Módulo de Reconhecimento de Imagens de Sistemas Coloidais geradas em simulações no ambiente MASPN

Lucio Colusso Barnewitz[1], **Alexandre O. Zamberlan**[1]

[1]Curso de Ciência da Computação – Universidade Franciscana (UFN)
Santa Maria – RS

luciocbarnewitz@gmail.com, alexz@ufn.edu.br

Abstract. *This research is associated with the Computer Science Practice Laboratory and the Nanosciences Graduate at UFN. The objective is to model and implement a module for the MASPN Project in order to recognize patterns in generated images by MASPN tool simulation. The recognition process utilizes feedback propagating Artificial Neural Networks and multi-layer supervised training (image samples) implemented in Python using OpenCV library, using createsample, traincascade, for example, in order to identify if the images have particle agglomeration process. The module aims to add functionality to the MASPN project, making it more complete and operational for researchers in Nanosciences area.*

Resumo. *Este trabalho está inserido em projeto do Laboratório de Práticas dos cursos de Computação em parceria com a Pós-graduação em Nanociências da UFN. O objetivo principal é modelar e implementar um módulo para o projeto MASPN que realize reconhecimento de padrões em imagens geradas pela simulação da ferramenta MASPN. O processo de reconhecimento utiliza Redes Neurais Artificiais com propagação recorrente e treinamento supervisionado (via amostras de imagens), multicamada implementadas em Python com biblioteca OpenCV, usando createsample, traincascade, por exemplo, para identificar se as imagens apresentam aglomeração de partículas. Esse módulo tem o intuito de agregar funcionalidades ao projeto MASPN, deixando-o mais completo e operacional para pesquisadores da área de Nanociências.*

1. Introdução

O projeto MASPN possui um Portal[1] de gestão de pesquisadores, fármacos, polímeros e experimentos [Pereira et al. 2018], disponível na Web para que laboratórios e centros de pesquisa possam compartilhar experimentos de sistemas coloidais. Entretanto, há inúmeros módulos ainda necessários no ambiente, inclusive um módulo de reconhecimento de padrões em imagens produzidas em simulações da ferramenta MASPN para analisar se houve ou não aglomeração do sistema, ou seja, instabilidade do experimento [Zamberlan et al. 2019].

O objetivo geral do trabalho é modelar, implementar, treinar e avaliar um módulo de reconhecimento de padrões em imagens de simulação produzidas no MASPN [Zamberlan et al. 2019]. Como objetivos específicos, assume-se:

[1]http://maspn.lapinf.com.br.

- Estudar o funcionamento da ferramenta MASPN na parte de geração de simulações;
- Identificar qual rede neural é mais adequada para o reconhecimento de padrão em imagens de simulações;
- Definir variáveis e pesos de entrada na rede neural;
- Definir o mecanismo de treinamento da rede neural, ou seja, treinamento supervisionado ou não supervisionado;
- Prototipar uma rede neural em um módulo do MASPN.

A principal motivação desta pesquisa, justamente é construir um módulo junto ao projeto MASPN que agregue mais funcionalidades às ferramentas, gerando recursos aos pesquisadores da área de Nanociências para que possam identificar sistemas instáveis ou estáveis.

Para melhor compreensão da proposta, o texto está dividido em 6 seções. A seção que segue, trata dos conceitos, técnicas, metodologias e trabalhos relacionados às áreas Reconhecimento de Imagem (RI) e Redes Neurais (RNA). Na Seção 3, são apresentados a metodologia de trabalho e a de desenvolvimento do módulo de reconhecimento, bem como as ferramentas utilizadas no projeto e na implementação. A Seção 4 apresenta os resultados alcançados. Por fim, as Conclusões e as Referências Bibliográficas.

2. Revisão bibliográfica

Nesta seção é apresentado o projeto MASPN, conceitos, técnicas, metodologias e trabalhos relacionados nas áreas da Inteligência Artificial (IA).

2.1. Ambiente MASPN

Produção e caracterização de nanopartículas poliméricas (NPPs), também conhecidas como dispersões coloidais, são processos que exigem tempo e habilidades técnicas para produzir resultados precisos [Pereira et al. 2018]. Conceitualmente, as nanopartículas poliméricas são definidas como partículas coloidais sólidas, que incluem tanto nanoesferas quanto nanocápsulas [Zamberlan et al. 2016a]. Os polímeros, portanto, tem como objetivo proteger o fármaco da degradação, promovendo a liberação sustentada desse fármaco, mantendo concentrações em níveis terapêuticos por determinados períodos de tempo [Zamberlan et al. 2016a].

Simulações computacionais em Nanociência têm auxiliado nesses processos, fornecendo suporte e agilidade para alcançar melhores resultados. Essas simulações necessitam (além de um ambiente de execução e de visualização de resultados por animações 2D) de processos de validação, muitas vezes relacionados com modelos matemáticos embutidos em banco de dados, presentes em sistemas locais (*Desktop*) ou em sistemas distribuídos na Web [Pereira et al. 2018].

O projeto conhecido como "*Multiagent System for Polymerics Nanoparticles MASPN*" [Zamberlan et al. 2016b] possui um ambiente de simulação em que é possível executar simulações para acompanhar e avaliar o efeito de aglomeração em nanopartículas poliméricas, a partir de um conjunto de parâmetros, como distribuição de tamanho das partículas, conteúdo do fármaco presente na partícula, potencial zeta da partícula, pH do ambiente, tipo de polímero e fármaco encapsulado. Os dados de experimentos podem

