UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DAELN - DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
CURSO DE ENGENHARIA ELETRÔNICA

ADRIANO CESAR ALVES DA CRUZ E FELIPE AUGUSTO SCHWAB

# CARACTERIZAÇÃO DE MICRO VESTÍGIOS ENTOMOLÓGICOS POR IMAGEM: APLICAÇÃO DE REDES NEURAIS PROFUNDAS

TRABALHO DE CONCLUSÃO DE CURSO

CURITIBA
2019

ADRIANO CESAR ALVES DA CRUZ E FELIPE AUGUSTO SCHWAB

# CARACTERIZAÇÃO DE MICRO VESTÍGIOS ENTOMOLÓGICOS POR IMAGEM: APLICAÇÃO DE REDES NEURAIS PROFUNDAS

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia Eletrônica da Universidade Tecnológica Federal do Paraná, como requisito parcial para a obtenção do título de Bacharel em Engenharia Eletrônica.

Orientador: Valfredo Pilla Jr.
UTFPR

Coorientador: Rubens Alexandre de Faria
UTFPR

CURITIBA
2019

ADRIANO CESAR ALVES DA CRUZ
FELIPE AUGUSTO SCHWAB

# CARACTERIZAÇÃO DE MICRO VESTÍGIOS ENTOMOLÓGICOS POR IMAGEM: Aplicação de redes neurais profundas

Este Trabalho de Conclusão de Curso de Graduação foi apresentado como requisito parcial para obtenção do título de Engenheiro Eletrônico, do curso de Engenharia Eletrônica do Departamento Acadêmico de Eletrônica (DAELN) outorgado pela Universidade Tecnológica Federal do Paraná (UTFPR). Os alunos foram arguidos pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

Curitiba, 08 de julho de 2019.

_____
Prof. Dr. Robinson Vida Noronha
Coordenador de Curso
Engenharia Eletrônica

_____
Profª. Drª. Carmen Caroline Rasera
Responsável pelos Trabalhos de Conclusão de Curso
de Engenharia Eletrônica do DAELN

**BANCA EXAMINADORA**

| | |
|---|---|
| Prof. Me. Valfredo Pilla Jr.<br>Universidade Tecnológica Federal do Paraná<br>Orientador | Prof. Dr. Gustavo Benvenutti Borba<br>Universidade Tecnológica Federal do Paraná |
| Prof. Dr. Rubens Alexandre de Faria<br>Universidade Tecnológica Federal do Paraná<br>Coorientador | Prof. Dr. Luís Alberto Lucas<br>Universidade Tecnológica Federal do Paraná |

A folha de aprovação assinada encontra-se na Coordenação do Curso de Engenharia Eletrônica.

# AGRADECIMENTOS

Aos familiares e amigos pelo amor, incentivo e apoio incondicional durante toda essa jornada.

A esta universidade, seu corpo docente, direção e administração por propiciar um ambiente criativo e adequado aos estudos.

Ao professor Doutor Gustavo Benvenutti Borba por apresentar e instigar o interesse em processamento digital de imagens e redes neurais convolucionais.

Ao coorientador e professor Doutor Rubens Alexandre de Faria por apresentar a necessidade da criação de um novo processo em ciência forense.

À professora Doutora Ozana Andrade Maia pela orientação de conceitos básicos de entomologia.

Ao orientador e professor Mestre Valfredo Pilla Júnior por toda orientação e compartilhamento de conhecimento sobre redes neurais, pelas correções e suporte durante toda a execução deste trabalho.

E a todos que direta ou indiretamente fizeram parte da nossa formação, o nosso muito obrigado.

*"Ocasionalmente, deixe a trilha já traçada e mergulhe na floresta. Cada vez que fizer isso, certamente encontrará algo que você nunca viu antes." Alexander Graham Bell.*

# RESUMO

Caracterização de micro vestígios entomológicos por imagem: Aplicação de redes neurais profundas. 2019. 62 f. Trabalho de Conclusão de Curso – Curso de Engenharia Eletrônica, Universidade Tecnológica Federal do Paraná. Curitiba, 2019.

Análise de micro vestígios é uma nova e crescente área de pesquisa e atuação dentro da ciência forense, podendo ser vital na solução de diversos casos criminais, auxiliando e acelerando na identificação de criminosos e áreas em que os mesmos atuam. Atualmente a área de entomologia forense vem sendo cada vez mais utilizada na caracterização de insetos, de modo a apurar a origem de carregamentos de entorpecentes, malotes de dinheiro ou até mesmo cadáveres. No entanto este ainda é um processo lento e complexo, principalmente pelo fato dos espécimes serem encontrados, em sua maioria, aos pedaços. Visando melhorar esse processo, o trabalho a seguir apresenta a utilização de uma rede neural profunda para a caracterização, até o nível taxonômico de família, de insetos ou pedaços de insetos. A aplicação do sistema a ser descrito apresentou uma acurácia de 77,77% entre as cinco famílias entomológicas consideradas: Buprestidade, Formicidae, Pentatomidae, Muscidae e Calliphoridae.
**Palavras-chave**: Redes neurais profundas. Ciência forense. Micro vestígios. Entomologia. Insetos.

## ABSTRACT

Characterization of micro traces entomological imaging: Application of deep neural networks. 2019. 62 f. Trabalho de Conclusão de Curso – Curso de Engenharia Eletrônica, Universidade Tecnológica Federal do Paraná. Curitiba, 2019.

Micro-traces analysis is a new and growing area of research within forensic science and may be vital in solving lots of criminal cases, assisting and accelerating the identification of criminals and areas in which they act. Nowadays the forensic entomology area has been increasingly used for insects characterization, in order to ascertain the origin of loads of narcoticts, money pouches or even corpses. However, this is still a demanding process, mainly because of the fact that specimens are mostly found in pieces. In order to improve this process, this thesis presents the use of a deep neural network for characterization of insects or pieces of it, up to the taxonomic level of family. The application of the system showed an accuracy of 77,77% among five families: Buprestidae, Formicidae, Pentatomidae, Muscidae and Calliphoridae.


**Keywords**: Deep neural Networks. Forensic Science. Micro-traces. Entomology. Insects.

## LISTA DE FIGURAS


Figura 1 – Níveis dos Táxons . . . . . 15
Figura 2 – Modelo matemático genérico de um nó . . . . . 16
Figura 3 – Diferença entre redes neurais convencionais e redes neurais convolucionais . 17
Figura 4 – Filtros na camada de convolução . . . . . 18
Figura 5 – Passo dos filtros na camada de convolução . . . . . 19
Figura 6 – Gráfico função ReLU . . . . . 20
Figura 7 – Tipos de operação *Pooling* . . . . . 21
Figura 8 – Operação *Max Pooling* na dimensão de profundidade . . . . . 21
Figura 9 – Transição da camada de convolução para totalmente conectada . . . . . 22
Figura 10 – Comparação de redes neurais convolucionais . . . . . 23
Figura 11 – *Inception* módulo A . . . . . 25
Figura 12 – Redução da área do filtro . . . . . 25
Figura 13 – *Inception* módulo B . . . . . 26
Figura 14 – *Inception* módulo C . . . . . 26
Figura 15 – *Inception* redução da imagem . . . . . 27
Figura 16 – Hiperplano de separação otimizado . . . . . 28
Figura 17 – Hiperplano em 2D e 3D . . . . . 28
Figura 18 – Transformação de pontos de treino . . . . . 29
Figura 19 – Comparação parâmetro C . . . . . 30
Figura 20 – Exemplo variação parâmetro *C* . . . . . 30
Figura 21 – Parâmetro *gamma* . . . . . 31
Figura 22 – Amostra da Família *Pentatomidae* Vista Superior. . . . . 32
Figura 23 – Arquitetura do sistema. . . . . 33
Figura 24 – Inseto da Família Buprestidae. . . . . 33
Figura 25 – Inseto da Família Formicidae. . . . . 34
Figura 26 – Inseto da Família Muscidae. . . . . 34
Figura 27 – Inseto da Família Calliphoridae. . . . . 34
Figura 28 – Inseto da Família Pentatomidae. . . . . 35
Figura 29 – Imagem de entrada, com fundo branco e dimensões 2756x2756x3. . . . . 35
Figura 30 – Máscara de pixels (*Grayscale*) para *Lower Bound* = [0,0,0] e *Upper Bound* = [250,250,250]. . . . . 36
Figura 31 – Imagem após operação *not* com a máscara de bordas. . . . . 36
Figura 32 – Máscara final contendo o objeto de interesse. . . . . 37
Figura 33 – Imagem com inseto separado do fundo branco. . . . . 37
Figura 34 – Imagem com dimensões (X,Y) iguais. . . . . 38
Figura 35 – Imagem reduzida de 250x250x3. . . . . 38

Figura 36 – Imagem final de 299x299x3. . . . . 39
Figura 37 – Imagem de dimensões 250x250x3. . . . . 40
Figura 38 – Imagens de *Data Augmentation* . . . . . 41
Figura 39 – Vantagens da implementação de *Transfer Learning* . . . . . 42
Figura 40 – Comparação entre classificadores . . . . . 43
Figura 41 – Divisão dos conjuntos de dados . . . . . 44
Figura 42 – Validação Cruzada . . . . . 45
Figura 43 – Matriz Confusão Exemplo . . . . . 46
Figura 44 – Gráfico *Log Loss* . . . . . 48
Figura 45 – Comparação dos Classificadores . . . . . 49
Figura 46 – Matriz Confusão após treino com *Data Augmentation* de 5 variantes . . . . 50
Figura 47 – Variáveis de Desempenho com *Data Augmentation* de 5 variantes . . . . . 50
Figura 48 – Matriz Confusão após treino com *Data Augmentation* de 10 variantes . . . 51
Figura 49 – Variáveis de Desempenho com *Data Augmentation* de 10 variantes . . . . 51
Figura 50 – Matriz Confusão após treino com *Data Augmentation* de 30 variantes . . . 52
Figura 51 – Variáveis de Desempenho com *Data Augmentation* de 30 variantes . . . . 52
Figura 52 – Teste com Insetos da Família Buprestidae. . . . . 53
Figura 53 – Teste com Insetos da Família Formicidae. . . . . 53
Figura 54 – Teste com Insetos da Família Muscidae. . . . . 54
Figura 55 – Teste com Insetos da Família Calliphoridae. . . . . 54
Figura 56 – Teste com Insetos da Família Pentatomidae. . . . . 54
Figura 57 – Teste com Insetos da Família Pentatomidae Deteriorados - 1. . . . . 55
Figura 58 – Teste com Insetos da Família Pentatomidae Deteriorados - 2. . . . . 55
Figura 59 – Teste com Insetos Deteriorados. . . . . 55

## LISTA DE TABELAS

Tabela 1 – Arquitetura Inception v3 . . . . . 24
Tabela 2 – Número de imagens por família entomológica . . . . . 33
Tabela 3 – Acurácia Média e *Log Loss* . . . . . 53

## LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| FC | Fully Connected |
| IPM | Intervalo *Post Mortem* |
| ReNEF | Rede Nacional de Entomologia Forense |
| SVM | Máquina de vetores de suporte |
| SVC | Classificador de vetores de suporte |
| CNN | Rede neural convolucional |
| SQL | Structured Query Language |

## SUMÁRIO


**1 – INTRODUÇÃO . . . . . 12**

**2 – FUNDAMENTAÇÃO TEÓRICA . . . . . 13**

2.1 CIÊNCIA FORENSE . . . . . 13

2.1.1 ENTOMOLOGIA FORENSE . . . . . 13

2.2 REDES NEURAIS PROFUNDAS . . . . . 15

2.2.1 REDES NEURAIS CONVOLUCIONAIS . . . . . 16

2.2.1.1 CAMADA CONVOLUCIONAL . . . . . 17

2.2.1.2 CAMADA *POOLING* . . . . . 20

2.2.1.3 CAMADA TOTALMENTE CONECTADA . . . . . 21

2.2.2 INCEPTION v3 . . . . . 22

2.3 CLASSIFICADOR SVM . . . . . 27

**3 – METODOLOGIA . . . . . 32**

3.1 PROCESSAMENTO DE IMAGENS . . . . . 35

3.2 DATA AUGMENTATION . . . . . 39

3.3 TRANSFER LEARNING . . . . . 41

3.4 SISTEMA DE CLASSIFICAÇÃO . . . . . 42

3.5 TREINAMENTO . . . . . 44

**4 – RESULTADOS . . . . . 46**

4.1 INDICADORES DOS RESULTADOS . . . . . 46

4.2 RESULTADOS . . . . . 48

4.2.1 RESULTADOS DO TREINAMENTO . . . . . 49

4.2.2 RESULTADOS TESTES . . . . . 53

4.3 DISCUSSÃO . . . . . 56

**5 – CONCLUSÃO . . . . . 58**

5.1 TRABALHOS FUTUROS . . . . . 58

**Referências . . . . . 60**

# 1 INTRODUÇÃO

A ciência forense visa auxiliar investigações, no âmbito criminal, utilizando-se de técnicas com embasamento e comprovação científica, como por exemplo, testes de DNA, reconhecimento de impressões digitais, etc. Métodos cada vez mais sofisticados vem sendo utilizados para examinar provas criminais, buscando também, o aperfeiçoamento dos procedimentos já aplicados (CALAZANS; CALAZANS, 2005). Um campo de pesquisa dentro das ciências forenses é o de análise de micro vestígios, que são vestígios microscópicos, como grãos de polens, pedaços de insetos e gotículas de sangue (ZIęBA-PALUS, 2017). Analisar essas amostras acaba sendo uma tarefa complexa e que demanda muito tempo devido ao tamanho e estado de conservação das amostras. No âmbito dos vestígios entomológicos a dificuldade vem também da grande quantidade de espécies a serem identificadas e de encontrar profissionais especializados na identificação de cada grupo (PUJOL-LUZ; ARANTES; CONSTANTINO, 2008). Ainda não há, no Brasil, uma forma automatizada de caracterização de insetos, dificultando o aproveitamento de amostras, consequentemente alongando e por vezes inviabilizando investigações.

Atualmente o processo de caracterização de vestígios entomológicos, dentro da Polícia Federal do Brasil, é feito basicamente em três etapas, coleta, triagem e detalhamento de espécie. A primeira etapa é feita em campo, geralmente pelo próprio agente ou perito. Esse espécime, inteiro ou em partes, é então entregue a um profissional, normalmente um biólogo, que fará a triagem do inseto. A triagem é feita a partir do uso de chaves dicotômicas e visa encontrar a família taxonômica do espécime coletado. Tendo definido a família, o inseto é então levado a um especialista, daquela família entomológica, para que seja então identificada a espécie do inseto. Apenas então com essa informação é que o perito e profissionais da investigação podem começar a traçar as características desta espécie para auxiliar nas investigações. Esse processo é bastante lento e complexo, por vezes, sendo impossível a caracterização do exemplar.

Redes neurais convolucionais tem ganhado bastante espaço na área de caracterização de objetos por imagem. Apesar de ser um campo ainda em desenvolvimento já existem diversas aplicações em que tais redes obtiveram excelente sucesso. O trabalho em questão busca apresentar uma nova aplicação para tal tipo de rede, visando auxiliar na segunda fase, triagem, do processo de análise entomológica forense. O método proposto neste trabalho utiliza uma rede neural profunda para realizar a caracterização de micro vestígios entomológicos, identificando automaticamente, por meio de fotografia, o nível taxonômico de família. Automatizando-se esta identificação a fase de triagem poderia ser eliminada, o que possibilitaria o encaminhamento direto do exemplar ao especialista para identificação da espécie.

# 2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são abordados, inicialmente, conceitos básicos a respeito de ciência forense e entomologia forense, sendo feita uma contextualização básica. Em seguida são explicados conceitos sobre redes neurais profundas, aprofundando-se em redes neurais convolucionais. Na seção sobre a arquitetura *Inception V3*, são mostradas suas principais características, bem como as camadas que a constituem. Por fim, é apresentado o classificador SVM, com uma breve explicação a respeito deste método de classificação.

## 2.1 CIÊNCIA FORENSE

A ciência forense é a área de estudos que visa auxiliar na resolução de investigações nos âmbitos civil e criminal. É uma ciência que envolve diferentes áreas como a física, química, biologia, matemática e muitas outras ciências de fronteira (CHEMELLO, 2006). As técnicas implementadas estão voltadas para a confirmação da presença ou não do suspeito no local do crime, tempo de *post mortem*, local e arma utilizada no crime, entre outros.

No início da ciência forense, não existiam profissionais especializados na área criminal, assim a justiça contava com profissionais de áreas mais abrangentes, como biólogos e químicos, para auxiliar nas investigações. Porém, com o passar do tempo os crimes começaram a aumentar de complexidade e então foi necessário que profissionais especializados fossem formados para solucionar casos cada vez mais desafiadores (CALAZANS; CALAZANS, 2005).

Uma grande parte das técnicas utilizadas pela ciência forense vem da coleta e análise de vestígios físicos das cenas dos crimes. Vestígios como impressões digitais, fios de cabelo, resquícios de sangue ou até mesmo insetos podem ser utilizados para relacionar e/ou identificar o suspeito ao local do crime. Dentre estas diversas possibilidades o trabalho em questão foca nos vestígios entomológicos, que estão ligados à identificação e caracterização de insetos.

### 2.1.1 ENTOMOLOGIA FORENSE

A entomologia forense é um ramo da ciência forense responsável pelo estudo da taxonomia, biologia e ecologia de insetos e outros artrópodes voltados a questões criminais (CATTS; GOFF, 1992). Pode ser aplicada em investigações de maus tratos, danos imobiliários, tráfico de entorpecentes e vítimas de mortes violentas.

A entomologia forense surge, em seu primeiro registro, na China por Sung Tzu, no século XII, no caso de uma morte por golpes de foice. Após um dia do crime, Tzu, advogado e investigador, pediu para que os trabalhadores que trabalhavam próximos ao local do crime, colocassem suas foices no chão, fazendo com que moscas fossem atraídas pelo odor de sangue até a arma do crime, levando o proprietário a confessar o crime (LEITE; Sá, 2010).

Ao longo dos anos a entomologia forense foi avançando, com a primeira estimativa de *post mortem* em 1855 pelo médico francês Bergeret D'Arboois ao estudar larvas e ovos de moscas presentes no corpo de uma criança que fora coberta por uma camada de gesso sobre o piso de uma residência, realizando a associação da fauna necrófaga encontrada e a decomposição do cadáver . Seus estudos, porém, não foram suficientes para impulsionar a expansão da área, principalmente, devido ao distanciamento entre entomólogos e profissionais da criminalística. No ano 1894, através da publicação do livro *La faune des cadavres* de Megin na França, o estudo desta ciência passa a ser retomado, adquirindo uma maior popularização no âmbito mundial com o surgimento de grupos de pesquisa, especialmente na América do Norte e Europa, no fim do século XX (LEITE; Sá, 2010).

No Brasil, a entomologia forense está ligada ao trabalho de Oscar Freire, que em 1998, apresenta a sociedade médica da Bahia a primeira coleção de insetos necrófagos, juntamente com os resultados de investigações obtidas com cadáveres humanos e pequenos animais. De acordo com Freire, o Brasil é o país que possui a maior biodiversidade no mundo, refletindo, também na fauna associada a cadáveres. Além disso, cada bioma possui sua fauna e condições próprias, sendo necessários estudos regionais, limitando a entomologia forense no Brasil, uma vez que técnicas desenvolvidas em outros países nem sempre podem ser aplicadas diretamente no país (PUJOL-LUZ; ARANTES; CONSTANTINO, 2008).

