leon F. de; LUDERMIR, T. B. *Redes Neurais Artificias: Teoria e Aplicações*. Travessa do Ouvidor, 11, Rio de Janeiro: LTC editora, 2000.

CARVALHO, A. de; FAIRHURST, M. C.; BISSET, D. L. An integrated boolean neural network for pattern classification. *Pattern Recognition Letters*, n. 15, p. 807 – 813, 1994.

FORECHI, A. et al. Fat-fast vg-ram wnn: A high performance approach. *Neurocomputing*, v. 183 (2016), p. 56 – 69, 2016.

FUKUSHIMA, K. Neocognitron for handwritten digit recognition. *Neuro-computing*, v. 51, p. 161 – 180, 2003.

GARCIA, L. A. C. *Métodos de otimização global para escolha do padrão de conectividade de redes neurais sem peso*. Dissertação (Mestrado) — Universidade Federal de Pernambuco, 2003.

GREGORIO, M. D.; GIORDANO, M. A wisard-based approach to cdnet. *BRICS Congress on Computational Intelligence and 11th Brazilian Congress on Computational Intelligence*, 2013.

GREGORIO, M. D.; GIORDANO, M. Change detection with weightless neural networks. *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2014.

HAYKIN, S. *Redes Neurais Princípios e Práticas*. Av. Angélica, 1091 Higienópolis - São Paulo SP: Bookman, 2001.

HOWELLS, G.; FAIRHURST, M. C.; BISSET, D. Bcn: A novel network architecture for ram-based neurons. *Pattern Recognition Letters*, v. 16, p. 297 – 303, 1995.

HULL, J. J. A database for handwritten text recognition research. *IEEE transactions on pattern analysis and machine inteligence*, v. 16, n. 5, 1994.

JúNIOR, L. J. L. et al. Image-based global localization using vg-ram weightless neural networks. *International Joint Conference on Neural Networks (IJCNN)*, 2014. Manuscript received on March 1st, 2013.

JORGENSEN, T. M.; CHRISTENSEN, S. S.; ANDERSEN, A. W. Detecting danger labels with ram-based neural networks. *Pattern Recognition Letters*, v. 17, p. 399 – 412, 1996.

KEYS, R. G. Cubic convolution interpolation for digital image processing. *IEEE Transactions on Acoustics, Speech, Signal Processing*, v. 29, p. 1153 – 1160, 1981.

LUDERMIR, T. B. Computability and learnability of weaghtless neural networks. *Periódico Desconhecido*, 1994.

LUDERMIR, T. B.; OLIVEIRA, W. R. de. Weightless neural models. *Computer Standards and Interfaces*, v. 16, p. 253 – 263, 1994.

MOREIRA, R. S. et al. Tracking targes in sea surface with the weightless neural netwrok. *2013 BRICS Congress on Computational Intelligence and 11th Brazilian Congress on Computational Intelligence*, 2013.

MYERS, C. E. Learning algorithms for probabilistic neural nets. *Proc. First INNS Annual Meeting*, 1988.

MYERS, C. E. Output functions for probabilistic logic nodes. *Artificial Neural Networks*, IET, London, UK, 1989.

NEDJAH, N. et al. A massively parallel pipelined reconfigurable design for m-pln based neural networks for efficient image classification. *Neurocomputing*, v. 183, p. 39 – 55, 2016.

PEARSON, K. Notes on regression and inheritance in the case of two parents. In: *Proceedings of the Royal Society of London*. London: Royal Society of London, 1895. v. 58, p. 240 – 242.

PENNY, W. D.; STONHAM, T. J. Learning algorithms for logical neural networks. In: *Systems Engineering*. Pittsburgh, PA, USA: IEEE, 1990. ISBN 0-7803-0173-0.

RAMANAN, S. et al. pram nets for detection of small targets in sequences of infra-red images. *Neural Networks*, Elsevier Science Ltd, v. 8, n. 7, p. 1227 – 1237, 1995.

ROSENBLATT, F. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Washington: Spartan Book, 1962.

SILVA, I. N.; SPATI, D. H.; FLAUZINO, R. A. *Redes Neurais Artificiais para engenharia e ciências aplicadas*. São Paulo, SP: Artliber, 2010.

SOARES, C. M.; SILVA, C. L. F.; GREGORIO, M. Uma implementação em software do classificador wisard. In: *Anais do V Simpósio Brasileiro de Redes Neurais*. Belo Horizonte: Sociedade Brasileira de Computação, 1998. p. 225 – 229.

STONHAM, T. Automatic classification of mass spectra. *Patttern Rcognition*, v. 7, p. 235, 1975.

STONHAM, T. J. Practical face recognition and verification with wisard. In: _____. *Aspects of Face Processing*. Dordrecht: Springer Netherlands, 1986. v. 28, p. 426 – 441. ISBN 978-94-009-4420-6.

STONHAM, T. J.; FAIRHUST, M. C. A classification systems for alpha-numeric characteres based on learning network techniques. *Digital Processes*, v. 2, p. 321, 1976.

WICKERT, I.; FRANÇA, F. M. G. Autowisard: Unsupervised modes for the wisard. *Lecture Noted in Computer Science*, Springer, Berlin, Heidelberg, v. 2084, p. 435 – 441, 2001.

WILAMOWSKI, B. M.; HUNTER, D.; MALINOWSKI, A. Solving parity-n problems with feedforward neural networks. In: *Neural Networks*. Portland, OR, USA: IEEE, 2003. ISBN 0-7803-7898-9/03. ISSN 1098-7576.

# APÊNDICE A – Resultados de Simulação

Este apêndice apresenta os valores das simulações apresentadas no Capítulo 5 com relação às métricas avaliadas, a saber, acurácia, épocas de treinamento e tempo de processamento médio.

## A,1 Resultados da classificação de bit de paridade

Abaixo são mostrados os resultados obtidos nas simulações para o problema de paridade, conforme discutido no Seção 5.1.

### A,1.1 Resultados da variação da tupla e números de estágios para paridade

Tabela 15: Resultados da Arq1 para paridade

| | **Acurácia (%)** | | **Épocas** | | **Tempo (s)** | |
|---|---|---|---|---|---|---|
| | **Média** | **Desvio padrão** | **Média** | **Desvio padrão** | **Média** | **Desvio Padrão** |
| **G1** | 48,8 | 19,96 | 55,22 | 46,12 | 85,49 | 71,41 |
| **G2** | 92,50 | 6,48 | 375,18 | 362,39 | 1182,43 | 1142 |
| **G3** | 97,87 | 7,32 | 156,46 | 137,68 | 854,79 | 752 |

Tabela 16: Resultados da Arq2 para paridade

| | **Acurácia (%)** | | **Épocas** | | **Tempo (s)** | |
|---|---|---|---|---|---|---|
| | **Média** | **Desvio padrão** | **Média** | **Desvio padrão** | **Média** | **Desvio Padrão** |
| **G1** | 44 | 18,0702 | 92,5800 | 90,8749 | 113,4407 | 111,3514 |
| **G2** | 84,3 | 19,033 | 100,0800 | 98,5523 | 236,1054 | 233 |
| **G3** | 96,3125 | 4,7953 | 24,2200 | 22,1445 | 94,8466 | 87 |

Tabela 17: Resultados da Arq3 para paridade

| | **Acurácia (%)** | | **Épocas** | | **Tempo (s)** | |
|---|---|---|---|---|---|---|
| | **Média** | **Desvio padrão** | **Média** | **Desvio padrão** | **Média** | **Desvio Padrão** |
| **G1** | 60,6 | 13,4635 | 14,8400 | 7,2036 | 27,8839 | 13,5353 |
| **G2** | 87,8 | 6,0373 | 704,2400 | 623,2093 | 6617,8043 | 5856 |
| **G3** | 98,12 | 1,4934 | 57,2200 | 42,0594 | 1178,7890 | 866 |

Tabela 18: Resultados da Arq4 para paridade

| | **Acurácia (%)** | | **Épocas** | | **Tempo (s)** | |
|---|---|---|---|---|---|---|
| | **Média** | **Desvio padrão** | **Média** | **Desvio padrão** | **Média** | **Desvio Padrão** |
| **G1** | 59,8 | 8,204 | 14,0200 | 5,3126 | 17,4071 | 6,5961 |
| **G2** | 86,16 | 5,7263 | 233,6200 | 170,6677 | 1552,3662 | 1134 |
| **G3** | 98,46 | 2,4678 | 17,6200 | 11,5191 | 208,5596 | 136 |

