Universidade Federal de Campina Grande - UFCG

Centro de Engenharia Elétrica e Informática - CEEI

Departamento de Engenharia Elétrica - DEE

Relatório de Estágio Integrado

João Roberto Cavalcanti de Araújo

Campina Grande, Março de 2018

Universidade Federal de Campina Grande - UFCG
Centro de Engenharia Elétrica e Informática - CEEI
Departamento de Engenharia Elétrica - DEE

João Roberto Cavalcanti de Araújo

Relatório de Estágio Integrado

Relatório de Estágio Integrado submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Orientadora:
Luciana Ribeiro Veloso, Dr. Sc

Campina Grande, Março de 2018

Universidade Federal de Campina Grande - UFCG
Centro de Engenharia Elétrica e Informática - CEEI
Departamento de Engenharia Elétrica - DEE

João Roberto Cavalcanti de Araújo

Relatório de Estágio Integrado

Relatório de Estágio Integrado submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Banca examinadora.

Prof. Luciana Ribeiro Veloso, Dr. Sc
Orientadora

Professor Edmar Candeia Gurjão, Dr. Sc
Prof. Convidado

Campina Grande, Março de 2018

AGRADECIMENTOS

Agradeço a minha mãe, Valéria Carvalho, meu pai, Marcos Augusto, meu irmão, Galego e a meu padrasto Celso Peixoto por todo constante apoio que me foi dado durante o período de graduação.

Agradeço a minha namorada Natália pelo apoio e carinho.

Agradeço a Professora Dra. Luciana Ribeiro Veloso, primeira professora com quem tive o prazer de trabalhar, como orientadora de PIBIC, e agora novamente aceitou me acompanhar como orientadora de estágio.

Agradeço ao amigo e colega de trabalho Arthur Cruz de Araújo pela recomendação dentro da Vsoft. A ele assim como aos amigos de trabalho Arnaldo, João, Eric, Igor, Rafael Aline e Daniel, por ajudarem a fazer do ambiente de estágio um ambiente de aprendizado e descontração.

Agradeço a toda a equipe da Vsoft pelo agradável ambiente de trabalho e a Pedro Alves pela oportunidade e confiança em me aceitar como estagiário.

Resumo

Neste documento são relatadas as atividades realizadas durante o período de estágio obrigatório para conclusão do curso de Engenharia Elétrica da UFCG. O estágio ocorreu na empresa Vsoft Tecnologia, onde foi desenvolvido um sistema de detecção de faces, baseado em *Deep learning* e usando redes neurais convolucionais. Foi criado um banco de imagens para treinamento, validação e teste, escolhida uma arquitetura para a rede, foram elaboradas estratégias de treinamento e algoritmos de pré-processamento e pós-processamento e foram feitos com testes e ajustes dos parâmetros dos algoritmos desenvolvidos até ser obtida uma rede de desempenho superior ao detector já existente.

Palavras-chave: Rede convolucional, Deep-learning, Detector facial.

Lista de figuras

Lista de tabelas

Sumário

1 Introdução

Existem diversas partes do corpo humano que são usadas em sistemas de identificação, como a face, iris e a impressão digital. Os traços biométricos da face são os mais usados devido as suas singularidades e conveniência prática. A imagem da face de um indivíduo ainda tem a vantagem de ser adquirida usando uma câmera fotográfica simples ou de vídeo. Essas características tornam o reconhecimento e a detecção facial, alternativas de interesse para sistemas de identificação, seja complementando-os ou substituindo os que são mais antiquados.

No contexto de contagem e controle de pessoas em ambientes fechados como salas de aula e auditórios, sistemas antigos usam métodos baseados em assinatura ou senha para registrar o número de pessoas presentes e suas identidades. Tais métodos são imprecisos no sentido de não poderem garantir a identidade de um sujeito ou quanto tempo o mesmo esteve presente. Para poder registrar se um certo indivíduo esteve em um ambiente em um certo horário, métodos alternativos devem ser usados.

Nos últimos anos, sistemas de detecção facial tem sido vastamente estudados. O primeiro método a apresentar resultados promissores foi o trabalho realizado por Viola-Jones (VIOLA; JONES, 2004). Os autores propuseram um método baseado em *cascade-learning* para alcançar melhores taxas de detecção. Este método usa classificadores em cascata cada vez mais complexos, de forma que, caso uma sub-janela seja tomada como não-face por algum classificador, não há necessidade de mais computação e a próxima sub-janela deve ser analisada. Mais tarde, métodos baseados em DPM (*Deformable Part Model*) foram introduzidos, (FELZENSZWALB et al., 2010). Basicamente, esses métodos consideram um objeto formado por partes detectáveis, de tal forma que, detectando tais partes é possível detectar o objeto na totalidade. Ultimamente, esses métodos foram substituídos por técnicas baseadas em *Deep Learning*, que demonstraram alcançar melhores resultados que as técnicas anteriores.

Entretanto, a maioria dos métodos que usam *Deep Learning* apenas demonstra bom desempenho em faces posicionadas verticalmente e viradas para a câmera e em imagens de alta resolução, ou eles tem um alto custo computacional, inviabilizando seu uso prático. Comumente, faces com oclusão parcial ou em posições diferentes são ignoradas. É de grande importância, no contexto abordado, que um sistema seja capaz de detectar faces parcialmente ocultas e em diferentes posições.

1.1 Objetivos

O estágio realizado na empresa Vsoft Tecnologia teve como principal objetivo o desenvolvimento de um detector de faces para ser usado em ambientes de salas de aula. Trata-se de um sistema baseado em *Deep Learning* que faz uso de redes neurais convolucionais organizadas em uma arquitetura que requer baixo custo computacional de forma que possa ser usada em tempo real.

Pode-se especificar os seguintes passos necessários para o desenvolvimento do sistema em questão:

- Especificação de um banco de dados para treinamento e teste
- Escolha de uma arquitetura para a rede convolucional
- Determinar estratégias e parâmetros para algoritmos de pré-processamento, treinamento e pós-processamento
- Análise dos resultados e validação do sistema

2 Vsoft Tecnologia

"A Vsoft Tecnologia é uma empresa que atua no segmento de software, como fornecedora independente especializada em Identificação Biométrica". Caracterizada pelo investimento em pesquisa e desenvolvimento de tecnologia biométrica nacional em parceria com a academia.

"Sediada em João Pessoa - Paraíba, a Vsoft foi constituída em 2000 e se especializou ao longo dos anos no desenvolvimento de software voltado ao processamento digital de imagens. A Biometria tem sido seu foco principal, com a pesquisa e desenvolvimento de algoritmos de reconhecimento de faces e de impressões digitais em parceria com a UFPB (Universidade Federal da Paraíba)", (VSOFT. . . ,).

Figura 1 – Logo da Vsoft

Fonte: https://www.vsoft.com.br/

3 Fundamentação

Este capítulo apresenta o conteúdo necessário para o embasamento teórico que justifica as atividades realizadas. É feita uma breve explicação sobre redes neurais, seguido de redes neurais convolucionais. Partindo do conceito de neurônio o capítulo aborda funções de ativação, arquitetura das redes neurais, algoritmos de aprendizado (SIMON, 2008), e finaliza com as camadas principais das redes neurais convolucionais.

3.1 Redes Neurais (NN)

As redes neurais são modelos matemáticos que tentam simular o sistema nervoso central biológico. Composto pela interconexão de bilhões de neurônios, o cérebro dos animais é bastante poderoso no que diz respeito à capacidade de aprendizado e reconhecimento de padrões.

Neurônios são células nervosas cuja função corresponde à transmissão de impulsos elétricos e são compostos basicamente por três estruturas; dendritos, corpo celular e axônio. Os dendritos são estruturas bastante ramificadas e sensíveis a estímulos nervosos, é onde ocorre a recepção dos sinais nervosos. O corpo celular contém o núcleo e as organelas do neurônio. O axônio é uma estrutura alongada responsável pela transmissão do impulso nervoso para outras células.

Os neurônios só transmitem impulsos nervosos quando os dendritos são estimulados acima de um certo limiar, caso contrário não há transmissão da informação. A característica das células nervosas que permitem a variação desse limiar é chamada de neuroplasticidade. A neuroplasticidade corresponde à capacidade que sistemas nervosos tem que se adaptar à presença de certos estímulos. Em outras palavras, quanto mais se estimula uma célula nervosa, menor é o limiar necessário para que a mesma transmita o estímulo.

A seção segue com a definição matemática de *neurônio*, algumas funções de ativação usadas, algumas arquiteturas de redes neurais, regras e paradigmas de aprendizagem e conclui com um exemplo de um algoritmo de aprendizagem.

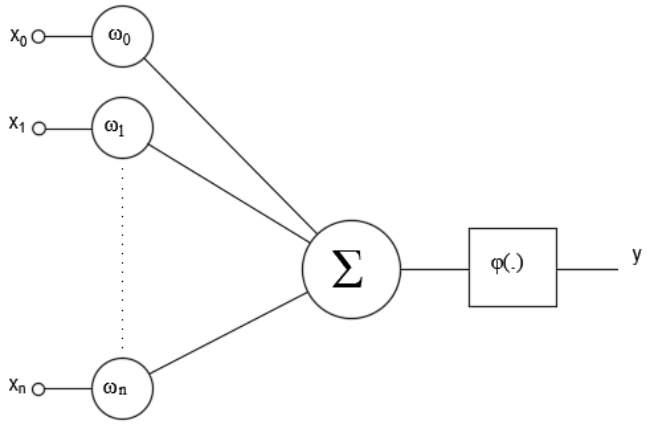
As redes neurais consistem em redes de “neurônios”, que apresentam uma função de transferência ou função de ativação, de forma que cada neurônio obedece ao seguinte modelo matemático

$$y = \varphi(v)$$

$$v = w_0 + \sum_{i=1}^{j} w_i x_i$$

onde y é a saída do neurônio e v é o nível de ativação, que corresponde a uma soma de produtos dos sinais de entrada x_i por um peso w_i somado de uma constante de polarização w_0 (Figura 2).