O ramo de entomologia forense é classificado em três subáreas: urbana, produto estocado e médico-legal. A primeira área diz respeito a presença de insetos em imóveis ou estruturas, a segunda faz referência à contaminação de reservatórios de alimentos e de artigos armazenados. Por último, a médico-legal, sendo a mais popularizada, trata de casos de morte violenta e da investigação de negligência por maus tratos, buscando, principalmente, determinar o intervalo *post mortem* (IPM) através da dedução das circunstâncias ocorridas antes e depois do crime, como por exemplo, se o corpo foi movido de um local para outro e se houve interferência de animais ou do próprio criminoso (PUJOL-LUZ; ARANTES; CONSTANTINO, 2008).

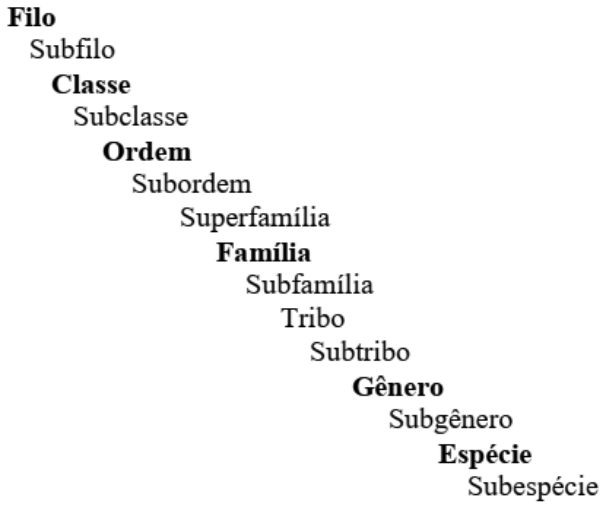
Um dos maiores problemas para a expansão da utilização da entomologia forense na solução de crimes no Brasil é o distanciamento entre estudiosos da área e profissionais da Polícia Judiciária. Atualmente, conhecimentos produzidos por entomologistas, peritos criminais e médicos legistas são escassos, sendo de grande importância para o desenvolvimento da entomologia forense no Brasil. Reconhecendo a necessidade de investimentos nesta área, o governo brasileiro, em 2008, criou a Rede Nacional de Entomologia Forense (ReNEF), sendo um grupo constituído por cinco pesquisadores e cinco peritos criminais dos estados do Amapá, Amazonas, Bahia, São Paulo, Rio de Janeiro, Paraná e Distrito Federal. Este grupo foi criado de modo a estimular o desenvolvimento de pequisas dentro da entomologia forense, fortalecer a cooperação entre acadêmicos e Polícia Judiciária e padronizar métodos e técnicas (PUJOL-LUZ; ARANTES; CONSTANTINO, 2008).

Atualmente, dentro da Polícia Federal do Brasil, um agente ou perito faz a coleta

do espécime e o entrega a um biólogo, que fará a classificação do inseto. Dentro destas classificações, os animais são distribuídos de acordo com suas características, em grupos denominados táxons.

Na Figura 1 observam-se as principais categorias, em negrito, e as secundárias dos níveis em que os táxons são dispostos. Importante ressaltar que espécie é a categoria básica, relacionando uma população de animais de uma área, onde ocorre cruzamento e há reprodução, enquanto que subespécie é um agregado que possui características semelhantes dentro da categoria de espécie e que difere taxonomicamente de outras (LEITE; Sá, 2010).

Figura 1 – Níveis dos Táxons




Fonte: (LEITE; Sá, 2010)

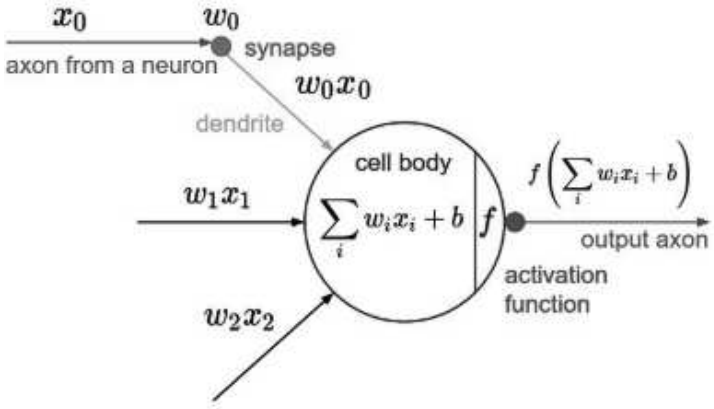
Para realizar esta classificação é necessário utilizar chaves dicotômicas, as quais são chaves de identificação que agrupam espécimes com características semelhantes, sendo que há uma para cada ordem, diferindo entre cada uma. Isto acaba por trazer mais um problema para a identificação dos mesmos, uma vez que há uma grande variedade de espécimes, além de que muitos são pequenos, dificultando a observação de características específicas.

## 2.2 REDES NEURAIS PROFUNDAS

O conceito de redes neurais não é algo novo, ele foi primeiro introduzido por McCulloch e Pitts em 1943. A ideia inicial dos autores era imitar o funcionamento dos neurônios em uma rede computacional. No cérebro, cada célula neural, de forma extremamente simplificada, recebe pulsos elétricos de diversos outros neurônios simultaneamente. Esses sinais são repassados através de sinapses, que determinam a força deste sinal. Ao chegar no corpo da célula estes sinais são somados e, através de uma lógica molecular, é decido se o neurônio será ativado ou não. De maneira equivalente, um modelo matemático foi desenvolvido que designa, aos sinais de entrada, pesos variantes, análogo às sinapses biológicas. Dentro dos chamados nós ou neurônios, faz-se o somatório destes sinais e então por meio de uma função de ativação

determina-se qual será a saída daquele nó. A Figura 2 representa esse processo, ilustrando como diversos sinais de entrada $x_0, x_1, x_2$ recebem pesos $w_0, w_1, w_2$ na "sinapse", são somados no "corpo celular" e através da função de ativação $f$ é definido qual será o sinal de saída (MCCULLOCH; PITTS, 1943).

Figura 2 – Modelo matemático genérico de um nó




Fonte: (LI JUSTIN JOHNSON, 2018)

Por muitos anos esse conceito foi aprofundado e desenvolvido, como o modelo ADALINE proposto por Bernard Widrow e Marcian Hoff em Standford em 1959. No entanto, devido às limitações em processamento computacional e em métodos estatísticos e matemáticos o avanço não foi muito significativo. Em 1986, com a invenção de novos métodos estatísticos o conceito voltou a chamar a atenção de pesquisadores. Atualmente, as limitações estão virtualmente abolidas e os métodos altamente refinados, desta forma, o conceito vem ganhando cada vez mais força e mais aplicações (KURENKOV, 2015).

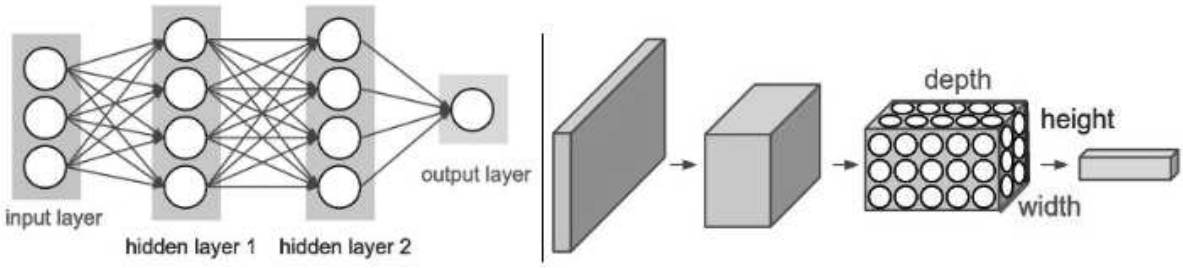
As redes neurais possuem diversos modelos e formatos, este trabalho trata das redes neurais profundas, mais especificamente das redes neurais convolucionais, que surgiram por uma necessidade de mudança no conceito básico de redes neurais profundas, onde todos os neurônios de uma camada estão relacionados com todos os neurônios da próxima.

### 2.2.1 REDES NEURAIS CONVOLUCIONAIS

Dentre os vários tipos existentes de redes neurais profundas as redes neurais convolucionais ou CNN foram projetadas para otimizar a resolução de problemas envolvendo imagens. Em uma rede neural convencional os valores, ou neurônios de entrada, conectam-se com todos os neurônios das camadas escondidas. Ao final tem-se uma última camada totalmente conectada que gera o vetor de neurônios de saída. Redes neurais convolucionais funcionam de forma diferente. Como é certo que a informação de entrada é uma imagem, os neurônios dessa arquitetura foram modificados para possuírem três dimensões: largura, comprimento e profundidade. Outra diferença é que os neurônios de uma camada não são totalmente

conectados aos da camada seguinte, e sim apenas à uma parte deles. Podemos notar essas diferenças na Figura 3 onde cada círculo branco representa um neurônio da camada.

Figura 3 – Diferença entre redes neurais convencionais e redes neurais convolucionais




Fonte: (LI JUSTIN JOHNSON, 2018)

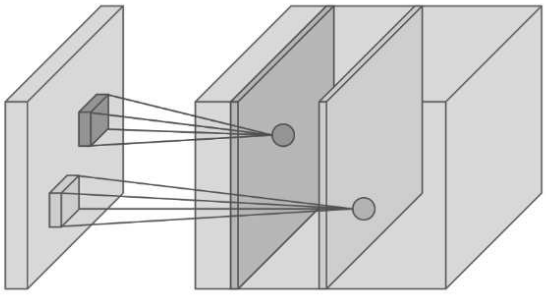
Outa parte bastante importante das CNN são os tipos de camadas que formam sua arquitetura, essas são: convolução, *pooling* e totalmente conectada. A camada mais importante é a de convolução que busca extrair e relacionar as características de cada parte da imagem de entrada. Já a camada de *Pooling* busca reduzir as dimensões da figura, para diminuir a complexidade computacional. Finalmente a camada totalmente conectada, que sempre é a última da rede, é utilizada para fazer a classificação final da imagem, isto é, transforma a matriz em um vetor que será posteriormente analisado para acessar a porcentagem referente a cada categoria. Cada um desses passos serão, ao decorrer desta secção, explicados.

#### 2.2.1.1 CAMADA CONVOLUCIONAL

Como foi citado anteriormente, seria inviável conectar todos os valores de entrada de uma imagem com diferentes neurônios, assim a camada de convolução busca apenas fazer a associação de uma pequena parte da imagem com cada neurônio. Esse processo é realizado através de um filtro de pesos, que terá uma dimensão arbitrária porém com profundidade sempre igual à da matriz de entrada. Por exemplo, se decide-se computar um filtro de dimensão 5x5 *pixels* e a imagem de entrada possui dimensão 200x200 *pixels* e é colorida, os filtros da primeira camada de convolução da rede neural terão o formato 5x5x3 *pixels* (sendo a última dimensão relativa às três cores da imagem).

É realizada a convolução dos filtros com a imagem, ou seja, esses são deslizados por toda a sua dimensão, gerando então uma matriz de neurônios de apenas duas dimensões. No entanto, cada camada de convolução pode possuir diversos filtros, que são ao final agregados na dimensão de profundidade. Logo, se 32 filtros são utilizados em uma determinada camada, o resultado desta será uma matriz com dimensão de profundidade igual 32. Podemos visualizar esse princípio melhor na Figura 4 onde cada cor representa um filtro e a matriz bidimensional de saída, gerada pela convolução do filtro na imagem completa, ao final todas as matrizes de saída são agrupadas na dimensão de profundidade.

Figura 4 – Filtros na camada de convolução



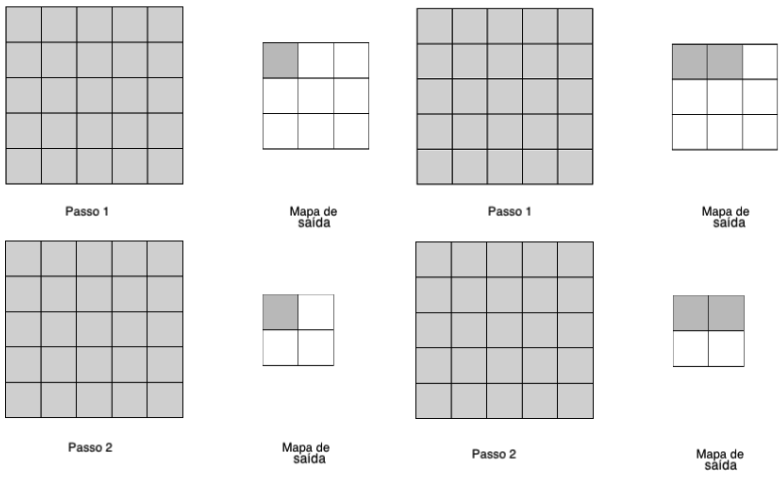
Fonte: (DERTAT, 2017)

A matriz de saída de um filtro de convolução também é chamada de mapa de ativação, isto é, ela é uma representação de quanto certa característica está presente em cada respectiva parte da matriz de entrada. Essa característica é treinada e definida pela própria rede e pode representar uma borda, uma linha curva ou uma marca de alguma cor. Apesar das características que ativam cada filtro serem determinadas pela rede, existem alguns parâmetros que definem como cada filtro se comporta, esses são denominados, hiperparâmetros.

Os hiperparâmetros da camada de convolução são: campo de ativação, profundidade, passo e *padding*. O primeiro diz respeito às dimensões dos filtros que compõem essa camada. O tamanho do campo influencia diretamente no número de pesos atrelado a um neurônio, por exemplo, em uma figura de entrada de 30x30x3 (imagem de 30x30 *pixels* e 3 cores) e um filtro de campo de ativação 3x3 *pixels*, significa que cada neurônio do mapa de ativação estará ligado a 3x3x3 $= 27$ valores da imagem de entrada.

Como mencionado anteriormente o hiperparâmetro de profundidade é definido pelo número de filtros que são utilizados em determinada camada de convolução. Já o passo determina a quantidade de espaços (*pixels*) separando uma convolução e a próxima, e isso está firmemente ligado às dimensões da imagem de saída. Conforme mostra a Figura 5 é possível notar a diferença entre os mapas de saída quando varia-se o passo da camada.

Figura 5 – Passo dos filtros na camada de convolução

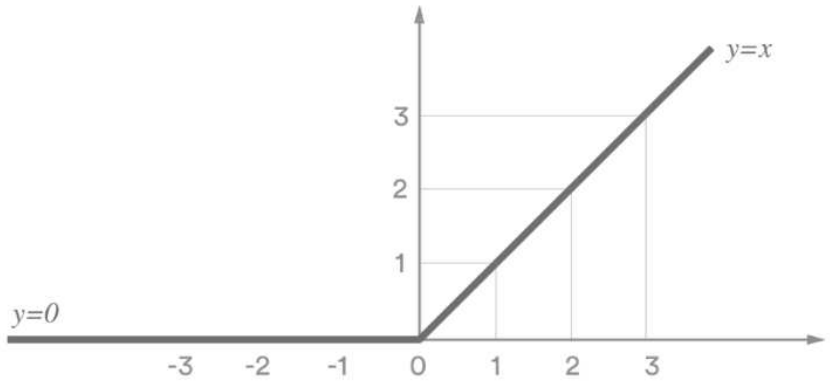



Fonte: Adaptado de: (DERTAT, 2017)

Por último o hiperparâmetro *padding*, também chamado de preenchimento, diz respeito a como a figura é aumentada para que o mapa de ativação tenha tamanho adequado. Uma vez que uma rede neural profunda pode ter diversas camadas é fácil observar que o tamanho dos mapas de ativação se reduz rapidamente ao ponto de que seria inviável possuir muitas camadas. Uma maneira de mitigar este problema é adicionando-se uma "borda" à matriz de entrada, que não influencie significativamente no resultado. Faz-se isso acrescentando zeros ao redor da matriz de entrada, esse método é conhecido como *zero-padding*. O valor do hiperparâmetro então diz qual será o tamanho desta "borda".

Outra etapa muito importante na camada de convolução é a chamada não-linearidade, isto é, uma função não linear que é processada ao final de cada convolução. Assim o valor do neurônio no mapa de ativação não corresponde ao valor da convolução apenas, mas o valor após o cálculo da função de ativação. A função mais comumente utilizada é a ReLU (*Rectfied Linear Unit*), essa função retorna zero para valores negativos e é linear para valores positivos, como pode-se observar na Figura 6.

Figura 6 – Gráfico função ReLU

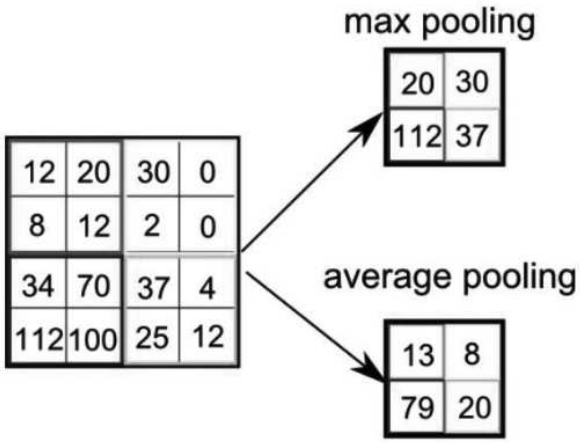



Fonte: (LIU, 2017)

#### 2.2.1.2 CAMADA *POOLING*

A camada *Pooling* tem o objetivo de reduzir dimensionalmente a imagem de saída, isto é utilizado tanto para reduzir o esforço computacional, quanto para marcar as características mais acentuadas da imagem de entrada. É comum que essa camada seja implementada após algumas convoluções dentro de uma rede profunda. Outra vantagem dessa camada é evitar o *overfitting*. Existem, basicamente, dois tipos de camada *Pooling*: *Max Pooling* e *Average Pooling*. O primeiro preserva apenas o maior valor da área de atuação do filtro, buscando extrair apenas a característica dominante deste quadrante. Uma das vantagens deste tipo de *Pooling*, que é o mais comumente utilizado, é o fato de reduzir o número de características secundárias das imagens, já que preserva apenas a característica mais marcante. O segundo tipo faz uma média dos valores da região de atuação do filtro. Aqui os ruídos não são suprimidos totalmente, por isso esse tipo de operação não é tão utilizada. A Figura 7 apresenta os dois tipos de *Pooling*.

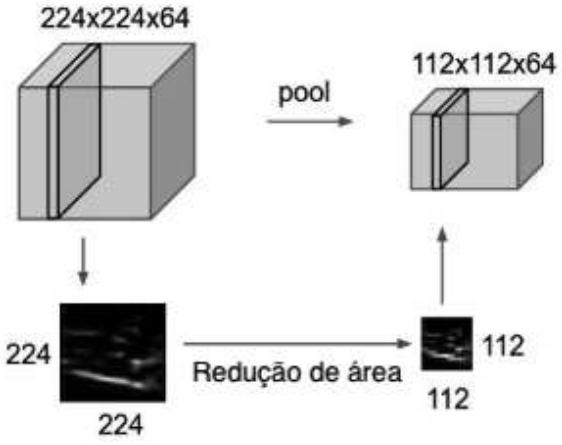
Figura 7 – Tipos de operação *Pooling*




Fonte: (SAHA, 2018)

Essa etapa também possui hiperparâmetros, são eles: extensão espacial e passo. Como na camada convolucional, eles determinam as dimensões da matriz do filtro e o passo que esse desliza na imagem de entrada, respectivamente. No entanto, ao contrário dos filtros de convolução que possuem profundidade igual à da imagem de entrada, aqui o filtro é aplicado separadamente à cada fatia de profundidade da imagem de entrada, como ilustrado na Figura 8. Nesta figura também pode-se observar como as características dominantes são mantidas mesmo com a redução de dimensão, quando utiliza-se a operação *Max Pooling*.