Tabela 19: Resultados da Arq5 para paridade

| | **Acurácia (%)** | | **Épocas** | | **Tempo (s)** | |
|---|---|---|---|---|---|---|
| | **Média** | **Desvio padrão** | **Média** | **Desvio padrão** | **Média** | **Desvio Padrão** |
| **G1** | 59,6 | 8,797 | 16,7400 | 11,2137 | 49,4328 | 33,1138 |
| **G2** | 99,84 | 0,6809 | 611,5000 | 582,6991 | 8458,8374 | 8060 |
| **G3** | 98,9 | 7,7782 | 76,2200 | 58,1249 | 2095,2980 | 1598 |

Tabela 20: Resultados da Arq6 para paridade

| | **Acurácia (%)** | | **Épocas** | | **Tempo (s)** | |
|---|---|---|---|---|---|---|
| | **Média** | **Desvio padrão** | **Média** | **Desvio padrão** | **Média** | **Desvio Padrão** |
| **G1** | 42,4 | 13,9328 | 34,7000 | 26,9658 | 53,9609 | 41,9337 |
| **G2** | 94,5 | 2,6515 | 1382,6000 | 1137,2 | 20669,5104 | 7001 |
| **G3** | 99,97 | 0,1144 | 34,7600 | 21,3881 | 4222,7822 | 2598 |

Tabela 21: Resultados da Arq7 para paridade

| | **Acurácia (%)** | | **Épocas** | | **Tempo (s)** | |
|---|---|---|---|---|---|---|
| | **Média** | **Desvio padrão** | **Média** | **Desvio padrão** | **Média** | **Desvio Padrão** |
| **G1** | 52,8 | 16,418 | 56,8600 | 53,0256 | 65,7363 | 61,3033 |
| **G2** | 83,28 | 5,544 | 180,3000 | 161,4543 | 2051,2053 | 1837 |
| **G3** | 99,976 | 0,0870 | 6,9600 | 3,2761 | 375,9357 | 177 |

Tabela 22: Resultados da Arq8 para paridade

| | **Acurácia (%)** | | **Épocas** | | **Tempo (s)** | |
|---|---|---|---|---|---|---|
| | **Média** | **Desvio padrão** | **Média** | **Desvio padrão** | **Média** | **Desvio Padrão** |
| **G1** | 47,4 | 14,82 | 15,8400 | 5,3236 | 40,1794 | 13,5037 |
| **G2** | 79,5 | 4,9497 | 688,84 | 600 | 1000 | 871 |
| **G3** | 99,982 | 0,0596 | 55,4000 | 29,1078 | 15938,2509 | 8374 |

Tabela 23: Resultados da Arq9 para paridade

| | **Acurácia (%)** | | **Épocas** | | **Tempo (s)** | |
|---|---|---|---|---|---|---|
| | **Média** | **Desvio padrão** | **Média** | **Desvio padrão** | **Média** | **Desvio Padrão** |
| **G1** | 43,6 | 15,0861 | 12,7600 | 3,0072 | 15,3135 | 3,6090 |
| **G2** | 55,76 | 6,1663 | 2533,8000 | 2216,3 | 23062,2579 | 20172 |
| **G3** | 99,87 | 0,1865 | 7,7200 | 3,4761 | 513,3511 | 231 |

Tabela 24: Resultados da Arq10 para paridade

| | **Acurácia (%)** | | **Épocas** | | **Tempo (s)** | |
|---|---|---|---|---|---|---|
| | **Média** | **Desvio padrão** | **Média** | **Desvio padrão** | **Média** | **Desvio Padrão** |
| **G1** | 47,04 | 5,7355 | 18,5200 | 8,3842 | 33,5381 | 15,1831 |
| **G2** | 99,72 | 0,7296 | 688,84 | 612,1845 | 17709,5920 | 15739 |
| **G3** | 98,96 | 7,3539 | 71,1200 | 58,9630 | 20482,4323 | 6981 |

## A,1.2 Resultados da variação da resolução da memória para paridade

Tabela 25: Resultados da Arq1 para paridade com 5% ruído

| | **Acurácia** | | **Épocas** | | **Tempo** | |
|---|---|---|---|---|---|---|
| **Resolução** | **Média** | **Desvio padrão** | **Média** | **Desvio padrão** | **Média** | **Desvio Padrão** |
| $\omega = \mathbf{5}$ | 79,6875 | 8,983 | 28,8500 | 12,7415 | 2,0942 | 0,9194 |
| $\omega = \mathbf{7}$ | 83,5938 | 8,4752 | 34,5000 | 22,8646 | 2,5150 | 1,7376 |
| $\omega = \mathbf{11}$ | 86,875 | 5,9638 | 51,4500 | 46,8548 | 3,8903 | 3,3394 |
| $\omega = \mathbf{21}$ | 87,0313 | 11,6166 | 265,3500 | 236,5297 | 21,7942 | 19,3525 |

Tabela 26: Resultados da Arq2 para paridade com 5% ruído

| | **Acurácia** | | **Épocas** | | **Tempo** | |
|---|---|---|---|---|---|---|
| **Resolução** | **Média** | **Desvio padrão** | **Média** | **Desvio padrão** | **Média** | **Desvio Padrão** |
| $\omega = \mathbf{5}$ | 88,5938 | 7,6058 | 18 | 8,2590 | 0,9615 | 0,4468 |
| $\omega = \mathbf{7}$ | 88,2813 | 6,1568 | 23,6000 | 7,7690 | 1,2698 | 0,4402 |
| $\omega = \mathbf{11}$ | 92,0313 | 5,0159 | 46,9500 | 40,5079 | 2,6985 | 2,2894 |
| $\omega = \mathbf{21}$ | 88,75 | 7,1979 | 96,4000 | 94,2343 | 5,6316 | 5,2212 |

Tabela 27: Resultados da Arq3 para paridade com 5% ruído

| | **Acurácia** | | **Épocas** | | **Tempo** | |
|---|---|---|---|---|---|---|
| **Resolução** | **Média** | **Desvio padrão** | **Média** | **Desvio padrão** | **Média** | **Desvio Padrão** |
| $\omega = \mathbf{5}$ | 91,6 | 11,6547 | 70,0500 | 49,7990 | 22,0576 | 16,0245 |
| $\omega = \mathbf{7}$ | 94,65 | 6,0199 | 34,7000 | 18,8012 | 11,0269 | 6,4035 |
| $\omega = \mathbf{11}$ | 95,6 | 4,8276 | 90,2000 | 60,6731 | 23,1649 | 17,7883 |
| $\omega = \mathbf{21}$ | 93 | 15,8513 | 535,7000 | 327,0617 | 169,7558 | 103,6651 |

Tabela 28: Resultados da Arq4 para paridade com 5% ruído

| | **Acurácia** | | **Épocas** | | **Tempo** | |
|---|---|---|---|---|---|---|
| **Resolução** | **Média** | **Desvio padrão** | **Média** | **Desvio padrão** | **Média** | **Desvio Padrão** |
| $\omega = \mathbf{5}$ | 97,75 | 2,4034 | 7,2500 | 2,8631 | 1,6712 | 0,8947 |
| $\omega = \mathbf{7}$ | 97,3 | 2,3642 | 12 | 6,8518 | 2,0359 | 1,2629 |
| $\omega = \mathbf{11}$ | 96,25 | 2,7314 | 25,5500 | 24,4508 | 3,4915 | 3,4646 |
| $\omega = \mathbf{21}$ | 96,75 | 2,8074 | 65,2000 | 63,1499 | 13,9588 | 13,1759 |

