Problema de Detecção Facial utilizando RCNN

Rodolfo Labiapari Mansur Guimarães
Departamento de Computação
Universidade Federal de Ouro Preto - UFOP
Ouro Preto - MG – Brasil
rodolfolabiapari@decom.ufop.br

***Abstract*—In this article, a face detector implementation is presented using a neural network architecture using the *neon* tool, developed to detect robustly face patterns. The use of neural network to detect objects is necessary because of the high difficulty of computational models to be successful in detecting objects in images where there are complex environments and variations of the object as the faces. The implemented system automatically synthesizes the problem-specific for extracting features from a standard set of faces and nonfaces, without any human interference to extract and create patterns. The procedure works as a pipeline convolutions and subsampling, eliminating data that is not relevant for processing. Techniques that use convolutional neural network type architecture are emphasizing in the academic environment, because it is a procedure that obtains results with good accuracy and without having great losses with the respective trade-off and can even be used for classifications in low-power processing environments such as personal computers and even embedded systems.**

***Keywords*-Face Detection; Convolutional Neural Network; API *neon* Intel Nervana.**

***Resumo*—Neste artigo, é apresentado uma implementação de detector de face utilizando uma arquitetura de rede neural convolucional por meio da ferramenta *neon*, desenvolvido para detectar padrões de faces de forma robusta. O uso de rede neural para detecção de objetos é necessário pela grande dificuldade de modelos computacionais terem sucesso na detecção de objetos em imagens onde existam ambientes complexos e variações do objeto como as faces. O sistema implementado sintetiza automaticamente o problema específico para extração de características de um conjunto padrão de faces e não-faces, sem qualquer interferência humana para extrair e criar padrões. O procedimento funciona como um *pipeline* convoluções e subamostragens eliminando dados que não são relevantes para processamento. Técnicas que utilizam arquitetura tipo *convolutional neural network* estão destacando no ambiente acadêmico, por ser um procedimento que obtêm resultados com boa acurácia e sem ter grandes perdas com o respectivo *trade-off* podendo até ser usado para classificações em ambientes de baixo poder de processamento como computadores pessoais e até sistemas embarcados.**

***Keywords*-Detecção de Face; Rede Neural Convolucional; API *neon* Intel Nervana.**

I. Introdução

Detecção de face é um problema pertencente à visão computacional no qual, métodos têm como propósito a detecção de faces.

Esse problema se torna difícil quando não se tem imagens com ambientes controlados tal como um fundo controlado, ou seja, com um *background* que não respeite um padrão pré-estabelecido, com obstrução facial como óculos, barba, etc [1].

Como o estado da arte é concreto em detecção de face em ambientes controlados, pesquisas nas últimas décadas partem a buscar métodos que focam principalmente na detecção de face em ambientes onde existe maior complexidade, sem perder a eficiência já conseguida até então. As pesquisas em detecção de face hoje procuram métodos eficientes e rápidos onde as faces podem se apresentar não só posicionada em perfil mas sim, com distâncias, angulações e rotações variadas, oclusões,níveis extremos de iluminação, entre outras variações que poderiam representar situações que representassem o mundo real na captura de imagens comuns [2].

Deep convolutional neural network também conhecida como *convolutional neural network* (CNN) é uma arquitetura de rede neural que foi introduzida com grande sucesso por Le Cun et al [3] [4] [5] que é baseada em rede neural, apresentada por McCulloch [6]. São bioinspirados em rede neural multicamada hierárquica que combinam três técnicas arquitetônicas para assegurar graus de invariância de mudança, escala e distorção com procedimentos de campos receptivos locais, pesos compartilhados e sub-amostragem (*subsampling*) espacial. Com isso, diferentes arquiteturas baseadas na rede neural convolucional tem sido utilizadas com sucesso pela literatura em inúmeras aplicações com grau de dificuldade elevada como reconhecimento de escrita à mão [4], reconhecimento de caracteres de máquinas de escrever [7] [8] e reconhecimento de faces [9].

Vaillant et al. [10] usou uma rede convolucional para detecção de objetos baseado em imagens e considerou o caso como um problema o de detecção de face. Garcia et al. [1], em seu trabalho, propõe uma abordagem diferente de reconhecimento de faces alterando as disposições e configurações das camadas da rede neural. Nomeiam seu algoritmo como *Convolutional Face Finder* no qual consiste de num algoritmo de rede neural composto por um *pipeline* de camadas convolucionais. O algoritmo apresentado por eles possui seis camadas, com exceção da camada de recebimento da imagem de entrada. As quatro primeiras camadas são um grupo chamado de mapa de características onde é realizado uma convolução e *subsampling* na imagem, no qual é repetido por duas vezes. Após o mapas de características, há uma camada de rede neural totalmente conectada com o propósito de definir de fato o resultado dos cálculos feitos pelas redes convolucionais, que no caso é um problema de decisão.

O trabalho conta também com o uso de uma API para a confecção do trabalho onde utilizou-se da API *neon* da Intel Nervana.

O artigo está disposto da seguinte forma: Seção II define justificativas para a realização deste trabalho e um referencial teórico abordando o funcionamento da ferramenta para cumprir seu propósito na Seção III. Na Seção IV é descrito em detalhes o processo de reconhecimento de face utilizando arquitetura *convolutional neural network* enquanto na Seção V é feita as considerações e na Seção VI é descrito os resultados obtidos. Na Seção VII é feito conclusões e propostas para futuras pesquisas.

