Handwritten Signatures Classification Using Machine Learning

Daniel Correia, nº mec. 90480, danielvalacorreia@ua.pt

***Resumo*—A verificação de identidade é uma necessidade em variadíssimas áreas e aplicações de diversos ramos distintos. Uma das técnicas de identificação mais antigas e ainda usadas na atualidade é a assinatura manuscrita. A caligrafia é considerada a impressão digital do cérebro, porque todas são diferentes, logo muitas organizações como bancos, ministério público, serviço nacional forense e finanças usam-nas para fins de identificação de identidade. Porém as assinaturas podem ser facilmente falsificadas e existe uma necessidade de verificação das mesmas. Esta verificação pode ser feita por especialistas de deteção de falsificação, ou classificando as assinaturas genuínas e falsificadas utilizando modelos de aprendizagem automática. Neste projecto em específico, é abordada a classificação de assinaturas manuscritas genuínas e falsificadas a partir de um conjunto de imagens, em que cada imagem correponde a uma assinatura.**

***Index Terms*—Signature, Forgery, Detection, Logistic, Regression, Support, Vector, Machine, Neural, Network, Image, Classification.**

I. Introdução

A Assinatura manuscrita é utilizada por muitas organizações como bancos, ministério público, serviço nacional forense e finanças para verificação e autentificação de uma pessoa. Mesmo dois gémeos idênticos têm caligrafias e assinaturas diferentes. Isso também acontece quando a mesma pessoa faz duas assinaturas diferentes, a probabilidade dessas duas assinaturas serem idênticas é muito baixa. Várias propriedades da assinatura de uma pessoa variam entre assinaturas o que torna a deteção de assinaturas falsas uma tarefa desafiante.

Para classificar assinaturas genuínas e assinaturas falsificadas, foram criados e treinados dois modelos de aprendizagem automática diferentes, que foram avaliados e comparados de modo a obter o melhor *recall* possível e uma boa precisão das classificações.

Este tema foi escolhido, devido ao facto de ter sido exposto à resolução de diversos problemas de *Inteligência Artificial* numa série televisiva, e a classificação de assinaturas genuínas de falsificadas era um dos problemas abordados. Também é um problema atual no mundo real, pois não existem especialistas de deteção de falsificação suficientes o que torna necessário soluções baseadas em *software*.

II. Estado da Arte

Algumas das técnicas utilizadas para classificação de assinaturas offline são:

- *Template Matching* - é uma técnica no processamento de imagem digital para encontrar pequenas partes de uma imagem que correspondem a uma imagem de modelo.
- *Hidden Markov Model* - é um modelo estatístico em que o sistema modelado é assumido como um processo de Markov com parâmetros desconhecidos, e o desafio é determinar os parâmetros ocultos a partir dos parâmetros observáveis.
- *Redes Neurais Convolucionais* - [Haf17] utiliza redes neuronais convolucionais numa abordagem de duas fases. Numa primeira fase em que tenta identificar a que pessoa a assinatura pertence (não depende de quem escreveu a assinatura e extrai *features*). Na segunda fase classifica a assinatura como falsificada ou genuína (classifica dependendo de quem a escreveu). outra abordagem foi [S18] que classifica uma assinatura independentemente de quem a escreveu.

Destes três modelos, o único que é interessante para o tema em questão, *Machine Learning*, são as *Redes Neurais Convolucionais* que foi um modelo que foi abordado não exaustivamente e por isso abordado sem profundidade.

III. Dataset

A. Aquisição dos dados

As assinaturas manuscritas foram obtidas do dataset[Rai] as quais no formato .png. Este dataset contêm:

- Número Total de Assinaturas: 300 amostras
- Número de Pessoas: 30 pessoas
- Número de Assinaturas genuínas de uma pessoa: 5 amostras
- Número de Assinaturas falsificadas de uma pessoa: 5 amostras
- Número Total de Assinaturas falsificadas: 150 amostras
- Número Total de Assinaturas genuínas: 150 amostras

As 150 amostras de assinaturas genuínas estão num diretório e as 150 amostras de assinaturas falsificadas estão noutro diretório.

B. Estrutura

No âmbito do desenvolvimento deste projeto, foram comparados os resultados da aprendizam por computador de dois modelos diferentes. Estes modelos, tiveram todos como base o mesmo conjunto de dados, *dataset*[Rai] esse composto por dois subconjuntos: treino e teste. Cada amostra consiste numa imagem de tamanho variável, com 3 canais de cor (BGR).