Figura 2 – Modelo do neurônio



Fonte: Autoria própria

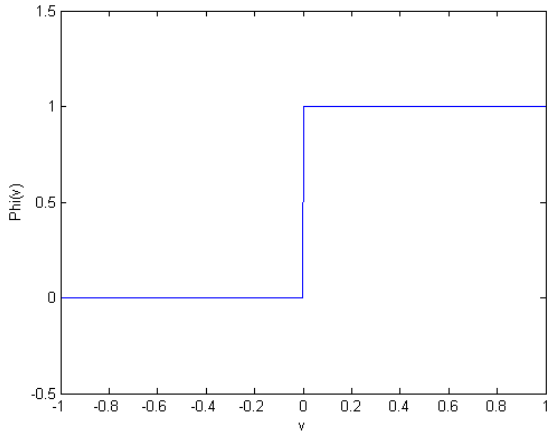
3.1.1 Tipos de Funções de Ativação

Existem diversos tipos de funções de ativação, algumas delas são:

Função de Heaviside (Figura 3). Definida como

$$\varphi(v) = \begin{cases} 1, & \text{se } v \geq 0 \\ 0, & \text{se } v < 0 \end{cases}$$

Figura 3 – Função Heaviside

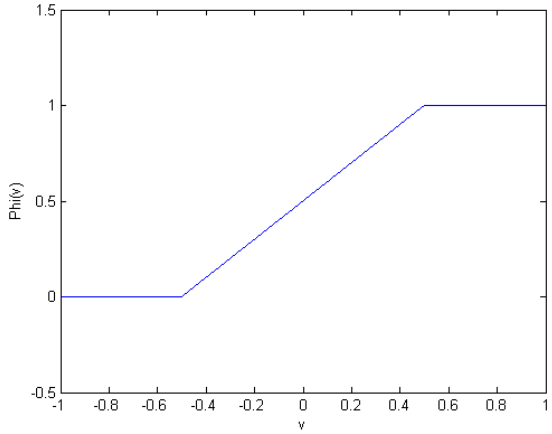


Fonte: Gráfico gerado pelo autor em Matlab

Função Linear por Partes (Figura 4). Definida como

$$\varphi(v) = \begin{cases} 1, & v \geq \frac{1}{2} \\ 0, & \frac{1}{2} > v > -\frac{1}{2} \\ \text{-1}, & v \leq -\frac{1}{2} \end{cases}$$

Figura 4 – Função Linear



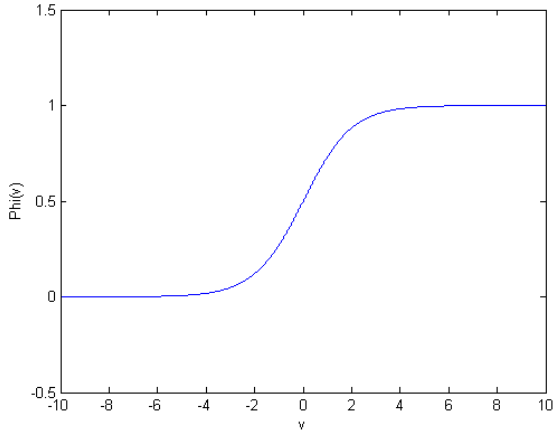
Fonte: Gráfico gerado pelo autor em Matlab

Função Sigmoide (Figura 5). Definida como

$$\varphi(v) = \frac{1}{1 + exp^{-av}}$$

onde a é o parâmetro de inclinação da função sigmoide.

Figura 5 – Função Sigmoide



Fonte: Gráfico gerado pelo autor em Matlab

3.1.2 Arquiteturas de Redes Neurais

A estrutura de uma rede neural está relacionada com o propósito da mesma e corresponde à disposição dos neurônios na rede. Quanto a sua topologia, pode-se classificar

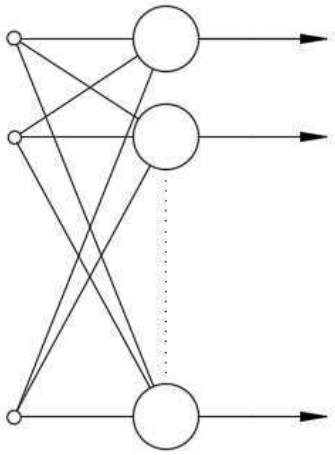
as redes neurais em dois tipos, *redes feed-forward* e *redes recorrentes*.

Redes Feed-forward

As *redes feed-forward* podem ser subdivididas em dois grupos, *monocamada* e *multicamada* com alimentação unidirecional. Como o nome sugere as *redes feed-forward* não possuem realimentação, o sinal é sempre enviado para os neurônios a frente quando possível.

As *redes monocamada* com alimentação unidirecional só possuem uma camada de neurônios. Os mesmos neurônios que recebem os sinais de entrada fornecem os sinais de saída (Figura 6).

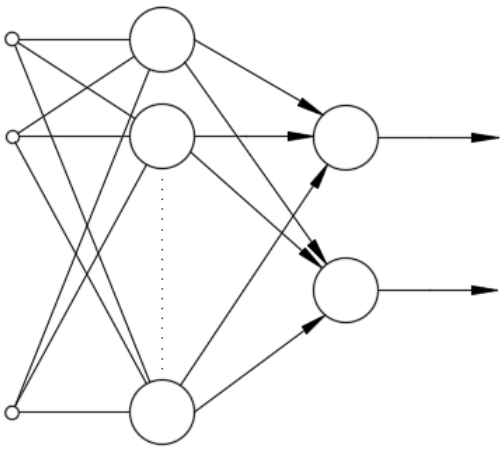
Figura 6 – Rede monocamada



Fonte: Autoria própria

As *redes multicamada* com alimentação unidirecional possuem ao menos duas camadas, uma camada de saída e uma camada escondida (Figura 7).

Figura 7 – Rede multicamada

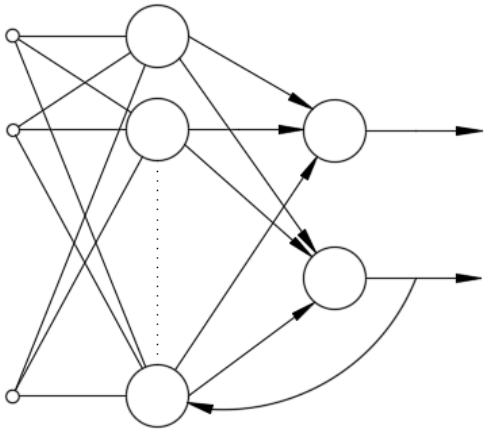


Fonte: Autoria própria

Redes Recorrentes

As redes neurais são classificadas como *Redes Recorrentes* se tiverem ao menos um laço de realimentação (Figura 8). Uma realimentação acontece "sempre que um elemento do sistema influencia em parte a entrada aplicada àquele elemento em particular", (SIMON, 2008).

Figura 8 – Rede Recorrente



Fonte: Autoria própria

3.1.3 Aprendizagem

A característica mais importante de uma rede neural é a sua capacidade de *aprender*. A aprendizagem de uma rede neural está diretamente ligada com sua eficiência. Ela consiste na modificação de seus parâmetros devido à estímulos do ambiente no qual ela está inserida. Esse processo ocorre de forma iterativa, ou seja, a cada iteração os pesos sinápticos e de polarização mudam de forma que a rede se adapte cada vez mais ao seu ambiente.

Os dados usados para fazer com que uma rede neural *aprenda* são chamados de dados de treinamento. Eles são formados por dados representativos do universo abordado. Os dados de treinamento passam para a rede informações sobre o seu ambiente e, no caso de treinamento supervisionado, sobre seu comportamento desejado. A forma com a qual a rede modifica seus parâmetros define o algoritmo de treinamento.

No processo de aprendizagem exitem as chamadas *Regras de aprendizagem* e os *Paradigmas de aprendizagem*. A seguir serão abordadas três regras de aprendizagem; *aprendizagem por correção de erro*, *aprendizagem baseada em memória* e *aprendizagem Hebbiana*. Logo após serão abordados dois paradigmas de aprendizagem; *supervisionada* e *não-supervisionada*.

