PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
FACULDADE DE INFORMÁTICA
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

DESCOBERTA DE ELENCO EM TRAILERS DE FILMES UTILIZANDO RECONHECIMENTO FACIAL EM *DEEP LEARNING*

ALEXANDRE MADRINI MONTEIRO FURTADO
DOUGLAS MATOS DE SOUZA

Trabalho de Conclusão II apresentado como requisito parcial à obtenção do grau de Bacharel em Sistemas de Informação na Pontifícia Universidade Católica do Rio Grande do Sul.

Orientador: Prof. Duncan Ruiz

Porto Alegre
2016

“Inteligência é a capacidade de se adaptar às mudanças.”
(Stephen Hawking)

AGRADECIMENTOS

A realização deste trabalho foi uma grande conquista. Tal conquista não seria possível sem o apoio de pessoas importantes. Primeiramente, gostaríamos de agradecer às nossas famílias, por todo o suporte, compreensão e incentivo durante todo este período. Gostaríamos de agradecer à todos os membros do Grupo de Pesquisa em Inteligência de Negócio e Aprendizado de Máquina (GPIN) por todo o apoio e valiosos conselhos durante o desenvolvimento deste trabalho e também ao Laboratório de Sistemas Autônomos (LSA) por prover o acesso aos recursos computacionais necessários para a realização dos experimentos.

DESCOBERTA DE ELENCO EM TRAILERS DE FILMES UTILIZANDO RECONHECIMENTO FACIAL EM *DEEP LEARNING*

RESUMO

A introdução de técnicas avançadas de *deep learning* na área de aprendizado de máquina trouxe um novo horizonte de possíveis soluções, em especial na área classificação de arquivos de mídias audiovisuais. Através destas técnicas, problemas complexos como o problema de reconhecimento facial em imagens em ambientes não controlados pôde ser solucionado de forma satisfatória. Entretanto, o problema de reconhecimento facial ocorrendo em vídeos ainda não foi explorado de forma significativa. Este trabalho tem como objetivo utilizar técnicas de Redes Neurais em *deep learning* para identificação do elenco em trailers de filmes através de reconhecimento facial. Para tal, foi utilizado um algoritmo para fazer o alinhamento tridimensional das faces contidas em um conjunto de dados que possui imagens de 2.606 celebridades. Este conjunto foi utilizado para treinar uma Rede Neural Convolucional (RNC). Com a modelo treinado, foram submetidos vídeos como entrada, representados por seus respectivos *frames*. Por fim, modelo de classificação treinado foi testado na tarefa de identificar atores em trailers reais onde métricas foram aplicadas para avaliar a qualidade do modelo e das técnicas empregadas. Os resultados obtidos foram promissores.

Palavras-Chave: Aprendizado de máquina. *Deep learning*. Redes neurais convolucionais. Reconhecimento facial. Pré-processamento de vídeo.

CAST RECOGNITION IN MOVIE TRAILERS USING FACIAL RECOGNITION IN DEEP LEARNING

ABSTRACT

The introduction of advanced deep learning techniques in machine learning field brought a new horizon of possible solutions, especially in the field of audio-visual media files classification. By making use of these techniques, complex problems such as the facial recognition in unconstrained images, has been satisfactorily solved. However, the facial recognition problem ocurring in videos has not yet been fully explored. The objective of this work is to use deep neural networks to identify the cast of movie trailers through the use of facial recognition. For such, an algorithm was applied to perform three-dimensional alignment in the faces of a dataset of 2.606 celebrities, this dataset was used to train a convolutional neural network. Finally, the trained classification model was tested on the task of identifying the cast in real movie trailers, where metrics were applied to evaluate the quality of the model and the techniques employed. We achieved promising results.

Keywords: Machine learning. Deep learning. Convolutional neural networks. Facial recognition. Video preprocessing.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

IA – Inteligência Artificial

RNA – Rede Neural Artificial

RNC – Rede Neural Convolucional

MLP – *Multilayer Perceptron*

MSSRC – *Mean Sequence Sparse Representation-based Classification*

PEP – Parte Elástica Probabilística

LFW – *Labeled Faces in the Wild*

YTF – *Youtube Faces Dataset*

SGD – *Stochastic Gradient Descent*

URL – *Uniform Resource Locator*

CPU – *Central Processing Unit*

GPU – *Graphics Processing Unit*

LISTA DE ABREVIATURAS

MI-NS. – Missão: Impossível - Nação Secreta

MATLAB. – *Matrix Laboratory*

VGG Face Dataset. – *Visual Geometry Group Face Dataset*

OpenCV. – *Open Source Computer Vision*

T:OMS. – Thor: O Mundo Sombrio

SUMÁRIO

1. INTRODUÇÃO

Reconhecimento facial é um problema desafiador que tem sido pesquisado extensivamente nas últimas décadas. Os sistemas de reconhecimento facial mais recentes apresentam excelente desempenho ao reconhecer imagens de faces que foram capturadas em condições controladas, ou seja, sem variação de pose, expressão, luz, maquiagem e traços de idade. Entretanto, sua qualidade cai significativamente no reconhecimento de imagens que contém tais variações [3]. Os bons resultados obtidos em ambientes controlados, em vez de sinalizarem o final das pesquisas, apontaram para uma redefinição do problema, movendo a atenção de imagens de faces capturadas em ambientes controlados para imagens de faces capturadas em ambientes não controlados, ou seja, ao ar livre [6].

Comparado com reconhecimento facial em imagens, reconhecimento facial em vídeos traz novos desafios e oportunidades. Faces em vídeos, além de estarem em ambientes não controlados, estão geralmente em uma qualidade inferior, apresentam mais poses assim como traços de movimento. Estes fatores podem significar uma grande variação visual que pode influenciar negativamente o desempenho de sistemas de reconhecimento. Por outro lado, vídeos geralmente contém centenas de *frames* que apresentam uma aparência variada dos mesmos rostos, a qual oferece a possibilidade de combinação de *frames* para gerar sistemas mais robustos [13].

Estes fatores combinados com a introdução de técnicas avançadas de aprendizado de máquina na área de *deep learning* trouxeram um novo horizonte de possíveis soluções na área classificação de arquivos de mídias audiovisuais. Tarefas fáceis de serem feitas por humanos que são feitas automaticamente, de forma intuitiva, como reconhecer objetos e pessoas em imagens são extremamente complexas para computadores. Neste sentido, modelos de aprendizado em *deep learning* habilitam computadores a aprender por experiência e entender o mundo como uma hierarquia de conceitos, fator este que é crucial para o aprendizado sobre mídias audiovisuais [1].

Neste trabalho, a solução proposta foi o desenvolvimento de um sistema automatizado, que faz uso das técnicas mais recentes na área de *deep learning*, para reconhecer elenco presente em trailers de filmes. Esta solução é composta por uma série de etapas, executadas de forma sequencial, que tiveram como objetivo treinar um modelo de aprendizado de máquina que foi utilizado para a descoberta do elenco.

Este trabalho está organizado da seguinte maneira: no capítulo 2 é apresentada a descrição do problema, o cenário de estudos, trabalhos similares e identificação da oportunidade. No capítulo 3 são descritos os objetivos do trabalho. No capítulo 4 é mostrada a implementação da solução proposta. No capítulo 5 são relatados os testes e, por fim, no capítulo 6 é apresentada a conclusão.

2. CARACTERIZAÇÃO DO PROBLEMA

Neste capítulo será apresentado em detalhes o problema de reconhecimento facial em trailers de filmes, bem como as dificuldades envolvidas e soluções propostas por trabalhos similares.

2.1 Cenário de estudo

O cenário de estudo deste trabalho é composto por três principais áreas: trailers de filmes, reconhecimento facial e aprendizado de máquina.

2.1.1 Vídeos

Vídeos são conjuntos de imagens que geram uma sequência de movimento. Atualmente vídeos estão presentes em diversas áreas, como no entretenimento por exemplo com filmes, clipes, entre outros.

Trailers são pequenos vídeos feitos para promover um filme [5]. Por serem reproduzidos em sua maioria através de serviços de vídeo *online*, não contam com informações sobre seu respectivo elenco. Portanto, para identificar atores presentes em um determinado trailer é necessário a visualização do mesmo e o conhecimento pessoal do usuário ou que o mesmo faça pesquisa em serviços de busca *online*. Como o tempo de duração do trailer é normalmente curto e suas tomadas são rápidas e dinâmicas este reconhecimento acaba sendo dificultado. Nas Figuras 2.1, 2.2 e 2.3, que fazem parte do trailer do filme Missão: Impossível - Nação Secreta (MI-NS), podemos ver um bom exemplo, em que, no período de dois segundos três atores diferentes são mostrados.

Figura 2.1 – *Frame* do trailer MI-NS com o ator Jeremy Renner aos 11 segundos

Figura 2.2 – *Frame* do trailer MI-NS com o ator Ving Rhames aos 11 segundos

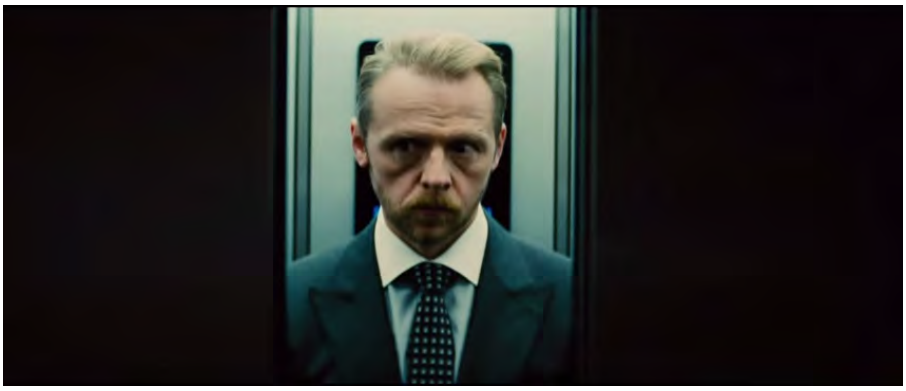

Figura 2.3 – *Frame* do trailer MI-NS com o ator Simon Pegg aos 12 segundos

2.1.2 Reconhecimento facial

Reconhecimento facial é um problema desafiador que tem sido pesquisado extensivamente nas últimas décadas. Sistemas atuais oferecem um bom desempenho ao classificar imagens capturada em condições controladas, ou seja, com invariação de pose, expressão, luz e maquiagem [3], como mostrado na Figura 2.4. A captura de imagens em ambientes controlados requer controle sobre um grande número de variáveis. Para problemas cotidianos e, especialmente para vídeos, o cumprimento destes requisitos é inviável.

O desempenho de sistemas de reconhecimento facial utilizando imagens em ambientes controlados é capaz de superar o de humanos. Contudo, ao invés de sinalizar o final das pesquisas, estes resultados apontaram para uma redefinição do problema, movendo a atenção de imagens de faces em ambientes controlados para imagens em ambientes não controlados, ou seja, ao ar livre, como mostrado na Figura 2.5.

Comparado com reconhecimento facial em imagens, reconhecimento facial em vídeos traz novos desafios e oportunidades. Faces em vídeos, além de apresentarem todas as variações encontradas em ambientes não controlados, contam também com uma qualidade inferior, apresentam mais poses assim como traços de movimentos. A Figura 2.6 é um exemplo, onde o rosto do ator Jeremy Renner contém pesadas expressões faciais além

Figura 2.4 – Face capturada em ambiente controlado. (Fonte: *The PUT Face Database* [9])

Figura 2.5 – Face capturada em ambiente não controlado. (Fonte: *Labeled Faces in the Wild Database* [7])

de borrões decorrentes de movimento. Estes fatores podem influenciar negativamente o desempenho de sistemas de reconhecimento facial. Por outro lado, vídeos geralmente contém centenas de *frames* que apresentam uma aparência variada dos mesmos rostos. Isto oferece a possibilidade de combinação de *frames* para gerar sistemas mais robustos [13].

Figura 2.6 – Face do ator Jeremy Renner em um *frame* do trailer MI-NS aos 11 segundos

2.1.3 Aprendizado de máquina

Quando máquinas programáveis foram concebidas, pessoas se perguntaram se tais máquinas poderiam se tornar inteligentes, mesmo centenas de anos antes de o primeiro computador ser construído. A Inteligência Artificial (IA) é uma área próspera, com várias aplicações reais, e que tem sido alvo de grande atenção tanto na área de pesquisa quanto na indústria. Dentro do contexto de IA, o aprendizado de máquina pode ser aplicado para diversas finalidades, entre elas estão: automatização de tarefas, reconhecimento da linguagem natural, imagens e vídeos, automatização de diagnósticos médicos e suporte à pesquisa científica [1].

Aprendizado de máquina é, portanto, sobre fazer computadores adaptarem suas ações, sejam elas elaborar previsões ou controlar um robô, de modo que estas ações se tornem cada vez mais precisas. A acurácia, reflete o quão corretas foram as decisões tomadas. Em um suposto jogo entre um humano e um computador, é provável que o usuário vença as primeiras partidas, mas que depois de várias partidas o computador passe a vencer, até chegar em um ponto que é praticamente impossível vencer o computador: ou o usuário está piorando, ou o computador aprendeu a jogar. Tendo aprendido as estratégias de um usuário, o computador pode usá-las contra outros usuários, de modo que não é mais necessário começar o jogo sem nenhum conhecimento prévio: esta é uma forma de generalização [14].

Definindo formalmente segundo Mitchell [15, p. 2], podemos dizer que o aprendizado se dá quando:

> "Um programa de computador aprende sobre uma experiência E com respeito a alguma classe de tarefas T e uma medida de desempenho P, se o seu desempenho numa tarefa T, medido por P, aumentar com a experiência E."

Analisando, de forma específica, aprendizado como a ação de melhorar o desempenho P na execução de uma tarefa T através de prática repetitiva, ou seja, adquirindo experiência E, traz à tona algumas questões importantes: como o computador sabe se está melhorando ou não? Como ele sabe como melhorar? Existem várias respostas possíveis

para estas perguntas e cada uma leva a diferentes técnicas de aprendizado de máquina. Segundo Marsland [14, p. 5], estas diferentes respostas trazem uma maneira intuitiva de classificar os algoritmos de aprendizado de máquina quanto a sua forma de aprendizado:

- Aprendizado supervisionado: Um conjunto de exemplos de treino com repostas (também chamadas de rótulos) corretas é fornecido. Baseado neste conjunto de treino, o algoritmo é treinado, e então generaliza para responder a qualquer possível dado de entrada.
- Aprendizado não supervisionado: Diferentemente do aprendizado supervisionado, rótulos não são fornecidos. No caso de *clustering* o algoritmo busca similaridade entre os dados de entrada visando agrupar exemplos que contêm características em comum.
- Aprendizado por reforço: Assim como no aprendizado não supervisionado os rótulos não são fornecidos, porém é realizado um *feedback* positivo ou negativo. A rede utiliza esta informação para ajustar seu comportamento visando melhorar os resultados.
- Aprendizado evolutivo: A evolução biológica pode ser vista como um processo de aprendizado: organismos biológicos se adaptam para melhorar as chances de sobrevivência e reprodução. Algoritmos evolutivos funcionam de maneira semelhante e empregam uma medida chamada *fitness* para medir quão boa é a solução atual.

A abordagem mais comum de aprendizado é a supervisionada. Nesta abordagem, para a etapa de treinamento do algoritmo, é utilizado um conjunto de dados de treino, que é representado por uma matriz X e por um vetor y. A matriz X é do tipo $X \in \mathbb{R}^{m \times n}$, onde m é o número de exemplos de treino e n é o número de atributos de cada exemplo. O vetor y é denotado por $y \in \mathbb{R}^m$ e contém os rótulos para cada exemplo de treino de X. Um exemplo de treino específico dentro da matriz X é denotado por um vetor $\boldsymbol{x}^{(i)} \in \{x^{(1)}, x^{(2)}, ..., x^{(m)}\}$, onde i indica a posição do exemplo em X.

Dentro da abordagem supervisionada existem dois tipos de problemas que podem ser abordados: problemas de classificação e problemas de regressão. De acordo com Marsland [14, p. 6], algoritmos para tais problemas são conceituados da seguinte forma:

- Regressão: Algoritmos de regressão têm como saída um número real, que é determinado a partir de uma função matemática que captura o comportamento apresentado pelo conjunto de treino. A previsão de preços de imóveis pode ser modelada por um algoritmo de regressão, por exemplo. Em um suposto um conjunto de treino $X \in \mathbb{R}^{m \times 1}$, que contém a metragem de imóveis e seus respectivos preços $y \in \mathbb{R}^m$, o algoritmo busca uma função matemática que possa da melhor forma possível se aproximar de todos os pontos $\{x^{(i)}, y^{(i)}\}$. Na etapa de generalização, a função encontrada é utilizada para prever o preço de qualquer imóvel dada sua metragem.