II. Justificativa

Sistemas computacionais anteriores à redes neurais possuíam características bastante ineficientes para imagens que tinham como propriedade fundos não-padronizados [1].

Considerando o sucesso de algoritmos que utilizam *convolutional neural network* em reconhecimento de padrão em tarefas, como será descrito na Seção III, a literatura afirma que é possível ter bons resultados utilizando tal API para reconhecer faces faces, em imagens não padronizadas. E como a API não é *ad-hoc*, é possível utilizá-la em vários outros conjuntos de problemas criando assim um algoritmo robusto para detecção de padrões [2].

III. Referencial Teórico

A. Convolutional Neural Network

O algoritmo baseia-se em vários pequenos processos. São eles a *Convolution*, *Subsampling* e a mescla dos dois processos formando o maior chamado de mapa de características e por fim uma rede totalmente conectada para gerar a decisão final [1].

O primeiro passo do mapa de características é a extração dessas. A primeira camada realiza uma convolução de um filtro gerado pelo algoritmo sobre a imagem a fim de encontrar características específicas da imagem. Com o resultado da convolução, para dar continuidade ao algoritmo e iniciar o próximo procedimento deve-se realizar o processo de filtro onde a saída do primeiro passa por avaliações antes de dar entrada no segundo. Nesse filtro, realiza-se o processo de *Subsampling* no qual utiliza o algoritmo de *max_pooling* para reduzir o tamanho de amostras a ser processada. Esses dois processos descritos formam o mapa de características. A finalidade deste mapa trata-se da prática de aplicar repetidas vezes a saída do grupo como entrada do próximo e quantas vezes forem estabelecidas. Assim, o resultado de uma convolução, após o *subsampling*, resulta na entrada do próximo processo a ser realizado [1] [11].

1) Alimentação e Funcionamento: Diferente das redes neurais totalmente conectadas, oo invés de usar uma imagem inteira como alimento para a rede neural, utiliza técnicas mais elaboradas para extrair uma vantagem maior.

O processo de convolução é realizado sobre imagens de pequeno porte, contornando o gasto de energia processando uma imagem inteira. Dessa forma, é selecionado uma imagem de no mínimo 20×20 e os processos de convolução, *pooling* e decisão são feitas somente nesta pequena imagem. Com isso, não possui o gasto desnecessário de energia e tempo processando uma imagem completa. Para este procedimento ser aplicado em qualquer imagem, realiza o processo de seleção de regiões para as classificações. Este processo é realizado primeiramente quebrando a imagem em vários quadros[1] e realiza o processo do mapa de característica em cada uma das pequenas imagens. Os resultados dos *pooling* serão salvos numa matriz que segue a indexação da imagem original. Ela é gerada com a avaliação da convolução gerada em conjunto com o algoritmo de `max_pooling` onde selecionará o melhor resultado em determinado quadrante resultando em uma matriz só de soluções com valores avaliados como melhores. Este processo chama-se *Redução de Amostragem* [1] [11].

A função `max_pooling` é uma forma de sub-amostragem não-linear. Ela particiona a imagem de entrada em um conjunto não sobreposto de retângulos e para cada tipo de sub-região, retorna o máximo valor encontrado. Seu propósito é a eliminação de valores não maximais reduzindo o processamento computacional nas camadas posteriores. Ela também provê uma forma de transação invariante no processo realizado. Isso provê uma robustez adicional na posição, ela também é uma forma inteligente de reduzir a dimensionalidade de representações intermediárias do processo. O processo é demonstrado na Figura 1.

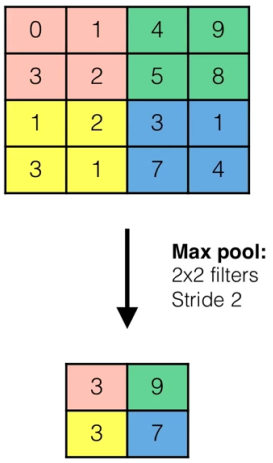


Figura 1. O processo de seleção de melhores resultados. Procedimento executado pela função `max_pooling`.

Feito todos os passos do mapa de características, utiliza-se cada quadro gerado pela última redução em uma pequena rede neural para a geração da decisão final.

Com esses processos que são anteriores a pequena rede neural, elimina-se os itens irrelevantes para o processamento sem perder a acurácia do algoritmo.

Os itens resultantes serão entrada para outra rede neural que fará de fato a decisão final. Esse último explora a correlação espacial reforçando um padrão de conectividade local entre

[1]Semelhante a ideia de Janelas Deslizantes.

neurônios nas camadas adjacentes, ou seja, as entradas da camada m são de um conjunto da camada anterior $m-1$ relacionados de uma forma contígua como é exibido na Figura 2 [3] [4].

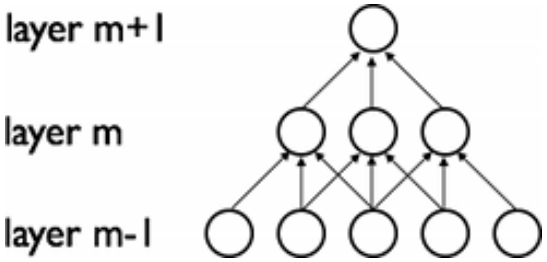


Figura 2. Exemplo das camadas de uma rede *convolutional neural network*.