Figura 8 – Operação *Max Pooling* na dimensão de profundidade



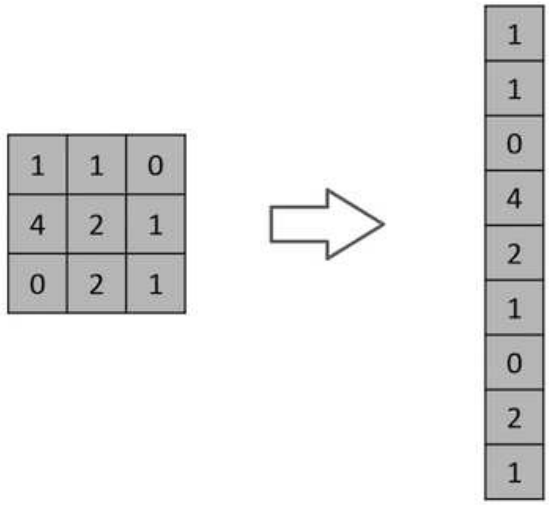

Fonte: Adaptado de: (LI JUSTIN JOHNSON, 2018)

### 2.2.1.3 CAMADA TOTALMENTE CONECTADA

As camadas totalmente conectadas ou *Fully Connected* (FC), geralmente vem ao final das redes neurais convolucionais e são utilizadas em número reduzido, por serem camadas que exigem bastante esforço computacional. Essas camadas relembram as utilizadas nas redes

neurais tradicionais, onde todos os neurônios são conectados a todos os da próxima camada. Como esse processo só pode ser feito com vetores é necessário que a matriz de saída da última camada de convolução ou *pooling*, seja transformada em um vetor. Essa etapa é também conhecida como *flatten* ou achatamento da matriz e pode ser observada na Figura 9. As camadas FC são também utilizadas por serem capazes de identificar não linearidades nas matrizes (SAHA, 2018).

Figura 9 – Transição da camada de convolução para totalmente conectada




Fonte: (SAHA, 2018)

A camada totalmente conectada é utilizada também por classificadores, de forma a estabelecer correlações entre a imagem de entrada e as classes avaliadas. Neste trabalho utiliza-se um classificador SVM. Após a matriz de saída ser achatada, o vetor resultante será avaliado, através dos parâmetros do classificador e devolverá um vetor com valores relacionando a imagem de entrada à cada classe avaliada. Porém, esses valores são difíceis de interpretar, por isso esse vetor é normalizado para que os valores correspondam as probabilidades de cada classe (LI JUSTIN JOHNSON, 2018).

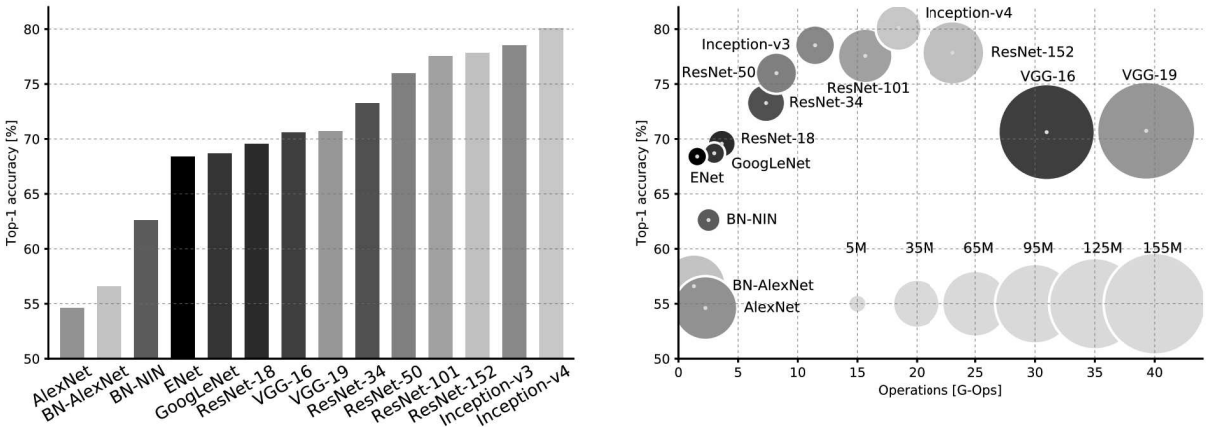
### 2.2.2 INCEPTION v3

No presente trabalho utiliza-se a terceira versão da rede neural criada pela *Google* em 2014, denominada em sua primeira versão *GoogLeNet* (SZEGEDY et al., 2015), que foi reformulada em 2015, conforme Szegedy et al. (2016) e renomeada para *Inception*. Essa rede ganhou o prêmio em 2014 da ILSVRC (*ImageNet Large Scale Visual Recognition Challenge*), que é uma das maiores competições de CNN do mundo, atingindo apenas 6,66 % em *top-5* erros de classificação (IMAGENET, 2014). A arquitetura dessa rede ficou bastante conhecida, em sua primeira versão, por ter sido a primeira a introduzir os chamados módulos *Inception*, que

são operações realizadas "em paralelo" e depois concatenadas, assim, sendo possível aumentar a eficiência computacional e diminuir os *bottlenecks* da rede.

Outros fatores relevantes para a escolha desta rede, especificamente, se basearam em tentar encontrar um equilíbrio entre acurácia, demanda computacional e número de hiperparâmetros. Como pode ser observado na Figura 10 a *Inception v3* está entre as redes com maior acurácia, porém não possui muitos hiperparâmetros e mantém o número de operações reduzidas.

Figura 10 – Comparação de redes neurais convolucionais




Fonte: Canziani e Culurciello (2017)

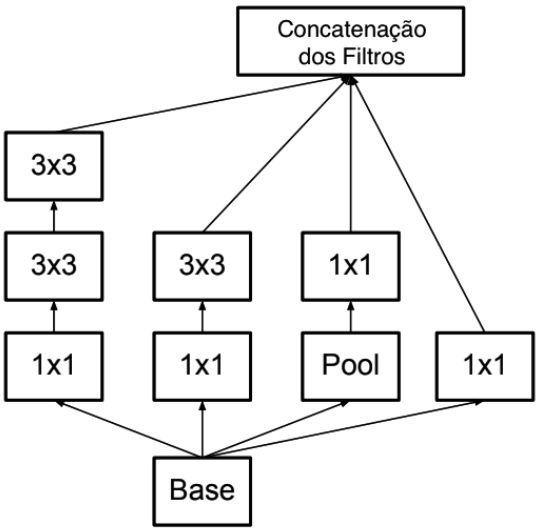
No que diz repeito à arquitetura da rede neural *Inception v3*, pode-se observar na Tabela 1 as diversas camadas que a compõem, que são descritas no restante desta seção.

Tabela 1 – Arquitetura Inception v3.

| Nome da camada | Estrutura |
|---|---|
| Imagem de entrada | RGB, 299 x 299 pixels |
| Conv1 | Camada de convolução: 32 filtros de tamanho 3x3 com passo 2 |
| ReLU | ReLU |
| Conv2 | Camada de convolução: 32 filtros de tamanho 3x3 com passo 1 |
| ReLU | ReLU |
| Conv3 | Camada de convolução: 64 filtros de tamanho 3x3 com passo 1 |
| ReLU | ReLU |
| Pool1 | Camada Max Pooling: tamanho 3x3 com passo 2 |
| Conv3 | Camada de convolução: 80 filtros de tamanho 1x1 com passo 1 |
| ReLU | ReLU |
| Conv4 | Camada de convolução: 192 filtros de tamanho 3x3 com passo 1 |
| ReLU | ReLU |
| Pool2 | Camada Max Pooling: tamanho 3x3 com passo 2 |
| *Inception* módulo A | ver *Inception* módulo A |
| *Inception* módulo A | ver *Inception* módulo A |
| *Inception* módulo A | ver *Inception* módulo A |
| redução de Imagem | ver redução de Imagem |
| *Inception* módulo B | ver *Inception* módulo B |
| *Inception* módulo B | ver *Inception* módulo B |
| *Inception* módulo B | ver *Inception* módulo B |
| *Inception* módulo B | ver *Inception* módulo B |
| redução de Imagem | ver redução de Imagem |
| *Inception* módulo C | ver *Inception* módulo C |
| *Inception* módulo C | ver *Inception* módulo C |
| Pool3 | Camada Avg Pooling |
| Pool4 | Camada Drop-out |
| FC | Camada Fully connected com 2048 neurônios |
| Prob | Softmax |

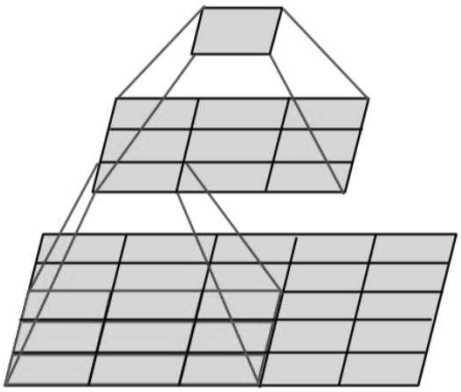
Na Figura 11 pode-se observar a técnica utilizada para aumentar a eficiência computacional das camadas de convolução. No caso do módulo aqui denominado de *Inception* módulo A, faz-se uso de operações em escala menor, em paralelo, para alcançar o mesmo tamanho de profundidade e área de dependência. No caminho mais à esquerda, representado no módulo A, a mudança, da versão original da rede neural (*GoogLeNet*), ficou na convolução que antes era feita com uma matriz 5x5 e agora são utilizadas duas camadas 3x3, assim reduzindo o número de parâmetros necessários. Conforme ilustrado na Figura 12 a troca de um filtro convolucional 5x5 por dois filtros 3x3 em sequência não altera a área de dependência do filtro (SZEGEDY et al., 2016).

Figura 11 – *Inception* módulo A




Fonte: Adaptado de: (SZEGEDY et al., 2016)

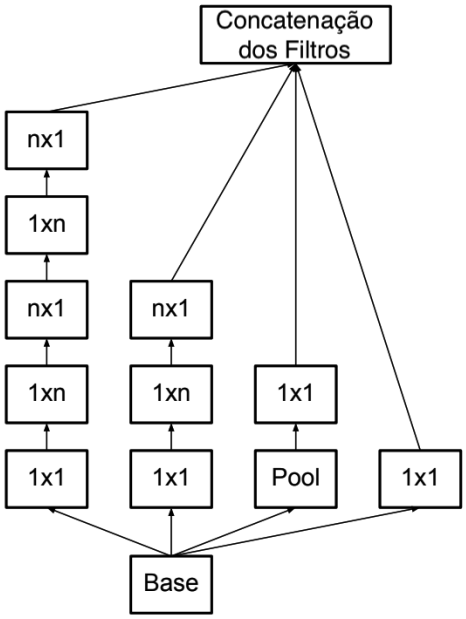
Figura 12 – Redução da área do filtro



Fonte: Adaptado de: (SZEGEDY et al., 2016)

O *Inception* módulo B é mostrado de forma genérica na Figura 13. Na *Inception v3* utiliza-se $n = 7$, assim a ideia utilizada no módulo A repete-se também neste módulo, porém, agora transforma-se operações convolucionais complexas e de grande exigência computacional, neste caso, convoluções de campo 7x7, em operações assimétricas de formato 1x$n$ e $n$x1 (SZEGEDY et al., 2016).

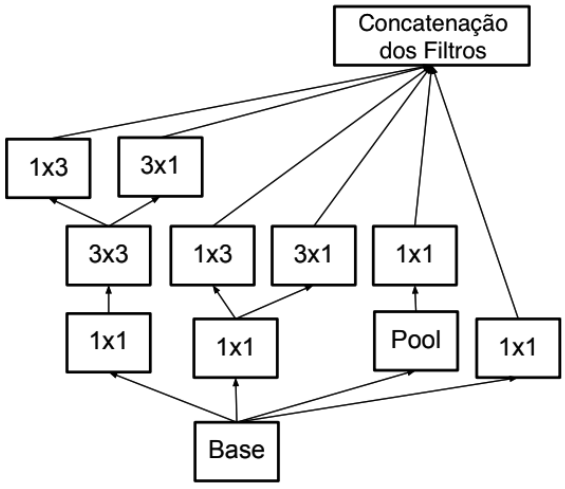
Figura 13 – *Inception* módulo B




Fonte: Adaptado de: (SZEGEDY et al., 2016)

De forma similar o módulo C utiliza os mesmos princípios mas agora na tentativa de aumentar o número de filtros sem diminuir o tamanho da imagem. Já que ao final da rede as imagens já estão com dimensões bem reduzidas, seria inviável apenas ganhar profundidade, isso acarretaria na perda de informações. Assim o módulo apresentado na Figura 14 busca ampliar a área dos filtros sem reduzir as dimensões.

Figura 14 – *Inception* módulo C

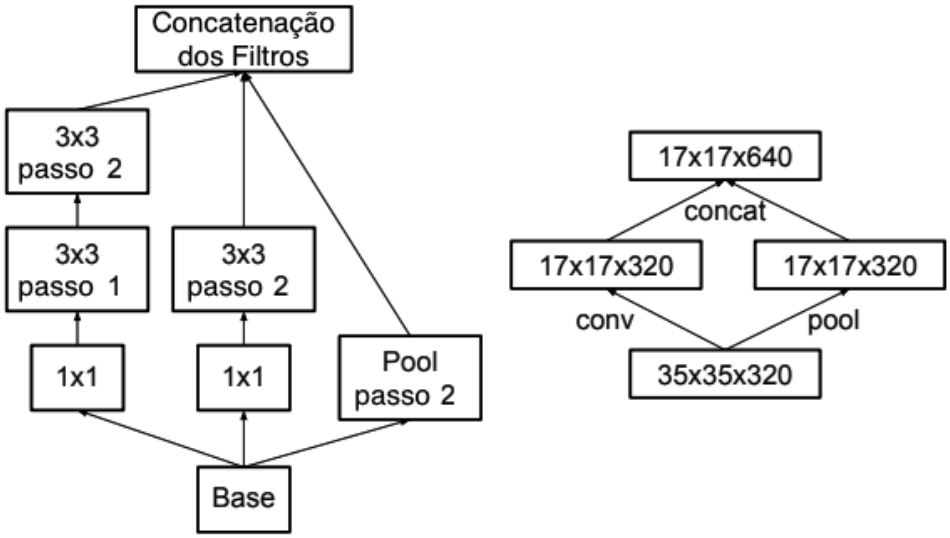



Fonte: Adaptado de: (SZEGEDY et al., 2016)

Em alguns momentos da rede é necessário fazer a redução das imagens que estão sendo trabalhadas, para ganhar-se em efetividade. No entanto, os métodos utilizados anteriormente

apresentavam algumas falhas como: ou a perda de representatividade das imagens para ganho computacional ou uma sobrecarga computacional para evitar a perda de representatividade. O artigo de Szegedy et al. (2016) propõe uma nova solução para este problema (ver Figura 15). Neste caso fazendo-se as operações de *Pooling* e convolução, paralelamente e com passo 2, é possível atingir-se a mesma redução de imagem sem comprometer a representatividade e reduzindo o custo computacional. O efeito dessa técnica pode ser observado no esquemático na parte direita da Figura 15.

Figura 15 – *Inception* redução da imagem




Fonte: Adaptado de: (SZEGEDY et al., 2016)

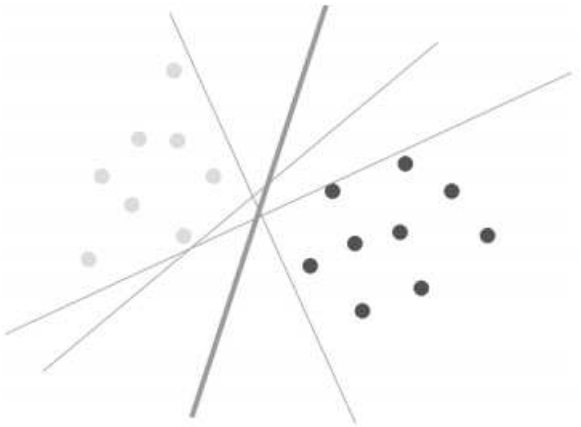
## 2.3 CLASSIFICADOR SVM

*Support Vector Machine* ou SVM são funções muito conhecidas no meio de aprendizagem de máquinas por atingirem acurácia significativa com pouco esforço computacional. Os algoritimos SVM podem ser utilizados para resolver problemas de classificação ou de regressão, como neste trabalho utilizou-se SVM para classificação final dos objetos, tratou-se apenas desta aplicação. É comum encontrar a utilização da sigla SVM, em artigos e livros, para referir-se a classificadores, porém isso pode gerar confusão nos conceitos. Assim, neste trabalho Classificadores com Vetores de Suporte são abreviados por SVC (*Support Vector Classification*) (GUNN, 1998).

A base teórica do algoritimo de classificação, neste caso, é encontrar um hiperplano que separe os objetos em suas respectivas classes (treinamento supervisionado). O problema mais simples possível seria de uma estrutura com apenas duas categorias que pudessem ser separadas por um hiperplano linear (GUNN, 1998). No entanto, mesmo neste caso é possível encontrar diversos hiperplanos que atendem os requisitos. É necessário então encontrar o hiperplano ótimo, que para o SVC é definido como o hiperplano que possui a maior distância

entre os pontos mais próximos dele (GANDHI, 2018). Como ilustra a Figura 16, existe apenas um hiperplano (em verde) que é ótimo.

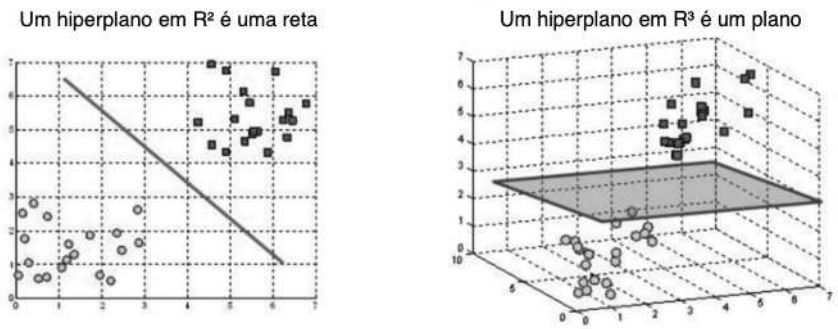
Figura 16 – Hiperplano de separação otimizado



Fonte: (GUNN, 1998)

É importante notar que no exemplo anterior são consideradas apenas duas dimensões. Para os SVC cada dimensão representa um característica a ser avaliada. Assim, para o caso em que apenas duas características são avaliadas, o hiperplano é uma reta, para o caso de três características seria um plano, como mostra a Figura 17.