- Classificação: Algoritmos de classificação têm como saída a classe no qual o dado de entrada pertence. A classificação é determinada a partir de uma função matemática que separa exemplos de acordo com suas classes, esta função é conhecida também como fronteira de decisão. A fronteira de decisão captura o comportamento apresentado pelo conjunto de treino. A classificação de emails entre *spam* e não *spam*, por exemplo, pode ser resolvida por um algoritmo de classificação. Em um conjunto de treino X contendo emails e um vetor y contendo suas classes, onde $y^{(i)} \in \{0, 1\}$, 1 em caso de *spam* ou 0 em caso de não *spam*, o algoritmo busca uma função matemática que separe os exemplos $\{x^{(i)}, y^{(i)}\}$ quando $y^{(i)} = 0$ de quando $y^{(i)} = 1$. Na fase de generalização, novos emails são classificados de acordo com qual área da função em que se enquadram.

Neste trabalho, o foco será a abordagem de aprendizado supervisionado, em especial, sua sub-área de algoritmos de classificação. A seguir, é introduzido o conceito de *deep learning*, que compreende uma área de algoritmos de classificação.

Deep learning

Nos primórdios, a área de IA rapidamente resolveu problemas que eram intelectualmente difíceis para humanos, porém, fáceis para computadores, problemas estes que podem ser modelados utilizando regras matemáticas formais. Recentemente, o grande desafio na área de IA mostrou-se resolver problemas que são fáceis para humanos, mas que são difíceis de descrever formalmente, problemas que pessoas resolvem intuitivamente, que parecem automáticos, como reconhecer objetos em uma imagem [1].

A área de *deep learning* tem como objetivo prover soluções para tais problemas, habilitando computadores a aprender e entender o mundo como uma hierarquia de conceitos, onde cada conceito é baseado na sua relação com conceitos mais simples. Esta abordagem habilita computadores a aprender conceitos complicados a partir de conceitos mais simples. Se um grafo for desenhado mostrando como os conceitos são construídos uns em cima dos outros, o grafo seria profundo, com várias camadas. Por esta razão, estas técnicas são chamadas de *deep learning* [1].

Apesar de o termo *deep learning* ser relativamente novo, a área de estudo não é nada recente. O estudo de *deep learning* remete a década de 1950, onde desde então a área de estudo tem sido renomeada para refletir a influência de diferentes pesquisadores e perspectivas. Existiram três ondas de desenvolvimento na área de *deep learning*: *deep learning* conhecido como cibernética entre 1940-1960, *deep learning* como conexionismo entre 1980-1990, e o presente retorno com o nome de *deep learning* em 2006 [1].

Alguns dos primeiros modelos de *deep learning* foram modelos inspirados em modelos de aprendizado biológico, como é o caso das Redes Neurais Artificiais (RNAs). A

seguir, é discutida a motivação por trás de tais modelos bem como suas aptidões e limitações.

Redes Neurais Artificiais

Assim como em outras abordagens em IA, RNAs foram inspiradas pela tentativa de simular o sistema nervoso biológico. O cérebro humano consiste de células nervosas chamadas neurônios, que são ligadas a outros neurônios por um feixe chamado axônio. Axônios são responsáveis por transmitir impulsos elétricos para outro neurônio sempre que o neurônio em questão é estimulado. A interface de conexão entre um neurônio e o axônio de outro neurônio é chamada de dendrito, como mostrado na Figura 2.7. O ponto de contato entre um dendrito e um axônio é chamado de sinapse. Neurocientistas descobriram que o aprendizado humano se dá pela mudança da força da conexão sináptica entre neurônios devido a estímulos repetitivos [22].

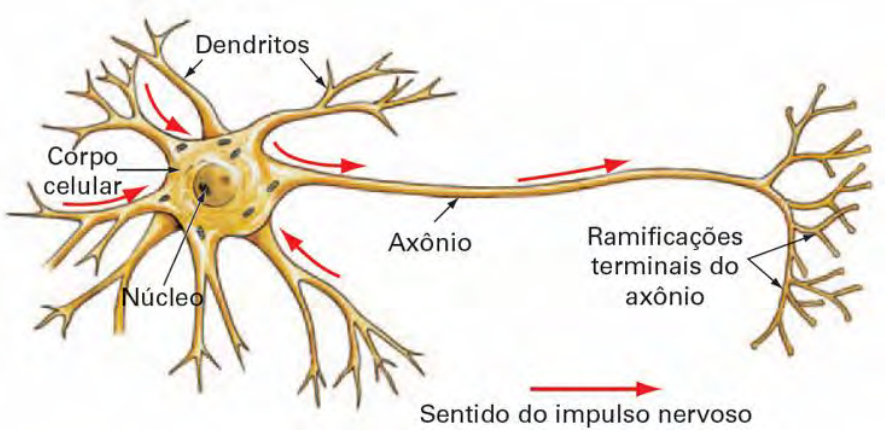

Figura 2.7 – Anatomia de um neurônio. (Fonte: Wasserman [24])

Análoga a estrutura cerebral humana, uma RNA é composta por nós conectados entre si de forma dirigida. O número de neurônios e sua disposição em uma RNA são chamados de arquitetura da RNA [22]. Existem três componentes que compõem a arquitetura de uma RNA. São eles:

- Camada de entrada
- Camada oculta
- Camada de saída

A camada de entrada corresponde aos atributos de entrada do problema. A camada oculta são os neurônios intermediários responsáveis pelo processamento, e recebe este nome pois estas camadas tem como entrada a saída de outros neurônios. Por este motivo, os atributos por elas computados são difusos. Por fim, a camada de saída é a camada que apresenta o resultado, ou seja, a classe no qual o dado de entrada foi classificado.

A RNA mais simples conhecida foi criada em 1958 por Rosenblatt e chama-se *Perceptron* [19]. A rede *Perceptron* possui apenas duas camadas em sua arquitetura, uma camada de entrada e outra de saída. É capaz de capturar o comportamento de operadores lógicos mais comumente utilizados, como o operador NÃO, E e OU.

A *Perceptron* possui um funcionamento muito simples. A rede tem como entrada um vetor x que possui um número j de atributos, os atributos são multiplicados por um vetor w de j pesos, ou seja, cada atributo x_j é multiplicado pelo seu respectivo peso w_j. O resultado das multiplicações dos atributos pelos respectivos pesos é somado. Caso esta soma seja menor ou igual a um limiar, a rede apresenta 0 como saída; caso contrário, apresenta 1. A Figura 2.8 ilustra uma rede *perceptron* que possui 3 atributos como entrada.

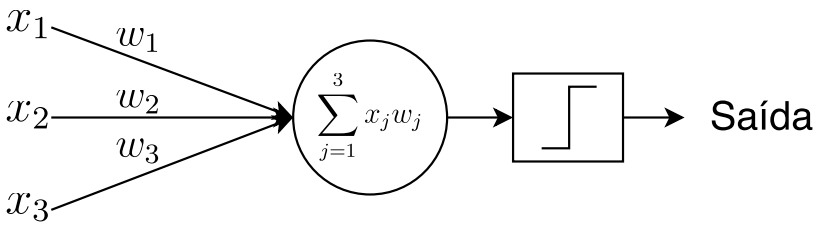

Figura 2.8 – Perceptron

Na Equação 2.1 é mostrado formalmente a definição da rede *perceptron*.

$$\text{saída} = \begin{cases} 0 & \text{se } \sum_j w_j x_j \leq \text{limiar} \\ 1 & \text{se } \sum_j w_j x_j > \text{limiar} \end{cases} \tag{2.1}$$

Observando a Equação 2.1 é possível perceber que existem múltiplas operações de multiplicação entre os atributos de entrada e seus respectivos pesos seguidas de um somatório. Simplificando, podemos descrever a rede simplesmente pelo produto escalar entre o vetor de entrada x e o vetor de pesos w. A outra alteração é mover o limiar para o outro lado da inequação e substituí-lo por um *bias*[1] b [16]. Redefinindo, temos uma fórmula simplificada como mostra a Equação 2.2.

$$\text{saída} = \begin{cases} 0 & \text{se } w \cdot x + b \leq 0 \\ 1 & \text{se } w \cdot x + b > 0 \end{cases} \tag{2.2}$$

Em relação ao aprendizado, a RNA *Perceptron* utiliza um algoritmo de treinamento que resolve um cálculo simples para ajustar neurônios que computam saídas incorretas. Isto é possível pelo fato de a *Perceptron* possuir apenas uma camada de processamento: a camada de saída. Portanto é possível saber qual neurônio necessita ser ajustado. A Equação 2.3 mostra como o vetor de pesos w é ajustado para refletir o comportamento

[1] Será utilizado o termo em inglês *bias* para referenciar a parte fixa da função.

de um exemplo de treino $x^{(i)}$ (vetor de entrada) do conjunto de treino X. Em seguida, na Equação 2.4 é mostrado como é atualizado o *bias*. Em ambas equações, α denota a taxa de aprendizado e $\hat{y}$ denota o valor de classe predito pela rede [19].

$$w_{novo} = w_{antigo} + \alpha(\hat{y}^{(i)} - y^{(i)})x^{(i)} \tag{2.3}$$

$$b_{novo} = b_{antigo} + \alpha(\hat{y}^{(i)} - y^{(i)}) \tag{2.4}$$

Contudo, a *Perceptron* possui uma limitação. Pelo fato de possuir apenas uma camada de processamento, a rede não é capaz criar fronteiras de decisão não lineares, ou seja, problemas não linearmente separáveis, como é o caso do operador lógico OU EXCLUSIVO, não podem ser resolvidos.

Fronteiras de decisão mais complexas podem ser criadas adicionando camadas ocultas na arquitetura da RNA. RNA *Perceptron* que faz uso de arquiteturas multicamadas são chamadas de *Multilayer Perceptron* (MLP), como mostra a Figura 2.9. Combinando camadas ocultas é possível modelar problemas de classificação que possuem fronteiras de decisão polinomiais de alta ordem.

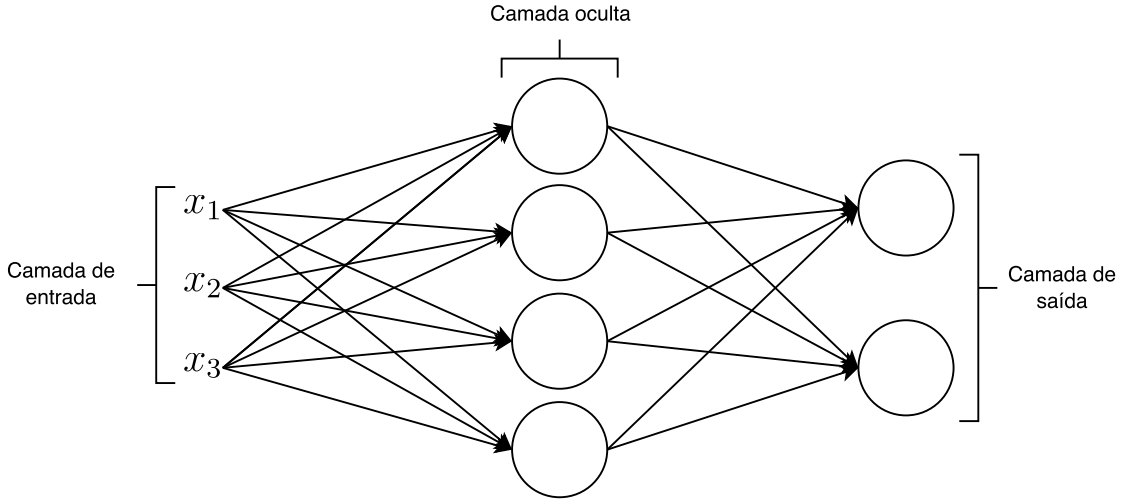

Figura 2.9 – Arquitetura da RNA *Multilayer Perceptron*

Entretanto, no caso da MLP, o treinamento é mais complexo. Em uma suposta rede multicamadas de *perceptrons*, um neurônio que computa uma saída incorreta sofre uma pequena alteração nos seus pesos e *bias*, como mostra a Equação 2.3 e 2.4. O neurônio atualizado deve então passar a computar a saída correta, mas embora a atualização tenha sido pequena, ela pode causar um grande impacto no neurônio seguinte, mudando a sua saída de 0 para 1, por exemplo. Este comportamento acaba por desbalancear a rede, dificultando o aprendizado [16]. Para contornar este problema das redes *Perceptron*, foi introduzido um novo tipo de neurônio, chamado de neurônio sigmóide.

Neurônios sigmóide são semelhantes aos neurônios utilizados pela rede *Perceptron*, exceto, que uma pequena alteração nos seus pesos, causa uma pequena alteração na sua saída. Assim como os neurônios da rede *Perceptron*, neurônios sigmóide tem como entrada um vetor x de j atributos, mas em vez de computar valores de saída 0 ou 1, eles computam valores *entre* 0 e 1 [16]. Por exemplo, 0.638 é uma saída válida para um neurônio sigmóide. Portanto, podemos alterar a rede *perceptron* para computar $\sigma(w \cdot x + b)$, onde σ é a função sigmóide, que pode ser vista na Equação 2.5.

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.5}$$

A Figura 2.10 mostra a diferença entre a função sigmóide e a utilização do limiar. É notável a suavidade da função sigmóide, significando que uma alteração Δ no peso w e no *bias* b produz uma pequena mudança Δ na saída da rede.

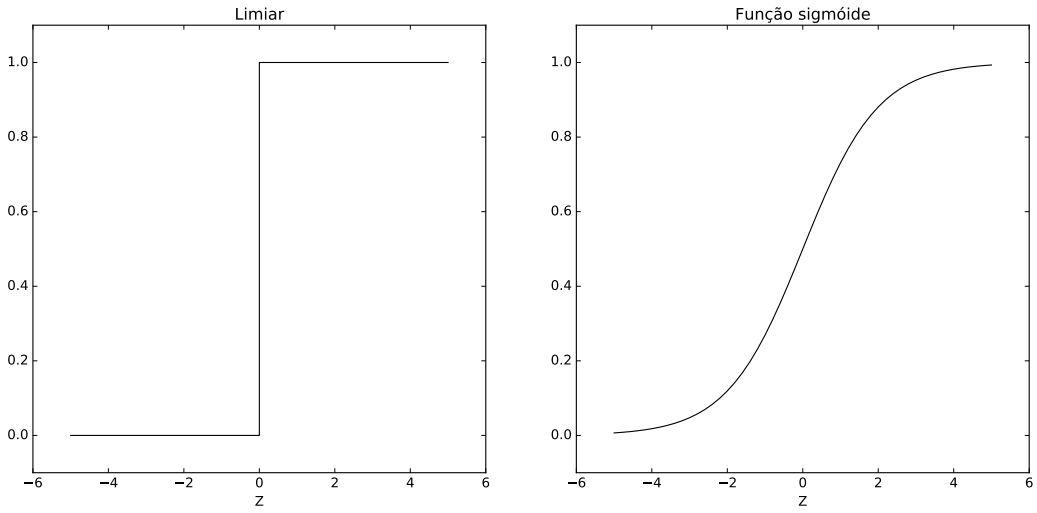

Figura 2.10 – Comparação entre função sigmóide e limiar

O algoritmo *Backpropagation* foi um marco divisor na pesquisa em RNAs. Ele foi originalmente introduzido na década de 70, porém, sua importância não foi apreciada até a divulgação de um trabalho feito por David Rumelhart, Geoffrey Hinton e Ronald William [25] em 1986. Este trabalho mostrou como o *Backpropagation* funciona mais rapidamente que outras abordagens de aprendizado, tornando possível a solução de problemas que até então não eram solucionáveis. Atualmente, o *Backpropagation* é o principal algoritmo de aprendizado em RNAs [16].

Backpropagation trata-se de um algoritmo iterativo, onde cada iteração é composta por duas fases: a fase de *feed forward* e a fase de *backward propagation*. Durante a fase de *feed forward*, os pesos inicializados de forma aleatória, ou adquiridos de iterações anteriores, são utilizados para computar a saída de cada neurônio da rede, de forma sequencial começando pela primeira camada. Na fase de *backward propagation*, a fórmula de atuali-

zação de pesos é aplicada no sentido reverso. Esta abordagem habilita estimar o erro da camada l utilizando a saída da camada $l+1$ [22].

Para compreender o funcionamento do *backpropagation* é necessário definir uma notação para que seja possível se referir a alguns componentes da MLP, como por exemplo, neurônios, ativações e *bias*. A notação definida a seguir será a mesma utilizada por Nielsen [16]. Para se referir aos pesos da rede de forma direta e não ambígua, será utilizado w^l_{jk} para denotar o peso entre o $k^{\text{ésimo}}$ neurônio na $(l-1)^{\text{ésima}}$ camada e o $j^{\text{ésimo}}$ neurônio da $l^{\text{ésima}}$ camada.