Tal teoria é exibida na Figura 3. Posta uma imagem de qualquer tamanho como entrada, ela será dividida em vários blocos cujo tamanho é referente à especificado pelo algoritmo de convolução. Serão realizado os métodos de convolução em cada bloco e estes serão salvos numa matriz. Um procedimento de otimização avaliará a matriz a fim de eliminar todos os dados que forem irrelevante para o problema, sendo esses dois processos podendo repetir caso pré-definido. Com a matriz de resultados promissores em mão, é executado um algoritmo de rede neural para avaliar quais blocos são de suas respectivas classes, de acordo com o treino realizado.

Com tais procedimentos, conclui-se o processo de avaliação utilizando *convolutional neural network*. Um exemplo geral é exibido na Figura 4 no qual, cada pedaço da imagem R é recortado e selecionado, feito os processos de convolução por meio dos mapas de características φ_n e em seguida a pela rede neural totalmente conectada Ω que produz a decisão final Ψ.

B. neon

Segundo a sua documentação, *neon* é uma *API* de código-aberto, um projeto de *deep learning* referente ao Intel Nervana desenvolvido para ter excelente performance em qualquer *hardware* além de ser planejado para ser fácil de uso e extensível à outros recursos [12].

Entre suas várias características, as principais são:

- Suporte aos modelos mais comuns utilizados hoje como *convnets*, RNN, LSTM e outros;
- Modelos pré-treinados, pronto para uso;
- Integração com o bibliotecas Kernel de GPUs;
- Suporte a diferenciação automática básica;
- *Backends* de *hardware* adaptável, ou seja, é possível escrever o código em um local e portar para CPU, GPU, ou *hardware* próprio da Nervana.

Para seu treinamento e classificação, o algoritmo necessita especificamente de um *data set*, uma lista de camadas que fará a composição da arquitetura da rede, uma função de custo e uma regra de aprendizagem que no qual gerará a rede neural requerida para detecção.

A API disponibilizada é desenvolvida em linguagem de programação Python. Como é uma API de distribuição gratuita e executada em modo local, necessita de instalação de suas bibliotecas *kernel* na máquina a ser realizada os treinos e teste. Assim, para realizar a instalação, basta utilizar os seguintes comandos:

```
git clone \
https://github.com/NervanaSystems/neon.git
cd neon && make sysinstall
```

O processo de construção da rede neural passa por cinco itens principais. O primeiro é a geração de um *backend*, no qual realiza o procedimento de inicialização e definição de onde o algoritmo será executado (CPU, GPU, ...). O segundo é o carregamento do *data set*, podendo ser alguns já disponíveis pelo próprio projeto ou mesmo realizando o carregamento de um específico para treino e avaliação. O na parte de treino terceiro, deve-ser realizar a especificação do modelo de arquitetura da rede onde se define itens como as camadas, funções de ativação e os pesos dos iniciadores. Após o treinamento, é possível realizar a avaliação da rede.

IV. Desenvolvimento

A. Uso dos Data set

Para o treino e avaliação, utilizou-se de dois conjuntos de dados e *benchmark* disponibilizado gratuitamente.

Todos os data sets foram encontrados por meio do sítio http://www.face-rec.org/databases/ no qual fornece informações básicas das mais variadas bases para estudo e aplicação em pesquisa.

1) faces94: Para geração de faces, utilizou-se de um conjunto de dados disponível pela Universidade de Essex, na *School of Computer Science and Electronic Engineering*, Inglaterra, disponível em http://cswww.essex.ac.uk/mv/allfaces/faces94.html.

No conjunto utilizado, os indivíduos foram fotografados numa distância fixa da câmera e solicitados para que falem enquanto as fotos foram tiradas. O processo de fala produz expressões faciais nos indivíduos, criando maior variabilidade no conjunto de dados a ser utilizado para treino.

Todas as imagens coloridas possuem plano de fundo verde, as cabeças possuem variações naturais de rotação, movimento e posição, não possui variação de luminosidade, possui um numero considerável de expressões mas nenhuma com alto nível.

O *data set* possui um total de 153 indivíduos, com imagens de resolução fixa no valor de 180×200 e com três sub-diretórios nomeados: *male*, *female* e *malestaff* e a quantidade de seus respectivos indivíduos: 20, 113 e 20.

2) FDDB: Face Detection Data Set and Benchmark: Disponibilizado pela Universidade de Massachusets Amherst [13], o *data set* possui um conjunto de anotações para 5171 faces identificadas num conjunto de 2845 imagens retiras do conjunto nomeado *Faces in the Wild*, disponível em http://tamaraberg.com/faceDataset/index.html. O conjunto inteiro possui um total de 30281 imagens com faces coletadas da fonte *News Photographs*. Todas foram automaticamente etiquetadas