Foi então necessário reorganizar as imagens das assinaturas por pessoa, pois é preciso que cada pessoa seja vista como uma classe, bem como cada assinatura por falsificada ou genuína. Assim, a divisão dos dados é balanciada e contêm o mesmo número de assinaturas de cada pessoa.
Foram feitas comparações dos resutados obtidos dos modelos no caso em que não foram consideradas classes por pessoas, onde a divisão dos dados de treino e teste foram feitas utilizando a função *train_test_split()* da biblioteca *Sklearn*[Bos], onde os dados foram divididos em:

- Total Dados de Treino: 210 assinaturas (70%)
- Total Dados de Teste: 90 assinaturas (30%)

e no caso de terem sido consideradas classes por cada pessoa, onde os dados foram divididos em:

- Dados de Treino: 4 assinaturas genuínas e 4 assinaturas falsificadas por pessoa
- Dados de Teste: 1 assinatura genuína e 1 assinatura falsificada por pessoa
- Total Dados de Treino: 240 assinaturas (80%)
- Total Dados de Teste: 60 assinaturas (20%)

Como previsto, os modelos obtiveram melhores resultados quando a divisão foi considerando cada pessoa como uma classe e dividir as amostras balanceadamente.

C. Tratamento dos dados

Podemos observar em baixo uma assinatura genuína e uma assinatura falsificada do dataset fornecido antes de ser processada.

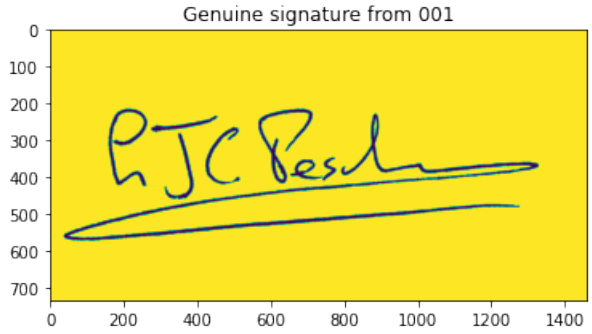

Figura 1. Imagem antes de ser processada da assinatura genuína da pessoa 001

1) BGR para RGB: As imagens fornecidas vêm no formato *BGR* que é equivalente ao formato *RGB*, mas com os channels em ordens diferentes. Como as assinaturas têm tamanho variável elas vão estar representadas numa matrix com dimensões X, Y e com profundidade de 3 (Blue, Green, Red). A transformação para RGB provavelmente não terá impacto nos modelos, mas foi realizada devido a ser o formato mais usual.

2) Redimensionamento: Como as imagens têm um tamanho variável X, Y é necessário redimensionar para um tamanho geral para que todas as imagens tenham o mesmo número de *features*, o que é necessário para ser possível desenvolver os

Figura 2. Imagem antes de ser processada da assinatura falsificada da pessoa 001 feita pela pessoa 003

modelos de *Machine Learning*. O tamanha escolhido foi 224px por 224px o que significa que cada imagem irá ter 150528 *features*. Mais tarde, em memória, irá ser feito o *resizing* das imagens (224,224,3) para (150528).
Na figura abaixo está uma imagem de uma assinatura depois de ser processada.

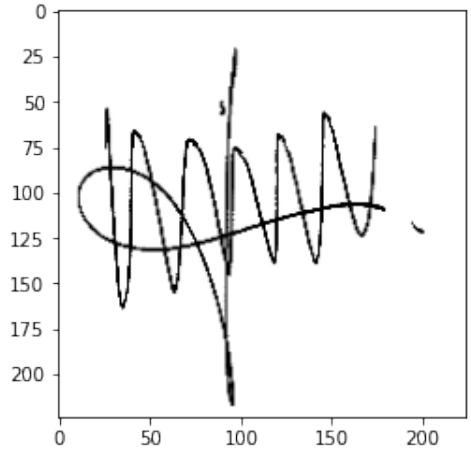

Figura 3. Exemplo de uma imagem depois de ser processada

IV. Modelos

Os modelos utilizados foram *Logistic Regression* e *Support Vector Machine*, ambos implementados recorrendo à biblioteca *open-source* sklearn[Bos], escrita em Python.

A. Logistic Regression

O modelo de regressão logística é bastante utilizado para classificar uma variável por classes, geralmente binária, ajusta-se assim ao problema em questão.

B. Support Vector Machine

O modelo de máquina de vetores de suporte é um modelo que se rege pelo aprendizado supervisionado, analisando dados e reconhecendo padrões. A sua maior utilização advém dos problemas de classificação, sendo por isso um dos modelos escolhidos para este problema.

C. Neural Network

O modelo de *Neural Network* é um modelo muito eficaz vagamente baseado no cérebro humano, utilizando diversas células dispostas em camadas. Foi experimentado, mas não houve tempo para testes e ajustes do modelo.

