Inteligência Computacional

Meta II | Skin Diseases

2025/2026

Daniel Silva

2023144551

Guilherme Martins

2023144573

Índice

i. Computação Swarm: O Paradigma da Inteligência Coletiva

O Paradigma "Swarm"

A Computação Swarm é um paradigma da Inteligência Artificial focado no estudo e modelação do comportamento coletivo de sistemas descentralizados e auto-organizados. A sua inspiração é puramente biológica, observando como é que "enxames" de agentes relativamente simples conseguem, através da cooperação, gerar um comportamento global inteligente e complexo.

Exemplos clássicos na natureza incluem:

- Colónias de formigas a encontrar o caminho mais curto para uma fonte de alimento.
- Bandos de pássaros ou cardumes de peixes a moverem-se de forma coordenada para evitar predadores.
- Enxames de abelhas a otimizar a recolha de néctar.

O pilar deste paradigma, introduzido por Beni e Wang em 1989, não assenta na inteligência de um agente individual, mas sim na inteligência emergente que surge das interações locais entre os membros do grupo e entre os membros e o seu ambiente.

Este comportamento coletivo é guiado por regras simples seguidas por cada indivíduo, que se baseiam no *feedback* que recebem do ambiente e das ações dos seus vizinhos. Conceitos fundamentais para que este comportamento de enxame funcione incluem a auto-organização e a divisão de trabalho.

Uma das características mais valiosas dos algoritmos de *swarm* é a sua robustez. A presença de múltiplos agentes a explorar o espaço de soluções de forma estocástica (com um elemento de aleatoriedade) torna-os particularmente eficazes a **evitar mínimos locais** — soluções que parecem ótimas localmente, mas que não o são globalmente.

Aplicações no Treino de Redes Neuronais

No contexto do treino de redes neuronais, o paradigma *swarm* não é tipicamente usado para o processo de aprendizagem em si, mas sim como uma poderosa meta-heurística de otimização.

A sua aplicação mais direta e relevante para este projeto é a otimização de hiper-parâmetros.

Uma rede neuronal possui dois tipos de parâmetros:

1. **Parâmetros (Pesos):** São os valores das ligações entre neurónios, afinados durante o processo de treino (ex: pela retropropagação).

2. **Hiper-parâmetros:** São as definições de configuração da arquitetura da rede, que *não são aprendidas* durante o treino, mas que o definem. Exemplos incluem a taxa de aprendizagem, o número de camadas ocultas, o número de neurónios por camada ou o batch size.

O desempenho de uma rede neuronal é extremamente sensível a estes hiper-parâmetros. Encontrar a combinação ideal manualmente é impraticável, e métodos como a "pesquisa em grelha" tornam-se computacionalmente excessivos à medida que o número de hiper-parâmetros aumenta.

É aqui que a Computação Swarm demonstra o seu valor:

- **Representação da Solução:** Num algoritmo de *swarm* (como o PSO ou o Whale Swarm Algorithm), cada "agente" (partícula, baleia, etc.) representa uma **solução candidata**. No nosso caso, a "posição" de cada agente no espaço de procura é um conjunto completo de hiper-parâmetros para a rede neuronal.
    - *Exemplo:* A Partícula 1 pode representar {*learning rate*=0.01, *neurónios*=64, *camadas*=2}.
- **Função de Aptidão (Fitness):** Para avaliar a "qualidade" ou "aptidão" de cada agente, o sistema treina (ou valida) uma rede neuronal com os hiper-parâmetros que esse agente representa. A *fitness* do agente é, então, o desempenho da rede.
- **Processo de Otimização:** O enxame explora coletivamente o espaço de possíveis configurações. Os agentes partilham informação (direta ou indiretamente) sobre quais as configurações que produziram melhores resultados, e o enxame converge, iterativamente, para a configuração de hiper-parâmetros que maximiza o desempenho da rede.

II. Algoritmo WSA

O Algoritmo Selecionado: Whale Optimization Algorithm (WOA)

O WOA é um algoritmo meta-heurístico mais recente, inspirado na estratégia de caça única das baleias-de-bossa (jubartes), conhecida como **"bubble-net feeding"** (alimentação com rede de bolhas).