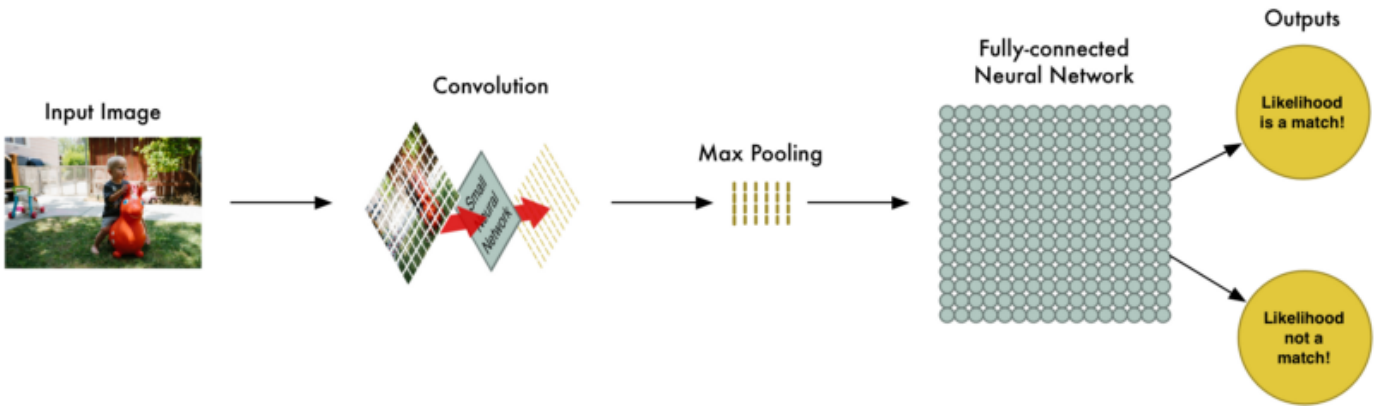


Figura 3. O processo simples de uma arquitetura de reconhecimento de padrão *convolutional neural network* final após todos os processos.

por meio do sistema descritivo *Who's in the Picture*, no qual possui 80% de acurácia [14].

Os dados estão em formato compactado `jpg`. Junto com o pacote de imagens, existe um arquivo com o seguinte formato:

```
...
<image name i>
<number of faces in this image =im>
<face i1>
<face i2>
...
<face im>
...
```

onde cada face é denotada por

```
<major_axis_radius minor_axis_radius angle
 center_x center_y 1>
```

no qual tais valores representam a posição da face como é exibido na Figura 5.

Possui ao total dez sub-diretórios separados aleatoriamente, todos com informações de localização de face. Tais diretórios possuem quantidades de imagens diferentes, mas todos possuem uma média de 280 imagens.

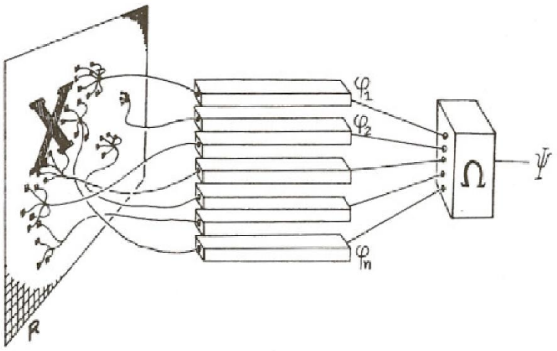


Figura 4. Exemplo de identificação de rosto especificado pelo *data set* disponibilizado pela Universidade de Massachusets Amherst.

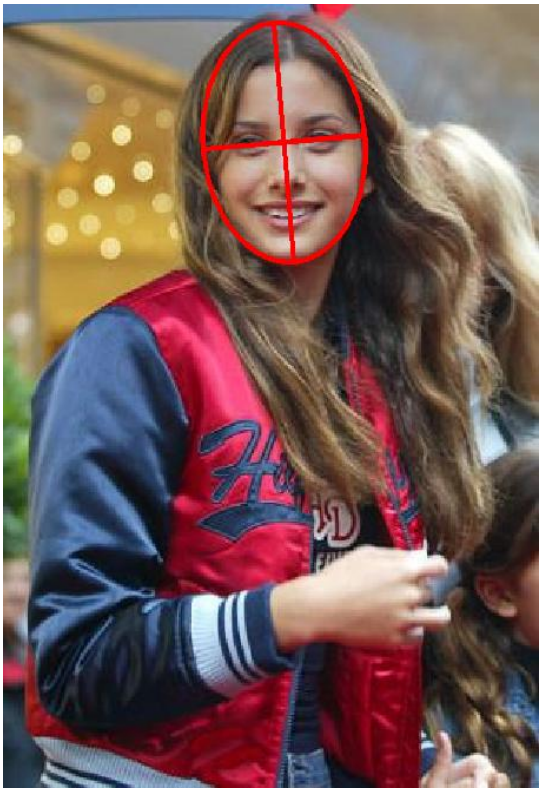
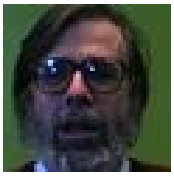


Figura 5. Exemplo de identificação de rosto especificado pelo *data set* disponibilizado pela Universidade de Massachusets Amherst.

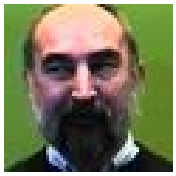
3) Disposição para Uso:

a) Conjunto `faces_set`*:* Como as imagens do *faces94* só possui a face dos indivíduos, tais como são os porta-retratos, esse conjunto de dados foi exclusivamente utilizado para treinar a rede sobre as características de faces. As imagens são modificadas para escala de cinza (Figura 6) e recortadas para terem o formato quadrado (Figura 7), padrão de entrada do algoritmo.

Utilizando todas as imagens `.jpg` deste conjunto, tem-se o total de 2980 faces prontas para treino.