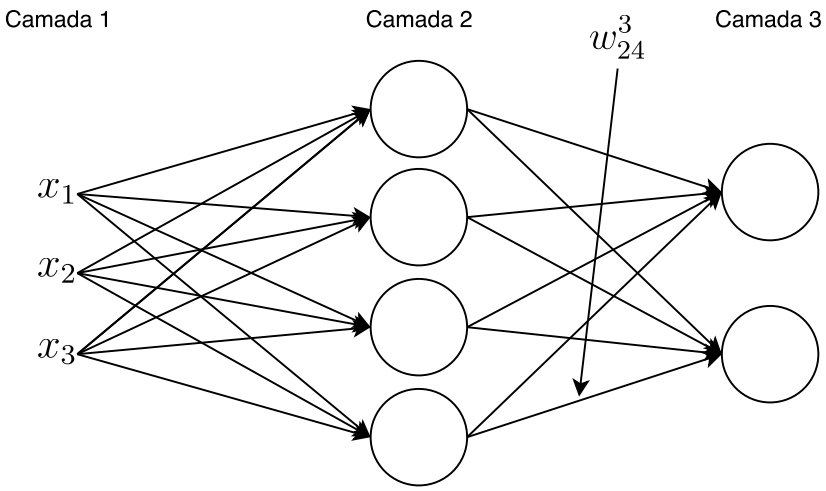

Figura 2.11 – MLP

Por exemplo, a Figura 2.11 demonstra como referenciar o peso w^l_{jk} originado do $k^{\text{eśimo}}$ neurônio na camada $(l-1)$ para o neurônio $j^{\text{ésimo}}$ da $l^{\text{ésima}}$ camada. A primeira vista, esta notação parece um pouco intimidadora, mas com o tempo ela passa a fazer sentido e se torna natural [16]. Da mesma maneira, será utilizado b^l_j para referenciar o $j^{\text{ésimo}}$ *bias* da $l^{\text{ésima}}$ camada, como também a^l_j para referenciar a ativação do $j^{\text{ésimo}}$ neurônio da $l^{\text{ésima}}$ camada.

Na Equação 2.6, z^l_j denota o somatório das multiplicações dos pesos pelos atributos de entrada do $j^{\text{ésimo}}$ neurônio da $l^{\text{ésima}}$ camada. Simplificando, a Equação 2.7 demonstra o cálculo de forma vetorizada para todos os neurônios da $l^{\text{ésima}}$ camada. O resultado de z^l é então usado como entrada na função de ativação $\sigma(z)$, que calcula a ativação de cada um dos neurônios da $l^{\text{ésima}}$ camada, como demonstra a Equação 2.8.

$$z^l_j = \sum_k w^l_{jk} a^{l-1}_k + b^l_j \tag{2.6}$$

$$z^l = w^l a^{l-1} + b^l \tag{2.7}$$

$$a^l = \sigma\left(z^l\right) \tag{2.8}$$

A partir deste ponto, o objetivo é encontrar um conjunto de pesos w e *bias* b de modo que a saída da rede a^L para um exemplo $x^{(i)}$ seja igual ao seu respectivo rótulo $y^{(i)}$. Para tanto, é necessário definir uma função de custo $C(w, b)$ que informe quão adequado o conjunto atual de pesos w e *bias* b estão em relação a saída esperada. A Equação 2.9 mostra a função de custo conhecida como função de custo do erro médio quadrático, ou *mean squared error (MSE)* em inglês. Observando a função de custo, é possível perceber que o resultado de $C(w, b)$ nunca é negativo, visto que os termos da soma são sempre positivos. Adicionalmente, $C(w, b) \approx 0$ quando o resultado da rede $a^L(x^{(i)})$ é aproximadamente igual ao valor esperado $y^{(i)}$ [16].

$$C(w, b) = \frac{1}{2}\sum_{i}^{m}(y^{(i)} - a^L(x^{(i)}))^2 \tag{2.9}$$

A função de custo mostrada na Equação 2.9 já está escrita como a média das funções de custo aplicadas a cada exemplo de treino $x^{(i)}$. Isto é necessário para que se possa calcular a média das derivadas parciais $\partial C/\partial w$ e $\partial C/\partial b$ com respeito aos pesos w e aos *bias* b, respectivamente.

O algoritimo *Backpropagation* tem como objetivo estimar o erro para cada neurônio da rede, afim de atualizar os seus pesos para que a rede como um todo passe a computar a saída esperada. O primeiro passo do algoritmo é estimar o erro δ_j^l para o $j^{\text{ésimo}}$ neurônio da $l^{\text{ésima}}$ camada para posteriormente atualizar os pesos w_j^l e *bias* b_j^l. As Equações 2.10, 2.11, 2.12 e 2.13 resumem a forma vetorizada dos passos executados pelo *backpropagation*, onde $\odot$ representa o produto de Hadamard, que calcula simplesmente a multiplicação elemento por elemento de duas matrizes e ∇ representa o vetor de derivadas parciais.

Resumo do Backpropagation:

$$\delta^L = \nabla_a C \odot \sigma'(z^L) \tag{2.10}$$

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l) \tag{2.11}$$

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l \tag{2.12}$$

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1}\delta_j^l \tag{2.13}$$

A seguir, é apresentada uma técnica mais recente chamada Rede Neural Convolucional (RNC), que utiliza algoritmos avançados de classificação para classificar mídias audiovisuais.

Redes Neurais Convolucionais

Redes Neurais Convolucionais (RNCs) são um tipo especial de RNA especializadas no processamento de dados que possuem uma topologia no formato de grade. RNCs têm sido bem sucedidas em aplicações práticas, especialmente com imagens e vídeos. Como o nome implica, RNCs são redes que empregam uma operação matemática chamada de convolução. Convolução é um tipo especial de operação linear. Segundo [1, p. 274]: "RNCs são simples RNAs que usam convoluções no lugar da tradicional multiplicação de matrizes em pelo menos uma de suas camadas".

A operação de convolução, também conhecida como correlação, quando aplicada em imagens, consiste em deslizar um filtro bidimensional, conhecido também como *kernel*, sobre as camadas da imagem de entrada e calcular o produto escalar entre as entradas do filtro e as da imagem. O tamanho do deslocamento do filtro é chamado de *stride*. O resultado da convolução são imagens chamadas de *feature maps*. O ponto chave da camada de convolução em RNCs é que os filtros são compostos de parâmetros que podem ser aprendidos. Estes parâmetros podem modelar o comportamento dos filtros para que cada um seja ativado ao ver determinado comportamento na imagem de entrada, como contornos, formas e até objetos. Matematicamente a operação de convolução geralmente é denotada por um asterisco. Um exemplo de convolução aplicado em uma imagem I utilizando um filtro K pode ser vista na Equação 2.14.

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n) \tag{2.14}$$

A motivação por trás da convolução consiste em três conceitos principais: interações esparsas, compartilhamento de pesos e representações equivariantes. Ao contrário da tradicional multiplicação de matrizes, redes convolucionais possuem interações esparsas (também chamadas de conectividade esparsa ou pesos esparsos). Por exemplo, ao processar imagens, a imagem de entrada pode ter milhares ou milhões de *pixels*, mas é possível detectar atributos significativos como bordas e contornos utilizando *kernels* que ocupam apenas centenas de *pixels*. O compartilhamento de pesos refere-se ao fato de que cada membro do *kernel* é utilizando em diferentes posições na imagem de entrada. Isto significa que em vez de aprender uma combinação de pesos para cada região da imagem de entrada, é aprendido apenas uma combinação para toda a imagem. Por fim, no caso da convolução, o compartilhamento de pesos implica em uma propriedade chamada equivariancia a translações, ou seja, o comportamento presente em determinada região da imagem de entrada é tratado da mesma forma quando o mesmo aparece em regiões diferentes [1].

Um dos grandes desafios de usar modelos de RNCs é construir uma boa arquitetura e configurar os parâmetros da rede. As camadas convolucionais, que diferenciam as RNCs de RNAs comuns, são compostas de quatro hiper-parâmetros principais. São eles:

- Número de filtros ***K***.
- Dimensão do filtro ***F***.
- *Stride* ***S***, que representa o tamanho do passo.
- *Padding* ***P***, que representa o tamanho do preenchimento.

Além das camadas convolucionais, geralmente outras camadas são combinadas para otimizar a rede. As mais comuns são as camadas de não linearidade, que aplicam funções de ativação nas saídas da camada anterior, e também camadas *subsampling*, que reduzem o número de atributos a serem processados pela camada posterior. Algumas das camadas mais comumente utilizadas são:

- *Max-Pooling*: é uma camada de *subsampling*, responsável por reduzir a quantidade dados a serem processados pela camada posterior ao mesmo passo em que mantêm aspectos importantes dos mesmos. O *Max-Pooling* consistem em capturar o valor máximo de uma vizinhança definida por um *kernel* [1]. Um exemplo de *Max-Pooling* pode ser visto na Figura 2.12.

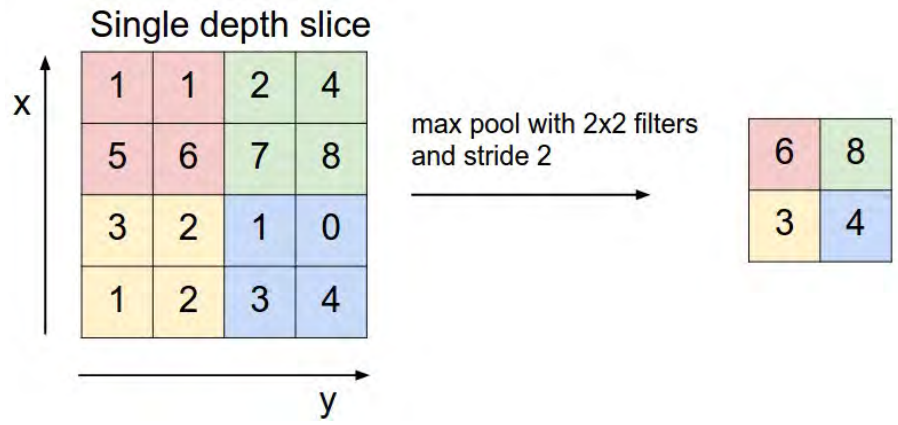

Figura 2.12 – Exemplo de Max-Pooling (Fonte: [11])

- *ReLu*: o *ReLu* (*Rectified Linear Unit*) é um tipo de ativação muito comum em redes profundas. Consiste em calcular $max(0, x)$ para saída de cada neurônio da camada anterior. O comportamento do *ReLu* pode ser observado na Figura 2.13.
- *Softmax*: o *softmax* também um tipo de ativação, consiste em calcular a probabilidade distribuída de um vetor. Em outras palavras, o *softmax* transforma um vetor para outro vetor cuja soma de seus elementos é igual a 1. Esta camada é utilizada especialmente na saída das redes para calcular a probabilidade de o dado de entrada pertencer a cada uma das classes.
- *Dropout*: o *dropout* [20] é uma técnica de regularização, consiste no descarte de um número aleatório de valores resultantes da camada anterior.

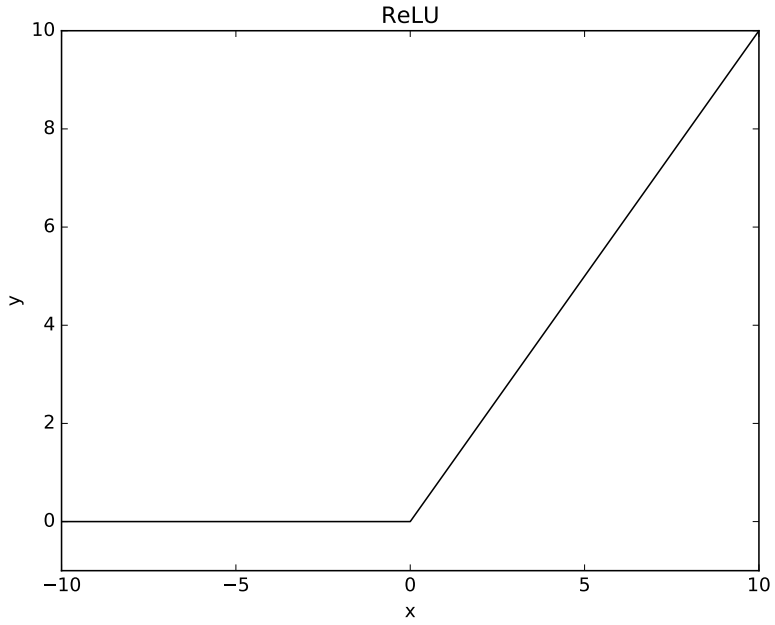

Figura 2.13 – Comportamento do *ReLu*

- Totalmente conectada: a camada totalmente conectada é uma MLP, é utilizada geralmente como uma das últimas camadas com o propósito de realizar a classificação.

Existem várias outras camadas que possuem diferentes propósitos. Inclusive, novas camadas estão surgindo em decorrência do grande esforço de pesquisa e da rápida evolução da área de *deep learning*. As camadas acima foram citadas por serem populares e por que foram utilizadas no desenvolvimento deste trabalho.

2.2 Trabalhos similares

Existem diversas abordagens diferentes para o reconhecimento facial em vídeos. Nesta seção estão presentes algumas delas que foram avaliadas nos seguintes estudos.

2.2.1 Face Recognition in Movie Trailers via Mean Sequence Sparse Representation-based Classification [17]

No estudo proposto por Enrique G. Ortiz *et al.* [17] é apresentado o mesmo problema estudado neste projeto, porém, com um algoritmo que utiliza a sequência de *frames* de uma cena para obter uma gama maior de poses da mesma face e, assim, aumentando as chances de reconhecimento facial do ator.

Na Figura 2.14 é possível ver melhor como é o funcionamento do programa. Primeiro é feita a detecção de uma cena e o enquadramento do rosto. Posteriormente, os *frames* da cena são sobrepostos para gerar uma imagem melhor. São, então, selecionados até quatro *frames* para gerar uma sequência. As imagens são finalmente alinhadas e a comparação é feita. Neste caso é utilizado um método de votação entre as imagens da sequência para descobrir de quem é o rosto.

Com este método foi possível alcançar 80.75% de acertos no problema YouTube Faces Dataset de Wolf *et al.* [26].

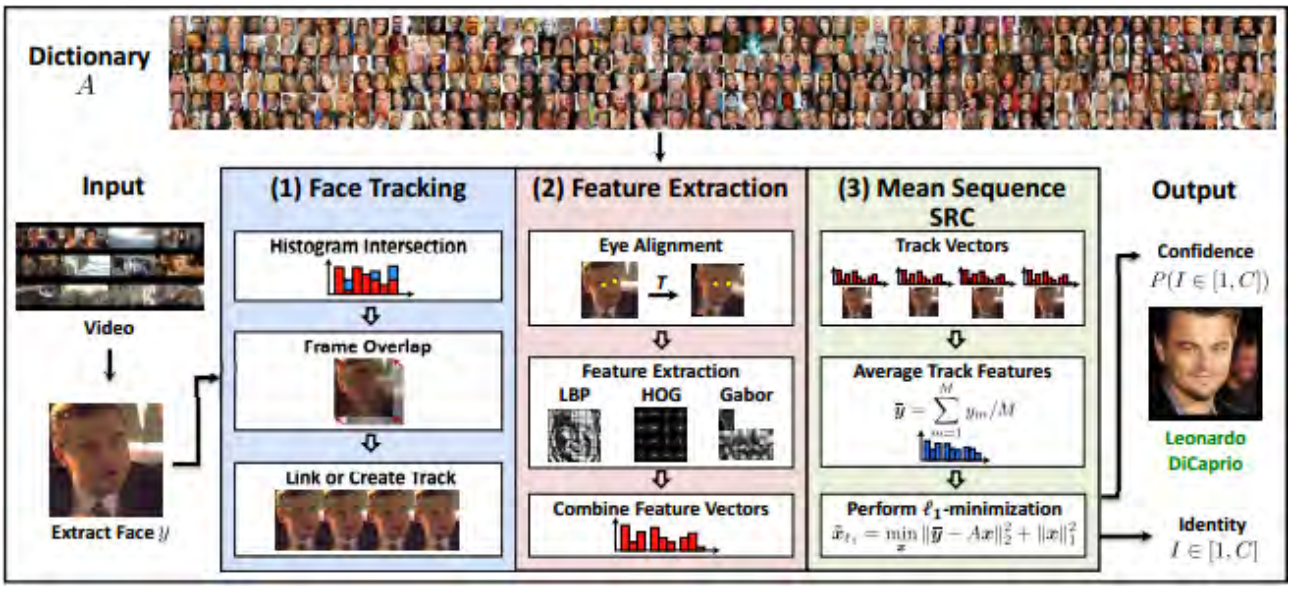

Figura 2.14 – Funcionamento do método MSSRC (Fonte [17])

2.2.2 Eigen-PEP for Video Face Recognition [13]

O método Eigen-PEP utiliza o modelo de parte elástica probabilística (PEP). O modelo PEP é treinado com diversas imagens de faces para determinar os pontos chaves de uma face que possuam locais fixos em um rosto como a boca por exemplo. Na Figura 2.15 podemos observar a comparação realizada pelo método, na qual são comparados cada um dos pontos gerados anteriormente, que são as linhas, dentre todas as imagens fornecidas que são as colunas.