3.1.4 Aprendizagem por correção de erro

Esse tipo de aprendizagem se baseia na diferença entre a saída real da rede e a saída desejada para uma entrada específica. Tomando como exemplo o caso simples de apenas um neurônio k. Considere que esse neurônio recebe um vetor de entrada $x(n)$ ocasionando em uma saída $y_k(n)$. Esta saída é comparada com o sinal de saída desejado $d_k(n)$, gerando assim um sinal de erro definido como

$$e_k(n) = d_k(n) - y_k(n)$$

O sinal de erro é usado como entrada para um sistema de controle que modifica os pesos das sinapses de forma que a resposta real do neurônio se aproxime da resposta desejada. Isso é feito minimizando-se a *função de custo*, que é definida como

$$E(n) = \frac{1}{2}e_k^2(n)$$

Essa função de custo é interpretada como a energia instantânea do erro. Quando sua derivada é zero, quer dizer que ela foi minimizada, ou seja, foi encontrado um mínimo local. O objetivo da aprendizagem por correção de erro é definir os pesos sinápticos necessários para se obter um erro menor ou igual ao erro mínimo tolerável $e(n) \leq \varepsilon$. Mais adiante será mostrado detalhadamente um algoritmo chamado *Gradiente Descendente*, que usa a regra de aprendizagem por correção de erro.

3.1.5 Aprendizagem baseada em memória

Na aprendizagem baseada em memória, exemplos de experiências que contém informações corretas sobre entradas e saídas são armazenadas em memórias. A classificação de um vetor x que não tenha sido visto antes é feita analisando-se a *vizinhança local* de x.

Os algoritmos que fazem uso da aprendizagem baseada em memória são diferenciados entre si pela forma com a qual fazem uso de duas escolhas necessárias. Como definir a *vizinhança local* de x e a regra de aprendizagem aplicada aos exemplos de treinamento na *vizinhança local* de x

Uma regra bastante simples usada em algoritmos de aprendizagem baseada em memória é a chamada *regra do vizinho mais próximo*. Ela consiste em classificar um vetor de teste x_t no vetor de treinamento x_n que possuir a menor distância euclidiana com relação ao vetor de teste. Ou seja, caso

$$\min_i d(x_i, x_t) = d(x_n, x_t)$$

em que

$$i = 1, 2, ..., N$$

e $d(x_i, x_t)$ é a distância euclidiana entre o vetor de teste e o vetor de treinamento x_i. Então x_t será classificado como x_n.

3.1.6 Aprendizagem Hebbiana

Baseada nos estudos do neuropsicólogo Hebb(1949) a aprendizagem Hebbiana consiste no fortalecimento ou enfraquecimento de sinapses como função da correlação temporal entre atividades *pré-sinápticas* e *pós-sinápticas*. A regra de Hebb dita que

- "Se dois neurônios em ambos os lados de uma sinapse são ativados simultaneamente, então a força daquela sinapse é seletivamente aumentada"
- "Se dois neurônios em ambos os lados de um sinapse são ativados assincronamente, então aquela sinapse é seletivamente enfraquecida ou eliminada"

Uma forma bastante simples de aprendizagem Hebbiana é apresentada pelo seguinte modelo matemático

$$\Delta\omega(n) = \beta \times y(n)x(n)$$

onde $y(n)$ e $x(n)$ representam os sinais pós-sinápticos e pré-sinápticos respectivamente, e β é uma constante positiva definida como *taxa de aprendizagem*.

3.1.7 Aprendizagem supervisionada

No que diz respeito aos paradigmas de aprendizagem, a aprendizagem supervisionada consiste naquela quando existe um indicador da resposta desejada que é apresentada a rede, ou um *professor*. Nela tanto o professor quanto a rede são apresentados à vetores de treinamento característicos de um certo ambiente. O professor é capaz de informar à rede qual comportamento da mesma é o desejado.

Supondo que inicialmente a rede não possui qualquer conhecimento acerca do ambiente, é de se esperar que as primeiras saídas obtidas dos vetores de treinamento estejam completamente incoerentes com as saídas desejadas, de forma que se tem sinal de erro considerável. O erro é usado em *funções de custo* como o sinal de referência que o algoritmo de aprendizado possui para fazer os ajustes dos parâmetros da rede.

3.1.8 Aprendizagem não supervisionada

Neste tipo de aprendizagem não há a presença de um *professor*, a rede usa apenas as entradas como parâmetros de classificação. A rede aprende a classificar padrões semelhantes e diferenciá-los uns dos outros.

3.1.9 Gradiente Descendente

Para que uma rede neuronal funcione é preciso, após a fase de arquitetura, passar por uma etapa de treinamento. Para o caso de treinamento supervisionado, a etapa de

treinamento consiste em informar a rede valores de entrada e seus respectivos valores desejados de saída, para que os pesos w_i sejam calculados de forma a se obter o menor erro tolerável. Uma vez treinada, a rede está pronta para ser testada.

Um algoritmo bastante simples usado no treinamento supervisionado de NN é o chamado **gradiente descendente**. Ele é implementado da seguinte forma, considere o seguinte sinal de saída de uma unidade linear

$$o = \sum_{i=1}^{j} w_i x_i$$

e a seguinte função de custo

$$E(\vec{w}) = \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

onde

- D é o conjunto de treinamento
- t_d é a saída desejada para o dado de treinamento d
- od é a saída da unidade linear para o dado de treinamento d

Deseja-se encontrar o vetor $\vec{w}$ que minimiza a função E. O gradiente da função de custo é dado por

$$\nabla E(\vec{w}) = \left[\frac{\partial E}{\partial w_0} \frac{\partial E}{\partial w_1} \cdots \frac{\partial E}{\partial w_j} \right]$$

A atualização dos pesos é feita da seguinte forma

$$\vec{w} \leftarrow \vec{w} + \Delta \vec{w}$$

em que

$$\Delta \vec{w} = -\beta \nabla E(\vec{w})$$

e β é chamado de taxa de aprendizagem. Assim, temos

$$\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i} \frac{1}{2} \sum_d (t_d - o_d)^2$$

$$= \frac{1}{2} \sum_d \frac{\partial}{\partial w_i} (t_d - o_d)^2$$

$$= \sum_d (t_d - o_d) \frac{\partial}{\partial w_i} (t_d - o_d)$$

$$\frac{\partial E}{\partial w_i} = \sum_d (t_d - o_d)(-x_{id})$$

Logo, a atualização dos pesos se torna

$$w_i \leftarrow w_i + \beta \sum_d (t_d - o_d) x_{id}$$

Os pesos são modificados até que o erro seja menor ou igual à um limiar de tolerância, $e \leq \varepsilon$

É necessário ter em mente que o desempenho de uma rede neural varia bastante tanto com a arquitetura escolhida, quanto com os elementos usados na fase de treinamento. Uma vez que uma rede foi treinada ela já pode ser imersa no ambiente para o qual foi projetada e ser usada.

3.2 Redes Neurais Convolucionais (CNN)

O estudo do funcionamento e implementação das redes neurais convolucionais foi feito basicamente pelo material disponível na *web* pelo curso CS231n - *Convolutional Neural Networks for Visual Recognition* da Universidade de Stanford, (Karpathy).

As redes neurais convolucionais possuem todas as características das redes neurais comuns. São formadas por neurônios que possuem pesos para os sinais de entrada, constantes de polarização e uma função de ativação. Entretanto, seu uso se dá primordialmente para o reconhecimento de padrões quando a entrada consiste em sinais bidimensionais, como uma imagem.

Ao contrário das redes neurais comuns, uma CNN não conecta a um neurônio de uma certa camada todos os sinais de saída da camada anterior. Considere uma imagem de tamanho $100 \times 100 \times 3$, nesse caso, para uma rede neural comum, cada neurônio da primeira camada possuiria 30.000 pesos a serem aprendidos. Como se usa vários neurônios, a quantidade de parâmetros cresceria demasiadamente levando a um alto custo computacional e possível *overfitting*, quando a rede não consegue generalizar e se especializa nas amostras de treinamento.. Em uma CNN cada neurônio está ligado apenas a uma região da camada anterior.

A arquitetura de uma CNN é feita de forma que cada camada de neurônios possui três dimensões: altura, largura e profundidade. As redes convolucionais possuem três tipos principais de camadas, são elas: camada convolucional, camada *pooling* e camada *fully-connected*.

3.2.1 Camadas Convolucionais

As camadas convolucionais correspondem a grupos de neurônios com pesos aprendíveis e que se conectam apenas a uma região da camada anterior. Podem ser vistas como filtros dispersos espacialmente ao longo da altura e largura, entretanto, com a mesma

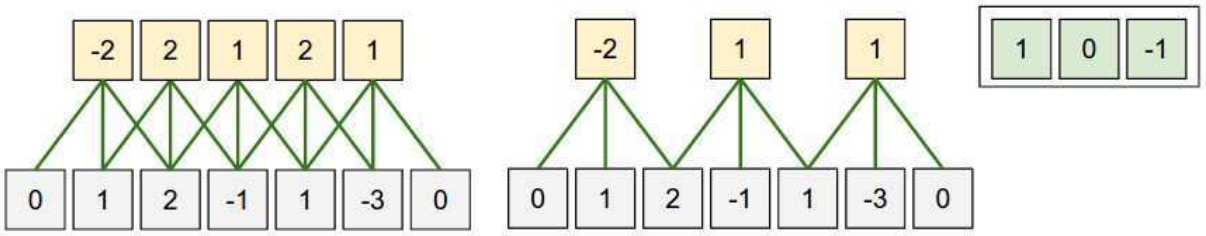
profundidade do sinal de entrada. Por exemplo, pode-se pensar em filtros da primeira camada de uma CNN como tendo dimensões $5 \times 5 \times 3$ *pixels*, considerando que a imagem de entrada possui largura e altura maior ou igual a 5×5. A profundidade do filtro corresponde a profundidade da imagem, que possui 3 canais de cor.