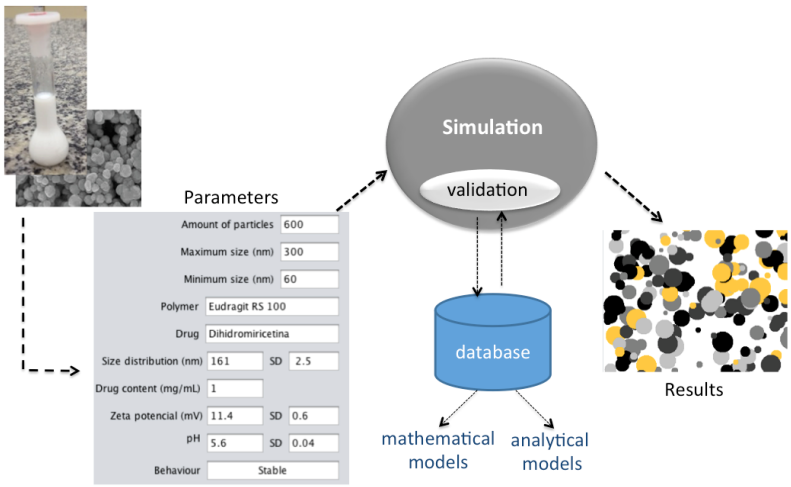


Figura 1. Fluxo para avaliar o processo de aglomeração na ferramenta MASPN [Zamberlan et al. 2016a].

ser inseridos diretamente na ferramenta, contudo, esse recurso é local e isolado (*Desktop*). A Figura 1 ilustra a ideia de uso da ferramenta MASPN no processo de avaliação de aglomeração. A Figura 2 apresenta a ferramenta MASPN de forma geral, ou seja sua interface principal.

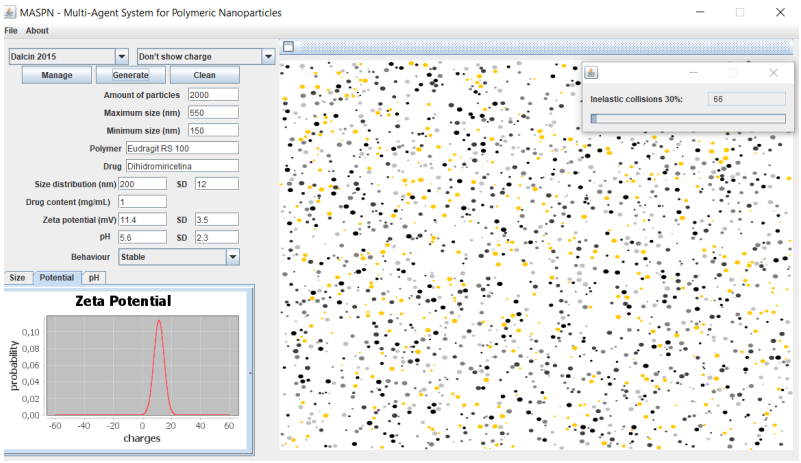


Figura 2. Interface principal do MASPN *Desktop* [Zamberlan et al. 2019].

Na Figura 3 consta os módulos existentes no portal para a gestão de usuários, instituições, fármacos (*drug*), polímeros (*polymer*) e experimentos (*experiment*). Nessa visão, o administrador do sistema pode gerenciar amplamente dados de experimentos com determinados fármacos e polímeros[2].

Registra-se que o módulo de reconhecimento de padrões, a partir das imagens da simulação, é um recurso da ferramenta *Desktop*. No início deste trabalho, pensou-se em associar ao Portal, ou seja, uma funcionalidade Web. Entretanto, decidiu-se inserir o módulo à ferramenta *Desktop* por motivos de fluxo de funcionamento da simulação, que fica na ferramenta, enquanto o Portal gerencia os dados de simulação.

[2]O Portal MASPN está disponível em https://maspn.lapinf.com.br.

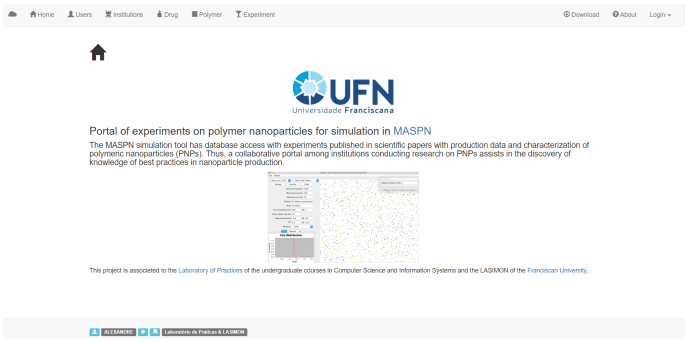


Figura 3. Interface Web da Visão do Administrador [Autor].

2.2. Redes Neurais Artificiais

Uma rede neural artificial (RNA) pode ser comparada ao cérebro humano, bem como suas entradas podem ser comparadas aos neurônios. Cada entrada na rede neural passa por uma análise de importância, associada a um peso. Sendo assim, uma rede neural pode ser composta por várias entradas e cada uma delas recebe um valor peso. Como citado em [Norvig and Russell 2014], as redes neurais são compostas por nós conectados por ligações direcionadas, em que a ligação da unidade *i_entradas* para a unidade *j_intermediários* serve para propagar a ativação e definir um peso numérico para ligação.

Dessa forma, uma RNA pode ser resumidamente definida:

- processador massivamente paralelo que armazena conhecimento experimental;
- modela o cérebro em dois aspectos;
  - o conhecimento é adquirido pela rede por processo de aprendizagem/treinamento;
  - as conexões entre neurônios são usadas para armazenar conhecimento.

A Figura 4 mostra a imagem 'clássica' de uma RNA, com as entradas (a) e seus pesos (w), seguido por um somatório das entradas e pesos, com a função de ativação e a saída esperada. Uma função de ativação tem o papel de tratar o somatório de entradas e pesos e validá-los como uma saída esperada (limiar ou bias). Quando a função de entrada não atingir um limite desejado, é porque é preciso reestruturar as entradas e talvez seus respectivos pesos.

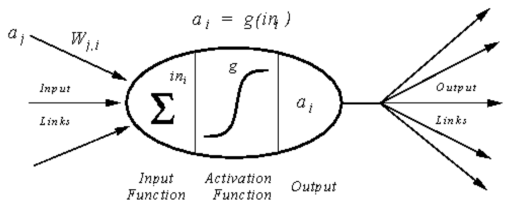


Figura 4. Estrutura matemática de uma RNA [Norvig and Russell 2014].