Como Funciona: O WOA não utiliza os conceitos de velocidade ou de memória pessoal (pbest) como o PSO. Em vez disso, o algoritmo simula três comportamentos de caça distintos para atualizar a posição das "baleias" (soluções)

1. Fase de Explotação: Encircling Prey (Cercar a Presa)

Inicialmente, assume-se que a melhor solução candidata encontrada até ao momento (o gbest do enxame) é a "presa" ou está muito perto dela. As outras baleias (agentes) tentam atualizar a sua posição nadando em direção a esta melhor solução, "cercando-a".

2. Fase de Explotação: Bubble-Net Attack (Ataque com Rede de Bolhas)

Esta é a fase que define o WOA e simula o comportamento real das baleias. Para explorar localmente e com precisão a área em torno da melhor solução (gbest), o WOA utiliza uma **equação de espiral logarítmica**. Cada baleia atualiza a sua posição calculando uma trajetória em espiral entre a sua posição atual e a posição da presa (gbest). Isto permite uma **explotação (refinamento) muito fina** da solução, simulando a baleia a nadar em espiral enquanto liberta bolhas.

Durante a fase de explotação, o algoritmo escolhe aleatoriamente entre o método de "cercar" (fase 1) e o método da "espiral" (fase 2).

3. Fase de Exploração: Search for Prey (Procurar a Presa)

Para garantir que o algoritmo não fica preso num ótimo local (exploração global), o WOA força algumas baleias a procurar aleatoriamente. Em vez de nadar em direção à *melhor baleia* (gbest), o agente atualiza a sua posição com base na posição de uma **baleia selecionada aleatoriamente** do enxame. Esta "deslocação" força o agente a mover-se para uma região diferente do espaço de procura, promovendo a exploração.

O algoritmo gere o balanço entre exploração (fase 3) e explotação (fases 1 e 2) através de um parâmetro de controlo A que diminui ao longo das iterações, favorecendo a exploração no início e a explotação no final.

Análise Comparativa: WOA vs. PSO

Característica	Particle Swarm Optimization (PSO)	Whale Optimization Algorithm (WOA)
Inspiração	Comportamento social de pássaros/peixes.	Caça "bubble-net" das baleias-de-bossa.
Componentes Principais	Posição, **Velocidade**.	Apenas Posição.
Memória do Agente	**pbest** (Memória pessoal da melhor posição).	Nenhuma. O agente "esquece" as suas posições anteriores.
Memória Coletiva	**gbest** (Melhor posição global do enxame).	**gbest** (Usada como a "presa" ou melhor solução).
Mecanismo de Explotação	Movimento linear em direção ao pbest e gbest.	**Duplo:** Movimento de "cerco" e um movimento em **espiral logarítmica** (bubble-net).
Mecanismo de Exploração	Inércia da velocidade; balanço com pbest e gbest.	Movimento em direção a um agente **aleatório** do enxame.
Parâmetros de Controlo	Inércia (w), Fator Cognitivo (c1), Fator Social (c2).	Menos parâmetros. O balanço exploração/explotação é gerido internamente (ex: a, p).

Vantagens e Desvantagens do WOA (vs. PSO)

Vantagens

- **Melhor Explotação Local:** A principal vantagem do WOA é o seu mecanismo de ataque em espiral. Esta explotação baseada numa espiral logarítmica permite um refinamento muito preciso da solução em torno do ótimo encontrado, sendo (em teoria) mais eficaz na "afinação" final do que o movimento linear do PSO.
- **Menos Parâmetros para Afinar:** O PSO clássico depende de três parâmetros cruciais (w, c1, c2) que afetam drasticamente o seu desempenho e que, ironicamente, teriam de ser otimizados. O WOA tem menos parâmetros de controlo, com o seu balanço exploração/explotação a ser gerido de forma mais fluida e adaptativa.
- **Simplicidade Conceptual:** A ausência de velocity e pbest torna a atualização de cada agente mais simples de implementar e calcular em cada passo.