O Eigen-PEP gera a representação PEP de cada *frame* da cena. Os pontos gerados são, então, sobrepostos para gerar uma melhor imagem de descrição. Estes passos são executados tanto para gerar a base de treino como para gerar a imagem a ser comparada de uma determinada cena.

A avaliação do Eigen-PEP em cima do problema YouTube Faces Dataset de Wolf *et al.* [26] foi superior ao método apresentado anteriormente atingindo 82.40% de acertos.

Com alguns ajustes e correções na base de treino, o resultado foi ainda melhor chegando a 85.04%.

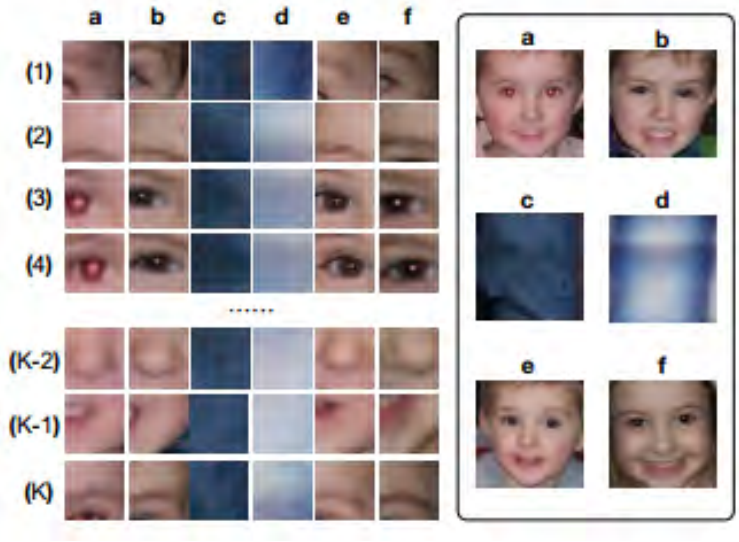

Figura 2.15 – Exemplo de comparação de seis imagens pelo método PEP (Fonte [12])

2.3 Identificação da oportunidade

Com base na evolução de técnicas de reconhecimento facial por aprendizado de máquina em *deep learning*, e na recente utilização de técnicas de frontalização facial para o alinhamento de faces, foi levantada a possível utilização de ambas tecnologias para resolver o problema de reconhecimento de pessoas em vídeos, empregando as técnicas de frontalização tanto na base de treino quanto no reconhecimento dos indivíduos presente nos trailers, visando assim fazer com que todas as faces possuíssem o mesmo alinhamento, e com isto ter esperanças de aumentar a eficácia em comparação aos métodos já existentes.

3. OBJETIVOS

Neste capítulo são discutidos os objetivos gerais e específicos deste trabalho bem como os requisitos da solução.

3.1 Objetivo Geral

Este trabalho tem como objetivo a criação de uma solução capaz de descobrir o elenco de um determinado trailer utilizando aprendizado de máquina em *deep learning* para realizar o reconhecimento facial.

3.2 Objetivos específicos

A solução deve receber uma base de dados de imagens contendo faces de celebridades juntamente com o rótulo para formar os conjuntos de treino, validação e teste. Os conjuntos serão então utilizados para treinar, validar e avaliar um modelo de aprendizado de máquina em *deep learning*. Os dados de treino devem conter no mínimo todos atores presentes nos trailers a serem utilizados, ocorrendo caso contrário a não identificação de determinado ator pela falta do mesmo na base de dados utilizada durante o treinamento.

É desejável que os trailers utilizados no reconhecimento possuam uma boa qualidade, visto que serão utilizados seus respectivos *frames* para realizar o reconhecimento facial e, portanto, caso um trailer de baixa qualidade seja utilizado, dificultaria o reconhecimento devido a consequente baixa qualidade da imagem da face extraída dele.

3.3 Requisitos da solução

Receber conjunto dados

A solução deve poder receber um conjunto de dados e separá-los em três subconjuntos: treino, teste e validação.

Pré-processar conjunto dados

A solução deve realizar o pré-processamento do conjunto de dados, que inclui: recorte e frontalização de todas as faces e o cálculo da média das imagens presentes no conjunto de treino.

Treinar modelo em RNC

Realizar o treinamento do modelo com os dados do conjunto de treino.

Receber trailer

A solução deve ser capaz de receber um trailer em um formato de vídeo padrão.

Selecionar *frames*

A solução deve selecionar os *frames* do vídeo recebido que contenham faces.

Frontalizar faces

A solução deve frontalizar todas faces encontradas no vídeo.

Reconhecer atores

A solução deve reconhecer os atores presentes nos *frames* selecionados com a premissa que eles estão contidos na base de dados recebida anteriormente.

Gerar lista com resultado

A solução deve gerar um arquivo texto contendo os nomes dos atores e número de vezes em que eles aparecem no trailer.

4. IMPLEMENTAÇÃO

Neste capítulo são apresentados os métodos utilizados para o desenvolvimento do projeto.

4.1 Visão Geral

A solução proposta é a implementação de um sistema automatizado que faz uso de diferentes técnicas em aprendizado de máquina afim de buscar um melhor resultado na tarefa de identificação do elenco. De modo a oferecer uma solução para reconhecimento do elenco, foi necessário treinar um modelo em RNC. Neste sentido, a solução contempla duas etapas principais: a etapa de treinamento, que envolve todo esforço necessário de captura e pré-processamento das imagens contidas em um conjunto de imagens de atores, e a etapa de generalização que, por sua vez, contempla o esforço necessário para processar um trailer, extrair seus *frames* e submeter as faces encontradas para a RNC realizar o reconhecimento.

O diagrama apresentado na Figura 4.1 mostra, de maneira geral, as atividades executadas durante a etapa de treinamento.

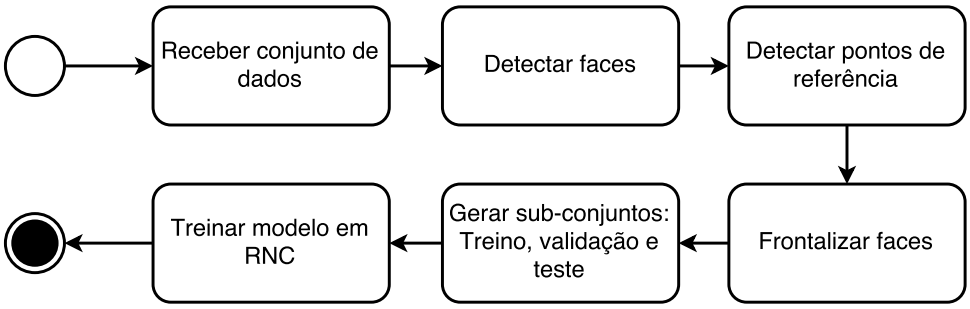

Figura 4.1 – Etapa de treinamento

A etapa de generalização, ou seja, de reconhecimento do elenco presente em trailers, ocorre após a etapa de treinamento e pode ser vista no diagrama da Figura 4.2. É possível perceber a repetição de três atividades em ambos os modos de execução: detectar faces, detectar pontos de referência, frontalizar.

Visando organizar a implementação da solução e habilitar a reutilização de código e componentes, foi utilizada uma implementação separada em módulos. A solução é dividida em três grandes módulos:

- Módulo de entrada de dados: este módulo compreende as atividades de recebemento do conjunto de dados para treinamento e trailer para descoberta do elenco.

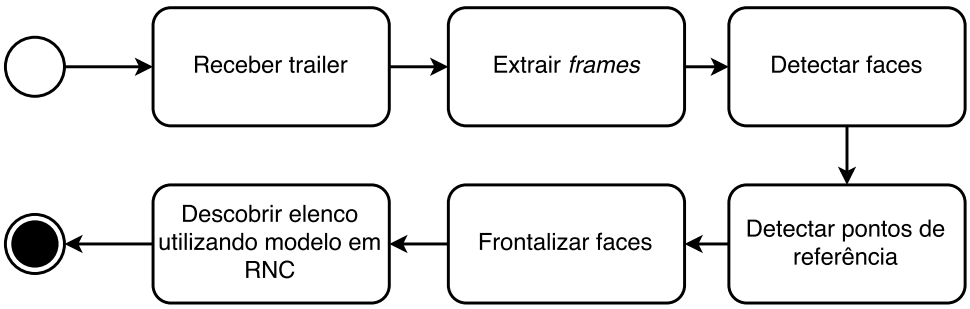

Figura 4.2 – Etapa de generalização

- Módulo de Pré-processamento: este módulo compreende as atividades de detecção de faces, detecção pontos de referência e frontalização de faces. Este módulo é responsável pelo pré-processamento dos dados, de forma a deixá-los no formato correto para o módulo de RNC.
- Módulo de RNC: Este módulo compreende as atividades de treinamento do modelo em RNC e descborta do elenco utilizando a RNC. É responsável pelo treinamento da RNC e pela descoberta do elenco de trailers.

Os detalhes do conteúdo de cada módulo são descritos nas seções seguintes.

4.1.1 Tecnologias Utilizadas

Durante o desenvolvimento do projeto foram utilizadas diversas tecnologias, nas seções a seguir essas tecnologias são detalhadas e sua utilização explicada.

- Python: A linguagem de desenvolvimento escolhida foi Python pela sua facilidade de uso e pela sua versatilidade. A linguagem foi empregada em todas as fases do projeto, devido a sua praticidade foi possível criar códigos para tarefas complexas com facilidade, como por exemplo o *download* e verificação das imagens, graças a módulos existentes.
- Git: Como a implementação do projeto foi executada por duas pessoas, foi necessário utilizar um sistema de controle de versão para manter o controle das mudanças de código. Para esta finalidade, optou-se por utilizar Git, pois ele é um software livre e conhecido, de fácil utilização e vastamente empregado.
- Dlib [10]: Dlib é uma biblioteca de código aberto desenvolvida em C++ que disponibiliza funções para ajudar no desenvolvimento de aplicações de aprendizado de máquina. Neste projeto ela foi empregada para detectar os pontos de referência de cada face. Estes pontos, por usa vez, foram utilizados para realizar a frontalização facial. A

biblioteca em questão também foi utilizada para a detecção facial, tanto nas imagens da base de treino quanto nos *frames* de trailers utilizados para o reconhecimento do elenco.

- *Open Source Computer Vision* (OpenCV) [2]: Visto que o projeto tem um grande foco no processamento e manipulação de imagens e vídeos, foi necessária uma biblioteca robusta para realizar essas tarefas. Para suprir essa necessidade foi escolhida a biblioteca OpenCV que possui todos os recursos necessários e é amplamente utilizada.
- Caffe [8]: Atualmente existem diversos *frameworks* para o treinamento de redes profundas. Optou-se por utilizar o Caffe pela sua facilidade de uso, por ter uma vasta comunidade de usuários e por permitir treinamento acelerado utilizando GPUs (*Graphics Processing Unit*). Adicionalmente, assim como as bibliotecas mencionadas anteriormente, o Caffe também possui uma interface que pode ser utilizada em Python, que possibilitou uma implementação integrada com os demais módulos deste projeto.

4.2 Obtenção do conjunto de dados

Para realizar o treinamento da RNC é necessário um conjunto de dados de acordo com o objetivo desejado. No caso de detecção de carros por exemplo, seria necessário uma base de dados com diversas fotos de carros para que com isso, durante o treinamento, a rede identifique e aprenda o que deve ser encontrado. Porém para o reconhecimento facial, como tem de ser feito o reconhecimento da pessoa em si, se torna necessário múltiplas imagens de cada uma dessas pessoas para treinar a rede. Por isso foram utilizadas duas bases de dados já existentes, a *Labeled Faces in the Wild* (LFW) e a *Visual Geometry Group Face Dataset* (VGG Face Dataset).

4.2.1 Labeled Faces in the Wild

A LFW é uma base de dados com 13.233 imagens de 5.749 pessoas gerada com o intuito de estudar o problema de verificação facial em ambientes não controlados, ela foi concebida pelo estudo *Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments* por Huang *et al.* [7] e visa comparar duas imagens para saber se pertecem a mesma pessoa. Como nossa solução não tem como objetivo comparar duas imagens, mas sim treinar uma rede para que ela possa reconhecer as faces das pessoas que já foram ensinadas a ela, utilizamos esta base para um teste inicial da rede e do método de frontalização, visto que a quantidade de imagens presentes nesta base é consideravelmente menor que o presente na base VGG Face Dataset.

Devido a necessidade de mais de uma imagem por pessoa para realizar o treinamento e teste da rede foi tomada a decisão de remover todas pessoas da base de dados que possuíam somente uma imagem, restando assim 9.164 imagens de 1.680 pessoas.

4.2.2 Visual Geometry Group Face Dataset

A base de dados gerada pelo estudo *Deep Face Recognition* por Parkhi *et al.* [18], possui dados sobre mais de $2.000.000$ de imagens de mais de 2.600 celebridades distribuídos em múltiplos arquivos texto separados por nomes de celebridades. Cada arquivo texto possui uma lista com as URLs (*Uniform Resource Locator*) das respectivas imagens.

Para realizar o treinamento da RNC, foi necessário fazer o *download* das imagens presentes nesta base de dados. Um programa específico para percorrer as URLs presentes nos arquivos foi desenvolvido para esta finalidade.

Devido ao fato de URLs poderem se tornar indisponíveis a qualquer momento, uma verificação se tornou necessária. Foi decidido pelo descarte de qualquer URL caso a mesma não retornasse uma resposta em até 2 segundos. Outro problema encontrado durante o *download* da base de dados foi que, mesmo com a URL disponível, algumas imagens haviam sido apagadas, retornando uma mensagem de erro no lugar da imagem. Para que o programa de *download* não gerasse arquivos inválidos, foi implementada uma verificação na qual os dados retornados pela URL foram verificados para saber se se tratava de uma imagem de fato: caso os dados não fossem de uma imagem, o arquivo era descartado.

Após o descarte das URLs não acessíveis e dos arquivos que não eram imagens foi gerada uma base de dados com $1.958.805$ imagens de 2.606 celebridades. Este processo teve a duração de aproximadamente 16 dias rodando paralelamente em 2 computadores.

4.3 Pré-processamento

O pré-processamento é responsável por aumentar a pureza dos dados, bem como transformar para o formato aceito pela RNC. Portanto, os seguintes métodos de pré-processamento foram utilizados com a intenção de otimizar o desempenho da RNC.

4.3.1 Frontalização Facial

A frontalização facial, ou alinhamento tridimensional, é processo de síntese de uma visão frontal de faces capturadas em condições não controladas. Resultados sugerem que o processo de frontalização pode melhorar a performance de sistemas de reconhecimento facial [6]. Para tanto, a frontalização transforma o problema desafiador de reconhecimento facial em ambientes não controlados para uma abordagem mais simplista de reconhecimento em ambientes controlados.

Para realizar a frontalização, a solução proposta por Hassner *et al.* [6] conta a utilização de técnicas de computação gráfica e de visão computacional para calcular de forma aproximada a matriz da câmera utilizada para capturar a imagem. Utilizando a matriz da câmera e um modelo de rosto tridimensional é possível renderizar uma visão frontalizada inicial do rosto.

Adicionalmente, o alinhamento tridimensional da face pode acarretar em partes do rosto menos visíveis que outras, especialmente do lado no nariz e da cabeça. A solução de Hassner *et al.* [6] calcula de forma estimada a visibilidade de cada lado do rosto, para decidir qual lado precisa de ajustes. O lado que sofre oclusão recebe o espelhamento do outro lado da face com uma intensidade proporcional ao grau da oclusão. Os passos executados durante a frontalização podem ser vistos na Figura 4.3.

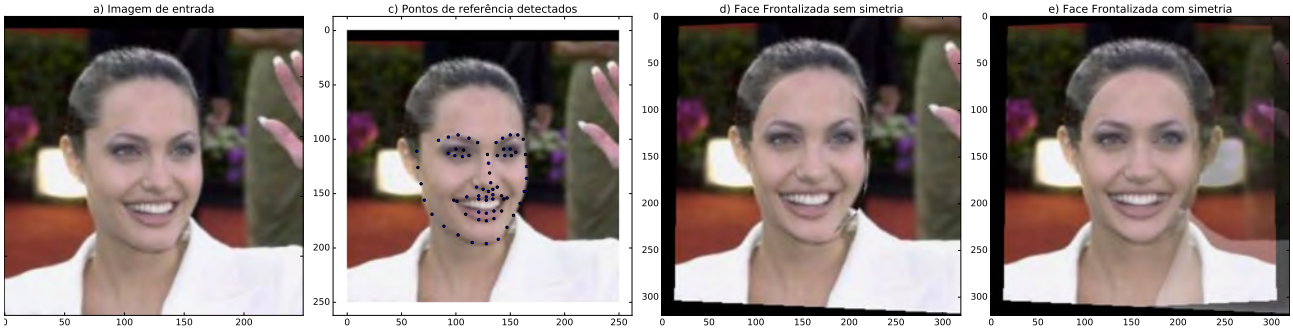

Figura 4.3 – Exemplo de frontalização facial

Hassner *et al.* [6] disponibilizaram na página do projeto[1] uma implementação do algoritmo de frontalização facial para a plataforma MATLAB. Como parte deste trabalho, foi realizada uma migração desta implementação para a linguagem Python para que esta solução pudesse ser utilizada de forma integrada com as outras ferramentas e bibliotecas. A implementação da solução de frontalização em Python está disponível publicamente e pode ser encontrada em: https://www.github.com/dougsouza/face-frontalization.