Em uma RNA também se tem diferentes formas de realizar a conexão entre nós, ou seja, diferentes classificações. São elas a rede com alimentação para frente (*feed-forward propagation*) e a rede recorrente (*feedback propagation*), onde a primeira gera uma ligação de um nó com seu anterior e este com seu posterior, ou seja, nunca formará

ciclos e é totalmente linear, já na segunda opção os nós alimentam suas saídas de volta às suas próprias entradas, o que demonstra os níveis de ativação da rede formam um sistema dinâmico que podem atingir um estado estável ou apresentar oscilações ou comportamento caótico, discutido por [Norvig and Russell 2014]. Esse sistema de alimentação de rede quando bem utilizado podem dar suporte a memória de curto prazo, o que não é possível na rede de alimentação para a frente, o que torna ela mais interessante como modelos de cérebro.

Segundo [Lohmann 2016], uma RNA é composta por dois processos importantes, são eles o processo de treinamento e de reconhecimento, em que no treinamento consiste em apresentar uma grande quantidade de padrões para a RNA a fim desta se adaptar aos possíveis cenários que estão em questão. O treinamento pode ser supervisionado ou não supervisionado. Sendo assim, no padrão supervisionado o comportamento é baseado em duplas, as quais são compostas por algo a ser reconhecido e o que é esperado dele (uma entrada e uma saída esperada). Dessa forma, o treinamento da RNA baseia-se nas diversas duplas que receberá como entrada e irá realizar a devida análise a ponto de conseguir receber apenas o dado a ser reconhecido. Já o treinamento não supervisionado, como consta em [Lohmann 2016], recebe apenas vários dados de entrada (ao contrário do supervisionado que recebe a dupla entrada-saída) e tem a tarefa de analisar o mesmo e reunir com seus similares a partir de algum padrão e pesos determinados pela máquina. A técnica de RNA aplicada em sistemas de comportamento inteligente para reconhecimento de padrões tem sido amplamente utilizada, pois apresenta comportamento humano similar: reconhecer, aprender e reconhecer.

2.3. Tipos de RNA

Na prática, pode-se categorizar RNAs em dois grandes tipos: i) RNAs Recorrentes (Problema imobiliário para definir preço de imóveis; em imagens como colocar *tag* em fotos; em reconhecimento de fala; em traduções); ii) RNAs Convolucionais (Problema de reconhecimento de objetos em movimento em vídeos).

Entretanto, de forma mais específica, pode-se categorizar RNAs em:

- Perceptron - segundo [Cardon et al. 1994], uma Rede Neural Perceptron conta com um algoritmo simples, ou seja, com pequena complexidade computacional, não precisando um grande pré processamento e sendo composto por poucas variáveis. Por outro lado, este método está restrito a realizar reconhecimentos mais básicos, como exemplo se o imóvel é caro de acordo com seu tamanho. Esse método conta com entradas de dados com algum intervalo, aprendizado supervisionado e alimentação à frente. As entradas são informadas e os pesos delas são gerados aleatoriamente. Após realizar a iteração, os pesos se adaptam de acordo com o erro gerado na iteração anterior;
- Kohonen - de acordo com [Martins et al. 2002], a rede Kohonen é uma Rede Neural não supervisionada, a qual conta com um mapa auto-organizável e pode ser formada por neurônios uni ou bidimensionais, os quais modificam seus pesos num processo de aprendizagem competitivo. Assim podem formar uma grade de saída com um sistema de coordenadas significativas para diferentes características de entrada. O processo de aprendizagem deve-se ao neurônio mais próximo do padrão de entrada ser declarado vencedor. Com o neurônio vencedor é determinada uma vizinhança de neurônios entre eles, assim constroem uma base de

cooperação entre vizinhos para correlacionar o padrão de entrada com uma região do mapa auto-organizável. Como explicado em [Martins et al. 2002], essa rede pode desenvolver um classificador de texturas de materiais;

- Convolucionais - segundo [de Andrade Kovaleski 2018], esse tipo de Rede Neural consiste em operações de convolução camada a camada, sendo muito utilizada em reconhecimento em vídeos, analise de objetos, identificação de faces. Isso acontece porque redes convolucionais foram desenvolvidas para trabalhar com dados estruturados localmente, como *pixels*. Cada Neurônio ficará encarregado de uma zona do video, ou seja, uma quantidade de *pixels*, sendo que no final essas partes se complementam e formam a totalidade da imagem. Portanto, da mesma maneira cada neurônio analisa seus vizinhos e passa para a próxima camada essa informação. Ainda informado por [de Andrade Kovaleski 2018], esse tipo de rede é capaz de identificar cores ou bordas, sendo realmente a melhor opção para reconhecimento de elementos em vídeos.

2.4. Reconhecimento de Padrões

Para um Sistema de Reconhecimento de Padrão em imagens operar, é necessário percorrê-las. Sabendo isso, pode-se considerar uma imagem sendo uma grande matriz composta por vários *pixels* horizontais e vários *pixels* verticais. Além de ser estudada as posições de cada ponto da imagem, é necessária a análise de luminosidade e cores, com propósito de uma otimização no tamanho de arquivo da imagem estudada [Erpen 2004].

As cores da imagem podem ser distinguidas em três tipos, sendo o mais básico, uma imagem binária, composta por *pixels* apenas de cor branca, representada pelo símbolo 0, ou da cor preta, representada pelo símbolo 1. Já se a imagem for composta por tons de cinza, tem-se o preto, com a mesma representação pelo símbolo 1 e a quantidade de branco será medida por uma variável, onde será determinado um valor máximo e mínimo para essa variância. E por último haverão as imagens compostas por diversas cores, onde serão representadas pelo espectro RGB, onde se tem 3 variáveis de 0 a 255 e a combinação desses valores gera a cor exata, esse tipo de imagem é o que mais consome espaço em disco e processamento.

Uma imagem é uma matriz bidimensional, que neste estudo, armazena partículas espalhadas no plano (x,y), contendo tamanho e cor.