Desvantagens

- **Convergência Prematura:** Tal como o PSO, o WOA não está imune a convergir prematuramente para ótimos locais. Se a "presa" (gbest) for um ótimo local, a forte capacidade de explotação (espiral) do WOA pode fazer com que todo o enxame "mergulhe" muito rapidamente nessa solução sub-ótima, abandonando a exploração global.
- **Ausência de Memória Pessoal:** A falta de pbest é uma faca de dois gumes. No PSO, se uma partícula encontra uma boa região (o seu pbest) que não é o gbest, ela retém uma "memória" dessa área. No WOA, se o gbest se mover para longe, o agente "esquece" as boas soluções que encontrou individualmente, dependendo mais da posição do líder ou de um agente aleatório.

iii. Função Ackley

Para validar o desempenho do algoritmo selecionado (WSA) contra a base de referência (PSO), foi utilizada a função de *benchmark* **Ackley**. O objetivo é encontrar o seu mínimo global de 0.0 (um *fitness* mais baixo indica um melhor desempenho).

$$f(x_1, x_2) = -20exp([0.2\sqrt{0.5(x_1^2 + x_2^2)}]) - exp([0.5(\cos 2\pi x_1 + \cos 2\pi x_2)]) + e + 20$$

Metodologia de Teste

Os testes foram realizados em **dimensão 2 (D2)** e **dimensão 3 (D3)**. Cada cenário foi executado 10 vezes para garantir a robustez estatística, sendo reportados os valores médios e mínimos.

A análise foi dividida em duas partes:

1. **Parte A (Comparação Direta):** Uma comparação "justa" entre WSA e PSO (30 agentes, 50 iterações).
2. **Parte B (Análise de Sensibilidade):** Uma análise focada apenas no WSA, variando o número de agentes (n) e de iterações (iteration).

Parte A: Comparação WSA vs. PSO

Ambos os algoritmos foram executados com configurações idênticas. Os resultados médios e mínimos estão compilados na Tabela 1.

Tabela 1: Comparação de Desempenho (WSA vs. PSO) na Função de Ackley *Valores de Fitness (Média e Mínimo) após 10 execuções. Menor = Melhor.*

Dimensão	Algoritmo	Fitness Médio	Melhor Fitness (Min)
D2	**WSA**	2.310881	0.000001
D2	**PSO**	**0.000008**	**0.000001**
D3	**WSA**	6.277727	2.335517
D3	**PSO**	**0.000070**	**0.000034**

Análise dos Resultados (Parte A):

Os resultados da Tabela 1 demonstram uma **superioridade clara e esmagadora do PSO** sobre a implementação do WSA testada.

1. **Dimensão 2 (D2):** Embora ambos os algoritmos tenham conseguido encontrar o ótimo global (Min: 0.000001) pelo menos uma vez, o desempenho *médio* do PSO (0.000008) foi ordens de magnitude melhor que o do WSA (2.310). Isto indica que o PSO converge de forma muito mais consistente para o ótimo.

2. **Dimensão 3 (D3):** A diferença é ainda mais pronunciada. O PSO continuou a convergir consistentemente para o ótimo global (Média: 0.000070). Em contraste, o WSA falhou em todas as 10 execuções, apresentando um *fitness* médio de 6.277 e um melhor caso de apenas 2.335, mostrando grande dificuldade em otimizar num espaço de procura ligeiramente mais complexo.

Parte B: Análise de Sensibilidade do WSA

Foi analisado o impacto dos principais parâmetros do WSA no seu fraco desempenho.

Tabela 2: Análise de Sensibilidade do WSA na Função de Ackley (D2 e D3) *Valores de Fitness Médio após 10 execuções. Menor = Melhor.*

Parâmetro em Teste	Valor	Fitness Médio (D2)	Fitness Médio (D3)
Nº Agentes	10	5.940281	9.892389
(Iter. fixas = 50)	30	1.187439	4.236208
	50	0.019352	1.940404
	100	**0.002573**	**0.826130**
Nº Iterações	20	1.063675	5.437898
(Agentes fixos = 30)	50	1.517295	3.206530
	100	1.437320	4.389949
	200	1.602032	3.619954

Análise dos Resultados (Parte B):

1. **Sensibilidade ao Nº de Agentes:** O desempenho do WSA é **extremamente dependente** do número de agentes. Em ambas as dimensões, o *fitness* médio melhorou drasticamente à medida que o enxame aumentou (ex: em D2, de 5.94 para 0.0025). Isto sugere que a versão do WSA testada sofre de uma exploração muito fraca e requer um enxame anormalmente grande para "tropeçar" na solução correta.