Tabela 29: Resultados da Arq5 para paridade com 5% ruído

| | **Acurácia** | | **Épocas** | | **Tempo** | |
|---|---|---|---|---|---|---|
| **Resolução** | **Média** | **Desvio padrão** | **Média** | **Desvio padrão** | **Média** | **Desvio Padrão** |
| $\omega = \mathbf{5}$ | 79,5 | 17,8134 | 291,0500 | 264,0643 | 119,9430 | 110,9460 |
| $\omega = \mathbf{7}$ | 96,25 | 5,7754 | 81,6500 | 72,2287 | 35,8762 | 31,2384 |
| $\omega = \mathbf{11}$ | 95,6 | 6,5486 | 124,8500 | 84,1861 | 47,3457 | 31,1078 |
| $\omega = \mathbf{21}$ | 83,45 | 24,0492 | 618,2500 | 373,8950 | 292,2025 | 184,7673 |

Tabela 30: Resultados da Arq1 para paridade com 10% ruído

| | **Acurácia** | | **Épocas** | | **Tempo** | |
|---|---|---|---|---|---|---|
| **Resolução** | **Média** | **Desvio padrão** | **Média** | **Desvio padrão** | **Média** | **Desvio Padrão** |
| $\omega = \mathbf{5}$ | 74,0625 | 11,4036 | 425,3 | 409,7247 | 30,86 | 26,12 |
| $\omega = \mathbf{7}$ | 81,0625 | 8,8255 | 384,78 | 327,79 | 29,6 | 23,64 |
| $\omega = \mathbf{11}$ | 87,4375 | 6,576 | 470,42 | 400,2861 | 38,057 | 32,93 |
| $\omega = \mathbf{21}$ | 92,0625 | 5,2523 | 593,1 | 473,1476 | 51,22 | 39,54 |

Tabela 31: Resultados da Arq2 para paridade com 10% ruído

| | Acurácia | | Épocas | | Tempo | |
|---|---|---|---|---|---|---|
| **Resolução** | **Média** | **Desvio padrão** | **Média** | **Desvio padrão** | **Média** | **Desvio Padrão** |
| $\omega = \mathbf{5}$ | 80,3125 | 9,2196 | 289,48 | 286,5641 | 16,509 | 13,75 |
| $\omega = \mathbf{7}$ | 87,0625 | 6,4697 | 174,26 | 163,0325 | 9,6 | 7,1 |
| $\omega = \mathbf{11}$ | 91,5625 | 7,3695 | 184,64 | 162,7448 | 10,203 | 8,02 |
| $\omega = \mathbf{21}$ | 92,75 | 5,261 | 238,4 | 203,6133 | 13,88 | 9,45 |

Tabela 32: Resultados da Arq3 para paridade com 10% ruído

| | Acurácia | | Épocas | | Tempo | |
|---|---|---|---|---|---|---|
| **Resolução** | **Média** | **Desvio padrão** | **Média** | **Desvio padrão** | **Média** | **Desvio Padrão** |
| $\omega = \mathbf{5}$ | 52,62 | 11,0378 | 1000 | 0 | 377,0290 | 2,58 |
| $\omega = \mathbf{7}$ | 81 | 19,1833 | 1000 | 0 | 345,0605 | 3,67 |
| $\omega = \mathbf{11}$ | 75,9 | 23,5817 | 1000 | 0 | 255,87 | 8,41 |
| $\omega = \mathbf{21}$ | 53,85 | 18,8213 | 1000 | 0 | 291,68 | 15,9323 |

Tabela 33: Resultados da Arq4 para paridade com 10% ruído

| | Acurácia | | Épocas | | Tempo | |
|---|---|---|---|---|---|---|
| **Resolução** | **Média** | **Desvio padrão** | **Média** | **Desvio padrão** | **Média** | **Desvio Padrão** |
| $\omega = \mathbf{5}$ | 88,65 | 4,6597 | 1000 | 0 | 163,5091 | 6,25 |
| $\omega = \mathbf{7}$ | 92,1 | 3,4012 | 1000 | 0 | 163,1281 | 7,02 |
| $\omega = \mathbf{11}$ | 87,05 | 16,2529 | 1000 | 0 | 165,6737 | 3,0478 |
| $\omega = \mathbf{21}$ | 66,15 | 24,171 | 1000 | 0 | 162,7404 | 8,2689 |

Tabela 34: Resultados da Arq5 para paridade com 10% ruído

| | Acurácia | | Épocas | | Tempo | |
|---|---|---|---|---|---|---|
| **Resolução** | **Média** | **Desvio padrão** | **Média** | **Desvio padrão** | **Média** | **Desvio Padrão** |
| $\omega = \mathbf{5}$ | 48,5 | 5,6242 | 1000 | 0 | 413,2645 | 5,72 |
| $\omega = \mathbf{7}$ | 58,5 | 16,4908 | 1000 | 0 | 571,6657 | 3,66 |
| $\omega = \mathbf{11}$ | 67,65 | 24,8496 | 1000 | 0 | 406,5217 | 6,6311 |
| $\omega = \mathbf{21}$ | 48,55 | 11,274 | 1000 | 0 | 464,0678 | 4,1483 |

Tabela 35: Resultados da Arq1 para paridade com 15% ruído

| | Acurácia | | Épocas | | Tempo | |
|---|---|---|---|---|---|---|
| **Resolução** | **Média** | **Desvio padrão** | **Média** | **Desvio padrão** | **Média** | **Desvio Padrão** |
| $\omega = \mathbf{5}$ | 60,1563 | 12,2035 | 1000 | 0 | 69,5194 | 0,4451 |
| $\omega = \mathbf{7}$ | 70,7813 | 12,9949 | 969,6500 | 140,2015 | 69,5928 | 11,4483 |
| $\omega = \mathbf{11}$ | 61,25 | 18,9518 | 1000 | 0 | 70,8684 | 1,0296 |
| $\omega = \mathbf{21}$ | 49,6875 | 15,4117 | 1000 | 0 | 89,9788 | 3,6300 |

Tabela 36: Resultados da Arq2 para paridade com 15% ruído

| | **Acurácia** | | **Épocas** | | **Tempo** | |
|---|---|---|---|---|---|---|
| **Resolução** | **Média** | **Desvio padrão** | **Média** | **Desvio padrão** | **Média** | **Desvio Padrão** |
| $\omega = \mathbf{5}$ | 50,1563 | 15,1227 | 1000 | 0 | 55,0589 | 5,4023 |
| $\omega = \mathbf{7}$ | 64,2188 | 19,8803 | 1000 | 0 | 42,4063 | 0,8814 |
| $\omega = \mathbf{11}$ | 62,5 | 21,219 | 1000 | 0 | 58,3585 | 2,0919 |
| $\omega = \mathbf{21}$ | 55 | 20,2094 | 1000 | 0 | 64,3777 | 4,8214 |

Tabela 37: Resultados da Arq3 para paridade com 15% ruído

| | **Acurácia** | | **Épocas** | | **Tempo** | |
|---|---|---|---|---|---|---|
| **Resolução** | **Média** | **Desvio padrão** | **Média** | **Desvio padrão** | **Média** | **Desvio Padrão** |
| $\omega = \mathbf{5}$ | 48,7 | 4,8785 | 1000 | 0 | 279,4220 | 4,4806 |
| $\omega = \mathbf{7}$ | 63,3 | 16,9243 | 1000 | 0 | 237,0061 | 18,8055 |
| $\omega = \mathbf{11}$ | 66,4 | 20,5257 | 1000 | 0 | 295,5661 | 23,7642 |
| $\omega = \mathbf{21}$ | 55,5 | 17,8901 | 1000 | 0 | 307,2587 | 10,6632 |

Tabela 38: Resultados da Arq4 para paridade com 15% ruído

| | **Acurácia** | | **Épocas** | | **Tempo** | |
|---|---|---|---|---|---|---|
| **Resolução** | **Média** | **Desvio padrão** | **Média** | **Desvio padrão** | **Média** | **Desvio Padrão** |
| $\omega = \mathbf{5}$ | 72,75 | 7,3332 | 1000 | 0 | 160,7020 | 7,1994 |
| $\omega = \mathbf{7}$ | 75,15 | 2,9429 | 1000 | 0 | 145,0594 | 22,7570 |
| $\omega = \mathbf{11}$ | 77,6 | 3,9789 | 1000 | 0 | 194,1654 | 5,5233 |
| $\omega = \mathbf{21}$ | 68,3 | 13,606 | 1000 | 0 | 177,2944 | 2,1200 |

Tabela 39: Resultados da Arq5 para paridade com 15% ruído

| | **Acurácia** | | **Épocas** | | **Tempo** | |
|---|---|---|---|---|---|---|
| **Resolução** | **Média** | **Desvio padrão** | **Média** | **Desvio padrão** | **Média** | **Desvio Padrão** |
| $\omega = \mathbf{5}$ | 49,6 | 5,5098 | 1000 | 0 | 579,6872 | 526,6509 |
| $\omega = \mathbf{7}$ | 47,85 | 3,6602 | 1000 | 0 | 443,2004 | 19,9731 |
| $\omega = \mathbf{11}$ | 72,15 | 18,8101 | 1000 | 0 | 430,4312 | 29,7525 |
| $\omega = \mathbf{21}$ | 47,1 | 5,8395 | 1000 | 0 | 464,0678 | 409,1483 |

# A,2 Resultados da identificação de algarismos

Abaixo são mostrados os resultados obtidos nas simulações para o problema de reconhecimento de algarismos manuscritos, conforme discutido no Seção 5.2.2.