2.5. Trabalhos relacionados

No trabalho de [Erpen 2004], foi estudado reconhecimento de padrões em imagens com descritores de forma em que se conseguiu desenvolver uma *toolbox* para o reconhecimento de formas através de caracteres. Esse trabalho usou como estudo de caso a leitura de placas de automóveis. No trabalho de [Brito Bisneto 2011], foi estudado reconhecimento de padrões de objetos utilizando Redes Neurais Artificiais e Geometria Fractal, em que se provou que utilizar a geometria fractal geraria melhores resultados, pois a média de acertos da RNA utilizando ela foi mais elevado do que não utilizando RNA. Esse teste foi realizado usando a base de imagens composta por 184 amostras criadas pelos pesquisadores. No trabalho seguinte [Silva 2005], verificou-se a eficiência do uso de uma RNA para reconhecimento de imagens usando treinamento para o algoritmo realizar a analise pixel a pixel, também testando as análises em imagens que não possuem objetos similares na base criada para o projeto. Finalmente, o trabalho de [Silva 2005] foram estudadas

técnicas de RNA para reconhecimento de marcos nas rodovias. Foi utilizado imagens aéreas e o detector *Canny* para identificar os marcos. No trabalho, obteve-se dificuldades em identificar com precisão alguns marcos importantes como passarelas e viadutos. Na Tabela 1, pode-se verificar uma análise quanto às características específicas dos trabalhos citados anteriormente.

Tabela 1. Análise quanto a características dos trabalhos relacionados.

	[Erpen 2004]	[Brito Bisneto 2011]	[Silva 2005]
Linguagem	Matlab	C++	C++
Plataforma	Desktop	Multiplataforma	Multiplataforma
Treinamento	Supervisionado	Supervisionado	Supervisionado
Topologia	Multicamada	Multicamada	Multicamada
Algoritmo	Backpropagation	Backpropagation	Backpropagation

2.6. Scrum Solo

É uma metodologia ágil contendo boas práticas e adaptada para trabalhos com uma ou duas pessoas, como ocorre em um Trabalho Final de Graduação. Ela é composta por tarefas que devem durar uma semana, que também é o tempo intervalo para um *meet* (reunião) e outro. Scrum Solo é muito utilizada com a técnica Kanban, estruturada em (no mínimo) 3 colunas principais nomeadas como *to do* (a fazer), *doing* (em processo), *Done* (finalizado). Uma quarta coluna pode ser o planejamento de *meets* (encontros) e a quinta é alguma tarefa paralisada para se resolver no encontro semanal. Nesta pesquisa, a ferramenta Trello é usada. Em Scrum tem-se nomenclaturas específicas, como *Product backlog* (onde são as pendências, usados como o *to do* do Kanban), o *Sprint Backlog* (onde as tarefas são colocadas na barra *doing*) e os *Sprints* que se trata de um período de tempo que pode variar de uma semana até um mês para realização de alguma implementação de funcionalidade do módulo [Boaglio 2016].

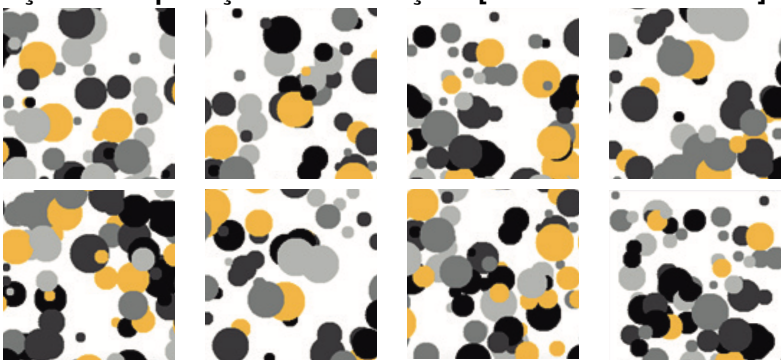
3. Metodologia do trabalho

Para avaliar o processo de reconhecimento foram utilizados trabalhos científicos coletados e discutidos em [Zamberlan et al. 2019], pois foram elencados experimentos com seus respectivos parâmetros e se houve ou não aglomeração (instabilidade). Na Tabela 2, é possível visualizar algumas amostras de imagens para o treinamento supervisionado da RNA do módulo proposto. Destaca-se que as imagens foram geradas pela ferramenta MASPN [Zamberlan et al. 2019]. No trabalho de [Zamberlan et al. 2016b], decidiu-se usar nas simulações, partículas com cores, justamente para dar a impressão de profundidade, aglomeração e facilitar a visualização dos leitores.

No módulo RNA desenvolvido para identificar aglomeração em Sistemas Nanoparticulados, há necessidade de se ter imagens que compõem o conjunto de amostras de imagens com aglomeração e sem aglomeração. Além disso, é necessário um conjunto de amostras de imagens que representam o que é de fato uma partícula.

Neste trabalho, o conjunto de amostra do que é uma partícula pode ser imagens de tons de cinza de círculos com diferentes tamanhos. Essa decisão foi tomada, pois o módulo precisa trabalhar e processar os falsos-positivos e falsos-negativos. Por exemplo, um falso-positivo seria a identificação de uma partícula na imagem sem que fosse de fato uma partícula. E o contrário, o falso-negativo seria uma partícula não identificada pelo módulo.

Tabela 2. Imagens de amostras particuladas de um sistema coloidal usadas para o treinamento da RNA. As imagens contêm partículas em algumas para facilitar a visualização e interpretação das simulações [Zamberlan et al. 2019].



3.1. Tecnologias

Neste trabalho, pesquisou-se e experimentou-se algumas tecnologias para processar imagens e reconhecer padrões. Como por exemplo: Linguagem Python, versão 3.7; OpenCV para o processamento de reconhecimento de imagens.

Destaca-se que na biblioteca OpenCV há funções destinadas à preparação de imagens (*opencv_createsamples*), bem como treinamento do reconhecedor de padrões ou classificador (*opencv_traincascade*). Com o *createsamples* foi pego uma imagem positiva e replicado em diversas imagens negativas (sem conter imagens de círculos/partículas). Dessa forma, foram criadas diversas imagens positivas.