O formato do sinal de saída de cada camada convolucional depende dos hiperparâmetros *depth*, *stride* e *zero-padding*. *Depth* significa a profundidade do sinal de saída e corresponde à quantidade de filtros da camada. *Stride* é a distância em *pixels* que um filtro percorre ao ser transladado para o local do filtro seguinte. As vezes, é conveniente envolver o sinal com valores nulos. A quantidade de grupos de zeros inseridas para envolver o sinal é chamado *zero-padding*.

O volume do sinal de saída é dado como função do tamanho do sinal de entrada (W), o tamanho dos filtros (F), o *stride* utilizado (S) e a quantidade de vezes que o sinal foi envolto com zeros, *zero-padding*, (P). A equação do tamanho do sinal de saída é dada por

$$\frac{W - F + 2P}{S} + 1$$

Figura 9 – Arranjo espacial unidimensional



Fonte: http://cs231n.github.io/convolutional-networks/

Na Figura 9 é ilustrado como seriam dispostos os nerônios para um sinal de tamanho 5 com filtros de tamanho 3 e *zero-padding* igual à 1. Pode-se observar que, para um *stride* igual à 1, são necessários 5 neurônios e, para um *stride* igual à 2, são necessários 3 neurônios. Todos os neurônios possuem como pesos os valores 1, 0 e -1, representados pelos quadrados verdes. Os quadrados laranja e cinza representam os sinais de saída e entrada, respectivamente.

Mesmo com neurônios sendo responsáveis por uma única região da imagem, é provável que o número de parâmetros seja ainda bastante alto. Considere uma imagem inicial de tamanho $100 \times 100 \times 3$ e a primeira camada convolucional possuindo 10 filas de filtros 5×5 com *stride* de 1 e *zero-padding* igual à 0. Isso implica que o número de neurônios da primeira camada convolucional será de $96 \times 96 \times 10 = 92.160$ neurônios,

sendo que cada neurônio possui $5 \times 5 \times 3 = 75$ e mais uma constante de polarização, ou seja, será ao todo 7.004.160 parâmetros apenas na primeira camada convolucional.

Como forma de diminuir drasticamente o número de parâmetros e melhorar o custo computacional da rede, faz-se o chamado *compartilhamento de parâmetros*. Esse esquema se baseia da presunção de que se uma característica pode ser computada em uma posição (x, y) arbitrária, então a mesma característica pode ser computada em outra posição (x_2, y_2). Desta forma, cada fila de filtros em uma camada convolucional possuirá os mesmos pesos. Com isso em mente pode-se pensar que o sinal de saída de uma camada convolucional corresponde a convolução de filtros com o sinal de entrada, por isso o nome de camada convolucional.

Recalculando o número de parâmetros do exemplo anterior, agora para o caso em que se usa o compartilhamento de parâmetros temos $10 \times 5 \times 5 \times 3 + 10 = 760$ parâmetros

3.2.2 Camada *Pooling*

A camada *pooling* existe para diminuir o tamanho de representação da imagem, reduzindo assim o número de parâmetros e o custo computacional. Ela é geralmente posicionada logo após uma camada convolucional.

Essa camada recebe um sinal de tamanho $W \times H \times D$ e retorna um sinal de tamanho $W_2 \times H_2 \times D_2$, onde

$$W_2 = \frac{W - F}{S} + 1$$

,

$$H_2 = \frac{H - F}{S} + 1$$

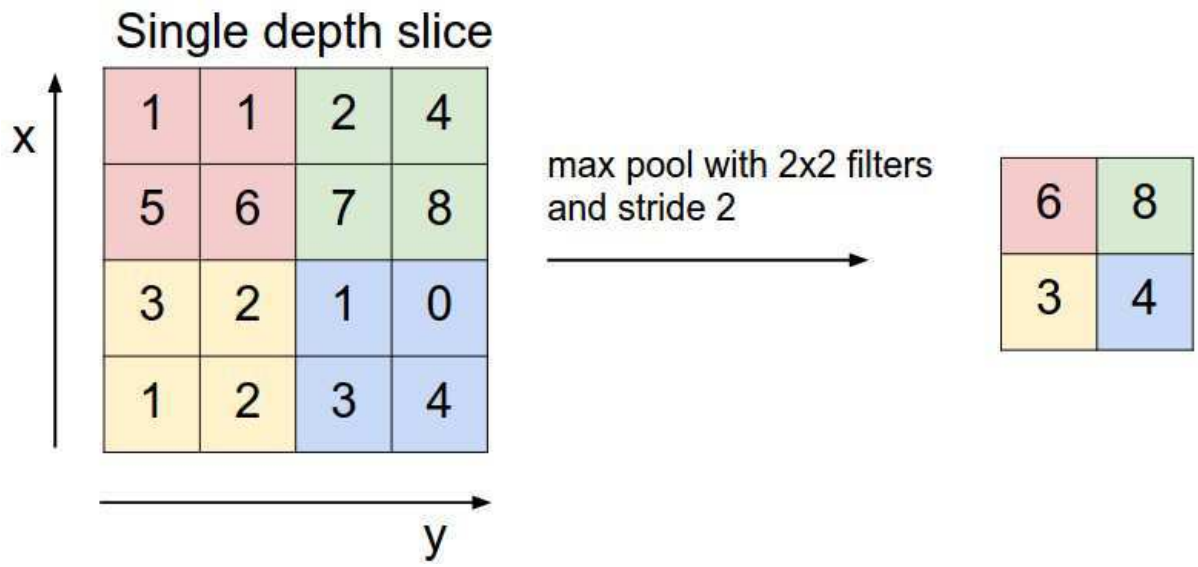
,

$$D_2 = D$$

.

Geralmente, a camada *pooling* usa filtros 2×2 com *stride* igual à 2 e a operação MAX, essa operação retorna o valor máximo das entradas do filtro. O processo realizado por essa camada é ilustrado na Figura 10

Figura 10 – Max *Pooling*



Fonte: http://cs231n.github.io/convolutional-networks/

3.2.3 Camada *Fully-Connected*

A camada *fully-connected* se comporta como as camadas em redes neurais regulares já vistas, cada neurônio dela possui conexões para todos os sinais de saída da camada anterior mais uma constante de polarização.

Além dessas camadas, existem técnicas comumente usadas para melhorar a performance de uma rede neural convolucional, entre elas, estão a *batch normalization* e a adição de uma camada *dropout*.

3.2.4 Camada *Dropout*

A camada *dropout* funciona eliminando aleatoriamente os sinais dos neurônios da camada anterior. Ela só possui um parâmetro que corresponde à probabilidade de um neurônio ter seu sinal desconsiderado. A vantagem desta camada é que ela previne o efeito de *overfitting*.

Esta técnica deve ser usada apenas na etapa de treinamento. Durante o teste da rede a camada *dropout* deve ter seu parâmetro de probabilidade selecionado como 0.0 ou ser substituída por uma camada de multiplicação constante.

3.2.5 Batch Normalization

A técnica chamada *batch normalization* corresponde à normalização do sinal de entrada de cada camada, fixando os valores de média para 0 e variância para 1, (IOFFE; SZEGEDY, 2015).

Essa estratégia tende a acelerar o aprendizado da rede, aumentar sua taxa de acerto e reduz a necessidade do uso de camadas *dropout*.

4 Atividades realizadas

Neste capítulo são relatados os procedimentos realizados durante o período de estágio. O objetivo principal foi de desenvolver um detector de faces, baseado em *deep convolutional neural networks* para uso a priori a fiscalização de salas de aula por contagem de pessoas.

A criação de tal detector envolve uma série de etapas. A obtenção de um banco de dados, a escolha de uma arquitetura para a rede neural, etapas de pré-processamento, treinamento e pós-processamento, sendo o processamento tido como a fase em que a rede recebe uma imagem e retorna um *heatmap*.

O desempenho de uma rede neural está diretamente relacionado ao banco de dados de treinamento. Porque o banco tem que ser específico o suficiente para que a rede aprenda e generalize as informações fornecidas.

Não existe uma regra que determina qual arquitetura uma rede deve ter em função de sua aplicação. Entretanto, existem técnicas que tendem a melhorar a eficiência de uma rede. A utilização de camadas *dropout* tende a prevenir *overfitting*. A técnica de *batch normalization* pode acelerar o processo de treinamento e prevenir *overfitting*. Também se deve ter em mente que o tamanho de uma rede está diretamente ligado ao seu tempo de processamento.

A etapa de pré-processamento pode ser qualquer tratamento que a imagem de entrada receba antes de ser enviada à rede. Neste caso, o pré-processamento é feito redimensionando a imagem de entrada. Na etapa de treinamento, é necessário que todas as imagens sejam do mesmo tamanho. Na etapa de teste, é interessante que uma mesma imagem seja enviada a rede em tamanhos distintos, pois a rede sempre "procura"por faces de um tamanho fixo de *pixels*, que correspondem ao tamanho das imagens de treinamento. Logo, ao se redimensionar a imagem, é possível que a rede detecte uma face antes não detectada.

A etapa de treinamento corresponde ao aprendizado da rede. Fazendo uso da estratégia de aprendizado supervisionado, imagens de face e de não-face, do banco de treinamento, são passadas para a rede junto com o sinal de saída desejado para cada uma das imagens.