2. **Sensibilidade ao Nº de Iterações:** Os resultados para as iterações (com agentes fixos em 30) são erráticos e não mostram uma melhoria clara com mais tempo de execução. Em D2, o desempenho até piorou com mais iterações. Isto reforça a conclusão da Parte A: o algoritmo está a convergir prematuramente para um ótimo local, e mais iterações (mais tempo) não o ajudam a escapar desse mínimo local.

Conclusão: Os testes de *benchmark* demonstram que, para a função Ackley, a implementação base do PSO é vastamente superior à implementação do WSA fornecida (Functions/wsa.py). O WSA demonstrou grande dificuldade em convergir, ficando preso em

ótimos locais. A sua única melhoria clara provém de um aumento substancial do número de agentes, sugerindo uma fraca capacidade de exploração global.

iv. Otimização de Hiper-parâmetros na Rede CNN

1. Configuração da Otimização

A otimização focou-se em determinar a melhor configuração para dois dos hiperparâmetros mais influentes no desempenho de uma Rede Neuronal Convolucional (CNN):

1. **Coeficiente de Aprendizagem (learning_rate):** Controla a magnitude do ajuste dos pesos durante o treino. Um valor ótimo previne uma convergência demasiado lenta ou uma divergência instável.
2. **Número de Neurónios (num_neurons):** Define a capacidade da camada densa final (camada oculta). Um número ótimo equilibra a capacidade de aprender padrões complexos (underfitting) com o risco de memorização dos dados de treino (overfitting).

Espaço de Procura (Search Space) Para guiar os algoritmos, foram definidos os seguintes limites (inferior e superior) para a procura:

- **learning_rate:** [0.0001, 0.01]
- **num_neurons:** [32, 128]

Função de Fitness e a Metodologia do objetivo (fitness) dos algoritmos foi minimizar a Perda de Validação (val_loss).

Treinar centenas de modelos no *dataset* completo (29.889 imagens) seria computacionalmente complexo. Para agilizar o processo, foi adotada uma estratégia de aproximação:

1. **Dataset de Otimização (64x64):** Foi utilizado um *dataset* de treino reduzido (temp_train, 5.000 imagens) e um *dataset* de validação (val, 3.788 imagens), ambos redimensionados para 64x64.
2. **Treino Rápido:** Cada agente (solução candidata) foi treinado por apenas 10 épocas.

Esta abordagem permitiu aos algoritmos (WSA e PSO) avaliarem “rapidamente” se uma configuração era "promissora" ou "desastrosa", sem o custo do treino completo.

Para a competição, ambos os algoritmos foram executados com 5 agentes durante 10 iterações.

2. Resultados da Otimização (WSA vs. PSO)

Ambos os algoritmos exploraram o espaço de procura. O algoritmo WSA demonstrou um comportamento de convergência estável, explorando e refinando progressivamente a sua melhor solução.

O PSO, no entanto, demonstrou uma instabilidade notável. Conforme o *log* da sua execução, vários agentes "explodiram", resultando em learning_rates negativos (ex: -0.001606) — um valor inválido. Isto levou a uma val_loss astronómica (ex: 2.409e+11), indicando uma falha catastrófica do modelo. Esta instabilidade sugere que os parâmetros de velocidade (w, c1, c2) do PSO não estavam otimizados, fazendo com que as partículas "disparassem" para fora do espaço de procura válido.