(a) Face 1



(b) Face 2

Figura 6. Faces do *data set Faces94*. Imagens colorias, com tamanho 180×200 e com expressões faciais

(a) Face 1 Cortada

(b) Face 2 Cortada

Figura 7. As respectivas faces já processadas e prontas para treino.

O conjunto *FDDB* poderia ser utilizado para a geração de novas imagens de grupo `face` complementando o grupo já formado pelo *faces94* já que é grande o suficiente para o recorte das faces já que tem-se a localização de cada uma. Entretanto, mesmo com suas indicações de faces em documento texto, não foi viável utilizá-las para treino pelo motivo de que algumas unidades de faces estão incompletas por causa de obstruções ou por estarem não totalmente exibidas por completo. Faces com desfoque, com pedaços obstruídos por óculos, barbas, distorções da imagem ou mesmo incompleta por algum objeto.

Tentar utilizá-lo eliminando as faces obstruídas entraríamos novamente no problema de detecção já que seria inviável fazer o processo manualmente pelo motivo do conjunto conter inúmeras imagens podendo ter até várias unidades de rostos cada uma.

Dessa forma, o conjunto *FDDB* foi desconsiderado totalmente para a geração de faces para treino.

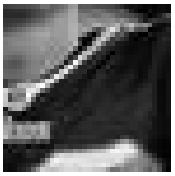
b) Conjunto `non_faces_set`*:* Para a geração de imagens não-faces, utilizou-se da técnica de selecionar pedaços aleatórios de imagens e etiquetá-las como não-face.

Como o conjunto *faces94* não possui ambiente suficiente para a geração de imagens não-faces, utilizou-se das imagens do conjunto *FDDB* para a sua geração. Por mais que ele não fosse suficiente para a geração de faces, para não-faces, seu conjunto possui grande utilidade para o corte de inúmeras não-faces para treino.

Para evitar a geração de imagens que contenham faces ou parte delas, utilizou-se das coordenadas conhecidas para evitar áreas que contenham partes de faces. Esse procedimento foi necessário para que os conjuntos de dados não confunda o treino da rede mostrando, por exemplo, uma parte de uma boca e dizer que aquilo não faz parte de uma face, sendo que essa afirmação é falsa.

Com exceção desta restrição, todas as imagens não-faces são geradas de forma aleatória como é exibido na Figura 8.

(a) Não-face 1



(b) Não-face 2

(c) Não-face 3

Figura 8. Algumas imagens recortadas para treino de não-faces.

4) Treino, Teste e Validação: Tendo definido as restrições de como será composto cada *data set*, deve-se então separar os grupos de treino, teste e validação.

Para o conjunto *faces94*, todas as imagens foram utilizadas para treino de reconhecimento faces.

O conjunto *FDDB* possui dez subconjuntos nomeados de 1 à 10, todos com descrição de localização de faces. Para a divisão de treino, teste e validação, separou os os conjuntos em três grupos onde os subconjuntos 1 à 4 irá pertencer à imagens utilizadas para recorte de não-faces para treinos, os subconjuntos de 5 à 8 serão utilizados para testes e refinamento dos resultados e os para avaliação e geração de resultados, será utilizado os subconjuntos 9 e 10.

B. API neon

Para o uso do *neon*, deve-ser realizar configurações sistêmicas no algoritmo antes de sua execução. As configurações realizadas foram:

- ***Backends*:** NervanaCPU;
- ***Data sets*:** *face94* e FDDB;
- ***Initializers*:** Uniforme;
- ***Optimizers*:** *Gradient Descent* com Momento;
- ***Activations*:** Linear Retificada, Softmax;
- ***Layers*:** Convolução, Pooling; e
- ***Cost*:** *Multi Cross Entropy*.

Para a arquitetura da rede, precisamente a descrição das suas camadas, utilizou-se de duas camadas de convolução, de *pooling* e de *Affine* que é uma rede neural totalmente conectada. A arquitetura da rede e suas configurações são exibidas a seguir:

```
layers = [
    Convolution(
        fshape=(5, 5, 4),
        init=init_uni),
    Pooling(fshape=(2, 2)),
    Convolution(
        fshape=(3, 3, 14),
        init=init_uni),
    Pooling(fshape=(2, 2)),
    Affine(nout=14,
        init=init_uni,
        activation=Rectlin(),
        batch_norm=True),
    Affine(nout=2,
        init=init_uni,
        activation=Softmax())    ]
```

As configurações utilizadas para a realização dos testes foram:

- ***Batch size*:** 512;
- ***Numbers of epoch*:** 1250;
- ***Size of image* (altura e largura):** 36;
- ***Learning Rate*:** 0.01;
- ***Momentum*:** 0.9;
- ***Type of datas*:** inteiro não sinalizado de 8-bits.

V. Experimentos

Para a realização dos experimentos, foi necessário adaptar o algoritmo para o reconhecimento de regiões, também chamado de *Region-based Convolutional Neural Network* (RCNN). Tais algoritmos necessitam que, dada uma imagem de entrada, seja gerado várias regiões da foto que são compatíveis com o tamanho de entrada do algoritmo.

