isec
Engenharia

LICENCIATURA EM INFORMÁTICA E SISTEMAS

INTELIGÊNCIA COMPUTACIONAL

Fase I - Modelo com Redes Neuronais

Autores

Inês Raquel Rodrigues Ferreira - 2023111442

Tiago Miguel Leitão Manata - 2023147352

INSTITUTO POLITÉCNICO
DE COIMBRA

INSTITUTO SUPERIOR
DE ENGENHARIA
DE COIMBRA

Coimbra, outubro de 2025

ÍNDICE

1. DESCRIÇÃO DO CASO DE ESTUDO E OBJETIVOS DO PROBLEMA 3
2. DATASET E TRATAMENTO DE IMAGEM 4
3. DESCRIÇÃO DO MODELO E IMPLEMENTAÇÃO DOS ALGORITMOS 5
3.1 Multilayer Perceptron (MLP) 5
3.2. Modelo CNN (Convolutional Neural Network) 7
4. ANÁLISE DE RESULTADOS 9
4.1 Modelo MLP 9
4.2 Modelo CNN 10
5. CONCLUSÕES 12
6. REFRÊNCIAS 13

1. DESCRIÇÃO DO CASO DE ESTUDO E OBJETIVOS DO PROBLEMA

O presente trabalho tem como objetivo principal aplicar um algoritmo capaz de identificar automaticamente instrumentos cirúrgicos. Esta ideia surge a partir da realidade observada em contexto clínico, em especial na sala do bloco operatório, onde o enfermeiro instrumentista necessita de atuar com rapidez e precisão. Num ambiente cirúrgico, qualquer engano pode traduzir-se em consequências graves, pelo que tudo o que contribua para aumentar a segurança do utente e melhorar o desempenho da equipa é crucial.

Enquanto profissionais de enfermagem, com experiência clínica, reconhecemos que a evolução da medicina deve acompanhar o avanço das novas tecnologias. Só assim será possível reduzir o risco, minimizar o erro e tornar os cuidados de saúde mais seguros. A inteligência artificial (IA) surge como uma aliada promissora neste processo, especialmente em contextos críticos como o bloco operatório.

Assim, este trabalho centra-se em desenvolver um modelo de inteligência artificial capaz de reconhecer de forma autónoma diferentes instrumentos cirúrgicos, com o menor erro possível. No futuro, um sistema deste tipo poderá servir de apoio ao enfermeiro instrumentista e à equipa cirúrgica, auxiliando na identificação rápida e precisa dos instrumentos, prevenindo trocas entre profissionais em cirurgias longas, o que poderá reduzir o tempo de resposta e melhorar a dinâmica global da intervenção cirúrgica. Adicionalmente, a implementação de um sistema automático poderá traduzir-se na diminuição do risco de contaminação do campo operatório e contribuir para a manutenção das condições assépticas.

Para o desenvolvimento do projeto foram selecionados cinco instrumentos cirúrgicos de uso frequente, nomeadamente o bisturi, a pinça, a pinça hemostática, a tesoura de Mayo reta e a tesoura de Mayo curva. Estes instrumentos foram escolhidos por serem fundamentais em praticamente todos os tipos de cirurgia e por exigirem uma identificação célere e exata.

O modelo foi treinado com imagens destes instrumentos, de forma a aprender a reconhecê-los com o maior grau de precisão possível. Assim, pretende-se criar um sistema capaz de identificar automaticamente o instrumento solicitado pelo cirurgião, contribuindo para aumentar a eficiência do trabalho da equipa cirúrgica e melhorar a segurança e qualidade dos cuidados prestados.

2. DATASET E TRATAMENTO DE IMAGEM

Tendo como meta obter aproximadamente dez mil imagens (2000 para cada classe), de forma a garantir um conjunto de dados suficientemente robusto e equilibrado, procedeu-se inicialmente à captação manual de imagens reais dos instrumentos cirúrgicos selecionados. Para cada tipo de instrumento foram registadas cerca de vinte fotografias, captadas num ambiente controlado, mas com variação intencional de ângulos, distâncias e condições de iluminação, de modo a evitar que o modelo ficasse "viciado" num único tipo de cenário.

Após esta fase, procedeu-se à complementação do conjunto de dados com imagens provenientes de bases de dados públicas, nomeadamente o Kaggle e o Mendeley Data, de forma a aumentar a representatividade e diversidade visual do dataset.