Figura 17 – Hiperplano em 2D e 3D

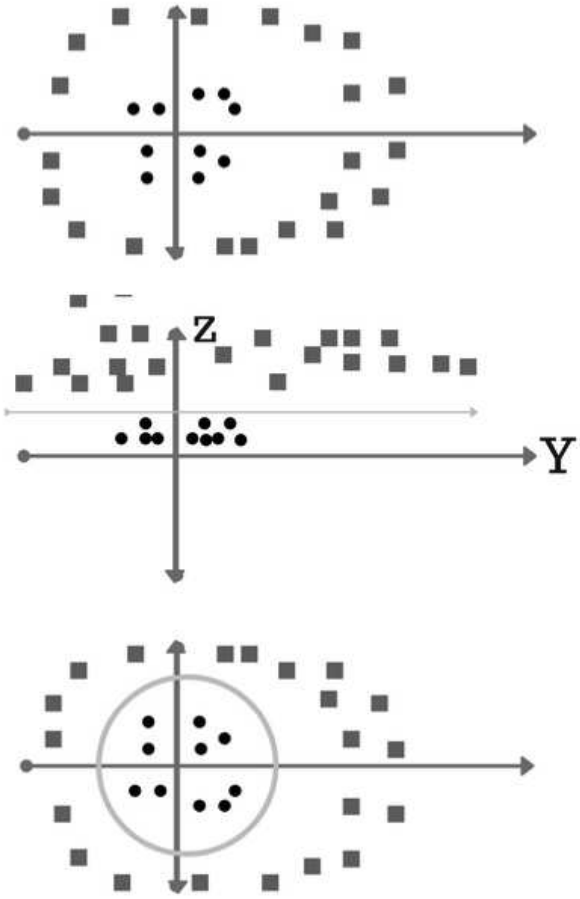



Fonte: (GANDHI, 2018)

Entretanto é fácil imaginar que em um caso real, onde as características avaliadas estão na casa das dezenas ou centenas e existem milhares de pontos de treino, a separação não seja linearmente possível. Para solucionar o problema de não linearidade, é necessário primeiramente transformar os pontos até que sejam separáveis. Como mostra a Figura 18, as características não são linearmente separáveis no espaço original, no entanto, é possível transformá-los utilizando um novo espaço dimensional, através de uma equação de transformação $w = x^2 + y^2$, por exemplo, (PATEL, 2017). Obtendo pontos linearmente separáveis, no estágio intermediário e

podendo então encontrar o hiperplano linear que otimiza a classificação. Ao final retorna-se ao espaço original mantendo-se uma separação válida entre as classes de pontos.

Figura 18 – Transformação de pontos de treino

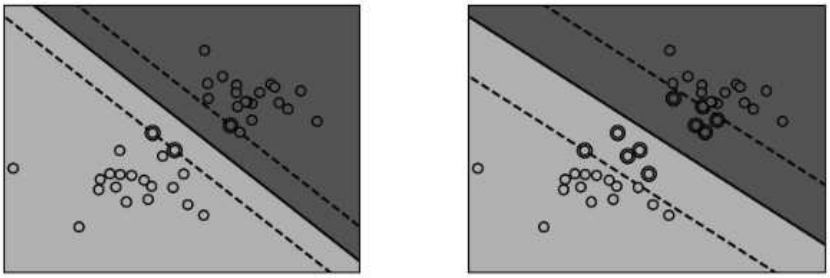



Fonte: Adaptado de: (PATEL, 2017)

Essas técnicas de transformação de pontos é definida pelo chamado *kernel* do SVC. Os *kernels* podem ter formato linear, que usa uma função baseada em produto escalar do tipo $f(x) = B(0) + \sum(a_i * x, x_i)$ onde $B(0)$ e $a_i$ são parâmetros aprendidos pelo algoritimo (PATEL, 2017). Existem ainda outros *kernels* que podem solucionar problemas de não linearidade i.e., Polinomial, Gaussiano RBF, Sigmoidal, Multiquadrática Inversa, entre outras (Müller, 2001).

Outras características dos SVC's envolvem parâmetros relacionados com a margem, ou distância mínima entre um dado de treino e o hiperplano, e o peso que será atribuído a cada ponto no cálculo de erro. O primeiro é conhecido como parâmetro *C*, que foi introduzido por Cortez e Vapnik (1995) para generalizar a função de otimização do hiperplano. Como ilustra a Figura 19, quanto menor o seu valor, maior é a tolerância por pontos classificados incorretamente, assim a margem aumenta, mesmo que isso implique em classificação errada de alguns pontos (quadro da direta) (PATEL, 2017). Os pontos aos quais a margem é apoiada, são os denominados vetores de suporte, de onde o classificador recebeu seu nome.

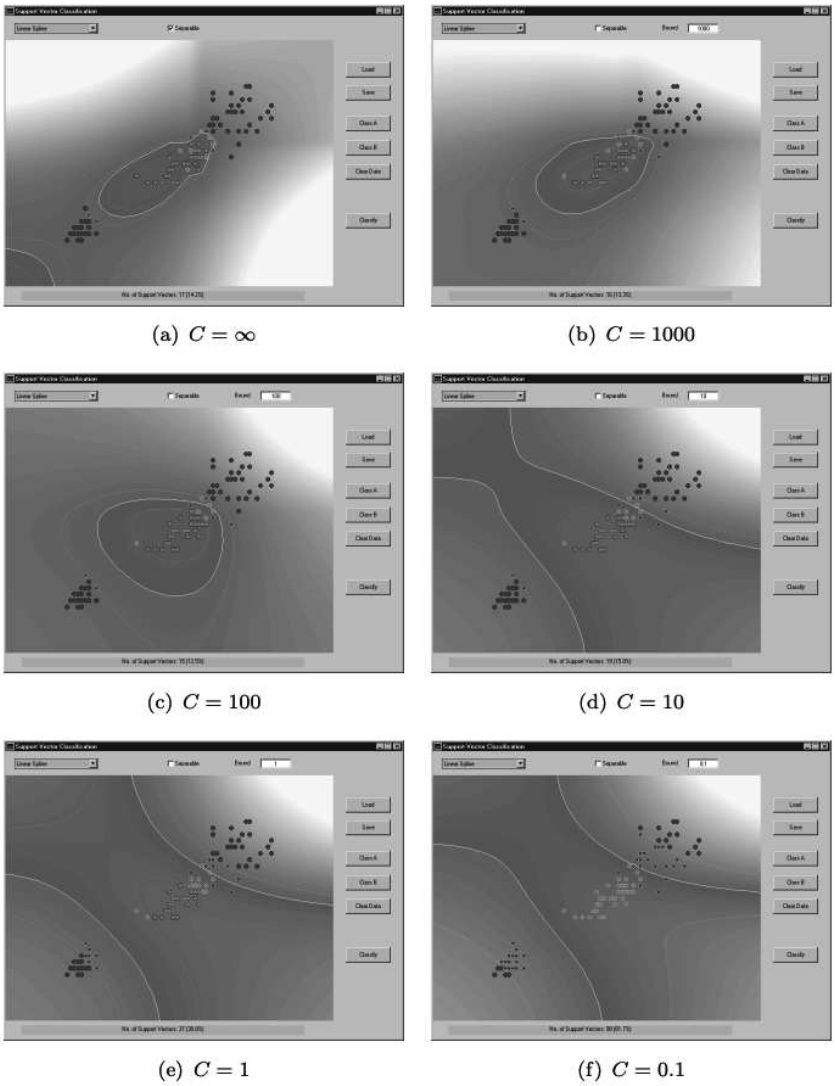
Figura 19 – Comparação parâmetro C



Fonte: (SCIKITLEARN, 2018c)

A Figura 20 apresenta um exemplo de como os valores de $C$ influenciam na formação dos planos de classes.

Figura 20 – Exemplo variação parâmetro $C$



(a) $C = \infty$ (b) $C = 1000$

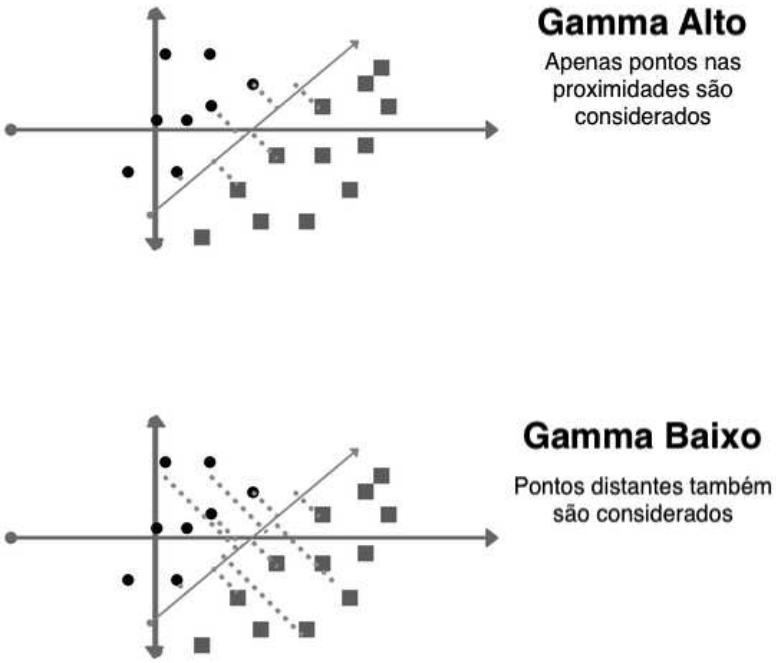
(c) $C = 100$ (d) $C = 10$

(e) $C = 1$ (f) $C = 0.1$

Fonte: (GUNN, 1998)

Finalmente o parâmetro *gamma* indica qual a distância, em relação ao hiperplano, de influência dos pontos de cada categoria, isto é, quanto menor o valor de *gamma* maior a distância de influência, como mostra a Figura 21. Quanto maior a distância de influência, maior é a variação de dados aceita em cada classe, isso pode trazer benefícios uma vez que aumenta a robustez do sistema, por diminuir o viés dos valores utilizados nos cálculos do hiperplano. No entanto, grandes distâncias de influência podem causar má classificação das imagens, uma vez que a variação aceita para determinada classe é tão abrangente que agrega características de outra classe. Para maiores detalhes nos cálculos é possível visitar Vapnik (1995) ou Gunn (1998).

Figura 21 – Parâmetro *gamma*



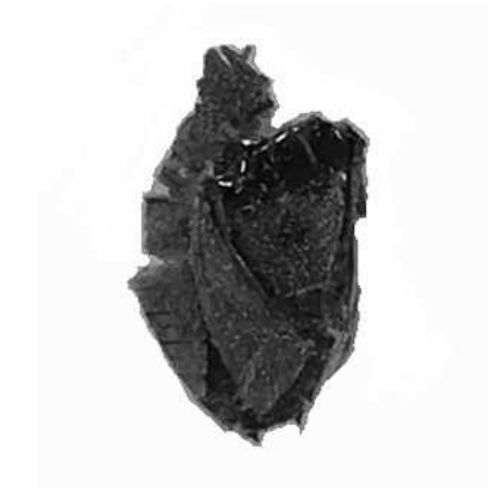

Fonte: Adaptado de: (PATEL, 2017)

# 3 METODOLOGIA

Utilizando-se como base a arquitetura *Inception v3*, pré-treinada no *dataset* ImageNet (SZEGEDY et al., 2016), substituiu-se o sistema de classificação da rede original, ver Tabela 1, acrescentando-se um SVC e fez-se o treinamento da CNN com o banco de dados entomológico. Os insetos avaliados são comumente encontrados em cadáveres ou em cargas de entorpecentes apreendidas. A coleta e identificação destes auxilia na identificação de algumas características do crime, como a origem da carga, local de execução da vítima ou tempo de *post mortem*.

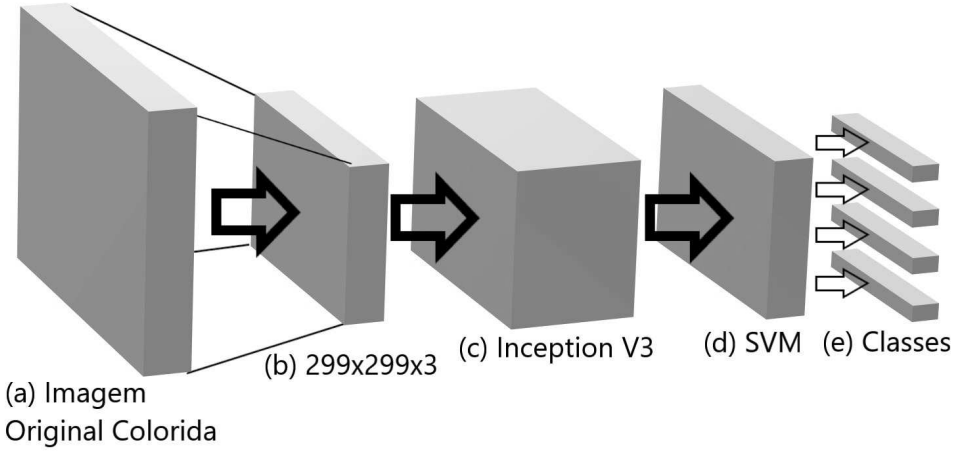
A arquitetura de sistema implementada tem como objetivo classificar não apenas insetos inteiros mas também parte deles, como exemplifica a Figura 22. Isso é de extrema importância, pois em uma situação real onde os insetos são coletados em cenas de crime ou cargas de entorpecentes apreendidas, é bastante provável que o espécime encontrado esteja parcialmente destruído. Como parte do processo de utilização da rede neural as imagens dos insetos devem ser tiradas em um fundo branco para que o sistema de processamento de imagens preserve o maior número de detalhes possível.

Figura 22 – Amostra da Família *Pentatomidae* Vista Superior.



Na Figura 23, pode-se observar a sequência de operações que constituem a rede. Primeiramente a imagem é pré-processada, ou seja, redimensionada para 299x299x3. Depois, caso seja requisitado pelo usuário, é feito o processo de *Data Augmentation*, criando-se novas variantes da imagem que são utilizadas para o treinamento da rede. Após a criação das várias imagens, essas são submetidas à rede *Inception v3* que agora, porém, possui o classificador SVM nas últimas camadas i.e, as camadas de classificação. Cada procedimento do sistema será explanado ao longo deste capítulo.

Figura 23 – Arquitetura do sistema.

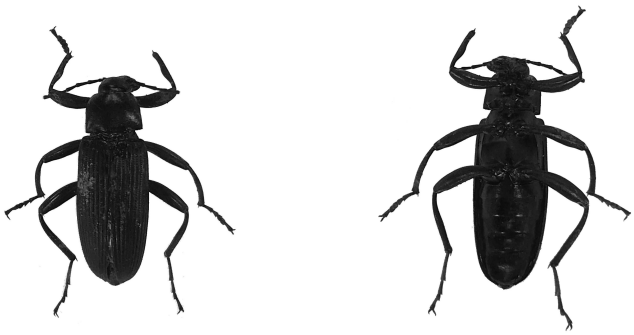



O sistema abrange cinco famílias entomológicas: *Buprestidae*, *Calliphoridae*, *Formicidae*, *Muscidae* e *Pentatomidae*. Essas são subdivididas em vistas superior e inferior, conforme exemplificado nas figuras 24, 25, 26, 27 e 28, totalizando dez classes, sendo possível a adição futura de novas vistas e famílias, conforme a aquisição de novas imagens pelo usuário. Abaixo segue a Tabela 2 com o número atual de imagens por classe.

Tabela 2 – Número de imagens por família entomológica.

| | Superior | Inferior |
|---|---|---|
| Buprestidae | 21 | 22 |
| Calliphoridae | 13 | 13 |
| Formicidae | 21 | 17 |
| Muscidae | 6 | 4 |
| Pentatomidae | 46 | 45 |

Figura 24 – Inseto da Família Buprestidae.



(a) Vista Superior (b) Vista Inferior

Figura 25 – Inseto da Família Formicidae.

(a) Vista Superior (b) Vista Inferior

Figura 26 – Inseto da Família Muscidae.

(a) Vista Superior (b) Vista Inferior

Figura 27 – Inseto da Família Calliphoridae.

(a) Vista Superior (b) Vista Inferior

Figura 28 – Inseto da Família Pentatomidae.

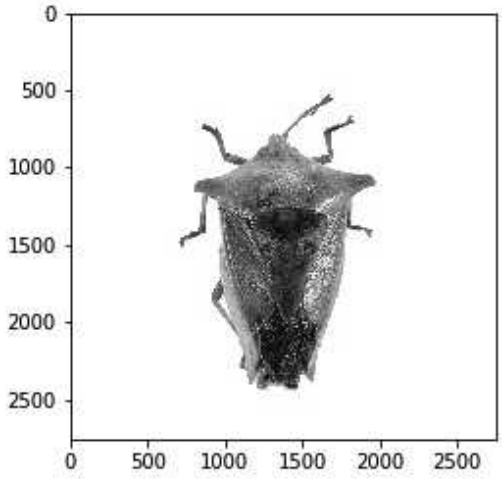
(a) Vista Superior (b) Vista Inferior

## 3.1 PROCESSAMENTO DE IMAGENS

Para esta etapa de processamento de imagens são utilizadas funções do módulo *OpenCV* em linguagem Python, faz-se o redimensionamento das mesmas para atender as especificações de utilização otimizada da CNN *Inception v3*. Já as funções para selecionar o objeto são implementadas para facilitar a inserção futura de novas imagens, tanto para retreinamento como para identificação de objetos.

As imagens de entrada utilizadas são coloridas, de dimensões (X,Y) variadas e com fundo branco, como pode-se observar na Figura 29, sendo redimensionadas para 299x299x3, com o uso de interpolação linear, durante esta etapa de processamento de imagens.

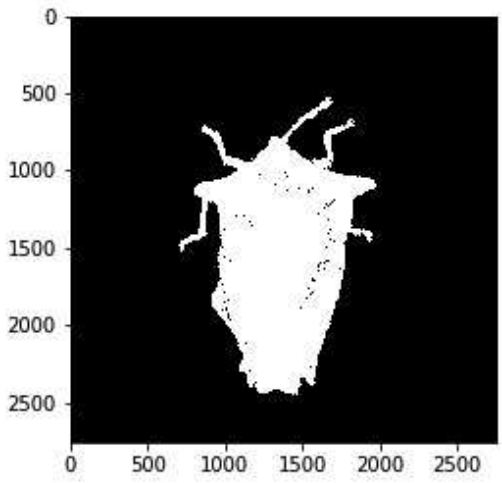
Figura 29 – Imagem de entrada, com fundo branco e dimensões 2756x2756x3.




Inicialmente o programa verifica se os pixels da imagem RGB (*Red*, *Green*, *Blue*) estão entre os limites fornecidos ([R, G, B]), com valores inferiores (*Lower Bound*) e superiores (*Upper Bound*) fornecidos pelo usuário. Caso os valores do pixel estejam entre os limites é

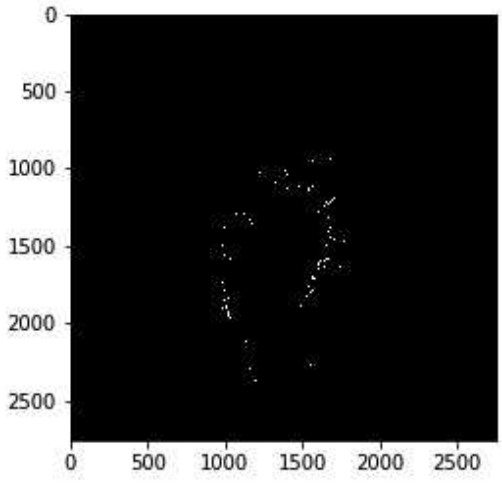
retornado 255 para o mesmo, do contrario, é retornado um valor de 0. A Figura 30 mostra o resultado desta primeira etapa sendo aplicada a Figura 29.

Figura 30 – Máscara de pixels (*Grayscale*) para *Lower Bound* = [0,0,0] e *Upper Bound* = [250,250,250].




Em seguida a imagem da Figura 30 é binarizada, onde pixels com valores inferiores e superiores a 254 recebem, respectivamente, *bits 1* e *bits 0*. Com a nova imagem, em níveis de cinza, é obtida uma máscara contendo os contornos encontrados na Figura 30, realizando então, uma operação *not* com a mesma. O resultado final desta operação *not* com a máscara de contornos pode ser observada abaixo na Figura 31.