Tabela 40: Resultados obtidos com a rede MPLN para 2 classes de algarismos

| | | **Acurácia GB** | | **Acurácia GM** | | **Acurácia GR** | |
|---|---|---|---|---|---|---|---|
| **Arq.** | **Amostras** | **Média** | **Desvio Padrão** | **Média** | **Desvio Padrão** | **Média** | **Desvio Padrão** |
| **Arq2** | **10** | 51,9 | 3,1768 | 50,7600 | 3,1916 | 53,5800 | 3,7367 |
| | **50** | 78,22 | 8,7931 | 84,4000 | 7,0682 | 77,4000 | 4,3846 |
| | **100** | 94,88 | 2,4631 | 89,6200 | 4,8693 | 78,32 | 4,4878 |
| **Arq3** | **10** | 49,78 | 2,6131 | 50,3400 | 1,7912 | 52,7000 | 2,8803 |
| | **50** | 68,08 | 7,3258 | 68,5400 | 7,4675 | 65,7400 | 5,7136 |
| | **100** | 84,66 | 4,4704 | 75,7600 | 6,6011 | 72,9600 | 6,4173 |

Tabela 41: Resultados obtidos com a rede MPLN para 4 classes de algarismos

| | | **Acurácia GB** | | **Acurácia GM** | | **Acurácia GR** | |
|---|---|---|---|---|---|---|---|
| **Arq.** | **Amostras** | **Média** | **Desvio Padrão** | **Média** | **Desvio Padrão** | **Média** | **Desvio Padrão** |
| **Arq2** | **10** | 24,66 | 6,1764 | 24,4000 | 3,7261 | 23,7200 | 6,2206 |
| | **50** | 25,04 | 6,7005 | 23,0400 | 7,0652 | 24,3000 | 5,3321 |
| | **100** | 30,0323 | 10,6285 | 24,5500 | 9,3103 | 23,15 | 4,8153 |
| **Arq3** | **10** | 25,64 | 6,2914 | 26,1600 | 3,9865 | 24,7000 | 4,8917 |
| | **50** | 25,0952 | 4,9690 | 26,3500 | 4,6823 | 23,7727 | 6,9757 |
| | **100** | 28,8 | 6,8718 | 25,6800 | 6,4402 | 23,6765 | 6,0439 |

Tabela 42: Resultados obtidos com a rede WiSARD sem pré-processamento das imagens

| | | **Acurácia GB** | | **Acurácia GM** | | **Acurácia GR** | |
|---|---|---|---|---|---|---|---|
| **Arq.** | **Amostras** | **Média** | **Desvio Padrão** | **Média** | **Desvio Padrão** | **Média** | **Desvio Padrão** |
| **Arq1** | **10** | 92,51 | 1.7592 | 76,41 | 2.7004 | 55,76 | 4.7451 |
| | **50** | 93,75 | 3.7623 | 74,56 | 2.8704 | 52,43 | 4.3395 |
| | **100** | 88,36 | 3.7683 | 67,72 | 3.7989 | 31,91 | 4.2212 |
| **Arq4** | **10** | 95,38 | 1.1517 | 83,16 | 1.6182 | 68,75 | 2.5976 |
| | **50** | 99,29 | 0.5501 | 90,96 | 2.2542 | 62,72 | 3.1267 |
| | **100** | 98,42 | 0.8522 | 82,58 | 2.0995 | 51,23 | 2.5603 |
| **Arq5** | **10** | 97,43 | 1.2312 | 87,76 | 1.8202 | 55,42 | 2.8373 |
| | **50** | 99,21 | 0.8885 | 90,46 | 1.2085 | 67,13 | 2.7511 |
| | **100** | 98,91 | 0.8256 | 88,78 | 1.7554 | 64,36 | 3.4470 |
| **Arq10** | **10** | 100 | 0 | 72,87 | 3.0136 | 26,34 | 3.0655 |
| | **50** | 100 | 0 | 86,69 | 2.4623 | 32,41 | 3.0689 |
| | **100** | 100 | 0 | 87,62 | 1.9974 | 29,75 | 2.8819 |

Tabela 43: Resultados obtidos com a rede WiSARD com pré-processamento das imagens

| | | **Acurácia GB** | | **Acurácia GM** | | **Acurácia GR** | |
|---|---|---|---|---|---|---|---|
| **Arq.** | **Amostras** | **Média** | **Desvio Padrão** | **Média** | **Desvio Padrão** | **Média** | **Desvio Padrão** |
| **Arq1** | **10** | 100 | 0 | 79,33 | 2.9105 | 59,08 | 2.7891 |
| | **50** | 100 | 0 | 77,67 | 3.5433 | 46,28 | 9.0053 |
| | **100** | 100 | 0 | 72 | 3.6832 | 33 | 3.2053 |
| **Arq4** | **10** | 95,82 | 1.9494 | 84,11 | 2.1343 | 55,37 | 2.3709 |
| | **50** | 90,93 | 1.8638 | 90,06 | 1.4464 | 59,07 | 3.2521 |
| | **100** | 91 | 2.08 | 88 | 1.5424 | 48 | 2.49 |
| **Arq5** | **10** | 98,41 | 1.2312 | 86,28 | 1.8202 | 54,46 | 2.8373 |
| | **50** | 92,73 | 1.7290 | 87,55 | 1.3139 | 62,74 | 2.1740 |
| | **100** | 95 | 1.7502 | 92 | 1.1743 | 67 | 2.0641 |
| **Arq10** | **10** | 94,74 | 0.8826 | 80,04 | 1.7130 | 61,95 | 2.4121 |
| | **50** | 94 | 0.8751 | 83 | 0.8507 | 61 | 1.8771 |
| | **100** | 94 | 0.8127 | 91 | 0.6708 | 72 | 2.0995 |

Tabela 44: Resultados obtidos com a rede Mod-MPLN com pré-processamento das imagens

| | | **Acurácia GB** | | **Acurácia GM** | | **Acurácia GR** | |
|---|---|---|---|---|---|---|---|
| **Arq.** | **Amostras** | **Média** | **Desvio Padrão** | **Média** | **Desvio Padrão** | **Média** | **Desvio Padrão** |
| **Arq1** | **10** | 88.4 | 8.7585 | 69.9000 | 7.4005 | 52.6000 | 7.0111 |
| | **50** | 95.70 | 3.8020 | 79.60 | 4.4771 | 54.9000 | 5.1521 |
| | **100** | 95.5000 | 2.4152 | 78.4000 | 3.4059 | 49 | 6.2539 |
| **Arq4** | **10** | 98.2 | 4.8192 | 76.6 | 7.9821 | 35.8 | 13.1071 |
| | **50** | 87.54 | 1.6686 | 84.98 | 2.2812 | 52.72 | 2.5075 |
| | **100** | 90.48 | 1.6443 | 89.52 | 2.2812 | 53.1 | 2.5075 |
| **Arq5** | **10** | 98.6 | 3.5051 | 79.4 | 2.399 | 33.4 | 10.224 |
| | **50** | 73.8 | 4.0908 | 77.26 | 2.9679 | 50.94 | 2.9583 |
| | **100** | 71.8 | 7.8921 | 76.12 | 5.34 | 47.96 | 4.4674 |
| **Arq10** | **10** | 93.4 | 5.1942 | 66.4 | 6.6271 | 56.8 | 3.7167 |
| | **50** | 94 | 3.0551 | 85.8000 | 4.4422 | 55.52 | 3.5297 |
| | **100** | 81.02 | 2.4619 | 76.14 | 3.828 | 56.98 | 3.1525 |