Por fim, recorreu-se à plataforma Roboflow para a realização do processo de aumento de dados (data augmentation), aplicando transformações como ajustes de brilho e contraste, rotações e pequenas distorções geométricas. Esta etapa permitiu uniformizar o conjunto de dados e atingir o total de dez mil imagens requisitado, garantindo a variabilidade necessária para o treino eficaz do modelo de redes neuronais.

3. DESCRIÇÃO DO MODELO E IMPLEMENTAÇÃO DOS ALGORITMOS

3.1 Multilayer Perceptron (MLP)

Nesta fase do projeto desenvolvemos um classificador de imagens baseado numa rede neuronal do tipo Multilayer Perceptron (MLP). Escolhemos este modelo por ter uma arquitetura simples e eficaz na resolução de problemas de classificação. A implementação foi realizada em Python, recorrendo às bibliotecas *NumPy*, *Pillow* e *Scikit-learn*.

O primeiro passo consistiu em garantir a reprodutibilidade do processo de treino através da fixação de uma *seed* aleatória.

```
import numpy as np
np.random.seed(20)
```

Isto permite que a divisão entre conjuntos de treino e teste, bem como os pesos iniciais do modelo, se mantenham constantes durante as execuções, o que assegura a comparabilidade dos resultados.

Foi também realizado um pré-processamento das imagens das subpastas do dataset (cada uma correspondente a uma classe) de forma a carregar as imagens, convertê-las para tons de cinzento, redimensioná-las para o tamanho 32x32.

Depois de carregar os dados, as classes são codificadas numericamente através do LabelEncoder, convertendo os nomes das pastas em valores inteiros.

```
X, y_text = load_images(DATA_DIR)
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y_text)
```

Esta codificação é necessária porque os algoritmos de aprendizagem automática não operam diretamente sobre rótulos em formato textual.

De seguida efetuamos a divisão do *dataset* em dois subconjuntos: **80% para treino** e **20% para teste através do** *train_test_split*, assegurando que a proporção de exemplos por classe é mantida em ambos os conjuntos.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=10,
stratify=y)
```

Após a preparação dos dados, definimos o modelo MLP.

```
from sklearn.neural_network import MLPClassifier
```

```
model = MLPClassifier(
    solver="lbfgs",
    activation="relu",
    hidden_layer_sizes=(400,400),
    max_iter=1000,
    validation_fraction=0.1,
    random_state=10
)
model.fit(X_train_scaled, y_train)
```

Após treinar o modelo procedemos à avaliação do desempenho e para isso utilizamos várias métricas: *accuracy*, *recall*, *f1-score*, *especificidade* e *AUC*.

```
from sklearn.metrics import (
    accuracy_score, recall_score, f1_score,
    confusion_matrix, roc_auc_score
)
y_pred = model.predict(X_test_scaled)
y_prob = model.predict_proba(X_test_scaled)
acc = accuracy_score(y_test, y_pred)
recall = recall_score(y_test, y_pred, average="macro")
f1 = f1_score(y_test, y_pred, average="macro")
```

Para a especificidade média foi calculada manualmente a partir da matriz de confusão, de modo a avaliar a capacidade do modelo em evitar falsos positivos, e a métrica AUC (Area Under the Curve) foi utilizada em regime *one-vs-rest*, medindo o desempenho discriminativo geral do classificador multiclasse.

```
cm = confusion_matrix(y_test, y_pred)
specificity = []
for i in range(len(cm)):
    tn_i = cm.sum() - (cm[i, :].sum() + cm[:, i].sum() - cm[i, i])
    fp_i = cm[:, i].sum() - cm[i, i]
    specificity.append(tn_i / (tn_i + fp_i))
specificity_mean = np.mean(specificity)
auc = roc_auc_score(y_test, y_prob, multi_class="ovr")
```

Por ultimo, o desempenho do modelo foi representado graficamente através de uma matriz de confusão, que permite visualizar as classes que foram classificadas de forma correta e incorreta.