V. Logistic Regression

A. Leitura dos Dados

Os dados de treino e de teste são armazenados em estrutura de dados do tipo *numpy.array*. São utilizados 4 arrays de dados, o primeiro é o X_train (X, 150528) que contém as imagens utilizadas para criar o modelo, X_test (Y, 150528) que contém as imagens utilizadas para teste do modelo, y_train (Z,) que contém as labels das assinaturas utilizadas no treino se são genuínas ou falsificadas, y_test (K,) que contém as labels das assinaturas utilizadas no teste se são genuínas ou falsificadas.

B. Pré-Processamento

As imagens já se encontram processadas como foi explicado no capítulo III nas dimensões 224x224x3 (este último valor a corresponder ao sistema de cores RGB, transformado do formato original BGR).

C. Estrutura Geral

Foi utilizado como base do modelo o parâmetro imutável de 150528 features como entrada (224x224x3) e as 2 classes como saída, em que as assinaturas falsificadas têm valor 1 e as genuínas valor 0.
Já na função de *loss*, foi utilizada a função *Logistic Regression Cost Function* com o regularizador *L2 norm*;
O algoritmo utilizado foi o de *Limited-memory BFGS*
Foi procurado o melhor híper-parâmetro C, de modo a que a métrica *Recall* seja maximizada, uma vez que é mais importante não haverem assinaturas falsificadas que sejam classificadas como genuínas. Dos valores testados [0.01, 0.1, 1, 10] o que obteve melhores resultados foi 0.01, portante o híper-parâmetro C = 0.01

Da matriz de confusão resultante, foram calculados os valores de *accuracy*, *precision*, *recall* e *f1 score*.

Accuracy	0.8166
Precision	0.8170
Recall	0.8166
F1 Score	0.8166

Tabela I
Accuracy, Precision, Recall e F1 Score de Logistic Regression

Portanto, para um modelo de *Logistic Regression* foi conseguida uma precisão final de 81.7% e Recall de 81.7%.

VI. Support Vector Machine

A. Leitura dos Dados

Os dados de treino e de teste são armazenados em estrutura de dados do tipo *numpy.array*. São utilizados 4 arrays de dados, o primeiro é o X_train (X, 150528) que contém as imagens utilizadas para criar o modelo, X_test (Y, 150528) que contém as imagens utilizadas para teste do modelo, y_train (Z,) que contém as labels das assinaturas utilizadas no treino se são genuínas ou falsificadas, y_test (K,) que contém as labels das assinaturas utilizadas no teste se são genuínas ou falsificadas.

B. Pré-Processamento

As imagens já se encontram processadas como foi explicado no capítulo III nas dimensões 224x224x3 (este último valor a corresponder ao sistema de cores RGB, transformado do formato original BGR).

C. Estrutura Geral

Foi utilizado como base do modelo o parâmetro imutável de 150528 features como entrada (224x224x3) e as 2 classes como saída, em que as assinaturas falsificadas têm valor 1 e as genuínas valor 0[Gupa].

Foram testados os algoritmos *Linear Support Vector Machine* e *Nonlinear Support Vector Machine* com diferentes *kernel*, mas era esperado que o *Linear Support Vector Machine* obtivesse melhores resultados visto que é um problema de classificação binário.
Foram procurados os melhores híper-parâmetros C, gamma e sigma, de modo a que a métrica *Recall* seja maximizada, uma vez que é mais importante não haverem assinaturas falsificadas que sejam classificadas como genuínas. Foram testadas todas as combinações entre os valores [0.01, 0.03, 0.1, 1, 3, 10] para cada um dos *kernel* utilizados.
Os valores ótimos obtidos para cada um dos *kernel* foram os seguintes:

Kernel	C	gamma	sigma
Linear SVM	0.01	100	0.01
RBF SVM	0.01	100	0.01
sigmoid SVM	1	100	0.01
poly SVM	0.01	100	0.01

Tabela II
Accuracy, Precision, Recall e F1 Score de SVM

Depois de calcular os mnelhores híper-parâmetros foi calculada a matriz de confusão, da qual foram calculados os valores de *accuracy*, *precision*, *recall* e *f1 score* para cada modelo.

Kernel	Accuracy	Precision	Recall	F1 Score
Linear SVM	0.8	0.8013	0.8	0.7998
RBF SVM	0.5	0.25	0.5	0.3333
sigmoid SVM	0.5667	0.5679	0.5667	0.5647
poly SVM	0.6667	0.6697	0.6667	0.6652

Tabela III
MELHORES HÍPER-PARÂMETROS C E GAMMA PARA CADA KERNEL

Podemos então observar que o modelo *Linear Support Vector Machine* obteve melhores resultados que o *Nonlinear Support Vector Machine* e foi conseguida uma precisão final de 80% e Recall também de 80%.