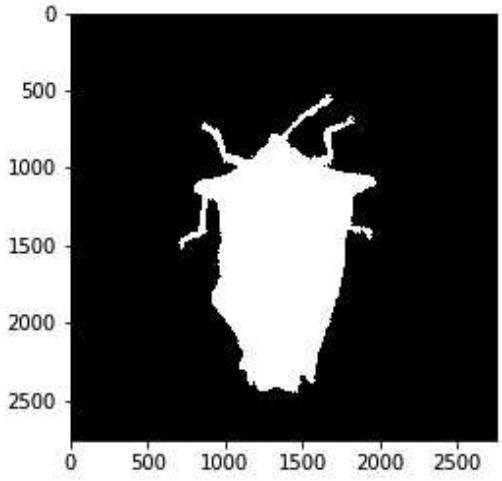
Figura 31 – Imagem após operação *not* com a máscara de bordas.




Com as máscaras da Figura 30 e da Figura 31 é possível realizar uma operação do tipo *or* entre as duas, uma vez que ambas tem formato de imagem binário. Esta operação é

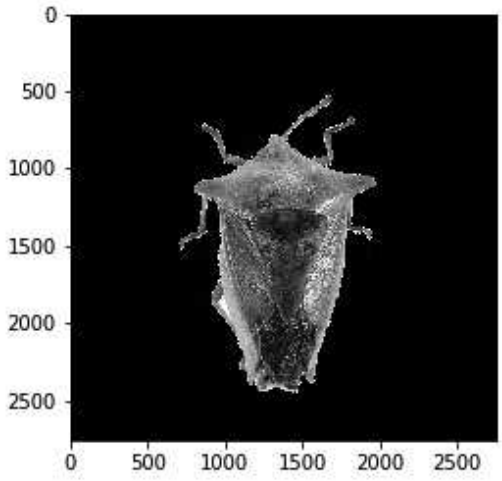
realizada de maneira a preencher os pixels pretos contidos dentro do objeto de interesse da máscara da Figura 30, obtendo-se o resultado da Figura 32.

Figura 32 – Máscara final contendo o objeto de interesse.




Após a etapa de identificação das bordas do objeto é realizado uma operação do tipo *and* entre a Figura 29 e a Figura 32, sendo que a segunda é utilizada como máscara para a identificação do objeto dentro do quadro. Com esta operação se obtém a Figura 33, na qual observa-se o objeto de interesse, i.e., inseto.

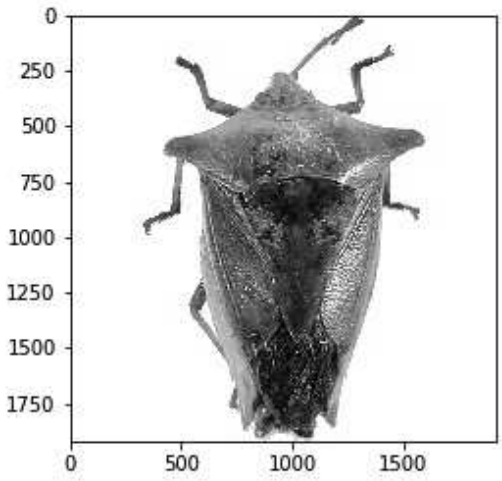
Figura 33 – Imagem com inseto separado do fundo branco.




Em seguida, o fundo da imagem acima é modificado de preto para branco e seu formato para níveis de escala de cinza, facilitando o recorte do objeto da Figura 33. É realizado então o *crop* do modelo, obtendo uma imagem de dimensões (X,Y) diferentes. Estas dimensões são então analisadas, buscando identificar qual das duas apresenta um valor superior, de forma a

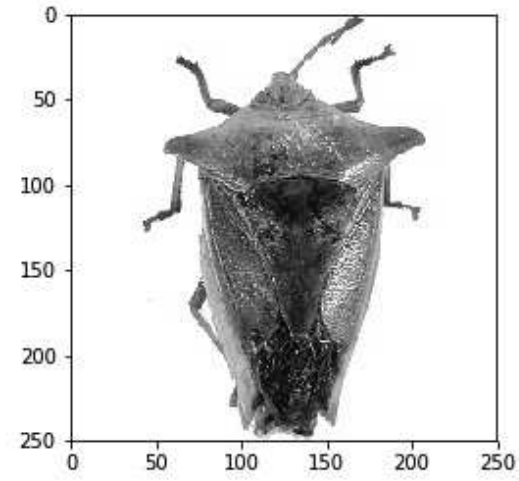
adicionar pixels brancos na coordenada de menor extensão, sendo estas distribuídas igualmente em cada direção. Com estas ações é obtida a imagem da Figura 34 de dimensões iguais nos dois eixos da mesma.

Figura 34 – Imagem com dimensões (X,Y) iguais.




Com a imagem da Figura 34, na qual as dimensões estão normalizadas e com o objeto de interesse centralizado, é realizado o redimensionamento do quadro. Este redimensionamento é feito com o uso de interpolação linear, obtendo uma imagem de 250x250x3 que pode ser vista na figura Figura 35.

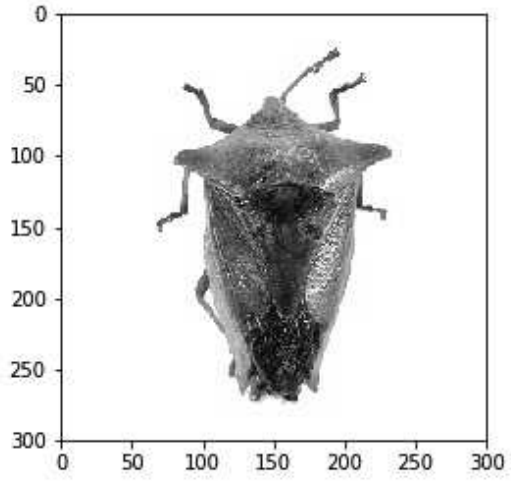
Figura 35 – Imagem reduzida de 250x250x3.




Por fim, são adicionados bordas no entorno da imagem da Figura 35. Como essa possui altura e largura de 250 pixels, é necessário a adição de pixels para atender a especificação da *Inception V3*. São então adicionados 50 pixels brancos nas dimensões (X,Y) do modelo,

distribuídos igualmente em cada direção, ou seja, 25 pixels acima, abaixo, à esquerda e à direita do mesmo, resultando na imagem final da Figura 36 de dimensão 299x299x3.

Figura 36 – Imagem final de 299x299x3.




Ao final deste pré-processamento das imagens, o sistema as salva em suas respectivas pastas de classes, as quais encontram-se no mesmo diretório dos *Scripts* em Python. Os nomes das pastas e subpastas tem o formato "FamíliaInseto VistaImagem" dentro das pastas "val" e "train".

## 3.2 DATA AUGMENTATION

Redes neurais profundas são capazes de aprender diversas relações entre dados de entrada e saída, entretanto, com um número limitado de dados de treinamento, muitas destas relações podem resultar em ruídos existentes apenas nestes dados, mas não nos de teste, levando ao sobre-ajuste (*overfitting*) (SRIVASTAVA et al., 2014). De maneira geral, o *overfitting* ocorre quando o modelo é muito bem modelado durante o treinamento, porém falha durante o teste, interpretando ruídos existentes apenas no *trainning data* como conceitos do modelo, impactando negativamente na generalização do mesmo (PAUL, 2018).

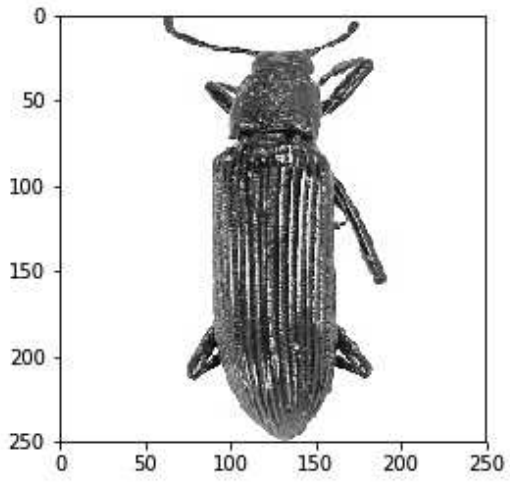
Dentro deste problema, *Data Augmentation* é um método que pode ser utilizado para reduzir o *overfitting* em modelos de treinamento, aumentando o número de dados de treinamento utilizando informações contidas no mesmo (WANG; PEREZ, 2017). Desta maneira, buscando extrair os melhores resultados possíveis da *CNN* utilizada, foi necessário fazer uso deste método, uma vez que o número de imagens disponíveis por classe é muito baixo (menos de 400 imagens por classe de objeto), aumentando a robustez da rede neural.

Para o sistema apresentado optou-se pela geração de novas imagens através da translação do objeto dentro do quadro de dimensões 299x299x3. Esta escolha ocorreu pelo fato de que a amostra deve ser fornecida ao sistema com iluminação adequada, fundo branco e

alinhada ao eixo horizontal, descartando a necessidade de rotacionar, espelhar, ou de alterar os níveis RGB da imagem. Em relação ao recorte de partes do inseto, isto foi inicialmente descartado pelo fato de não haver amostras deterioradas suficientes de teste para uma melhor análise deste método, sendo uma possível melhora para uma continuação futura do sistema.

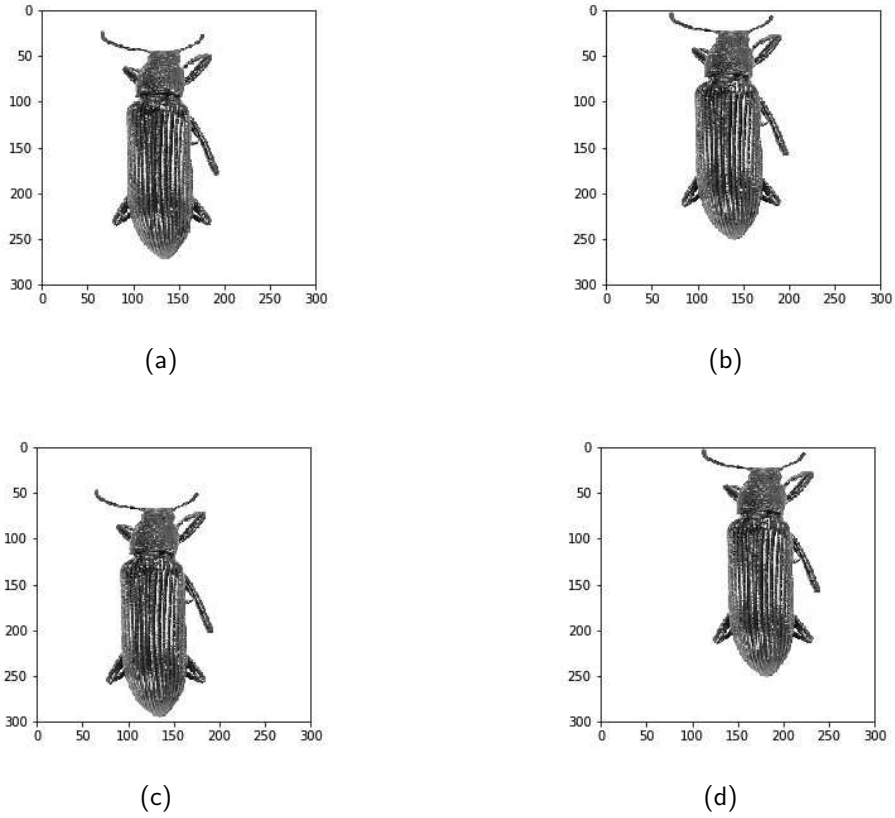
Para a geração de novas imagens, através da translação do objeto dentro do quadro, primeiramente é realizado o *Crop* do mesmo, centralizado e com dimensões 250x250x3, obtendo uma imagem semelhante à da Figura 37. Esta operação é realizada através da extração dos pixels localizados entre as posições 24 e 274 das linhas e colunas da imagem 299x299x3.

Figura 37 – Imagem de dimensões 250x250x3.




São então armazenados em um vetor todos os números inteiros entre o intervalo aberto de zero a 51. Com estes números são geradas todas as possibilidades, sem repetições, de pares de valores entre os dois limites, armazenando as mesmas em um *array* e realizando o *shuffle* do mesmo. Após, é realizado um número de iterações fornecidas pelo usuário, relativo a quantidade de novas imagens a serem geradas, para obter os valores das bordas laterais que são adicionadas à imagem da Figura 37. Para cada *loop* é obtido um par aleatório, caso o primeiro número (X) seja menor ou igual a 25, são adicionados X pixels à esquerda do quadro, do contrário o número 50 é subtraído de X e o número de pixels a serem adicionados será igual ao resultado da operação. Para os pixels à direita é realizado o processo inverso, sendo primeiro realizada subtração para o caso de ser menor que 25. Analogamente, o mesmo ocorre para o segundo valor do par de valores obtido, aplicando o mesmo método para as bordas superior e inferior. Ou seja, caso seja retornado pelo programa um par de números (25,25), serão adicionados 25 pixels acima e à esquerda da imagem, depois os mesmos são subtraídos de 50, obtendo 25 para as duas operações, adicionando as bordas à direita e inferior faltantes. Abaixo, na Figura 38 são apresentadas quatro imagens do mesmo objeto da Figura 37, porém com o objeto transladado de diferentes formas dentro do quadro de 299x299x3.

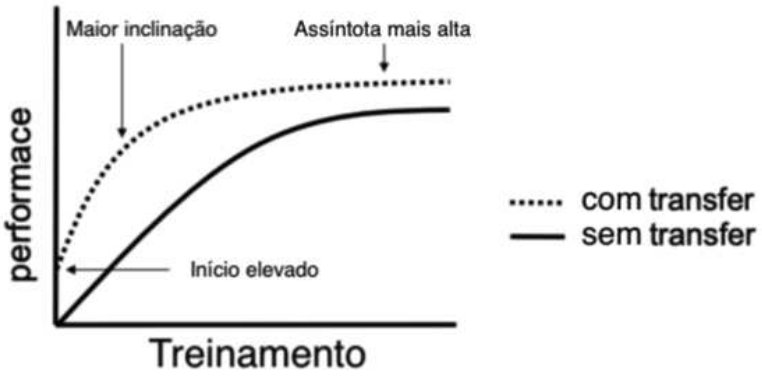
Figura 38 – Imagens de *Data Augmentation*




## 3.3 TRANSFER LEARNING

*Transfer Learning* é uma técnica de aprendizagem utilizada em diversos lugares, desde os seres humanos até rede neurais. Essa técnica se baseia na ideia de que um conhecimento aprendido para uma tarefa poderá ser aplicado a uma tarefa diferente com adaptações mínimas (TORREY; SHAVLIK, 2009). No ramo das CNN é raro ter dados e processamento suficientes para realizar o treinamento de uma rede neural totalmente nova, o que pode levar semanas, dependendo da profundidade da rede. Assim, popularizou-se a ideia de *Transfer Learning*. Utilizando-se redes, como *Inception*, *ResNet* ou VGG, que são treinadas em um *dataset* genérico (*ImageNet*) é possível atingir-se uma acurácia alta em pouco tempo, mesmo com um banco de dados reduzido. Conforme mostra a Figura 39 o método de *transfer learning* pode impulsionar a aprendizagem em três momentos: iniciando o treinamento da rede mais próximo da assíntota de acurácia, aumentando a velocidade de aprendizagem (maior inclinação) e finalmente alcançando uma assíntota (acurácia) mais alta (TORREY; SHAVLIK, 2009).

Figura 39 – Vantagens da implementação de *Transfer Learning*




Fonte: Adaptado de: (TORREY; SHAVLIK, 2009)

Existem três possíveis implementações de *transfer learning*, na primeira modifica-se apenas a camada de classificação mantendo-se todos os valores aprendidos estáticos, na segunda é feito um ajuste fino nos valores aprendidos pela rede, e o último caso é utilizar apenas as camadas iniciais da rede e construir uma nova a partir disso (LI JUSTIN JOHNSON, 2018). Neste trabalho um dos problemas a serem contornados é de o banco de dados disponível ser bastante reduzido, assim é ideal que utilize-se o primeiro conceito de implementação. Desta forma utilizou-se a rede neural convolucional *Inception v3* com parâmetros congelados, isto é, nenhum novo treinamento na rede foi feito. Isso foi definido exatamente por não haver exemplares suficientes que justificassem um ajuste fino ou até mesmo um retreinamento.

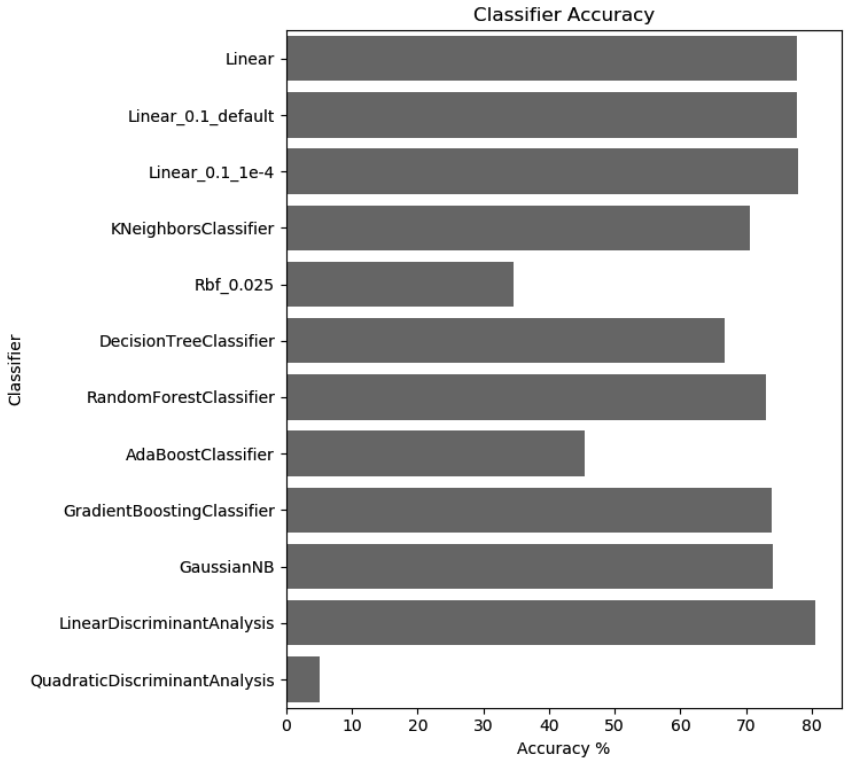
## 3.4 SISTEMA DE CLASSIFICAÇÃO

Seguindo a arquitetura *Inception v3*, decidiu-se remover as três últimas camadas da rede original. Assim a *Pool3* (ver Tabela 1) tornou-se a camada final e após esta camada foi acoplado um classificador SVM (*Support Vector Machine*). Classificadores SVM, ou SVC, são comumente utilizados quando o número de categorias é pequeno e também quando o banco de dados é reduzido. Conceitualmente este classificador é concebido de maneira a trabalhar com apenas duas classes, porém, graças a um suporte *multiclass* dentro do módulo *sklearn*, que é tratado de acordo com um esquema de um contra todos, é possível a utilização do mesmo com um maior número de categorias.

O suporte multiclasse é tratado de acordo com um esquema de um contra um.

Como já citado na Seção 2.3, uma das características mais importante do classificador é o *kernel* a ser utilizado para a separação de prováveis não linearidades no grupo de dados. Por meio da comparação entre diferentes *kernels*, ver Figura 40, escolheu-se implementar o método linear, que atingiu boa acurácia e é o de menor esforço computacional.