Foi pesquisado sobre o Portal MASPN, a ferramenta de simulação MASPN e sobre o módulo que foi implementado. O entendimento de funcionamento da ferramenta e do Portal demandou tempo significativo, por se tratarem de ambientes com funcionalidades e peculiaridades. Outro foco de estudo, foram as necessidades a serem supridas na Rede Neural, como atributos de entrada e saída, bem como os pesos dos atributos (suas importâncias para a RNA). Por fim, foram modelados aspectos estruturais e funcionais do módulo inserido na ferramenta *Desktop*.

Na Figura 5, é possível visualizar tanto as funcionalidades do portal MASPN, quanto as funcionalidades do simulador no *Desktop*. Destaca-se na figura a funcionalidade *Patterns Recognize - Image* como módulo de trabalho desta pesquisa. A ideia principal dessa funcionalidade é justamente reconhecer padrões presentes nas imagens geradas da simulação da ferramenta MASPN (como o exemplo de simulação da Figura 2).

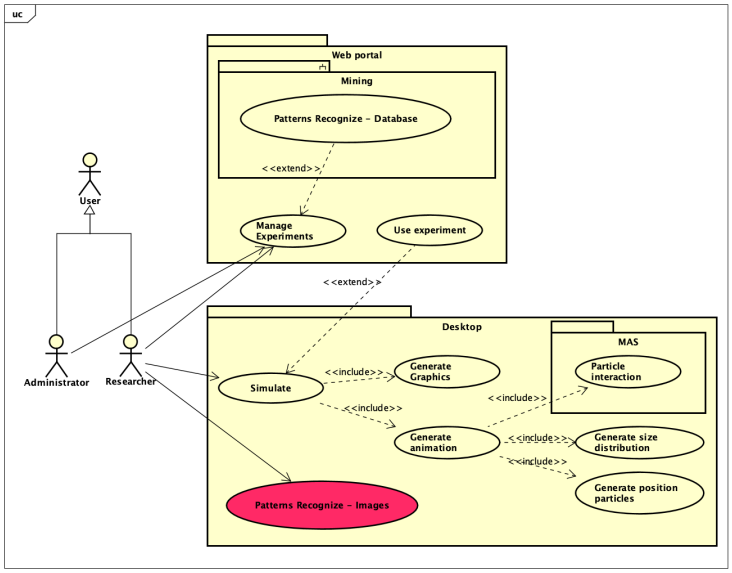


Figura 5. Casos de uso com a funcionalidade de reconhecimento de padrão do módulo (Patterns Recognize - Images)

4. Resultados

A Figura 6 mostra o menu com o módulo de reconhecimento. Como a ferramenta MASPN foi construída em inglês, o módulo também terá diálogos nessa língua. Ressalta-se que o treinamento é realizado fora da ferramenta MASPN, em um momento bem anterior ao seu uso, enquanto que o processo de reconhecimento é chamado na ferramenta, via um processo Java de execução de código externo.

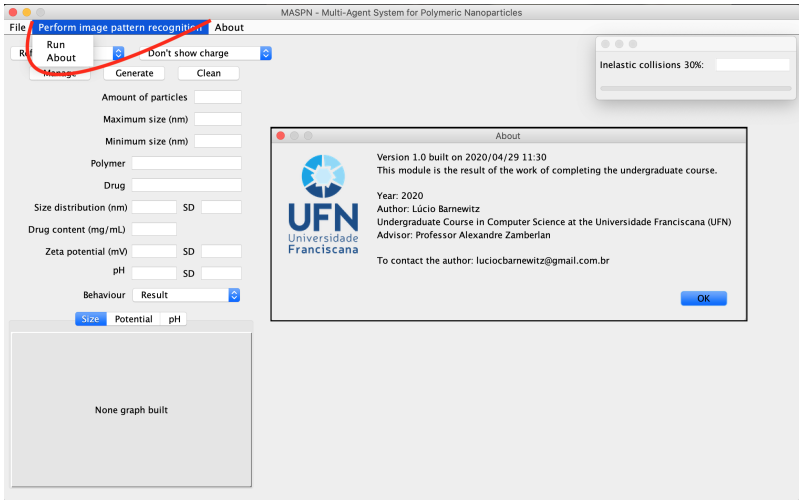


Figura 6. MASPN com menu para reconhecimento de padrões nas imagens geradas.

4.1. Processo de treinamento do classificador

Neste trabalho foi utilizado o método *traincascade* (dentro da biblioteca *OpenCV*) que implementa uma Rede Neural Convolucional com treinamento supervisionado (amostras) e multicamada.

A Figura 7 ilustra o fluxo do processo de preparação do classificador, com imagens positivas e negativas. É o processo de treinamento do classificador para reconhecer partículas em imagens de sistemas particulados.