A implementação em Python foi utilizada para frontalizar e extrair somente o rosto de cada imagem do conjunto de dados. Entretanto, como pode ser visto na Figura 4.4,

[1] http://www.openu.ac.il/home/hassner/projects/frontalize

diversas imagens possuíam mais do que um rosto, o que poderia fazer com que a face da pessoa errada fosse extraída. A primeira solução proposta para contornar este problema foi selecionar sempre a maior face presente na imagem, contudo, em algumas imagens como a Figura 4.4, as pessoas presentes estavam lado a lado e a maior face detectada não era da pessoa certa. Logo, para que o conjunto de dados pré-processado contivesse a menor quantidade de erros possível, optou-se por descartar as imagens em que a maior face possuísse uma diferença de área menor que 10% em relação as demais.

Figura 4.4 – Imagem do ator Tom Hiddleston ao lado do ator Chris Hemsworth presente na base VGG Face Dataset.

Dependendo da intensidade de oclusão presente na face, o algoritmo de frontalização facial pode gerar imagens distorcidas. Portanto, uma lógica teve de ser implantada, na qual, caso não seja detectada uma face na imagem após a mesma ter sido frontalizada, o rosto sem frontalização é utilizado. A Figura 4.5 mostra a visão geral de todo o processo de frontalização do conjunto de dados.

O processo de frontalização durou aproximadamente 10 dias e após o descarte das imagens que não passaram pelo mesmo restaram $1.710.269$ imagens de 2.606 pessoas do VGG Face Dataset prontas para o treino da RNC.

4.3.2 Subtração do valor médio das imagens

Para aumentar a acurácia do reconhecimento é sugerida a subtração de cada imagem pela média de todas as imagens presentes na base de dados, que é calculada pixel a pixel com a soma dos canais RGBs de todas imagens dividido pelo número de imagens

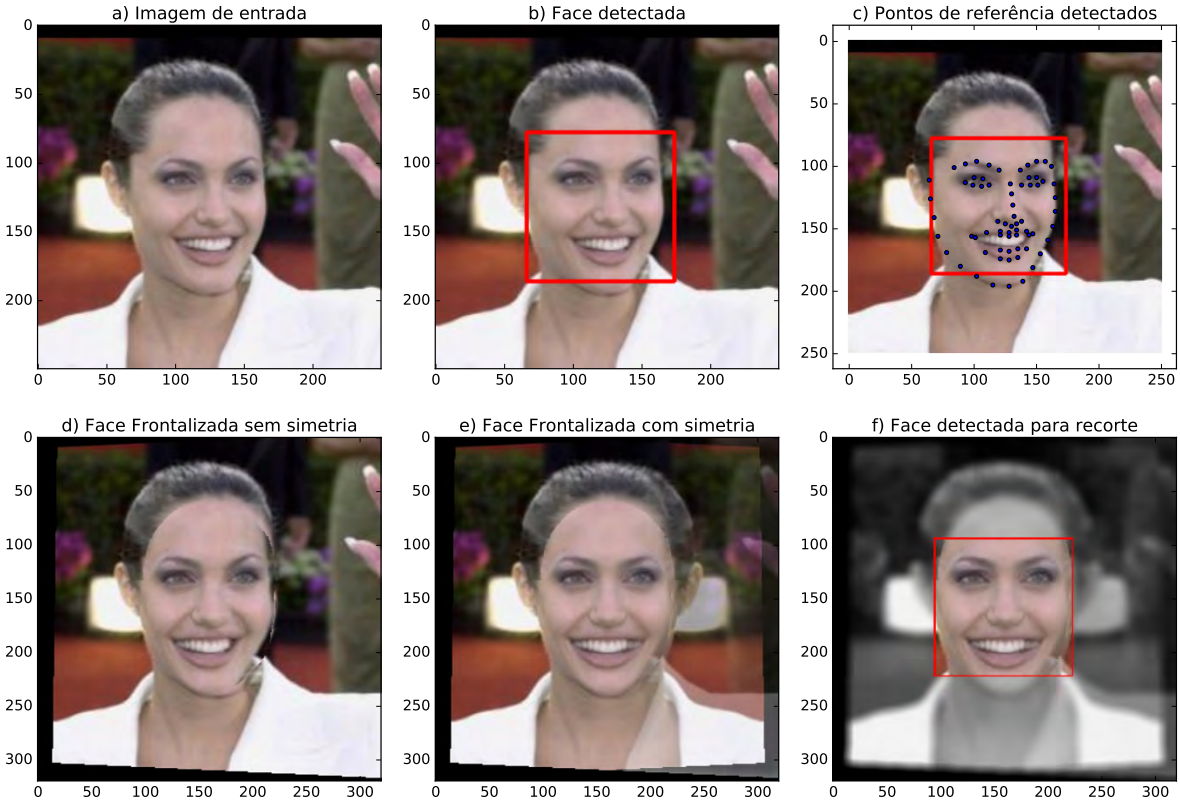

Figura 4.5 – Visão geral do processo de frontalização do conjunto de dados

presentes na base. Na Figura 4.6 é possível ver a média da VGG Face Dataset com aproximadamente $1.200.000$ imagens, também é possível notar que com o uso do algoritmo de frontalização a maior parte das imagens ficou com o mesmo alinhamento.

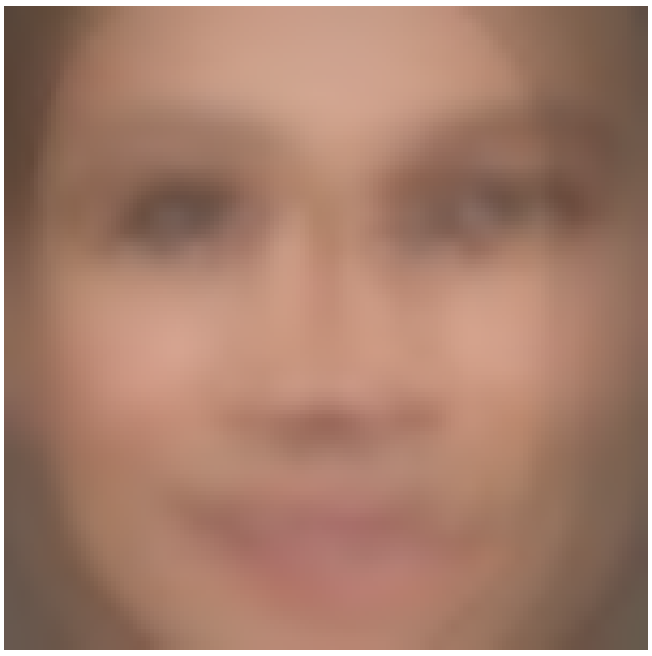

Figura 4.6 – Média das imagens presentes na base gerada pelo VGG Face Dataset.

4.4 Modelo em RNC

Redes Neurais Convolucionais têm sido largamente utilizadas para realizar reconhecimento facial. Visando utilizar tal solução para identificar os atores, foi treinada uma RNC utilizando o *framework* Caffe [8]. O Caffe possibilita o treinamento de redes neurais profundas utilizando GPUs, que fornecem um desempenho de cálculos de álgebra linear muito superior ao de CPUs (*Central Processing Unit*). A seguir são apresentadas especificações da RNC bem como os detalhes da execução do treinamento e teste.

4.4.1 Arquitetura

A Arquitetura de RNC escolhida foi uma arquitetura idêntica a utilizada pelo *DeepFace* [21]. O trabalho *DeepFace*, obteve um excelente desempenho, em especial no *benchmark Youtube Faces Dataset* (YTF) [26], que mede a verificação de faces em vídeos. A verificação de faces consiste em apontar se um par de faces pertencem ao mesmo indivíduo ou não. Embora a verificação não seja a finalidade deste trabalho, a versatilidade desta arquitetura utilizada pelo *DeepFace* foi o motivo da escolha. Logo, como o objetivo deste trabalho é reconhecer os atores, o modelo de RNC treinado não foi avaliado no *benchmark* YTF.

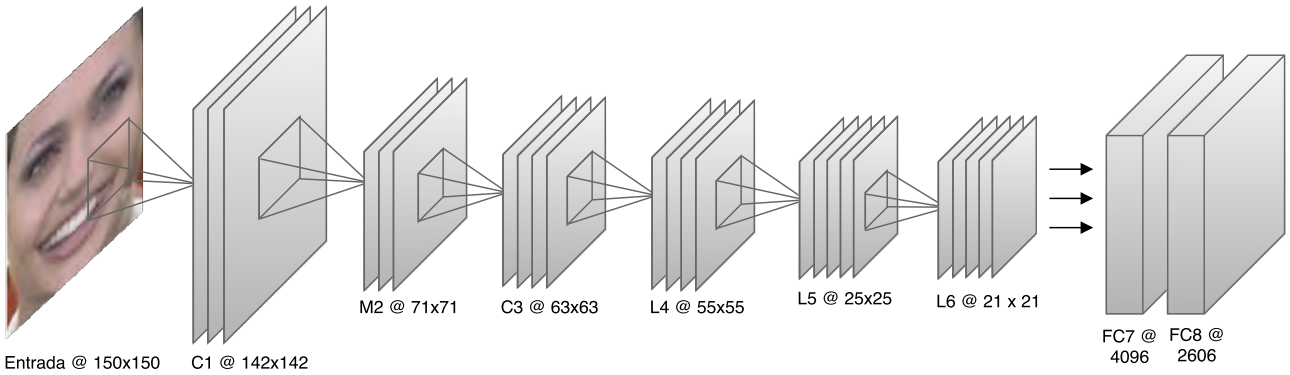

Figura 4.7 – Arquitetura da RNC

As camadas mostradas na Figura 4.7 têm as seguintes especificações:

- C1: Convolução com 32 filtros de 11×11, *padding* de 1 e *stride* 1, resultando em 32 *feature maps* de 142×142;
- M2: *Max-Pooling* aplicado a uma região de 3×3 e *stride* 2, resultando em 32 *feature maps* de 71×71;

- C3: Convolução com 16 filtros de 9×9 e *stride* 1, resultando em 16 *feature maps* de 63×63;
- L4: Localmente conectada com 16 filtros de de 9×9 e *stride* 1, resultando em 16 *feature maps* de 55×55;
- L5: Localmente conectada com 16 filtros de 7×7 e *stride* 2, resultando em 16 *feature maps* de 25×25;
- L6: Localmente conectada com 16 filtros de 5×5 e *stride* 1, resultando em 16 *feature maps* de 21×21;
- FC7: Totalmente conectada com 4096 unidades de saída;
- FC8: Totalmente conectada com 2606 unidades de saída, cada uma representando uma classe, ou seja, uma celebridade.

Adicionalmente, depois de cada camada de convolução (C1 e C3), camada localmente conectada (L4, L5 e L6) e da primeira camada totalmente conectada (FC7) é aplicada a ativação *ReLU*. A função custo utilizada foi a função de entropia cruzada multinomial.

A camada localmente conectada [4] não é muito popular em redes profundas. Ela se comporta de forma semelhante a camada de convolução; porém, não há compartilhamento de pesos entre diferentes regiões da imagem de entrada. Diferentemente, cada filtro é localmente conectado a uma área da imagem de entrada. O uso deste tipo de camada é interessante quando a suposição de que características semelhantes podem aparecer em diferentes regiões da imagem não é verdadeira, como é o caso das faces frontalizadas.

Até o momento da implementação deste trabalho, o *framework* Caffe não possuía suporte nativo para o uso de camadas localmente conectadas. Foi realizada uma alteração no código fonte do *framework* para adicionar uma implementação de camada localmente conectada disponibilizada por um membro da comunidade no *GitHub*.

Por fim, durante o treinamento, foi utilizado o *droupout* para prevenir *overfitting*. Ele foi aplicado com o valor de $0,5$ depois da primeira camada totalmente conectada (FC7), ou seja, para cada passagem de uma instância de treino pela rede, um conjunto aleatório de 50% dos valores resultantes da primeira camada totalmente conectada (FC7) é descartado.

4.4.2 Treino e teste

A escolha de parâmetros para o treinamento da RNC é uma tarefa complexa. Atualmente, não é possível saber de antemão quais parâmetros irão produzir um melhor resultado. A melhor solução é testar diferentes combinações de parâmetros realizando vários

treinamentos. Infelizmente, para este trabalho, nós não obtivemos recurso computacional necessário para o treinamento disponível a todo momento. Dentro do espaço que nos foi concedido, foi possível realizar uma execução do treinamento.

Na execução do treinamento optou-se por utilizar o algoritmo de otimização Adadelta [27]. Assim como maioria dos otimizadores, este algoritmo também é baseado no cálculo do gradiente. O Adadelta foi escolhido pelo fato de ser um algoritmo adaptativo, ou seja, a taxa de aprendizado por ele utilizada é ajustada automaticamente conforme o andamento do treino. Os parâmetros utilizados para a execução do treinamento foram:

- Taxa de aprendizado inicial: a taxa utilizada foi de 1. Apesar desta ser uma taxa demasiadamente alta, este não é um problema, já que o Adadelta a ajusta conforme necessário.
- Delta: Este é único parâmetro necessário para o Adadelta. Foi utilizado o valor padrão de 1×10^{-6}.
- Tamanho do *batch*: Foram utilizado *batchs* de 512 imagens.
- Número de iterações: O treinamento foi executado por 7000 iterações, que equivale a aproximadamente 3 épocas.
- Teste durante o treino: Para acompanhar o treinamento, foi utilizada uma amostra aleatória de 5% do conjunto de validação para realizar testes a cada 250 iterações.

A evolução da função de custo durante o treinamento pode ser vista na Figura 4.8. A execução do treinamento levou aproximadamente 32 horas utilizando uma *GPU* Nvidia Tesla K40. Após o término da execução, o modelo resultante foi avaliado medindo sua acurácia nos conjuntos de validação e teste. A acurácia no conjunto de validação foi de $52,70\%$ enquanto a acurácia no conjunto de teste foi de $52,81\%$.

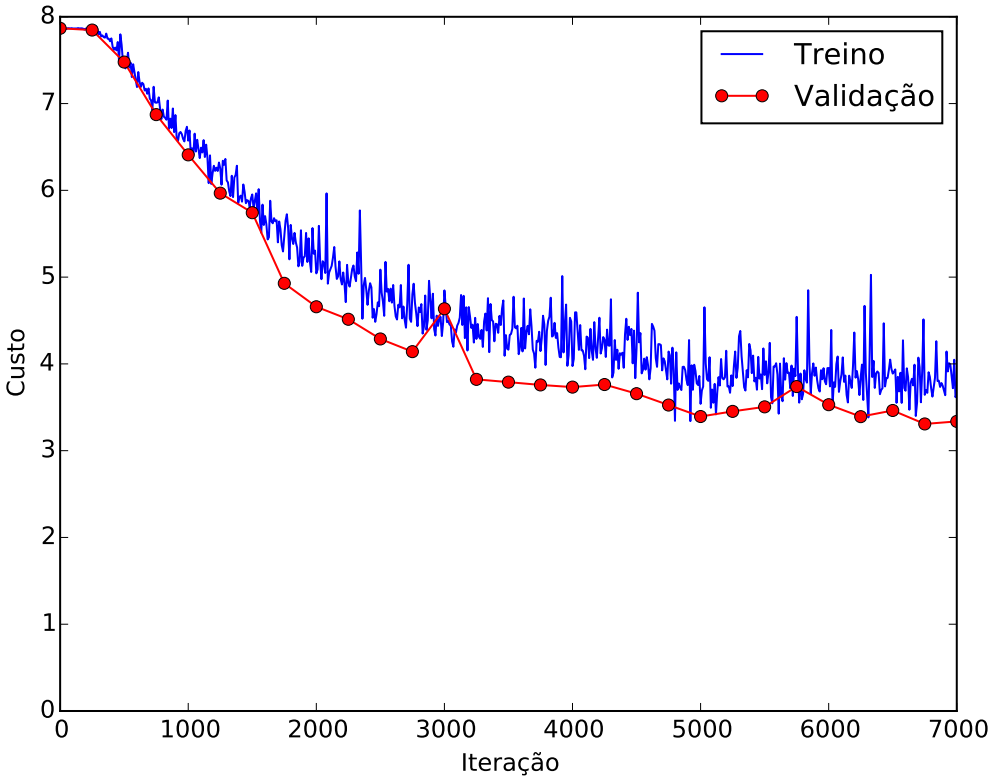

Figura 4.8 – Função de custo durante o treinamento com Adadelta.