VII. NEURAL NETWORK

A. Estrutura Geral

Foi também testado o modelo de *Convolutional Neural Network*, porém devido há falta de tempo, não houve tentativas de melhorar a rede neuronal e feita apenas uma iteração funcional.
Foram usadas a Neural Network pré treinada Inception-V3 para criar uma nova Neural Network adequada a este problema [Gupb].
Os parâmetros de input têm tamanho (X, 224, 224, 3) e (X, 2), é utilizada a função de *loss Categorical Crossentropy* com otimizador *Adagrad*, o brush_size é de 2 e o número de epochs é 20.
Podemos ver nas imagens abaixo a *Accuracy* e a *Loss* dos dados de treino e de teste. É possível ver que a precisão de validação é muito inferior à de treino, o que demonstra um caso claro de overfitting. Este não foi abordado devido à falta de tempo.

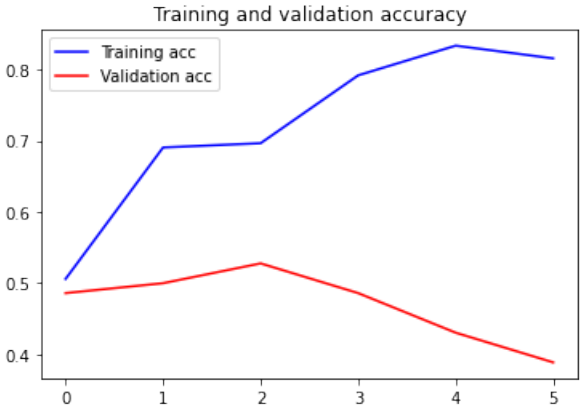

Figura 4. Exemplo de uma imagem depois de ser processada

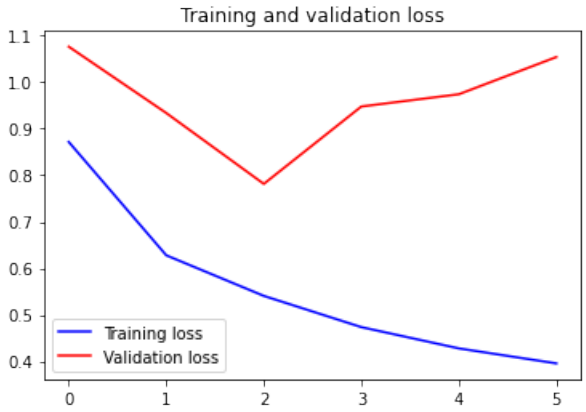

Figura 5. Exemplo de uma imagem depois de ser processada

Este modelo com os dados de teste tem uma Accuracy de 65% e Loss de 65%.

VIII. RESULTADOS

Após uma análise dos modelos, é evidente que os mais adequados são a Logistic Regression e Linear Support Vector Machine. Isto comprova que estes modelos são indicados para classificação de problemas binários.

IX. CONCLUSÃO

Apesar de conseguir uma precisão e recall de 80%, com o modelo Linear Support Vector Machine e de 81.7% com o modelo Logistic Regression, fica algo aquém dos modelos conseguidos actualmente. Uma possível solução para melhorar o modelo seria fazer uso da técnica de *data augmentation*, que conseguiria gerar, a partir do *dataset* usado, uma maior variedade de dados, através de transformações às imagens, ou num melhor caso, ter acesso a um dataset maior do que o usado, que é bastante pequeno. Também penso que poderia ser melhorada a precisão e recall com o uso de uma Rede Neural indicada e aprimorando as configurações das camadas usadas na rede como foi feito no artigo [S18].

X. CONTRIBUIÇÕES

O trabalho foi realizado apenas por uma pessoa.

REFERÊNCIAS

[Haf17] Oliveira L. S. Hafemann L. G. Sabourin R. "Learning features for offline handwritten signature verification using deep convolutional neural networks". Em: *ISSN* (2017). URL: https://www.sciencedirect.com/science/article/abs/pii/S0031320317302017. (accessed: 16.12.2021).

[S18] Jerome Gideon S. "Handwritten Signatures Forgery Detection using Convolutional Neural Networks". Em: *ICACC* (2018). URL: https://www.sciencedirect.com/science/article/pii/S1877050918320301. (accessed: 16.12.2021).

[Bos] Joris Van den Bossche. *Sklearn*. URL: https://scikit-learn.org/stable/. (accessed: 16.12.2021).

[Gupa] Yash Gupta. *Handwritten Signature - Classification*. URL: https://www.kaggle.com/eryash15/handwritten-signature-classification. (accessed: 16.12.2021).

[Gupb] Yash Gupta. *Handwritten Signature - Feature Extraction*. URL: https://www.kaggle.com/eryash15/handwritten-signature-feature-extraction. (accessed: 16.12.2021).

[Rai] Divyansh Rai. *Dataset Handwritten Signatures*. URL: https://www.kaggle.com/divyanshrai/handwritten-signatures. (accessed: 16.12.2021).