A etapa de pós-processamento corresponde a qualquer tratamento pelo qual o sinal de saída da rede passa para que possa ser interpretado. Neste caso, no pós-processamento é estabelecido um limiar para se considerar o valor de saída da rede como sendo uma face, em seguida esses valores são usados para determinar o lugar da imagem de entrada onde

cada face foi detectada.

Inicialmente, foram analisados dois bancos de dados para a construção da rede, o WIDER face (YANG et al., 2016) e o LFW (HUANG et al., 2007).

4.1 WIDER face

O WIDER face é um banco de dados de referência no que diz respeito a sistemas de detecção facial, além de ser disponível ao público. Ele consiste em 32.203 imagens separadas em 61 classes de eventos onde são classificadas 393.703 faces, com alto grau de variação em escala, ângulo e oclusão. Para cada classe são selecionadas aleatoriamente 40% das imagens para treinamento, 10% para teste e 50% imagens para validação.

O banco de validação é usado para acompanhar e confirmar o aprendizado da rede. Durante a etapa de treinamento são enviadas, eventualmente, imagens de validação para a rede. As respostas da mesma para as imagens de validação são comparadas com o resultado esperado de forma que se pode quantificar sua eficiência.

O WIDER face disponibiliza gabaritos para as imagens de treinamento e validação apenas. Este corresponde a um arquivo com informações sobre os retângulos que identificam as posições das faces. Para cada retângulo o gabarito informa também quão borrada está a imagem, presença de expressão facial forte, iluminação, oclusão, pose e se a imagem é válida ou não (Figura 11).

Do banco de treinamento e validação foram retiradas as faces oclusas, inválidas, borradas e menores que 20×20, restando 60.813 faces para serem usadas.

Figura 11 – WIDER face

Fonte: http://mmlab.ie.cuhk.edu.hk/projects/WIDERFace/index.html

4.2 LFW

O LFW - *Labeled Faces in the Wild* (Figura 12) consiste em um banco de dados de faces retiradas da web. O banco possui 13.233 imagens rotuladas com o nome da pessoa

a qual pertence. Das pessoas presentes no banco, 1.680 possuem duas ou mais imagens. O propósito do LFW está relacionado ao problema de reconhecimento facial, que inclui, porém, não se limita as seguintes situações:

- Dadas duas imagens cada qual contendo uma face, decidir se elas pertencem à mesma pessoa ou não.
- Dada uma imagem de uma face, decidir a qual pessoa dentro de um grupo de indivíduos esta face pertence.

Mesmo sendo criado para o problema de reconhecimento de faces, não há razão aparente que impossibilite o uso do LFW para abordar o problema de detecção de faces, que é o objetivo principal das atividades desse estágio.

O problema de reconhecimento difere do problema de detecção pelo seguinte motivo. No problema de reconhecimento, já se supõe que a imagem a ser analisada corresponde a uma face e o desafio está em classificá-la como pertencente a um determinado indivíduo. Já o problema de detecção está em classificar uma determinada imagem como face ou encontrar faces presentes em uma imagem.

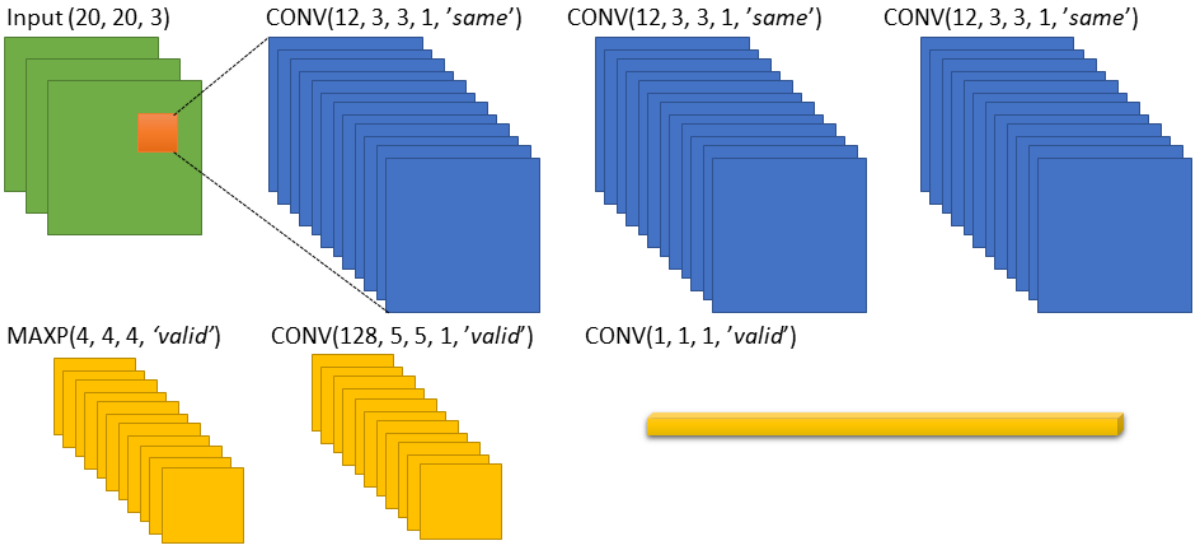
Figura 12 – LFW

Fonte: http://vis-www.cs.umass.edu/lfw/#download

4.3 Análise dos bancos de dados

Após a escolha dos bancos de dados para treinamento da rede, foram feitos três treinamentos em uma rede com a arquitetura ilustrada na Figura 13

Figura 13 – Arquitetura 1



Fonte: Autoria prórpia

A camada de entrada é composta por uma imagem de tamanho 20×20 *pixels* em seguida uma camada convolucional composta por 12 filtros de dimensões 3×3 com *stride* igual a 1, seguido de uma camada convolucional composta por 12 filtros de dimensões 3×3 com *stride* igual a 1, seguido de outra camada convolucional composta por 12 filtros de dimensões 3×3 com *stride* igual a 1, seguido de uma camada *max pooling* de dimensões 4×4 e *stride* igual a 4, seguido de uma camada convolucional composta por 128 filtros de dimensões 5×5 com *stride* igual a 1, e por fim uma camada convolucional de dimensões 1×1 e *stride* igual a 1. Os parâmetros preenchidos com 'same', ou 'valid' correspondem ao *zero padding*. Onde 'same' implica que o sinal de entrada será envolvido com zeros de forma que o tamanho da imagem de saída seja igual ao tamanho da imagem de entrada. Já o *padding* igual a 'valid' implica que o sinal de entrada não será envolvido com zeros.

A arquitetura inicial escolhida é bastante simples, porque se tem a intenção de usá-la em tempo real, futuramente. Logo, o tempo de processamento da rede precisa ser curto, no máximo um segundo para cada *frame*.

O tamanho da imagem de entrada, 20×20, foi escolhido com base nos resultados de experimentos anteriores com detecção facial realizados pela Vsoft.

A rede foi treinada em linguagem Phyton utilizando a biblioteca TensorFlow. TensorFlow é uma biblioteca de código aberto para computação numérica que usa grafos para fluxo de dados. Foi desenvolvida pela Google com o propósito de utilização em aprendizado de máquina e redes neurais; entretanto, seu uso pode ir além dessas aplicações.

Cada uma dos treinamentos foi realizado com um dos bancos de dados: WIDER face, LFW e WIDER face + LFW. Em que este último corresponde a um banco formado pela junção dos bancos WIDER face e LFW.

Após o treinamento, a rede foi testada no banco de teste, composto por 485 imagens de salas de aula com 8.165 faces gabaritadas. A rede apresentou, em todos os três treinamentos, resultados insatisfatórios, fazendo com que os bancos LFW e WIDER face fossem desconsiderados para o treinamento.

A baixa eficiência apresentada pela rede provavelmente foi influenciada pelas seguintes razões:

- Os bancos WIDER face e LFW são gerais demais para serem usados no caso específico de detectar faces em salas de aula.
- Os bancos possuem uma quantidade relativamente pequena de imagens.
- A estrutura da rede é simples demais

Como solução a esse problema, pensou-se na construção de um banco de imagens de faces próprio.

4.4 Construção de um banco de dados

Foi construído um banco de dados com imagens de salas de aula fornecidas pelas auto escolas que utilizam o serviço da Vsoft. Um total de 8.919 imagens de salas de aula foram usadas para extrair 124.716 imagens de faces e 174.763 imagens de não-faces.

Com auxílio do *software* Detector Tool Visualizer, desenvolvido pela Vsoft e para uso exclusivo da mesma, construiu-se o banco. Este programa permite que o usuário gabarite e classifique retângulos em uma imagem, além de fornecer informações acerca do desempenho de uma rede em um banco de teste, dentre outras funcionalidades.

A construção do banco se deu por duas etapas: marcação dos positivos e dos falso negativos.