5. TESTES

5.1 Reconhecimento em trailers

Para realizar o reconhecimento facial em trailers foi necessário desenvolver um algoritmo para percorrer vídeos e detectar faces. Como um vídeo é composto de diversos *frames* foi necessário utilizar uma técnica de amostragem para diminuir a quantidade de dados a ser processada. Para isto, utilizou-se uma técnica de amostragem conhecida como amostragem intencional, na qual os dados selecionados são baseados em um propósito específico [23]. O critério utilizado para amostragem é a existência de faces no *frame*.

Como pode ser visto nas Figuras 5.1 e 5.2, que fazem parte do trailer do filme Thor: O Mundo Sombrio (T:OMS), faces presentes em vídeos podem possuir má iluminação ou serem sobrepostas por efeitos dificultando assim no seu reconhecimento.

Figura 5.1 – *Frame* do trailer do filme T:OMS onde o ator Chris Hemsworth aparece com má iluminação.

5.1.1 Trailers utilizados para teste

Devido a base do VGG Face Dataset não conter as celebridades presentes no LFW, encontrar um trailer de filme no qual o elenco esteja presente na base se torna uma tarefa difícil. Após algumas pesquisas foram escolhidos três trailers para teste, que são mostrados a seguir.

Figura 5.2 – *Frame* do trailer do filme T:OMS onde a atriz Natalie Portman aparece sobreposta por outra imagem.

Thor: O Mundo Sombrio (T:OMS)

Neste trailer aparecem diversos atores que estão presentes na base de dados utilizada para o treinamento da rede. Durante o trailer os rostos destes atores são mostrados em diversas poses, como exemplificado na Figura 5.3 onde o ator Chris Hemsworth é reconhecido com sucesso mesmo estando em uma posição pouco favorável para seu reconhecimento.

O Apêndice A contém uma tabela com os atores reconhecidos pela rede, o número de *frames* em que ele foi reconhecido e a média da probabilidade do reconhecimento.

Figura 5.3 – *Frame* do trailer do filme T:OMS onde o ator Chris Hemsworth é reconhecido com 20% de probabilidade.

(500) Dias Com Ela

O trailer do filme (500) Dias Com Ela foi escolhido por possuir uma boa iluminação e diversos *closes* de faces dos atores principais, porém seu resultado não foi o esperado. Foi encontrada somente a atriz Zooey Deschanel e em apenas 2 *frames*, já o ator Joseph Gordon-Levitt que interpreta o personagem principal não foi reconhecido pela RNC. Na Figura 5.4 é mostrado um dos frames no qual a rede reconheceu a atriz Zooey Deschanel mesmo estando de lado para a câmera.

A pessoa com a maior quantidade de reconhecimentos neste trailer foi a atriz Selma Blair com sua face sendo reconhecida em 355 *frames* mesmo não estando presente no mesmo. Uma lista completa com todos atores reconhecidos pode se vista no Apêndice B.

Figura 5.4 – *Frame* do trailer do filme (500) Dias Com Ela onde a atriz Zooey Deschanel é reconhecida com 1% de probabilidade.

A Rede Social

No trailer do filme A Rede Social aparecem diversas faces, a maior parte delas não está presente na base de dados, porém como pode ser visto na Figura 5.5 o ator Jesse Eisenberg, que faz o personagem principal é reconhecido. Diversos outros atores são reconhecidos erroneamente devido a RNC não conhecer todas as pessoas presentes no trailer, a lista completa de reconhecimento pode ser vista no Apêndice C.

5.2 Visualização

Para visualização dos resultados foi desenvolvido um algoritmo para alterar o trailer. O algoritmo consiste em desenhar um retângulo em volta dos rostos detectados jun-

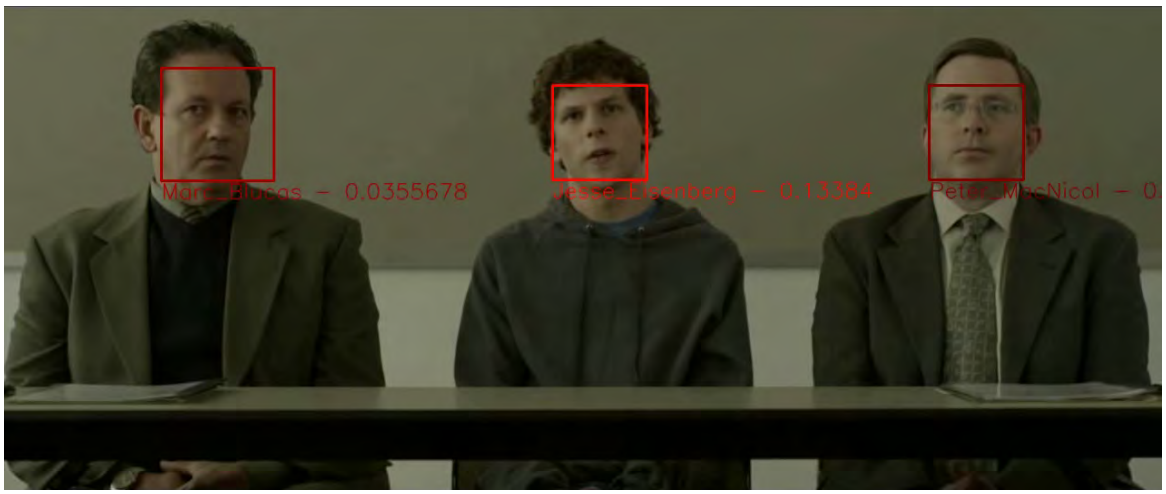

Figura 5.5 – *Frame* do trailer do filme A Rede Social onde o ator Jesse Eisenberg é reconhecido com 13% de probabilidade.

tamente de seus respectivos rótulos e probabilidades. Os rótulos e probabilidades foram previstos pela RNC. Uma visualização dinâmica das probabilidades foi realizada utilizando uma escala de cores chamada *jet*, onde a cor vermelho escuro representa um baixa probabilidade e a cor azul escuro representa uma alta probabilidade. A escala completa de cores pode ser vista na Figura 5.6.

Na Figura 5.7 pode ser visto um exemplo de visualização onde a atriz Natalie Portman é reconhecida com $27,53\%$ de probabilidade. Na escala de cores *jet* o número 0.2753 corresponde a um tom da cor laranja.

Como a probabilidade resultante da rede é distribuída entre as 2606 classes, durante os testes não foram comuns predições com mais de 30% de probabilidade.

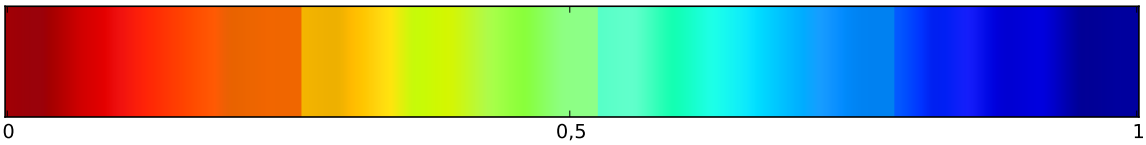

Figura 5.6 – Escala de cores *jet*.

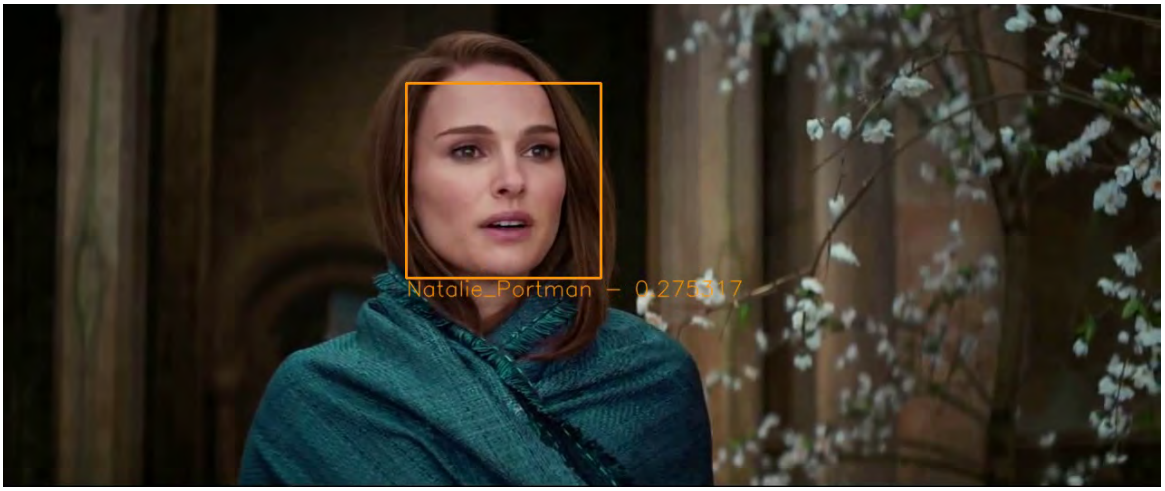

Figura 5.7 – *Frame* do trailer do filme T:OMS onde a atriz Natalie Portman é reconhecida com 27% de probabilidade.

6. CONCLUSÃO

Reconhecimento facial é um problema desafiador que tem sido pesquisado extensivamente nas últimas décadas, e que nos últimos anos obteve um grande progresso decorrente das novas descobertas na área de *deep learning*. Os sistemas de reconhecimento facial mais recentes apresentam excelentes desempenhos ao reconhecer imagens de faces que foram capturadas em condições controladas, ou seja, com invariação de pose, expressão, luz, maquiagem e traços de idade. Entretanto, a qualidade cai significativamente no reconhecimento de imagens que contém tais variações.

É notável o aumento da complexidade ao migrar a mesma abordagem para realizar reconhecimento de vídeos. Faces presentes em *frames* contêm uma variedade imensa de variações de pose e luz, além de apresentar em uma qualidade de imagem inferior devido a borrões decorrentes de movimento.

Além das dificuldades relacionadas a utilização de vídeos, obter um conjunto de dados com um grande número de instâncias e de boa qualidade também é um desafio. Após o treinamento da RNC, percebemos que existe uma grande variedade de ruídos e dados rotulados incorretamente na base de dados *VGG Face Dataset*, que acaba por acarretar em dificuldades no treinamento da RNC e impacta diretamente sua capacidade de generalização.

Ao utilizar uma série de algoritmos de aprendizado de máquina em conjunto, como é caso deste trabalho: detecção da face, detecção de pontos de referência da face, frontalização facial e treinamento de uma RNC, onde cada algoritmo depende do resultado do algoritmo anterior, é importante avaliar o desempenho de cada um deles para saber o quanto cada um pode impactar na solução final. Nós observamos que a área da detecção facial causa um grande impacto na qualidade da solução, por ser o ponto de partida para os outros algoritmos.

Por fim, mesmo com todas as dificuldades e desafios mencionados anteriormente, a solução foi capaz de encontrar nos trailers os atores que estavam presentes na base de dados de treinamento da RNC. Ainda que a lista de atores encontrados nos trailers contenha vários atores que não estejam presentes, em alguns *frames* a solução foi capaz de encontrar o rótulo certo para o ator presente no respectivo *frame*. Estes resultados sugerem que esta abordagem funciona, e que pode ser melhorada a ponto de atingir grandes taxas de acerto.

6.1 Trabalhos futuros

Devido recursos limitados durante o decorrer do trabalho não foi possível a execução de algumas etapas, que acabaram ficando para segundo plano. Os tópicos seguintes apontam algumas ideias para uma possível melhora na taxa de acertos da RNC.

6.1.1 Limpeza da base de dados

Mesmo com todas precauções tomadas, a base de dados gerada possui muitas imagens rotuladas incorretamente. Isto prejudica o treinamento da RNC visto isto a impede de aprender uma representação condizente com a realidade.

Para melhorar a qualidade do treino da RNC é necessário fazer uma limpeza da base removendo as imagens rotuladas incorretamente, ou até mesmo atribuindo o rótulo correto. Infelizmente, não houve tempo hábil para realizar esta tarefa dentro do prazo do trabalho visto que são mais do que $1.200.000$ imagens para serem verificadas.

6.1.2 Detecção da face

Devido as diversas poses em que uma face pode ser encontrada, a extração de uma face do *frame* de um trailer nem sempre retorna o mesmo recorte, podendo assim o rosto extraído possuir um corte maior ou menor do rosto dificultando assim o reconhecimento. Nas Figuras 6.1 e 6.2 podemos ver claramente que com a detecção mais abrangente do primeiro *frame* a rede reconhece a face errada, mas no *frame* seguinte com a área de detecção menor a face é reconhecida corretamente.

Para que isto seja resolvido seria necessária uma configuração mais restrita sobre a detecção e extração de uma face de uma imagem para que todas as faces extraídas possuíssem as mesmas características.

6.1.3 *Tunning* de parâmetros para treinamento da RNC

A escolha de parâmetros para o treinamento de uma RNC é uma tarefa complexa. Uma maneira de procurar um modelo com melhor desempenho, é executar um treinamento para cada combinação de parâmetros. Realizar este *tunning* de parâmetros não foi possível

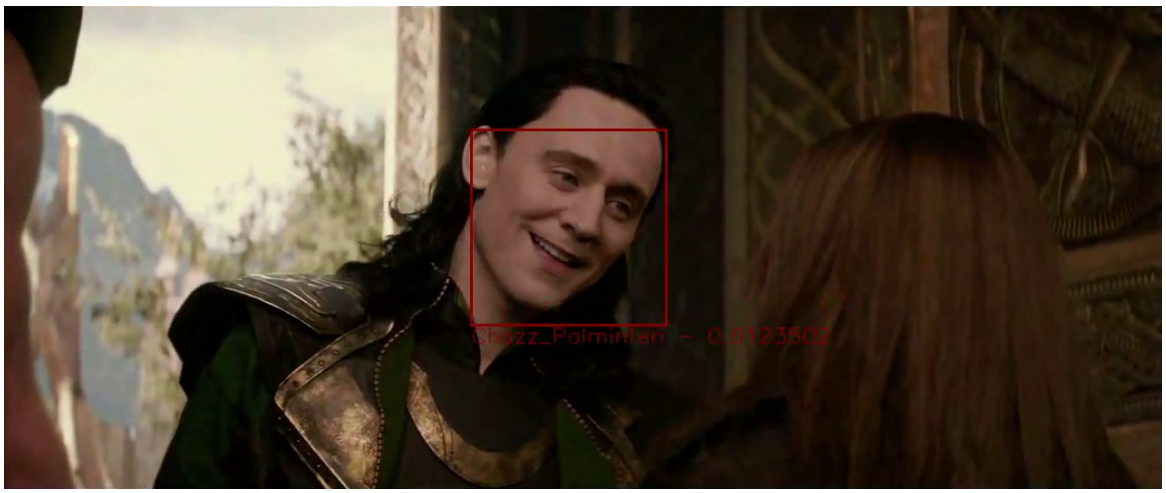

Figura 6.1 – *Frame* do trailer do filme T:OMS onde o ator Chazz Palminteri é reconhecido erroneamente com 1% de probabilidade.

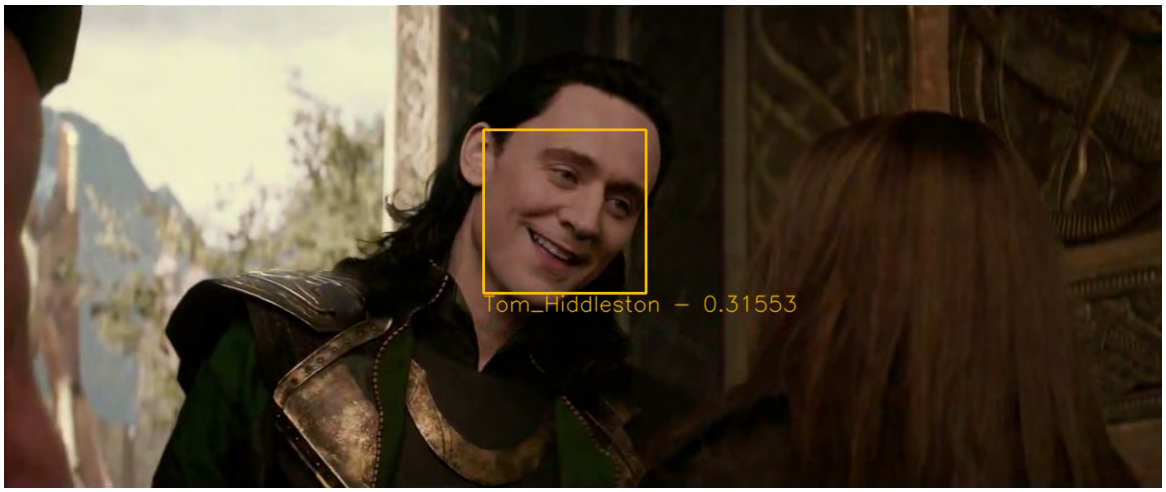

Figura 6.2 – *Frame* do trailer do filme T:OMS onde o ator Tom Hiddleston é reconhecido com 31% de probabilidade.

neste trabalho por limitação de recursos computacionais. Mas acreditamos que utilizando esta técnica é possível alcançar melhores resultados.