No final da execução, os melhores resultados encontrados por cada algoritmo (após reavaliação) foram os seguintes:

Tabela 1: Resultados da Competição de Otimização (WSA vs. PSO) ***Objetivo: Minimizar Val Loss.***

Algoritmo	Melhor Val Loss (Min)	Learning Rate (LR) Vencedor	Neurónios Vencedores
WSA	**0.98427**	**0.000352**	**96**
PSO	**1.02821**	**0.000100**	**51**

Conclusão da Otimização: O WSA foi o vencedor. Demonstrou ser um algoritmo mais robusto e estável para este problema, encontrando uma configuração de hiperparâmetros (LR=0.000352, Neurónios=96) que resultou na menor perda de validação.

3. Treino e Avaliação do Modelo Final

A configuração vencedora encontrada pelo WSA foi então usada para treinar o modelo final. Este treino utilizou a metodologia completa:

- **Dataset:** O conjunto de treino completo (29.889 imagens) com *data augmentation*.
- **Tamanho:** Imagens de 128x128.
- **Épocas:** 30 épocas (com EarlyStopping e ReduceLROnPlateau).

O modelo final foi treinado durante 25 épocas, parando devido ao EarlyStopping (indicando que a melhor performance foi atingida).

4. Avaliação de Desempenho no Conjunto de Teste

O modelo final otimizado foi avaliado no conjunto de teste (3.792 imagens) para aferir o seu desempenho em dados nunca vistos.

Tabela 2: Métricas Finais do Modelo Otimizado (WSA)

Métrica	Valor
Acurácia (Accuracy)	**0.701 (70.1%)**
AUC (macro, OVR)	**0.923 (92.3%)**

O modelo demonstrou uma capacidade de generalização robusta, com uma acurácia de 70.1% e uma pontuação AUC (que mede a capacidade de distinguir entre classes) muito elevada de 92.3%.

Figura 1: Matriz de Confusão do Modelo Otimizado (WSA)

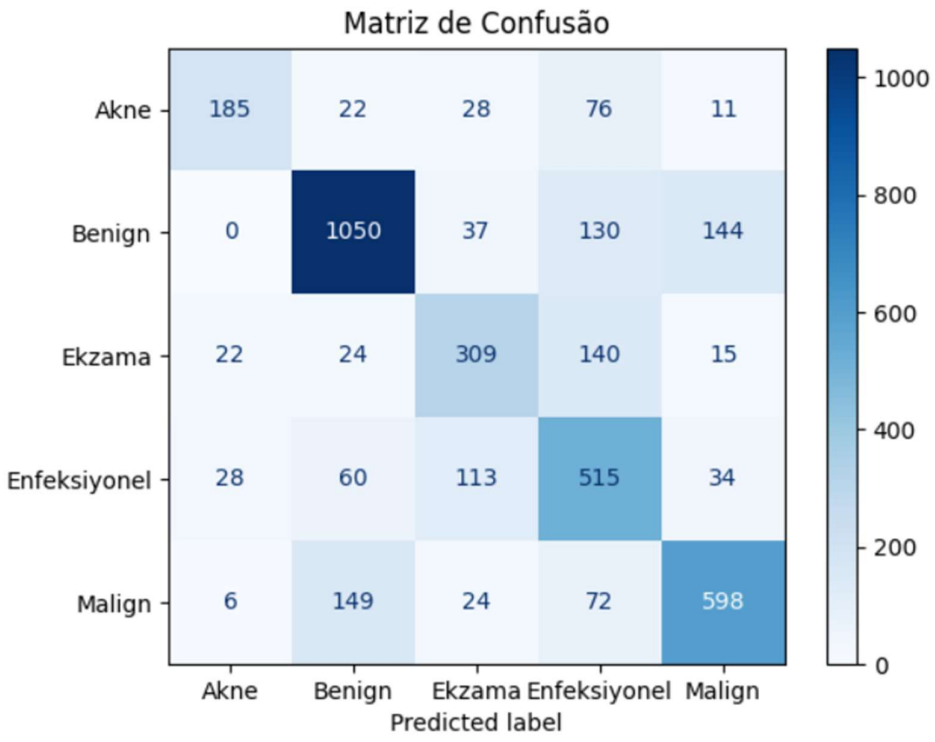

Tabela 3: Métricas de Desempenho por Classe (F1-Score e Sensibilidade)

Classe	Sensibilidade (Recall)	F1-Score
Akne	**0.575**	**0.657**
Benign	**0.771**	**0.788**
Ekzama	**0.606**	**0.605**
Enfeksiyonel	**0.687**	**0.612**
Malign	**0.704**	**0.724**

Discussão dos Resultados (Secção iv): O WSA conseguiu identificar um conjunto de hiperparâmetros (LR=0.000352, Neurónios=96) que produziu um modelo final robusto.