Figura 40 – Comparação entre classificadores




Outros hiperparâmetros importantes para a ajuste do classificador são os fatores C e *gamma*. O fator C diz respeito ao peso que será dado para a punição dos erros, ou seja, quanto maior o valor de C mais o erro durante o treinamento é punido. Assim o algorítimo tenta classificar de forma correta o máximo de objetos possível, o que implica diretamente na redução da margem do hiperplano. Aqui outra vez, a escolha foi determinada de forma empírica, sendo escolhido $C = 1$ como valor final.

Já o fator *gamma* diz respeito ao raio de influência dos vetores de suporte, isto é, quanto maior o valor de *gamma* menor é a área de influência dos pontos e consequentemente menor a variação aceita para objetos de uma mesma classe. Levando em conta o pequeno banco de dados, foi utilizado o valor padrão de *gamma*, que é $1/$(número de características) (SCIKITLEARN, 2018b). Esse valor permite que variações maiores, nas imagens de teste, sejam absorvidos.

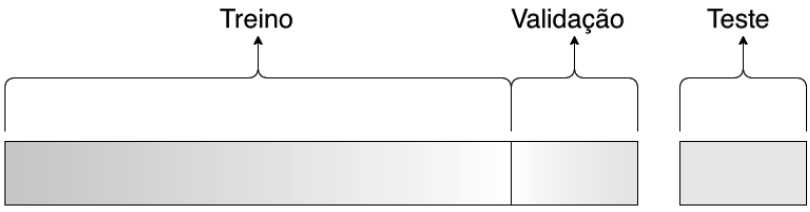
É importante ter em mente que como os valores e tipo de *kernel* foram definidos de forma empírica, conforme o banco de dados venha a aumentar ou a incluir novas categorias, talvez estas já não sejam as melhores opções. Então é indicado que de tempos em tempos sejam refeitos os testes de comparação entre métodos para que possa sempre incluir as variações ótimas. Outro ponto importante a ressaltar é a própria utilização da rede *Inception v3* de maneira estática. É possível que uma vez que o banco de dados tenha um tamanho considerável, seja adequado utilizar a técnica de *transfer learning* com ajuste fino à rede, assim aumentando ainda mais a sua acurácia.

## 3.5 TREINAMENTO

Após finalizar a implementação da arquitetura e ajustes de hiperparâmetros o sistema foi então submetido ao processo completo de aprendizagem i.e. treino, validação e teste. Aqui é importante distinguir cada parte do processo para evitar confusão de nomenclatura, outro ponto importante é saber que os conjuntos de dados ou *dataset* de cada fase são distintos, como ilustra a Figura 41. Treino diz respeito a parte supervisionada do processo, isto é, as imagens e suas classes foram definidas por humanos, e são conhecidas pela rede neural. Durante esse processo é que ocorre a aprendizagem da rede neural. Como a técnica utilizada neste trabalho é a de *transfer learning* estático, não ocorrerá aprendizagem, ou modificação, na rede *Inception v3* e sim apenas no classificador SVC que foi implementado ao final da rede.

A aferição dos resultados obtidos durante o treinamento é feita durante a fase de validação, quando o *dataset* de validação é utilizado para validar a acurácia da rede. Neste caso não há aprendizado, são mantidos os valores aprendidos durante o treino e tenta-se classificar os dados de validação. Dependendo dos resultados de validação, isto é, caso não seja atingido o nível de acurácia definido como objetivo, o processo de aprendizagem é repetido.

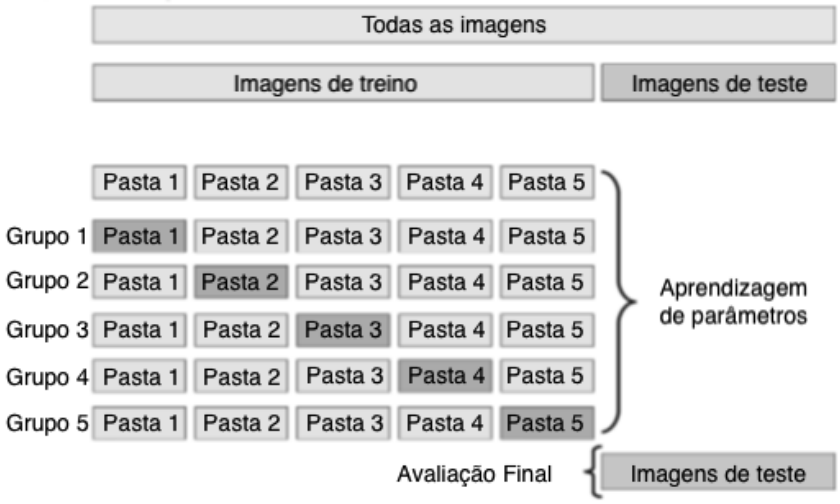
Figura 41 – Divisão dos conjuntos de dados




Fonte: Adaptado de: (SHAH, 2017)

Neste trabalho utilizou-se o método denominado *Cross Validation*, ou validação cruzada, para avaliar o processo de treinamento. A validação cruzada é feita através da divisão do *dataset* de treino em duas partes, a menor, neste caso 10 %, é reservada para a validação e o restante para treino. Esse processo é iterado tantas vezes quanto for necessário até atingir a acurácia desejada, sempre tomando um grupo de dados diferente para validação. A Figura 42 ilustra um exemplo deste método, a pasta em azul representa o grupo de validação daquela iteração. Ao final das iterações necessárias é feita então a fase de teste.

Figura 42 – Validação Cruzada




Fonte: Adaptado de: (SCIKITLEARN, 2018a)

A fase de teste é finalmente onde o sistema será avaliado com dados inéditos para a rede. É desta fase que são obtidos todos os indicadores de desempenho da rede (apresentados e discutidos no Capítulo 5). Apresenta-se uma imagem que deseja ser classificada e recebe ao final a indicação de qual categoria essa imagem pertence, assim como o valor da acurácia desta informação. Não existe reiteração ou aprendizado.

# 4 RESULTADOS

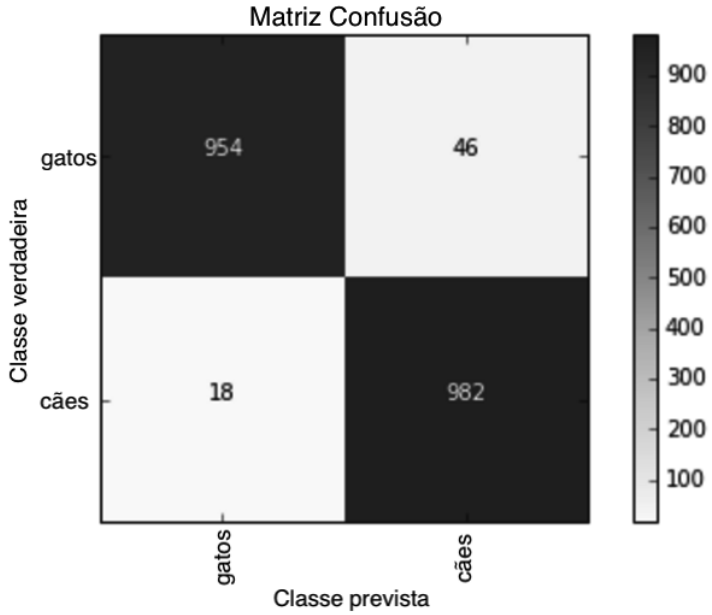
Uma maneira de avaliar um sistema de classificação baseado em algoritimos estatísticos é através de indicadores quantitativos. Neste capítulo são discutidos todos os medidores utilizados para avaliar a rede implementada, assim como as implicações dos resultados obtidos. Ao final são sugeridas possíveis melhorias e eventuais modificações necessárias no sistema.

## 4.1 INDICADORES DOS RESULTADOS

Existem diversos tipos de análises que podem ser feitas sobre os resultados obtidos de uma rede neural profunda. Foi decidido que os melhores indicadores para caracterizar o sistema implementado são: sensitividade, especificidade, acurácia e *log loss*. Para obter os valores necessários para os cálculos dos indicadores é preciso primeiramente criar-se uma matriz confusão.

A matriz confusão indica, em formato de matriz, como foram classificados os valores de entrada do sistema. Como mostra o exemplo da Figura 43 o eixo vertical mostra qual é o valor real do objeto, neste exemplo [cães e gatos], e o eixo horizontal indica os valores de saída do sistema. Os valores classificados corretamente estão identificados pelos quadrados em azul escuro.

Figura 43 – Matriz Confusão Exemplo




Fonte: Adaptado de (NABI, 2019)

Deste exemplo pode-se identificar quatro conceitos importantes para as análises realizadas em seguida, são eles: verdadeiro positivo, verdadeiro negativo, falso positivo e

falso negativo. No caso da Figura 43 para o valor cães tem-se 982 verdadeiros positivos, ou valores corretamente classificados, e 46 falsos positivos, valores que não são cães porém foram identificados como cães. De forma semelhante temos 18 falsos negativos, objetos que são cães mas foram identificados como gatos e 954 verdadeiros negativos, objetos que não são cães e foram classificados como gatos.

Sensitividade e especificidade são termos amplamente utilizados em classificação de padrões, para medir desempenho de classificadores. O primeiro termo, também conhecido como taxa de verdadeiros positivos, diz respeito à proporção de verdadeiros positivos entre todos os positivos, i.e., verdadeiros positivos mais falsos negativos. De maneira semelhante a especificidade indica a relação entre verdadeiros negativos e todos os negativos, i.e., verdadeiros negativos e falsos positivos. No exemplo em estudo:

$$\begin{aligned} \text{Sensitividade} &= \frac{\text{verdadeiros positivos}}{\text{verdadeiros positivos} + \text{falsos negativos}} \\ &= \frac{982}{982 + 46} \\ &= 0.95 \end{aligned} \tag{1}$$

$$\begin{aligned} \text{Especificidade} &= \frac{\text{verdadeiros negativos}}{\text{verdadeiros negativos} + \text{falsos positivos}} \\ &= \frac{954}{954 + 18} \\ &= 0.98 \end{aligned} \tag{2}$$

Acurácia é talvez o indicador mais comumente utilizado, porém nem sempre revela a verdadeira característica do sistema. A acurácia indica qual a porcentagem de imagens foram corretamente classificadas, isto é, a cada imagem testada o valor da classe prevista é comparado ao da classe real e, baseado em um sistema binário, certo ou errado, esse valor será contabilizado. A partir da Figura 43 pode-se calcular a acurácia do sistema exemplo:

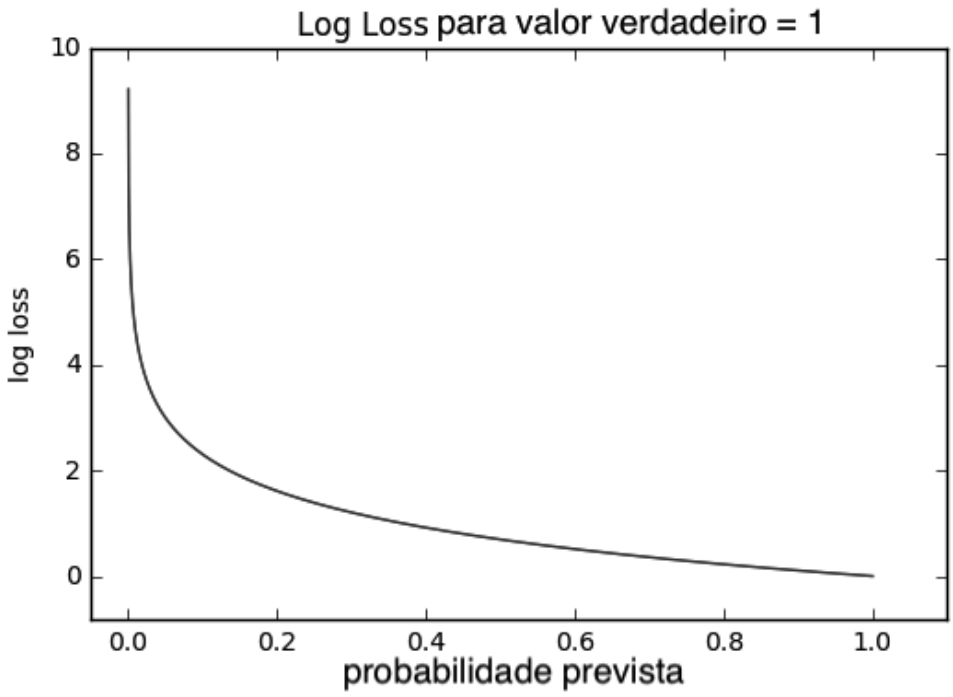
$$\begin{aligned} \text{Acurácia} &= \frac{\text{verdadeiros negativos} + \text{verdadeiros positivos}}{\text{total de imagens}} \\ &= \frac{954 + 982}{954 + 18 + 46 + 982} \\ &= 0.968 \end{aligned} \tag{3}$$

Possuir uma acurácia de 1 significa, então, ter classificado todas as imagens corretamente. O maior problema de utilizar apenas este indicador é que não se leva em conta a certeza de cada objeto classificado, ou seja, apesar de ter acertado todas as classes é possível que o sistema seja bastante instável e com baixa confiabilidade.

Para melhor mensurar a confiabilidade de classificação do sistema utiliza-se o indicador de perda. Toda rede neural profunda possui uma função de perda. No caso da rede em estudo

a função é chamada de *Log loss*, que indica e penaliza desvios na precisão da classificação de uma imagem. Essa função é utilizada durante a aprendizagem da rede, isto é, a cada iteração durante o treinamento será determinado um valor de penalização para desvios na certeza de classificação. Assim o algoritimo avalia os valores de pesos utilizados nas camadas da rede e faz modificações adequadas para a próxima iteração. Quanto menor a certeza da precisão, maior a penalização para o grupo de valores de pesos avaliados. Como mostra a Figura 44 quando mais longe o valor previsto estiver do valor real (1 neste caso) maior é a penalização gerada pela função de perda.

Figura 44 – Gráfico *Log Loss*



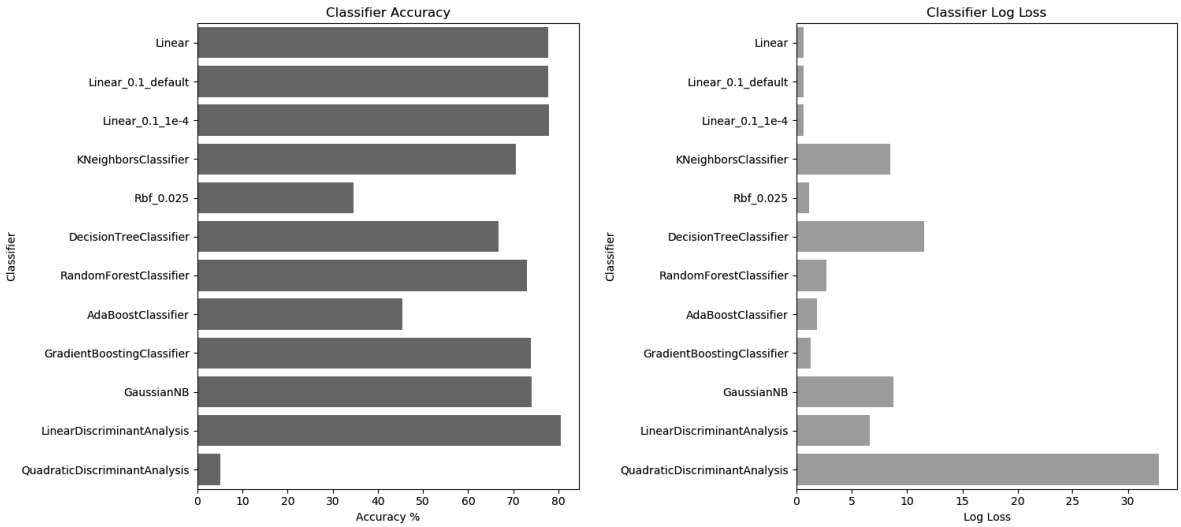

Fonte: Adaptado de: (WIKIFAST, 2017)

## 4.2 RESULTADOS

No trabalho apresentado existem dez categorias sendo avaliadas, assim cada classe possui valores de especificidade e sensitividade independentes, isso ocorre pois os valores de verdadeiro positivo, verdadeiro negativo, falso positivo e falso negativo são relativos a ser ou não de certa classe. No entanto o valor de acurácia é universal para o sistema.

Inicialmente realizou-se uma análise empírica dos desempenhos de dez tipos de *kernel* para o SVC, além de três variações do *kernel* linear. O teste foi feito apenas para o *data set* com 30 variantes, justificando a utilização do SVC linear padrão como classificador do modelo final. Como mostra a Figura 45 esse SVC obteve uma acurácia de 77.7% e um *log loss* de apenas 0.67, sendo o conjunto de resposta entre os classificadores analisados.

Figura 45 – Comparação dos Classificadores

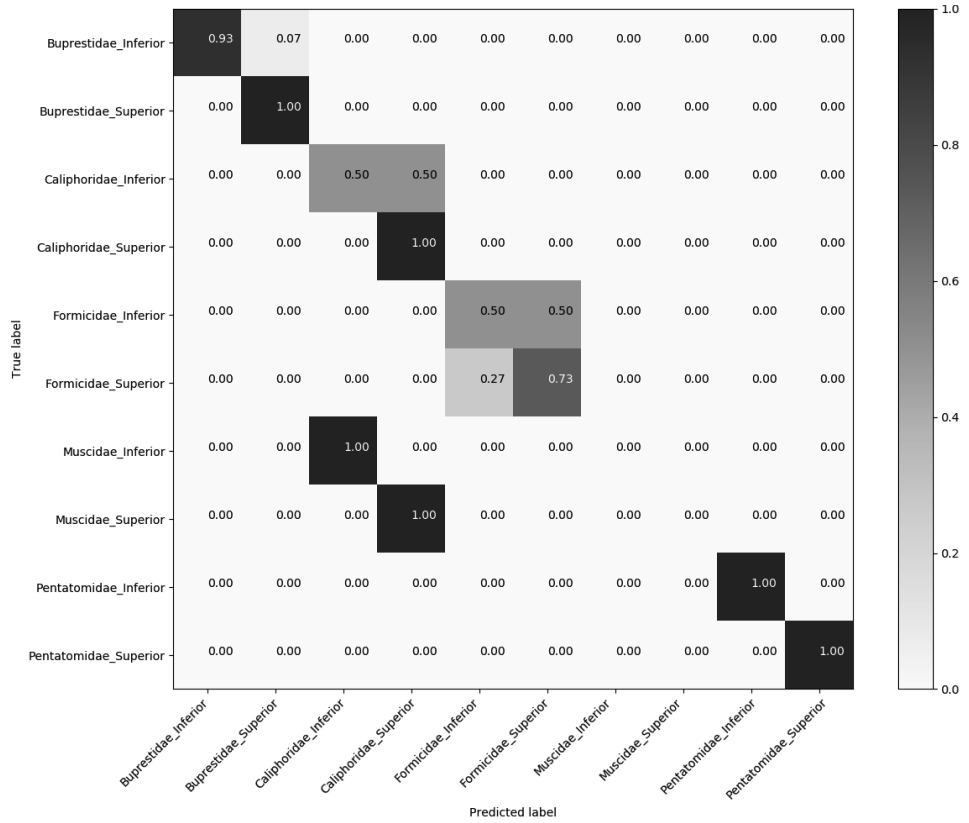



### 4.2.1 RESULTADOS DO TREINAMENTO

Para fins de comparação, criou-se três *data sets* com diferentes tamanhos de acordo com o número de variantes criadas através de *data augmentation*: cinco, dez e trinta variantes. Para todos os grupos são geradas a matriz confusão e gráfico de variáveis de desempenho, também é calculado o *log loss* e a acurácia do classificador.