6.1.4 Detecção de pessoas desconhecidas

Os resultados obtidos mostram em forma de uma lista a grande quantidade de pessoas reconhecidas nos trailers. A maioria dessas pessoas foram detectadas em faces de atores que não estão presentes na base de treinamento da RNC. Na solução atual não é possível saber quais destas faces estão presentes na base de treino, portanto não há como apontar quais faces não são conhecidas pela RNC. Com o intuito de aprimorar os resultados, uma possível extensão deste trabalho seria um mecanismo capaz de apontar as faces que não estão presentes na base de treino.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Bengio, I. G. Y.; Courville, A., “Deep learning”, 2016, book in preparation for MIT Press, Capturado em: http://www.deeplearningbook.org.

[2] Bradski, G.; et al.. “The opencv library”, *Doctor Dobbs Journal*, vol. 25–11, 2000, pp. 120–126.

[3] De Carrera, P. F.; Marques, I. “Face recognition algorithms”, *Master’s thesis in Computer Science, Universidad Euskal Herriko*, 2010.

[4] Gregor, K.; LeCun, Y. “Emergence of complex-like cells in a temporal product network with local receptive fields”, *arXiv preprint arXiv:1006.0448*, 2010.

[5] Hanjalic, A.; Xu, L.-Q. “Affective video content representation and modeling”, *Multimedia, IEEE Transactions on*, vol. 7–1, 2005, pp. 143–154.

[6] Hassner, T.; Harel, S.; Paz, E.; Enbar, R. “Effective face frontalization in unconstrained images”. In: IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2015.

[7] Huang, G. B.; Ramesh, M.; Berg, T.; Learned-Miller, E. “Labeled faces in the wild: A database for studying face recognition in unconstrained environments”, Relatório Técnico 07-49, University of Massachusetts, Amherst, 2007.

[8] Jia, Y.; Shelhamer, E.; Donahue, J.; Karayev, S.; Long, J.; Girshick, R.; Guadarrama, S.; Darrell, T. “Caffe: Convolutional architecture for fast feature embedding”, *arXiv preprint arXiv:1408.5093*, 2014.

[9] Kasinski, A.; Florek, A.; Schmidt, A. “The put face database”, *Image Processing and Communications*, vol. 13–3-4, 2008, pp. 59–64.

[10] King, D. E. “Dlib-ml: A machine learning toolkit”, *Journal of Machine Learning Research*, vol. 10, 2009, pp. 1755–1758.

[11] Li, F.-F.; Karpathy, A.; Johnson, J. “Lecture notes in convolutional neural networks for visual recognition”, July 2016.

[12] Li, H.; Hua, G.; Lin, Z.; Brandt, J.; Yang, J. “Probabilistic elastic part model for unsupervised face detector adaptation”. In: Computer Vision (ICCV), 2013 IEEE International Conference on, 2013, pp. 793–800.

[13] Li, H.; Hua, G.; Shen, X.; Lin, Z.; Brandt, J. “Eigen-pep for video face recognition”. In: *Computer Vision–ACCV 2014*, Springer, 2015, pp. 17–33.

[14] Marsland, S. “Machine learning: an algorithmic perspective”. CRC press, 2014.

[15] Mitchell, T. M. “Machine Learning”. Boston, MA: WCB/McGraw-Hill, 1997.

[16] Nielsen, T. A. “Neural Networks and Deep Learning”. Determination Press, 2015.

[17] Ortiz, E. G.; Wright, A.; Shah, M. “Face recognition in movie trailers via mean sequence sparse representation-based classification”. In: Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on, 2013, pp. 3531–3538.

[18] Parkhi, O. M.; Vedaldi, A.; Zisserman, A. “Deep face recognition”. In: British Machine Vision Conference, 2015.

[19] Rosenblatt, F. “The perceptron: a probabilistic model for information storage and organization in the brain.”, *Psychological review*, vol. 65–6, 1958, pp. 386.

[20] Srivastava, N.; Hinton, G. E.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. “Dropout: a simple way to prevent neural networks from overfitting.”, *Journal of Machine Learning Research*, vol. 15–1, 2014, pp. 1929–1958.

[21] Taigman, Y.; Yang, M.; Ranzato, M.; Wolf, L. “Deepface: Closing the gap to human-level performance in face verification”. In: Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on, 2014, pp. 1701–1708.

[22] Tan, P.-N.; Steinbach, M.; Kumar, V. “Introduction to Data Mining”. Pearson, 2006.

[23] Teddlie, C.; Yu, F. “Mixed methods sampling a typology with examples”, *Journal of mixed methods research*, vol. 1–1, 2007, pp. 77–100.

[24] Wasserman, P. D. “Neural computing”. Van Nostrand Reinhold, New York, 1989.

[25] Williams, D.; Hinton, G. “Learning representations by back-propagating errors”, *Nature*, vol. 323, 1986, pp. 533–536.

[26] Wolf, L.; Hassner, T.; Maoz, I. “Face recognition in unconstrained videos with matched background similarity”. In: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, 2011, pp. 529–534.

[27] Zeiler, M. D. “Adadelta: an adaptive learning rate method”, *arXiv preprint arXiv:1212.5701*, 2012.

APÊNDICE A – ATORES ENCONTRADOS NO TRAILER DO FILME T:OMS

Tabela A.1: Atores encontrados no trailer T:OMS.

Rótulo	Aparições	Probabilidade Média
Aidan_Turner	8	$0,0385748276021$
Alexandra_Chando	7	$0,00552878162957$
Alexis_Dziena	1	$0,0773274675012$
Andrew_Dice_Clay	2	$0,00845994846895$
Andrew_Keegan	1	$0,0114924637601$
Andrew_Scott	1	$0,032498229295$
Andy_Milonakis	1	$0,00721820676699$
Anson_Mount	1	$0,00477227848023$
Anthony_Anderson	4	$0,0151446962263$
Antonia_Campbell-Hughes	1	$0,0120168719441$
Arian_Foster	2	$0,00349043798633$
Barry_Watson	1	$0,00990300346166$
Billy_Campbell	1	$0,0129614649341$
Bridget_Regan	20	$0,0533360279631$
Cameron_Bright	1	$0,0124130928889$
Catalina_Sandino_Moreno	1	$0,028054183349$
Chazz_Palminteri	13	$0,0111517326262$
Chris_Hemsworth	**29**	$0,0674683553135$
Christian_Kane	30	$0,0207812801314$
Claudia_Black	41	$0,0324909600213$
Connor_Trinneer	2	$0,0111121858936$
Danica_McKellar	6	$0,00812825428632$
Della_Reese	1	$0,0290944520384$
Desmond_Harrington	2	$0,0155278011225$
Diogo_Morgado	3	$0,0340028780823$
Dominic_Keating	7	$0,0188569728551$
Eddie_Vedder	1	$0,00264345156029$
Eion_Bailey	2	$0,0156845417805$
Electra_Avellan	4	$0,0144390282221$
Emily_Watson	1	$0,013458433561$
Enrique_Murciano	3	$0,0207025315613$
Eric_Schweig	17	$0,0189553819706$

Continua na próxima página

Tabela A.1 – *Continuada da página anterior*

Rótulo	Aparições	Probabilidade Média
Eva_Green	2	$0,0111853508279$
Fanny_Ardant	14	$0,232742577791$
Fiona_Shaw	1	$0,0242709349841$
Francesca_Capaldi	1	$0,0694678649306$
Francesca_Neri	31	$0,0973991986154$
Fred_Armisen	1	$0,0047928635031$
Gabourey_Sidibe	2	$0,0192021278199$
Garrett_Hedlund	3	$0,0219562345495$
Gates_McFadden	2	$0,0515106590465$
Gemma_Arterton	3	$0,18114917477$
Gina_Carano	1	$0,059044085443$
Gina_McKee	2	$0,0120700788684$
Giovanni_Ribisi	1	$0,00483872182667$
Grace_Zabriskie	1	$0,00815163832158$
Hattie_Morahan	2	$0,0195230506361$
Holly_Marie_Combs	1	$0,0229053478688$
India_Eisley	1	$0,0143817765638$
Indira_Varma	7	$0,00859780330211$
Jai_Courtney	2	$0,00495882495306$
James_Corden	1	$0,00776275387034$
James_Lafferty	8	$0,0907073336421$
James_Roday	2	$0,0927633568645$
Jane_Badler	3	$0,00445983434717$
Janeane_Garofalo	14	$0,0156671998557$
Janet_McTeer	1	$0,0316069945693$
Janet_Montgomery	1	$0,0578650049865$
Jason_David_Frank	1	$0,02002408728$
Jason_Mantzoukas	41	$0,0337063232437$
Jason_Momoa	1	$0,0209690593183$
Jeff_Conaway	1	$0,00859202910215$
Jemaine_Clement	77	$0,0098259470988$
Jo_Wilfried_Tsonga	3	$0,0712454244494$
Joanne_Kelly	48	$0,0897648894849$
Joe_Lando	3	$0,00685362005606$
John_Ridley	1	$0,00546576688066$
John_de_Lancie	1	$0,0255554113537$
Jonathan_Frakes	3	$0,0211638531958$
Jordana_Brewster	5	$0,0102367150597$

Continua na próxima página

Tabela A.1 – *Continuada da página anterior*

Rótulo	Aparições	Probabilidade Média
Jorge_Garcia	1	$0,00681312382221$
Joseph_Lawrence	1	$0,00547545868903$
Justin_Chatwin	2	$0,0240533361211$
Karolina_Wydra	1	$0,00470311334357$
Katey_Sagal	1	$0,00349382823333$
Katie_Lee_Joel	2	$0,00395187630784$
Kevin_Hart	6	$0,0134449368343$
Kristen_Stewart	1	$0,0142047535628$
Kyle_Richards	1	$0,0134419081733$
Lalit_Modi	1	$0,0040415902622$
Lee_Tergesen	2	$0,00622090476099$
Leonor_Varela	1	$0,00627755746245$
Linda_Fiorentino	56	$0,0467951612622$
Lola_Glaudini	1	$0,0261983182281$
Louis_Gossett_Jr	1	$0,00386952981353$
Luke_Arnold	3	$0,021385038582$
Marina_Sirtis	1	$0,030810970813$
Mark_Boone_Junior	9	$0,0140646305453$
Martin_Henderson	7	$0,00709190000115$
Mary_Jo_Deschanel	2	$0,0143340067007$
Maxim_Knight	1	$0,0226326342672$
Meat_Loaf	1	$0,0165409855545$
Michelle_Hurd	1	$0,00421191845089$
Michelle_Money	1	$0,00370119279251$
Mindy_Kaling	2	$0,00781850540079$
Mindy_Sterling	1	$0,0157744791359$
Morgan_Lily	2	$0,0173180620186$
Nana_Visitor	1	$0,0149686336517$
Natalie_Portman	**9**	$0,207733349668$
Nestor_Carbonell	2	$0,0280956393108$
Nicholas_Lea	2	$0,00829939870164$
Nick_Cave	1	$0,0185727626085$
Noel_Fielding	1	$0,0189963318408$
Oliver_Platt	3	$0,00379044702277$
Olivia_Palermo	1	$0,00578967714682$
Omari_Hardwick	4	$0,0403458629735$
Pamela_Adlon	1	$0,0115391202271$
Patrick_Swayze	1	$0,00859029218554$

Continua na próxima página

Tabela A.1 – *Continuada da página anterior*

Rótulo	Aparições	Probabilidade Média
Paul_Dano	2	$0,0298694977537$
Randeep_Hooda	12	$0,0391808929853$
Rekha	1	$0,0433166250587$
Rick_Springfield	1	$0,0140685196966$
Robbie_Amell	1	$0,00978896394372$
Ron_Eldard	1	$0,00696738949046$
Russell_Tovey	2	$0,0135478149168$
Rutina_Wesley	27	$0,116429746289$
Sandra_Bernhard	1	$0,00614366820082$
Scott_Porter	1	$0,0119185075164$
Sean_Patrick_Flanery	2	$0,00721289636567$
Serinda_Swan	1	$0,00372867914848$
Shannen_Doherty	1	$0,0122981797904$
Shawn_Ashmore	1	$0,0571923963726$
Shiri_Appleby	4	$0,0459278309718$
Sidse_Babett_Knudsen	6	$0,0176016669332$
Steven_Culp	3	$0,00732051177571$
Thomas_Ian_Nicholas	1	$0,00541586335748$
Thomas_McDonell	18	$0,0453802273454$
Tinsel_Korey	1	$0,0249726381153$
Tom_Hiddleston	**26**	$0,116456082879$
Tyler_Labine	19	$0,0331305374489$
Valeria_Golino	2	$0,332698464394$
Vincent_Pastore	2	$0,00541312294081$
Virginie_Ledoyen	3	$0,124865487218$
Wendell_Pierce	2	$0,00683089834638$
Willem_Dafoe	3	$0,093483996888$

APÊNDICE B – ATORES ENCONTRADOS NO TRAILER DO FILME (500) DIAS COM ELA

Tabela B.1: Atores encontrados no trailer de (500) Dias Com Ela.

Rótulo	Aparições	Probabilidade Média
Adelaide_Kane	1	$0,0202270299196$
Adrienne_Barbeau	2	$0,0119382590055$
Aidan_Turner	3	$0,018145116667$
Alessandra_Torresani	2	$0,0912762954831$
Alex_Borstein	11	$0,0169679423879$
Alexander_Gould	1	$0,0310611817986$
Alexandra_Breckenridge	1	$0,0332369171083$
Alexia_Fast	1	$0,00221417285502$
Alice_Cooper	1	$0,0121802259237$
Ana_Mulvoy-Ten	1	$0,0796466171741$
Andy_Milonakis	1	$0,00332913501188$
Angel_Coulby	1	$0,0159216299653$
Anson_Mount	3	$0,018097746186$
Arian_Foster	4	$0,00316903449129$
Avan_Jogia	5	$0,023637778312$
Beau_Mirchoff	4	$0,0117092859$
Ben_Miller	1	$0,0057112518698$
Ben_Whishaw	1	$0,00900549441576$
Billy_Campbell	7	$0,00334630879973$
Brendan_Hines	2	$0,00631813379005$
Bret_McKenzie	2	$0,00656611472368$
Bridget_Regan	15	$0,0906364078323$
Cameron_Boyce	2	$0,0118377204053$
Cameron_Bright	7	$0,0242722834061$
Carter_Jenkins	5	$0,0750751573592$
Caterina_Murino	19	$0,042719800785$
Charlotte_Gainsbourg	64	$0,0258455328221$
Charlyne_Yi	3	$0,0097365429004$
Chazz_Palminteri	5	$0,00459025800228$
Chris_Brown	2	$0,00472496496513$
Chris_Colfer	2	$0,0438693445176$

Continua na próxima página

Tabela B.1 – *Continuada da página anterior*

Rótulo	Aparições	Probabilidade Média
Chris_Marquette	23	$0,0189142649102$
Chris_Riggi	2	$0,0189199538436$
Christian_Kane	8	$0,016273248475$
Clark_Duke	12	$0,0277398857288$
Clark_Gregg	1	$0,00709469150752$
Claudia_Black	87	$0,0354380867465$
Clea_DuVall	1	$0,0505976080894$
Corey_Sevier	2	$0,0229694629088$
Dane_Cook	1	$0,0418217107654$
Daphne_Zuniga	2	$0,225192122161$
David_James_Elliott	1	$0,0128971384838$
David_Kross	8	$0,055669257883$
David_Morrissey	35	$0,0502965047423$
David_Thewlis	1	$0,00558195076883$
Devon_Bostick	3	$0,0411981455982$
Dexter_Fletcher	2	$0,00621016568039$
Diane_Farr	2	$0,0160478530452$
Donald_Glover	1	$0,0171535778791$
Dylan_Moran	1	$0,0258698258549$
Eartha_Kitt	2	$0,00997427967377$
Eion_Bailey	6	$0,0632804332611$
Elaine_Cassidy	1	$0,0389007814229$
Eli_Wallach	2	$0,00687454990111$
Ellen_Wong	3	$0,0597394468884$
Elodie_Yung	42	$0,0525154323938$
Emily_Watson	1	$0,068795979023$
Enrique_Murciano	4	$0,0203957336489$
Eoin_Macken	3	$0,0174243273214$
Eric_Schweig	7	$0,0132481179732$
Eva_Green	3	$0,0347832804546$
Fanny_Ardant	1	$0,0345725864172$
Fernanda_Romero	1	$0,0192942321301$
Francesca_Neri	17	$0,0308350504321$
Freddie_Highmore	3	$0,0577677090963$
Gabourey_Sidibe	28	$0,0255096804384$
Gabriel_Macht	1	$0,0212531033903$
Gbenga_Akinnagbe	7	$0,0214700344285$
Gemma_Arterton	39	$0,0767448831541$