Isso é necessário pois a rede neural deve operar sobre toda a imagem, de vários tamanhos possível pois as faces pode estar dispostas de vários tamanhos e posições diferentes na imagem. Entretanto, a entrada do algoritmo requisita uma imagem fixa em formato quadrado para processamento. Para que este requisito possa ser satisfeito, foi feito um processamento de geração de regiões que pega a imagem original a ser avaliada, recorta-a em várias regiões de vários tamanhos diferentes e redimensiona-as para o formato padrão de entrada da rede neural a fim de tentar localizar todas as faces na imagem, independente de sua posição ou tamanho.

Após a definição de regiões, executa-se o algoritmo em cada uma das regiões. Cada região que for definida, ao final do algoritmo, como uma face, será destacada sobre a imagem original em escala de cinza, como mostra a Figura 9 e as Figuras 12, 13, 14 e 15 em anexo.

VI. Resultados

Para obtenção de resultados factíveis, as saídas de cada região foram comparadas com os dados fornecidos pelos descritores dos *data sets* de validação. Calculando acertos e distância, é possível encontrar a taxa de acerto do algoritmo e também possíveis melhorias para trabalhos futuros.

A Tabela I exibe o total e percentual de acertos e erros de cada tipo de decisão, processado o número de 929.977 regiões.

Tabela I
Valores totais e percentuais de cada tipo de erro dos *data sets* de validação.

	Numbers	**%**	**Numbers**	**%**
True Positives	2995	55,391	2801	51,038
False Positives	2412	44,608	2687	48,961
True Negatives	239.142	49,694	239.347	54,662
False Negatives	242.078	50,305	198.515	45,337
Total Regions	*486.627*	*100*	*443.350*	*100*

É possível ver que o algoritmo teve um pouco mais de cerca de 50% de acerto em suas decisões em todos os *benchmark* testados. Como uma segunda métrica para visualização de sucesso, foi feito uma comparação de todas as regiões testadas com os respectivos pontos centrais de cada face, analisando a distância absoluta entre cada ponto reconhecido e o ponto da face. Na Figura 10, é possível perceber que os valores de Verdadeiro Positivo possuem, em média, uma distância menor que cem pixels enquanto os Falsos Positivos, por sua vez, possuem uma distância maior do ponto, o que é justificado já que não pertencem à área da face. Os pontos de Falso Positivo são pontos que estão próximo da face, mas que o algoritmo não soube decidir com tanta certeza se foram de fato regiões que pertencem a face ou não. O mesmo padrão pode ser visualizado na Figura 11.

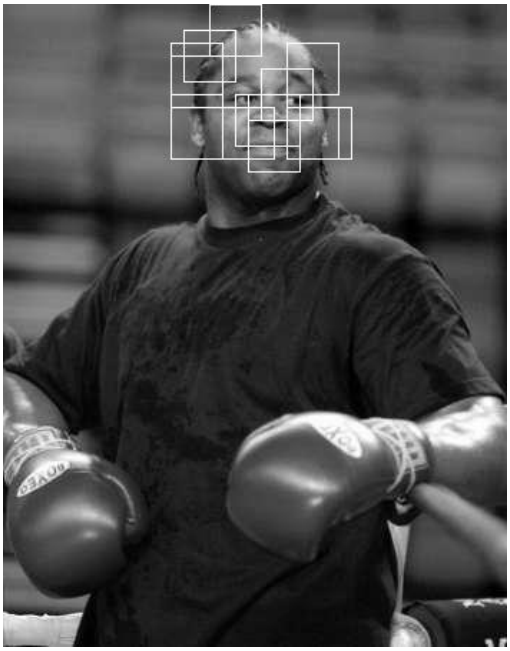


Figura 9. Exemplo de saída de uma foto após o processo de detecção.

O tempo médio gasto para treino e teste, utilizando os parâmetros e *data sets* mencionados, é de 218,6 minutos com um total de 4131[2] imagens. Com isso, cada é treinado cerca de $3,17$ imagens por segundo. Para validação, o tempo gasto foi de 8,05 minutos avaliando 539[3] imagens ao total.

VII. Conclusão

Por mais que exista ferramentas de fácil acesso e gratuitas, todas elas exigem conhecimento em teoria de inteligência artificial para sua interpretação de arquitetura, funcionamento e suporte e principalmente em sua configuração para a obtenção de resultados mais promissores.

Para o desenvolver deste trabalho, não é necessário ter conhecimento no tema de processamento digital de imagens, entretanto, deve-se ter conhecimento em teoria convolucional de imagens para escolher parâmetros adequados para cada tipo de solução requisitada. Este trabalho baseou-se nos relatos de

[2]Soma de todos os quatro *data sets* e com o de face: $290 + 285 + 274 + 302 + 2980$.

[3]Dois *data sets* de $259 + 280$.

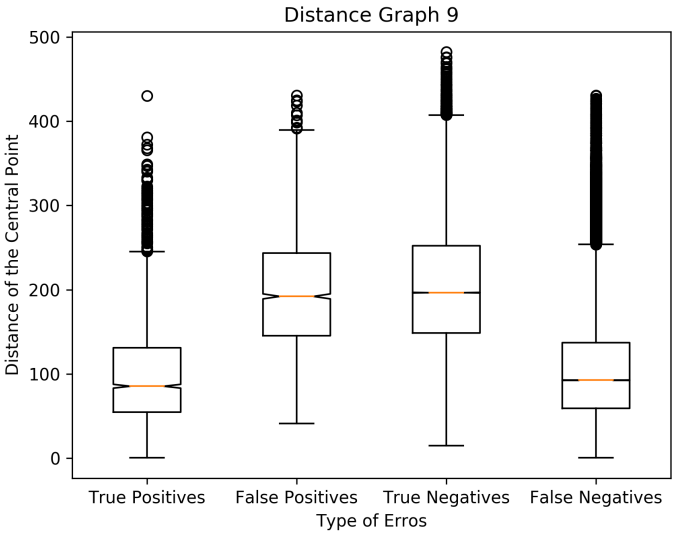


Figura 10. Exibição das distâncias e valores de erro do *data set* 09 em cada tipo de região.