## A,2.1 Resultados da variação da tupla e números de estágios para algarismos

Tabela 45: Resultados da Arq1 para o problema de reconhecimento de algarismos

| | | Acurácia | | Épocas | | Tempo | |
|---|---|---|---|---|---|---|---|
| **Grupo** | **Amostras** | **Média** | **Desvio padrão** | **Média** | **Desvio padrão** | **Média** | **Desvio Padrão** |
| **GB** | **10** | 90,3800 | 3,9892 | 50,2000 | 1,8516 | 81,62 | 3,0105 |
| | **50** | 94,9400 | 2,0644 | 48,3800 | 1,0079 | 202,66 | 4,2220 |
| | **100** | 95,16 | 1,7066 | 48,1200 | 1,2229 | 351,73 | 8,9387 |
| **GM** | **10** | 71,4800 | 5,3155 | 50,3200 | 1,9737 | 81,73 | 3,2057 |
| | **50** | 80,6400 | 3,0221 | 48,0600 | 1,1678 | 254,15 | 6,1755 |
| | **100** | 77,4 | 3,6197 | 48,1000 | 1,2976 | 333,53 | 8,9977 |
| **GR** | **10** | 49,4400 | 5,4592 | 50,0200 | 1,6474 | 97,08 | 3,1973 |
| | **50** | 60,5000 | 4,5051 | 48,1400 | 1,2779 | 202,39 | 5,3725 |
| | **100** | 59,72 | 3,6086 | 48,1000 | 1,1824 | 486,48 | 11,9587 |

Tabela 46: Resultados da Arq2 para o problema de reconhecimento de algarismos

| | | Acurácia | | Épocas | | Tempo | |
|---|---|---|---|---|---|---|---|
| **Grupo** | **Amostras** | **Média** | **Desvio padrão** | **Média** | **Desvio padrão** | **Média** | **Desvio Padrão** |
| **GB** | **10** | 91,6400 | 2,4054 | 80,8400 | 3,9194 | 37,88 | 1,8366 |
| | **50** | 95,9200 | 1,4961 | 57,6200 | 3,1551 | 88,77 | 4,8608 |
| | **100** | 95,66 | 1,6612 | 55,2000 | 3,2576 | 155,81 | 9,1950 |
| **GM** | **10** | 74,7600 | 3,317 | 81,6200 | 5,3866 | 40,36 | 2,6636 |
| | **50** | 83,1200 | 3,1793 | 58 | 3,1429 | 81,28 | 4,4044 |
| | **100** | 85,08 | 2,5939 | 54,2000 | 2,0996 | 211,42 | 8,1900 |
| **GR** | **10** | 53,1600 | 4,3301 | 82,2600 | 5,6957 | 38,65 | 2,6761 |
| | **50** | 63,8400 | 4,7223 | 56,8400 | 2,5663 | 77,50 | 3,4991 |
| | **100** | 65,62 | 4,6065 | 54,8200 | 3,4386 | 306,69 | 19,2372 |

Tabela 47: Resultados da Arq3 para o problema de reconhecimento de algarismos

| | | Acurácia | | Épocas | | Tempo | |
|---|---|---|---|---|---|---|---|
| **Grupo** | **Amostras** | **Média** | **Desvio padrão** | **Média** | **Desvio padrão** | **Média** | **Desvio Padrão** |
| **GB** | **10** | 91,8600 | 2,5952 | 87,6200 | 5,8306 | 62,20 | 4,1390 |
| | **50** | 96,7000 | 1,4463 | 58,3800 | 2,5546 | 192,31 | 8,4151 |
| | **100** | 97,84 | 1,2014 | 52,4000 | 1,8070 | 318,65 | 10,9886 |
| **GM** | **10** | 74,2800 | 2,7333 | 89,0800 | 5,3789 | 70,36 | 4,2485 |
| | **50** | 84,3200 | 2,5269 | 58,4800 | 2,5970 | 184,03 | 8,1725 |
| | **100** | 89,1 | 2,1309 | 51,6200 | 1,3686 | 325,7 | 8,6353 |
| **GR** | **10** | 56,6200 | 4,0552 | 90,2600 | 5,6598 | 60,98 | 3,8238 |
| | **50** | 69,1400 | 4,3986 | 57,3200 | 2,4780 | 207,07 | 8,9518 |
| | **100** | 72,18 | 3,8845 | 51,4200 | 1,1622 | 312,45 | 7,0620 |

Tabela 48: Resultados da Arq4 para o problema de reconhecimento de algarismos

| | | Acurácia | | Épocas | | Tempo | |
|---|---|---|---|---|---|---|---|
| **Grupo** | **Amostras** | **Média** | **Desvio padrão** | **Média** | **Desvio padrão** | **Média** | **Desvio Padrão** |
| **GB** | **10** | 94,52 | 2,866 | 50,4600 | 2,2151 | 142,02 | 6,2344 |
| | **50** | 98,62 | 0,8053 | 48,1600 | 1,0947 | 409,17 | 9,3006 |
| | **100** | 98,6 | 0,8571 | 48,4200 | 1,1445 | 788,48 | 18,6372 |
| **GM** | **10** | 72,04 | 4,5399 | 50,2200 | 1,9197 | 73,44 | 2,8073 |
| | **50** | 89,5600 | 2,2054 | 48,1800 | 1,2237 | 404,93 | 10,2846 |
| | **100** | 86,2 | 2,9829 | 48,5600 | 0,8609 | 705,54 | 12,5082 |
| **GR** | **10** | 51,74 | 4,7286 | 50,2000 | 2,2223 | 72,34 | 3,2024 |
| | **50** | 62,6800 | 3,9094 | 48,3600 | 1,0645 | 404,27 | 8,8988 |
| | **100** | 65,48 | 3,8078 | 48,1400 | 1,1954 | 720,52 | 17,8918 |

Tabela 49: Resultados da Arq5 para o problema de reconhecimento de algarismos

| | | Acurácia | | Épocas | | Tempo | |
|---|---|---|---|---|---|---|---|
| **Grupo** | **Amostras** | **Média** | **Desvio padrão** | **Média** | **Desvio padrão** | **Média** | **Desvio Padrão** |
| **GB** | **10** | 93,7000 | 3,1574 | 62,9800 | 3,1265 | 85,61 | 4,2499 |
| | **50** | 97,5600 | 1,1278 | 50,8000 | 1,2122 | 231,45 | 5,5229 |
| | **100** | 97,68 | 1,0962 | 50,5400 | 1,2651 | 451,45 | 11,3005 |
| **GM** | **10** | 76,0000 | 4,513 | 62,1400 | 3,9124 | 80,04 | 5,0394 |
| | **50** | 88,8400 | 2,6981 | 50,9000 | 1,5682 | 207,1 | 6,3806 |
| | **100** | 88,58 | 2,4915 | 50,5400 | 0,9521 | 420,04 | 7,9129 |
| **GR** | **10** | 52,3400 | 5,8401 | 62,8400 | 3,9660 | 81,85 | 5,1658 |
| | **50** | 61,5600 | 4,0666 | 50,7800 | 1,7876 | 187,87 | 6,6136 |
| | **100** | 65,58 | 4,3193 | 50,2400 | 1,2382 | 398,77 | 9,8280 |