Continua na próxima página

Tabela B.1 – *Continuada da página anterior*

Rótulo	Aparições	Probabilidade Média
Gina_Carano	1	$0,0289894789457$
Gina_McKee	12	$0,0449817782501$
Giuseppe_Tornatore	1	$0,00505365757272$
Gloria_Macapagal-Arroyo	11	$0,0105343509296$
Harry_Treadaway	2	$0,00303793500643$
Harvey_Keitel	3	$0,00631105853245$
Heather_Hemmens	1	$0,00765694584697$
Hiroyuki_Sanada	3	$0,0256634407366$
Hutch_Dano	3	$0,0315760994951$
India_Eisley	3	$0,0244280767317$
Indira_Varma	4	$0,00661760801449$
Ivana_Baquero	3	$0,00899720719705$
Jack_Lemmon	13	$0,00336885276752$
Jacqueline_MacInnes_Wood	1	$0,0172153823078$
James_Anderson	7	$0,00829666876234$
James_Lafferty	14	$0,0666272456625$
James_Lipton	2	$0,00774524896406$
James_Nesbitt	2	$0,00604671612382$
Jane_Badler	4	$0,0432247680146$
Jane_Campion	2	$0,0054969759658$
Janeane_Garofalo	12	$0,0243929433636$
Janice_Dickinson	2	$0,0220147129148$
Jared_Gilmore	26	$0,112032216257$
Jasmine_Waltz	5	$0,00336409986485$
Jason_Behr	1	$0,0385067686439$
Jason_David_Frank	1	$0,0217234063894$
Jason_Dolley	1	$0,0694224387407$
Jason_Gedrick	4	$0,00264495430747$
Jason_George	17	$0,00674294726923$
Jason_Mantzoukas	26	$0,0679336010407$
Jeffrey_Hunter	1	$0,00558574358001$
Jemaine_Clement	60	$0,0123191182696$
Jemima_Rooper	1	$0,0153910815716$
Jennifer_Beals	2	$0,0288531007245$
Jeremy_Sisto	3	$0,00428995210677$
Jessica_De_Gouw	2	$0,0166208297014$
Jessie_J	58	$0,0514741097416$
Jet_Li	1	$0,0247218739241$

Continua na próxima página

Tabela B.1 – *Continuada da página anterior*

Rótulo	Aparições	Probabilidade Média
Jim_Caviezel	1	$0,00551403174177$
Jim_Sturgess	2	$0,0186793766916$
Jo_Wilfried_Tsonga	4	$0,0301534235477$
Joanne_Kelly	49	$0,0284307669962$
Joe_Flanigan	1	$0,01101388596$
Joel_Courtney	2	$0,00886759068817$
John_Cho	1	$0,00554315093905$
John_Gallagher_Jr	2	$0,0331648401916$
John_Hensley	8	$0,00451445754152$
Johnny_Whitworth	9	$0,00281764691075$
Jonathan_Frakes	1	$0,00336029427126$
Jordana_Brewster	11	$0,0315295015038$
Josh_Gad	3	$0,00475994955438$
Josh_Holloway	1	$0,031763151288$
Joyce_DeWitt	12	$0,0289570327538$
Joyce_Giraud	11	$0,0121987773613$
Justin_Chatwin	19	$0,0118407730846$
Kali_Hawk	4	$0,0173212261871$
Karla_Souza	2	$0,0107989651151$
Kate_Voegele	7	$0,00877280167437$
Kathryn_Newton	1	$0,00831752642989$
Katie_Findlay	5	$0,0164456751198$
Katie_Lowes	1	$0,00581631995738$
Keegan_Connor_Tracy	1	$0,00659427139908$
Kel_Mitchell	2	$0,0103777409531$
Kellie_Martin	2	$0,0120475105941$
Kevin_Hart	1	$0,0151064451784$
Kevin_McHale	1	$0,0044826567173$
Kit_Harington	6	$0,0118149190675$
Kristen_Stewart	1	$0,0183109622449$
Kyle_MacLachlan	1	$0,0141533808783$
Lavell_Crawford	1	$0,0101513406262$
Leonor_Varela	4	$0,0100656119175$
Liam_Aiken	4	$0,0531881218776$
Linda_Fiorentino	9	$0,0143535344137$
Liya_Kebede	3	$0,00584916848068$
Logan_Marshall-Green	2	$0,0159640875645$
Lola_Glaudini	1	$0,0180624388158$

Continua na próxima página

Tabela B.1 – *Continuada da página anterior*

Rótulo	Aparições	Probabilidade Média
Lou_Diamond_Phillips	7	$0,0108530265944$
Louise_Brealey	1	$0,032007817179$
Luke_Arnold	3	$0,0075002744173$
Marie_Osmond	1	$0,0380060523748$
Marina_Sirtis	8	$0,0169904346112$
Marion_Cotillard	1	$0,0783008337021$
Marla_Sokoloff	6	$0,0514595502367$
Martha_Higareda	13	$0,0443814318054$
Martine_McCutcheon	1	$0,012807443738$
Mary_Jo_Deschanel	6	$0,0734036844224$
Mathieu_Amalric	1	$0,0124072153121$
Mathilda_May	20	$0,04553764835$
Matt_Lanter	1	$0,0265191048384$
Matthew_Goode	1	$0,00310172513127$
Matthew_Underwood	1	$0,00661930814385$
Michael_C,_Hall	1	$0,0162250623107$
Michael_Trucco	3	$0,00289784317526$
Michele_Lee	2	$0,0375904543325$
Michelle_Krusiec	8	$0,00639767118264$
Mindy_Kaling	1	$0,00732323061675$
Mindy_Sterling	1	$0,0046435049735$
Mira_Furlan	3	$0,0263092654447$
Natalia_Tena	1	$0,00267822039314$
Nathalie_Kelley	1	$0,0853312686086$
Neil_deGrasse_Tyson	2	$0,00987443653867$
Nicholas_Lea	1	$0,00426140101627$
Nick_Cave	2	$0,00982844620012$
Noel_Fielding	2	$0,011262874119$
Olga_Kurylenko	1	$0,00970687158406$
Padma_Lakshmi	1	$0,0311778634787$
Pam_Dawber	2	$0,00450121215545$
Pamela_Adlon	1	$0,00938576646149$
Pat_Monahan	2	$0,00715303327888$
Patricia_Quinn	1	$0,0300516933203$
Patrick_Swayze	37	$0,0536912908224$
Paul_Feig	1	$0,00410454673693$
Paz_de_la_Huerta	1	$0,00804739259183$
Penn_Jillette	1	$0,00939070247114$

Continua na próxima página

Tabela B.1 – *Continuada da página anterior*

Rótulo	Aparições	Probabilidade Média
Perrey_Reeves	2	$0,0436987672001$
Peter_Berg	1	$0,00483508175239$
Rade_Serbedzija	1	$0,0103280069306$
Randeep_Hooda	19	$0,07440624527$
Raul_Julia	1	$0,00817508995533$
Rebecca_Da_Costa	1	$0,0160268694162$
Richard_Grieco	1	$0,0548089072108$
Richard_Jenkins	2	$0,00908089149743$
Rick_Springfield	1	$0,00605585193262$
Riz_Ahmed	4	$0,00700512412004$
Robert_Carlyle	2	$0,00942579237744$
Robin_Shou	179	$0,0349732417322$
Ron_Jeremy	2	$0,0213998174295$
Ruth_Jones	4	$0,0322903086781$
Rutina_Wesley	23	$0,0778300554856$
Sally_Hawkins	3	$0,0129636150474$
Sami_Gayle	1	$0,0151259750128$
Sarah_Palin	2	$0,0273107704706$
Scott_Elrod	1	$0,00291079981253$
Sean_Kanan	1	$0,00264555937611$
Sean_Pertwee	1	$0,0062362072058$
Selena	6	$0,0482764362047$
Selma_Blair	355	$0,0603371510203$
Senta_Berger	11	$0,07111258754$
Shane_West	4	$0,00452224526089$
Sharon_Rooney	1	$0,0237189102918$
Sibel_Kekilli	28	$0,0408178280507$
Sidse_Babett_Knudsen	13	$0,0355067743132$
Silas_Weir_Mitchell	3	$0,00598573482906$
Skandar_Keynes	5	$0,0386453192681$
Stepfanie_Kramer	2	$0,010052235797$
Stephen_Colbert	5	$0,0108899846673$
Stephen_Lee	20	$0,00687407387886$
Stephen_Merchant	2	$0,0195419276133$
Stephen_Root	1	$0,0252495370805$
Sterling_Beaumon	1	$0,0218834318221$
Steve_Harris	2	$0,00991919450462$
Steven_Culp	8	$0,0107246786938$

Continua na próxima página

Tabela B.1 – *Continuada da página anterior*

Rótulo	Aparições	Probabilidade Média
Sung_Kang	32	$0,0629754365655$
Tamara_Feldman	9	$0,0417308372756$
Tammy_Blanchard	1	$0,0129097849131$
Teala_Dunn	1	$0,0535300858319$
Thomas_McDonell	123	$0,02584406983$
Tiffany_Hines	1	$0,00884009338915$
Timothy_Spall	1	$0,00761776510626$
Tinsel_Korey	14	$0,0349184735013$
Tyler_Labine	33	$0,0300530124823$
Usher_Raymond	1	$0,0107239503413$
Valdis_Dombrovskis	1	$0,00441615143791$
Vincent_Spano	1	$0,00549314264208$
Virginie_Ledoyen	5	$0,0279995668679$
Wagner_Moura	1	$0,00347184762359$
Wayne_Knight	6	$0,00805019610561$
Wesley_Snipes	12	$0,0123095102608$
Whitney_Cummings	18	$0,0319617629155$
Will_Yun_Lee	19	$0,0414341387192$
William_Moseley	13	$0,0694286957956$
Zoe_McLellan	6	$0,0491620191994$
Zooey_Deschanel	**2**	$0,0193586889654$
Zulay_Henao	2	$0,0145440467168$

APÊNDICE C – ATORES ENCONTRADOS NO TRAILER DO FILME A REDE SOCIAL

Tabela C.1: Atores encontrados no trailer do filme a Rede Social.

Rótulo	Aparições	Probabilidade Média
Adelaide_Kane	1	$0,0176853183657$
Adina_Porter	30	$0,0135571520155$
Adrian_Grenier	1	$0,0066169379279$
Aidan_Turner	13	$0,0277995735837$
Alice_Cooper	4	$0,00504615285899$
Ann-Margret	1	$0,00370395928621$
Anthony_Anderson	2	$0,00609878869727$
Anthony_Rapp	15	$0,00320253909255$
Ariana_Richards	1	$0,00566682126373$
Barry_Watson	1	$0,0161744635552$
Ben_Stiller	1	$0,00928341969848$
Bridget_Regan	15	$0,0837809650227$
Carrie_Fisher	1	$0,00941138435155$
Cherie_Lunghi	1	$0,00413332041353$
Chris_Riggi	3	$0,00643871103724$
Christian_Kane	16	$0,0458426327095$
Claudia_Black	340	$0,0691090609454$
David_James_Elliott	1	$0,0246168524027$
David_Morrissey	2	$0,0147905629128$
Debra_Jo_Rupp	3	$0,00808579878261$
Dirk_Benedict	2	$0,00668721739203$
Dylan_Neal	1	$0,00665275007486$
Eddie_Vedder	2	$0,0175815040711$
Edgar_Wright	1	$0,0296191871166$
Eion_Bailey	32	$0,0414743039873$
Elisabetta_Canalis	1	$0,0245323292911$
Eoin_Macken	5	$0,00769955562428$
Eric_Schweig	25	$0,0251174960285$
Erik_Per_Sullivan	3	$0,0174460802227$
Fanny_Ardant	1	$0,01028691791$
Francesca_Neri	17	$0,0194787905716$

Continua na próxima página

Tabela C.1 – *Continuada da página anterior*

Rótulo	Aparições	Probabilidade Média
Gabourey_Sidibe	1	$0,0114047862589$
Gates_McFadden	1	$0,0308202914894$
Gemma_Arterton	2	$0,0205278601497$
Grace_Zabriskie	3	$0,00957628836234$
Holly_Marie_Combs	3	$0,0223400816321$
India_Eisley	7	$0,0231636068118$
Indira_Varma	7	$0,0271030430948$
Jamel_Debbouze	2	$0,0175232551992$
James_Anderson	1	$0,00331069971435$
Jane_Badler	4	$0,0237232381478$
Jason_David_Frank	3	$0,0138221199935$
Jason_Mantzoukas	101	$0,053487334488$
Jeff_Fahey	1	$0,00558784091845$
Jeffrey_Hunter	3	$0,00398775197876$
Jennifer_Ferrin	1	$0,00308315246366$
Jeremy_Shada	11	$0,0991364900361$
Jesse_Eisenberg	**30**	$0,0552454790721$
Joanne_Kelly	36	$0,0597686043216$
Joe_Lando	12	$0,00926207840287$
John_Ridley	3	$0,00345889641903$
Jonathan_Bennett	7	$0,00594750572262$
Jonathan_Frakes	6	$0,00333902138906$
Jonathan_Jackson	1	$0,00995425041765$
Jonathan_Taylor_Thomas	2	$0,012632206548$
Jorja_Fox	1	$0,0602771453559$
Juliet_Stevenson	1	$0,00919231027365$
Kali_Hawk	1	$0,0142124863341$
Kaya_Scodelario	2	$0,090128261596$
Kevin_Hart	1	$0,0261581502855$
Kit_Harington	18	$0,0456204289157$
Lainie_Kazan	2	$0,015599087812$
Lee_Remick	1	$0,0153293032199$
Leonor_Varela	5	$0,0734794043005$
Linda_Fiorentino	4	$0,0518504949287$
Lola_Glaudini	9	$0,0160448766934$
Lolo_Jones	5	$0,0133062742651$
Louise_Brealey	11	$0,0127487917515$
Luke_Arnold	1	$0,0258689466864$

Continua na próxima página

Tabela C.1 – *Continuada da página anterior*

Rótulo	Aparições	Probabilidade Média
Luke_Mitchell	8	$0,00324242023635$
Marc_Blucas	34	$0,0397459656986$
Margarita_Levieva	4	$0,00329776457511$
Mariel_Hemingway	3	$0,00658637595673$
Marine_Vacth	17	$0,0139137597426$
Mark_Ruffalo	1	$0,0248987097293$
Martha_Higareda	16	$0,256906832568$
Mary_Stuart_Masterson	1	$0,0107726911083$
Mathilda_May	3	$0,0187313420077$
Michael_Copon	10	$0,0647151622921$
Michelle_Krusiec	7	$0,0417219105044$
Mira_Furlan	9	$0,0107378510551$
Nana_Visitor	11	$0,0195420524952$
Nancy_Allen	13	$0,0140145673918$
Natassia_Malthe	1	$0,0345480404794$
Nicholas_Lea	51	$0,0255363929706$
Pat_Monahan	3	$0,00465234516499$
Paul_Gross	1	$0,00248475885019$
Peter_Berg	1	$0,00277947564609$
Peter_MacNicol	25	$0,0041280395817$
Richard_Grieco	1	$0,00456815306097$
Robin_Shou	9	$0,0114585728022$
Rutina_Wesley	96	$0,0668948395614$
Sam_Elliott	1	$0,00888779573143$
Samuel_Larsen	1	$0,0168636143208$
Sarah_Palin	1	$0,0050125089474$
Sarah_Wayne_Callies	1	$0,116351924837$
Sasha_Barrese	20	$0,0348077641334$
Shaun_Cassidy	1	$0,00852183811367$
Sidse_Babett_Knudsen	2	$0,0462945485488$
Stephen_Lee	3	$0,00562249341359$
Steve_Burton	5	$0,0214836749248$
Steven_Culp	2	$0,0259680105373$
Terence_Stamp	6	$0,00486836202132$
Thomas_McDonell	40	$0,0331290339353$
Tim_Guinee	2	$0,0102788843215$
Timothy_V,_Murphy	2	$0,00732819316909$
Tyler_Labine	31	$0,0277253442534$

Continua na próxima página

Tabela C.1 – *Continuada da página anterior*

Rótulo	**Aparições**	**Probabilidade Média**
Tyrone_Power	1	$0,00324647990055$
Tyson_Beckford	1	$0,00432960968465$
Viggo_Mortensen	2	$0,0122152534313$
Virginie_Ledoyen	5	$0,0185492811725$
Wagner_Moura	1	$0,00442496873438$
Walton_Goggins	2	$0,0060217557475$
Wayne_Knight	1	$0,00521295238286$
Wesley_Snipes	33	$0,0417253419644$