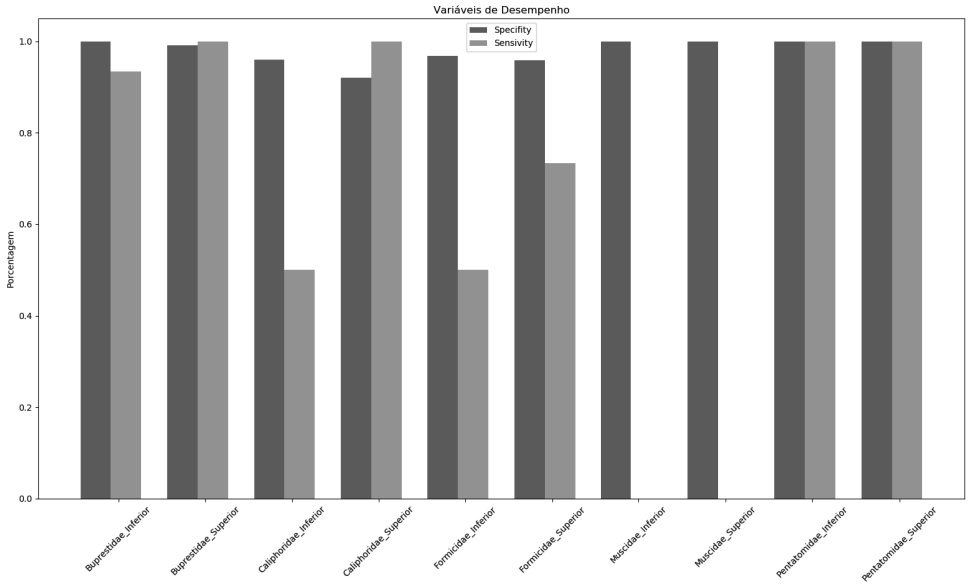
O primeiro grupo de imagens obteve os seguintes indicadores, das figuras 46 e 47:

Figura 46 – Matriz Confusão após treino com *Data Augmentation* de 5 variantes



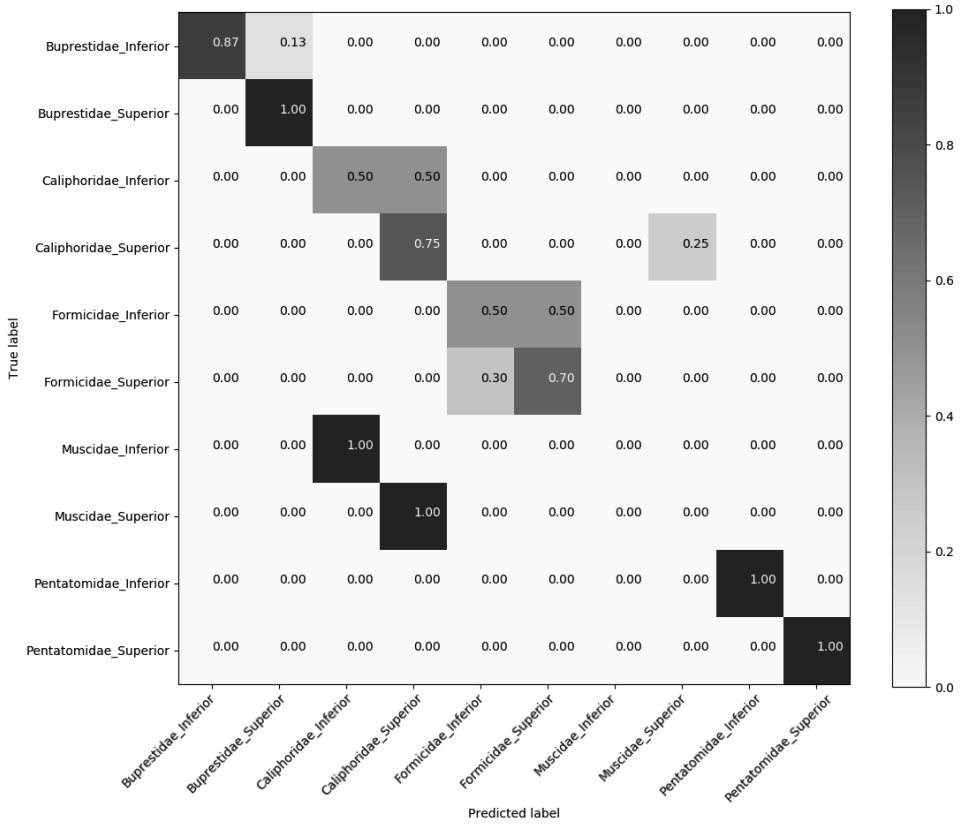

Os valores de especificidade e sensitividade são apresentados na Figura 47.

Figura 47 – Variáveis de Desempenho com *Data Augmentation* de 5 variantes



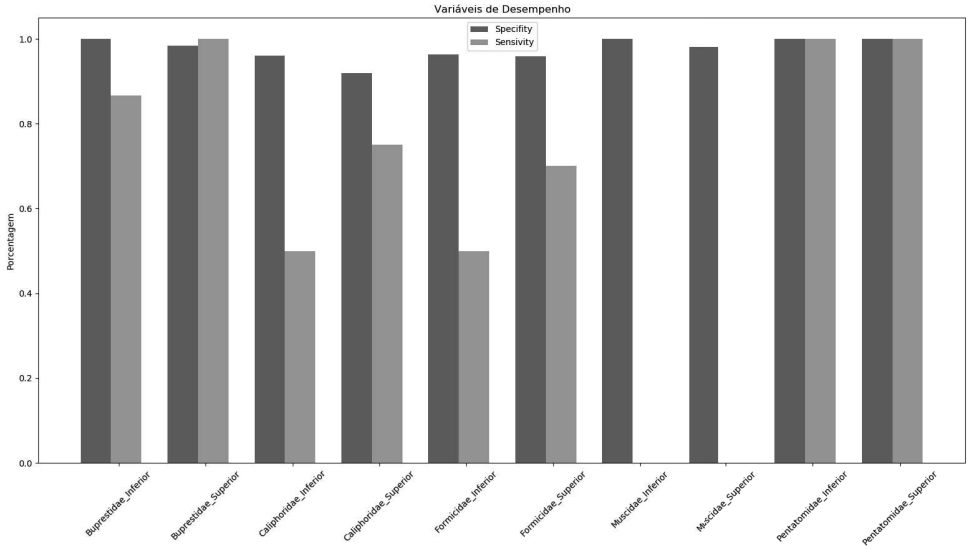

O grupo com *data augmentation* de 10 variantes obteve os indicadores das figuras 48 e 49:

Figura 48 – Matriz Confusão após treino com *Data Augmentation* de 10 variantes



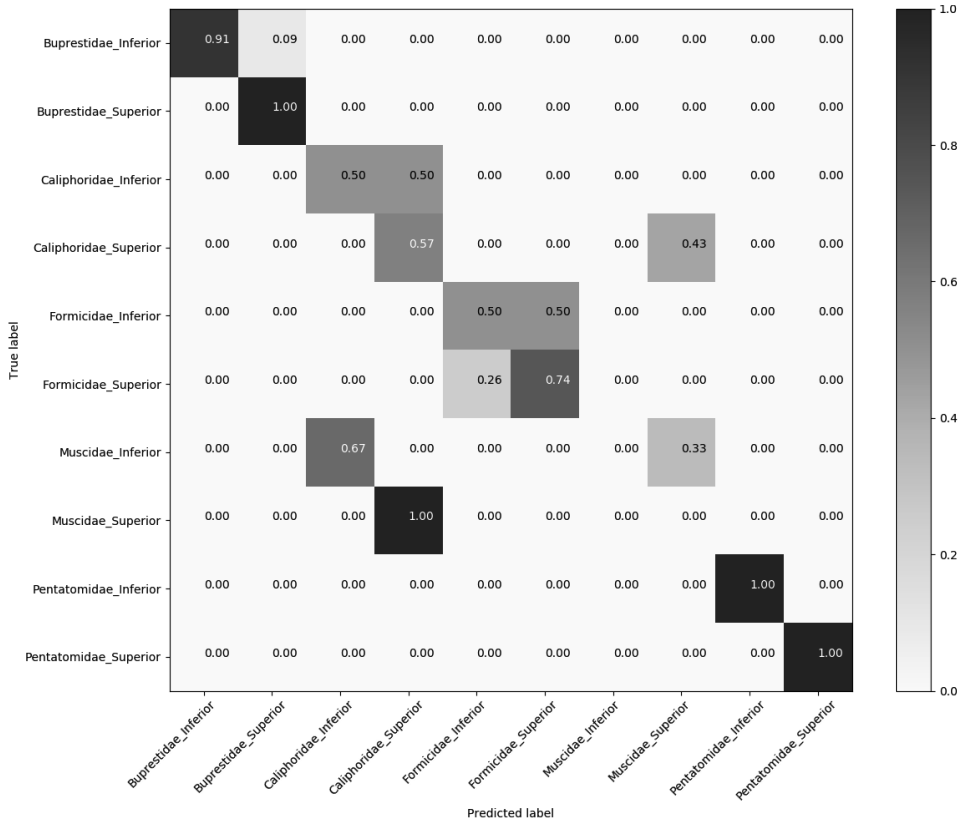

Os valores de especificidade e sensitividade são apresentados na Figura 49.

Figura 49 – Variáveis de Desempenho com *Data Augmentation* de 10 variantes



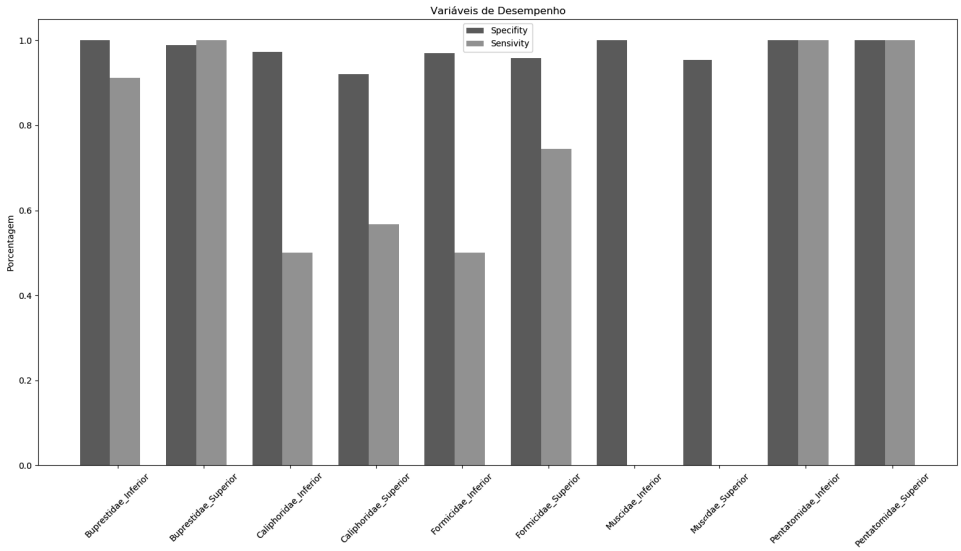

Os indicadores da última versão são apresentados nas figuras 50 e 51:

Figura 50 – Matriz Confusão após treino com *Data Augmentation* de 30 variantes




Os valores de especificidade e sensitividade são apresentados na Figura 51.

Figura 51 – Variáveis de Desempenho com *Data Augmentation* de 30 variantes



Os respectivos valores de acurácia e *log loss* para cada iteração são apresentados na Tabela 3

Tabela 3 – Acurácia Média e *Log Loss*.

| | Acurácia após treino | *Log Loss* |
|---|---|---|
| *Data Augmentation* 5 variantes | 78.52% | 0.57 |
| *Data Augmentation* 10 variantes | 78.52% | 0.66 |
| *Data Augmentation* 30 variantes | 77.77% | 0.67 |

### 4.2.2 RESULTADOS TESTES

Finalmente, os resultados obtidos com as imagens de teste, ou seja, totalmente desconhecidas pela rede, após treino com cada grupo de imagens, são mostrados abaixo nas figuras 52, 53, 54, 55, 56, 57, 58 e 59.

Figura 52 – Teste com Insetos da Família Buprestidae.

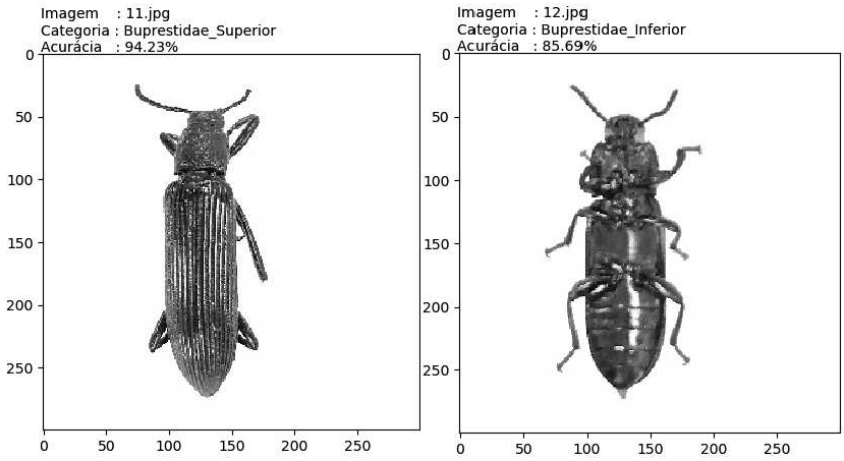



Figura 53 – Teste com Insetos da Família Formicidae.

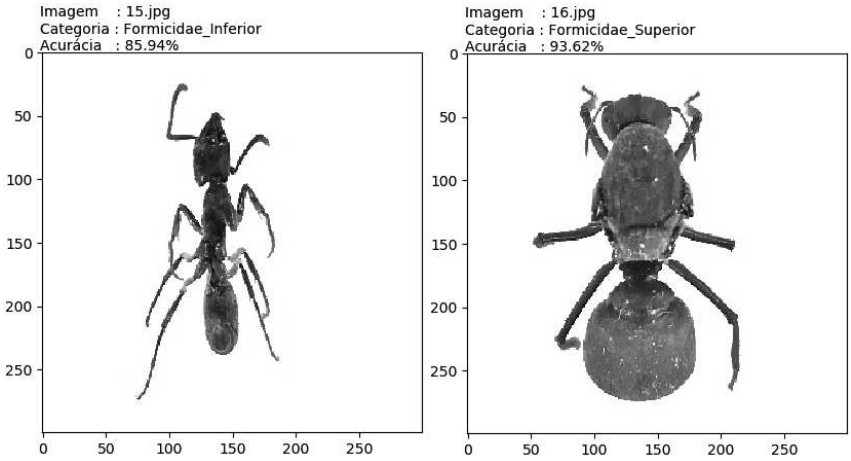


Figura 54 – Teste com Insetos da Família Muscidae.

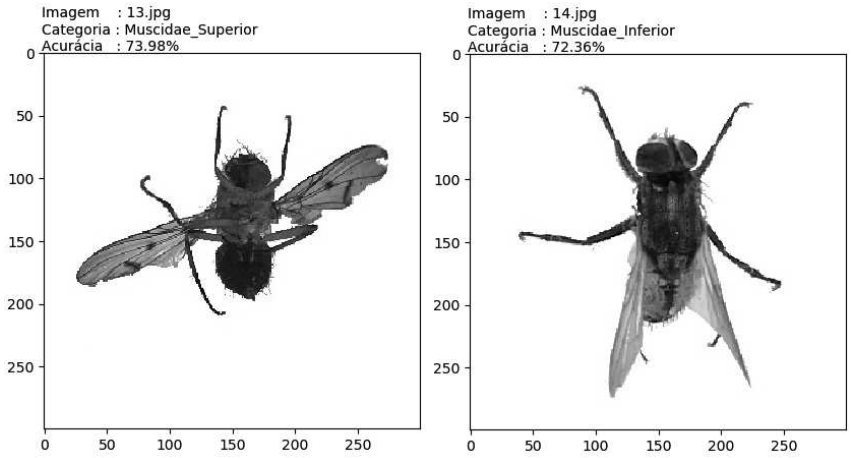



Figura 55 – Teste com Insetos da Família Calliphoridae.

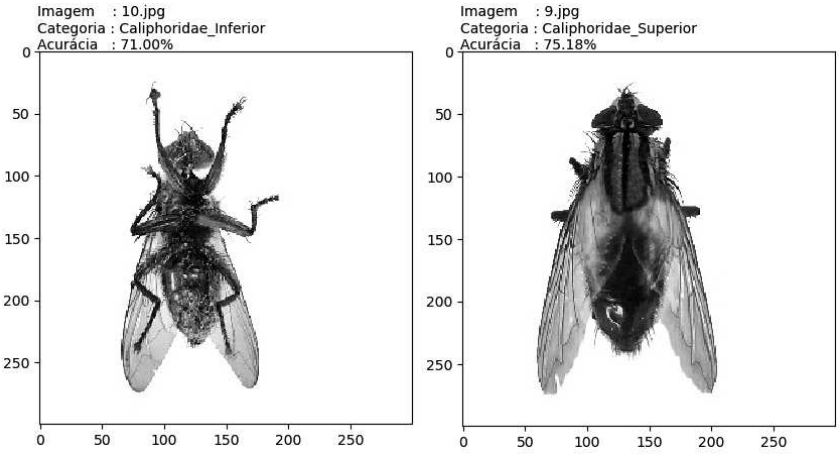



Figura 56 – Teste com Insetos da Família Pentatomidae.

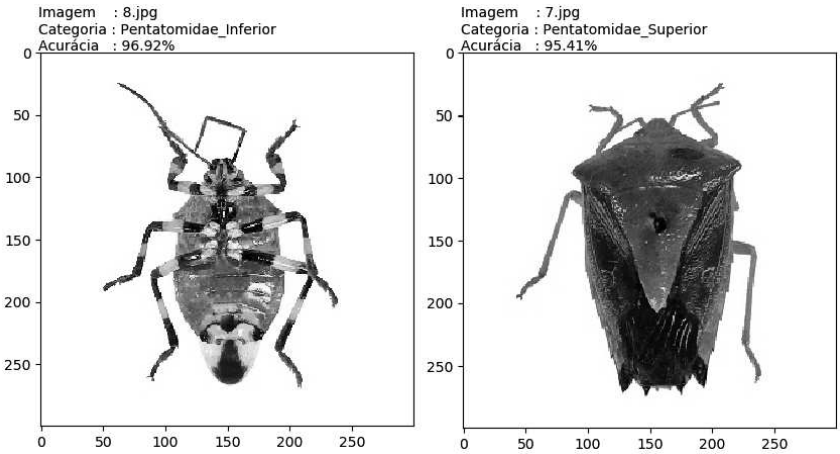


Figura 57 – Teste com Insetos da Família Pentatomidae Deteriorados - 1.