Tabela 50: Resultados da Arq6 para o problema de reconhecimento de algarismos

| | | Acurácia | | Épocas | | Tempo | |
|---|---|---|---|---|---|---|---|
| **Grupo** | **Amostras** | **Média** | **Desvio padrão** | **Média** | **Desvio padrão** | **Média** | **Desvio Padrão** |
| **GB** | **10** | 94,0200 | 2,3342 | 84,4800 | 4,1710 | 216,58 | 10,6931 |
| | **50** | 98,4800 | 0,8862 | 57,3800 | 2,4817 | 504,51 | 21,8202 |
| | **100** | 98,44 | 0,7602 | 53,7800 | 3,0592 | 851,13 | 48,4153 |
| **GM** | **10** | 79,1600 | 4,1422 | 87,0800 | 5,8688 | 173,01 | 11,6601 |
| | **50** | 89,0200 | 2,4701 | 57,0400 | 2,8209 | 673,84 | 33,3246 |
| | **100** | 91,7 | 2,0228 | 52,6400 | 1,9975 | 837,85 | 31,7934 |
| **GR** | **10** | 54,4800 | 4,2293 | 86,3000 | 4,8540 | 196,97 | 11,0787 |
| | **50** | 63,6000 | 4,1256 | 58,05 | 2,4780 | 304,7 | 16,0368 |
| | **100** | 66,22 | 4,3108 | 53,1200 | 2,3616 | 521,18 | 23,1705 |

Tabela 51: Resultados da Arq7 para o problema de reconhecimento de algarismos

| | | Acurácia | | Épocas | | Tempo | |
|---|---|---|---|---|---|---|---|
| **Grupo** | **Amostras** | **Média** | **Desvio padrão** | **Média** | **Desvio padrão** | **Média** | **Desvio Padrão** |
| **GB** | **10** | 93,3800 | 1,3231 | 415,6400 | 37,7401 | 2830,05 | 256,9685 |
| | **50** | 97,7000 | 1,3375 | 259,3000 | 4,0415 | 6810,72 | 464,1741 |
| | **100** | 97,8333 | 0,9374 | 178 | 10,32 | 11495 | 666,4517 |
| **GM** | **10** | 79,4000 | 2,2211 | 425 | 38,1430 | 2033,8256 | 182,5323 |
| | **50** | 86,4634 | 2,2372 | 283,25 | 3,1785 | 7962 | 413,7931 |
| | **100** | 85,9 | 1,792 | 210,50 | 10,2116 | 4781,9 | 231,9755 |
| **GR** | **10** | 57,2 | 2,9364 | 461,80 | 42,7442 | 2754,564 | 254,9624 |
| | **50** | 67,75 | 2,0616 | 257,22 | 2,8449 | 10288 | 511,5053 |
| | **100** | 70,9706 | 2,3026 | 227,22 | 11,95 | 15177,68 | 798,2276 |

Tabela 52: Resultados da Arq8 para o problema de reconhecimento de algarismos

| | | Acurácia | | Épocas | | Tempo | |
|---|---|---|---|---|---|---|---|
| **Grupo** | **Amostras** | **Média** | **Desvio padrão** | **Média** | **Desvio padrão** | **Média** | **Desvio Padrão** |
| **GB** | **10** | 99,6111 | 0,8711 | 49,6944 | 1,9246 | 848,1182 | 18,519 |
| | **50** | 100 | 0 | 48 | 2,0548 | 1565 | 163,4325 |
| | **100** | 100 | 0 | 44,7143 | 10,4994 | 3839,50 | 666,1096 |
| **GM** | **10** | 72,4 | 8,682 | 50,6000 | 1,6465 | 689,5818 | 44,8934 |
| | **50** | 83,3 | 3,4976 | 48,50 | 1,1785 | 1735,9 | 56,1963 |
| | **100** | 78,1 | 1,792 | 48,90 | 0,9944 | 4015,4 | 345,8552 |
| **GR** | **10** | 20,5 | 4,4159 | 47,7143 | 2,8702 | 989,9508 | 72,3764 |
| | **50** | 35,7778 | 3,0732 | 48,11 | 1,1667 | 1714,9 | 79,8709 |
| | **100** | 35,1667 | 2,3166 | 47,8333 | 1,1690 | 3762,7 | 96,6937 |

Tabela 53: Resultados da Arq9 para o problema de reconhecimento de algarismos

| | | Acurácia | | Épocas | | Tempo | |
|---|---|---|---|---|---|---|---|
| **Grupo** | **Amostras** | **Média** | **Desvio padrão** | **Média** | **Desvio padrão** | **Média** | **Desvio Padrão** |
| **GB** | **10** | 99,5 | 0,8864 | 86,8000 | 6,4429 | 331,6244 | 24,6155 |
| | **50** | 100 | 0 | 80,42 | 2,62 | 1149,8 | 144,7794 |
| | **100** | 100 | 0 | 52,8600 | 2,7331 | 1828,7 | 184,6191 |
| **GM** | **10** | 67,54 | 6,5222 | 83,7000 | 5,3880 | 922,28 | 59,3697 |
| | **50** | 87,3200 | 3,1065 | 87,3200 | 3,1065 | 925,5024 | 63,9158 |
| | **100** | 86,68 | 2,9516 | 53,2000 | 2,0898 | 1857,60 | 72,9702 |
| **GR** | **10** | 23,32 | 5,3278 | 85,4400 | 6,2177 | 956,38 | 69,5984 |
| | **50** | 28 | 4,4721 | 57,90 | 2,9981 | 1042,6 | 41,5865 |
| | **100** | 34,6 | 4,8785 | 54,6000 | 3,1305 | 1819,3 | 120,6313 |

Tabela 54: Resultados da Arq10 para o problema de reconhecimento de algarismos

| | | **Acurácia** | | **Épocas** | | **Tempo** | |
|---|---|---|---|---|---|---|---|
| **Grupo** | **Amostras** | **Média** | **Desvio padrão** | **Média** | **Desvio padrão** | **Média** | **Desvio Padrão** |
| **GB** | **10** | 98,34 | 2,0239 | 85,5000 | 5,4930 | 815,43 | 52,3878 |
| | **50** | 100 | 0 | 56,66 | 2,8449 | 2443,4 | 175,5830 |
| | **100** | 100 | 0 | 53 | 1,2382 | 4207 | 817,67 |
| **GM** | **10** | 66,18 | 5,8018 | 87,8800 | 4,8472 | 823,10 | 45,3998 |
| | **50** | 83,7600 | 3,4674 | 57 | 2,6458 | 1846,326 | 85,7019 |
| | **100** | 89,16 | 2,2889 | 52,3333 | 0,5774 | 5198,4 | 729,2706 |
| **GR** | **10** | 27 | 2,4037 | 87,0800 | 4,5168 | 806,29 | 41,8219 |
| | **50** | 26,2600 | 4,5775 | 56,8800 | 2,5123 | 2452,77 | 108,3350 |
| | **100** | 31,14 | 3,2577 | 51,9545 | 1,9042 | 4427,3 | 941,84 |

## A,2.2 Resultados da variação da resolução da memória para algarismos

Tabela 55: Resultados da Arq1 para o problema de reconhecimento de algarismos com ruído de 10%

| | | **Acurácia** | | **Épocas** | | **Tempo** | |
|---|---|---|---|---|---|---|---|
| **Grupo** | **Resolução** | **Média** | **Desvio padrão** | **Média** | **Desvio padrão** | **Média** | **Desvio Padrão** |
| **GB** | $\omega = 3$ | 38,4000 | 8,7636 | 42,8000 | 2,3875 | 318,1923 | 20,9534 |
| | $\omega = 7$ | 56,4000 | 13,5204 | 47,8000 | 1,7889 | 290,9484 | 12,5663 |
| | $\omega = 11$ | 84 | 1,7321 | 48,2000 | 1,6432 | 322,3373 | 11,8441 |
| | $\omega = 21$ | 92,8000 | 2,4900 | 50 | 1,4142 | 438,9831 | 44,4846 |
| **GM** | $\omega = 3$ | 38 | 2,2361 | 42,8000 | 1,6432 | 345,9500 | 10,4307 |
| | $\omega = 7$ | 47,8000 | 4,3818 | 48 | 1,2247 | 295,4132 | 7,8030 |
| | $\omega = 11$ | 64 | 2,2361 | 48,6000 | 1,6733 | 321,1415 | 12,3836 |
| | $\omega = 21$ | 76,6000 | 3,8471 | 49,8000 | 0,4472 | 544,9645 | 100,9324 |
| **GR** | $\omega = 3$ | 20 | 2 | 43,4000 | 1,6733 | 276,7893 | 4,9198 |
| | $\omega = 7$ | 27,6000 | 1,6733 | 48 | 1,8708 | 293,1560 | 11,9489 |
| | $\omega = 11$ | 50,6000 | 4,8270 | 47,4000 | 1,1402 | 431,7106 | 26,3634 |
| | $\omega = 21$ | 57 | 3,9370 | 50 | 0 | 462,4798 | 14,6396 |