4.4.1 Marcação dos positivos

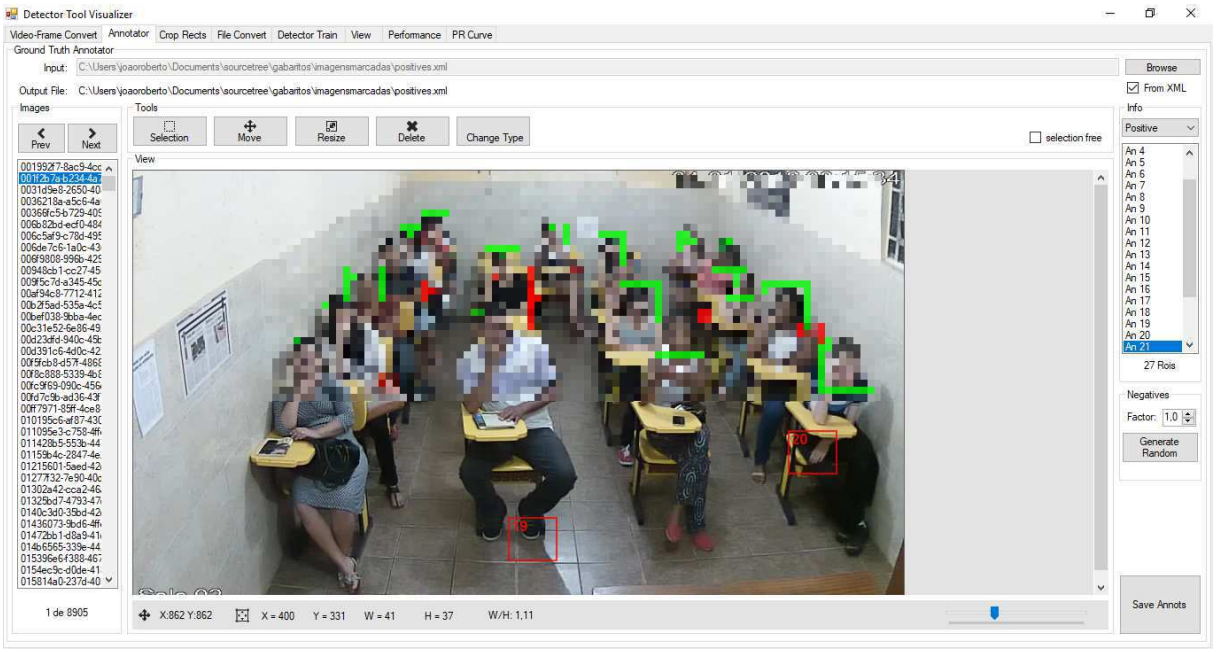
Os positivos correspondem às imagens de faces e foram obtidos marcando-se manualmente cada um dos 124.716 rostos das 8.919 imagens de sala de aula.

4.4.2 Marcação dos negativos

Os negativos correspondem às imagens de não-faces. A maior parte deles foi obtida da seguinte maneira. Primeiro foi treinada uma rede com os positivos e alguns negativos já presentes no banco de dados da Vsoft. Em seguida foi passado o banco de 8.919 imagens por essa rede e os resultados foram salvos. Por fim, usando o Detector Tool Visualizer

nos resultados salvos, os verdadeiros positivos (imagens de faces que foram corretamente detectadas pela rede) foram separados manualmente dos falso positivos (imagens detectadas pela rede que não são faces), restando assim os negativos.

Figura 14 – Detector Tool Visualizer



Fonte: *Print screen* do Detector Tool Visualizer

É ilustrado na Figura 14 a interface do Detector Tool Visualizer onde as imagens são gabaritadas. Os retângulos verdes correspondem às imagens de faces e os retângulos vermelhos correspondem às imagens de não-faces.

Uma vez que o banco de dados estava construído se deu início aos passos de identificar a melhor arquitetura para a rede, melhores valores para os parâmetros de limiar do *heatmap*, limiar do algoritmo de agrupamento de retângulos e melhores valores para redimensionamento da imagem de entrada.

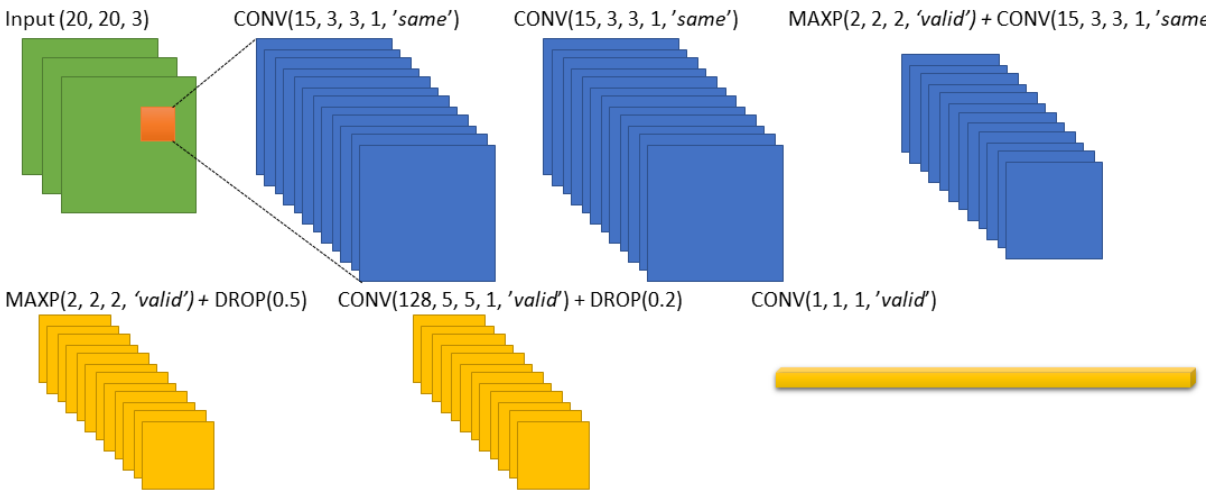
4.5 Definição da arquitetura

Para encontrar a arquitetura que apresentasse melhor performance com o banco de teste diversas redes diferentes foram treinadas e testadas. As arquiteturas testadas variam entre si no sentido de quantidade de camadas convolucionais, número de filtros em cada camada, o tamanho dos mesmos e a ordem de cada uma das camadas.

É preciso deixar claro que todas as arquiteturas testadas se assemelhavam no sentido de complexidade. Todas eram simples, pois, como mencionado anteriormente, é desejável que o tempo de processamento da rede seja pequeno.

A rede que apresentou melhor desempenho possui a arquitetura ilustrada pela Figura 15.

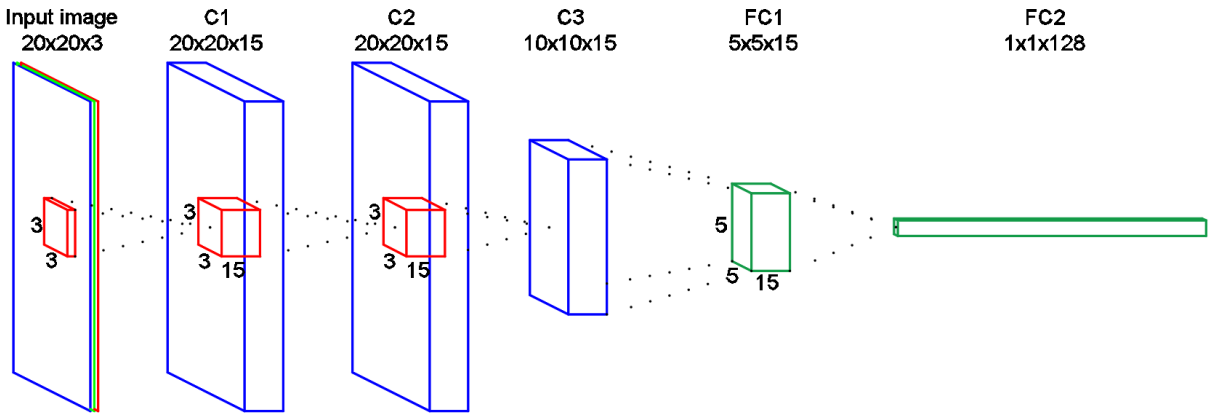
Figura 15 – Arquitetura 1



Fonte: Autoria prórpia

A interpretação dessa estrutura já foi discutida, o único ponto que vale mencionar é o parâmetro da camada DROPOUT, que corresponde à probabilidade de um sinal de um neurônio da camada anterior ser zero. Essa arquitetura é ilustrada graficamente na Figura 16

Figura 16 – Arquitetura mais eficiente



Fonte: Autoria própria

4.6 Pré-processamento

Durante a etapa de treinamento das redes, o pré-processamento consistia no redimensionamento das imagens de treino para 20×20. Já durante a etapa de teste, cada imagem de entrada era passada por uma rede em dois tamanhos diferentes, 400×640 e

200×320. Essas dimensões foram obtidas empiricamente como as dimensões que resultaram no melhor desempenho das redes.

4.7 Treinamento

Devido ao grande número de imagens do banco de treinamento, não foi possível fazer o treinamento da rede com todo banco de uma só vez. Ao invés disso o treinamento foi realizado com pequenos lotes de 128 imagens, sendo 64 imagens de faces e 64 de não-faces.

Por ser maior, o banco de não-faces foi tido como referência no treinamento. Uma vez que todas as imagens de não-faces fossem passadas para a rede diz-se que se completou uma época. Assim, o período de treinamento foi definido como o número de épocas desejadas. A rede que demonstrou melhor performance foi treinada com 13 épocas.

O algoritmo de treinamento utilizado foi o *Adam*, *Adaptive moment estimation*. O *Adam* é um algoritmo de otimização baseado em gradientes de primeira ordem de funções objetivas estocásticas, com base em estimativas adaptativas de momentos de ordem inferior. Resultados mostram que o *Adam* funciona e sugerem que ele supera outros métodos estocásticos de otimização, (KINGMA; BA, 2014).