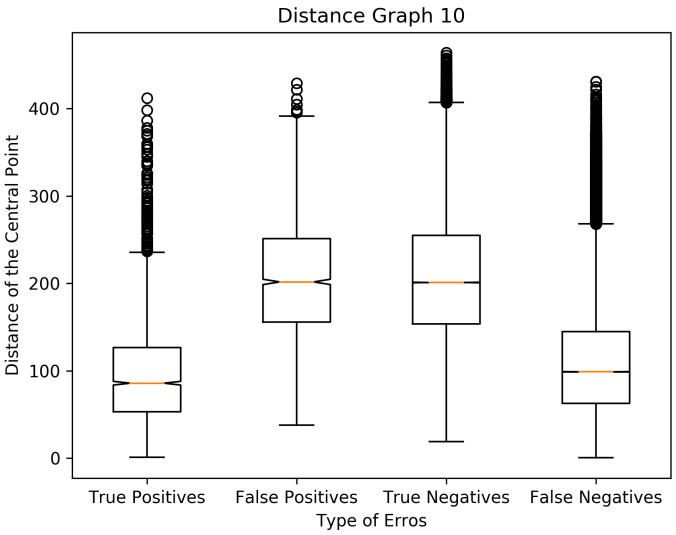


Figura 11. Exibição das distâncias e valores de erro do *data set* 09 em cada tipo de região.

Garcia [1] no qual utiliza imagens e máscaras de pequeno porte e obtendo resultados mais promissores do que a utilização de uma rede neural totalmente conectada, isso sem levar em consideração o gasto energético desta.

Todos os parâmetros utilizados para configuração da rede que não foram explicitados por Garcia [1], como *rate learning* e *momentum* foram obtidos de forma empírica, avaliando o resultado obtido, alterando os valores e realizando novos testes.

A. Trabalhos Futuros

Uma possível forma para melhorar os resultados seria modificar o *data set* de faces de forma a rotacionar e redimensionar as tais para que se tenha mais situações mapeadas em treino, reconhecendo mais faces que não estão posicionadas em perfil ou com com leves giros. Também realizar um filtro nas imagens do *data set FDDB* para utilizar as faces que estão visíveis e sem obstrução para treino a fim de deixar a rede mais abrangente para outros tipos de disposição de faces.

Alterar o parâmetro de aprendizagem do algoritmo para que consiga diferenciar por sí só com mais certeza as regiões apresentadas à ele.

Alteração do processo de criação de saída selecionando o contorno de maior área que abrange o maior espaço que contenha a face, eliminando todos os seus sub-quadrados internos.

Outra possível modificação no trabalho seria alterar as saídas da rede neural. Atualmente, possui-se duas saídas dizendo a probabilidade de cada uma. Poderia realizar uma modificação de forma a colocar somente uma saída, mostrando a porcentagem de ser ou não uma face.

REFERÊNCIAS

[1] C. Garcia and M. Delakis, "Convolutional face finder: A neural architecture for fast and robust face detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 11, pp. 1408–1423, 2004.

[2] L. Haoxiang and Lin, "A Convolutional Neural Network Approach for Face Identification," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5325–5334, 2015. [Online]. Available: http://www.cv-foundation.org/openaccess/content{_}cvpr{_}2015/html/Li{_}A{_}Convolutional{_}Neural{_}2015{_}CVPR{_}paper.htmlhttp://www.cs.nyu.edu/{~}xd283/face2012.pdf

[3] Y. LeCun, "Generalization and network design strategies," pp. 143–155, 1989. [Online]. Available: http://masters.donntu.edu.ua/2012/fknt/umiarov/library/lecun.pdf{%}5Cnpapers3://publication/uuid/2C3E3B3C-DE43-4109-A816-7220A5C08617

[4] L. D. Le Cun Jackel, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, B. L. Cun, J. Denker, and D. Henderson, "Handwritten Digit Recognition with a Back-Propagation Network," *Advances in Neural Information Processing Systems*, pp. 396–404, 1990. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.32.5076{%}5Cnhttp://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.32.5076{&}rep=rep1{&}type=pdf

[5] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2323, 1998.