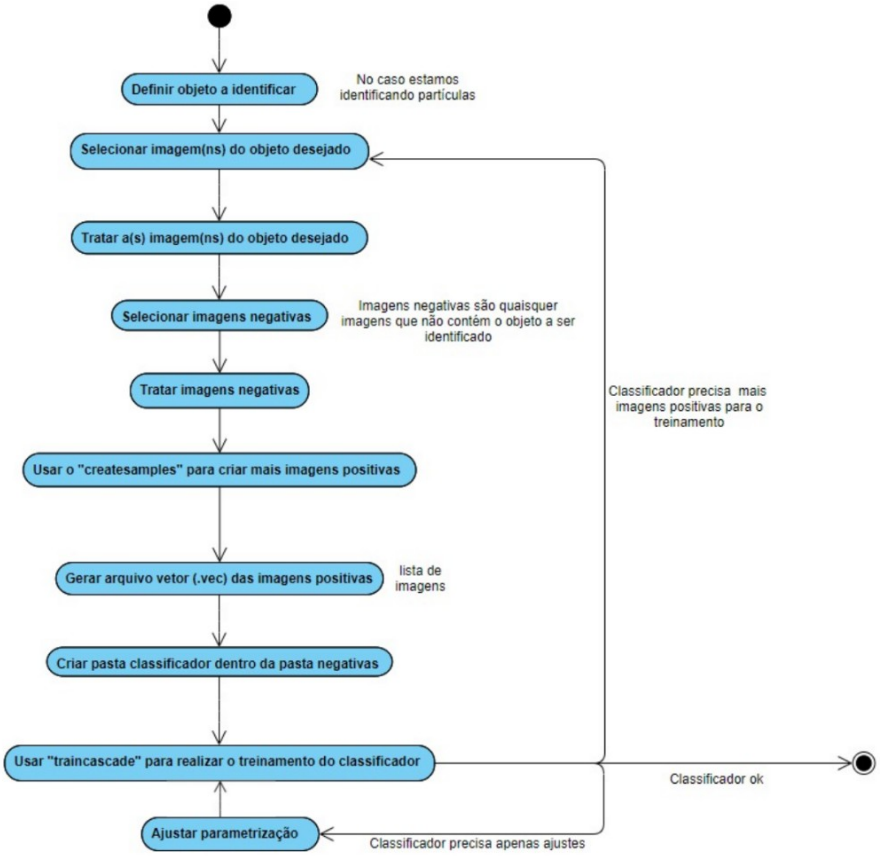


Figura 7. Diagrama de atividades com o fluxo para treinar o classificador.

Para iniciar o processo de treinamento, deve-se criar uma pasta contendo imagens negativas (imagens que não contém em hipótese alguma partículas, círculos ou bolas). Dentro dessa pasta, deve-se conter uma listagem com o nome de todas imagens, para isso foi usado *script criar_lista.bat* que gera o arquivo *bg.txt* (Figura 8) contendo todos os nomes das imagens negativas. Fora da pasta das negativas, há as imagens positivas (imagens com exemplos precisos de partículas), que foram selecionadas manualmente (Figura 9).

Figura 8. Exemplo de como foi feito o arquivo bg. Essa são imagens consideradas negativas baixadas do site especializado em imagens negativas (www.image-net.org).

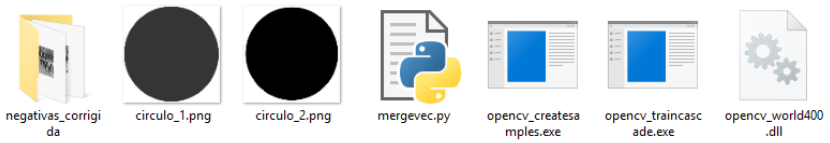


Figura 9. Demostração de pasta pronta para inicio do reconhecimento.

Em seguida, foi utilizado *opencv_createsamples* para multiplicar a quantidade de imagens positivas. Próximo passo, o *opencv_createsamples* foi executado novamente para gerar os arquivos vetoriais referente a cada pasta de imagens positivas.

É preciso ter acesso ao console do sistema (no sistema operacional Windows, *command*, em Linux ou Unix, *terminal*) e realizar os processos. O primeiro deles é sobrepor as imagens negativas com as positivas que estão dentro da pasta, gerando uma nova pasta que conterá imagens mescladas para cada imagem positiva. Na Figura 11 há um exemplo e na Figura 10 pode-se ver como fica a pasta gerada. O arquivo *bg.txt* é uma listagem dos nomes das imagens negativas. A pasta *positivas1* conterá as 300 imagens geradas pela linha de comando e o arquivo *lst*. Ou seja, é uma lista mostrando o tamanho e posição onde está a imagem positiva sobreposta na respectiva negativa.

Figura 10. Exemplo de pasta com figuras positivas.

Os parâmetros de angulação são usados para realizar rotações na imagem positiva antes de sobrepô-la, a fim de gerar mais variabilidade. O parâmetro *bgcolor* é usado para deixar a imagem em tons de cinza e o *bgtresh* usado para fazer o fundo da positiva ficar transparente (cuidado ao usar esse parâmetro para não deixar o objeto transparente também). Isso deve ser repetido para cada imagem positiva existente.

```
F:\back\facul\tfg\codigos\TFG\curso_opencv\circulo3 (master -> origin)
λ opencv_createSmaples -img circulo01.jpg -bg negativas/bg.txt -info positivas1/positivas1.lst -maxxangle 0.5 -maxyagl
e -maxzangle 0.5 -w 48 -h 48 -num 300 -bgcolor 255 -bgtresh 8
```

Figura 11. Uso do opencv_createsamples para realizar a criação de novas imagens positivas.

Após cada pasta de imagens positivas serem criadas, deve-se criar um arquivo de vetor dinâmico para elas. Para isso se usa novamente o *opencv_createsamples* como pode-se ver na Figura 12 e esse passo se faz para cada uma das pastas de imagens positivas que foi gerada.

```
F:\back\facul\tfg\codigos\TFG\curso_opencv\circulo3 (master -> origin)
λ opencv_createSamples -info positivas1/positivas1.lst -num 2000 -w 20 -h 20 -vec vetor1.vec
```

Figura 12. Uso do opencv_createsamples para realizar a criação de novas imagens positivas.

Depois disso, há inúmeros vetores na pasta principal. Em seguida, deve-se colocar todos os vetores dentro de uma pasta *vec* e executar o programa *mergevec.py* (programa construído pela comunidade para ser utilizado com OpenCV - Figura 13).

```
F:\back\facul\tfg\codigos\TFG\curso_opencv\circulo3 (master -> origin)
λ py mergevec.py -v vec/ -o vetor_final.vec
```

Figura 13. Demonstração de como juntar todos os vetores em um só arquivo.

Agora tem-se o arquivo de vetor final, assim deve-se copiar esse arquivo, a *dll opencv_world400*, bem como o arquivo *opencv_traincascade* para dentro da pasta das imagens negativas. Também deve ser criada outra pasta chamada *classificador* dentro da pasta das imagens negativas, pois é nesta pasta que o classificador será criado (Figura 14).

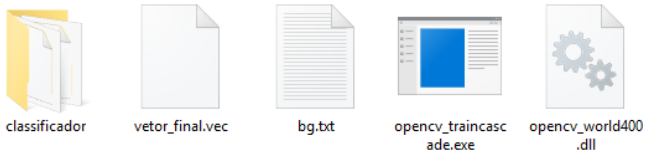


Figura 14. Pasta das imagens negativas pronta para o treinamento.