4.8 Pós-processamento

A saída da rede neural convolucional treinada é uma matriz de valores em ponto flutuante, ou *heatmap*, de tamanho $W = \frac{W_i}{4} - 4$, em que W_i é o tamanho da imagem de entrada. Para as imagens de tamanho 400×640, a rede retorna uma matriz de dimensões 96×156, e para as imagens de tamanho 200×320, a rede retorna uma matriz de dimensões 48×78. Se faz necessário um pós-processamento para interpretar a saída da rede de forma a facilitar a visualização da resposta da rede e o seu desempenho.

O pós-processamento corresponde à utilização de um limiar para definir o que é considerado face ou não, chamado de limiar de confiança, seguido de uma representação gráfica, por meio de retângulos em torno de qual parte da imagem original foi considerada como face e por fim, um algoritmo de supressão de retângulos.

O limiar de confiança para se considerar um elemento do *heatmap* como face foi determinado como 0,9.

Para cada elemento do *heatmap* maior ou igual a 0,9 foi criado um retângulo onde sua localização na imagem original é proporcional a sua localização no *heatmap*, e seu tamanho é proporcional à relação de tamanhos da imagem de entrada com a matriz de saída da rede.

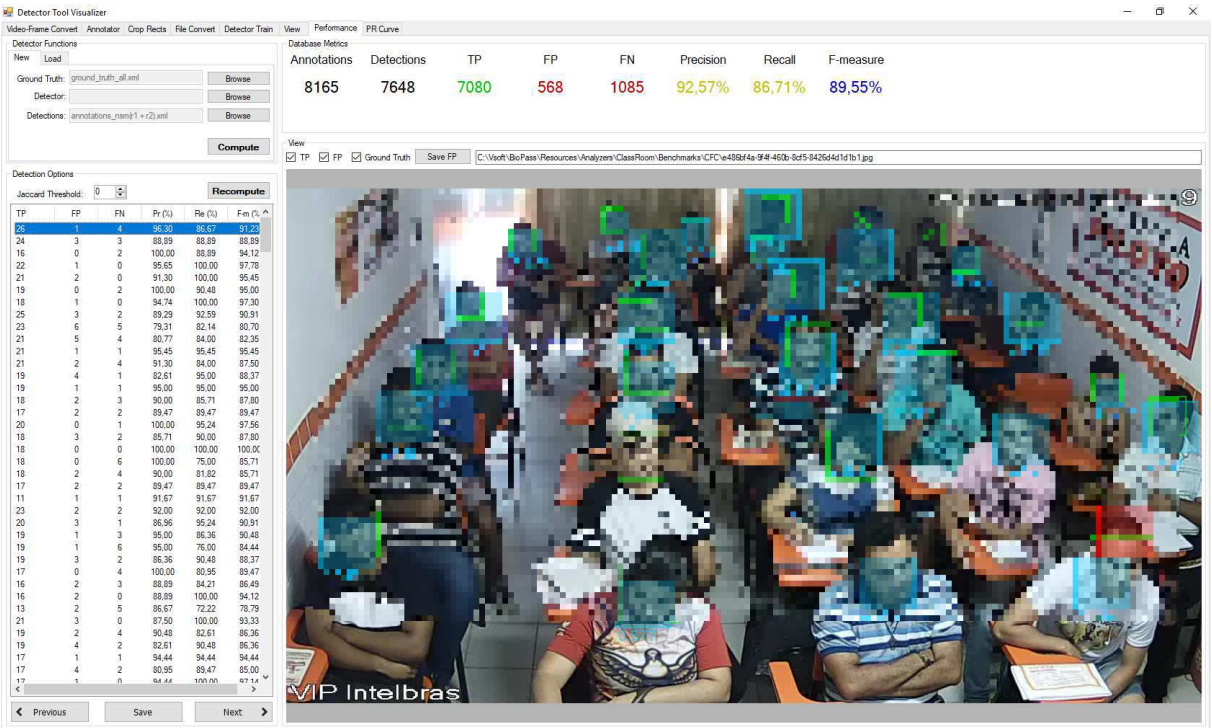
Acontece muitas vezes que uma mesma face é apontada por diversos retângulos,

ligeiramente transladados, uns dos outros de forma que se faz necessário um algoritmo de supressão de retângulos. Essa supressão é feita em dois passos, uma eliminação dos retângulos grandes e um NMS, *Non-Maximum Suppression*. A eliminação dos retângulos grandes acontece quando um retângulo maior contém um retângulo menor. O NMS faz com que quando existem dois retângulos cuja intersecção relativa excede um determinado limiar relativo um deles é eliminado. O cálculo da intersecção relativa é feito dividindo o valor da área da intersecção pela área de um dos retângulos, e o limiar de intersecção usado foi 0,38.

4.9 Análise dos resultados

A análise dos resultados foi feita com o auxílio do *software* Detector Tool Visualizer na aba *Performance*, Figura 17.

Figura 17 – Análise da performance da rede



Fonte: *Print screen* do Detector Tool Visualizer / Performance

Na Figura 17 é ilustrado os dados relevantes para avaliar o desempenho da rede, presentes no canto superior da imagem (*Annotations, Detections*, TP, FP, FN, *Precision, Recall, F-measure*). *Annotations* corresponde ao número de faces, do banco de teste, que foram gabaritadas previamente. *Detections* corresponde ao número de faces que foram acusadas pela rede. TP, *true positives*, corresponde ao número de faces acusadas pela rede que coincidem com faces gabaritadas. FP, *false positives*, corresponde ao número de faces acusadas pela rede que não estão gabaritadas. FN, *false negatives*, corresponde ao

número de faces gabaritadas que não foram identificadas pela rede. *Precision* corresponde à precisão da rede, que é definida como

$$Precisao = \frac{TP}{TP + FP}$$

Recall corresponde à revocação da rede, que é definida como

$$Revocacao = \frac{TP}{TP + FN}$$

F-measure é uma medida que combina os valores de precisão de revocação, de forma que foi tomada como melhor valor para análise do desempenho da rede, e é definida como

$$F_{measure} = 2 \times \frac{Recall \times Precision}{Recall + Precision}$$

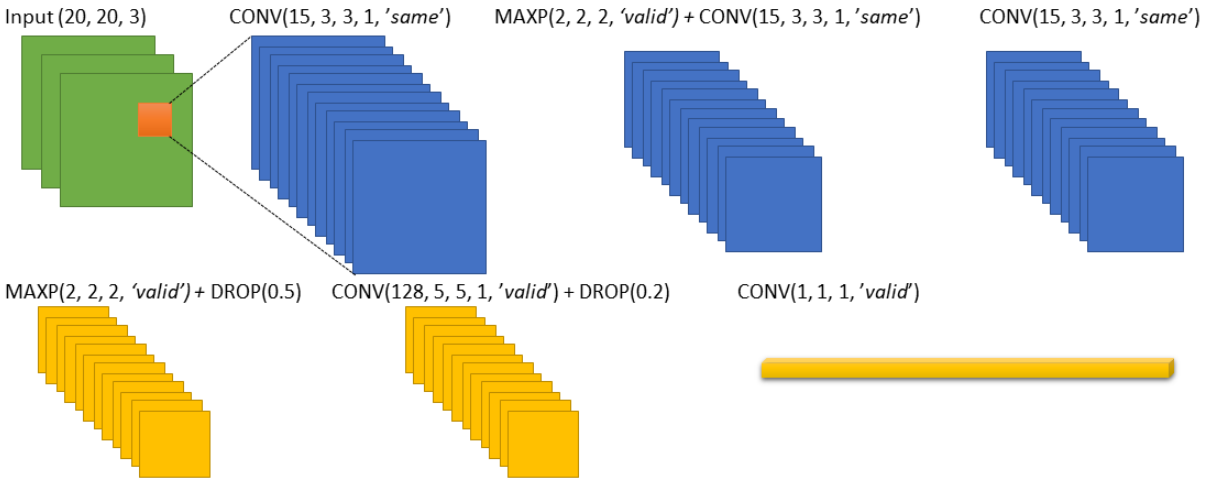
.

No canto esquerdo da Figura 17 se encontra uma lista de todas as imagens do banco de teste junto com os dados TP, FP, FN, *Precision, Recall* e *F-measure*. Na parte central da imagem é ilustrado graficamente o resultado da rede para cada imagem selecionada no canto esquerdo. Os retângulos verdes correspondem às faces gabaritadas. Os retângulos azuis correspondem aos resultados da rede que se encaixam no grupo TP, *true positives*, e os retângulos vermelhos correspondem aos resultados da rede que se encaixam no grupo FP, *false positives*.