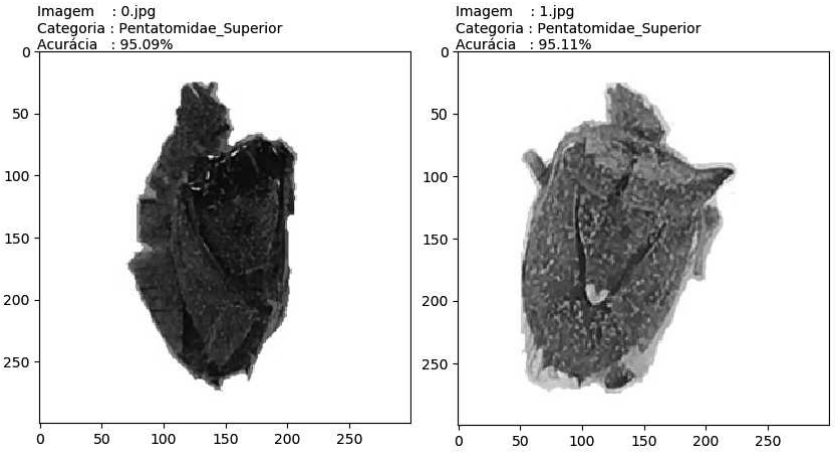



Figura 58 – Teste com Insetos da Família Pentatomidae Deteriorados - 2.

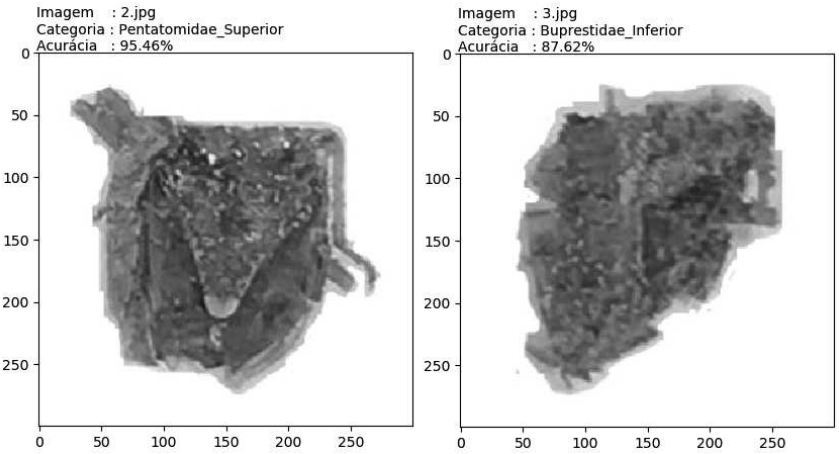
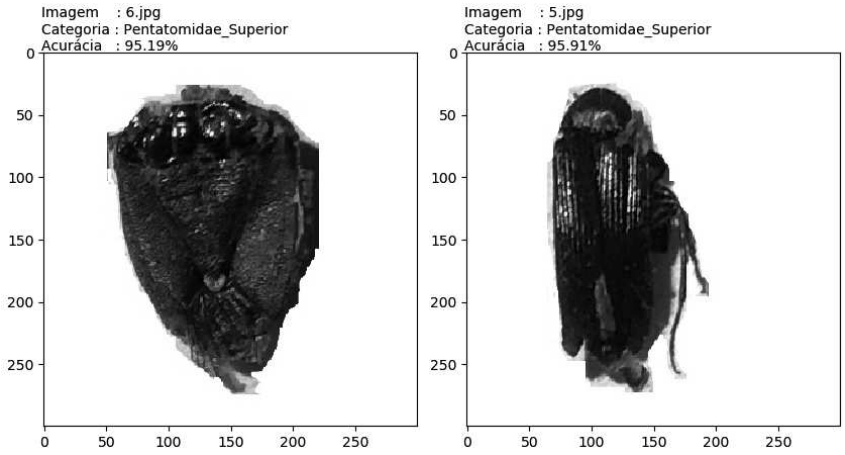



Figura 59 – Teste com Insetos Deteriorados.



## 4.3 DISCUSSÃO

Tomando os resultados obtidos durante os treinos pode-se fazer algumas análises. Observando-se as matrizes confusão, em todos os cenários de *data augmentation* experimentados, nota-se como o sistema reagiu às diferentes classes, por exemplo, percebe-se que classes muito similares, como a *Calliphoridae* e *Muscidae*, tendem a ser confundidas pela rede, causando o erro completo da segunda. É fato que o número extremamente reduzido do *data set* da classe *Muscidae*, principalmente, amplifica essa falha do sistema. Isso é corroborado pela diminuição dos erros nos casos onde utilizou-se mais variantes para o *data augmentation*.

A classe *Formicidae* também sofreu com a similaridade entre classes, no entanto este caso não é tão crítico, pois a família identificada é a mesma e apenas a vista é confundida. Assim podemos afirmar que o inseto é classificado corretamente em todos os casos. Por fim as categoria com maior diferença entre todas as outras, *Buprestidae* e *Pentatomidae*, obtiveram resultados perfeitos ou muito próximo disso. Não por coincidência são também as famílias com maior número de exemplares no banco de dados de treino.

Analisando-se a especificidade e sensitividade em todos os casos pode-se ver que a possibilidade de falsos positivos é quase nula (especificidade média $= 97.7\%$), já a taxa de verdadeiros positivos é bastante variada, sendo alta nas classes "mais exóticas" e reduzida em classes similares.

Uma análise bastante marcante é sobre a variação da acurácia e *Log loss* nos diferentes cenários. Inicialmente, tem-se como objetivo melhorar esses indicadores utilizando a técnica de *data augmentation*. Porém após os teste empíricos fica evidente que, neste caso, o aumento virtual do banco de dados piorou, ou no máximo manteve estável, os indicadores de desempenho. Pode-se teorizar que o motivo para esse comportamento, de *overfitting* da rede, é resultado da pouca variação real das imagens. Isto é, apenas transladar o objeto dentro do quadro não foi suficiente para diminuir o viés das características de cada classe. Assim, aumentar o número de variantes também aumenta o *overfitting*, ou seja, acarreta na piora de desempenho do sistema.

Para contornar essa situação pode-se, primeiramente, fazer testes com outras técnicas de *data augmentation*, como rotação, mudança de cores, entre outras, visando buscar que outros traços sejam capturados e evitar que as características específicas do *data set* sejam supervalorizadas. Ou de maneira trivial, é necessário aumentar o banco de dados com imagens originais e variadas, e não de forma virtual.

Contudo, é possível perceber que o modelo final da rede apresentou bons resultados tanto em absorver variações dentro de cada categoria, como em identificar pedaços de espécimes. A acurácia dos teste ficou em 75%, muito próximo à acurácia apresentada na validação do treinamento, de 77.77%, o que novamente corrobora a robustez do sistema.

Observando-se a acurácia das imagens com pedaços de insetos, quatro acima de 95% e duas erradas, é possível afirmar que a rede cumpre o seu principal objetivo, de identificar a família dos insetos apenas com parte deles. Mesmo no caso das imagens classificadas erradas (Figura 58 - Imagem 3 e Figura 59 - Imagem 5), é possível, também, discutir que mesmo um

especialista teria dificuldades em classificar os exemplares.

Em relação a Figura 59, observa-se também a importância da correta iluminação do objeto. O espécime menos deteriorado, à esquerda da figura, foi corretamente identificado como *Pentatomidae Superior*, entretanto o espécime à direita, também identificado como *Pentatomidae Superior*, é na realidade um *Buprestidae Superior*. Este erro de classificação pode ser atribuído ao fato de que o quadro possui diversas imperfeições relativas ao sombreamento da imagem, acarretando na incorreta atribuição de pesos pela rede treinada.

# 5 CONCLUSÃO

O trabalho desenvolvido e aqui apresentado visa trazer uma nova abordagem para a solução do problema de classificação de insetos para a entomologia forense. Em busca de facilitar o trabalho dos investigadores e peritos criminais, assim como melhorar e agilizar processos para que a justiça possa ser mais rápida e eficiente. Com a implementação desta abordagem de classificação de famílias entomológicas, espécimes encontrados em locais de crime e em carregamentos de entorpecentes podem ser direcionadas a profissionais especializados, evitando assim o desperdício de provas potencialmente importantes e acelerando o processo de investigações criminais.

Buscando na computação e em conceitos de *deep learning*, foi possível encontrar um sistema que agregasse ao trabalho de profissionais dentro da ciência forense. Aplicando-se conceito de redes neurais convolucionais, *transfer learning* e classificadores por vetores de suporte criou-se uma arquitetura que apresenta resultados bastante satisfatórios já em sua primeira versão. Os resultados finais obtidos neste trabalho, acurácia de 75% e *log loss* de apenas 0.67 deixam espaço para melhorias e aperfeiçoamento, no entanto já permitem a utilização da solução em casos reais. Vale ressaltar que estes resultados foram obtidos com um *dataset* muito reduzido, ou seja, os resultados tendem a ser cada vez melhores com a aquisição de novos exemplares.

No que diz respeito a técnica de *data augmentation*, pode-se notar que no contexto deste trabalho não se mostrou eficiente. Isso porque a partir de um certo momento o processo, que deveria impactar positivamente no treinamento, passou a ser um fator negativo na classificação de algumas classes. Ainda assim, para o trabalho apresentado, se fez necessário seu uso, devido ao pequeno tamanho do banco de imagens, de forma a diferenciar classes muito semelhantes.

Adicionalmente aos conhecimentos e soluções implementados é importante ressaltar que este trabalho é apenas o início de uma área que ainda pode ser expandida dentro do âmbito nacional, pois pode-se fazer uso de novas técnicas que são desenvolvidas constantemente. Diversos estudos dentro do assunto de *deep learning* e redes neurais continuam sendo realizados, bem como novos modelos matemáticos e estatísticos. Através desses os resultados podem ser otimizados, mesmo com um *dataset* reduzido, porém sendo sempre melhor com um grande número de exemplares.

## 5.1 TRABALHOS FUTUROS

Qualquer sistema e/ou processo digital possui potencial de melhoria, devido à rápida evolução de *softwares* e ferramentas digitais. No caso do trabalho apresentado existem diversas formas de melhorar o resultado final do sistema. A forma mais evidente de melhoria é o aumento

do banco de dados para treinamento da rede, sendo talvez a melhoria mais fácil de ser alcançada porém, também a mais demorada já que, idealmente, seriam coletados insetos em situações reais e não apenas de coleções de entomólogos. Ainda no que diz respeito ao banco de dados, o armazenamento dos *paths* das imagens poderia ser feito através de uma estrutura SQL ou semelhante, buscando melhor organizar o *dataset*.

Com o aumento do *dataset* e a utilização de uma estrutura de dados mais robusta, também seria possível associar a cada família de insetos o contato de um ou mais especialista da área de entomologia, de maneira a direcionar os espécimes encontrados em locais de crime mais rapidamente. Além disso, com esse aumento, o sistema poderia ser gradativamente alterado de forma a reconhecer não apenas famílias, mas também espécies de insetos.

Em relação ao processamento digital das imagens, esse poderia ser muito aperfeiçoado, fazendo uso de algoritimos que não se baseiem apenas nos níveis de cores do objeto e de fundo. Com a atual implementação, imperfeições localizadas em regiões que visualmente são brancas podem ser detectadas, interferindo na qualidade do treinamento da rede. Portanto, o ideal é fazer uso de técnicas que identifiquem as bordas do objeto de interesse. Ainda no âmbito do processamento da imagem, também poderiam ser implementados algoritmos que minimizem os sombreamentos da imagem, buscando destacar detalhes importantes do quadro.

Outro aperfeiçoamento possível, como citado anteriormente, é testar novas formas e técnicas de *data augmentation*, buscando evitar *overfitting* e melhorar as respostas do classificador. De maneira similar o experimento de diferentes classificadores é recomendado, neste trabalho foram considerados 12 classificadores, no entanto, todos eles possuem diferentes parâmetros que podem ser ajustados de forma a atingir melhores resultados.

Com a evolução das redes neurais convolucionais, cada vez mais profundas, atingindo melhores resultados, entende-se que outras redes neurais podem ser utilizadas futuramente para *transfer learning*, como por exemplo a nova versão da rede implementada neste trabalho, *Inception v4*. De certa forma até o banco de dados em que essa rede base é treinada pode interferir nos resultados finais.

Por fim o mais recomendado seria a criação de uma rede neural convolucional totalmente específica para essa aplicação. Especializando-se a arquitetura da rede para a solução e as características das imagens que são analisadas, é certamente a forma ideal de se atingir os melhores resultados. Obviamente para que isso seja possível é necessário ter-se um vasto banco de dados. Porém considerando que a cada momento mais imagens são coletadas e armazenadas, em um âmbito federal, pode-se alcançar um banco suficientemente grande.

# Referências


CALAZANS, C. H.; CALAZANS, S. M. **Ciência Forense: das Origens à Ciência Forense Computacional**. 2005. Disponível em: <http://www.truzzi.com.br/blog/wp-content/uploads/2010/07/Monografia_CienciaForense.pdf>. Acesso em: 29 de maio de 2019. Citado 2 vezes nas páginas 12 e 13.

CANZIANI, A.; CULURCIELLO, E. **An Analysis od Deep Neural Network Models for Practical Applications**. [S.l.], 2017. 7 p. Disponível em: <https://arxiv.org/pdf/1605.07678.pdf>. Acesso em: 02 de abril de 2019. Citado na página 23.

CATTS, E. P.; GOFF, M. L. Forensic entomology in criminal investigations. **Annual Review of Entomology**, 1992. Citado na página 13.

CHEMELLO, E. **Ciência Forense: Impressões Digitais**. 2006. Disponível em: <http://www.quimica.net/emiliano/artigos/2006dez_forense1.pdf>. Acesso em: 29 de maio de 2019. Citado na página 13.

CORTEZ, C.; VAPNIK, V. Support-vector networks. **Machine Learning**, p. 273–297, 1995. Citado na página 29.

DERTAT, A. **Applied Deep Learning - Part 4: Convolutional Neural Networks**. 2017. Disponível em: <https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2>. Acesso em: 01 de maio de 2019. Citado 2 vezes nas páginas 18 e 19.

GANDHI, R. **Support Vector Machine—Introduction to Machine Learning Algorithms**. 2018. Disponível em: <https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47>. Acesso em: 07 de maio de 2019. Citado na página 28.

GUNN, S. R. **Support Vector Machines for Classification and Regression**. [S.l.], 1998. 53 p. Disponível em: <http://ce.sharif.ir/courses/85-86/2/ce725/resources/root/LECTURES/SVM.pdf>. Acesso em: 18 de abril de 2019. Citado 4 vezes nas páginas 27, 28, 30 e 31.

IMAGENET. **Results of ILSVRC2014**. [S.l.], 2014. 1 p. Disponível em: <http://www.image-net.org/challenges/LSVRC/2014/results>. Acesso em: 10 de abril de 2019. Citado na página 22.

KURENKOV, A. **A 'Brief' History of Neural Nets and Deep Learning**. 2015. Disponível em: <https://www.andreykurenkov.com/writing/ai/a-brief-history-of-neural-nets-and-deep-learning/>. Acesso em: 30 de maio de 2019. Citado na página 16.

LEITE, G. L. D.; Sá, V. G. M. de. **Apostila: Taxonomia, Nomenclatura e Identificação de Espécies**. 1. ed. [S.l.: s.n.], 2010. Citado 3 vezes nas páginas 13, 14 e 15.

LI JUSTIN JOHNSON, S. Y. F.-F. **CS231n: Convolutional Neural Networks for Visual Recognition**. 2018. Disponível em: <http://cs231n.github.io/convolutional-networks/>. Acesso em: 01 de maio de 2019. Citado 5 vezes nas páginas 16, 17, 21, 22 e 42.

LIU, D. **A Practical Guide to ReLU**. 2017. Disponível em: <https://medium.com/tinymind/a-practical-guide-to-relu-b83ca804f1f7>. Acesso em: 30 de maio de 2019. Citado na página 20.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. **The bulletin of mathematical biophysics**, v. 5, n. 4, p. 115–133, Dec 1943. ISSN 1522-9602. Disponível em: <https://doi.org/10.1007/BF02478259>. Citado na página 16.

Müller, K.-R. An introduction to kernel-based learning algorithms. **IEEE Transactions on Neural Network**, v. 12, p. 181–202, 2001. Citado na página 29.

NABI, J. **Machine Learning—Text Classification, Language Modelling using fast.ai**. 2019. Disponível em: <https://towardsdatascience.com/machine-learning-text-classification-language-modelling-using-fast-ai-b1b334f2872d>. Acesso em: 18 de maio de 2019. Citado na página 46.

PATEL, S. **SVM (Support Vector Machine)—Theory**. 2017. Disponível em: <https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72>. Acesso em: 07 de maio de 2019. Citado 3 vezes nas páginas 28, 29 e 31.

PAUL, S. **Towards Preventing Overfitting: Regularization**. 2018. Disponível em: <https://www.datacamp.com/community/tutorials/towards-preventing-overfitting-regularization>. Acesso em: 18 de maio de 2019. Citado na página 39.

PUJOL-LUZ, J. R.; ARANTES, L. C.; CONSTANTINO, R. Cem anos da entomologia forense no brasil(1908-2008). **Revista Brasileira de Entomologia**, 2008. Citado 2 vezes nas páginas 12 e 14.

SAHA, S. **A Comprehensive Guide to Convolutional Neural Networks**. 2018. Disponível em: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>. Acesso em: 01 de maio de 2019. Citado 2 vezes nas páginas 21 e 22.

SCIKITLEARN. **Cross-validation: evaluating estimator performance**. 2018. Disponível em: <https://scikit-learn.org/stable/modules/cross_validation.html>. Acesso em: 14 de maio de 2019. Citado na página 45.

SCIKITLEARN. **SVC**. 2018. Disponível em: <https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC>. Acesso em: 08 de maio de 2019. Citado na página 43.

SCIKITLEARN. **SVM Margins Example**. 2018. Disponível em: <https://scikit-learn.org/stable/auto_examples/svm/plot_svm_margin.html>. Acesso em: 08 de maio de 2019. Citado na página 30.

SHAH, T. **Train, Validation and Test Sets**. 2017. Disponível em: <https://tarangshah.com/blog/2017-12-03/train-validation-and-test-sets/>. Acesso em: 14 de maio de 2019. Citado na página 44.

SRIVASTAVA, N. et al. A simple way to prevent neural networks from overfitting. **Journal of Machine Learning Research**, v. 15, p. 1929–1958, Jun 2014. Disponível em: <http://www.jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf?utm_content=

buffer79b43&utm_medium=social&utm_source=twitter.com&utm_campaign=buffer>. Citado na página 39.

SZEGEDY, C. et al. Going deeper with convolutions. In: **Computer Vision and Pattern Recognition (CVPR)**. [s.n.], 2015. Disponível em: <http://arxiv.org/abs/1409.4842>. Citado na página 22.

SZEGEDY, C. et al. Rethinking the inception architecture for computer vision. In: **Proceedings of IEEE Conference on Computer Vision and Pattern Recognition,**. [s.n.], 2016. Disponível em: <http://arxiv.org/abs/1512.00567>. Citado 6 vezes nas páginas 22, 24, 25, 26, 27 e 32.

TORREY, L.; SHAVLIK, J. **Handbook of Research on Machine Learning Applications**. 1. ed. [S.l.]: IGI Global, 2009. Citado 2 vezes nas páginas 41 e 42.

VAPNIK, V. **The Nature of Statistical Learning Theory**. 2. ed. [S.l.]: Springer, 1995. Citado na página 31.

WANG, J.; PEREZ, L. The effectiveness of data augmentation in image classification using deep learning. Dec 2017. Disponível em: <https://arxiv.org/abs/1712.04621>. Citado na página 39.

WIKIFAST. **Log Loss**. 2017. Disponível em: <http://wiki.fast.ai/index.php/Log_Loss>. Acesso em: 20 de maio de 2019. Citado na página 48.

ZIęBA-PALUS, J. Forensic examinations of micro-traces. **Forensic Science and Criminology**, v. 2, 2017. Citado na página 12.