Para realizar o treinamento, novamente deve-se acessar o terminal console (Figura 15), que interage com o vetor criado anteriormente. O numero máximo de imagens positivas e de negativas é passado como parâmetro, bem como o tamanho das imagens, o número máximo de gerações de treinamento (sendo que ele pode parar antes, caso o *traincascade* note que não consegue aprender mais do que o estado atual). As quantidades de memória utilizadas para o sistema durante o aprendizado são definidos também via parâmetro. Assim, o arquivo *cascade* (arquivo cascade_particulas.xml) é gerado na pasta *classificador* e é ele que é usado no código para realizar a análise na amostra desejada.

O objetivo do treinamento (aprendizado da máquina) é chegar o mais próximo possível do valor 1 no *hit rate* (taxa de acerto) e o mais próximo possível do valor 0 no *false alarme* (alarme falso), como mostra a Figura 16.

4.2. Aplicando o classificador

Na Figura 17, pode-se ver parte da codificação que contempla: i) a leitura de imagem(ns), ii) classificador(es), iii) formação de retângulos em volta das imagens identificadas na amostra e iv) contagem desses retângulos.

```
F:\back\facul\tfg\codigos\TFG\curso_opencv\circulo3\negativas (master -> origin)
λ opencv_traincascade -data classificador -vec vetor_final.vec -bg bg.txt -numPos 1800 -numNeg 1200 -numStages 10 -w 2
0 -h 20 -precalcBufSize -1024 -precalcIdxBufSize 1024
```

Figura 15. Exemplo do *traincascade* em funcionamento.

```
===== TRAINING 1-stage =====
<BEGIN
POS count : consumed   1800 : 1807
NEG count : acceptanceRatio    1200 : 0.427503
Precalculation time: 11.395
+----+---------+---------+
|  N |    HR   |    FA   |
+----+---------+---------+
|   1|        1|        1|
+----+---------+---------+
|   2| 0.998889| 0.709167|
+----+---------+---------+
|   3| 0.998889| 0.709167|
+----+---------+---------+
```

Figura 16. Exemplo de taxa de acerto e alarme falso.

```
1  import cv2
2
3  imagem1 = cv2.imread('fig_amostras/fig1_amostra.png')
4
5  classificador1 = cv2.CascadeClassifier('cascade_particulas.xml')
6
7  imagemcinza1 = cv2.cvtColor(imagem1, cv2.COLOR_BGR2GRAY)
8
9  deteccoes1 = classificador1.detectMultiScale(imagemcinza1, scaleFactor=1.33, minNeighbors= 10)
10
11 count = 0
12
13 for (x, y, l, a) in deteccoes1:
14     cv2.rectangle(imagem1, (x, y), (x + l, y + a), (0,255,0), 2)
15     count += 1
16
17 print ('Quantidade de particulas: ', count)
18
19 cv2.imshow('Detector de particulas 1', imagem1)
20
21 cv2.waitKey(0)
22 cv2.destroyAllWindows()
```

Figura 17. Código Python para a análise da amostra desejada.

Em relação ao código Figura 17, destaca-se a importação da biblioteca *opencv*. A função *imread* (linha 3) passa o endereço da amostra que deve ser analisada e a amostra fica gravada na variável *imagem1*. Na função *CascadeClassifier* (linha 5) é passado o endereço onde está o classificador que é usado para análise da amostra salva em *imagem1* (variável *classificador1*).

A função *cvtColor*, neste caso, tem o intuito de colocar a imagem em tons de cinza, que é feito pelo *COLOR_BGR2GRAY* e essa nova imagem fica representada pela variável *imagemcinza1* (linha 7). No *detectMultiScale* (linha 9) são passados parâmetros para identificação junto com o arquivo *cascade_particulas.xml*. Na função *detectMultiScale* é passada a imagem cinza gerada, o *scalefactor* que é o parâmetro que especifica a escala de tamanho em que a imagem é reduzida. Já o *minNeighbors* especifica quantos vizinhos positivos um certo bloco precisa para ser considerado positivo também, assim minimizando o número de falsos positivos (falsos alarmes) que serão encontrados.

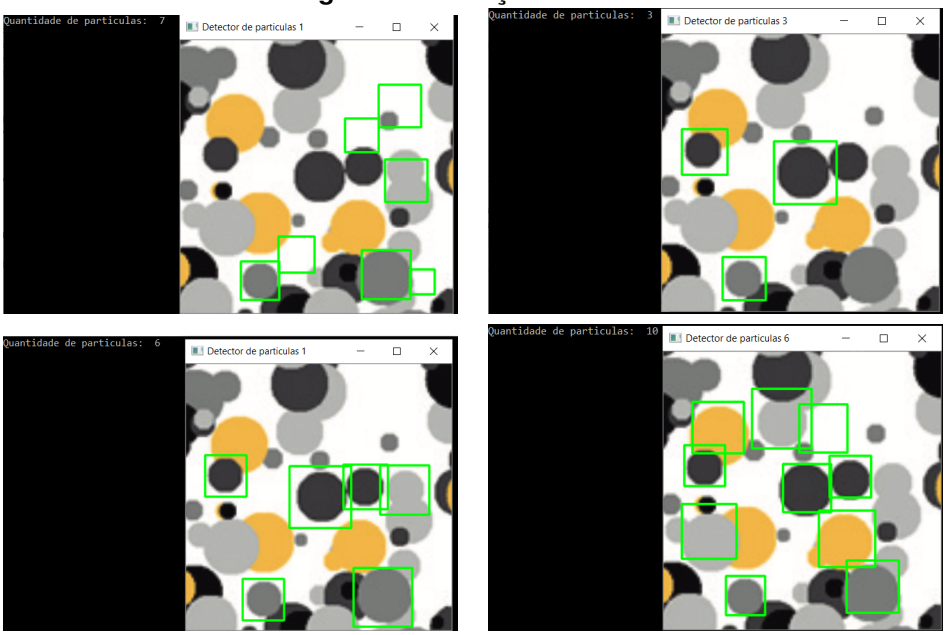
A rotina de repetição tem o trabalho de desenhar retângulos em torno das figuras dos objetos encontrados e fazer a contagem deles (exibido na linha 17).