Dentre as arquiteturas testadas, pode-se destacar três:

- Arquitetura 1, ilustrada na Figura 15
- Arquitetura 2, ilustrada na Figura 18

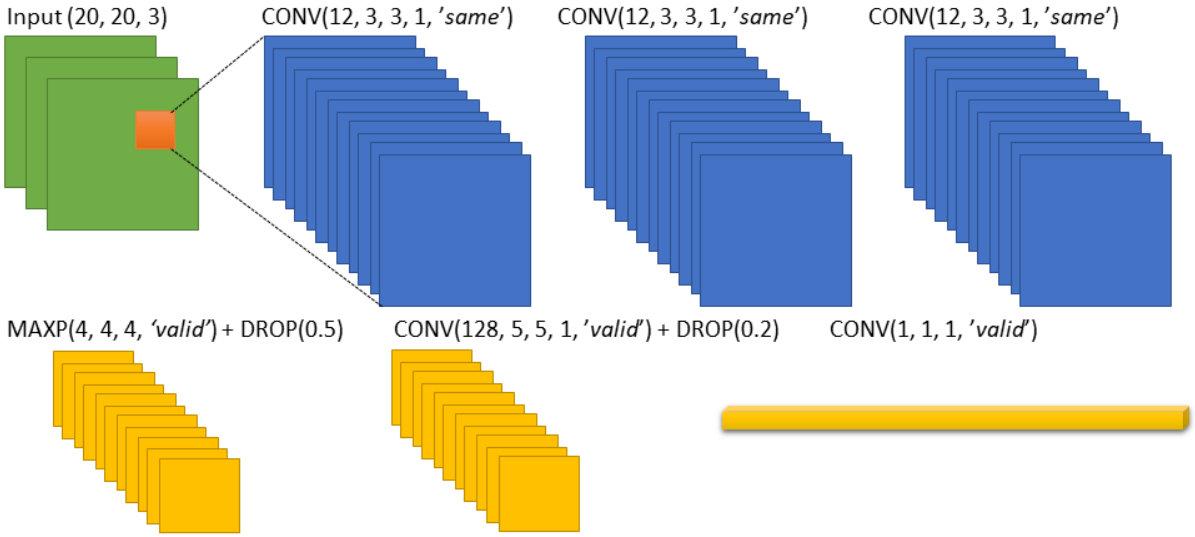
Figura 18 – Arquitetura 2



Fonte: Autoria prórpia

- Arquitetura 3, ilustrada na Figura 19

Figura 19 – Arquitetura 3



Fonte: Autoria prórpia

Diversas redes diferentes foram treinadas e testadas. Os resultados obtidos com alguns dos testes realizados estão presentes nas tabelas a seguir.

É ilustrado na Tabela 1 resultados obtidos com uma rede treinada segundo a Arquitetura 1. A primeira coluna indica o número de épocas com qual a rede foi treinada. A segunda coluna mostra o limiar de confiança utilizado. A terceira coluna indica as duas escalas usadas para cada imagem do banco de teste. A quarta coluna indica o limiar usado pelo algoritmo de NMS. Em seguida são mostrados os valores percentuais de precisão, revocação e *F-measure*, respectivamente.

Tabela 1 – Análise de épocas para a Arquitetura 1

#epo.	**conf.**	**limiar**	**Precisão**	**Revocação**	***F-measure***
2	0,99	0,3	80,65%	68,42%	74,03%
3	0,99	0,3	88,54%	63,85%	74,19%
4	0,99	0,3	80,01%	76,69%	78,31%
5	0,99	0,3	92,85%	63,56%	75,46%
6	0,99	0,3	78,02%	79,33%	78,67%
7	0,99	0,3	83,08%	76,91%	79,88%
8	0,99	0,3	84,50%	76,53%	80,32%
9	0,99	0,3	87,23%	73,73%	79,92%
10	0,99	0,3	83,68%	77,08%	80,25%
11	0,99	0,3	77,53%	81,16%	79,31%
12	0,99	0,3	87,38%	75,57%	81,05%
13	0,99	0,3	75,15%	82,87%	78,82%

As mesmas análises foram feitas para as arquiteturas 2 e 3 e são mostradas nas tabelas 2 e 3, respectivamente.

Tabela 2 – Análise de épocas para a Arquitetura 2

#epo.	**conf.**	**limiar**	**Precisão**	**Revocação**	***F-measure***
6	0,99	0,3	80,42%	74,84%	77,53%
7	0,99	0,3	77,64%	74,79%	76,19%
8	0,99	0,3	83,10%	75,86%	79,31%
9	0,99	0,3	81,10%	72,72%	76,68%
10	0,99	0,3	67,96%	82,49%	74,52%
11	0,99	0,3	84,24%	72,86%	78,14%
12	0,99	0,3	59,70%	84,75%	70,05%
13	0,99	0,3	63,75%	83,59%	72,33%

Tabela 3 – Análise de épocas para a Arquitetura 3

#epo.	**conf.**	**limiar**	**Precisão**	**Revocação**	***F-measure***
8	0,99	0,3	62,45%	86,25%	72,45%
9	0,99	0,3	80,24%	79,78%	80,01%
10	0,99	0,3	79,30%	79,25%	79,28%
11	0,99	0,3	86,17%	77,08%	81,37%

É possível observar pelas Tabelas 1, 2 e 3 que o *F-measure* aumenta com o número de épocas com a qual a rede é treinada, entretanto, após um certo número de épocas o rendimento da rede não aumenta e passa a oscilar.

Pode-se observar também que a rede treinada com a Arquitetura 1 apresentou desempenho melhor do que a rede treinada com a Arquitetura 2, chegando a 81% de *F-measure*, mas mostrou um desempenho levemente pior que a treinada com a Arquitetura 3.

A Arquitetura 1 foi escolhida, pois a rede treinada com Arquitetura 3 necessitou de um maior custo computacional para operar. Isso se deve ao fato de existirem três camadas convolucionais antes de o redimensionamento dado pela camada *pooling*.

A Tabela 4 contém resultados dos testes realizados para definir a melhor combinação de duas escalas para as imagens de entrada. Uma rede de arquitetura arbitrária foi treinada, os parâmetros de pós-processamento foram fixados com valores arbitrários e a rede foi testada diversas vezes com imagens do banco de teste. Para cada teste duas escalas diferentes foram usadas para cada imagem de teste. Como é possível observar na Tabela 4, o melhor resultado ocorreu com as escalas 400×640 e 200×320.

Tabela 4 – Análise de escalas

Escalas	**Precisão**	**Revocação**	***F-measure***
608x1080, 400x640	49,58%	69,65%	57,92%
450x880, 400x640	45,82%	69,22%	61,18%
450x880, 300x480	60,67%	76,68%	67,74%
400x640, 300x480	68,83%	76,76%	72,58%
430x764, 300x480	63,87%	76,76%	69,73%
400x640, 200x320	72,96%	79,82%	76,24%
430x764, 200x320	68,11%	78,19%	72,80%
430x764, 215x382	67,17%	78,66%	72,47%

Vários testes semelhantes aos testes mencionados anteriormente foram feitos para definição do limiar de confiança e do limiar dos algoritmos de agrupamento de retângulos que gerariam o sistema mais eficiente.

A Tabela 5 contém os resultados das melhores redes obtidas após o treinamento, por ordem de *F-measure*. Elá está estruturada de forma semelhante às tabelas anteriores com exceção da terceira coluna, que indica os algoritmos de supressão de retângulos utilizados e seus limiares.

Como apresentado na Tabela 5, a rede que mostrou melhor eficiência foi treinada com 13 épocas, usou-se um limiar de confiança de 0,90, um limiar de 0,38 para os algoritmos de supressão de retângulos e um *F-measure* de 89,55%.

Tabela 5 – *Ranking* das melhores redes

#epo.	**conf.**	**Agrup.**	**Precisão**	**Revocação**	***F-measure***
13	0,90	nms + big_rects (0,38)	92,57%	86,71%	89,55%
12	0,87	nms + big_rects (0,48)	92,74%	86,18%	89,34%
10	0,87	nms + big_rects (0,48)	94,46%	84,21%	89,04%
8	0,91	nms + big_rects (0,46)	93,81%	84,36%	88,84%
8	0,90	nms + big_rects (0,25)	93,58%	84,26%	88,67%
8	0,90	nms (0,3)	94,25%	82,69%	88,09%
8	0,90	nms (0,3)	95,47%	81,36%	87,85%

5 Conclusão

Durante este estágio foi possível entrar em contato com um ambiente profissional que usa sistemas baseados em *Deep learning* para detecção facial e aplicar os conhecimentos de Redes Neurais e programação adquiridos durante a graduação.

O estágio consistiu na realização de todas as etapas necessárias para o desenvolvimento de um sistema de detecção de faces baseado em redes neurais convolucionais. Iniciando com a criação de bacos de dados para treinamento, validação e teste, seguido da escolha de uma arquitetura para a rede, mais tarde a elaboração de estratégias de treinamento e de algoritmos de pós-processamento e pré-processamento e finalizando com testes e ajustes dos parâmetros dos algoritmos desenvolvidos até ser obtida uma rede de desempenho satisfatório.

Referências

FELZENSZWALB, P. F. et al. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 32, n. 9, p. 1627–1645, 2010. Citado na página 12.

HUANG, G. B. et al. *Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments*. [S.l.], 2007. Citado na página 29.

IOFFE, S.; SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: *International conference on machine learning*. [S.l.: s.n.], 2015. p. 448–456. Citado na página 26.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. Citado na página 35.

SIMON, H. *Redes Neurais: Princípios e Prática*. [S.l.]: Bookman, 2008. ISBN 0132733501. Citado 2 vezes nas páginas 15 e 19.

VIOLA, P.; JONES, M. J. Robust real-time face detection. *International journal of computer vision*, Springer, v. 57, n. 2, p. 137–154, 2004. Citado na página 12.

VSOFT Tecnologia. <https://www.vsoft.com.br/>. Accessed: 2018-02-20. Citado na página 14.

YANG, S. et al. Wider face: A face detection benchmark. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2016. Citado na página 29.