A análise por classe (Tabela 3) e a Matriz de Confusão (Figura 1) mostram que o modelo é particularmente eficaz a identificar as classes "Benign" (1050 corretos) e "Malign" (598 corretos). As classes "Ekzama" (309 corretos) e "Enfeksiyonel" (515 corretos) mostraram-se mais difíceis de distinguir, o que é consistente com a sua similaridade visual (ex: 140 casos de Ekzama foram confundidos com Enfeksiyonel).

Nota: Dada a limitação da procura (apenas 5 agentes por 10 iterações), é muito provável que um val_loss ainda menor exista. Um futuro trabalho deveria executar uma procura mais exaustiva (ex: 20 agentes, 30 iterações), agora que a robustez do WSA foi comprovada.

v. Conclusão e Discussão de resultados

1. Desempenho Teórico Não Garante Sucesso Prático.

Os resultados revelaram uma discrepância clara entre teoria e prática. No *benchmark* de Ackley (Secção III), o PSO demonstrou uma superioridade esmagadora, convergindo para o ótimo global (Média 0.000070) enquanto o WSA falhou (Média 6.277).

No entanto, na aplicação prática na rede neuronal (Secção IV), o cenário inverteu-se: o PSO revelou-se extremamente instável, com agentes a "explodir" para valores inválidos e perdas (val_loss) catastróficas (ex: 2.409e+11). Em contraste, o WSA demonstrou ser muito mais robusto e estável, convergindo de forma fiável. Isto sugere que o WSA se adaptou melhor ao problema "ruidoso" do treino da rede.

2. O *Swarm Intelligence* Superou a Otimização Manual da Fase I.

O WSA cumpriu o objetivo principal, encontrando um conjunto de hiperparâmetros (LR=0.000352, Neurónios=96) superior ao determinado manualmente na Fase I. A otimização automática resultou numa melhoria clara das métricas globais.

Tabela 4: Comparação de Desempenho (Fase I vs. Fase II)

Métrica	Fase I (Otimização Manual)	Fase II (Otimização WSA)	Melhoria
Acurácia (Accuracy)	**0.686 (68.6%)**	**0.701 (70.1%)**	**+1.5%**
AUC (macro, OVR)	**0.915 (91.5%)**	**0.923 (92.3%)**	**+0.8%**

3. A Otimização de Hiperparâmetros Não Corrigiu a Falha Clínica Principal.

Apesar da melhoria na acurácia global, a análise da matriz de confusão da Fase II revela que o problema crítico identificado na Fase I não foi resolvido.

- Fase I (Modelo Manual): Classificou 148 casos de "Malign" (Maligno) como "Benign" (Benigno).
- Fase II (Modelo WSA): Classificou 149 casos de "Malign" como "Benign".

O erro clínico mais perigoso (falso negativo) persistiu. Esta é uma descoberta crucial: sugere que o problema desta confusão específica não reside nos hiperparâmetros (learning_rate ou num_neurons), mas sim em fatores mais profundos, como a arquitetura simplista da CNN ou a similaridade visual extrema entre estas classes.

Conclusão Final e Trabalhos Futuros

Este projeto demonstrou a viabilidade da *Swarm Intelligence*, com o robusto WSA a superar o PSO (instável) e o modelo manual da Fase I. Contudo, a falha em resolver o erro crítico (Maligno vs. Benigno) define claramente os próximos passos. Um trabalho futuro deve focar-se em otimizar a própria arquitetura (ex: número de filtros) ou, como sugerido na conclusão da Fase I, aplicar *Transfer Learning* (VGG16, ResNet) para obter a capacidade de extração de características necessária para distinguir, de forma fiável, as classes clinicamente críticas.