A função *imshow* (linha 19) mostra a imagem aberta no inicio do código com os retângulos desenhados no passo anterior.

O *waitkey* é usado para manter essa nova imagem aberta, esperando um próximo clique do usuário para o *destroyAllWindows* fechar a interface gráfica aberta.

As figuras da Tabela 3 mostram a evolução dos classificadores implementados, de uma classificação com muitos falsos positivos ou poucas partículas reconhecidas até uma melhor. Pode-se notar que o melhor classificador gerado não se comportou como desejado. Durante vários testes realizados, foram detectados inconsistências na classificação (como mostram as figuras).

Tabela 3. Imagens da evolução dos classificadores.



4.3. Análise dos resultados

Como é possível perceber na Tabela 3, o protótipo não conseguiu detectar todas as partículas nas imagens. Houveram falsos positivos e falsos negativos acima do desejado. Dessa forma, o processo de reconhecer se houve aglomeração ficou comprometido. Acredita-se que o que impossibilitou a detecção foram:

- conjunto de imagens negativas não é adequado;
- o acerto nos valores dos parâmetros;
- falta de qualidade nas imagens positivas (uso de arquivos .png);
- quantia não suficiente de amostras (2800 amostras[3]).

Entretanto, registra-se que o processo de treinamento do classificador esteja correto, só o conjunto de imagens (tanto negativas quanto positivas) pode estar gerando um falso aprendizado para o classificador.

[3]Imagens baixadas de site especializado em gerar imagens de teste.

Ressalta-se que o classificador identifica a quantidade de partículas, que ajudaria a definir o quão o sistema está aglomerado. A ferramenta MASPN fornece o total de partículas a serem simuladas (Figura 2) e o reconhecedor também conta o número de partículas reconhecidas. Dessa forma, seria possível realizar uma subtração de total de partículas geradas para simulação pelo MASPN pela quantidade de partículas identificadas pelo reconhecedor. Quanto mais a resposta tender a zero, menos aglomerado está a imagem. Quanto mais se afastar do zero, significa que há partículas uma sobre outras, ou seja, mais aglomeração (como mostram as Figuras da Tabela 3 que contém um conjunto de partículas aglomeradas).

5. Conclusões

Ao final deste trabalho, destaca-se que o texto buscou apresentar e discutir aspectos do Portal MASPN e da ferramenta de simulação MASPN, para uma melhor compreensão de todo o processo, inclusive para a inserção adequada do módulo de reconhecimento modelado e implementado. Além disso, o texto mostrou informações sobre RNA (como tipos, formas de treinamento, aplicações) e um passo-a-passo para treinar o classificador e usá-lo no reconhecimento de imagens.

Por se tratar de sistemas já implementados (MASPN e portal) e avaliados, o processo de compreensão de seus aspectos estruturais e funcionais consumiu tempo significativo. Entretanto, foi possível mapear onde e quando o módulo de reconhecimento teria função.

Os algoritmos de treinamento e da aplicação do classificador foram baseados em sugestões dos trabalhos relacionados, bem como as tecnologias utilizadas.

Finalmente, tem-se consciência que os resultados não foram adequados, mas deixa-se como trabalhos futuros essas adequações como melhorar o conjunto de amostras negativas, ajustar os parâmetros do classificador, trabalhar com imagens de melhor qualidade.

Referências

Boaglio, F. (2016). Scrum solo. http://www.boaglio.com/index.php/2007/12/27/scrum-solo/.

Brito Bisneto, C. R. d. (2011). Reconhecimento de objetos utilizando redes neurais arti ciais e geometria fractal.

Cardon, A., Müller, D. N., and Navaux, P. (1994). Introdução às redes neurais artificiais. *Instituto de Informática. Universidade Federal do Rio Grande do Sul. Porto Alegre*.

de Andrade Kovaleski, P. (2018). *IMPLEMENTAÇÃO DE REDES NEURAIS PROFUNDAS PARA RECONHECIMENTO DE AÇÕES EM VÍDEO*. PhD thesis, Universidade Federal do Rio de Janeiro.

Erpen, L. R. C. (2004). Reconhecimento de padrões em imagens por descritores de forma.

Lohmann, E. S. (2016). Técnica de redes neurais artificiais aplicada ao reconhecimento de imagens para a interação com um jogo computacional.

Martins, M. P., Guimarães, L. N. F., and Fonseca, L. M. G. (2002). Classificador de texturas por redes neurais. In *Anais do II Congresso Brasileiro de Computação, Itajaí-SC*.

Norvig, P. and Russell, S. (2014). *Inteligência Artificial: Tradução da 3a Edição*, volume 1. Elsevier Brasil.

Pereira, G. G., Brum, J. V. R., Vieira, S., Fagan, S., Laporta, L., and Zamberlan, A. (2018). Portal web para simulação no ambiente MASPN. In *XXII Simpósio de Ensino, Pesquisa e Extensão da UFN*, pages 1–10, Santa Maria, Brasil.

Silva, R. R. (2005). Reconhecimento de imagens digitais utilizando redes neurais artificiais. *Monografia de graduação em Ciência da Computação–Universidade Federal de Lavras. Lavras*.

Zamberlan, A., Dalcin, A. J., Kurtz, G., Bordini, R., Raffin, R., and Fagan, S. (2016a). Simulation environment for polymeric nanoparticle: experiment database. *Disciplinarum Scientia*, 17(3):429–446.

Zamberlan, A. d. O., Kurtz, G. C., Gomes, T. L., Bordini, R. H., and Fagan, S. B. (2019). A simulation environment for polymeric nanoparticles based on multi-agent systems. *Journal of molecular modeling*, 25(1):5.

Zamberlan, A. O., Dalcin, A. J. F., Kurtz, G. C., Bordini, R. H., Raffin, R. P., and Fagan, S. B. (2016b). Simulation environment for polymeric nanoparticle: experiment database. *Disciplinarum Scientia— Naturais e Tecnológicas*, 17(3):429–446.