[6] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[7] G. L. Martin and J. A. Pittman, "Recognizing Hand-Printed Letters and Digits Using Backpropagation Learning," *Neural Computation*, vol. 3, no. 2, pp. 258–267, jun 1991. [Online]. Available: http://www.mitpressjournals.org/doi/10.1162/neco.1991.3.2.258

[8] J. Wang and J. Jean, "Multiresolution neural networks for omnifont character recognition," *Neural Networks, 1993., IEEE International*, 1993. [Online]. Available: http://ieeexplore.ieee.org/abstract/document/298793/

[9] S. Lawrence, C. Giles, Ah Chung Tsoi, and A. Back, "Face recognition: a convolutional neural-network approach," *IEEE Transactions on Neural Networks*, vol. 8, no. 1, pp. 98–113, 1997. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=554195

[10] R. Vaillant, "Original approach for the localisation of objects in images," *IEE Proceedings - Vision, Image, and Signal Processing*, vol. 141, no. 4, p. 245, 1994. [Online]. Available: http://digital-library.theiet.org/content/journals/10.1049/ip-vis{_}19941301

[11] A. Giusti, D. C. Cireşan, J. Masci, L. M. Gambardella, and J. Schmidhuber, "Fast image scanning with deep max-pooling convolutional neural networks," in *2013 IEEE International Conference on Image Processing, ICIP 2013 - Proceedings*, 2013, pp. 4034–4038. [Online]. Available: http://ieeexplore.ieee.org/abstract/document/6738831/

[12] “neon documentation,” http://neon.nervanasys.com/docs/latest/index.html, accessed: 2017-03-20.
[13] V. Jain and E. Learned-Miller, “Fddb: A benchmark for face detection in unconstrained settings,” University of Massachusetts, Amherst, Tech. Rep. UM-CS-2010-009, 2010.
[14] T. L. Berg, A. C. Berg, J. Edwards, and D. A. Forsyth, “Who’s in the picture,” *Advances in neural information processing systems*, vol. 17, pp. 137–144, 2004.

ANEXOS

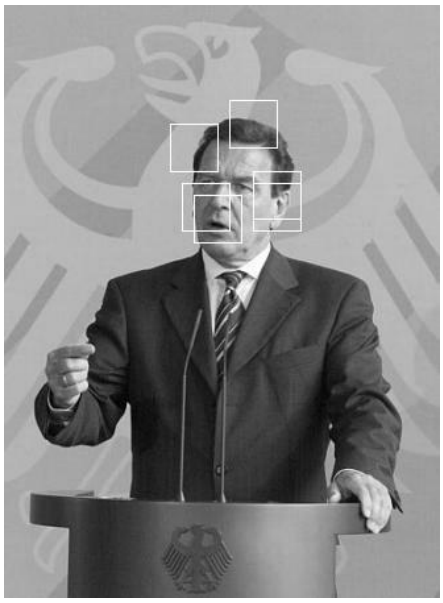


Figura 12. Resultado do processamento de reconhecimento de face no qual retornou com sucesso a face detectada.

Figura 13. Exemplo de processamento de reconhecimento de face no qual não obteve sucesso total na detecção de todas as faces. Resultado com Falso Positivos, no emblema, na bandeira e na cortina.

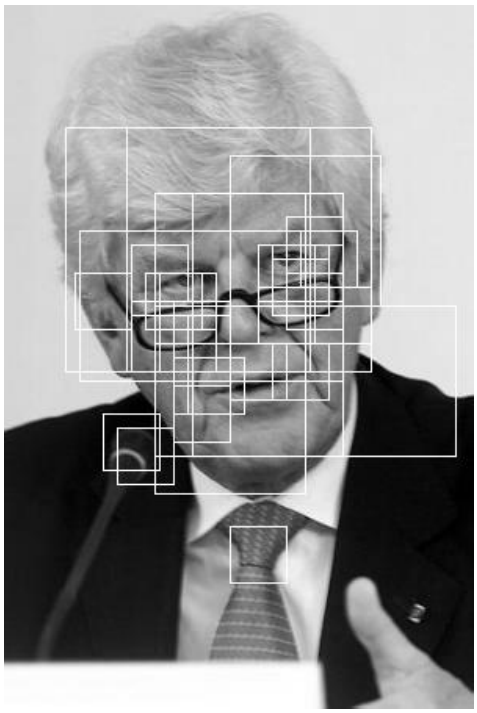


Figura 14. Exemplo de processamento de reconhecimento de face no qual não obteve sucesso total na detecção de todas as faces. Resultado com Falso Positivos, na gravata, no terno e no microfone.

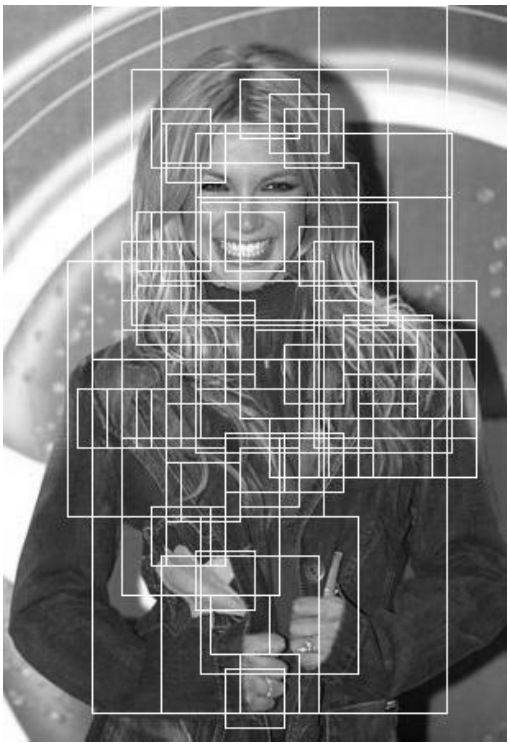


Figura 15. Exemplo de processamento de reconhecimento de face no qual não obteve sucesso total na detecção de todas as faces.