Tabela 56: Resultados da Arq2 para o problema de reconhecimento de algarismos com ruído de 10%

| | | **Acurácia** | | **Épocas** | | **Tempo** | |
|---|---|---|---|---|---|---|---|
| **Grupo** | **Resolução** | **Média** | **Desvio padrão** | **Média** | **Desvio padrão** | **Média** | **Desvio Padrão** |
| **GB** | $\omega = 3$ | 80,2000 | 2,7749 | 50 | 1,2247 | 119,6343 | 10,4080 |
| | $\omega = 7$ | 86 | 1,5811 | 52,6000 | 4,2778 | 121,9974 | 11,5707 |
| | $\omega = 11$ | 89,6000 | 1,6733 | 53,8000 | 1,4832 | 122,4965 | 5,1961 |
| | $\omega = 21$ | 94,6000 | 2,3022 | 58 | 3 | 198,2749 | 15,0340 |
| **GM** | $\omega = 3$ | 62,2000 | 3,2711 | 50,2000 | 1,6432 | 135,5269 | 16,4342 |
| | $\omega = 7$ | 64,2000 | 2,1679 | 51 | 2 | 123,0107 | 5,6658 |
| | $\omega = 11$ | 78 | 3,6742 | 53,2000 | 2,7749 | 95,9877 | 5,3128 |
| | $\omega = 21$ | 83 | 2,1213 | 60 | 1,5811 | 211,2409 | 13,7861 |
| **GR** | $\omega = 3$ | 47,6000 | 1,8166 | 50 | 1,8708 | 120,4885 | 11,9094 |
| | $\omega = 7$ | 52 | 2,2361 | 53,2000 | 3,1145 | 117,0043 | 8,1311 |
| | $\omega = 11$ | 60,6000 | 6,2290 | 53,8000 | 1,3038 | 97,2257 | 2,5162 |
| | $\omega = 21$ | 57,8000 | 6,7231 | 59 | 2,8284 | 189,5037 | 21,3433 |

Tabela 57: Resultados da Arq3 para o problema de reconhecimento de algarismos com ruído de 10%

| | | **Acurácia** | | **Épocas** | | **Tempo** | |
|---|---|---|---|---|---|---|---|
| **Grupo** | **Resolução** | **Média** | **Desvio padrão** | **Média** | **Desvio padrão** | **Média** | **Desvio Padrão** |
| **GB** | $\omega = 3$ | 80,2000 | 2,7749 | 50 | 1,2247 | 119,6343 | 10,4080 |
| | $\omega = 7$ | 86 | 1,5811 | 52,6000 | 4,2778 | 121,9974 | 11,5707 |
| | $\omega = 11$ | 89,6000 | 1,6733 | 53,8000 | 1,4832 | 122,4965 | 5,1961 |
| | $\omega = 21$ | 94,6000 | 2,3022 | 58 | 3 | 198,2749 | 15,0340 |
| **GM** | $\omega = 3$ | 62,2000 | 3,2711 | 50,2000 | 1,6432 | 135,5269 | 16,4342 |
| | $\omega = 7$ | 64,2000 | 2,1679 | 51 | 2 | 123,0107 | 5,6658 |
| | $\omega = 11$ | 78 | 3,6742 | 53,2000 | 2,7749 | 95,9877 | 5,3128 |
| | $\omega = 21$ | 83 | 2,1213 | 60 | 1,5811 | 211,2409 | 13,7861 |
| **GR** | $\omega = 3$ | 47,6000 | 1,8166 | 50 | 1,8708 | 120,4885 | 11,9094 |
| | $\omega = 7$ | 52 | 2,2361 | 53,2000 | 3,1145 | 117,0043 | 8,1311 |
| | $\omega = 11$ | 60,6000 | 6,2290 | 53,8000 | 1,3038 | 97,2257 | 2,5162 |
| | $\omega = 21$ | 57,8000 | 6,7231 | 59 | 2,8284 | 189,5037 | 21,3433 |

Tabela 58: Resultados da Arq1 para o problema de reconhecimento de algarismos com ruído de 15%

| | | **Acurácia** | | **Épocas** | | **Tempo** | |
|---|---|---|---|---|---|---|---|
| **Grupo** | **Resolução** | **Média** | **Desvio padrão** | **Média** | **Desvio padrão** | **Média** | **Desvio Padrão** |
| **GB** | $\omega = 3$ | 32 | 5,7879 | 42,6000 | 2,0736 | 221,9508 | 33,0397 |
| | $\omega = 7$ | 59,2000 | 6,7231 | 48 | 1 | 256,8481 | 7,9351 |
| | $\omega = 11$ | 83,2000 | 3,5637 | 47,6000 | 0,8944 | 356,7938 | 17,4386 |
| | $\omega = 21$ | 91,2000 | 3,3466 | 49,8000 | 0,4472 | 293,4557 | 15,7388 |
| **GM** | $\omega = 3$ | 24 | 2 | 43,2000 | 1,3038 | 228,8024 | 18,8148 |
| | $\omega = 7$ | 38,4000 | 5,1769 | 48,8000 | 0,4472 | 259,9985 | 2,6330 |
| | $\omega = 11$ | 57,8000 | 5,3572 | 46,6000 | 2,3022 | 314,6945 | 55,2197 |
| | $\omega = 21$ | 73,8000 | 2,2804 | 49,8000 | 0,4472 | 265,0981 | 3,2508 |
| **GR** | $\omega = 3$ | 19 | 1,4142 | 43,6000 | 1,1402 | 211,4202 | 13,9271 |
| | $\omega = 7$ | 26,8000 | 4,0866 | 47,6000 | 0,5477 | 248,2073 | 10,7377 |
| | $\omega = 11$ | 45,6000 | 2,3022 | 49 | 0,7071 | 359,3239 | 69,3742 |
| | $\omega = 21$ | 60,2000 | 2,2804 | 49,6000 | 0,5477 | 265,9550 | 3,3042 |

Tabela 59: Resultados da Arq2 para o problema de reconhecimento de algarismos com ruído de 15%

| | | **Acurácia** | | **Épocas** | | **Tempo** | |
|---|---|---|---|---|---|---|---|
| **Grupo** | **Resolução** | **Média** | **Desvio padrão** | **Média** | **Desvio padrão** | **Média** | **Desvio Padrão** |
| **GB** | $\omega = 3$ | 77 | 1,4142 | 50 | 1,2247 | 98,8409 | 2,6170 |
| | $\omega = 7$ | 80,2000 | 2,9496 | 51,6000 | 0,8944 | 97,3285 | 2,4172 |
| | $\omega = 11$ | 87,6000 | 2,5100 | 55,2000 | 3,4928 | 111,4965 | 7,7654 |
| | $\omega = 21$ | 94,2000 | 1,9235 | 58,6000 | 1,5166 | 147,0770 | 13,7135 |
| **GM** | $\omega = 3$ | 57,2000 | 2,2804 | 49,6000 | 1,5166 | 91,0267 | 2,9914 |
| | $\omega = 7$ | 63 | 5,8310 | 53,2000 | 1,7889 | 102,7519 | 3,5774 |
| | $\omega = 11$ | 74,6000 | 2,8810 | 55 | 2,4495 | 112,0787 | 4,9340 |
| | $\omega = 21$ | 78,8000 | 2,7749 | 60,4000 | 3,2863 | 241,3064 | 23,9993 |
| **GR** | $\omega = 3$ | 38,2000 | 2,9496 | 50,6000 | 1,6733 | 97,7220 | 8,4581 |
| | $\omega = 7$ | 41,6000 | 4,1593 | 53 | 1,7321 | 100,5834 | 3,3828 |
| | $\omega = 11$ | 57,6000 | 3,7815 | 53,4000 | 2,3022 | 150,7990 | 16,6327 |
| | $\omega = 21$ | 59,2000 | 4,2071 | 58,2000 | 1,0954 | 226,6346 | 14,7074 |