```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Oranges',
            xticklabels=le.classes_,
            yticklabels=le.classes_)
plt.xlabel('Previstas')
plt.ylabel('Reais')
plt.title('Matriz de Confusão')
plt.tight_layout()
plt.show()
```

3.2. Modelo CNN (Convolutional Neural Network)

De forma a compara o MLP com outro tipo de rede, desenvolvemos também um classificador de imagens com uma rede neuronal do tipo Convolutional Neural Network (CNN). Estas redes utilizam de filtros convolucionais que extraem padrões locais (bordos, texturas, formas), e são adequadas para visão por computador. A implementação foi realizada em Python, recorrendo a NumPy, Pillow e TensorFlow/Keras.

Tal como na MLP, começámos por garantir reprodutibilidade com a fixação da seed e redimensionamento para 32×32. Visto que o dataset contem algumas imagens a preto e branco, cada imagem foi convertida para o formato RGB (3 canais de cor).

Após pré processar as imagens foi feito, tal como no MLP, uma divisão do dataset em dois subconjuntos, 80% para treino e 20% para teste.

Os rótulos das classes foram convertidos para formato one-hot encondig, que transforma cada classe num vetor binário.

```
from tensorflow.keras.utils import to_categorical
yTrain_binario = to_categorical(y_train, 5)
yTest_binario  = to_categorical(y_test, 5)
```

O modelo CNN foi criado através da API Sequential do Keras. Optámos por uma arquitetura composta por uma camada convolucional com 32 filtros e ativação ReLu, que é responsável pela extração dos padrões locais. Depois adicionamos uma camada de achatamento (Flatten) que vai ser responsável por converter os mapas de ativação 2D num vetor 1D. Adicionamos também uma camada com 100 neurónios e ativação *ReLU* que vai realizar a combinação não linear das características extraídas.

Por fim, adicionamos uma camada de saída com 5 neurónios(um para cada classe) e ativação *softmax* para gerar as probabilidades de pertença.

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

model = Sequential([
    Conv2D(16, (3,3), activation='relu', input_shape=(32,32,3)),
    Flatten(),
    Dense(100, activation='relu'),
    Dense(5, activation='softmax')
])
```

Quanto ao treino do modelo, este decorreu durante 10 épocas com um batch size de 16 imagens e um validation split de 10%.

```
 print("**** Treinar o modelo ****")
modelo = model.fit(
    X_train, yTrain_binario,
    epochs=10,
    batch_size=16,
    validation_split=0.1,
    verbose=2
)
```

Após o treino, tal como na MLP, foram calculadas métricas de avaliação tal como a ACC, recall, F1-Score, AUC e especificidade média.

4. ANÁLISE DE RESULTADOS

4.1 Modelo MLP

Após o treino da rede MLP, realizamos uma análise comparativa entre as diferentes configurações da rede, com o intuito de avaliar o impacto da alteração de parâmetros no desempenho da rede.

Inicialmente testamos dois algoritmos de otimização, o lbfgs e o adam. Para obtermos resultados nas mesas condições, mantivemos constante o número de neurónios (100), o número de iterações (100) e a função de ativação *ReLU*. Os resultados obtidos foram os seguintes:

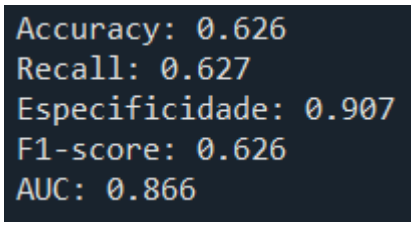

Figura 1 *– Resultados MLP com função lbfgs e camada de 100 neurónios*

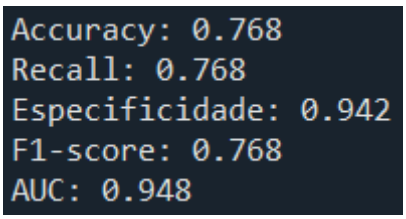

Figura 2 *– Resultados MLP com função adam e camada de 100 neurónios*

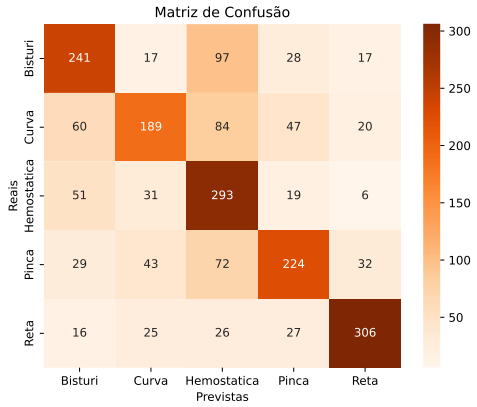

Figura 3 *– Matriz Confusão MLP com função lbfgs e camada de 100 neurónios*

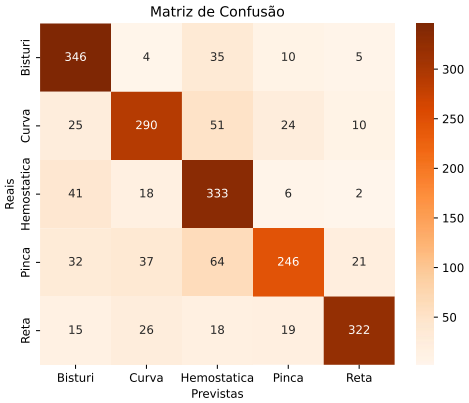

Figura 4 *– Matriz Confusão MLP com função adam e camada de 100 neurónios*

Tendo obtido melhores resultados com o solver adam, treinamos novamente o modelo com este algoritmo, mas desta vez com 2 camadas ocultas, contendo 100 e 200 neurónios, respetivamente. Aumentamos também as iterações para 500 e verificamos o seguinte:

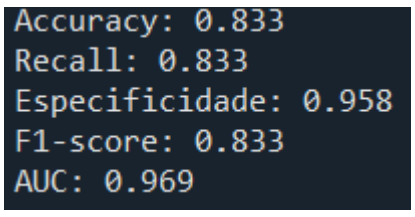

Figura 5 *– Resultados MLP com função adam e com 2 camadas de 100,200 neurónios e 500 iterações*

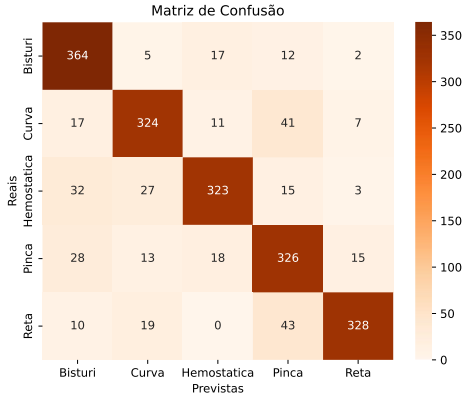

Figura 6 *– Matriz Confusão MLP com função adam e com 2 camadas de 100,200 neurónios e 500 iterações*

Os resultados obtidos constam com uma accuracy, recall e F1-score iguais (0.833), o que se poderá justificar devido ao equilíbrio entre acertos e erros em todas as classes. Este comportamento ocorre porque o conjunto de dados é balanceado e o modelo apresenta valores semelhantes de precisão e sensibilidade para cada classe. Assim, como não há grande desequilíbrio entre o número de falsos positivos e falsos negativos, as 3 métricas acabam por convergir para o mesmo valor.
No que diz respeito à especificidade, podemos verificar que esta é elevada (0.958), demonstrando que o modelo raramente confunde classes diferentes, enquanto o valor da AUC de 0.969 confirma uma boa capacidade de distinção entre categorias. Na última configuração o modelo foi capaz de classificar corretamente cerca de 83% das 10000 imagens, apresentando uma boa capacidade de generalização.

4.2 Modelo CNN

A rede CNN foi treinada com uma camada Conv2D, que no nosso caso, foi responsável por aprender 16 padrões visuais diferentes das imagens, utilizando a função de ativação ReLu. Em seguida, aplicou-se a camada Flatten(), que converte as características 2D extraídas com o Conv2D, num vetor linear, permitindo a sua ligação às camadas densas.

A camada de entrada (Dense) utilizada foi composta por 100 neurónios, também ativada com a função ReLu e a camada de saída foi composta por 5 neurónios, 1 por cada classe e ativada pela função softmax.

Após o treino, a rede alcançou resultados demostrados na imagem abaixo.

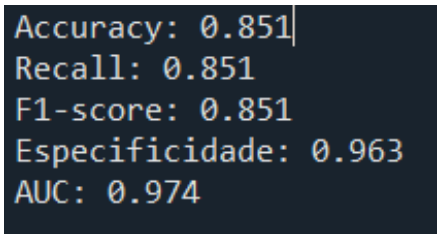

Figura 7 *– Resultados CNN*

Estes valores demostram que o modelo consegue classificar corretamente a grande maioria das imagens, distinguindo bem entre as diferentes classes. O valor elevado da AUC demonstra uma excelente capacidade da rede em separar as classes corretamente.