Tabela 60: Resultados da Arq3 para o problema de reconhecimento de algarismos com ruído de 15%

| | | **Acurácia** | | **Épocas** | | **Tempo** | |
|---|---|---|---|---|---|---|---|
| **Grupo** | **Resolução** | **Média** | **Desvio padrão** | **Média** | **Desvio padrão** | **Média** | **Desvio Padrão** |
| **GB** | $\omega = 3$ | 87,6000 | 2,3022 | 50,2000 | 0,8367 | 294,5546 | 14,5532 |
| | $\omega = 7$ | 86,4000 | 2,0736 | 56,6000 | 3,5071 | 254,1923 | 18,3116 |
| | $\omega = 11$ | 91,8000 | 1,9235 | 51,6000 | 1,3416 | 308,4159 | 17,4017 |
| | $\omega = 21$ | 96,4000 | 1,1402 | 55,8000 | 0,8367 | 679,0735 | 12,0957 |
| **GM** | $\omega = 3$ | 75,2000 | 2,5884 | 50,2000 | 0,4472 | 159,8186 | 35,6185 |
| | $\omega = 7$ | 72,6000 | 3,1305 | 58,4000 | 1,8166 | 223,0491 | 8,2368 |
| | $\omega = 11$ | 81,8000 | 4,2071 | 51,8000 | 1,4832 | 295,7728 | 29,5703 |
| | $\omega = 21$ | 85 | 2,3452 | 59,2000 | 1,7889 | 713,9768 | 56,4419 |
| **GR** | $\omega = 3$ | 54,6000 | 2,4083 | 50,6000 | 1,1402 | 153,4446 | 29,9346 |
| | $\omega = 7$ | 52,4000 | 3,2863 | 54,2000 | 1,7889 | 204,4445 | 8,0306 |
| | $\omega = 11$ | 60,6000 | 5,5946 | 51,2000 | 0,4472 | 308,1676 | 28,6052 |
| | $\omega = 21$ | 61,8000 | 5,9330 | 58,6000 | 1,1402 | 710,9235 | 38,8266 |

Tabela 61: Resultados da Arq1 para o problema de reconhecimento de algarismos com ruído de 20%

| | | **Acurácia** | | **Épocas** | | **Tempo** | |
|---|---|---|---|---|---|---|---|
| **Grupo** | **Resolução** | **Média** | **Desvio padrão** | **Média** | **Desvio padrão** | **Média** | **Desvio Padrão** |
| **GB** | $\omega = 3$ | 10,6000 | 6,3482 | 42,2000 | 1,7889 | 283,7329 | 12,9274 |
| | $\omega = 7$ | 17,6000 | 7,3007 | 47,6000 | 0,8944 | 442,3142 | 46,8266 |
| | $\omega = 11$ | 44 | 3,1623 | 48,4000 | 1,1402 | 305,9201 | 12,2580 |
| | $\omega = 21$ | 53,4000 | 3,2094 | 49,4000 | 0,5477 | 361,8761 | 5,2198 |
| **GM** | $\omega = 3$ | 17,2000 | 2,7749 | 42,4000 | 0,8944 | 260,1189 | 7,7304 |
| | $\omega = 7$ | 17,6000 | 1,1402 | 48,4000 | 1,1402 | 323,0202 | 31,6872 |
| | $\omega = 11$ | 40,6000 | 1,6733 | 47,2000 | 1,3038 | 305,7722 | 11,3619 |
| | $\omega = 21$ | 48,2000 | 4,9193 | 49,8000 | 0,4472 | 364,2709 | 3,3622 |
| **GR** | $\omega = 3$ | 20 | 1,8708 | 41,8000 | 1,0954 | 255,7596 | 5,1704 |
| | $\omega = 7$ | 25,4000 | 2,3022 | 48,6000 | 1,1402 | 267,0757 | 12,0057 |
| | $\omega = 11$ | 42,8000 | 4,7645 | 48,4000 | 0,5477 | 303,6072 | 19,7528 |
| | $\omega = 21$ | 52,8000 | 4,9699 | 49,6000 | 0,5477 | 367,6501 | 4,6066 |

Tabela 62: Resultados da Arq2 para o problema de reconhecimento de algarismos com ruído de 20%

| | | **Acurácia** | | **Épocas** | | **Tempo** | |
|---|---|---|---|---|---|---|---|
| **Grupo** | **Resolução** | **Média** | **Desvio padrão** | **Média** | **Desvio padrão** | **Média** | **Desvio Padrão** |
| **GB** | $\omega = 3$ | 33 | 1,2247 | 51 | 1,2247 | 87,3083 | 2,7980 |
| | $\omega = 7$ | 35,4000 | 3,1305 | 52,6000 | 1,3416 | 113,7554 | 8,3959 |
| | $\omega = 11$ | 52,4000 | 2,1909 | 53,4000 | 2,4083 | 137,7854 | 6,5594 |
| | $\omega = 21$ | 55,8000 | 1,6432 | 57,8000 | 1,6432 | 112,4251 | 3,8399 |
| **GM** | $\omega = 3$ | 32,8000 | 1,3038 | 50,8000 | 0,4472 | 109,7153 | 14,4550 |
| | $\omega = 7$ | 37,2000 | 2,6833 | 53,6000 | 3,4351 | 106,8620 | 7,8051 |
| | $\omega = 11$ | 51,4000 | 1,6733 | 53,4000 | 2,4083 | 135,8608 | 7,4678 |
| | $\omega = 21$ | 53,2000 | 2,2804 | 58,6000 | 1,8166 | 162,2008 | 8,1840 |
| **GR** | $\omega = 3$ | 38,8000 | 1,9235 | 50,8000 | 1,9235 | 97,9355 | 3,7765 |
| | $\omega = 7$ | 39,4000 | 3,3615 | 51,4000 | 1,5166 | 98,9864 | 2,4311 |
| | $\omega = 11$ | 52,2000 | 5,9749 | 52,4000 | 2,0736 | 103,0508 | 4,7130 |
| | $\omega = 21$ | 58,6000 | 5,0299 | 58,2000 | 1,6432 | 162,3120 | 5,0431 |

Tabela 63: Resultados da Arq3 para o problema de reconhecimento de algarismos com ruído de 20%

| | | **Acurácia** | | **Épocas** | | **Tempo** | |
|---|---|---|---|---|---|---|---|
| **Grupo** | **Resolução** | **Média** | **Desvio padrão** | **Média** | **Desvio padrão** | **Média** | **Desvio Padrão** |
| **GB** | $\omega = 3$ | 51,4000 | 1,6733 | 49,4000 | 1,9494 | 168,8378 | 7,1720 |
| | $\omega = 7$ | 51,8000 | 1,9235 | 54,6000 | 2,7019 | 250,5384 | 7,1178 |
| | $\omega = 11$ | 57,6000 | 1,3416 | 52,6000 | 1,5166 | 311,6811 | 44,7038 |
| | $\omega = 21$ | 59,6000 | 1,5166 | 59,6000 | 1,6733 | 635,3428 | 19,5136 |
| **GM** | $\omega = 3$ | 49,8000 | 3,8341 | 49,2000 | 0,8367 | 156,1385 | 14,4550 |
| | $\omega = 7$ | 48,4000 | 3,9115 | 56,4000 | 4,4497 | 256,4427 | 7,8051 |
| | $\omega = 11$ | 56 | 1,5811 | 51,4000 | 1,6733 | 317,2329 | 7,4678 |
| | $\omega = 21$ | 57,4000 | 2,4083 | 56,8000 | 2,5884 | 601,5546 | 8,1840 |
| **GR** | $\omega = 3$ | 54,8000 | 2,7749 | 50,4000 | 1,1402 | 159,1519 | 4,7592 |
| | $\omega = 7$ | 48,2000 | 3,1145 | 56,2000 | 3,4205 | 375,1963 | 94,2486 |
| | $\omega = 11$ | 60,2000 | 5,2154 | 52,4000 | 1,6733 | 335,3358 | 17,9655 |
| | $\omega = 21$ | 61,2000 | 2,9496 | 57,4000 | 1,5166 | 604,9992 | 16,2867 |