Uma vez que o *recall* mede a capacidade do modelo em identificar corretamente as imagens que realmente pertencem a uma determinada classe, ou seja, representa a proporção de acertos entre todas as amostras reais dessa categoria., decidimos calcular esta métrica em cada classe individualmente, obtendo os seguintes resultados:

```
**** Métricas por classe ****
Recall(Bisturi)=0.853
Recall(Curva)=0.812
Recall(Hemostatica)=0.887
Recall(Pinca)=0.792
Recall(Reta)=0.907
```

Figura 8 *– Resultados do recall CNN por classe*

Com estes dados percebemos que o modelo obteve melhor desempenho nas classes Hemostática (recall = 0.887) e Reta (recall = 0.907), enquanto a classe Pinça apresentou o valor mais baixo (recall = 0.792). Estes resultados indicam que o modelo reconhece bem a maioria das classes, mas demonstra alguma dificuldade em distinguir instrumentos com características visuais semelhantes, como acontece entre as classes Curva e Pinça. Esta tendência é confirmada pela matriz de confusão, onde se observa uma maior sobreposição entre estas categorias.

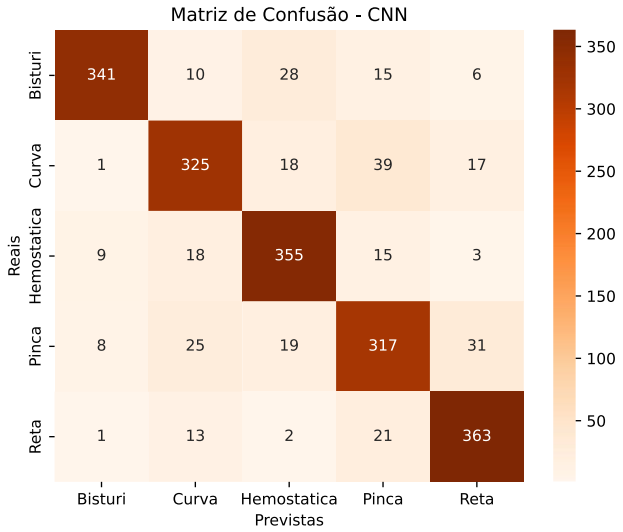

Figura 9 - Matriz de Confusão CNN

5. CONCLUSÕES

Com a elaboração do presente trabalho, foi possível perceber de que forma a IA consegue realizar uma boa classificação de classes, após a aplicação de algoritmos de treno. Apesar de no enunciado ser apenas pedido uma rede MLP ou uma CNN, como tivemos tempo e tínhamos o interesse em perceber a diferenças entre as redes, optámos por criar também uma CNN.

criar e comparar dois modelos de redes neuronais, um MLP e uma CNN, aplicados à identificação automática de instrumentos cirúrgicos.

Em ambas as redes obtivemos valores acima dos 50% de acertos, mas conseguimos perceber que a rede CNN é bastante mais profunda e adequada a imagens, uma vez que é capaz de entender padrões e olhar para a imagem como um todo, ao invés de uma rede MLP, que vê cada pixel isoladamente. Com os resultados obtidos, conseguimos concluir que apesar de o modelo MLP ter obtido resultados positvos, com uma accuracy de 0.833, o modelo CNN, mesmo tendo uma arquitetura simples e poucos parâmetros, apresentou um desempenho superior, alcançando uma accuracy de 0.851 e uma AUC de 0.974.

Através da matriz de confusão, foi possível observar que o modelo acerta na grande maioria das imagens, mas ainda apresenta alguma confusão entre classes com formas semelhantes, como é o caso das categorias Curva e Pinça. Estes pequenos erros mostram que, apesar do bom desempenho global, ainda há espaço para melhoria. A AUC elevada (0.974) demonstra que o modelo tem uma excelente capacidade de distinção entre classes, o que indica um grande potencial de evolução com pequenas otimizações na arquitetura ou no pré-processamento das imagens.

Apesar dos resultados serem promissores, a aplicação prática deste sistema em ambiente clínico ainda não é ideal, devido a pequenas confusões entre classes. Contudo, com mais dados e melhorias no modelo, este sistema tem potencial para se tornar uma ferramenta fiável de apoio clínico, aumentando a precisão e eficiência na identificação de instrumentos cirúrgicos.

6. REFRÊNCIAS

Lavado, D. (2018). *Labeled Surgical Tools and Images*. Recuperado de: https://www.kaggle.com/datasets/dilavado/labeled-surgical-tools

OpenAI. (2025). ChatGPT - *OpenAI.* Recuperado de: https://chatgpt.com

Reddy, S. (2025). Surgical Tools. Recuperado de: https://data.mendeley.com/datasets/cyghvmjrt3/1

Top Scalpel Datasets and Models(s.d.). Rcuperado de: https://universe.roboflow.com/search?q=class%3Ascalpel